

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC
Petitioner

v.

K.MIZRA LLC
Patent Owner

Case No. IPR2026-00304
Patent 8,438,120

DECLARATION OF JUNE ANN MUNFORD

EXHIBIT LIST

Exhibit No.	Description
1007	Pieter-Tjerk De de Boer <i>et al.</i> , “A tutorial on the Cross-Entropy Method,” ANNALS OF OPERATIONS RESEARCH 134:19-67 (2005) (“de Boer”)
1008	M. Kirsten <i>et al.</i> , “Distance Based Approaches to Relational Learning and Clustering,” in Relational Data Mining, Chpt. 9, pp. 213-232 (2001) (“Kirsten”)
1010	S. Mannor, “The Cross Entropy Method For Classification,” pp. 561-568 (2005) (“Mannor”)
1011	F. Friedrichs, <i>et al.</i> , “Evolutionary tuning of multiple SVM parameters,” pp107-117 (January 7, 2005) (“Friedrichs”)
1014	T. Horváth <i>et al.</i> , “Relational Instance-Based Learning with Lists and Terms,” MACHINE LEARNING 43:53-80 (2001) (“Horvath”)
1016	Caruana, Rich, Steve Lawrence, and C. Giles (2000). "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping." Advances in neural information processing systems 13 (“Caruana”).
1017	Xu, Rui, and Donald Wunsch. "Survey of clustering algorithms." IEEE Transactions on neural networks 16, no. 3 (2005): 645-678. (“Rui”).
1021	<i>Curriculum Vitae</i> of June Munford

I, June Ann Munford, hereby state as follows:

1. I am over the age of 18, have personal knowledge of the facts set forth herein, and am competent to testify to the same.

2. I earned a Master of Library and Information Science (MLIS) from the University of Wisconsin-Milwaukee in 2009. I have over ten years of experience in the library/information science field. Beginning in 2004, I have served in various positions in the public library sector including Assistant Librarian, Youth Services Librarian and Library Director. I have attached my Curriculum Vitae as Appendix CV and Exhibit 1021.

3. During my career in the library profession, I have been responsible for materials acquisition for multiple libraries. In that position, I have cataloged, purchased and processed incoming library works. That includes purchasing materials directly from vendors, recording publishing data from the material in question, creating detailed material records for library catalogs and physically preparing that material for circulation. In addition to my experience in acquisitions, I was also responsible for analyzing large collections of library materials, tailoring library records for optimal catalog search performance and creating lending agreements between libraries during my time as a Library Director.

4. I am fully familiar with the catalog record creation process in the library sector. In preparing a material for public availability, a library catalog record

describing that material would be created. These records are typically written in Machine Readable Catalog (herein referred to as “MARC”) code and contain information such as a physical description of the material, metadata from the material’s publisher, and date of library acquisition. In particular, the 008 field of the MARC record is reserved for denoting the date of creation of the library record itself. As this typically occurs during the process of preparing materials for public access, it is my experience that an item’s MARC record indicates the date of an item’s public availability.

5. Typically, in creating a MARC record, a librarian would gather various bits of metadata such as book title, publisher and subject headings among others and assign each value to a relevant numerical field. For example, a book’s physical description is tracked in field 300 while title/attribution is tracked in field 245. The 008 field of the MARC record is reserved for denoting the creation of the library record itself. As this is the only date reflecting the inclusion of said materials within the library’s collection, it is my experience that an item’s 008 field accurately indicates the date of an item’s public availability.

6. I am familiar with the Internet Archive, a digital library accessible to the general public. Among other services that the Internet Archive makes available to the general public is the Wayback Machine, an online archive of the web. The Internet Archive’s Wayback Machine service archives webpages as of a certain

capture date to track changes in the web over time. The Internet Archive has been in operation as a nonprofit library since 1996 and has hosted the Wayback Machine service since its inception in 2001. During my time as a librarian, I frequently used the Internet Archive's Wayback Machine for research and instruction purposes. This includes teaching instructional classes on using the Wayback Machine to library patrons and using the Wayback Machine to research reference inquiries that require hard-to-find online resources. I consider the Internet Archive's recordskeeping to be as rigorous and detailed as other formal library recordskeeping practices such as MARC records, OCLC records and Dublin Core.

7. I have been informed that for purposes of this proceeding, a person of ordinary skill in the art ("POSITA") would have had at least a Master's degree in electrical engineering, computer science, or a related discipline, knowledge of machine learning classification techniques as of this time, and at least two years of experience working in a field involving classifier development in machine learning systems. A person with a different degree but with additional relevant experience could still qualify if the additional experience compensates for the different educational background. A person with less than two years of experience working in a field involving networking and system security could still qualify if the person has additional education that compensates for their lesser experience.

8. As explained below, each of Exhibits 1007, 1008, 1010, 1011, 1014, 1016, and 1017 was accessible to the public (including a person of ordinary skill in the art) prior to April 25, 2007, which I understand is the earliest claimed priority date of U.S. Patent No. 8,438,120.

9. I have reviewed Exhibit 1007, "A Tutorial on the Cross-Entropy Method" by Pieter-Tjerk De Boer, et al. as published in Annals of Operations Research February 2005.

10. Attached hereto as Appendix DEBOER01 is a true and correct copy of the MARC record for Annals of Operations Research as held by the Penn State University Library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix DEBOER01 accurately describes the title, publisher and ISSN number of Annals of Operations Research. The Showing Results field present on page 3 of this record indicates the library's collection includes the February 2005 issue of Annals of Operations Research containing "A Tutorial on the Cross-Entropy Method".

11. Attached hereto as Appendix DEBOER02 is a true and correct copy of "A Tutorial on the Cross-Entropy Method" as published in Annals of Operations Research as held by the Penn State University Library. These scans were secured from the library's collection. In comparing Exhibit 1007 to Appendix DEBOER02, it is my determination that Exhibit 1007 is a true and correct copy of "A Tutorial on

the Cross-Entropy Method" by Pieter-Tjerk De Boer, et al. as published in Annals of Operations Research.

12. The 008 field of the MARC record in Appendix DEBOER01 indicates the date of record creation. The 008 field of Appendix DEBOER01 indicates the Penn State University Library first acquired Annals of Operations Research as of December 6, 1999 and held the journal in perpetuity until the publication of "A Tutorial on the Cross-Entropy Method". Considering this information, it is my determination that Annals of Operations Research February 2005 and therefore "A Tutorial on the Cross-Entropy Method" (Exhibit 1007) was publicly accessible and made available to the public shortly after its initial publication in February 2005.

13. I have reviewed Exhibit 1008, "Distance Based Approaches to Relational Learning and Clustering" by Mathias Kirsten, et al. as published in Relational Data Mining.

14. Attached hereto as Appendix DZEROSKI01 is a true and correct copy of the MARC record for Relational Data Mining as held by the Linda Hall Library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix DZEROSKI01 accurately describes the title, publisher and ISBN number of Relational Data Mining.

15. Attached hereto as Appendix DZEROSKI02 is a true and correct copy of "Distance Based Approaches to Relational Learning and Clustering" as published

in Relational Data Mining as held by the Linda Hall Library. These scans were secured from the library's collection. In comparing Exhibit 1008 to Appendix DZEROSKI02, it is my determination that Exhibit 1008 is a true and correct copy of "Distance Based Approaches to Relational Learning and Clustering" by Mathias Kirsten, et al. as published in Relational Data Mining.

16. The 008 field of the MARC record in Appendix DZEROSKI01 indicates the date of record creation. The 008 field of Appendix DZEROSKI01 indicates the Linda Hall Library first acquired Relational Data Mining as of July 30, 2001. Considering this information, it is my determination that Relational Data Mining and therefore "Distance Based Approaches to Relational Learning and Clustering" (Exhibit 1008) was publicly accessible and made available to the public shortly after July 30, 2001.

17. I have reviewed Exhibit 1010, "The Cross Entropy Method For Classification" by S. Mannor, et al as published in Proceedings of the 22nd International Conference on Machine Learning.

18. Attached hereto as Appendix MANNOR01 is a true and correct copy of the MARC record for Proceedings of the 22nd International Conference on Machine Learning as held by the Carnegie Mellon University Library. I secured this record myself from the library's public catalog. The MARC record contained within

Appendix MANNOR01 accurately describes the title, publisher and ISBN number of Proceedings of the 22nd International Conference on Machine Learning.

19. Attached hereto as Appendix MANNOR02 is a true and correct copy of “The Cross Entropy Method For Classification” as published in Proceedings of the 22nd International Conference on Machine Learning as held by the Carnegie Mellon University Library. These scans were secured from the library’s collection. In comparing Exhibit 1010 to Appendix MANNOR02, it is my determination that Exhibit 1010 is a true and correct copy of “The Cross Entropy Method For Classification” by S. Mannor, et al as published in Proceedings of the 22nd International Conference on Machine Learning.

20. The 008 field of the MARC record in Appendix MANNOR01 indicates the date of record creation. The 008 field of Appendix MANNOR01 indicates the Carnegie Mellon University Library first acquired Proceedings of the 22nd International Conference on Machine Learning as of September 22, 2005. Considering this information, it is my determination that Proceedings of the 22nd International Conference on Machine Learning and therefore “The Cross Entropy Method For Classification” (Exhibit 1010) was publicly accessible and made available to the public shortly after September 22, 2005.

21. I have reviewed Exhibit 1011, “Evolutionary Tuning of Multiple SVM Parameters” by F. Friedrichs, et al as published in Neurocomputing March 2005.

22. Attached hereto as Appendix FRIEDRICH01 is a true and correct copy of the MARC record for Neurocomputing as held by the Penn State University Library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix FRIEDRICH01 accurately describes the title, publisher and ISSN number of Neurocomputing. The Showing Results field present on page 3 of this record indicates the library's collection includes the March 2005 issue of Neurocomputing containing "Evolutionary Tuning of Multiple SVM Parameters".

23. Attached hereto as Appendix FRIEDRICH02 is a true and correct copy of "Evolutionary Tuning of Multiple SVM Parameters" as published in Neurocomputing as held by the Penn State University Library. These scans were secured from the library's collection. In comparing Exhibit 1011 to Appendix FRIEDRICH02, it is my determination that Exhibit 1011 is a true and correct copy of "Evolutionary Tuning of Multiple SVM Parameters" by F. Friedrichs, et al as published in Neurocomputing.

24. The 008 field of the MARC record in Appendix FRIEDRICH01 indicates the date of record creation. The 008 field of Appendix FRIEDRICH01 indicates the Penn State University Library first acquired Neurocomputing as of May 11, 1998 and held the journal in perpetuity until the publication of "Evolutionary Tuning of Multiple SVM Parameters". Considering this information, it is my

determination that Neurocomputing March 2005 and therefore “Evolutionary Tuning of Multiple SVM Parameters” (Exhibit 1011) was publicly accessible and made available to the public shortly after its initial publication in March 2005.

25. I have reviewed Exhibit 1014, "Relational Instance-Based Learning with Lists and Terms" by Tamas Horvath, et al as published in Machine Learning April 2001.

26. Attached hereto as Appendix HORVATH01 is a true and correct copy of the MARC record for Machine Learning as held by the Penn State University Library. I secured this record myself from the library’s public catalog. The MARC record contained within Appendix HORVATH01 accurately describes the title, publisher and ISSN number of Machine Learning. The Showing Results field present on page 3 of this record indicates the library's collection includes the April 2001 issue of Machine Learning containing "Relational Instance-Based Learning with Lists and Terms".

27. Attached hereto as Appendix HORVATH02 is a true and correct copy of "Relational Instance-Based Learning with Lists and Terms" as published in Machine Learning as held by the Penn State University Library. This PDF was secured from the library’s collection. In comparing Exhibit 1014 to Appendix HORVATH02, it is my determination that Exhibit 1014 is a true and correct copy of

"Relational Instance-Based Learning with Lists and Terms" by Tamas Horvath, et al as published in Machine Learning.

28. The 008 field of the MARC record in Appendix HORVATH01 indicates the date of record creation. The 008 field of Appendix HORVATH01 indicates the Penn State University Library first acquired Machine Learning as of January 26, 1998 and held the journal in perpetuity until the publication of "Relational Instance-Based Learning with Lists and Terms". Considering this information, it is my determination that Machine Learning April 2001 and therefore "Relational Instance-Based Learning with Lists and Terms" (Exhibit 1014) was publicly accessible and made available to the public shortly after its initial publication in April 2001.

29. I have reviewed Exhibit 1016, "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping" by Rich Caruana, et al as published in Proceedings of the International Conference on Neural Information Processing Systems July 2001.

30. Attached hereto as Appendix CARUANA01 is a true and correct copy of the MARC record for Proceedings of the International Conference on Neural Information Processing Systems as held by the Penn State University Library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix CARUANA01 accurately describes the title, publisher

and ISSN number of Proceedings of the International Conference on Neural Information Processing Systems. The Availability field present on page 3 of this record indicates the library's collection includes the July 2001 issue of Proceedings of the International Conference on Neural Information Processing Systems containing "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping".

31. Attached hereto as Appendix CARUANA02 is a true and correct copy of "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping" as published in Proceedings of the International Conference on Neural Information Processing Systems as held by the Penn State University Library. These scans were secured from the library's collection. In comparing Exhibit 1016 to Appendix CARUANA02, it is my determination that Exhibit 1016 is a true and correct copy of "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping" by Rich Caruana, et al as published in Proceedings of the International Conference on Neural Information Processing Systems.

32. The 008 field of the MARC record in Appendix CARUANA01 indicates the date of record creation. The 008 field of Appendix CARUANA01 indicates the Penn State University Library first acquired Proceedings of the International Conference on Neural Information Processing Systems as of September 4, 1998 and held the proceedings in perpetuity until the publication of

“Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping”. Considering this information, it is my determination that Proceedings of the International Conference on Neural Information Processing Systems July 2001 and therefore “Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping” (Exhibit 1016) was publicly accessible and made available to the public shortly after its initial publication in July 2001.

33. I have reviewed Exhibit 1017, "Survey of Clustering Algorithms" by Rui Xu, et al as published in IEEE Transactions on Neural Networks May 2005.

34. Attached hereto as Appendix XU01 is a true and correct copy of the MARC record for IEEE Transactions on Neural Networks as held by the Penn State University Library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix XU01 accurately describes the title, publisher and ISSN number of IEEE Transactions on Neural Networks. The Showing Results field present on page 3 of this record indicates the library's collection includes the May 2005 issue of IEEE Transactions on Neural Networks containing "Survey of Clustering Algorithms".

35. Attached hereto as Appendix XU02 is a true and correct copy of "Survey of Clustering Algorithms" as published in IEEE Transactions on Neural Networks as held by the Penn State University Library. These scans were secured from the library's collection. In comparing Exhibit 1017 to Appendix XU02, it is

my determination that Exhibit 1017 is a true and correct copy of "Survey of Clustering Algorithms" by Rui Xu, et al as published in IEEE Transactions on Neural Networks.

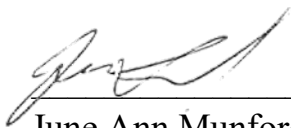
36. The 008 field of the MARC record in Appendix XU01 indicates the date of record creation. The 008 field of Appendix XU01 indicates the Penn State University Library first acquired IEEE Transactions on Neural Networks as of July 13, 2000 and held the journal in perpetuity until the publication of "Survey of Clustering Algorithms". Considering this information, it is my determination that IEEE Transactions on Neural Networks May 2005 and therefore "Survey of Clustering Algorithms" (Exhibit 1017) was publicly accessible and made available to the public shortly after its initial publication in May 2005.

37. I have been retained on behalf of the Petitioner to provide assistance in the above-illustrated matter in establishing the authenticity and public availability of the documents discussed in this declaration. I am being compensated for my services in this matter at the rate of \$200.00 per hour plus reasonable expenses. My statements are objective, and my compensation does not depend on the outcome of this matter.

38. In signing this Declaration, I understand that the Declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I acknowledge that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

39. I hereby declare under penalty of perjury that the foregoing is true and correct. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: March 19, 2026



June Ann Munford

June A. Munford
Curriculum Vitae

Education

University of Wisconsin-Milwaukee - MS, Library & Information Science, 2009
Milwaukee, WI

- Coursework included cataloging, metadata, data analysis, library systems, management strategies and collection development.
- Specialized in library advocacy, cataloging and public administration.

Grand Valley State University - BA, English Language & Literature, 2008
Allendale, MI

- Coursework included linguistics, documentation and literary analysis.
- Minor in political science with a focus in local-level economics and government.

Professional Experience

Researcher / Expert Witness, October 2017 – present
Freelance ● Pittsburgh, Pennsylvania & Grand Rapids, Michigan

- Material authentication and public accessibility determination. Declarations of authenticity and/or public accessibility provided upon research completion. Experienced with appeals and deposition process.
- Research provided on topics of public library operations, material publication history, digital database services and legacy web resources.
- Past clients include Alston & Bird, Arnold & Porter, Baker Botts, Fish & Richardson, Erise IP, Irell & Manella, O'Melveny & Myers, Perkins-Coie, Pillsbury Winthrop Shaw Pittman and Slayden Grubert Beard.

Library Director, February 2013 - March 2015
Dowagiac District Library ● Dowagiac, Michigan

- Executive administrator of the Dowagiac District Library. Located in Southwest Michigan, this library has a service area of 13,000, an annual operating budget of over \$400,000 and total assets of approximately \$1,300,000.

- Developed careful budgeting guidelines to produce a 15% surplus during the 2013-2014 & 2014-2015 fiscal years while being audited.
- Using this budget surplus, oversaw significant library investments including the purchase of property for a future building site, demolition of existing buildings and building renovation projects on the current facility.
- Led the organization and digitization of the library's archival records.
- Served as the public representative for the library, developing business relationships with local school, museum and tribal government entities.
- Developed an objective-based analysis system for measuring library services - including a full collection analysis of the library's 50,000+ circulating items and their records.

November 2010 - January 2013

Librarian & Branch Manager, Anchorage Public Library ● Anchorage, Alaska

- Headed the 2013 Anchorage Reads community reading campaign including event planning, staging public performances and creating marketing materials for mass distribution.
- Co-led the social media department of the library's marketing team, drafting social media guidelines, creating original content and instituting long-term planning via content calendars.
- Developed business relationships with The Boys & Girls Club, Anchorage School District and the US Army to establish summer reading programs for children.

June 2004 - September 2005, September 2006 - October 2013

Library Assistant, Hart Area Public Library
Hart, MI

- Responsible for verifying imported MARC records and original MARC cataloging for the local-level collection as well as the Michigan Electronic Library.
- Handled OCLC Worldcat interlibrary loan requests & fulfillment via ongoing communication with lending libraries.

Professional Involvement

Alaska Library Association - Anchorage Chapter

- Treasurer, 2012

Library Of Michigan

- Level VII Certification, 2008
- Level II Certification, 2013

Michigan Library Association Annual Conference 2014

- New Directors Conference Panel Member

Southwest Michigan Library Cooperative

- Represented the Dowagiac District Library, 2013-2015

Professional Development

Library Of Michigan Beginning Workshop, May 2008

Petoskey, MI

- Received training in cataloging, local history, collection management, children's literacy and reference service.

Public Library Association Intensive Library Management Training, October 2011

Nashville, TN

- Attended a five-day workshop focused on strategic planning, staff management, statistical analysis, collections and cataloging theory.

Alaska Library Association Annual Conference 2012 - Fairbanks, February 2012

Fairbanks, AK

- Attended seminars on EBSCO advanced search methods, budgeting, cataloging, database usage and marketing.

Depositions

2019 ● Fish & Richardson

Apple v. Qualcomm (Case No. IPR2018-001281, 39521-00421IP, IPR2018-01282 and 39521-00421IP2)

2019 ● Erise IP

Implicit, LLC v. Netscout Systems, Inc (Case No. 2:18-cv-53-JRG)

2019 ● Perkins-Coie

Adobe Inc. v. RAH Color Technologies LLC (Case No. IPR2019-00627, IPR2019-00628, IPR2019-00629 and IPR2019-00646)

2020 ● O'Melveny & Myers

Maxell, Ltd. v. Apple Inc. (Case No. 5:19-cv-00036-RWS)

2021 ● Pillsbury Winthrop Shaw Pittman LLP

Intel v. SRC (Case No. IPR2020-1449)

2022 ● Perkins-Coie

Realtek v. Future Link (Case No. IPR2021-01182)

2023 ● Fish & Richardson

Neuroderm Ltd. v. Abbvie, Inc (Case No. PGR2022-00040)

2023 ● Fish & Richardson

Nearmap US Inc. v. Pictometry International Corp. (Case No. IPR2022-00735)

2023 ● Fish & Richardson

Samsung Electronics v. MemoryWeb LLC (Case No. 39843-0136PS1)

2023 ● Pillsbury Winthrop Shaw Whitman LLP

Gravel Rating Systems v. Costco Wholesale Corp. (Case No. 4:21-cv-149-ALM)

2024 ● Willkie-Farr

Netflix, Inc. v. VideoLabs. Inc. (Case No. IPR2023-00628)

2024 ● Quinn Emanuel Urquhart & Sullivan

Purdue University v. Wolfsped (Case No. 1:21-CV-840)

2024 ● Fish & Richardson

Dish Network v. Entropic Communications (Case No. IPR2024-00393)

2025 ● Perkins Coie

Amazon.com, Inc. v. Nokia Technologies Oy (Case No. IPR2024-00572)

2025 ● Fish & Richardson

Dish Network v. Entropic Communications (Case No. IPR2024-00373)

Limited Case and Clientele History

Alston & Bird

- Ericsson

v. Collision Communications (Case No. IPR2022-01233)

- Nokia

v. Neptune Subsea, Xtera (Case No. 1:17-cv-01876)

- Universal Electronics Inc

v. Roku Inc (Case No. IPR2022-00818)

Alavi & Anaipakos PLLC

- Stingray Group Inc. and Stingray Music USA, Inc.

v. Edwin A. Hernandez-Mondragon and Eglia Corp. (Case No. 1:24-cv-21226-RAR)

Arnold & Porter

- Ivantis

v. Glaukos (Case No. 8:18-cv-00620)

- Samsung

v. Jawbone (Case No. 2:21-cv-00186)

Benesch Friedlander Coplan & Aronoff

- Voyis
v. Cathx (Case No. 5:21-cv-00077-RWS)

Buchanan, Ingersoll & Rooney PC

- Google LLC, AT&T Service Inc., T Mobile USA Inc., Cellco Partnership, Ericsson Inc. & Nokia of America Corp.
v. KT Corp & Pegasus Wireless Innovation LLC (Case No. IPR2025-00293)

Deschert LLP

- Smaxtec Animal Care
v. ST Reproductive Technologies, LLC (Case No. IPR2024-00885)

Duane Morris

- Kangxi Communication Technologies
v. Skyworks Solutions Inc. (Case No. IPR2025-00373)
- Microsoft
v. Edge Networking Systems LLC (Case No. 1:24-cv-00215)

Erise I.P.

- Apple
v. Ericsson Inc. (Case No. IPR2022-00715)
v. Future Link Systems (Case No. IPRs 6317804, 6622108, 6807505, and 7917680)
v. INVT (Case No. 20-1881)
v. Navblazer LLC (Case No. IPR2020-01253)
v. Qualcomm (Case No. IPR2018-001281, 39521-00421IP, IPR2018-01282, 39521-00421IP2)

- v. Quest Nettech Corp (Case No. 2:19-cv-00118-JRG)
- v. Telefonaktiebolaget LM Ericsson (Case No. IPR2022-00275)
- v. Theta IP, LLC (Case No. IPR2024-00818)
- v. THL Holding Company LLC (Case No. 1:23-cv-00548)

- Fanduel

- v. CGT (Case No. 19-1393)

- Garmin

- v. Phillips North America LLC (Case No. 2:19-cv-6301-AB-KS)

- Netscout

- v. Longhorn HD LLC (Case No. 2:20-cv-00349)
- v. Implicit, LLC (Case No. 2:18-cv-53-JRG)

- Sony Interactive Entertainment LLC

- v. Bot M8 LLC (Case No. IPR2020-01288)
- v. Infernal Technology LLC (Case No. 2:19-CV-00248-JRG)

- Tesla

- v. Charge Fusion Technologies LLC (Case No. IPR2025-00032)

- Unified Patents

- v. GE Video Compression (Case No. 2:19-cv-248)

Fish & Richardson

- Apple

- v. AliveCor (Case No. 3:21-cv-03958)
- v. LBS Innovations (Case No. 2:19-cv-00119-JRG-RSP)
- v. Koss Corporation (Case No. IPR2021-00305)

- v. Masimo (Case No. IPR 50095-0012IP1, 50095-0012IP2, 50095-0013IP1, 50095-0013IP2, 50095-0006IP1, 50095-0135IP1)
 - v. Neonode (Case No. 21-cv-08872-EMC)
 - v. Qualcomm (Case No. IPR2018-001281, 39521-00421IP, IPR2018-01282, 39521-00421IP2)
 - v. Resonant Sys. (Case No. 7:23-cv-00077-ADA)
- AsusTek Computer Inc.
 - v. Videolabs, Inc. (Case No. 22-CV-00720-ADA)
- Dell
 - v. Neo Wireless (Case No. IPR2022-00616)
- Dish Network
 - v. Entropic Communications, LLC (Case No. 2:2023-CV-01043)
 - v. Entropic Communications LLC (Case No. IPR2024-00393)
 - v. Realtime Adaptive Streaming (Case No. 1:17-CV-02097-RBJ)
 - v. TQ Delta LLC (Case No. 18-1798)
- Evapco Dry Cooling
 - v. SPG Dry Cooling (Case No. IPR2021-00688)
- Genetec
 - v. Sensormatic Electronics (Case No. 1:20-CV-00760)
- Huawei
 - v. Bell Northern Research LLC (Case No. IPR2019-01174)
- Kianxis
 - v. Blue Yonder (Case No. 3:20-cv-03636)

- LG Electronics
 - v. Bell Northern Research LLC (Case No. 3:18-cv-2864-CAB-BLM)
- MediaTek
 - v. MOSAID (Case No. IPR2024-00718)
- Metaswitch
 - v. Sonus Networks (Case No. IPR2018-01719)
- Microsoft
 - v. Throughputer Inc (Case No. IPR2022-00757)
- Mom Enterprises
 - v. Ddrops Company (Case No. 1:22-cv-00332)
- MLC Intellectual Property
 - v. MicronTech (Case No. 3:14-cv-03657-SI)
- Nearmap Inc
 - v. EagleView Technologies (Case No. IPR2022-01009)
- Neuroderm Ltd.
 - v. Abbvie, Inc (Case No. PGR2022-00040)
- Posco Co Ltd.
 - v. Pascal Drillet et al. (Case No. IPR2025-00370)
- Realtek Semiconductor
 - v. Future Link (Case No. IPR2021-01182)

- Quectel
 - v. Koninklijke Philips (Case No. 1:20-cv-01710)
- Samsung
 - v. Aire Technology (Case No. IPR2022-00877)
 - v. Bell Northern Research (Case No. 2:19-cv-00286-JRG)
 - v. Communication Technologies Inc (Case No. IPR2022-01221)
 - v. Jawbone Innovations (Case No. IPR2022-00865)
 - v. MemoryWeb LLC (Case No. IPR2022-00885)
 - v. Dodots Licensing Solutions (Case No. IPR2023-00701)
 - v. Telefonaktiebolaget LM Ericsson (Case No. IPR2021-00615)
- Texas Instruments
 - v. Vantage Micro LLC (Case No. IPR2020-01261)
- Xilinx
 - v. Sentient Sensors LCC (Case No. 1:22-cv-00173)

Goodwin Proctor

- Intel
 - v. Proxense, LLC (Case No. 6:24-cv-00283)
- Samsung Electronics
 - v. Mobile Data Technologies (Case No. 2:24-cv-00435)
- Taiwan Semiconductor Manufacturing Company Ltd.
 - v. Advanced Integrated Circuit Process LLC (Case No. 2:24-cv-00623)

Groombridge Wu

- Nearmap

v. Eagle View Technologies Inc. and Pictometry International Corp. (Case No. 2:21-cv-00283-TS-DAO)

Irell & Manella

- Curium

Latham & Watkins LLP

- Apple

v. Advanced Coding Research LLC

Leydig, Voit & Mayer, Ltd.

- Public Availability Consultancy

O'Melveny & Myers

- Apple

v. Maxell (Case No. 5:19-cv-00036-RWS)

- Disney Media & Entertainment Distribution LLC

v. Digital Media Technology Holdings, LLC (Case No. 2:22-cv-01642)

- Micron Technology Inc.

v. Besang Inc (Case No. IPR2023-00900)

- Samsung

v. Daedalus Prime LLC (1335 Investigation)

Perkins-Coie

- Amazon
 - v. Nokia (Case No. IPR2024-00847 and IPR2024-00848)
 - v. Nokia Technologies Oy (Case No. IPR2024-00572)
- Heru Industries
 - v. The UAB Research Foundation (Case No. IPR2022-01148)
- Intel Corporation
 - v. BeSang Inc. (Case No. IPR2023-00991)
 - v. BitMICRO (Case No. 5:23-cv-00625)
 - v. Telefonaktiebolaget L M Ericsson (Case No. IPR2024-00610)
- Realtek Semiconductor
 - v. Future Link (Case No. IPR2021-01182)
- r-pac
 - v. Adasa, Inc. (Case No. 1:2024cv06102)
- Twitter Inc
 - v. VOIP-Pal.com (Case No. 3:20-cv-02397-JD)
- TCL Industries
 - v. Koninklijke Philips NV (Case No. IPR2021-00495, IPR2021-00496 and IPR2021-00497)
- VusionGroup
 - v. Hanshow (Case No. IPR2024-00857)

Pillsbury Winthrop Shaw Pittman

- Intel
 - v. FG SRC LLC (Case No. 6:20-cv-00315)
- Gravel Rating Systems
 - v. Costco (Case No. 4:21-cv-149)
 - v. Lowe's Home Centers (Case No. 4:21-cv-150)
 - v. T-Mobile USA (Case No. 4:21-cv-152)
 - v. Kohl's Inc. (Case No. 4:21-cv-258)
 - v. Under Armor (Case No. 4:21-cv-356)

Quinn Emanuel

- Exact Sciences Corporation
 - v. Geneoscopy, Inc. (Case No. IPR2024-00459 and IPR2024-1330)
- ServiceNow, Inc.
 - v. InQuisient, Inc. (Case No. 22-900-CJB)
- Wolfsped
 - v. The Trustees of Purdue University (Case No. 1:2021-cv-00840)

Sheppard, Mullin, Richter & Hampton LLP

- Advanced Micro Devices, Inc and Pensando Systems, Inc.
 - v. Concurrent Ventures LLC and Xtreamedge, Inc. (Case No. IPR2025-00478)
- Cadence Design Systems Inc.
 - v. Semiconductor Design Technologies (Case No. 3:23-cv-01001)

Shook, Hardy & Bacon LLP

- HP Inc.

v. Universal Connectivity Technologies (Case No. 1:23-cv-01177)

Stern, Kessler, Goldstein & Fox

- Preliminary Research

Wilmer, Cutler, Pickering, Hale and Dorr

- Apple

v. Shunock (Case No. 1:23-cv-08598)

- Quest Diagnostics Inc., Haystack Oncology Inc. and Haystack Oncology GmbH

v. Natera, Inc. (Case No. 1:23-cv-08598)

- Roche Diabetes, Inc.

v. Trividia Health, Inc. (Case No. 1:24-cv-00668)

Willkie, Farr & Gallagher LLP

- Lenovo, Dell, HP

v. Universal Connectivity Technologies Inc. (Case No. 2:2023cv00449)

- Neurent

v. The Foundry, LLC (Case No. IPR2024-00669)

- Netflix, Inc.

v. VideoLabs, Inc. (Case No. IPR2023-00628)

APPENDIX DEBOER01



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)

Share

Bookmark

Report an Issue

MARC View

LEADER 02030nas a2200433 a 4500

001 49096423

003 SIRSI

005 20251014090047.0

006 m o d

007 cr n

008 991206c19849999ne sx pss 0 0eng c

010 a| 2001262345

016 7 a| 2021913-1 2| DE-600

019 a| DO NOT EDIT this 360 MARC RECORD

022 a| 1572-9338 l| 0254-5330 y| 0254-5330

035 a| (WaSeSS)ssj0001185

037 b| Baltzer Science Publishers, Raphaelplein 3, 1077 PW Amsterdam, The Netherlands

040 a| F#A b| eng c| F#A d| EYM d| CGU d| OCLCQ d| MYG d| U9S d| MYG d| OCLCQ d| BUF d| OCLCQ d| BUF d| OCLCF d| OCLCQ d| WT2 d| OCLCQ d| UKBOL d| WYU d| U3W d| NJT d| VT2 d| SFB d| DLC d| UBY d| CNMTR d| SRU d| OCLCQ d| UAB d| OCLCL d| WaSeSS

042 a| pcc

050 0 0 a| T57.6.A1

082 0 4 a| 519.2 1| <https://id.oclc.org/worldcat/ddc/E3m3d7kt3KQ7KTdh4XVbfC3PKm>

130 0 a| Annals of operations research (Online)

245 1 0 a| Annals of operations research h| [electronic resource].

260 3| <1984-2005> : a| [Amsterdam, the Netherlands] : b| Baltzer Science Publishers

260 3 3| <2006-> : a| Dordrecht : b| Springer Netherlands

310 a| 24 issues per year, b| 2014-

321 a| 10 v. a year, b| 2011-2013

321 a| Frequency varies

362 1 a| Began with v. 1, no. 1 (Feb. 1984).

538 a| Mode of access: World Wide Web.

588 a| Description based on: Vol. 1, no. 1 (Feb. 1984); title from journal contents screen (SpringerLink, viewed Oct. 19, 2009).

588 a| Latest issue consulted: Vol. 286, no. 1/2 (Mar. 2020) (Springer Link, viewed Feb. 19, 2020).

650 0 a| Operations research v| Periodicals.

655 0 a| Electronic journals.

776 0 8 i| Print version: t| Annals of operations research x| 0254-5330 w| (DLC)sn 86020000 w| (OCoLC)11521382

856 4 0 u| <http://SK8ES4MC2L.search.serialssolutions.com/?V=1.0&L=SK8ES4MC2L&S=JCs&C=ANNAOFOPERE&T=marc>

910 a| CONSER online

850 a| DLC

949 a| Electronic resource w| ASIS c| 1 i| 49096423-1001 l| ONLINE m| ONLINE r| Y s| Y t| ONLINE u| 10/18/2025 o| .STAFF. 360 MARC

ASK
A LIBRARIAN

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#)

(814) 865-6368

[Libraries Intranet \(Staff Only\)](#)

[Accessibility Help](#)

[Website Feedback](#)

[Policies and Guidelines](#)

[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#)

[Twitter](#)

[Instagram](#)



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)

Share

Bookmark

Report an Issue



Annals of operations research [electronic resource].

Uniform Title

Annals of operations research (Online)

Published

<1984-2005> : [Amsterdam, the Netherlands] : Baltzer Science Publishers

<2006-> : Dordrecht : Springer Netherlands

Access Online

serialssolutions.com

Dates of Publication and/or Sequential Designation

Began with v. 1, no. 1 (Feb. 1984).

Subject(s)

[Operations research—Periodicals](#)

Genre(s)

[Electronic journals](#)

ISSN

1572-9338

Publication Frequency

24 issues per year, 2014-

10 v. a year, 2011-2013

Frequency varies

Technical Details

Mode of access: World Wide Web.

ASK
A LIBRARIAN [View MARC record](#) | catkey: 49096423

PennState

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#)

(814) 865-6368

[Libraries Intranet \(Staff Only\)](#) [Accessibility Help](#)[Website Feedback](#)[Policies and Guidelines](#)[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#) [Twitter](#) [Instagram](#)



PennState
University Libraries

RENEW MATERIALS

Give Today

My Account



Services Research About Ask

⌚ Pattee and Paterno: View Hours

Online Journals, Books and more

All Journals Only Books Only Videos Only

Title begins with

Search

[Show more search options](#)

Showing results 1 through 1 of 1

for the search: "ANNAOFOPERE"

Refine Results: [All](#) | [Journals Only](#) | [Books Only](#) | [Videos Only](#)

Limit by: [Peer Reviewed](#) | [Open Access](#)



Annals of operations research

ISSN: 0254-5330

Peer Reviewed

[Springer Nature - Springer Journals Archive Collection - Springer](#)

01/01/1984 to 09/30/1996 in [Business and Economics](#)

01/01/1997 to Present in [Springer Nature BTAA Reading TA](#)

[SpringerLink Contemporary \(1997 - Present\)](#)

1997 to Present in [SpringerLINK Contemporary \(BTAA\)](#)

10/01/1997 to 12/31/1997 in [ABI/INFORM Collection](#)

[Advanced Technologies & Aerospace Database](#)

01/01/1999 to 12/31/2017 in [ABI/INFORM Collection](#)

[Advanced Technologies & Aerospace Database](#)

Search inside this journal

Search

[Back to Top](#)



PennState
University Libraries



SERVICES

Borrow & Renew

Course Reserves

Rooms & Spaces

Technology & Equipment

Adaptive Technology & Services

Interlibrary Loan

Teaching Support for Instructors

RESEARCH

Try These First

Citation Guides

ScholarSphere

Collections

Electronic Theses and Dissertations

Research Data Services

ABOUT

Hours

Directory

Libraries and Departments

Jobs

Message from the Dean

Diversity

News and Events

Your Donation at Work

Libraries Calendar

ASK A LIBRARIAN

By Chat

By Phone

In Person

By Email

Schedule a Research Appt.

FAQs (or How Do I?...)

Tutorials

The Search Bar

Thank a Library Worker

[Libraries Home](#) | [Libraries Intranet \(Employees Only\)](#) | [Accessibility Help](#) | [Website Feedback](#) | [Policies and Guidelines](#) | [Acknowledgement of Land](#) | (814) 865-6368

[Hotline](#) | [Privacy Statements](#) | [Non Discrimination](#) | [Accessibility](#) | [Legal Statements](#)

Copyright ©2025 The Pennsylvania State University.

All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

APPENDIX DEBOER02



A Tutorial on the Cross-Entropy Method

PIETER-TJERK DE BOER *

ptdeboer@cs.utwente.nl

Department of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands

DIRK P. KROESE

kroese@maths.uq.edu.au

Department of Mathematics, The University of Queensland, Brisbane 4072, Australia

SHIE MANNOR

shie@ece.mcgill.ca

Department of Electrical and Computer Engineering, McGill University, 3480 University Street, Montreal, Quebec, Canada

REUVEN Y. RUBINSTEIN

ierrr01@ie.technion.ac.il

Department of Industrial Engineering, Technion, Israel Institute of Technology, Haifa 32000, Israel

Received January 2003; Revised November 2003; Accepted January 2004

Abstract. The cross-entropy (CE) method is a new generic approach to combinatorial and multi-extremal optimization and rare event simulation. The purpose of this tutorial is to give a gentle introduction to the CE method. We present the CE methodology, the basic algorithm and its modifications, and discuss applications in combinatorial optimization and machine learning.

Keywords: cross-entropy method, Monte-Carlo simulation, randomized optimization, machine learning, rare events

Many everyday tasks in operations research involve solving complicated optimization problems. The travelling salesman problem (TSP), the quadratic assignment problem (QAP) and the max-cut problem are a representative sample of combinatorial optimization problems (COP) where the problem being studied is completely known and static. In contrast, the buffer allocation problem (BAP) is a *noisy* estimation problem where the objective function needs to be estimated since it is unknown. Discrete event simulation is one method for estimating an unknown objective function.

The purpose of this tutorial is to show that the CE method provides a simple, efficient and general method for solving such problems. Moreover, we wish to show that the CE method is also valuable for *rare event simulation*, where very small probabilities need to be accurately estimated – for example in reliability analysis, or performance analysis of telecommunication systems. This tutorial is intended for a broad audience of operations research specialists, computer scientists, mathematicians, statisticians and engineers. Our aim is to explain the foundations of the CE method and consider various applications.

*Corresponding author.

The CE method was motivated by an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks (Rubinstein, 1997), which involves variance minimization. It was soon realized (Rubinstein, 1999, 2001) that a simple cross-entropy modification of Rubinstein (1997) could be used not only for estimating probabilities of rare events but for solving difficult COPs as well. This is done by translating the “deterministic” optimization problem into a related “stochastic” optimization problem and then using rare event simulation techniques similar to Rubinstein (1997). Several recent applications demonstrate the power of the CE method (Rubinstein, 1999) as a generic and practical tool for solving NP-hard problems.

The CE method involves an iterative procedure where each iteration can be broken down into two phases:

1. Generate a random data sample (trajectories, vectors, etc.) according to a specified mechanism.
2. Update the parameters of the random mechanism based on the data to produce a “better” sample in the next iteration.

The significance of the CE method is that it defines a precise mathematical framework for deriving fast, and in some sense “optimal” updating/learning rules, based on advanced simulation theory. Other well-known randomized methods for combinatorial optimization problems are simulated annealing (Aarts and Korst, 1989), tabu search (Glover and Laguna, 1993), and genetic algorithms (Goldberg, 1989). Recent related work on randomized combinatorial optimization includes the nested partitioning method (Shi and Olafsson, 2000) and the ant colony optimization meta-heuristic (Colomi et al., 1996; Dorigo, Di Caro, and Gambardella, 1999; Gutjahr, 2000).

Many COPs can be formulated as optimization problems concerning a weighted graph. As mentioned before, in CE a deterministic optimization problem is translated into an associated *stochastic* optimization problem. Depending on the particular problem, we introduce randomness in either (a) the nodes or (b) the edges of the graph. We speak of *stochastic node networks* (SNN) in the former case and *stochastic edge networks* (SEN) in the latter. Examples of SNN problems are the maximal cut (max-cut) problem, the buffer allocation problem and clustering problems. Examples of SEN problems are the travelling salesman problem, the quadratic assignment problem, the clique problem, and optimal policy search in Markovian decision problems (MDPs).

It should be emphasized that the CE method may be successfully applied to both deterministic and stochastic COPs. In the latter the objective function itself is random or needs to be estimated via simulation. Stochastic COPs typically occur in stochastic scheduling, flow control and routing of data networks and in various simulation-based optimization problems (Rubinstein and Melamed, 1998), such as the optimal buffer allocation problem (Alon et al., 2005).

Estimation of the probability of rare events is essential for guaranteeing adequate performance of engineering systems. For example, consider a telecommunications system that accepts calls from many customers. Under normal operating conditions each

client may be rejected with a very small probability. Naively, in order to estimate this small probability we would need to simulate the system under normal operating conditions for a long time. A better way to estimate this probability is to use *importance sampling* (IS), which is a well-known variance reduction technique in which the system is simulated under a different set of parameters – or, more generally, a different probability distribution – so as to make the occurrence of the rare event more likely. A major drawback of the IS technique is that the optimal reference (also called tilting) parameters to be used in IS are usually very difficult to obtain. The advantage of the CE method is that it provides a simple adaptive procedure for estimating the optimal reference parameters. Moreover, the CE method also enjoys asymptotic convergence properties. For example, it is shown in Homem-de-Mello and Rubinstein (2002) that for *static* models – cf. Remark 2.1 – under mild regularity conditions the CE method terminates with probability 1 in a finite number of iterations, and delivers a consistent and asymptotically normal estimator for the optimal reference parameters. Recently the CE method has been successfully applied to the estimation of rare event probabilities in dynamic models, in particular queueing models involving both *light* and *heavy* tail input distributions (de Boer, Kroese and Rubinstein, 2002; Kroese and Rubinstein, 2004). In addition to rare event simulation and combinatorial optimization, the CE method can be efficiently applied for continuous multi-extremal optimization, see Rubinstein (1999) and the recent CE book (Rubinstein and Kroese, 2004).

An increasing number of applications is being found for the CE method. Recent publications on applications of the CE method include: buffer allocation (Alon et al., 2005); static simulation models (Homem-de-Mello and Rubinstein, 2002); queueing models of telecommunication systems (de Boer, 2000; de Boer, Kroese, and Rubinstein, 2004); neural computation (Dubin, 2002, 2004); control and navigation (Helvik and Wittner, 2001); DNA sequence alignment (Keith and Kroese, 2002); scheduling (Margolin, 2002, 2004); vehicle routing (Chepuri and Homem-de-Mello, 2005); reinforcement learning (Mannor, Rubinstein, and Gat, 2003; Menache, Mannor, and Shimkin, 2005); project management (Cohen, Golany, and Shtub, 2005); heavy-tail distributions (Asmussen, Kroese, and Rubinstein, 2005); (Kroese and Rubinstein, 2004); CE convergence (Margolin, 2005); network reliability (Hui et al., 2005); repairable systems (Ridder, 2005); and max-cut and bipartition problems (Rubinstein, 2002).

It is not our intention here to compare the CE method with other heuristics. Our intention is mainly to demonstrate its beauty and simplicity and promote CE for further applications to combinatorial and multi-extremal optimization and rare event simulation.

The rest of the tutorial is organized as follows. In Section 1 we present two toy examples that illustrate the basic methodology behind the CE method. The general theory and algorithms are detailed in Section 2. In Section 3 we discuss various applications and examples of using the CE method for solving COPs. In Section 4 two useful modifications of the CE method are discussed. Further developments are briefly reviewed in Section 5.

The CE *home page*, featuring many links, articles, references, tutorials and computer programs on CE, can be found at: <http://www.cemethod.org/>

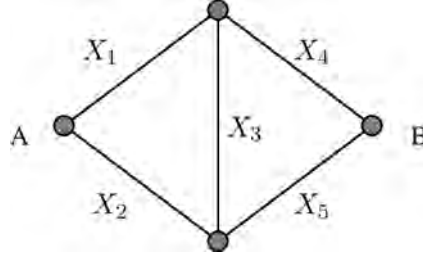


Figure 1. Shortest path from A to B.

1. Methodology: Two examples

In this section we illustrate the methodology of the CE method via two toy examples; one dealing with rare event simulation, and the other with combinatorial optimization.

1.1. A rare event simulation example

Consider the weighted graph of figure 1, with random weights $X_1, \dots, X_5$. Suppose the weights are independent of each other and are exponentially distributed with means $u_1, \dots, u_5$, respectively. Define $\mathbf{X} = (X_1, \dots, X_5)$ and $\mathbf{u} = (u_1, \dots, u_5)$. Denote the probability density function (pdf) of $\mathbf{X}$ by $f(\cdot; \mathbf{u})$; thus,

$$f(\mathbf{x}; \mathbf{u}) = \exp\left(-\sum_{j=1}^5 \frac{x_j}{u_j}\right) \prod_{j=1}^5 \frac{1}{u_j}. \quad (1)$$

Let $S(\mathbf{X})$ be the total length of the shortest path from node A to node B. We wish to estimate from simulation

$$\ell = \mathbb{P}(S(\mathbf{X}) \geq \gamma) = \mathbb{E}I_{\{S(\mathbf{X}) \geq \gamma\}}, \quad (2)$$

that is, the probability that the length of the shortest path $S(\mathbf{X})$ will exceed some fixed γ .

A straightforward way to estimate ℓ in (2) is to use *Crude Monte Carlo* (CMC) simulation. That is, we draw a random sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the distribution of $\mathbf{X}$ and use

$$\frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} \quad (3)$$

as the unbiased estimator of ℓ . However, for large γ the probability ℓ will be very small and CMC requires a very large simulation effort, that is, N needs to be very large in order to estimate ℓ accurately—that is, to obtain a small relative error, for example of 1%. A better way to perform the simulation is to use *importance sampling* (IS). That is, let

g be another probability density such that $g(\mathbf{x}) = 0 \Rightarrow I_{\{S(\mathbf{x}) \geq \gamma\}} f(\mathbf{x}) = 0$. Using the density g we can represent ℓ as

$$\ell = \int I_{\{S(\mathbf{x}) \geq \gamma\}} \frac{f(\mathbf{x})}{g(\mathbf{x})} g(\mathbf{x}) d\mathbf{x} = \mathbb{E}_g I_{\{S(\mathbf{X}) \geq \gamma\}} \frac{f(\mathbf{X})}{g(\mathbf{X})}, \quad (4)$$

where the subscript g means that the expectation is taken with respect to g , which is called the *importance sampling* (IS) density. An unbiased estimator of ℓ is

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i), \quad (5)$$

where $\hat{\ell}$ is called the *importance sampling* (IS) or the *likelihood ratio* (LR) estimator,

$$W(\mathbf{x}) = f(\mathbf{x})/g(\mathbf{x}) \quad (6)$$

is called the *likelihood ratio* (LR), and $\mathbf{X}_1, \dots, \mathbf{X}_N$ is a *random sample* from g , that is, $\mathbf{X}_1, \dots, \mathbf{X}_n$ are iid random vectors with density g . In the particular case where there is no “change of measure”, i.e., $g = f$, we have $W = 1$, and the LR estimator in (6) reduces to the CMC estimator (3).

If we restrict ourselves to g such that $X_1, \dots, X_5$ are independent and exponentially distributed with means $v_1, \dots, v_5$, then

$$W(\mathbf{x}; \mathbf{u}, \mathbf{v}) := \frac{f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{v})} = \exp \left(- \sum_{j=1}^5 x_j \left(\frac{1}{u_j} - \frac{1}{v_j} \right) \right) \prod_{j=1}^5 \frac{v_j}{u_j}. \quad (7)$$

In this case the “change of measure” is determined by the parameter vector $\mathbf{v} = (v_1, \dots, v_5)$. The main problem now is how to select a $\mathbf{v}$ which gives the most accurate estimate of ℓ for a given simulation effort. One of the strengths of the CE method for rare event simulation is that it provides a fast way to determine/estimate the optimal parameters. To this end, without going into the details, a quite general CE algorithm for rare event estimation is outlined below.

Algorithm 1.1

1. Define $\hat{\mathbf{v}}_0 := \mathbf{u}$. Set $t := 1$ (iteration counter).
2. Generate a random sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ according to the pdf $f(\cdot; \hat{\mathbf{v}}_{t-1})$. Calculate the performances $S(\mathbf{X}_i)$ for all i , and order them from smallest to biggest, $S_{(1)} \leq \dots \leq S_{(N)}$. Let $\hat{\gamma}_t$ be the $(1 - \rho)$ sample quantile of performances: $\hat{\gamma}_t := S_{(\lceil (1-\rho)N \rceil)}$, provided this is less than γ . Otherwise, put $\hat{\gamma}_t := \gamma$.
3. Use the **same** sample to calculate, for $j = 1, \dots, n (= 5)$,

$$\hat{v}_{t,j} = \frac{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_t\}} W(\mathbf{X}_i; \mathbf{u}, \hat{\mathbf{v}}_{t-1}) X_{ij}}{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_t\}} W(\mathbf{X}_i; \mathbf{u}, \hat{\mathbf{v}}_{t-1})}.$$

Table 1
Convergence of the sequence $\{(\hat{\gamma}_t, \hat{\mathbf{v}}_t)\}$.

t	$\hat{\gamma}_t$	$\hat{\mathbf{v}}_t$				
0		0.250	0.400	0.100	0.300	0.200
1	0.575	0.513	0.718	0.122	0.474	0.335
2	1.032	0.873	1.057	0.120	0.550	0.436
3	1.502	1.221	1.419	0.121	0.707	0.533
4	1.917	1.681	1.803	0.132	0.638	0.523
5	2.000	1.692	1.901	0.129	0.712	0.564

4. If $\hat{\gamma}_t = \gamma$ then proceed to step 5; otherwise set $t := t + 1$ and reiterate from step 2.
5. Let T be the final iteration. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_{N_1}$ according to the pdf $f(\cdot; \hat{\mathbf{v}}_T)$ and estimate ℓ via the IS estimate

$$\hat{\ell} = \frac{1}{N_1} \sum_{i=1}^{N_1} I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \hat{\mathbf{v}}_T).$$

Note that in steps 2–4 the optimal IS parameter is estimated. In the final step (step 5) this parameter is used to estimate the probability of interest. We need to supply the fraction ρ (typically between 0.01 and 0.1) and the parameters N and N_1 in advance.

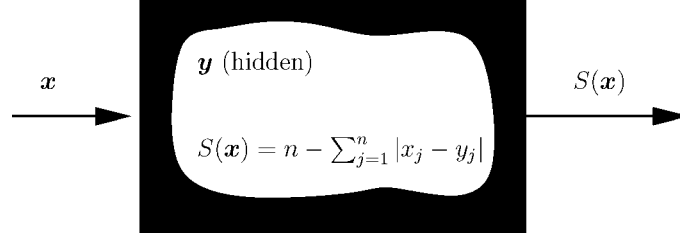
As an example, consider the case where the *nominal* parameter vector $\mathbf{u}$ is given by (0.25, 0.4, 0.1, 0.3, 0.2). Suppose we wish to estimate the probability that the minimum path is greater than $\gamma = 2$. Crude Monte Carlo with 10^7 samples gave an estimate $1.65 \cdot 10^{-5}$ with an estimated *relative error*, RE, (that is, $\text{Var}(\hat{\ell})^{1/2}/\ell$) of 0.165. With 10^8 samples we got the estimate $1.30 \cdot 10^{-5}$ with RE 0.03.

Table 1 displays the results of the CE method, using $N = 1,000$ and $\rho = 0.1$. This table was computed in less than half a second.

Using the estimated optimal parameter vector of $\hat{\mathbf{v}}_5 = (1.692, 1.901, 0.129, 0.712, 0.564)$, the final step with $N_1 = 10^5$ gave now an estimate of $1.34 \cdot 10^{-5}$ with an estimated RE of 0.03. The simulation time was only 3 seconds, using a Matlab implementation on a Pentium III 500 MHz processor. In contrast, the CPU time required for the CMC method with 10^7 samples is approximately 630 seconds, and with 10^8 samples approximately 6350. We see that with a minimal amount of work we have reduced our simulation effort (CPU time) by roughly a factor of 2000.

1.2. A combinatorial optimization example

Consider a binary vector $\mathbf{y} = (y_1, \dots, y_n)$. Suppose that we do not know which components of $\mathbf{y}$ are 0 and which are 1. However, we have an “oracle” which for each binary

Figure 2. A black box for decoding vector $\mathbf{y}$.

input vector $\mathbf{x} = (x_1, \dots, x_n)$ returns the performance or response,

$$S(\mathbf{x}) = n - \sum_{j=1}^n |x_j - y_j|,$$

representing the number of *matches* between the elements of $\mathbf{x}$ and $\mathbf{y}$. Our goal is to present a random search algorithm which reconstructs¹ (decodes) the unknown vector $\mathbf{y}$ by maximizing the function $S(\mathbf{x})$ on the space of n -dimensional binary vectors.

A naive way to find $\mathbf{y}$ is to repeatedly generate binary vectors $\mathbf{X} = (X_1, \dots, X_n)$ such that $X_1, \dots, X_n$ are independent Bernoulli random variables with success probabilities $p_1, \dots, p_n$. We write $\mathbf{X} \sim \text{Ber}(\mathbf{p})$, where $\mathbf{p} = (p_1, \dots, p_n)$. Note that if $\mathbf{p} = \mathbf{y}$, which corresponds to the degenerate case of the Bernoulli distribution, we have $S(\mathbf{X}) = n$, $\mathbf{X} = \mathbf{y}$, and the naive search algorithm yields the optimal solution with probability 1. The CE method for combinatorial optimization consists of casting the underlying problem into the rare event framework (2) and then creating a sequence of parameter vectors $\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \dots$ and levels $\hat{\gamma}_1, \hat{\gamma}_2, \dots$, such that the sequence $\hat{\gamma}_1, \hat{\gamma}_2, \dots$ converges to the optimal performance (n here) and the sequence $\hat{\mathbf{p}}_0, \hat{\mathbf{p}}_1, \dots$ converges to the optimal parameter vector ($\mathbf{y}$ here). Again, the CE procedure—which is similar to the rare event procedure described in Algorithm 1.1—is outlined below, without detail.

Algorithm 1.2

1. Start with some $\hat{\mathbf{p}}_0$, say $\hat{\mathbf{p}}_0 = (1/2, 1/2, \dots, 1/2)$. Let $t := 1$.
2. Draw a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ of Bernoulli vectors with success probability vector $\hat{\mathbf{p}}_{t-1}$. Calculate the performances $S(\mathbf{X}_i)$ for all i , and order them from smallest to biggest, $S_{(1)} \leq \dots \leq S_{(N)}$. Let $\hat{\gamma}_t$ be $(1 - \rho)$ sample quantile of the performances: $\hat{\gamma}_t = S_{(\lceil (1-\rho)N \rceil)}$.

¹Of course, in this toy example the vector $\mathbf{y}$ can be easily reconstructed from the input vectors $(0, 0, \dots, 0)$, $(1, 0, \dots, 0)$, $(0, 1, 0, \dots, 0)$, $\dots$, $(0, \dots, 0, 1)$ only.

Table 2
The convergence of the parameter vector.

t	$\hat{\gamma}_t$	$\hat{\mathbf{p}}_t$									
0		0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
1	7	0.60	0.40	0.80	0.40	1.00	0.00	0.20	0.40	0.00	0.00
2	9	0.80	0.80	1.00	0.80	1.00	0.00	0.00	0.40	0.00	0.00
3	10	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00
4	10	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00

3. Use the **same** sample to calculate $\hat{\mathbf{p}}_t = (\hat{p}_{t,1}, \dots, \hat{p}_{t,n})$ via

$$\hat{p}_{t,j} = \frac{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_t\}} I_{\{X_{ij}=1\}}}{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_t\}}}, \quad (8)$$

$j = 1, \dots, n$, where $\mathbf{X}_i = (X_{i1}, \dots, X_{in})$.

4. If the **stopping criterion** is met, then **stop**; otherwise set $t := t + 1$ and reiterate from step 2.

A possible stopping criterion is to stop when $\hat{\gamma}_t$ does not change for a number of subsequent iterations. Another possible stopping criterion is to stop when the vector $\hat{\mathbf{p}}_t$ has converged to a degenerate – that is, binary – vector.

Note that the interpretation of 8 is very simple: to *update* the j th success probability we count how many vectors of the last sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ have a performance greater than or equal to $\hat{\gamma}_t$ and have the j th coordinate equal to 1, and we divide (normalize) this by the number of vectors that have a performance greater than or equal to $\hat{\gamma}_t$.

As an example, consider the case $n = 10$, where $\mathbf{y} = (1, 1, 1, 1, 1, 0, 0, 0, 0, 0)$. Using a sample $N = 50$, $\rho = 0.1$ and the initial parameter vector $\hat{\mathbf{p}}_0 = (1/2, 1/2, \dots, 1/2)$, Algorithm 1.2 yields the results given in Table 2.

We see that the $\hat{\mathbf{p}}_t$ and $\hat{\gamma}_t$ converge very quickly to the optimal degenerated CE parameter vector $\mathbf{p}^* = \mathbf{y}$ and optimal performance $\gamma^* = n$, respectively.

Remark 1.1 (Likelihood ratio term). Note that algorithms 1.1 and 1.2 are almost the same. The most important difference is the absence of the likelihood ratio term W in step 3 of Algorithm 1.2. The reason is that the choice of the initial parameter vector $\hat{\mathbf{p}}_0$ is *quite arbitrary*, so using W would be meaningless, while in rare event simulation it is an essential part of the estimation problem. For more details see Remark 2.4 below.

2. The main algorithm(s)

2.1. The CE method for rare event simulation

In this subsection we discuss the main ideas behind the CE algorithm for rare event simulation. When reading this section, the reader is encouraged to refer back to the toy example presented in Section 1.1.

Let $\mathbf{X} = (X_1, \dots, X_n)$ be a random vector taking values in some space $\mathcal{X}$. Let $\{f(\cdot; \mathbf{v})\}$ be a family of *probability density functions* (pdfs) on $\mathcal{X}$, with respect to some base measure ν . Here $\mathbf{v}$ is a real-valued parameter (vector). Thus,

$$\mathbb{E}H(\mathbf{X}) = \int_{\mathcal{X}} H(\mathbf{x}) f(\mathbf{x}; \mathbf{v}) \nu(d\mathbf{x}),$$

for any (measurable) function H . In most (or all) applications ν is either a counting measure or the Lebesgue measure. In the former case f is often called a probability mass function, but in this paper we will always use the generic terms density or pdf. For the rest of this section we take for simplicity $\nu(d\mathbf{x}) = d\mathbf{x}$.

Let S be some real-valued function on $\mathcal{X}$. Suppose we are interested in the probability that $S(\mathbf{x})$ is greater than or equal to some real number γ , under $f(\cdot; \mathbf{u})$. This probability can be expressed as

$$\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geq \gamma) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}}.$$

If this probability is very small, say smaller than 10^{-5} , we call $\{S(\mathbf{X}) \geq \gamma\}$ a *rare event*.

A straightforward way to estimate ℓ is to use crude Monte-Carlo simulation: Draw a random sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from $f(\cdot; \mathbf{u})$; then

$$\frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}}$$

is an unbiased estimator of ℓ . However this poses serious problems when $\{S(\mathbf{X}) \geq \gamma\}$ is a rare event. In that case a large simulation effort is required in order to estimate ℓ accurately, i.e., with small relative error or a narrow confidence interval.

An alternative is based on importance sampling: take a random sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from an *importance sampling* (different) density g on $\mathcal{X}$, and evaluate ℓ using the LR estimator (see (5))

$$\hat{\ell} = \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} \frac{f(\mathbf{X}_i; \mathbf{u})}{g(\mathbf{X}_i)}. \quad (9)$$

It is well known that the best way to estimate ℓ is to use the change of measure with density

$$g^*(\mathbf{x}) := \frac{I_{\{S(\mathbf{x}) \geq \gamma\}} f(\mathbf{x}; \mathbf{u})}{\ell}. \quad (10)$$

Namely, by using this change of measure we have in (9)

$$I_{\{S(\mathbf{X}_i) \geq \gamma\}} \frac{f(\mathbf{X}_i; \mathbf{u})}{g^*(\mathbf{X}_i)} = \ell,$$

for all i . In other words, the estimator (9) has zero variance, and we need to produce only $N = 1$ sample.

The obvious difficulty is of course that this g^* depends on the unknown parameter ℓ . Moreover, it is often convenient to choose a g in the family of densities $\{f(\cdot; \mathbf{v})\}$. The idea now is to choose the parameter vector, called the *reference parameter* (sometimes called tilting parameter) $\mathbf{v}$ such that the *distance* between the densities g^* and $f(\cdot; \mathbf{v})$ is minimal. A particular convenient measure of distance between two densities g and h is the *Kullback-Leibler distance*, which is also termed the *cross-entropy* between g and h . The Kullback-Leibler distance is defined as:

$$\mathcal{D}(g, h) = \mathbb{E}_g \ln \frac{g(\mathbf{X})}{h(\mathbf{X})} = \int g(\mathbf{x}) \ln g(\mathbf{x}) d\mathbf{x} - \int g(\mathbf{x}) \ln h(\mathbf{x}) d\mathbf{x}.$$

We note that $\mathcal{D}$ is not a “distance” in the formal sense; for example, it is not symmetric.

Minimizing the Kullback-Leibler distance between g^* in (10) and $f(\cdot; \mathbf{v})$ is equivalent to choosing $\mathbf{v}$ such that $-\int g^*(\mathbf{x}) \ln f(\mathbf{x}; \mathbf{v}) d\mathbf{x}$ is minimized, which is equivalent to solving the maximization problem

$$\max_{\mathbf{v}} \int g^*(\mathbf{x}) \ln f(\mathbf{x}; \mathbf{v}) d\mathbf{x}. \quad (11)$$

Substituting g^* from (10) into (11) we obtain the maximization program

$$\max_{\mathbf{v}} \int \frac{I_{\{S(\mathbf{x}) \geq \gamma\}} f(\mathbf{x}; \mathbf{u})}{\ell} \ln f(\mathbf{x}; \mathbf{v}) d\mathbf{x}, \quad (12)$$

which is equivalent to the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}} \ln f(\mathbf{X}; \mathbf{v}), \quad (13)$$

where D is implicitly defined above. Using again importance sampling, with a change of measure $f(\cdot; \mathbf{w})$ we can rewrite (13) as

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{w}} I_{\{S(\mathbf{X}) \geq \gamma\}} W(\mathbf{X}; \mathbf{u}, \mathbf{w}) \ln f(\mathbf{X}; \mathbf{v}), \quad (14)$$

for any reference parameter $\mathbf{w}$, where

$$W(\mathbf{x}; \mathbf{u}, \mathbf{w}) = \frac{f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{w})}$$

is the *likelihood ratio*, at $\mathbf{x}$, between $f(\cdot; \mathbf{u})$ and $f(\cdot; \mathbf{w})$. The optimal solution of (14) can be written as

$$\mathbf{v}^* = \underset{\mathbf{v}}{\operatorname{argmax}} \mathbb{E}_{\mathbf{w}} I_{\{S(\mathbf{X}) \geq \gamma\}} W(\mathbf{X}; \mathbf{u}, \mathbf{w}) \ln f(\mathbf{X}; \mathbf{v}). \quad (15)$$

We may estimate $\mathbf{v}^*$ by solving the following stochastic program (also called *stochastic counterpart* of (14))

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \mathbf{w}) \ln f(\mathbf{X}_i; \mathbf{v}), \quad (16)$$

where $\mathbf{X}_1, \dots, \mathbf{X}_N$ is a random sample from $f(\cdot; \mathbf{w})$. In typical applications the function $\hat{D}$ in (16) is convex and differentiable with respect to $\mathbf{v}$, see also Rubinstein and Shapiro (1993) and, thus, the solution of (16) may be readily obtained by solving (with respect to $\mathbf{v}$) the following system of equations:

$$\frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \mathbf{w}) \nabla \ln f(\mathbf{X}_i; \mathbf{v}) = \mathbf{0}, \quad (17)$$

where the gradient is with respect to $\mathbf{v}$.

The advantage of this approach is that the solution of (17) can often be calculated *analytically*. In particular, this happens if the distributions of the random variables belong to a *natural exponential family* (NEF). For further details see Rubinstein and Kroese (2004) and Example 2.1 below.

It is important to note that with $\mathbf{w} = \mathbf{u}$ the CE program (16) is useful only in the case where the probability of the “target event” $\{S(\mathbf{X}) \geq \gamma\}$ is not too small, say $\ell \geq 10^{-5}$. For rare event probabilities, however (when, say, $\ell < 10^{-5}$), the program (16) is difficult to carry out. Namely, due to the rareness of the events $\{S(\mathbf{X}_i) \geq \gamma\}$, most of the indicator random variables $I_{\{S(\mathbf{X}_i) \geq \gamma\}}$, $i = 1, \dots, N$ will be zero, for moderate N . The same holds for the derivatives of $\hat{D}(\mathbf{v})$ as given in the left-hand side of (17).

A *multi-level* algorithm can be used to overcome this difficulty. The idea is to construct a sequence of reference parameters $\{\mathbf{v}_t, t \geq 0\}$ and a sequence of levels $\{\gamma_t, t \geq 1\}$, and iterate in both γ_t and $\mathbf{v}_t$ (see Algorithm 2.1 below).

We initialize by choosing a not very small ρ , say $\rho = 10^{-2}$ and by defining $\mathbf{v}_0 = \mathbf{u}$. Next, we let γ_1 ($\gamma_1 < \gamma$) be such that, under the original density $f(\mathbf{x}; \mathbf{u})$, the probability $\ell_1 = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma_1\}}$ is at least ρ . We then let $\mathbf{v}_1$ be the optimal CE reference parameter for estimating ℓ_1 , and repeat the last two steps iteratively with the goal of estimating the pair $\{\ell, \mathbf{v}^*\}$. In other words, each iteration of the algorithm consists of two main *phases*. In the first phase γ_t is updated, in the second $\mathbf{v}_t$ is updated. Specifically, starting with $\mathbf{v}_0 = \mathbf{u}$ we obtain the subsequent γ_t and $\mathbf{v}_t$ as follows:

1. **Adaptive updating of γ_t .** For a fixed $\mathbf{v}_{t-1}$, let γ_t be a $(1 - \rho)$ -quantile of $S(\mathbf{X})$ under $\mathbf{v}_{t-1}$. That is, γ_t satisfies

$$\mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geq \gamma_t) \geq \rho, \quad (18)$$

$$\mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \leq \gamma_t) \geq 1 - \rho, \quad (19)$$

where $\mathbf{X} \sim f(\cdot; \mathbf{v}_{t-1})$.

A simple estimator $\hat{\gamma}_t$ of γ_t can be obtained by drawing a random sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from $f(\cdot; \mathbf{v}_{t-1})$, calculating the performances $S(\mathbf{X}_i)$ for all i , ordering them from smallest to biggest: $S_{(1)} \leq \dots \leq S_{(N)}$ and finally, evaluating the $(1 - \rho)$ sample quantile as

$$\hat{\gamma}_t = S_{(\lceil (1-\rho)N \rceil)}. \quad (20)$$

Note that $S_{(j)}$ is called the j -th *order-statistic* of the sequence $S(\mathbf{X}_1), \dots, S(\mathbf{X}_N)$. Note also that $\hat{\gamma}_t$ is chosen such that the event $\{S(\mathbf{X}) \geq \hat{\gamma}_t\}$ is not too rare (it has a probability of around ρ), and therefore updating the reference parameter via a procedure such as (22) is not void of meaning.

2. **Adaptive updating of $\mathbf{v}_t$.** For fixed γ_t and $\mathbf{v}_{t-1}$, derive $\mathbf{v}_t$ from the solution of the following CE program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}} I_{\{S(\mathbf{X}) \geq \gamma_t\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v}_{t-1}) \ln f(\mathbf{X}; \mathbf{v}). \quad (21)$$

The stochastic counterpart of (21) is as follows: for fixed $\hat{\gamma}_t$ and $\hat{\mathbf{v}}_{t-1}$, derive $\hat{\mathbf{v}}_t$ from the solution of following program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_t\}} W(\mathbf{X}_i; \mathbf{u}, \hat{\mathbf{v}}_{t-1}) \ln f(\mathbf{X}_i; \mathbf{v}). \quad (22)$$

Thus, at the first iteration, starting with $\hat{\mathbf{v}}_0 = \mathbf{u}$, to get a good estimate for $\hat{\mathbf{v}}_1$, the target event is artificially made less rare by (temporarily) using a level $\hat{\gamma}_1$ which is chosen smaller than γ . The value for $\hat{\mathbf{v}}_1$ obtained in this way will (hopefully) make the event $\{S(\mathbf{X}) \geq \gamma\}$ less rare in the next iteration, so in the next iteration a value $\hat{\gamma}_2$ can be used which is closer to γ itself. The algorithm terminates when at some iteration t a level is reached which is at least γ and thus the original value of γ can be used without getting too few samples.

As mentioned before, the optimal solutions of (21) and (22) can often be obtained *analytically*, in particular when $f(\mathbf{x}; \mathbf{v})$ belongs to a NEF.

The above rationale results in the following algorithm.

Algorithm 2.1 (Main CE Algorithm for Rare Event Simulation)

1. Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set $t = 1$ (iteration = level counter).
2. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the density $f(\cdot; \mathbf{v}_{t-1})$ and compute the sample $(1 - \rho)$ -quantile $\hat{\gamma}_t$ of the performances according to (20), provided $\hat{\gamma}_t$ is less than γ . Otherwise set $\hat{\gamma}_t = \gamma$.
3. Use the **same** sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ to solve the stochastic program (22). Denote the solution by $\hat{\mathbf{v}}_t$.
4. If $\hat{\gamma}_t < \gamma$, set $t = t + 1$ and reiterate from step 2. Else proceed with step 5.
5. Estimate the rare-event probability ℓ using the LR estimate

$$\hat{\ell} = \frac{1}{N_1} \sum_{i=1}^{N_1} I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \hat{\mathbf{v}}_T), \quad (23)$$

where T denotes the final number of iterations (= number of levels used).

Example 2.1 We return to the example in Section 1.1. In this case, from (1) we have

$$\frac{\partial}{\partial v_j} \ln f(\mathbf{x}; \mathbf{v}) = \frac{x_j}{v_j^2} - \frac{1}{v_j},$$

so that the j th equation of (17) becomes

$$\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \mathbf{w}) \left(\frac{X_{ij}}{v_j^2} - \frac{1}{v_j} \right) = 0, \quad j = 1, \dots, 5;$$

therefore

$$v_j = \frac{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \mathbf{w}) X_{ij}}{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \mathbf{w})}, \quad (24)$$

which leads to the updating formula in step 3 of Algorithm 1.1. Actually, one can show that if the distributions belong to a natural exponential family, the updating formula always becomes (24).

We have intentionally used the notation $\hat{\gamma}_t$ and $\hat{\mathbf{v}}_t$ in Algorithm 2.1 above, rather than the more obvious γ_t and $\mathbf{v}_t$, in order to distinguish it from its *deterministic* counterpart, which is obtained by replacing sample means and sample quantiles by expectations and quantiles. For easy reference and better insight we present below the deterministic version of Algorithm 2.1. We have omitted the IS step in the algorithm below.

Algorithm 2.2 (Deterministic version of the CE algorithm)

1. Define $\mathbf{v}_0 := \mathbf{u}$. Set $t = 1$.
2. Calculate γ_t as

$$\gamma_t := \max\{s : \mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geq s) \geq \rho\}, \quad (25)$$

provided this is less than γ ; otherwise put $\gamma_t := \gamma$.

3. Calculate $\mathbf{v}_t$ as

$$\mathbf{v}_t = \underset{\mathbf{v}}{\operatorname{argmax}} \mathbb{E}_{\mathbf{v}_{t-1}} I_{\{S(\mathbf{X}) \geq \gamma_t\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v}_{t-1}) \ln f(\mathbf{X}; \mathbf{v}). \quad (26)$$

4. If $\gamma_t = \gamma$, then **stop**; otherwise set $t := t + 1$ and reiterate from step 2.

Remark 2.1 (Static Simulation). The above method has been formulated for finite-dimensional random vectors only; this is sometimes referred to as *static* simulation. For infinite-dimensional random vectors or stochastic processes we need a more subtle treatment.

We will not go into details here, but the main point is that Algorithm 2.1 holds true without much alteration and can be readily applied to estimation problems involving both light and heavy tail distributions (de Boer, Kroese, and Rubinstein, 2004; Rubinstein and Kroese, 2004; Asmussen, Kroese, and Rubinstein, 2005).

Remark 2.2 (Variance Minimization). An alternative way to obtain a good reference parameter is to choose $\mathbf{v}$ such that the *variance*, or equivalently, the second moment, of the IS estimator is minimal. In other words we wish to find

$$*\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \mathbb{E}_{\mathbf{v}} [I_{\{S(\mathbf{X}) \geq \gamma\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v})]^2. \quad (27)$$

More generally, using again the principle of importance sampling, this is equivalent to finding

$$*\mathbf{v} = \underset{\mathbf{v}}{\operatorname{argmin}} \mathbb{E}_{\mathbf{w}} I_{\{S(\mathbf{X}) \geq \gamma\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v}) W(\mathbf{X}; \mathbf{u}, \mathbf{w}) \quad (28)$$

for *any* reference parameter $\mathbf{w}$. As in (16), we can estimate $*\mathbf{v}$ as the solution to the stochastic program

$$\min_{\mathbf{v}} \hat{V}(\mathbf{v}) = \min_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) W(\mathbf{X}_i; \mathbf{u}, \mathbf{w}), \quad (29)$$

where $\mathbf{X}_1, \dots, \mathbf{X}_N$ is a random sample from $f(\cdot; \mathbf{w})$. However, the evaluation of (29) in general involves complicated *numerical* optimization, and it is much more convenient to use the closed-form updating formulas that follow from CE minimization.

2.2. The CE-method for combinatorial optimization

In this subsection we discuss the main ideas behind the CE algorithm for combinatorial optimization. When reading this section, the reader is encouraged to refer back to the toy example in Section 1.2.

Consider the following general maximization problem. Let $\mathcal{X}$ be a finite set of *states*, and let S be a real-valued *performance function* on $\mathcal{X}$. We wish to find the maximum of S over $\mathcal{X}$, and the corresponding state(s) at which this maximum is attained. Let us denote the maximum by γ^* . Thus,

$$S(\mathbf{x}^*) = \gamma^* = \max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x}). \quad (30)$$

The starting point in the methodology of the CE method is to associate an *estimation problem* with the optimization problem (30). To this end we define a collection of indicator functions $\{I_{\{S(\mathbf{x}) \geq \gamma\}}\}$ on $\mathcal{X}$ for various thresholds or *levels* $\gamma \in \mathbb{R}$. Next, let $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ be a family of (discrete) pdfs on $\mathcal{X}$, parameterized by a real-valued parameter (vector) $\mathbf{v}$.

For a certain $\mathbf{u} \in \mathcal{V}$ we associate with (30) the problem of estimating the number

$$\ell(\gamma) = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geq \gamma) = \sum_{\mathbf{x}} I_{\{S(\mathbf{x}) \geq \gamma\}} f(\mathbf{x}; \mathbf{u}) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}}, \quad (31)$$

where $\mathbb{P}_{\mathbf{u}}$ is the probability measure under which the random state $\mathbf{X}$ has pdf $f(\cdot; \mathbf{u})$, and $\mathbb{E}_{\mathbf{u}}$ denotes the corresponding expectation operator. We will call the estimation problem (31) the *associated stochastic problem* (ASP). To indicate how (31) is associated with (30), suppose for example that γ is equal to γ^* and that $f(\cdot; \mathbf{u})$ is the uniform density on $\mathcal{X}$. Note that, typically, $\ell(\gamma^*) = f(\mathbf{x}^*; \mathbf{u}) = 1/|\mathcal{X}|$ – where $|\mathcal{X}|$ denotes the number of elements in $\mathcal{X}$ – is a very small number. Thus, for $\gamma = \gamma^*$ a natural way to estimate $\ell(\gamma)$ would be to use the LR estimator (23) with reference parameter $\mathbf{v}^*$ given by

$$\mathbf{v}^* = \operatorname{argmax}_{\mathbf{v}} \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}} \ln f(\mathbf{X}; \mathbf{v}). \quad (32)$$

This parameter could be estimated by

$$\hat{\mathbf{v}}^* = \operatorname{argmax}_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} \ln f(\mathbf{X}_i; \mathbf{v}), \quad (33)$$

where the $\mathbf{X}_i$ are generated from pdf $f(\cdot; \mathbf{u})$.

It is plausible that if γ is close to γ^* then $f(\cdot; \mathbf{v}^*)$ assigns most of its probability mass close to $\mathbf{x}^*$, and thus can be used to generate an approximate solution to (30). However, it is important to note that the estimator (33) is only of practical use when $I_{\{S(\mathbf{X}) \geq \gamma\}} = 1$ for enough samples. This means for example that when γ is close to γ^* , $\mathbf{u}$ needs to be such that $\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geq \gamma)$ is not too small. Thus, the choice of $\mathbf{u}$ and γ in (31) are closely related. On the one hand we would like to choose γ as close as possible to γ^* , and find (an estimate of) $\mathbf{v}^*$ via the procedure above, which assigns almost all mass

to state(s) close to the optimal state. On the other hand, we would like to keep γ relative small in order to obtain an accurate (low RE) estimator for $\mathbf{v}^*$.

The situation is very similar to the rare event simulation case of Section 2.1. The idea, based essentially on Algorithm 2.1, is again to adopt a two-phase multi-level approach in which we simultaneously construct a *sequence* of levels $\hat{\gamma}_1, \hat{\gamma}_2, \dots, \hat{\gamma}_T$ and parameter (vectors) $\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots, \hat{\mathbf{v}}_T$ such that $\hat{\gamma}_T$ is close to the optimal γ^* and $\hat{\mathbf{v}}_T$ is such that the corresponding density assigns high probability mass to the collection of states that give a high performance.

This strategy is embodied in the following procedure, see e.g., Rubinstein (1999):

Algorithm 2.3 (Main CE Algorithm for Optimization)

1. Define $\hat{\mathbf{v}}_0 = \mathbf{u}$. Set $t = 1$ (level counter).
2. Generate a sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ from the density $f(\cdot; \mathbf{v}_{t-1})$ and compute the sample $(1 - \rho)$ -quantile $\hat{\gamma}_t$ of the performances according to (20).
3. Use the **same** sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ and solve the stochastic program (22) with $W = 1$. Denote the solution by $\hat{\mathbf{v}}_t$.
4. If for some $t \geq d$, say $d = 5$,

$$\hat{\gamma}_t = \hat{\gamma}_{t-1} = \dots = \hat{\gamma}_{t-d}, \quad (34)$$

then **stop** (let T denote the final iteration); otherwise set $t = t + 1$ and reiterate from step 2.

Note that the stopping criterion, the initial vector $\hat{\mathbf{v}}_0$, the sample size N and the number ρ have to be specified in advance, but that the rest of the algorithm is “self-tuning”.

Remark 2.3 (Smoothed Updating). Instead of updating the parameter vector $\hat{\mathbf{v}}_{t-1}$ to $\hat{\mathbf{v}}_t$ directly via (33) we often use a *smoothed* updating procedure in which

$$\hat{\mathbf{v}}_t = \alpha \hat{\mathbf{w}}_t + (1 - \alpha) \hat{\mathbf{v}}_{t-1}, \quad (35)$$

where $\hat{\mathbf{w}}_t$ is the vector derived via (22) with $W = 1$. This is especially relevant for optimization problems involving discrete random variables. The main reason why this heuristic smoothed updating procedure performs better is that it prevents the occurrences of 0s and 1s in the parameter vectors; once such an entry is 0 or 1, it often will remain so forever, which is undesirable. We found empirically that a value of α between $0.4 \leq \alpha \leq 0.9$ gives the best results. Clearly for $\alpha = 1$ we have the original updating rule in Algorithm 2.3.

In many applications we observed numerically that the sequence of pdfs $f(\cdot; \hat{\mathbf{v}}_0), f(\cdot; \hat{\mathbf{v}}_1), \dots$ converges to a degenerate measure (Dirac measure), assigning all probability mass to a single state $\mathbf{x}_T$, for which, by definition, the function value is greater than or equal to $\hat{\gamma}_T$.

Remark 2.4 (Similarities and Differences). Despite the great similarity between Algorithm 2.1 and Algorithm 2.3 it is important to note a number of differences. For example, the role of the initial reference parameter $\mathbf{u}$ is significantly different. In Algorithm 2.1 $\mathbf{u}$ is the unique *nominal* parameter for estimating $\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geq \gamma)$. However, in Algorithm 2.3 the choice for the initial parameter $\mathbf{u}$ is fairly arbitrary; it is only used to define the ASP. In contrast to Algorithm 2.1 the ASP for Algorithm 2.3 is redefined after each iteration. In particular, in steps 2 and 3 of Algorithm 2.3 we determine the optimal reference parameter associated with $\mathbb{P}_{\hat{\mathbf{v}}_{t-1}}(S(\mathbf{X}) \geq \hat{\gamma}_t)$, instead of $\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geq \hat{\gamma}_t)$. Consequently, the likelihood ratio term W that plays a crucial role in Algorithm 2.1 does not appear in Algorithm 2.3.

The above procedure can, in principle, be applied to any discrete and continuous (multi-extremal) optimization problem. However, for each individual problem two essential ingredients need to be supplied.

1. We need to specify how the samples are generated. In other words, we need to specify the family of pdfs $\{f(\cdot; \mathbf{v})\}$.
2. We need to calculate the updating rules for the parameters, based on cross-entropy minimization.

In general there are many ways to generate samples from $\mathcal{X}$, and it is not always immediately clear which way of generating the sample will yield better results or easier updating formulas.

Example 2.2. We return to the example from Section 1.2. In this case, the random vector $\mathbf{X} = (X_1, \dots, X_n) \sim \text{Ber}(\mathbf{p})$, and the parameter vector $\mathbf{v}$ is $\mathbf{p}$. Consequently, the pdf is

$$f(\mathbf{X}; \mathbf{p}) = \prod_{i=1}^n p_i^{X_i} (1 - p_i)^{1-X_i},$$

and since each X_j can only be 0 or 1,

$$\frac{\partial}{\partial p_j} \ln f(\mathbf{X}; \mathbf{p}) = \frac{X_j}{p_j} - \frac{1 - X_j}{1 - p_j} = \frac{1}{(1 - p_j)p_j} (X_j - p_j).$$

Now we can find the maximum in (33) by setting the first derivatives w.r.t. p_j to zero, for $j = 1, \dots, n$:

$$\frac{\partial}{\partial p_j} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} \ln f(\mathbf{X}_i; \mathbf{p}) = \frac{1}{(1 - p_j)p_j} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} (X_{ij} - p_j) = 0.$$

Thus, we get

$$p_j = \frac{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}} X_{ij}}{\sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \gamma\}}}, \quad (36)$$

which immediately implies (8).

Several remarks are in order.

Remark 2.5 (Maximum Likelihood Estimation). It is interesting to note the connection between (33) and *maximum likelihood estimation* (MLE). In the MLE problem we are given data $\mathbf{x}_1, \dots, \mathbf{x}_N$ which are thought to be the outcomes of i.i.d. random variables $\mathbf{X}_1, \dots, \mathbf{X}_N$ (random sample) each having a distribution $f(\cdot; \mathbf{v})$, where the parameter (vector) $\mathbf{v}$ is an element of some set V . We wish to estimate $\mathbf{v}$ on the basis of the data $\mathbf{x}_1, \dots, \mathbf{x}_N$. The *maximum likelihood estimate* (MLE) is that parameter $\hat{\mathbf{v}}$ which maximizes the joint density of $\mathbf{X}_1, \dots, \mathbf{X}_N$ for the given data $\mathbf{x}_1, \dots, \mathbf{x}_N$. In other words,

$$\hat{\mathbf{v}} = \operatorname{argmax}_{\mathbf{v}} \prod_{i=1}^N f(\mathbf{x}_i; \mathbf{v}).$$

The corresponding random variable, obtained by replacing $\mathbf{x}_i$ with $\mathbf{X}_i$ is called the *maximum likelihood estimator* (MLE as well), also denoted by $\hat{\mathbf{v}}$. Since $\ln(\cdot)$ is an increasing function we have

$$\hat{\mathbf{v}} = \operatorname{argmax}_{\mathbf{v}} \sum_{i=1}^N \ln f(\mathbf{X}_i; \mathbf{v}). \quad (37)$$

Solving (37) is similar to solving (33). The only difference is the indicator function $I_{\{S(\mathbf{x}_i) \geq \gamma\}}$. We can write Step 3 in Algorithm 2.3 as

$$\hat{\mathbf{v}}_t = \operatorname{argmax}_{\mathbf{v}} \sum_{\mathbf{X}_i: S(\mathbf{X}_i) \geq \hat{\gamma}_t} \ln f(\mathbf{X}_i; \mathbf{v}).$$

In other words, $\hat{\mathbf{v}}_t$ is equal to the MLE of $\hat{\mathbf{v}}_{t-1}$ based only on the vectors $\mathbf{x}_i$ in the random sample for which the performance is greater than or equal to $\hat{\gamma}_t$. For example, in Example 2.2 the MLE of p_j based on a random sample $\mathbf{X}_1, \dots, \mathbf{X}_N$ is

$$\hat{p}_j = \frac{\sum_{i=1}^N X_{ij}}{N}.$$

Thus, if we base the MLE only on those vectors that have performance greater than or equal to γ , we obtain (36) immediately.

Remark 2.6 (Parameters). The choice for the sample size N and the parameter ρ depends on the size of the problem and the number of parameters in the ASP. In particular, for a SNN-type problem it is suggested to take the sample size as $N = c n$, where n is the number of *nodes* and c a constant ($c > 1$), say $5 \leq c \leq 10$. In contrast, for a SEN-type problem it is suggested to take $N = c n^2$, where n^2 is the number of *edges* in the network.

It is crucial to realize that the sample sizes $N = cn$ and $N = cn^2$ (with $c > 1$) are associated with the number of ASP parameters (n and n^2) that one needs to estimate for the SNN and SEN problems, respectively (see also the max-cut and the TSP examples below). Clearly, in order to reliably estimate k parameters, one needs to take *at least* a sample $N = ck$ for some constant $c > 1$. Regarding ρ , it is suggested to take ρ around 0.01, provided n is reasonably large, say $n \geq 100$; and it is suggested to take a larger ρ , say $\rho \approx \ln(n)/n$, for $n < 100$.

Alternatively, the choice of N and ρ can be determined *adaptively*. For example, in Homem-de-Mello and Rubinstein (2002) an adaptive algorithm is proposed that adjusts N automatically. The FACE algorithm of Section 4 is another option.

3. Various applications

In this section we discuss applications of the CE method to two typical COPs. We start in Section 3.1 with an SNN problem—the max-cut problem. We explain how to apply the CE method for this problem and provide two simple examples. In Section 3.2 we consider a typical SEN problem—the travelling salesman problem. We demonstrate the usefulness of the CE method and its fast convergence in a number of numerical examples. We further illustrate the dynamics of the CE method and show how fast the reference parameters converge.

3.1. The max-cut problem

The max-cut problem in a graph can be formulated as follows. Given a weighted graph $G(V, E)$ with node set $V = \{1, \dots, n\}$ and edge set E , partition the nodes of the graph into two subsets V_1 and V_2 such that the sum of the weights of the edges going from one subset to the other is maximized. We assume the weights are non-negative. We note that the max-cut problem is an NP-hard problem. Without loss of generality, we assume that the graph is complete. For simplicity we assume the graph is not directed. We can represent the possibly zero edge weights via a non-negative, symmetric *cost* matrix $C = (c_{ij})$ where c_{ij} denotes the weight of the edge from i to j .

Formally, a *cut* is a partition $\{V_1, V_2\}$ of V . For example, if $V = \{1, \dots, 6\}$, then $\{\{1, 3, 4\}, \{2, 5, 6\}\}$ is a possible cut. The *cost* of a cut is the sum of the weights across the cut. As an example, consider the following 6×6 cost matrix

$$C = \begin{pmatrix} 0 & c_{12} & c_{13} & 0 & 0 & 0 \\ c_{21} & 0 & c_{23} & c_{24} & 0 & 0 \\ c_{31} & c_{32} & 0 & c_{34} & c_{35} & 0 \\ 0 & c_{42} & c_{43} & 0 & c_{45} & c_{46} \\ 0 & 0 & c_{53} & c_{54} & 0 & c_{56} \\ 0 & 0 & 0 & c_{64} & c_{65} & 0 \end{pmatrix} \quad (38)$$

corresponding to the graph in figure 3.

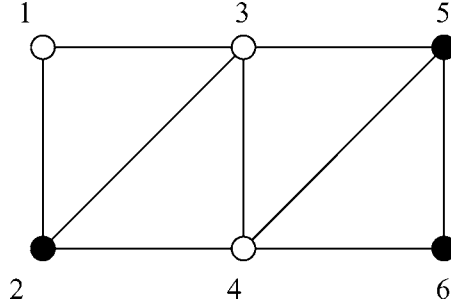


Figure 3. A 6-node network with the cut $\{\{1, 3, 4\}, \{2, 5, 6\}\}$.

For example, the cost of the cut $\{\{1, 3, 4\}, \{2, 5, 6\}\}$ is

$$c_{12} + c_{32} + c_{35} + c_{42} + c_{45} + c_{46}.$$

It will be convenient to represent a cut via a *cut vector* $\mathbf{x} = (x_1, \dots, x_n)$, where $x_i = 1$ if node i belongs to the same partition as node 1, and 0 otherwise. By definition $x_1 = 1$. For example, the cut in figure 3 can be represented via the cut vector $(1, 0, 1, 1, 0, 0)$.

Let $\mathcal{X}$ be the set of all cut vectors $\mathbf{x} = (1, x_2, \dots, x_n)$ and let $S(\mathbf{x})$ be the corresponding cost of the cut. We wish to maximize S via the CE method. Thus, (a) we need to specify how the random cut vectors are generated, and (b) calculate the corresponding updating formulas. The most natural and easiest way to generate the cut vectors is to let $X_2, \dots, X_n$ be independent Bernoulli random variables with success probabilities $p_2, \dots, p_n$, exactly as in the second toy example, see Section 1.2. It immediately follows, see Example 2.2, that the updating formula in Algorithm 2.3 at the t th iteration is given by

$$\hat{p}_{t,j} = \frac{\sum_{i=1}^N I_{\{S(\mathbf{x}_i) \geq \hat{\gamma}_t\}} I_{\{x_{ij}=1\}}}{\sum_{i=1}^N I_{\{S(\mathbf{x}_i) \geq \hat{\gamma}_t\}}}, \quad j = 2, \dots, n. \quad (39)$$

Remark 3.1 (r -partitions). We can readily extend the procedure to the case in which the node set V is partitioned into $r > 2$ subsets $\{V_1, \dots, V_r\}$ such that the sum of the total weights of all edges going from the subset V_i to subset V_j , $i, j = 1, \dots, r$, ($i \neq j$) is maximized. In this case one can follow the basic steps of Algorithm 2.3 using independent r point distributions instead of independent Bernoulli distributions.

To illustrate the convergence of the CE algorithm we provide an example in which the exact optimal updating parameter is computed via (32) rather than estimated via (33). In other words, instead of Algorithm 2.3 its deterministic version Algorithm 2.2 is used (with $W = 1$ in (26)).

Example 3.1. Consider the 5-node graph with the cost matrix

$$C = \begin{pmatrix} 0 & 1 & 3 & 5 & 6 \\ 1 & 0 & 3 & 6 & 5 \\ 3 & 3 & 0 & 2 & 2 \\ 5 & 6 & 2 & 0 & 2 \\ 6 & 5 & 2 & 2 & 0 \end{pmatrix}. \quad (40)$$

It is readily seen that in this case the optimal cut vector is $\mathbf{x}^* = (1, 1, 0, 0, 0)$ with $S(\mathbf{x}^*) = \gamma^* = 28$.

We shall show next that in the “deterministic” version of Algorithm 2.3 the parameter vectors $\mathbf{p}_0, \mathbf{p}_1, \dots$ converge to the optimal $\mathbf{p}^* = (1, 1, 0, 0, 0)$ after two iterations, provided $\rho = 10^{-1}$ and $\mathbf{p}_0 = (1, 1/2, 1/2, 1/2, 1/2)$.

In the first step of the first iteration, we have to determine γ_1 from

$$\gamma_t = \max \{ \gamma \text{ s.t. } \mathbb{E}_{\mathbf{p}_{t-1}} I_{\{S(\mathbf{X}) \geq \gamma\}} \geq 0.1 \}. \quad (41)$$

Under parameter vector $\mathbf{p}_0$, $S(\mathbf{X})$ can take values, $\{0, 10, 15, 18, 19, 20, 21, 26, 28\}$ with probabilities $\{1, 1, 4, 2, 2, 2, 2, 1, 1\}/16$. Hence, we find $\gamma_1 = 26$. In the second step, we need to solve

$$\mathbf{p}_t = \operatorname{argmax}_{\mathbf{p}} \mathbb{E}_{\mathbf{p}_{t-1}} I_{\{S(\mathbf{X}) \geq \gamma_t\}} \ln f(\mathbf{X}; \mathbf{p}), \quad (42)$$

which has a solution similar to (36), namely

$$p_{t,j} = \frac{\mathbb{E}_{\mathbf{p}_{t-1}} I_{\{S(\mathbf{X}) \geq \gamma_t\}} X_j}{\mathbb{E}_{\mathbf{p}_{t-1}} I_{\{S(\mathbf{X}) \geq \gamma_t\}}}.$$

There are only two vectors $\mathbf{x}$ for which $S(\mathbf{x}) \geq 26$, namely $(1, 1, 1, 0, 0)$ and $(1, 1, 0, 0, 0)$, and both have probability $1/16$ under $\mathbf{p}_0$. Thus,

$$p_{1,j} = \begin{cases} \frac{2/16}{2/16} = 1 & \text{for } j = 1, 2, \\ \frac{1/16}{2/16} = \frac{1}{2} & \text{for } j = 3, \\ \frac{0}{2/16} = 0 & \text{for } j = 4, 5. \end{cases}$$

In the second iteration $S(\mathbf{X})$ is 26 or 28 with probability $1/2$. Thus, step (41) yields the (optimal) $\gamma_2 = 28$, and step (42) gives $\mathbf{p}_2 = (1, 1, 0, 0, 0)$.

Example 3.2 (A synthetic max-cut problem). Since the max-cut problem is NP hard (Garey and Johnson, 1979; Papadimitriou and Yannakakis, 1991), no efficient method

for solving the max-cut problem exists. The naive total enumeration routine is only feasible for small graphs, say for those with $n \leq 30$ nodes. Although the Branch-and-Bound heuristic can solve medium size problems exactly, it too will run into problems when n becomes large.

In order to verify the accuracy of the CE method we construct an artificial graph such that the solution is available in advance. In particular, for $m \in \{1, \dots, n\}$ consider the following symmetric cost matrix:

$$C = \begin{pmatrix} Z_{11} & B_{12} \\ B_{21} & Z_{22} \end{pmatrix}, \quad (43)$$

where Z_{11} is an $m \times m$ symmetric matrix in which all the upper-diagonal elements are generated from a $U(a, b)$ distribution (and all lower-diagonal elements follow by symmetry), Z_{22} is a $(n - m) \times (n - m)$ symmetric matrix which is generated in a similar way as Z_{11} , and all the other elements are c , apart from the diagonal elements, which are of course 0.

It is not difficult to see that for $c > b(n - m)/m$, by construction, the optimal cut is given by $V^* = \{V_1^*, V_2^*\}$, with

$$V_1^* = \{1, \dots, m\} \quad \text{and} \quad V_2^* = \{m + 1, \dots, n\}, \quad (44)$$

and the optimal value of the cut is

$$\gamma^* = c m (n - m).$$

Of course the same optimal solution and optimal value can be found for the general case where the elements in Z_{11} and Z_{22} are generated via an arbitrary bounded support distribution with the maximal value of the support less than b .

Table 3 lists a typical output of Algorithm 2.3 applied to the synthetic max-cut problem, for a network with $n = 400$ nodes. In this table we list besides the $(1 - \rho)$ -quantile of the performances $\hat{\gamma}_t$ also the *best* of the performances in each iteration, denoted by $S_{t,(N)}$, and the Euclidean distance

$$\|\hat{\mathbf{p}}_t - \mathbf{p}^*\| = \sqrt{(\hat{p}_{t,i} - p_i^*)^2},$$

as a measure how close the reference vector is to the optimal reference vector $\mathbf{p}^* = (1, 1, \dots, 1, 0, 0, \dots, 0)$.

In this particular example, $m = 200$ and the Z_{11} and Z_{22} are generated from the $U(0, 1)$ distribution. The elements in B_{12} and B_{21} are constant $c = 1$. The CE parameter are chosen as follows: rarity parameter $\rho = 0.1$; smoothing parameter $\alpha = 1.0$ (no smoothing); stopping constant $d = 3$; and number of samples per iteration $N = 1000$.

The CPU time is only 100 seconds, using a Matlab implementation on a pentium III, 500 MHz processor. We see that the CE algorithm converges quickly, yielding the exact optimal solution 40000 in less than 23 iterations.

Table 3
A typical performance of Algorithm 2.3 for the synthetic max-cut problem with $n = 400$, $d = 3$, $\rho = 0.1$, $\alpha = 1.0$, $N = 1000$.

t	$\hat{\gamma}_t$	$S_{t,(N)}$	$\ \hat{\mathbf{p}}_t - \mathbf{p}^*\ $
1	30085.3	30320.9	9.98
2	30091.3	30369.4	10.00
3	30113.7	30569.8	9.98
4	30159.2	30569.8	9.71
5	30350.8	30652.9	9.08
6	30693.6	31244.7	8.37
7	31145.1	31954.8	7.65
8	31711.8	32361.5	6.94
9	32366.4	33050.3	6.27
10	33057.8	33939.9	5.58
11	33898.6	34897.9	4.93
12	34718.9	35876.4	4.23
13	35597.1	36733.0	3.60
14	36368.9	37431.7	3.02
15	37210.5	38051.2	2.48
16	37996.7	38654.5	1.96
17	38658.8	39221.9	1.42
18	39217.1	39707.8	1.01
19	39618.3	40000.0	0.62
20	39904.5	40000.0	0.29
21	40000.0	40000.0	0.14
22	40000.0	40000.0	0.00
23	40000.0	40000.0	0.00

Figures 4 and 5 illustrate the convergence of the reference vectors $\mathbf{p}_t$ to the optimal $\mathbf{p}^*$.

3.2. The travelling salesman problem

The travelling salesman problem (TSP) can be formulated as follows. Consider a weighted graph G with n nodes, labelled $1, 2, \dots, n$. The nodes represent cities, and the edges represent the roads between the cities. Each edge from i to j has weight or cost c_{ij} , representing the length of the road. The problem is to find the shortest *tour* that visits all the cities exactly once² (except the starting city, which is also the terminating city), see figure 6.

Without loss of generality, let us assume that the graph is *complete*, and that some of the weights may be $+\infty$. Let $\mathcal{X}$ be the set of all possible tours and let $S(\mathbf{x})$ the total length

²In some versions of the TSP cities can be visited more than once.

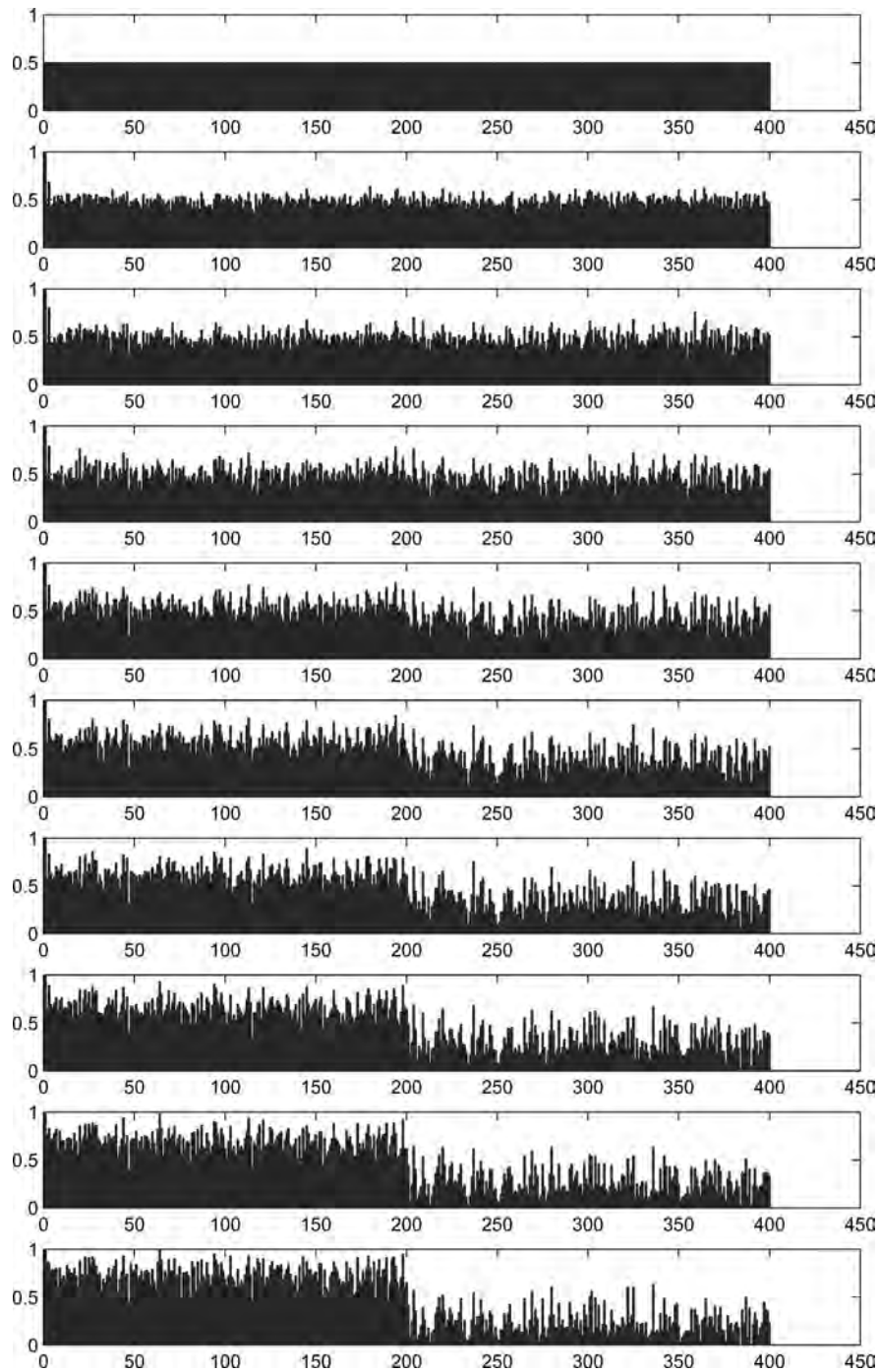


Figure 4. Sequence of reference vectors for the synthetic max-cut problem with 400 nodes. Iterations 0,1,...,9.

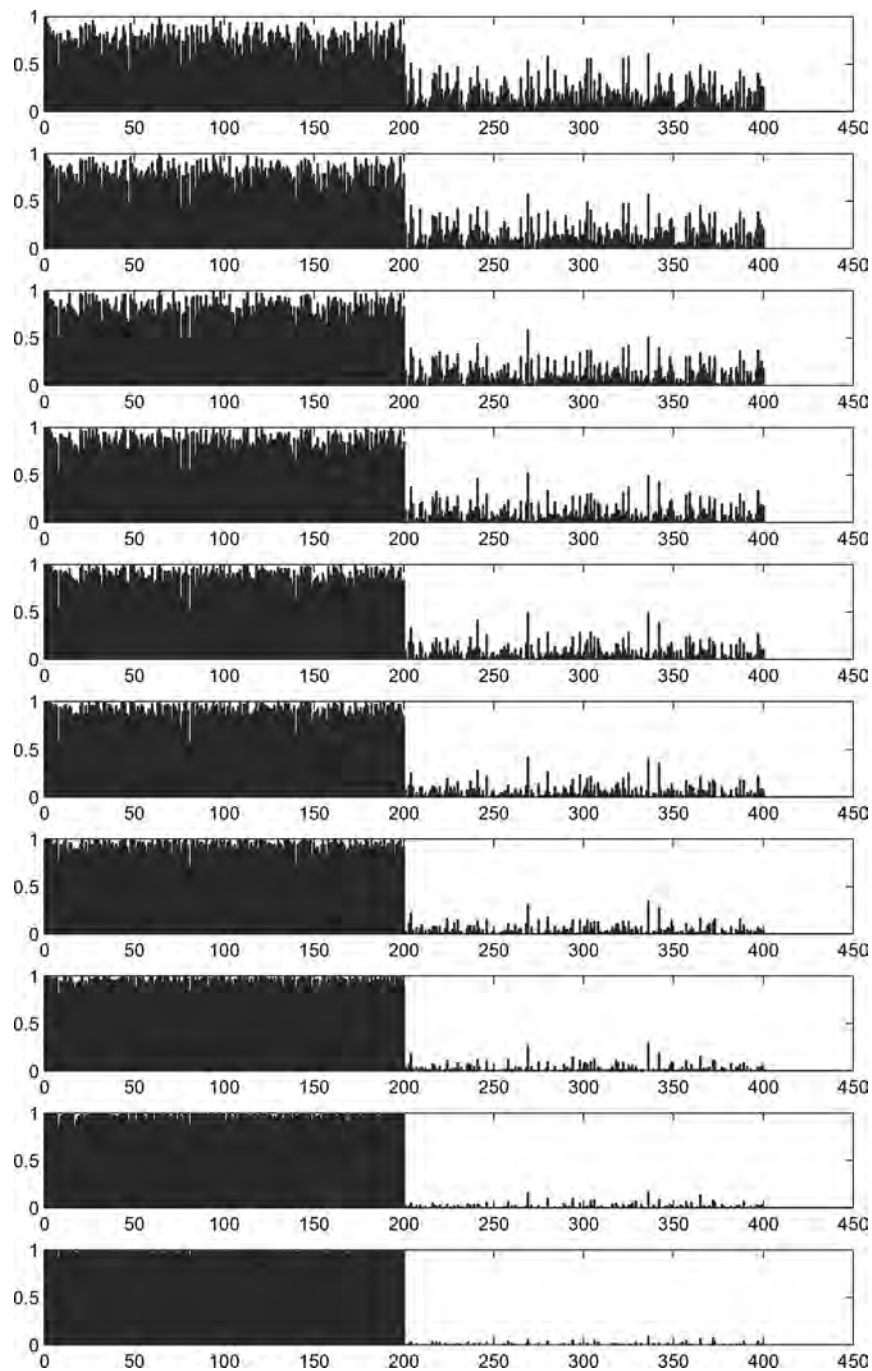
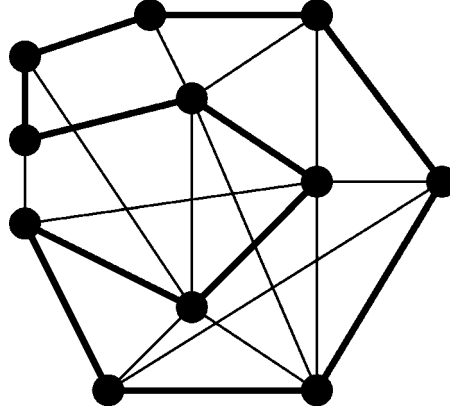


Figure 5. Sequence of reference vectors for the synthetic max-cut problem with 400 nodes. Iterations 10, ..., 19.

Figure 6. Find the shortest tour x visiting all nodes.

of tour $x \in \mathcal{X}$. We can represent each tour via a *permutation* of $(1, \dots, n)$. For example for $n = 4$, the permutation $(1, 3, 2, 4)$ represents the tour $1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 1$. In fact, we may as well represent a tour via a permutation $x = (x_1, \dots, x_n)$ with $x_1 = 1$. From now on we identify a tour with its corresponding permutation, where $x_1 = 1$. We may now formulate the TSP as follows:

$$\min_{x \in \mathcal{X}} S(x) = \min_{x \in \mathcal{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i, x_{i+1}} + c_{x_n, 1} \right\}. \quad (45)$$

Note that the number of elements in $\mathcal{X}$ is typically very large:

$$|\mathcal{X}| = (n-1)! \quad (46)$$

This is exactly the setting of Section 2.2, so we can use the CE-Method to solve (45). Note however that we need to modify Algorithm 2.3 since we have here a *minimization* problem.

In order to apply the CE algorithm we need to specify (a) how to generate the random tours, and (b) how to update the parameters at each iteration.

The easiest way to explain how the tours are generated and how the parameters are updated is to relate (45) to an *equivalent* minimization problem. Let

$$\tilde{\mathcal{X}} = \{(x_1, \dots, x_n) : x_1 = 1, x_i \in \{1, \dots, n\}, i = 2, \dots, n\},$$

be the set of vectors that correspond to paths that start and end in 1 and can visit the same city more than once. Note that $|\tilde{\mathcal{X}}| = n^{n-1}$, and $\mathcal{X} \subset \tilde{\mathcal{X}}$. When $n = 4$, we have for example $x = (1, 3, 1, 3) \in \tilde{\mathcal{X}}$, corresponding to the path $1 \rightarrow 3 \rightarrow 1 \rightarrow 3 \rightarrow 1$. Again we will identify the vectors with their corresponding paths. Define the function $\tilde{S}$ on $\tilde{\mathcal{X}}$ by $\tilde{S}(x) = S(x)$, if $x \in \mathcal{X}$ and $\tilde{S}(x) = \infty$, otherwise. Then, obviously (45) is equivalent

to the minimization problem

$$\text{minimize } \tilde{S}(\mathbf{x}) \text{ over } \mathbf{x} \in \tilde{\mathcal{X}}. \quad (47)$$

A simple method to generate a random path $\mathbf{X} = (X_1, \dots, X_n)$ in $\tilde{\mathcal{X}}$ is to use a Markov chain of the graph G , starting at node 1, and stopping after n steps. Let $P = (p_{ij})$ denote the one-step transition matrix of this Markov chain. We assume that the diagonal elements of P are 0, and that all other elements of P are strictly positive, but otherwise P is a general $n \times n$ stochastic matrix.

The pdf $f(\cdot; P)$ of $\mathbf{X}$ is thus parameterized by the matrix P and its logarithm is given by

$$\ln f(\mathbf{x}; P) = \sum_{r=1}^n \sum_{i,j} I_{\{\mathbf{x} \in \tilde{\mathcal{X}}_{ij}(r)\}} \ln p_{ij},$$

where $\tilde{\mathcal{X}}_{ij}(r)$ is the set of all paths in $\tilde{\mathcal{X}}$ for which the r th transition is from node i to j . The updating rules for this modified optimization problem follow from 22 ($W = 1$), with $\{S(\mathbf{X}_i) \geq \gamma\}$ replaced with $\{\tilde{S}(\mathbf{X}_i) \leq \gamma\}$, under the condition that the rows of P sum up to 1. Using Lagrange multipliers $u_1, \dots, u_n$ we obtain the maximization problem

$$\max_P \min_u \left[\mathbb{E}_P I_{\{\tilde{S}(\mathbf{X}) \leq \gamma\}} \ln f(\mathbf{X}; P) + \sum_{i=1}^n u_i \left(\sum_{j=1}^n p_{ij} - 1 \right) \right].$$

Differentiating the expression within square brackets above with respect to p_{ij} , yields, for all $j = 1, \dots, n$,

$$\frac{\mathbb{E}_P I_{\{\tilde{S}(\mathbf{X}) \leq \gamma\}} \sum_{r=1}^n I_{\{\mathbf{X} \in \tilde{\mathcal{X}}_{ij}(r)\}}}{p_{ij}} + u_i = 0.$$

Summing over $j = 1, \dots, n$ gives $\mathbb{E}_P I_{\{\tilde{S}(\mathbf{X}) \leq \gamma\}} \sum_{r=1}^n I_{\{\mathbf{X} \in \tilde{\mathcal{X}}_i(r)\}} = -u_i$, where $\tilde{\mathcal{X}}_i(r)$ is the set of paths for which the r -th transition starts from node i . It follows that the optimal p_{ij} is given by

$$p_{ij} = \frac{\mathbb{E}_P I_{\{\tilde{S}(\mathbf{X}) \leq \gamma\}} \sum_{r=1}^n I_{\{\mathbf{X} \in \tilde{\mathcal{X}}_{ij}(r)\}}}{\mathbb{E}_P I_{\{\tilde{S}(\mathbf{X}) \leq \gamma\}} \sum_{r=1}^n I_{\{\mathbf{X} \in \tilde{\mathcal{X}}_i(r)\}}}.$$

The corresponding estimator is

$$\hat{p}_{ij} = \frac{\sum_{k=1}^N I_{\{\tilde{S}(\mathbf{X}_k) \leq \gamma\}} \sum_{r=1}^n I_{\{\mathbf{X}_k \in \tilde{\mathcal{X}}_{ij}(r)\}}}{\sum_{k=1}^N I_{\{\tilde{S}(\mathbf{X}_k) \leq \gamma\}} \sum_{r=1}^n I_{\{\mathbf{X}_k \in \tilde{\mathcal{X}}_i(r)\}}}. \quad (48)$$

This has a very simple interpretation. To update p_{ij} we simply take the fraction of times that the transitions from i to j occurred, taking only those paths into account that have a total length less than or equal to γ .

This is how we could, *in principle*, carry out the sample generation and parameter updating for problem (47). We generate the path via a Markov process with transition matrix P , and use updating formula (48). However, *in practice*, we would never generate the paths in this way, since the majority of these paths would be irrelevant since they would not constitute a tour, and therefore their $\tilde{S}$ values would be ∞ . In order to avoid the generation of irrelevant paths, we proceed as follows.

Algorithm 3.1 (Generation of permutations (tours) in the TSP)

1. Define $P^{(1)} = P$ and $X_1 = 1$. Let $k = 1$.
2. Obtain $P^{(k+1)}$ from $P^{(k)}$ by first setting the X_k -th column of $P^{(k)}$ to 0 and then normalizing the rows to sum up to 1. Generate X_{k+1} from the distribution formed by the X_k -th row of $P^{(k)}$.
3. If $k = n - 1$ then **stop**; otherwise set $k = k + 1$ and reiterate from step 2.

It is important to realize that the updating formula remains the same; by using Algorithm 3.1 we are merely *speeding-up* our naive way of generating the tours. Moreover, since we now only generate *tours*, the updated value for p_{ij} can be estimated as

$$\hat{p}_{ij} = \frac{\sum_{k=1}^N I_{\{S(\mathbf{x}_k) \leq \gamma\}} I_{\{\mathbf{x}_k \in \mathcal{X}_{ij}\}}}{\sum_{k=1}^N I_{\{S(\mathbf{x}_k) \leq \gamma\}}}, \quad (49)$$

where $\mathcal{X}_{ij}$ is the set of tours in which the transition from i to j is made. This has the same “natural” interpretation as discussed for (48).

To complete the algorithm, we need to specify the initialization conditions and the stopping criterion. For the initial matrix $\hat{P}_0$ we could simply take all off-diagonal elements equal to $1/(n - 1)$ and for the stopping criterion use formula (34).

Numerical Examples

To demonstrate the usefulness of the CE algorithm and its fast and accurate convergence we provide a number of numerical examples. The first example concerns the benchmark TSP problem `ft53` taken from the URL

<http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/>

Table 4
A typical performance of Algorithm 2.3 for the TSP problem `ft53` with $n = 53$ nodes, $d = 5$, $\rho = 0.01$, $\alpha = 0.7$, $N = 10 n^2 = 28090$.

t	$\hat{\gamma}_t$	$S_{t,(1)}$	P_t^{mm}	fdiff
1	23234.00	21111.00	0.0354	0.3243
2	20611.00	18586.00	0.0409	0.3744
3	18686.00	16819.00	0.0514	0.3707
4	17101.00	14890.00	0.0465	0.3676
5	15509.00	13459.00	0.0698	0.3635
6	14449.00	12756.00	0.0901	0.3492
7	13491.00	11963.00	0.0895	0.3404
8	12773.00	11326.00	0.1065	0.3250
9	12120.00	10357.00	0.0965	0.3291
10	11480.00	10216.00	0.1034	0.3167
11	11347.00	9952.00	0.1310	0.3017
12	10791.00	9525.00	0.1319	0.3111
13	10293.00	9246.00	0.1623	0.3049
14	10688.00	9176.00	0.1507	0.2726
15	9727.00	8457.00	0.1346	0.3203
16	9263.00	8424.00	0.1436	0.2929
17	9422.00	8614.00	0.1582	0.2490
18	9155.00	8528.00	0.1666	0.2468
19	8661.00	7970.00	0.1352	0.2543
20	8273.00	7619.00	0.1597	0.2360
21	8096.00	7485.00	0.1573	0.2163
22	7868.00	7216.00	0.1859	0.1889
23	7677.00	7184.00	0.2301	0.1737
24	7519.00	7108.00	0.2421	0.1569
25	7420.00	7163.00	0.2861	0.1495
26	7535.00	7064.00	0.3341	0.1508
27	7506.00	7072.00	0.3286	0.1576
28	7199.00	7008.00	0.3667	0.1341
29	7189.00	7024.00	0.3487	0.1212
30	7077.00	7008.00	0.4101	0.0998
31	7068.00	7008.00	0.4680	0.1051

Table 4 presents the performance of the CE Algorithm for the problem `ft53`, which defines an asymmetric fully connected graph of size 53, where the value of each edge c_{ij} is given.

The CE parameters were: stopping parameter $d = 5$, rarity parameter $\rho = 0.01$, sample size $N = 10 n^2 = 28090$ and smoothing parameter $\alpha = 0.7$. The relative experimental error of the solution is

$$\varepsilon = \frac{\hat{\gamma}_T - \gamma^*}{\gamma^*} = 1.5\%,$$

Table 5
Case studies for the TSP.

File	n	$\bar{T}$	$\bar{\gamma}_1$	$\bar{\gamma}_T$	γ^*	$\bar{\varepsilon}$	ε_*	ε^*	CPU
br17	17	23.8	68.2	39.0	39	0.0	0.0	0.0	9
ftv33	34	31.2	3294.0	1312.2	1286	2.0	0.0	6.2	73
ftv35	36	31.5	3714.0	1490.0	1473	1.2	0.4	1.8	77
ftv38	39	33.8	4010.8	1549.8	1530	1.3	0.4	3.2	132
p43	43	44.5	9235.5	5624.5	5620	0.1	0.0	0.1	378
ftv44	45	35.5	4808.2	1655.8	1613	2.7	1.3	3.3	219
ftv47	48	40.2	5317.8	1814.0	1776	2.1	0.6	4.1	317
ry48p	48	40.8	40192.0	14845.5	14422	2.9	1.9	5.0	345
ft53	53	39.5	20889.5	7103.2	6905	2.9	2.5	3.5	373
ftv55	56	40.0	5835.8	1640.0	1608	2.0	0.2	4.3	408
ftv64	65	43.2	6974.2	1850.0	1839	0.6	0.0	1.4	854
ftv70	71	47.0	7856.8	1974.8	1950	1.3	0.4	3.7	1068
ft70	70	42.8	64199.5	39114.8	38673	1.1	0.3	1.9	948

where $\gamma^* = 6905$ is the best known solution. The CPU time was approximately 6 minutes. In Table 4 $S_{t,(1)}$ denotes the length of smallest tour in iteration t . We also included the two quantities P_t^{mm} and fdiff , which give extra information about the dynamics. Specifically, fdiff is the proportion, out of N , of different values for the performance function, for the current iteration t ; and P_t^{mm} is the minimum of the maximum elements in each row of matrix $\hat{P}_t$.

Similar performances were found for other TSP problems in the benchmark library above. Table 5 presents the performance of Algorithm 2.3 for a selection of case studies from this library. In all numerical results we use the same CE parameters as for the `ft53` problem, that is $\rho = 10^{-2}$, $N = 10n^2$, $\alpha = 0.7$ (smoothing parameter in (35)) and $d = 5$ (in (34)). To study the variability in the solutions, each problem was repeated 10 times. In Table 5 n denotes the number of nodes of the graph, $\bar{T}$ denotes the average total number of iterations needed before stopping, $\bar{\gamma}_1$ and $\bar{\gamma}_T$ denote the average initial and average final estimates of the optimal solution, γ^* denotes the best known (optimal) solution, $\bar{\varepsilon}$ denotes the average percentage relative experimental error based on 10 replications, ε_* and ε^* denote the smallest and the largest percentage relative error among the 10 generated solutions, and finally CPU denotes the average CPU time in seconds.

We found that decreasing the sample size N from $N = 10n^2$ to $N = 5n^2$ all relative experimental errors ε in Table 5 increase at most by a factor of 1.5.

Dynamics

Finally, as an illustration of the dynamics of the CE algorithm, we display below the sequence of matrices $\hat{P}_0, \hat{P}_1, \dots$ for a TSP with $n = 10$ cities, where the optimal tour is

$(1, 2, 3, \dots, 10, 1)$.

$$\hat{P}_0 = \begin{pmatrix} 0.00 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.00 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.00 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.00 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.00 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.00 & 0.11 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.00 & 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.00 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.00 & 0.11 \\ 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.11 & 0.00 \end{pmatrix}.$$

$$\hat{P}_1 = \begin{pmatrix} 0.00 & 0.31 & 0.04 & 0.08 & 0.04 & 0.19 & 0.08 & 0.08 & 0.12 & 0.08 \\ 0.04 & 0.00 & 0.33 & 0.08 & 0.17 & 0.08 & 0.08 & 0.04 & 0.04 & 0.12 \\ 0.08 & 0.08 & 0.00 & 0.23 & 0.04 & 0.04 & 0.12 & 0.19 & 0.08 & 0.15 \\ 0.12 & 0.19 & 0.08 & 0.00 & 0.12 & 0.08 & 0.08 & 0.08 & 0.19 & 0.08 \\ 0.08 & 0.08 & 0.19 & 0.08 & 0.00 & 0.23 & 0.08 & 0.04 & 0.15 & 0.08 \\ 0.04 & 0.04 & 0.08 & 0.04 & 0.12 & 0.00 & 0.50 & 0.08 & 0.08 & 0.04 \\ 0.23 & 0.08 & 0.08 & 0.04 & 0.08 & 0.04 & 0.00 & 0.27 & 0.08 & 0.12 \\ 0.08 & 0.15 & 0.04 & 0.04 & 0.19 & 0.08 & 0.08 & 0.00 & 0.27 & 0.08 \\ 0.08 & 0.08 & 0.04 & 0.12 & 0.08 & 0.15 & 0.08 & 0.04 & 0.00 & 0.35 \\ 0.21 & 0.08 & 0.17 & 0.08 & 0.04 & 0.12 & 0.08 & 0.12 & 0.08 & 0.00 \end{pmatrix}.$$

$$\hat{P}_2 = \begin{pmatrix} 0.00 & 0.64 & 0.03 & 0.06 & 0.04 & 0.04 & 0.06 & 0.04 & 0.04 & 0.06 \\ 0.03 & 0.00 & 0.58 & 0.07 & 0.07 & 0.05 & 0.05 & 0.03 & 0.03 & 0.08 \\ 0.05 & 0.05 & 0.00 & 0.52 & 0.04 & 0.03 & 0.08 & 0.04 & 0.05 & 0.15 \\ 0.04 & 0.13 & 0.05 & 0.00 & 0.22 & 0.18 & 0.05 & 0.04 & 0.25 & 0.05 \\ 0.06 & 0.04 & 0.09 & 0.04 & 0.00 & 0.60 & 0.04 & 0.03 & 0.04 & 0.06 \\ 0.03 & 0.03 & 0.05 & 0.03 & 0.04 & 0.00 & 0.71 & 0.05 & 0.05 & 0.03 \\ 0.20 & 0.04 & 0.05 & 0.03 & 0.05 & 0.03 & 0.00 & 0.51 & 0.05 & 0.04 \\ 0.05 & 0.08 & 0.03 & 0.04 & 0.23 & 0.05 & 0.05 & 0.00 & 0.42 & 0.05 \\ 0.05 & 0.05 & 0.04 & 0.07 & 0.07 & 0.10 & 0.05 & 0.03 & 0.00 & 0.54 \\ 0.50 & 0.05 & 0.04 & 0.05 & 0.04 & 0.08 & 0.05 & 0.14 & 0.05 & 0.00 \end{pmatrix}.$$

$$\hat{P}_3 = \begin{pmatrix} 0.00 & 0.76 & 0.02 & 0.04 & 0.03 & 0.03 & 0.04 & 0.03 & 0.03 & 0.04 \\ 0.02 & 0.00 & 0.73 & 0.05 & 0.05 & 0.04 & 0.04 & 0.02 & 0.02 & 0.05 \\ 0.03 & 0.03 & 0.00 & 0.70 & 0.02 & 0.02 & 0.05 & 0.02 & 0.03 & 0.09 \\ 0.02 & 0.07 & 0.03 & 0.00 & 0.59 & 0.10 & 0.03 & 0.02 & 0.13 & 0.03 \\ 0.04 & 0.03 & 0.06 & 0.03 & 0.00 & 0.73 & 0.03 & 0.02 & 0.03 & 0.04 \\ 0.02 & 0.02 & 0.04 & 0.02 & 0.03 & 0.00 & 0.79 & 0.04 & 0.04 & 0.02 \\ 0.12 & 0.02 & 0.03 & 0.02 & 0.03 & 0.02 & 0.00 & 0.69 & 0.03 & 0.02 \\ 0.03 & 0.05 & 0.02 & 0.02 & 0.14 & 0.03 & 0.03 & 0.00 & 0.66 & 0.03 \\ 0.03 & 0.03 & 0.02 & 0.05 & 0.05 & 0.06 & 0.03 & 0.02 & 0.00 & 0.71 \\ 0.69 & 0.03 & 0.02 & 0.03 & 0.02 & 0.05 & 0.03 & 0.09 & 0.03 & 0.00 \end{pmatrix}.$$

$$\hat{P}_4 = \begin{pmatrix} 0.00 & 0.82 & 0.01 & 0.03 & 0.02 & 0.02 & 0.03 & 0.02 & 0.02 & 0.03 \\ 0.01 & 0.00 & 0.80 & 0.03 & 0.03 & 0.03 & 0.03 & 0.01 & 0.01 & 0.04 \\ 0.02 & 0.02 & 0.00 & 0.79 & 0.02 & 0.01 & 0.03 & 0.02 & 0.02 & 0.07 \\ 0.01 & 0.04 & 0.02 & 0.00 & 0.73 & 0.06 & 0.02 & 0.01 & 0.09 & 0.02 \\ 0.03 & 0.02 & 0.04 & 0.02 & 0.00 & 0.81 & 0.02 & 0.01 & 0.02 & 0.03 \\ 0.01 & 0.01 & 0.03 & 0.01 & 0.02 & 0.00 & 0.84 & 0.03 & 0.03 & 0.01 \\ 0.09 & 0.02 & 0.02 & 0.01 & 0.02 & 0.01 & 0.00 & 0.78 & 0.02 & 0.02 \\ 0.02 & 0.03 & 0.01 & 0.02 & 0.09 & 0.02 & 0.02 & 0.00 & 0.76 & 0.02 \\ 0.02 & 0.02 & 0.02 & 0.03 & 0.03 & 0.05 & 0.02 & 0.01 & 0.00 & 0.79 \\ 0.78 & 0.02 & 0.02 & 0.02 & 0.02 & 0.03 & 0.02 & 0.06 & 0.02 & 0.00 \end{pmatrix}.$$

$$\hat{P}_5 = \begin{pmatrix} 0.00 & 0.86 & 0.01 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 & 0.02 \\ 0.01 & 0.00 & 0.85 & 0.03 & 0.03 & 0.02 & 0.02 & 0.01 & 0.01 & 0.03 \\ 0.02 & 0.02 & 0.00 & 0.84 & 0.01 & 0.01 & 0.03 & 0.01 & 0.02 & 0.05 \\ 0.01 & 0.03 & 0.01 & 0.00 & 0.80 & 0.05 & 0.01 & 0.01 & 0.06 & 0.01 \\ 0.02 & 0.02 & 0.03 & 0.02 & 0.00 & 0.85 & 0.02 & 0.01 & 0.02 & 0.02 \\ 0.01 & 0.01 & 0.02 & 0.01 & 0.02 & 0.00 & 0.88 & 0.02 & 0.02 & 0.01 \\ 0.06 & 0.01 & 0.02 & 0.01 & 0.02 & 0.01 & 0.00 & 0.84 & 0.02 & 0.01 \\ 0.02 & 0.02 & 0.01 & 0.01 & 0.07 & 0.02 & 0.02 & 0.00 & 0.82 & 0.02 \\ 0.02 & 0.02 & 0.01 & 0.02 & 0.02 & 0.03 & 0.02 & 0.01 & 0.00 & 0.84 \\ 0.84 & 0.02 & 0.01 & 0.02 & 0.01 & 0.03 & 0.02 & 0.05 & 0.02 & 0.00 \end{pmatrix}.$$

An illustration of the dynamics is given in figure 7.

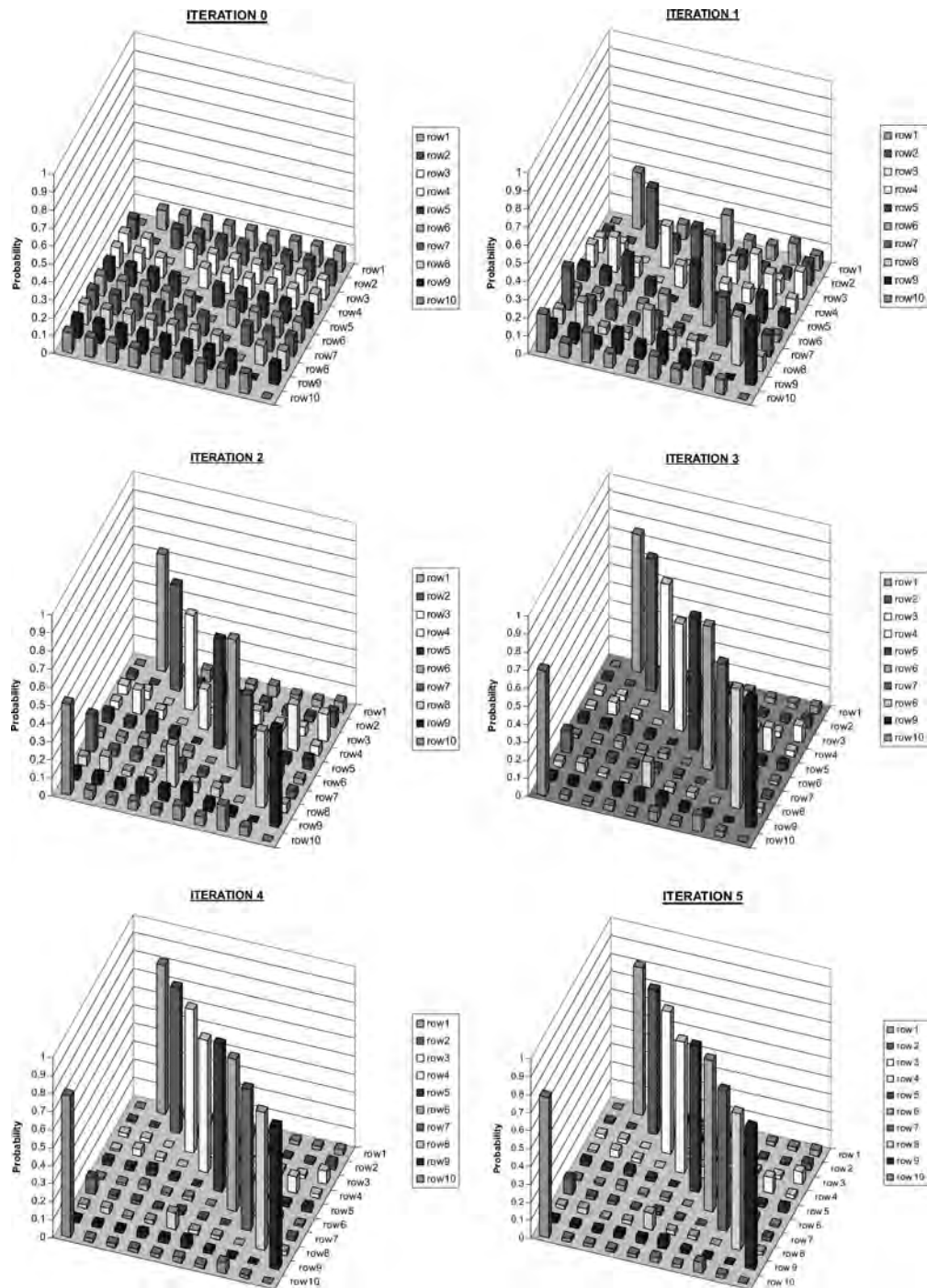


Figure 7. Convergence of the reference parameter (matrix) for a 10 node TSP.

4. Modifications

In this section we consider several modifications of the basic CE algorithm. We start by discussing alternative loss functions that favor samples with higher performance more than other samples. We continue with a fully automated version of the CE algorithm. This modified version allows automatic tuning of all the parameters of the CE algorithm.

4.1. Alternative performance functions

Consider the maximization problem, see (30),

$$\max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x}),$$

where $S(\mathbf{x})$ is some positive objective function defined on $\mathcal{X}$. In the associated stochastic problem, see (31), we consider instead estimating the rare-event probability

$$\ell(\gamma) = \mathbb{E}_{\mathbf{u}} I_{\{S(\mathbf{X}) \geq \gamma\}}.$$

Note that this can be written as

$$\ell(\gamma) = \mathbb{E}_{\mathbf{u}} \varphi(S(\mathbf{X}); \gamma),$$

where $\varphi(s; \gamma)$ is the indicator $I_{\{s \geq \gamma\}}$, that is,

$$\varphi(s; \gamma) = \begin{cases} 1 & \text{if } s \geq \gamma, \\ 0 & \text{if } s < \gamma, \end{cases} \quad (50)$$

for $s \geq 0$. Recall that the main CE Algorithm 2.3 contains the steps (a) updating $\hat{\gamma}_t$ via (20) and (b) updating $\hat{\mathbf{v}}_t$ via the (analytic) solution of (22). A natural modification of Algorithm 2.3 would be to update $\mathbf{v}_t$ using an alternative function $\varphi(s; \gamma)$. For a maximization problem such a function should be increasing in s for each fixed $\gamma \geq 0$, and decreasing in γ for each fixed $s \geq 0$. In particular one could use

$$\varphi(s; \gamma) = \psi(s) I_{\{s \geq \gamma\}},$$

for some increasing function $\psi(s)$.

Using such a $\varphi(s; \gamma)$ instead of the indicator (50) we now proceed similar as before. Specifically, the updating step (a) of $\hat{\gamma}_t$ remains *exactly the same*, and the updating step (b) of $\hat{\mathbf{v}}_t$ now reduces to the solution of the following program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^N I_{\{S(\mathbf{X}_i) \geq \hat{\gamma}_t\}} \psi(S(\mathbf{X}_i)) \ln f(\mathbf{X}_i; \mathbf{v}). \quad (51)$$

Numerical evidence suggests that $\psi(s) = s$ can lead to some speed-up of the algorithm. As an example, for the max-cut problem we obtain in this case (see (39)) the analytic updating formulas

$$\hat{p}_{t,i} = \frac{\sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geq \hat{\gamma}_t\}} S(\mathbf{X}_i) X_{ki}}{\sum_{k=1}^N I_{\{S(\mathbf{X}_k) \geq \hat{\gamma}_t\}} S(\mathbf{X}_i)}, \quad (52)$$

$i = 2, \dots, n$.

However, other numerical experiments suggest that high power polynomials, $\psi(s) = s^\beta$ with large β , and the Boltzmann function $\psi(s) = e^{-s/\beta}$ are not advisable, as they may lead more easily to local minima.

4.2. Fully adaptive CE algorithm

We present a modification of Algorithm 2.3, called the *fully automated CE* (FACE) algorithm in which the sample size is updated adaptively at each iterations t of the algorithm, i.e., $N = N_t$. In addition, this modification is able to identify some “difficult” and pathological problems.

Consider a *maximization* problem and let

$$S_{t,(1)} \leq \dots \leq S_{t,(N_t)}$$

denote the *ordered* sample performances of the N_t samples at the t -th iterations. To ease notations, we denote $S_{t,(N_t)}$ by S_t^* .

The main assumption in the FACE Algorithm is that at the end of each iteration t the updating of the parameters is done on the basis of a *fixed* number, N^{elite} say, of the best performing samples, the so-called *elite* samples. Thus, the set of elite samples $\mathcal{E}_t$ consist of those N^{elite} samples in $\{\mathbf{X}_1, \dots, \mathbf{X}_{N_t}\}$ for which the performances $S(\mathbf{X}_1), \dots, S(\mathbf{X}_{N_t})$ are highest. The updating steps 2 and 3 of Algorithm 2.3 are modified such that

$$\hat{\gamma}_t = S_{(N_t - N^{\text{elite}} + 1)},$$

and

$$\hat{\mathbf{v}}_t = \underset{\mathbf{v}}{\operatorname{argmax}} \sum_{\mathbf{X}_i \in \mathcal{E}_t} \ln f(\mathbf{X}_i; \mathbf{v}).$$

Note that $\hat{\gamma}_t$ is equal to the worst sample performance of the best N^{elite} sample performances, and S_t^* is the best of the elite performances (indeed, of all performances).

In the FACE algorithm the parameters ρ and N of Algorithm 2.3 are updated adaptively. Specifically, they are “replaced” by a single parameter: the number of elite samples N^{elite} . The above updating rules are consistent with Algorithm 2.3 provided we view ρ in Algorithm 2.3 as the parameter which changes inversely proportional to N_t : $\rho_t = N^{\text{elite}}/N_t$.

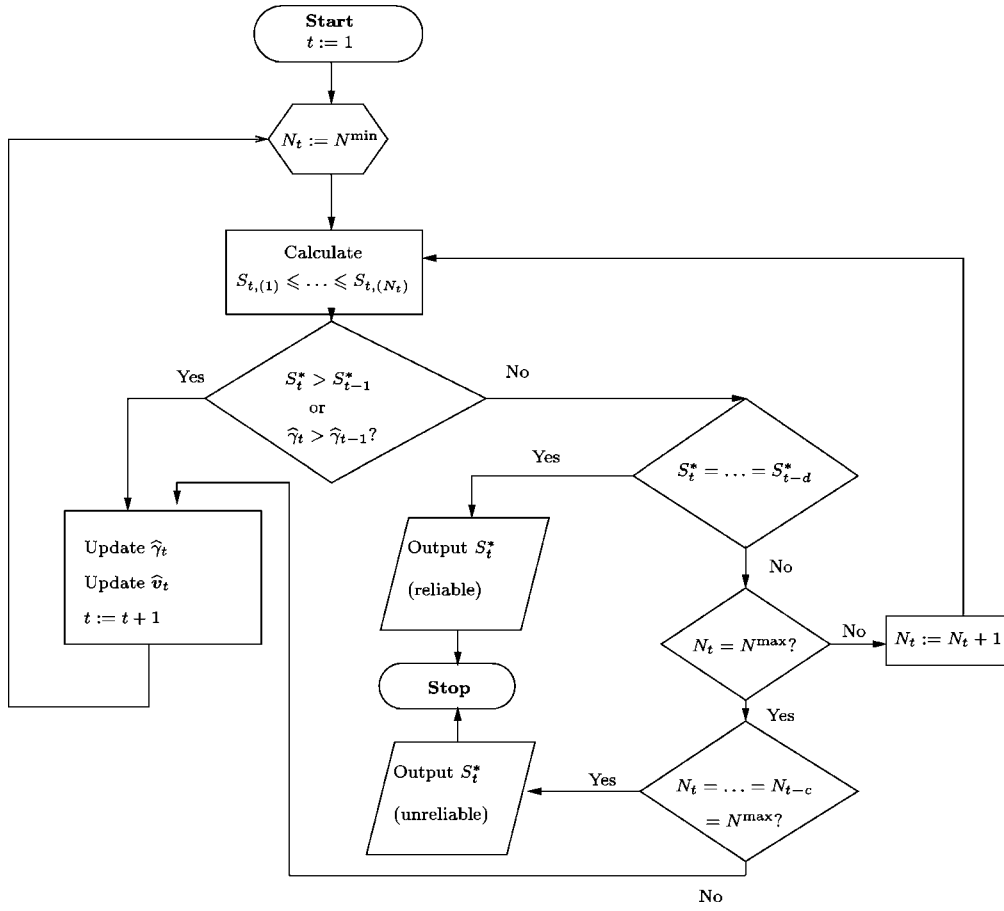


Figure 8. The flowchart for the FACE algorithm.

It was found experimentally that a sound choice is $N^{\text{elite}} = c_0 n$ and $N^{\text{elite}} = c_0 n^2$ for SNNs and SENs, respectively, where c_0 is a fixed positive constant (usually in the interval $0.01 \leq c_0 \leq 0.1$). The easiest way to explain FACE is via the flow chart in figure 8.

For each iteration t of FACE algorithm we design a sampling plan which ensures with high probability that

$$S_t^* > S_{t-1}^*. \quad (53)$$

Note that (53) implies *improvement* of the maximal order statistics (best elite performance) at each iteration. To ensure (53) with high probability, we allow N_t to vary at

each iteration t in a quite wide range

$$N^{\min} \leq N_t \leq N^{\max},$$

where, say, for an SNN-type problem, $N^{\min} = n$ and $N^{\max} = 20n$. Note that we always start each iteration by generating $N^{\min}$ samples. If at any iteration t , while increasing N_t we obtain simultaneously that $N_t = N^{\max}$ and (53) is violated, then we directly proceed with updating $(\hat{\gamma}_t, \hat{\mathbf{v}}_t)$ according to Algorithm 2.3 before proceeding with the next iteration of the FACE Algorithm. However, if FACE keeps generating samples of size $N^{\max}$ for several iterations in turn, say, for $c = 3$ iterations and (53) is violated then we stop, and announce that FACE identified a “hard” problem for which the estimate of the optimal solution is *unreliable*.

Besides if $S_{t,(N_t)} > S_{t-1,(N_{t-1})}$, we also continue with the next iteration if

$$\hat{\gamma}_t > \hat{\gamma}_{t-1}. \quad (54)$$

Note that (54) implies *improvement* of the worst elite sample performance $\hat{\gamma}_t$ at each iteration.

Similar to Algorithm 2.3 we initialize by choosing some $\hat{\mathbf{v}}_0$. For example, for the max-cut problem we choose (using $\mathbf{p}$ instead of $\mathbf{v}$) $\hat{\mathbf{p}}_0 = \mathbf{p}_0 = (1, 1/2, \dots, 1/2)$. We assume that the FACE parameters $N^{\min}$, $N^{\max}$, α and the stopping constant d are chosen in advance. For example, for the max-cut problem we take $N^{\min} = n$. We let $t = 1$ and $N_1 = N^{\min}$ and proceed as follows:

Algorithm 4.1 (FACE Algorithm)

1. At iteration t , $t = 1, 2, \dots$ take an initial sample of size N_t , with $N^{\min} \leq N_t \leq N^{\max}$ from $f(\cdot; \hat{\mathbf{v}}_{t-1})$.
2. If (53) or (54) holds, proceed with the updating steps (20) and (22) using the N_t samples in Step 1.
3. If (53) and (54) are violated, check whether or not

$$S_t^* = \dots = S_{t-d}^*. \quad (55)$$

If so, stop and deliver S_t^* as an estimate of the optimal solution. Call such S_t^* a **reliable** estimate of the optimal solution. If (55) does not hold, and if $N_t < N^{\max}$, increase N_t by 1, recalculate S_t^* and $\hat{\gamma}_t$, and go to Step 2.

4. If $N_t = N^{\max}$ and each of (53), (54) and (55) is violated, proceed with (20) and (22) using the $N^{\max}$ samples mentioned in Step 1 and go to step 2.
5. If $N_t = N^{\max}$ for several iterations in turn, say for $c = 3$ iterations, and each of (53), (54) and (55) are violated, stop and announce that FACE identified a “hard” problem. Call $S_{t,(N_t)}$ an **unreliable** estimate of the optimal solution.

The stopping criterion (55) means that the best samples in the last d iterations are the same. Note that if (53) holds for *all* $t \geq 1$ we automatically obtain that $N_t = N^{\min}$ for all t . In such case FACE reduces to the original Algorithm 2.3. We found that FACE typically speeds up the convergence up to 2 times as compared to CE with fixed N and ρ .

5. Further developments

In this section we present two additional applications of the CE method. In Section 5.1 we consider the problem of vector quantization and clustering. We describe a straightforward application of the CE method to this problem, and demonstrate high performance of the CE method. In Section 5.2 we consider the problem of learning the optimal policy in Markovian decision processes.

5.1. Vector quantization and clustering analysis

The clustering problem reads as follows: given a dataset $\mathcal{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_n\}$ of points in some d -dimensional Euclidean region, partition the data into K “clusters” $R_1, \dots, R_K$ (with $R_i \cap R_j = \emptyset$, for $i \neq j$, and $\cup_j R_j = \mathcal{Z}$), such that some empirical loss function is minimized. A typical loss function is (Webb, 1999):

$$\sum_{j=1}^K \sum_{\mathbf{z} \in R_j} \|\mathbf{z} - \mathbf{c}_j\|^2, \quad (56)$$

where

$$\mathbf{c}_j = \frac{1}{|R_j|} \sum_{\mathbf{z} \in R_j} \mathbf{z} \quad (57)$$

presents the *centroid* of the points of cluster R_j . Denoting by $\mathbf{x} = (x_1, \dots, x_n)$ the vector with $x_i = j$ when $\mathbf{z}_i \in R_j$, and letting $\mathbf{z}_{ij} = I_{\{x_i=j\}} \mathbf{z}_i$, we can write (56) as the following function of $\mathbf{x}$:

$$S(\mathbf{x}) = \sum_{j=1}^K \sum_{i=1}^n I_{\{x_i=j\}} \|\mathbf{z}_{ij} - \mathbf{c}_j\|^2, \quad (58)$$

where the centroids are given as

$$\mathbf{c}_j = \frac{1}{n_j} \sum_{i=1}^n \mathbf{z}_{ij},$$

with $n_j = \sum_{i=1}^n I_{\{x_i=j\}}$ the number of points in the j th cluster.

As we mentioned, our goal is to find a partition $\{R_j\}$ and a corresponding vector of centroids $(\mathbf{c}_1, \dots, \mathbf{c}_K)$ that minimizes (56). In the terminology of *vector quantization*

(Webb, 1999), we wish to “quantize” or “encode” the vectors in $\mathcal{Z}$ in such a way that each vector is represented by one of K *source vectors* $\mathbf{c}_1, \dots, \mathbf{c}_K$, such that the loss (56) of this representation is minimized. Most well-known clustering and vector quantization methods update the vector of centroids, starting from some initial choice $(\mathbf{c}_{0,1}, \dots, \mathbf{c}_{0,K})$ and using iterative (typically gradient-based procedures). It is important to realize that in that case (56) is seen as a function of the *centroids*, where each point $\mathbf{z}$ is assigned to the nearest centroid, thus determining the clusters. It is well known these type of problems (optimization with respect to the centroids) are multi-extremal and depending on the initial value of the clusters the gradient-based procedures converge to a *local minimum* rather than global minimum. A standard heuristic to minimize (57) is the *K-means algorithm* (Webb, 1999).

We can optimize (58) via the CE method, by viewing it as a minimal cut (min-cut) problem with K partitions. In particular, each partition $R_1, \dots, R_K$ is represented via a partition vector $\mathbf{x} \in \mathcal{X} = \{1, \dots, n\}^K$ as defined above. The “trajectory generation” of the CE Algorithm 2.3 consists of drawing random $\mathbf{X} \in \mathcal{X}$ according to an n -dimensional discrete distribution with independent marginals, such that $\mathbb{P}(X_i = j) = p_{ij}$, $i = 1, \dots, m$, $j = 1, \dots, K$. The updating rules are again very simple. For $K = 2$ we may, alternatively, use $\mathbf{X} \sim \text{Ber}(\mathbf{p})$. In that case the updating rules are given in (39), with $\geq$ replaced with $\leq$.

Example 5.1. Let $n = 5$ and $K = 2$. To generate a feasible cluster we draw $\mathbf{X}$ from a 5-dimensional Bernoulli distribution $\text{Ber}(\mathbf{p})$ with independent marginals. Assume that the outcome from a particular generation from $\text{Ber}(\mathbf{p})$ is $\mathbf{x} = (1, 0, 0, 1, 0)$. The associated partition is, clearly, $\{R_1, R_2\} = \{\{1, 4\}, \{2, 3, 5\}\}$. In this case the loss and the centroids can be explicitly calculated, provided the points $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4, \mathbf{z}_5$ are given. Namely, the loss is

$$(\|\mathbf{z}_1 - \mathbf{c}_1\|^2 + \|\mathbf{z}_4 - \mathbf{c}_1\|^2) + (\|\mathbf{z}_2 - \mathbf{c}_2\|^2 + \|\mathbf{z}_3 - \mathbf{c}_2\|^2 + \|\mathbf{z}_5 - \mathbf{c}_2\|^2), \quad (59)$$

and the centroids are

$$\mathbf{c}_1 = \frac{1}{2}(\mathbf{z}_1 + \mathbf{z}_4), \quad \mathbf{c}_2 = \frac{1}{3}(\mathbf{z}_2 + \mathbf{z}_3 + \mathbf{z}_5), \quad (60)$$

respectively.

The CE method for clustering works well even if the dataset is noisy. This is in contrast to most clustering methods that often require stochastic approximation procedures, which are very slow.

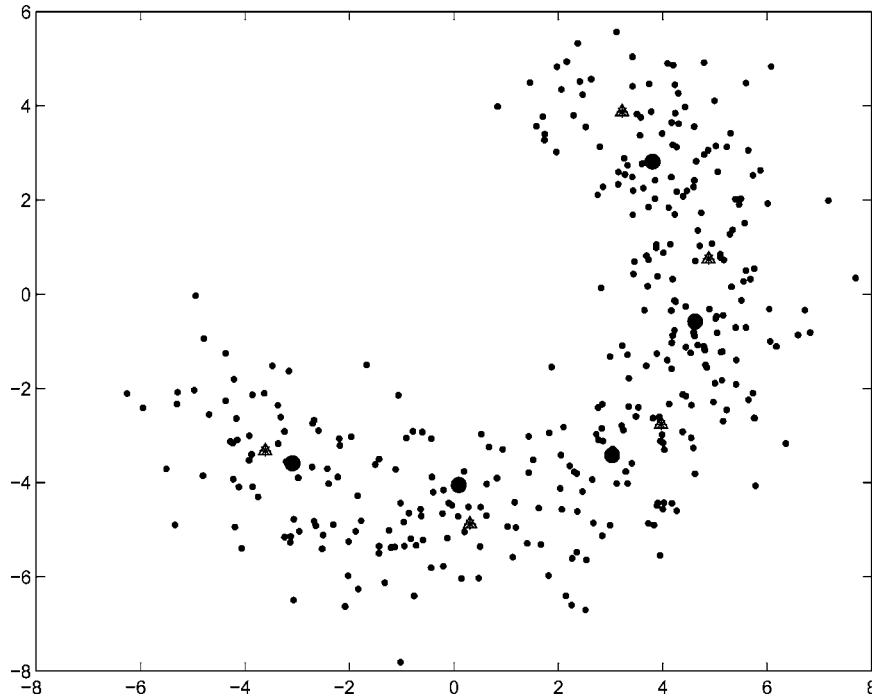


Figure 9. The CE results for vector quantization of the 2-D *banana* data set. Circles designate the final cluster centers produced by CE. Triangles are cluster centers of the *K*-means algorithm.

Numerical example

Consider as an application of CE the minimization of the loss function, given by (56) for a simple 2-dimensional clustering data set with $m = 5$ clusters and 700 points (depicted by the small dots in figure 9). The resulting locations of the cluster centers obtained by the CE Algorithm are depicted in figure 9 by the large filled circles. Figure 9 also compares the performance of the CE method with the standard *K*-means one (circles and triangles in figure 9 correspond to the final cluster centers produced by CE and the *K*-means algorithm, respectively). Although we found that *K*-means is a little faster (it takes approximately 2 seconds for *K*-means versus 49 for CE), it readily follows from figure 9 that CE is more accurate than *K*-means, (the average distortion is 1.428 and 2.408 for CE and *K*-means, respectively).

5.2. Markovian decision process

The *Markovian decision process* (MDP) model is standard in artificial intelligence, machine learning, operation research and related fields. We review briefly some basic

definitions and concepts related to the MDP model. For details see for example Puterman (1994) and Bertsekas (1995).

An MDP is defined by a tuple $(\mathcal{Z}, \mathcal{A}, \mathcal{P}, r)$ where

1. $\mathcal{Z} = \{1, \dots, n\}$ is a finite set of states.
2. $\mathcal{A} = \{1, \dots, m\}$ is the set of possible actions by the decision maker. We assume it is the same for every state – to ease notations.
3. $\mathcal{P}$ is the transition probability matrix with elements $\mathcal{P}(z' | z, a)$ presenting the transition probability from state z to state z' , when action a is chosen.
4. $r(z, a)$ is the reward for performing action a in state z (r may be a random variable).

At each time instance k the decision maker observes the current state z_k , and determines the action to be taken (say a_k). As a result, a reward given by $r(z_k, a_k)$, denoted by r_k , is received and a new state z' is chosen according to $\mathcal{P}(z' | z_k, a_k)$. A *policy* π is a rule that determines, for each history $H_k = z_1, a_1, \dots, z_{k-1}, a_{k-1}, z_k$ of states and actions, the probability distribution of the decision maker's actions at time k . A policy is called *Markov* if each action depends only on the current state z_k and time k . A Markov policy is called *stationary* if it does not depend on the time k . The goal of the decision maker is to maximize a certain reward as a function of the policy. There are several reward criteria of interest, see Puterman (1994) for a discussion. In this exposition we consider the *total reward* criterion which applies when there exists a finite stopping time τ at which the process terminates. The objective is to maximize the total reward

$$V(\pi, z_0) = \mathbb{E}_\pi \sum_{k=0}^{\tau-1} r_k, \quad (61)$$

starting from some fixed state z_0 . Here $\mathbb{E}_\pi$ denotes the expectation with respect to some probability measure induced by the policy π . We will restrict our attention to *stochastic shortest path* MDPs, where it is assumed that the process starts from a specific initial state $z_0 = z^{\text{start}}$, and that there is a absorbing state z^{fin} with zero reward. The objective is given in (61), with τ being the stopping time at which z^{fin} is reached (which we will assume will always happen eventually). For stochastic shortest path MDPs, it is a well known (Puterman, 1994) fact that there exists a stationary Markov policy which maximizes $V(\pi, z_0)$.

If the model (r and $\mathcal{P}$) is known, then there are several efficient methods, such as *value iteration* and *policy iteration*, for finding the optimal policy (Puterman, 1994). However, when the transition probability or the reward in an MDP are *unknown*, the problem is much more difficult, and is referred to as a *learning* one. A well-known framework for learning algorithms is *reinforcement learning* (RL), where an agent learns the behavior of the system through trial-and-error with an unknown dynamic environment, see Kaelbling, Littman, and Moore (1996). For reviews of RL see Barto and Sutton

(1998), Bertsekas and Tsitsiklis (1995), Kaelbling, Littman, and Moore (1996). Note that many of learning algorithm for MDPs rely on the Stochastic Approximation Algorithm and convergence is typically slow. Using the CE method to search for the optimal policy is similar to certain approaches in RL that attempt to search the policy space (or a subset thereof). Notable policy search algorithms used in RL are: a direct search in the policy space of Rosenstein and Barto (2001); a policy gradient search approach of Baxter, Bartlett, and Weaver (2001); and the actor critic framework of Barto, Sutton, and Anderson (1983). See also Sutton et al. (2000), and Konda and Tsitsiklis (2003).

Policy learning via the CE method

Since for the shortest path MDP an optimal stationary policy exists, we can represent each stationary policy as a vector $\mathbf{x} = (x_1, \dots, x_n)$ with $x_i \in \{1, \dots, m\}$ being the action taken when visiting state i . Writing the expectation in 61 as

$$S(\mathbf{x}) = \mathbb{E}_{\mathbf{x}} \sum_{k=0}^{\tau-1} r(Z_k, A_k), \quad (62)$$

where $Z_0, Z_1, \dots$ are the states visited, and $A_0, A_1, \dots$ the actions taken, we see that the optimization problem 61 is of the form 30. From a CE point of view we consider the maximization problem 62 as a *noisy* maximization problem. The idea now is to combine the random policy generation and the random trajectory generation in the following way: At each stage of the CE algorithm we generate random policies and random trajectories using an auxiliary $n \times m$ matrix $P = (p_{za})$, such that for each state z we choose action a with probability p_{za} . Once this “policy matrix” P is defined, each iteration of the CE algorithm comprises the following two standard phases:

1. Generation of N random trajectories $(Z_0, A_0, Z_1, A_1, \dots, Z_\tau, A_\tau)$ using the auxiliary policy matrix P . The cost of each trajectory is computed via

$$\hat{S}(\mathbf{X}) = \sum_{j=0}^{\tau-1} r(Z_j, A_j). \quad (63)$$

2. Updating of the parameters of the policy matrix (p_{za}) on the basis of the data collected at the first phase.

The matrix P is typically initialized to a uniform matrix ($p_{ij} = 1/m$). Generation of random trajectories for MDP is straightforward and is given for completeness. All one has to do is to start the trajectory from the initial state $z_0 = z^{\text{start}}$ and follow the trajectory by generating at each new state according to the probability distribution of P , until the absorbing state z^{fin} is reached at time τ , say.

Algorithm 5.1 (Trajectory generation for MDP)*Input: P auxiliary policy matrix.*

1. Start from the given initial state $Z_0 = z^{\text{start}}$, set $k = 0$.
2. Generate an action A_k according to the Z_k th row of P , calculate the reward $r_k = r(Z_k, A_k)$ and generate a new state Z_{k+1} according to $\mathcal{P}(\cdot | Z_k, A_k)$. Set $k = k + 1$. Repeat until $z_k = z^{\text{fin}}$.
3. Output the total reward of the trajectory $(Z_0, A_0, Z_1, A_1, \dots, Z_\tau)$, given by (63).

Given the N trajectories $\mathbf{X}_1, \dots, \mathbf{X}_N$ and their scores, $\hat{S}(\mathbf{X}_1), \dots, \hat{S}(\mathbf{X}_N)$, one can update the parameter matrix (p_{za}) using the CE method, namely as per

$$\hat{p}_{t,za} = \frac{\sum_{k=1}^N I_{\{\hat{S}(\mathbf{X}_k) \geq \hat{\gamma}_t\}} I_{\{\mathbf{X}_k \in \mathcal{X}_{za}\}}}{\sum_{k=1}^N I_{\{\hat{S}(\mathbf{X}_k) \geq \hat{\gamma}_t\}} I_{\{\mathbf{X}_k \in \mathcal{X}_z\}}}, \quad (64)$$

where the event $\{\mathbf{X}_k \in \mathcal{X}_z\}$ means that the trajectory $\mathbf{X}_k$ contains a visit to state z and the event $\{\mathbf{X}_k \in \mathcal{X}_{za}\}$ means the trajectory corresponding to policy $\mathbf{X}_k$ contains a visit to state z in which action a was taken.

We now explain how to take advantage of the Markovian nature of the problem. Let us think of a maze where a certain trajectory starts badly, that is, the path is not efficient in the beginning, but after some time it starts moving quickly towards the goal. According to (64), all the updates are performed in a similar manner in every state in the trajectory. However, the actions taken in the states that were sampled near the target were successful, so one would like to “encourage” these actions. Using the Markovian property one can substantially improve the above algorithm by considering for each state the part of the reward from the visit to that state onwards. We therefore use the same trajectory and simultaneously calculate the performance for every state in the trajectory separately. The idea is that each choice of action in a given state affects the reward from that point on, disregarding the past.

The sampling Algorithm 5.1 does not change in Steps 1 and 2. The difference is in Step 3. Given a policy $\mathbf{X}$ and a trajectory $(Z_0, A_0, Z_1, A_1, \dots, Z_\tau, A_\tau)$ we calculate the performance from every state until termination. For every state $z = Z_j$ in the trajectory the (estimated) performance is $\hat{S}_z(\mathbf{X}) = \sum_{k=j}^{\tau-1} r_k$. The updating formula for $\hat{p}_{za}$ is similar to (64), however *each state z is updated separately* according to the (estimated) performance $\hat{S}_z(\mathbf{X})$ obtained from state z onwards.

$$\hat{p}_{t,za} = \frac{\sum_{k=1}^N I_{\{\hat{S}_z(\mathbf{X}_k) \geq \hat{\gamma}_{t,z}\}} I_{\{\mathbf{X}_k \in \mathcal{X}_{za}\}}}{\sum_{k=1}^N I_{\{\hat{S}_z(\mathbf{X}_k) \geq \hat{\gamma}_{t,z}\}} I_{\{\mathbf{X}_k \in \mathcal{X}_z\}}}. \quad (65)$$

A crucial point here is to understand that in contrast to (64) the CE optimization is carried for every state separately and a different threshold parameter $\hat{\gamma}_{t,z}$ is used for every state z , at iteration t . This facilitates faster convergence for “easy” states where the optimal

policy is easy to find. Numerical results indicate that the CE algorithm with updating (65) is much faster than that with updating (64).

Numerical results

The CE method with the updating rule (65) and trajectory generation according to Algorithm 5.1 was implemented for a maze problem, which presents a two-dimensional grid world. We assume that

1. The moves in the grid are allowed in four possible directions with the goal to move from the upper-left corner to the lower-right corner.
2. The maze contains obstacles (“walls”) into which movement is not allowed.
3. The reward for every allowed movement until reaching the goal is -1 .

In addition we introduce:

1. A small (failure) probability not to succeed moving in an allowed direction.
2. A small probability of succeeding moving in the forbidden direction (“moving through the wall”).
3. A high cost for the moves in a forbidden direction.

In figure 10 we present the results for 20×20 maze. We set the following parameters: $N = 1000$, $\rho = 0.03$, $\alpha = 0.7$. The initial policy was a uniformly random one. The cost of the moves were assumed to be random variables uniformly distributed between 0.5 and 1.5 and uniformly distributed between 25 and 75 for the allowed and forbidden moves, respectively. Note that the expected cost for the allowed and forbidden moves are equal to 1 and 50, respectively. The success probabilities in the allowed and forbidden states were taken 0.95 and 0.05, respectively. The arrows $z \rightarrow z'$ in figure 10 indicate that at the current iteration the probability of going from z to z' is at least 0.01. In other words, if a corresponds to the action that will lead to state z' from z then we will plot an arrow from z to z' provided $p_{za} > 0.01$.

In all our experiments CE found the target exactly, within 5–10 iterations and CPU time was less than one minute (on a 500 MHz Pentium processor). Note that the successive iterations of the policy in figure 10 quickly converge to the optimal policy.

Additional numerical results can be found in Dubin (2002). Extensions, other reward criteria, and further applications appear in Mannor, Rubinstein, and Gat (2003). These results demonstrate that the CE algorithm may serve as a policy learning algorithm. In contrast to the popular Q-learning (e.g. Dayan and Watkins, 1992) and value iteration (e.g. Bertsekas and Tsitsiklis, 1995) based algorithms, no information is propagated back, but rather the policy is learned directly.

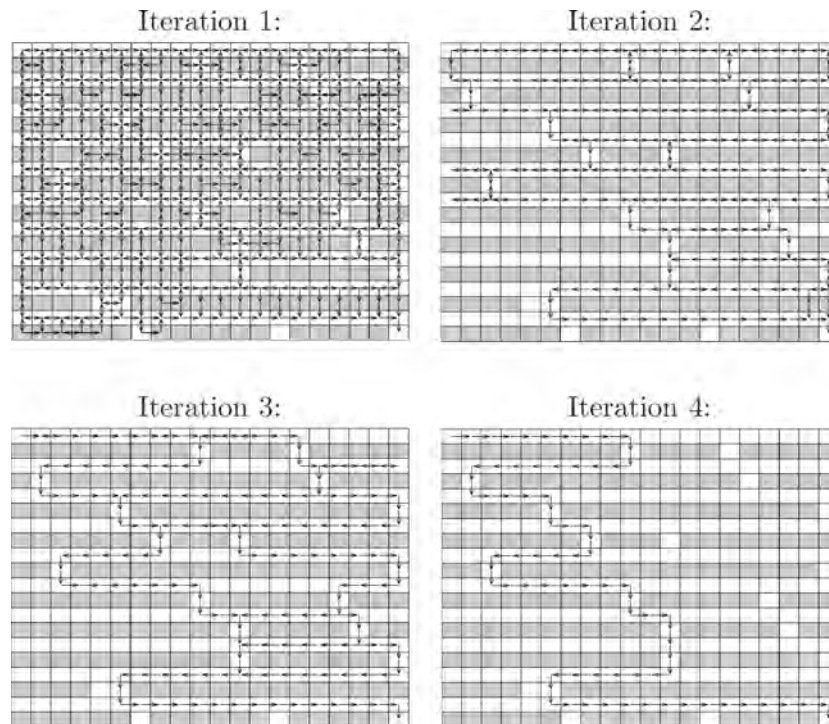


Figure 10. Performance of the CE Algorithm for the 20×20 maze. Each arrow indicates a probability > 0.01 of going in that direction.

6. Discussion and future directions

In this tutorial we introduced the CE method, a new generic approach to efficiently solve difficult deterministic and stochastic optimization problems (both combinatorial and continuous multi-extremal), and to adaptively find efficient estimators for rare event probabilities. We presented the CE methodology, the basic algorithm and its modifications, and discussed its applications to several combinatorial optimization and machine learning problems. We showed that the CE method is a simple, versatile and easy usable tool. Another virtue of the CE method is its robustness with respect to its parameters. In most cases, there is a wide range of parameters N , ρ , α and $\mathbf{p}$, P that leads to the desired solution with high probability, although sometimes choosing an appropriate parametrization and sampling scheme is more of an art than a science. Moreover, the foundations of the CE method are based on some well known classical probabilistic and statistical principles, and are therefore fairly problem-independent.

Although, as mentioned, it is out of the scope of this tutorial to give a detailed comparison between the CE method and other heuristics we shall point out a few similarities and differences. The main difference between the CE method and simulating annealing

(Aarts and Korst, 1989) is that the latter can be viewed as a local search algorithm whereas the CE is a global search one, and thus, it is less likely – at least in principle – to get stuck in a local optimal solution. The CE method and genetic algorithms (Goldberg, 1989) share the common idea of picking samples at random and improve the way the samples are generated from generation to generation. In both methods, the solutions fitness is estimated according to a score/performance function. When running genetic algorithms the genetic encoding of the solution is problem-specific and often requires significant effort. In addition, there are many parameters that appear to be critical - crossover and mutation probabilities and sample size. The way the CE method chooses the next sample is less ambiguous, as the change of population generation parameter is optimal in “Kullback-Leiber distance” sense. Also the ant colony optimization (ACO) method (Dorigo, Di Caro, and Gambardella, 1999) bears similarities with the CE method, especially for SEN. The difference is the way the next generation of solutions is generated. In ACO good solutions in the previous generation increase the probability that solutions in subsequent generations follow a similar path. The generation of future solutions is not based on a principled calculation, but rather on problem dependent heuristics. We note that methods in the spirit of tabu search (Glover and Laguna, 1993) and guided local search (Voudouris, 2003) that penalize the neighborhood of previously examined solutions, use a completely different mechanism. Finally, we do not claim that the CE method always finds the optimal solution. Although the numerical results with CE are typically quite reliable, there is no guarantee that it will not get trapped in a local minimum, at times.

There are several research topics that call for further development. First, the resilience of the CE method to local minima has been observed in many applications but is still not well understood. It seems that the CE method tends to overcome local minima based on both the randomization mechanism involving the elite sampling and the smoothed updating.

Second, the convergence of the CE method deserves further study. Convergence results concerning the CE method can be found in several sources. Typically, these are asymptotic in nature, similar to existing convergence results for genetic algorithms, simulated annealing and ant colony algorithms. As a result they do not always shed much light on the actual (non-asymptotic) behavior of the algorithm. In the context of rare event simulation, which can be readily adopted to combinatorial optimization, convergence of CE is given in Homem-de-Mello and Rubinstein (2002). The basic assumption there is that the probability of a rare event does not vanish in a neighborhood of the optimal solution. The algorithm proposed in Homem-de-Mello and Rubinstein (2002) is the same as Algorithm 2.1, only that ρ and N are determined adaptively. Another convergence theorem in the rare event setting is given in Lieber (1998). Margolin (2005) provides asymptotical convergence results of two modifications of the main CE Algorithm 2.3. The proof is similar to that of Gutjahr (2000) for the ant colony optimization. This asymptotic result provides no guidance on convergence rates. Much more on the convergence of CE should be done, in particular finding bounds of convergence rates, at least for some particular difficult optimization problems.

Another research issue is to develop and investigate some alternative stopping rules. The current heuristic rule only indicates the lack of improvement in the past few iterations and when the CE method stops it may happen that it is not even a local optimum. Finally, we consider application of the CE method to constrained optimization problems, and in particular to constrained integer programs, as a challenging topic which would be worth exploring.

For further reading we refer to the monograph (Rubinstein and Kroese, 2004).

Acknowledgments

We would like to thank the associate editor and the two anonymous referees for their detailed and most valuable comments. The research of R.Y. Rubinstein was supported by the Israel Science Foundation (grant No. 191-565).

References

- Aarts, E.H.L. and J.H.M. Korst. (1989). *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons.
- Alon, G., D.P. Kroese, T. Raviv, and R.Y. Rubinstein. (2005). "Application of the Cross-Entropy Method to the Buffer Allocation Problem in a Simulation-Based Environment." *Annals of Operations Research* 134, 19–67.
- Asmussen, S., D.P. Kroese, and R.Y. Rubinstein. (2005). "Heavy Tails, Importance Sampling and Cross-Entropy." *Stochastic Models* 21(1). To appear.
- Barto, A., R. Sutton, and C. Anderson. (1983). "Neuron-Like Adaptive Elements that can Solve Difficult Learning Control Problems." *IEEE Transactions on Systems, Man, and Cybernetics* 13, 834–846.
- Barto, A.G. and R.S. Sutton. (1998). *Reinforcement Learning*. MIT Press.
- Baxter, J., P.L. Bartlett, and L. Weaver. (2001). Experiments with Infinite-Horizon, Policy-Gradient Estimation." *Journal of Artificial Intelligence Research* 15, 351–381.
- Bertsekas, D.P. (1995). *Dynamic Programming and Optimal Control*. Athena Scientific.
- Bertsekas, D.P. and J.N. Tsitsiklis. (1995). *Neuro-Dynamic Programming*. Athena Scientific.
- Chepuri, K. and T. Homem-de-Mello. (2005). "Solving the Vehicle Routing Problem with Stochastic Demands using the Cross-Entropy Method." *Annals of Operations Research* 134, 153–181.
- Cohen, I., B. Golany, and A. Shtub. (2005). "Managing Stochastic Finite Capacity Multi-Project Systems Through the Cross-Entropy Method." *Annals of Operations Research* 134, 183–199.
- Coloni, A., M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini, and M. Trubian. (1996). "Heuristics from Nature for Hard Combinatorial Problems." *International Transactions in Operational Research* 3(1), 1–21.
- Dayan, P. and C. Watkins (1992). "Q-Learning." *Machine Learning* 8, 279–292.
- de Boer, P.T. (2000). "Analysis and Efficient Simulation of Queueing Models of Telecommunication Systems." Ph.D. thesis, University of Twente.
- de Boer, P.T., D.P. Kroese, and R.Y. Rubinstein. (2002). "Estimating Buffer Overflows in Three Stages using Cross-Entropy." In *Proceedings of the 2002 Winter Simulation Conference*. San Diego, pp. 301–309.
- de Boer, P.T., D.P. Kroese, and R.Y. Rubinstein. (2004). "A Fast Cross-Entropy Method for Estimating Buffer Overflows in Queueing Networks." *Management Science* 50(7), 883–895.
- Dorigo, M., G. Di Caro, and L.M. Gambardella. (1999). "Ant Algorithms for Discrete Optimization." *Artificial Life* 5(2), 137–172.
- Dubin, U. (2002). "Application of the Cross-Entropy Method to Neural Computation." Master's thesis, Technion, Electrical Engineering.

- Dubin, U. (2004). "Application of the Cross-Entropy Method for Image Segmentation." Unpublished.
- Garey, M.R. and D.S. Johnson. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and Company.
- Glover, F. and M. Laguna. (1993). *Modern Heuristic Techniques for Combinatorial Optimization*, chapter 3: Tabu search. Blackwell Scientific Publications.
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley.
- Gutjahr, W.J. (2000). "A Graph-Based Ant System and Its Convergence." *Future Generations Computing* 16, 873–888.
- Helvik, B.E. and O. Wittner. (2001). "Using the Cross-Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks." In *3rd International Workshop on Mobile Agents for Telecommunication Applications – MATA'01*.
- Homem-de-Mello, T. and R.Y. Rubinstein. (2002). "Rare Event Estimation for Static Models via Cross-Entropy and Importance Sampling." (submitted for publication).
- Hui, K.-P., N. Bean, M. Kraetzl, and D.P. Kroese. (2005). "The Cross-Entropy Method for Network Reliability Estimation." *Annals of Operations Research* 134, 101–118.
- Kaelbling, L.P., M. Littman, and A.W. Moore. (1996). "Reinforcement Learning – A Survey." *Journal of Artificial Intelligence Research* 4, 237–285.
- Keith, J. and D.P. Kroese. (2002). "SABRES: Sequence Alignment By Rare Event Simulation." In *Proceedings of the 2002 Winter Simulation Conference*, San Diego, pp. 320–327.
- Konda, V.R. and J.N. Tsitsiklis. (2003). "Actor-Critic Algorithms." *SIAM Journal on Control and Optimization* 42(4), 1134–1166.
- Kroese, D.P. and R.Y. Rubinstein. (2004). "The Transform Likelihood Ratio Method for Rare Event Simulation with Heavy Tails." *Queueing Systems* 46, 317–351.
- Lieber, D. (1998). "Rare-Events Estimation via Cross-Entropy and Importance Sampling." Ph.D. thesis, William Davidson Faculty of Industrial Engineering and Management, Technion, Haifa, Israel.
- Mannor, S., R.Y. Rubinstein, and Y. Gat. (2003). "The Cross-Entropy Method for Fast Policy Search." In *Proceedings of the Twentieth International Conference on Machine Learning*. Morgan Kaufmann, pp. 512–519.
- Margolin, L. (2002). "Application of the Cross-Entropy Method to Scheduling Problems." Master's thesis, Technion, Industrial Engineering.
- Margolin, L. (2004). "The Cross-Entropy Method for the Single Machine Total Weighted Tardiness Problem." Unpublished.
- Margolin, L. (2005). "On the Convergence of the Cross-Entropy Method." *Annals of Operations Research* 134, 201–214.
- Menache, I., S. Mannor, and N. Shimkin. (2005). "Basis Function Adaption in Temporal Difference Reinforcement Learning." *Annals of Operations Research* 134, 215–238.
- Papadimitriou, C.H. and M. Yannakakis. (1991). "Optimization, Approximation, and Complexity Classes." *J. Comput. System Sci.* 43, 425–440.
- Puterman, M. (1994). *Markov Decision Processes*. Wiley-Interscience.
- Ridder, A. (2005). "Importance Sampling Simulations of Markovian Reliability Systems Using Cross-Entropy." *Annals of Operations Research* 134, 119–136.
- Rosenstein, M.T. and A.G. Barto. (2001). "Robot Weightlifting by Direct Policy Search." In B. Nebel (ed.). *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, pp. 839–846.
- Rubinstein, R.Y. (1997). "Optimization of Computer Simulation Models with Rare Events." *European Journal of Operations Research* 99, 89–112.
- Rubinstein, R.Y. (1999). "The Simulated Entropy Method for Combinatorial and Continuous Optimization." *Methodology and Computing in Applied Probability* 2, 127–190.
- Rubinstein, R.Y. (2001). "Combinatorial Optimization, Cross-Entropy, Ants and Rare Events." In S. Uryasev and P.M. Pardalos (eds.). *Stochastic Optimization: Algorithms and Applications*. Kluwer, pp. 304–358.

- Rubinstein, R.Y. (2002). "Cross-Entropy and Rare-Events for Maximal Cut and Bipartition Problems." *ACM Transactions on Modeling and Computer Simulation*, 27–53.
- Rubinstein, R.Y. and D.P. Kroese. (2004). *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer-Verlag, New York.
- Rubinstein, R.Y. and B. Melamed. (1998). *Modern Simulation and Modeling*. Wiley series in probability and Statistics.
- Rubinstein, R.Y. and A. Shapiro. (1993). *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization via the Score Function Method*. Wiley.
- Shi, L. and S. Olafsson. (2000). "Nested Partitioning Method for Global Optimization." *Operations Research* 48(3), 390–407.
- Sutton, R.S., D. McAllester, S. Singh, and Y. Mansour. (2000). "Policy Gradient Methods for Reinforcement Learning with Function Approximation." In *Advances in Neural Information Processing Systems 12*. MIT Press, pp. 1057–1063.
- Voudouris, C. (2003). "Guided Local Search – An Illustrative Example in Function Optimisation." *BT Technology Journal* 16(3), 46–50.
- Webb, A. (1999). *Statistical Pattern Recognition*. Arnold.

APPENDIX DZEROSKI01

```
leader 00956cam a22003254a 4500
001 993595623405961
005 20011220125500.0
008 010730s2001 nyua b 001 0 eng
010 ##$a 2001049336
015 ##$aGBA1-66182
016 7#$a962002836 $2GyFmDB
020 ##$a9783540422891 (alk. paper)
020 ##$a3540422897 (alk. paper)
035 ##$a(OCOLC)47739799
035 ##$9494282
035 ##$a(MoKL)359562-lhalldb
035 ##$a(lhalldb)359562-lhalldb
040 ##$aDLC $cDLC $dOHX $dC#P $dUKM $dLHL
042 ##$apcc
049 ##$aLHLA
090 ##$aQA76.9.D3 $bR435 2001
245 00$aRelational data mining / $cSašo Džeroski, Nada Lavrač (eds.).
260 ##$aBerlin ; $aNew York : $bSpringer, $cc2001.
300 ##$axix, 398 p. : $bill. ; $c24 cm.
504 ##$aIncludes bibliographic references and indexes.
650 #0$aDatabase management
650 #0$aRelational databases
650 #0$aData mining
700 1#$aDžeroski, Sašo, $d1968-
700 1#$aLavrač, Nada
948 ##$aLTI 09/01/2008
994 ##$aE0 $bLHL
```



Browse
Search

Journal
Search

All
Databases

Citation
Linker

Themed
Collections

Document
Delivery



Sign in

Menu

Search anything

Library Catalog



Advanced Search

All items that contain my query words anywhere in the record

For optimum display, please use Chrome or Firefox. Safari users may not be able to see all the relevant information.



BOOK

Relational data mining / Sašo Džeroski, Nada Lavrač (eds.)

Džeroski, Sašo, 1968-; Lavrač, Nada

c2001

Available at Linda Hall Library Books - 2nd Floor (QA76.9 .D3 R435 2001)

Top

Send to

Get it

Details

Send to



Email



QR



Permalink



Citation



EasyBib



Export RIS



Export to Excel

Get it

Back to Locations

LOCATION ITEMS

Linda Hall Library

Available, Books - 2nd Floor; QA76.9 .D3 R435 2001

(1 copy, 1 available, 0 requests)



Item in place

Loanable

Details

Title Relational data mining / Sašo Džeroski, Nada Lavrač (eds.)

Creator/Contributor Džeroski, Sašo, 1968-

Lavrač, Nada

Subjects Database management

Relational databases

Data mining

Identifier ISBN : 9783540422891 (alk. paper)

ISBN : 3540422897 (alk. paper)

OCLC : (OCoLC)47739799

Publisher Berlin ; New York : Springer

Creation date c2001

Format xix, 398 p. : ill. ; 24 cm.

General notes Includes bibliographic references and indexes.

Virtual Browse



Access 2000
c1999



Understanding database management
c1984



Query processing in database systems
c1985



Relational data mining
c2001



Relational database systems : analysis and comparison
1983

Links

[Display Source Record](#)  >

APPENDIX DZEROSKI02

SAŠO DŽEROSKI
NADA LAVRAČ
EDITORS

Relational Data Mining



Springer

As the first book devoted to relational data mining, this coherently written multi-author monograph provides a thorough introduction and systematic overview of the area. The first part introduces the reader to the basics and principles of classical knowledge discovery in databases and inductive logic programming; subsequent chapters by leading experts assess the techniques in relational data mining in a principled and comprehensive way; finally, three chapters deal with advanced applications in various fields and refer the reader to resources for relational data mining.

This book will become a valuable source of reference for R&D professionals active in relational data mining. Students as well as IT professionals and ambitious practitioners interested in learning about relational data mining will appreciate the book as a useful text and gentle introduction to this exciting new field.

ISBN 3-540-42289-7



9 783540 422891

<http://www.springer.de>

Sašo Džeroski • Nada Lavrač (Eds.)

Relational Data Mining

With 79 Figures and 69 Tables



Springer

QA 76.9
. D3R435
2001

Editors

Sašo Džeroski
Nada Lavrač

Jožef Stefan Institute
Jamova 39
1000 Ljubljana, Slovenia
E-mail: saso.dzeroski@ijs.si

Relational
Data Mining

Library of Congress Cataloging-in-Publication data applied for

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Relational data mining/Sašo Džeroski; Nada Lavrač (ed.). – Berlin;
Heidelberg; New York; Barcelona; Hong Kong; London; Milan;
Paris; Tokyo: Springer, 2001

ISBN 3-540-42289-7

ACM Subject Classification (1998): H.2.8, I.2.6, I.2.4, D.1.6, I.5

ISBN 3-540-42289-7 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
is a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

The use of general descriptive names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by the editors
Cover Design: d&p, design & production, Heidelberg
Printed on acid-free paper SPIN 10843997 – 06/3142SR – 5 4 3 2 1 0

Foreword

The area of data mining, or knowledge discovery in databases, started to receive a lot of attention in the 1990s. Developments in sensing, communications and storage technologies made it possible to collect and store large collections of scientific and industrial data. The abilities to analyze such data sets had not developed as fast. Data mining research arose at the intersection of several different research areas, notably statistics, databases, machine learning and algorithms. The area can loosely be defined as the analysis of large collections of data for finding models or patterns that are interesting or valuable.

The development of data mining methods requires the solution of several different types of problems. The data can have a very large number of dimensions, indicating that for example examining every pair of variables is impractical. The data can have hundreds of millions of observations, and therefore only a limited number of passes through the data can be done. The data can be observations of a process about which very little is known; hence there is no background knowledge available, and thus selection of appropriate models can be challenging. Or there can be heaps of background knowledge available, and methods that overlook it are destined to fail.

Most data mining methods have been developed for data in the traditional matrix form: rows represent observations, and columns represent variables. This representation has been the traditional one used in statistics, and it has many advantages. For example, matrix operations can be used to represent several data analytic procedures quite succinctly, and these representations make it possible to devise efficient algorithms.

However, data about the real world is seldom of this form. Rather, the application domain contains several different types of entities, of which different types of data are known. Only recently has a large body of research aimed at data mining on such data emerged.

Relational data mining studies methods for knowledge discovery in databases when the database has information about several types of objects. This, of course, is usually the case when the database has more than one table. Hence there is little doubt as to the relevance of the area; indeed, one can wonder why most of data mining research has concentrated on the single table case.

LINDA HALL LIBRARY
Kansas City, Mo.

Table of Contents

Part I. Introduction

1. Data Mining in a Nutshell

Sašo Džeroski	3
1.1 Introduction	3
1.2 Data mining tasks	4
1.3 Patterns	8
1.4 Basic algorithms	16
1.5 Relational data mining	22
1.6 Data mining literature and Internet resources	25
1.7 Summary	26

2. Knowledge Discovery in Databases: An Overview

Usama Fayyad	28
2.1 Introduction	28
2.2 From transactions to warehouses to KDD	29
2.3 Why data mining?	31
2.4 KDD and data mining	32
2.5 Data mining methods: An overview	34
2.6 Applications in science data analysis	37
2.7 Research challenges for KDD	41
2.8 ILP and KDD: Prospects and challenges	43
2.9 Concluding remarks	45

3. An Introduction to Inductive Logic Programming

Sašo Džeroski and Nada Lavrač	48
3.1 Introduction	48
3.2 Logic programming and databases	49
3.3 Logic programming in a nutshell	51
3.4 The basic ILP task: Relational rule induction	55
3.5 Structuring the space of clauses	58
3.6 Searching the space of clauses	60
3.7 Bounding the search for clauses	62

3.8	Transforming ILP problems to propositional form	67
3.9	Relational data mining tasks addressed by ILP	69
3.10	ILP literature	70
3.11	Summary	71
 4. Inductive Logic Programming for Knowledge Discovery in Databases		
	Stefan Wrobel	74
4.1	Introduction	74
4.2	ILP: Relational analysis technology	81
4.3	ILP subgroup discovery: MIDOS	84
4.4	Using MIDOS and other ILP methods in KEPLER	93
4.5	Conclusion	99
<hr/>		
Part II. Techniques		
<hr/>		
5. Three Companions for Data Mining in First Order Logic		
	Luc De Raedt, Hendrik Blockeel, Luc Dehaspe, and Wim Van Laer	105
5.1	Introduction	105
5.2	Representation	106
5.3	ICL: Inductive classification logic	114
5.4	TILDE: Top-down induction of logical decision trees	116
5.5	CLAUDIEN: Clausal discovery	117
5.6	Practical use: Getting started	119
5.7	Sample application: Mutagenesis	133
5.8	An exercise	136
5.9	Conclusions and practical info	137
 6. Inducing Classification and Regression Trees in First Order Logic		
	Stefan Kramer and Gerhard Widmer	140
6.1	Introduction	140
6.2	Tree induction in logic	142
6.3	Structural classification and regression trees (S-CART): The top level algorithm	145
6.4	Growing a tree in first-order logic	145
6.5	Model selection by error/cost complexity pruning	150
6.6	First-order model trees	152
6.7	Applications	153
6.8	Related work	154
6.9	Conclusion	156

7. Relational Rule Induction with CPROGOL4.4: A Tutorial Introduction	
Stephen Muggleton and John Firth	160
7.1 Introduction	160
7.2 How to obtain CPROGOL4.4	161
7.3 Developing an input file for CPROGOL4.4	162
7.4 The theory	167
7.5 Estimating accuracy and significance	179
7.6 Declarative bias	182
7.7 Setting resource bounds	185
7.8 Debugging PROGOL input files	186
7.9 Summary	187
8. Discovery of Relational Association Rules	
Luc Dehaspe and Hannu Toivonen	189
8.1 Introduction	189
8.2 From association rules to query extensions	190
8.3 Evaluation measures	194
8.4 Declarative language bias	198
8.5 Query (extension) discovery with WARMR	201
8.6 A sample run	206
8.7 Discussion	208
9. Distance Based Approaches to Relational Learning and Clustering	
Mathias Kirsten, Stefan Wrobel, and Tamás Horváth	213
9.1 Introduction	213
9.2 A first-order distance measure	215
9.3 Instance-based learning with RIBL2	220
9.4 Hierarchical agglomerative clustering with RDBC	221
9.5 FORC - k -means for multirelational data	223
9.6 A case study in mRNA signal structures	225
9.7 Conclusion	230
<hr/>	
Part III. From Propositional to Relational Data Mining	
<hr/>	
10. How to Upgrade Propositional Learners to First Order Logic: A Case Study	
Wim Van Laer and Luc De Raedt	235
10.1 Introduction	235
10.2 Knowledge representation	236
10.3 The propositional learner CN2	240

10.4 Upgrading CN2	241
10.5 Some experimental results with ICL	252
10.6 Related work and conclusions	256
11. Propositionalization Approaches to Relational Data Mining	
Stefan Kramer, Nada Lavrač, and Peter Flach	262
11.1 Introduction	262
11.2 Background and definition of terms	265
11.3 An example illustrating a simple propositionalization	267
11.4 Feature construction for general-purpose propositionalization	271
11.5 Special-purpose feature construction	274
11.6 Related transformation approaches	277
11.7 A sample propositionalization method: Extending LINUS to handle non-determinate literals	279
11.8 Concluding remarks	286
12. Relational Learning and Boosting	
Ross Quinlan	292
12.1 Introduction	292
12.2 Boosting	293
12.3 FOIL	294
12.4 Overview of FFOIL	297
12.5 Boosting FFOIL	299
12.6 Experiments	300
12.7 Summary	304
13. Learning Probabilistic Relational Models	
Lise Getoor, Nir Friedman, Daphne Koller, and Avi Pfeffer	307
13.1 Introduction	307
13.2 Probabilistic models	309
13.3 Relational models	312
13.4 Probabilistic relational models	315
13.5 Learning PRMs	321
13.6 Experimental results	325
13.7 Discussion and related work	329
13.8 Extensions	331
13.9 Conclusions	333

Part IV. Applications and Web Resources

14. Relational Data Mining Applications:**An Overview**

Sašo Džeroski 339

14.1 Introduction 339

14.2 Drug design 340

14.3 Predicting mutagenicity and carcinogenicity 343

14.4 Predicting protein structure and function 346

14.5 Medical applications 349

14.6 Environmental applications 350

14.7 Mechanical engineering applications 353

14.8 Traffic engineering applications 356

14.9 Text mining, Web mining, and natural language processing . 357

14.10 Business data analysis 358

14.11 Miscellaneous applications 359

14.12 Summary and discussion 360

15. Four Suggestions and a Rule**Concerning the Application of ILP**

Ashwin Srinivasan 365

15.1 Introduction 365

15.2 Background 366

15.3 When and why ILP? 367

15.4 Encoding background knowledge 368

15.5 Utility mismatch 371

15.6 Comprehensibility 371

15.7 From nursery slopes to Darwin's rule 372

16. Internet Resources on ILP for KDDLjupčo Todorovski, Irene Weber, Nada Lavrač, Olga Stěpánková,
Sašo Džeroski, Dimitar Kazakov, Darko Zupanič, and Peter Flach 375

16.1 Introduction 375

16.2 Brief history on ILP Internet resources 376

16.3 ILPnet2 Internet resources 377

16.4 Other ILP-related Internet resources 383

16.5 KDD related Internet resources 385

16.6 Conclusion 385

Author Index 389**Subject Index** 391

9. Distance Based Approaches to Relational Learning and Clustering

Mathias Kirsten¹, Stefan Wrobel², and Tamás Horváth¹

¹ German National Research Center for Information Technology, GMD - AiS.KD
Schloß Birlinghoven, D-53754 Sankt Augustin, Germany

² School of Computer Science, IWS, University of Magdeburg
Universitätsplatz 2, D-39016 Magdeburg, Germany

Abstract

Within data analysis, *distance-based* methods have always been very popular. Such methods assume that it is possible to compute for each pair of objects in a domain their mutual distance (or similarity). In a distance-based setting, many of the tasks usually considered in data mining can be carried out in a surprisingly simple yet powerful way. In this chapter, we give a tutorial introduction to the use of distance-based methods for relational representations, concentrating in particular on predictive learning and clustering. We describe in detail one relational distance measure that has proven very successful in applications, and introduce three systems that actually carry out relational distance-based learning and clustering: RIBL2, RDBC and FORC. We also present a detailed case study of how these three systems were applied to a domain from molecular biology.

9.1 Introduction

Within data analysis, *distance-based* methods have always been very popular [9.23]. The central assumption of distance-based methods is that it is possible, for a particular domain under consideration, to specify for each pair of objects their mutual distance (or similarity). In certain settings, these distance values can be assumed to be given as part of the application, e.g., when humans are asked to rate distances between pairs of choices as done in certain psychological experiments. More often, however, in data analysis we are starting with objects described in a certain representation language (our *instance space* X) and use appropriate algorithms to compute distances for these objects.

Once we have such distances between object pairs, many of the tasks usually considered in data mining can be carried out in a disarmingly simple yet powerful way.

9.1.1 Predictive learning

Let us first consider how it is possible to perform predictive learning from examples given only distances between pairs of objects. Instead of forming an explicit hypothesis in the form of first-order clauses, we simply store all

available examples. Later, when we are asked to classify an unseen object, we compare this new object to all previously seen objects, and predict the function value associated with the *nearest neighbor* or our query object, i.e., the example object that has the smallest distance to our query object.

Considering only one object makes this algorithm very sensitive to noise, so generally, the method of choice is k -nearest neighbor, where the k nearest objects get to vote by majority on the prediction. Since this method of learning is strongly based on the instances available as examples, it is often referred to as *instance-based learning*. Many authors also call it *lazy learning*, emphasizing the fact that the learner does not bother to explicitly form a hypothesis, but lazily waits until an object to be classified comes along, and only then forms a (local, query-specific) hypothesis.

9.1.2 Clustering

The task of cluster analysis is, roughly speaking, the task of grouping a set of instances $\mathcal{I}$ from an instance space X into different subsets (clusters) such that objects belonging to the same cluster are maximally similar to each other while simultaneously being maximally dissimilar to objects from other clusters. Finding a clustering that is guaranteed to be optimal in terms of a chosen quality measure is an infeasible task, as it would require an exhaustive search through the space of all possible groupings of objects.

Hence, the available distance-based clustering algorithms use heuristic strategies, two of the more prominent of which are bottom-up *agglomerative* clustering strategies, and iterative *k-means* clustering. Hierarchical agglomerative clustering (c.f. [9.3]) constructs a hierarchy of nested clusterings by repeatedly merging the two existing clusters that are closest to each other, starting with all the individual instances as one-element clusters. The *k-means* method [9.15], on the other hand, first randomly selects k cluster centers, then repeatedly (a) assigns each instance to the cluster of the nearest cluster center, and (b) computes the new cluster center for each of the k current clusters, until a termination criterion is satisfied.

In this chapter, we give a tutorial introduction into the use of the above-described distance-based methods for learning and clustering. As a key component, in Section 9.2, we first explain what are the special challenges in defining a distance for first-order representations, and show in detail one first-order distance measure that has proven very successful in applications. We then turn to three systems that actually carry out first-order distance-based learning and clustering: RIBL2 (Section 9.3) is an instance-based learner, while RDBC and FORC (Sections 9.4 and 9.5) are agglomerative and k -means clustering systems, respectively. In each of the system's descriptions, we also provide more detail as to the workings of the chosen basic method, and discuss the particular adaptations to first-order problems. Finally, in Section 9.6, we present a detailed case study of how these three systems were applied to

Let us assume that $I = P(\text{set1}, 125, \text{personal})$ is an instance in our example and is further described by the ground atoms

```
cheese(set1, camembert, 150, france)
wine(set1, mouton, 1988, 0.75)
wine(set1, gallo, 1995, 0.5)
vineyard(gallo, famous, large, usa)
vineyard(mouton, famous, small, france) .
```

The set consisting of I and the above five ground atoms is the *case* of I .

Our example indicates that a *type* defining the set of possible values can be assigned to each argument of the predicate symbols. Furthermore, the case of an instance is usually not explicitly given. Instead, a set $\mathcal{B}$ of ground atoms, also referred to as *background knowledge*, is given, and for a given instance I , a subset of $\mathcal{B}$ is computed that defines (together with I) the case of I . This approach allows for a more succinct representation, as ground atoms may occur in more than one case.

More precisely, when applying one of our algorithms, the user has first to define the vocabulary of the predicate symbols to be used in the relational representation. Among these symbols there is a distinguished predicate symbol P that will be used to represent the elements of the set X of all possible *instances*, also called *instance space*. The other predicate symbols, called *background predicates*, are used to describe the user's background knowledge about the task. Each predicate symbol is associated (also by the user) with a declaration defining the types and modes of its arguments. The type of an argument is one of the basic types from $\{\text{number}, \text{discrete}, \text{list}, \text{term}, \text{name}\}$, while its mode defines whether or not the argument must be compared with corresponding arguments during the distance computation. The type 'name' is used for arguments that contain identifiers representing objects.

Example 9.2.2. The vocabulary of Example 9.2.1 consists of P and the background predicates 'cheese', 'wine', and 'vineyard' that are associated with the following declaration:

```
P(a1 : name, a2 : number, a3 : discrete)
cheese(a1 : name, a2 : discrete, a3 : number, a4 : discrete)
wine(a1 : name, a2 : discrete, a3 : number, a4 : number)
vineyard(a1 : name, a2 : discrete, a3 : discrete, a4 : discrete) .
```

In order to compute a case of an instance $I \in X$, we rely on the instance I , the background knowledge $\mathcal{B}$, and a non-negative integer $d \geq 0$ called *depth bound*. A case of I with respect to $\mathcal{B}$ and d is defined as the largest set $\{I\} \cup \mathcal{B}'$ such that $\mathcal{B}' \subseteq \mathcal{B}$ and for every background atom $B \in \mathcal{B}'$ it holds that there are $d' < d$ and $B_0, B_1, \dots, B_{d'-1}, B_{d'}$ satisfying

– $B_0 = I$, $B_{d'} = B$, and $B_i \in \mathcal{B}'$ for $i = 1, \dots, d' - 1$,

- B_i and B_{i+1} contain at least one common identifier for every $i = 0, \dots, d' - 1$.

Example 9.2.3. To illustrate the above definition let us assume that the background knowledge $\mathcal{B}$ in Example 9.2.1 consists of the following facts:

```
% set ID:name, cheese type:discrete, weight:number, origin:discrete
cheese(set1, camembert, 150, france)
cheese(set25, roquefort, 200, france)
cheese(set25, ricotta, 100, italy)
```

```
% set ID:name, vineyard ID:name, year:number, bottle size:number
wine(set1, mouton, 1988, 0.75)
wine(set1, gallo, 1995, 0.5)
wine(set25, mouton, 1995, 0.75)
```

```
% vineyard ID:name, popularity:discrete, size:number, country:discrete
vineyard(gallo, famous, large, usa)
vineyard(mouton, famous, small, france)
```

Then the case defined by the instance $P(\text{set1}, 125, \text{personal})$ with respect to $\mathcal{B}$ and $d = 2$ is given in Figure 9.1.

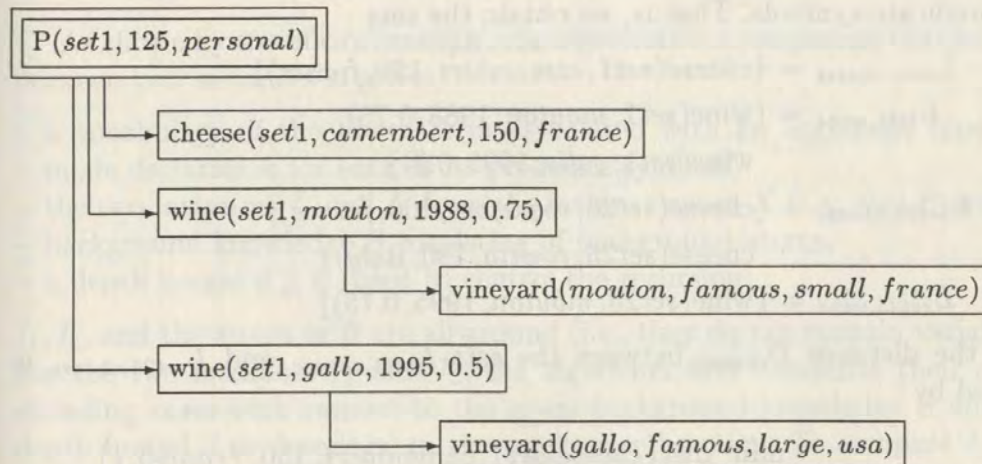


Fig. 9.1. The case defined by I_1 with respect to $\mathcal{B}$ and $d = 2$ in Example 9.2.1.

9.2.2 The distance measure

Before giving an informal description of the algorithm for computing the distance between two instances, we first give an example to illustrate how this algorithm works.

Example 9.2.4. Suppose we are given

- the instances $I_1 = P(\text{set1}, 125, \text{personal})$, $I_2 = P(\text{set25}, 195, \text{mail})$,
- the background knowledge B of Example 9.2.3,
- the depth bound $d = 2$,

and want to compute the distance between cases of I_1 and I_2 with respect to B and d .

This distance is computed as the normalized sum of the distances between the instances' corresponding arguments, i.e.,

$$\text{dist}(I_1, I_2, B, d) = \frac{1}{3} \left(\text{dist}(\text{set1}, \text{set25}) + \text{dist}(125, 195) + \text{dist}(\text{personal}, \text{mail}) \right). \quad (9.1)$$

For computing the last two terms, standard propositional distance measures can be applied, i.e.,

$$\text{dist}(125, 195) = |195 - 125|/500 = 0.14 \quad (9.2)$$

(here the denominator denotes the highest possible price) and

$$\text{dist}(\text{personal}, \text{mail}) = 1 \quad (9.3)$$

(as they are different). In order to compute the distance between the object identifiers *set1* and *set25*, from the cases of *set1* and *set25* we first collect the sets of literals containing *set1* and *set25*, respectively, and partition them by the predicate symbols. That is, we obtain the sets

$$L_{\text{set1}, \text{cheese}} = \{\text{cheese}(\text{set1}, \text{camembert}, 150, \text{france})\}$$

$$L_{\text{set1}, \text{wine}} = \{\text{wine}(\text{set1}, \text{mouton}, 1988, 0.75), \\ \text{wine}(\text{set1}, \text{gallo}, 1995, 0.5)\}$$

$$L_{\text{set25}, \text{cheese}} = \{\text{cheese}(\text{set25}, \text{roquefort}, 200, \text{france}), \\ \text{cheese}(\text{set25}, \text{ricotta}, 100, \text{italy})\}$$

$$L_{\text{set25}, \text{wine}} = \{\text{wine}(\text{set25}, \text{mouton}, 1995, 0.75)\}$$

Now the distance D_{cheese} between the sets $L_{\text{set1}, \text{cheese}}$ and $L_{\text{set25}, \text{cheese}}$ is defined by

$$D_{\text{cheese}} = \frac{\min_{l \in L_{\text{set25}, \text{cheese}}} \text{dist}(\text{cheese}(\text{set1}, \text{camembert}, 150, \text{france}), l)}{|L_{\text{set25}, \text{cheese}}|} \quad (9.4)$$

$$= \frac{\min \left(\left(1 + \frac{|150-200|}{300}\right) + 0 \right)/3, \left(1 + \frac{|150-100|}{300}\right) + 1 \right)/3}{2} \quad (9.5)$$

$$= 0.1944 .$$

We note that in (9.4) we normalize by the cardinality of the *larger* set $L_{\text{set25}, \text{cheese}}$, and by 3 the two terms in (9.5), as we do not compare again the first arguments containing the set identifiers. Applying that

$$\begin{aligned}
\text{dist}(\text{mouton}, \text{gallo}) &= \\
&= \frac{\min \left(\frac{\text{dist}(\text{famous}, \text{famous}) + \text{dist}(\text{small}, \text{large}) + \text{dist}(\text{france}, \text{usa})}{3} \right)}{1} \\
&= 0.6666,
\end{aligned}$$

the distance D_{wine} between $L_{\text{set1}, \text{wine}}$ and $L_{\text{set25}, \text{wine}}$ is computed in a similar manner, i.e.,

$$\begin{aligned}
D_{\text{wine}} &= \frac{\min_{l \in L_{\text{set1}, \text{wine}}} \text{dist}(\text{wine}(\text{set25}, \text{mouton}, 1995, 0.75), l)}{|L_{\text{set1}, \text{wine}}|} \\
&= \frac{\min \left((0 + \frac{|1995-1988|}{100} + 0)/3, (0.6666 + 0 + \frac{|0.75-0.5|}{1})/3 \right)}{2} \\
&= 0.0117.
\end{aligned}$$

Using the above values, the distance between object identifiers *set1* and *set25* is then computed by

$$\begin{aligned}
\text{dist}(\text{set1}, \text{set25}) &= \frac{D_{\text{cheese}} + D_{\text{wine}}}{2} \\
&= 0.1031.
\end{aligned}$$

Combining (9.2), (9.3), and the above equation with (9.1), we finally obtain

$$\text{dist}(I_1, I_2, \mathcal{B}, d) = (0.1031 + 0.14 + 0)/3 = 0.081.$$

As shown by the above example, the algorithm for computing the distance between two instances is given

- a vocabulary of the above form associated with an argument type and mode declaration for each of its predicate symbols,
- the two instances I_1 and I_2 from the instance space X (i.e., two P -atoms),
- background knowledge $\mathcal{B}$ consisting of background atoms,
- a depth bound $d \geq 0$ (used to control the recursion).

I_1, I_2 , and the atoms in $\mathcal{B}$ are all ground (i.e., they do not contain variables). For the two instances I_1 and I_2 , the algorithm first computes their corresponding *cases* with respect to the given background knowledge $\mathcal{B}$ and the depth bound d as described in the previous subsection. To compare I_1 and I_2 , the distance function recursively descends into their corresponding cases, beginning with the ground P -atoms I_1 and I_2 . In general, when comparing two ground atoms, each with the same predicate symbol, one first computes the distance between their corresponding arguments that have been marked for comparison in the mode declaration, then sums up these distances, and normalizes finally this sum with the number of marked arguments. Depending on their common type, two arguments are compared as follows.

number and discrete: For arguments of these basic types, some corresponding elementary distance function is called.

list and term: For arguments of these types, our measure employs edit distances. Intuitively, with an edit distance, we are given a set of edit operations such as insert, delete, and change with an associated cost function that tells us how expensive it is to delete or insert an element of the underlying alphabet, or how expensive it is to change one into another. The edit distance is then defined as the smallest cost of a sequence of such operations that turns the first object into the second. For both of lists and terms, there are efficient algorithms for computing the edit distance (see, e.g., [9.22, 9.24]).

name: When comparing two arguments of this type, the algorithm first checks whether the depth of the recursion has already reached the input parameter d (i.e., the depth bound d is used to ensure limited effort for the recursive computation). If yes, the arguments are compared as that of type discrete. Otherwise, all further facts from the cases about the respective named objects are first collected. Each of these two sets is then partitioned by the predicate symbols, and for every predicate symbol we compare its corresponding subsets from the two sets as follows. For each fact from the smallest set we compute recursively its distance from the larger set (i.e., the distance between this fact and its nearest fact from the larger set), then sum up these distances and normalize with the cardinality of the larger set.

9.3 Instance-based learning with RIBL2

RIBL2 is a first-order instance-based learner applying the classical k -nearest neighbor (kNN) classifier in a relational representation, using the distance measure described in the preceding section. In more detail, a k -nearest neighbor algorithm determines its predictions as follows. To simplify the discussion, note that our distance measure takes on values in the range $[0, 1]$, so we can equivalently work with the function

$$\text{sim}(x, y) := 1 - \text{dist}(x, y)$$

which is 1 for identical objects and falls towards 0 with increasing distance between objects. Given a set of examples E and a new instance x_{new} from X to be classified, let us call $N \subseteq E$ the set of k nearest neighbors of x_{new} according to sim . Further details depend on whether we are predicting a discrete or continuous variable. For a *discrete* variable, denote by

$$N_{y_j} := \{e = (x, y_j) \in N\}.$$

those neighbors that are voting for value y_j . The algorithm will then predict

$$h(x_{\text{new}}) := \text{argmax}_{y_j} \sum_{e=(x, y_j) \in N_{y_j}} \text{sim}(x, x_{\text{new}})^a,$$

where a is a parameter that determines how much an influence is allocated to farther-away neighbors. If we are predicting a continuous variable, we use a weighted average instead:

$$h(x_{new}) := \frac{\sum_{e=(x,y) \in N} \text{sim}(x, x_{new})^a \cdot y}{\sum_{e=(x,y) \in N} \text{sim}(x, x_{new})^a}.$$

In order to determine k , most k -nearest neighbor algorithm internally estimate the accuracy that can be achieved (e.g., by cross-validation), and then choose the value of k that has ranked highest for all future classifications. Despite this very simple principle, k -nearest neighbor methods are known to be competitive with and often superior to other approaches in terms of predictive accuracy. Thus, they are attractive choices whenever an explicit hypothesis is not needed.

9.4 Hierarchical agglomerative clustering with RDBC

Finding a clustering that is guaranteed to be optimal in terms of a chosen quality measure is an infeasible task, as it would require an exhaustive search through the space of all possible clusterings. Hence, the available distance-based clustering algorithms use heuristic strategies. The RDBC algorithm employs one of the most basic distance-based clustering methods, namely hierarchical agglomerative clustering (c.f. [9.3]).

9.4.1 Forming a cluster hierarchy

Together with the distance measure described in Section 9.2, hierarchical agglomerative clustering does not build flat partitions of the data but iteratively constructs a list of nested clusterings $\mathcal{C}_1, \dots, \mathcal{C}_n$. Figure 9.2 shows an example together with the typical visualization of such a clustering in a *dendrogram*.

The construction of the hierarchical clustering is done in a bottom-up manner by starting with the clustering $\mathcal{C}_1$ where each case forms its own one-elemental cluster. The next clustering is built by combining the two closest respectively most similar clusters to a new cluster. In this case, the distance between two clusters is simply the distance between the two cases, as each cluster consists of a single element. For all subsequent clusterings the step of combining the two most similar clusters is repeated until finally the clustering ($\mathcal{C}_n$) consists of a single cluster only. When joining two clusters to a new one, the distances between the new cluster and all other clusters can be computed from the already known distances of the current clusters. Depending on the chosen computation rule we get different variants of the hierarchical agglomerative clustering:

$$\text{dist}(C_1 \cup C_2, C_3) := f(\text{dist}(C_1, C_3), \text{dist}(C_2, C_3)).$$

RDBC includes the four most popular variants: $f = \min$, the so called *single-link* clustering, $f = \max$ the *complete-link* clustering, and for $f = \text{average}$ the *average-link* clustering. A somewhat more complicated choice of f results in *Ward's method*, which guarantees that the sum of (squared) distances is minimized for each aggregation step (this does not necessarily hold for the final result though).

So, suppose we have a set of instances $I_1, \dots, I_5$ together with their pairwise distance information computed by RIBL's distance measure:

	I_1	I_2	I_3	I_4	I_5
I_1	0	0.12	0.60	0.64	0.77
I_2		0	0.70	0.80	0.84
I_3			0	0.35	0.45
I_4				0	0.40
I_5					0

Employing the *complete-link* variant of RDBC on this data yields the cluster hierarchy shown in Figures 9.2 and 9.3.

9.4.2 Turning the cluster hierarchy into flat partitions

The chosen agglomerative approach renders a distance-based hierarchy of cases where the distance between subtrees decreases as we move from root towards the leaves. For many applications, such a hierarchical tree is all that is required. If however, the clustering task requires the selection of a single set of clusters, we need a technique for turning the tree into such a set. Since we are trying to minimize intra-cluster distance (respectively maximize intra-cluster similarity), it is natural to base cluster selection from the tree on this criterion. RDBC therefore uses a search procedure that "cuts up" the tree according to growing intra-cluster similarity thresholds, and selects the threshold and its corresponding group of clusters that minimizes average intra-cluster distance. Thus the exact number of clusters does not have to be predefined by the user but is automatically computed according to the quality function. More precisely, in RDBC the quality of a set of clusters $\mathcal{C}$ is defined as:

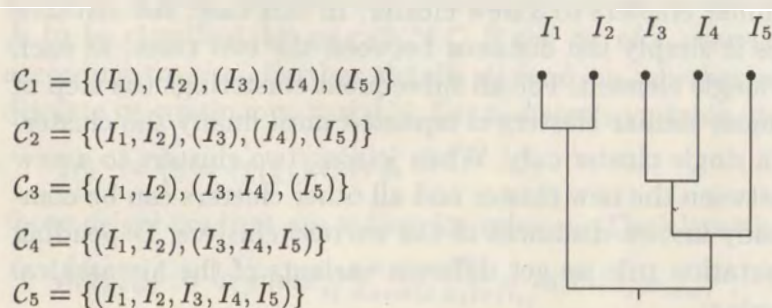


Fig. 9.2. Hierarchical clustering and its visualization in a dendrogram.

$$qual(C) = \frac{\sum_{C \in \mathcal{C}} dist_{intra}(C)}{|\mathcal{C}|},$$

where $dist_{intra}(C)$ denotes the intra-cluster distance of a cluster C , which is the averaged sum of all distances between element pairs of the cluster, and is hence a measure of the cluster's compactness.

It should be noted, that the above quality measure cannot be applied directly as it reaches its maximum for a partitioning where each cluster consists of a single case. RDBC solves the problem by moving all single-element clusters into an *outlier cluster*, and the optimization of $qual(C)$ takes place on the resulting clustering.

So, coming back to our example involving $I_1, \dots, I_5$, the transformation of the hierarchy into a set of partitions firstly requires to find a threshold for which the above quality function $qual(C)$ is maximal.

All potential thresholds are evaluated by removing those connections from the hierarchical tree structure which lay below the given threshold. Thus, the result is a clustering forest rather than a tree. A cluster (respectively a partition) is then the set of instances belonging to a single tree¹.

In the example, cutting up the hierarchy at $d = 0.5$ results in two partitions $C_1 = \{I_1, I_2\}$ and $C_2 = \{I_3, I_4, I_5\}$. The quality for this clustering is calculated as

$$qual(\{C_1, C_2\}) = \frac{dist(I_1, I_2) + (dist(I_3, I_5) + dist(I_4, I_5) + dist(I_4, I_3))/3}{2} = 0.26$$

Finding the best threshold in the example is left as an exercise to the reader.

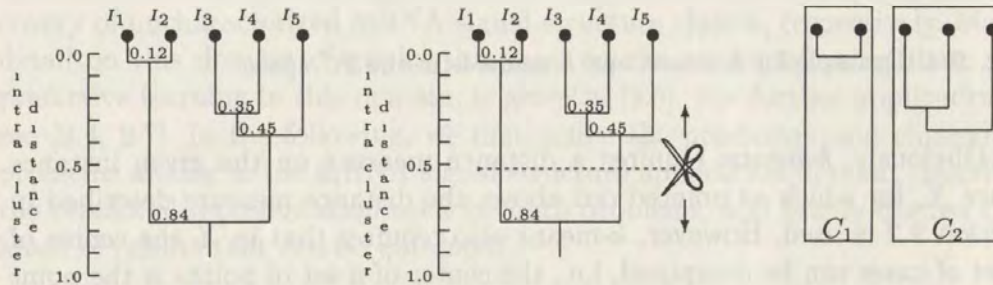


Fig. 9.3. Transforming a hierarchical clustering into flat partitions.

9.5 FORC - k -means for multirelational data

A very popular clustering technique for finding clusterings of a predefined size k is the k -means method [9.15]. FORC is the first clustering system to

¹ Isolated instances without any connections are not handled as one-elemental clusters but are gathered into the *outlier cluster*.

carry this algorithm over to first-order, relying on the distance measure of Section 9.2.

The basic idea of k -means is as follows. Firstly, an initial state has to be constructed by selecting or creating k cluster centers from X . In practice this is mostly done by randomly drawing k elements from the set of cases. With this initial state a two-step iteration is started:

1. Assign each case to the cluster of the nearest cluster center.
2. Compute the new cluster center for each of the k current cluster.
3. Go to step 1 unless clustering quality does not increase any more.

Thus, in each iteration the centers of the current clusters are determined in order to subsequently rebuild the k clusters on basis of the new centers. Figure 9.4 shows an example of this process for $X = \mathbb{R}^2$ (the corresponding cluster centers are pictured by the large circle respectively box).

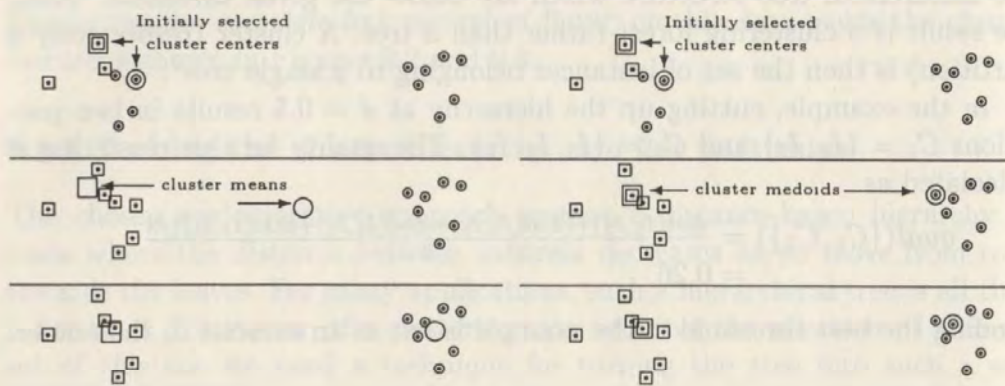


Fig. 9.4. Example for k -means and k -medoids in the $\mathbb{R}^2$ space.

Obviously, k -means requires a distance measure on the given instance space X , for which as pointed out above, the distance measure described in Section 9.2 is used. However, k -means also requires that in X the center of a set of cases can be computed, i.e., the center of a set of points is the point whose summed distances to the cases of the set is minimal. This is most easy if $X = \mathbb{R}^n$, i.e., for strictly numerical instance spaces. There the center of a set $C \subseteq X$ is simply given by its mean:

$$\text{center}(C) := \frac{1}{|C|} \sum_{c \in C} c$$

The corresponding (and usual) distance measure is the n -dimensional Euclidean metric but the Mahalanobis distance is [9.6, 9.14] also used.

For first-order data the problem is more challenging, since similarly obvious solutions for finding a center are not at hand. In fact, on multirelational data there is no longer any geometric or intuitive notion of the phrase “cluster

center". In addition, an analytic solution for the construction of the cluster center does not exist, therefore we have to make do with approximation.

FORC employs a solution that alleviates the problem of how to construct a previously not existing cluster center, which is to take an existing case as an approximation for the cluster center. This should be the most central instance of the set, i.e., the instance whose sum of squared distances to all other instances in the set is minimal and which is thus closest to the (unknown) cluster center. This approach is already known as the *k*-medoids method [9.10, 9.11], and first results of FORC obtained on the mutagenesis data set [9.21] using RIBL's distance measure show *k*-medoids' promising potential on first-order data [9.13].

Obviously, the whole process relies on distance information only, and is hence applicable to any representation for which an appropriate distance measure is available (e.g., a first-order representation + RIBL's distance measure). On the downside, when making no prior assumptions about the representation used, the computational complexity suffers from the expensive search for the medoid which requires time proportional to the squared number of instances in a cluster.

9.6 A case study in mRNA signal structures

In this section we present a case study designed to illustrate how relational distance-based learning and clustering algorithms can be applied in a real-world domain: the mRNA signal structure prediction and the automatic discovery of uncharacterized mRNA signal structure classes, respectively. More detail on this domain of application, and on the results of using RIBL for predictive learning in this domain, is given in [9.8]. For further applications, see [9.4, 9.7]. In the following, we first define the prediction and clustering problems arising in the mRNA signal structure application domain, describe the relational representation used for both problems, and briefly discuss the kinds of results that can be obtained.

9.6.1 The mRNA prediction and clustering problems

During the process of protein synthesis, messenger RNA (mRNA) molecules, a special kind of RNA molecules, mediate between the sites of information storage (i.e., DNA, which is functionally inactive) and protein synthesis. In general, an RNA is a linear sequence of the four bases (adenine – A, cytosine – C, guanine – G, and uracil – U) folded and coiled into specific conformations. The three-dimensional conformation of an RNA contains local regions of short complementary base-pairing, i.e., the two stable Watson-Crick base pairs A–U, G–C, and the weak G–U. The structure of RNA molecules is usually described at the following levels of abstraction:

primary structure: It is the linear sequence of the four bases that make up an RNA.

secondary structure: It is the collection of the base pairs occurring in the RNA's three-dimensional structure.

tertiary structure: It is the three-dimensional conformation of the RNA.

In addition, for the primary structures of messenger RNA molecules it also holds that they can be divided into three parts; they consist of a coding region (in their middle) defining the amino acid sequence of the target protein to be synthesized, and two non-coding regions (in their two ends) that contain regulatory motifs. Such regulatory motifs, called *signal structures*, control gene expression at the posttranscriptional level of the protein synthesis. Depending on their regulatory roles during the process of protein synthesis, signal structures are grouped into different signal structure *classes*. In this field of research, there are (at least) the following two challenging computational tasks:

- Detect (possibly new) signal structures belonging to known signal structure classes on a given uncharacterized mRNA.
- Find uncharacterized signal structure classes.

As a first step towards the direction of finding good algorithmical solutions to the above two problems, we consider the following two related problems, respectively. (In the description below, X denotes the set of all signal structures.)

signal structure prediction task: Given a set C of signal structure classes, a set $i \subseteq X$ of characterized signal structures (i.e., each with known class) belonging to one of the classes from C , and a signal structure $I \in X$ of unknown class from C , the aim is to give a good prediction of the class of I .

automatic discovery of uncharacterized signal structure classes:

In this task, we are given a set $i \subseteq X$ of uncharacterized signal structures (i.e., with unknown class information), the aim is to find a clustering $(I_1, \dots, I_k)$ of i with k as small as possible such that each cluster is a subset of some (possibly uncharacterized) signal structure class.

Now we turn to the problem of how to represent the components of the above computational tasks. When viewing the secondary structures of mRNA molecules, one can observe a set of different patterns [9.25], like hairpin loops, internal loops, etc, that make them up (see also Figure 9.5). These patterns are also referred to as *structure elements*. Furthermore, it also holds that mRNA's secondary structures can be viewed as trees (for more details see e.g., [9.20]), where the nodes represent the structure elements. As signal structures are (continuous) subsequences of the primary structures, this observation holds for them as well.

As an example, we consider a signal structure belonging to the class *Selenocysteine Insertion Sequence (SECIS)*. Its primary structure is

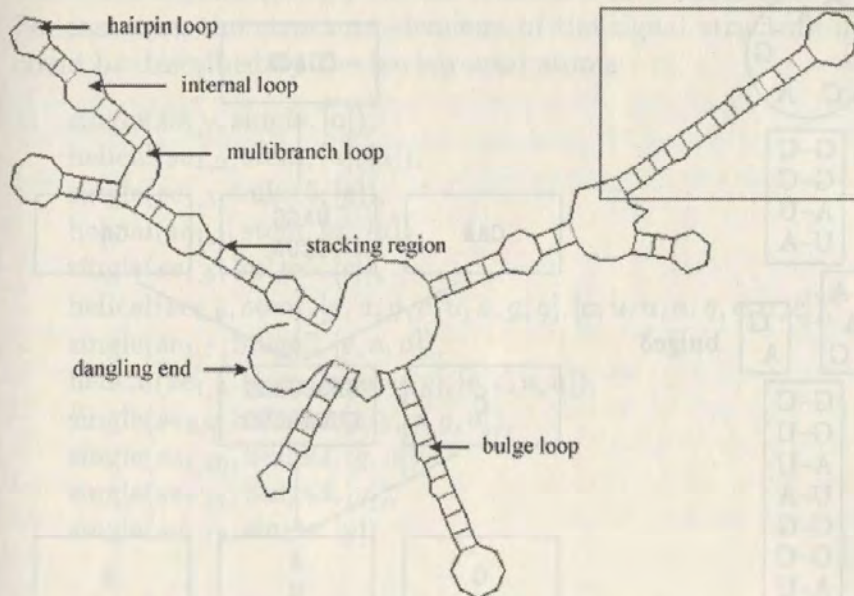


Fig. 9.5. mRNA structure with marked structure elements.

AUGACGAGCUAGGCAAUAGGCGAGACCUAGACUUAGCUCUGAG ,

and its secondary structure is given by the following complementary base pairs

```

AUGACGAGCUAGGCAAUAGGCG
| | ||||| | | | | | A
GAGU CUCGAUUCAG AUCCAG

```

In Figure 9.6, we give the structure elements that make up this secondary structure, and also the tree structure of these structure elements. As shown in the figure, the inner nodes on the longest path from the root all correspond to the pattern 'stem'. The root itself on this path is a dummy node, i.e., it does not represent any part of the signal structure. The other endpoint of the path corresponds to the hairpin loop (i.e., CGAGA) in the signal structure. This example indicates the following description of signal structures in our relational representation language.

The instances of this application are the signal structures. Using their tree structures, a signal structure, an instance from X , is described by the binary predicate

`structure(struct_id, tree_structure)`

where the first argument of type 'name' defines its identifier, and the second argument of type 'term' is used to describe its tree structure. For instance, the signal structure given in Figure 9.6 can be described by the ground atom

`structure(s1, dummy(se1,1, se1,2(se1,3, se1,4(...), se1,11), se1,12))`

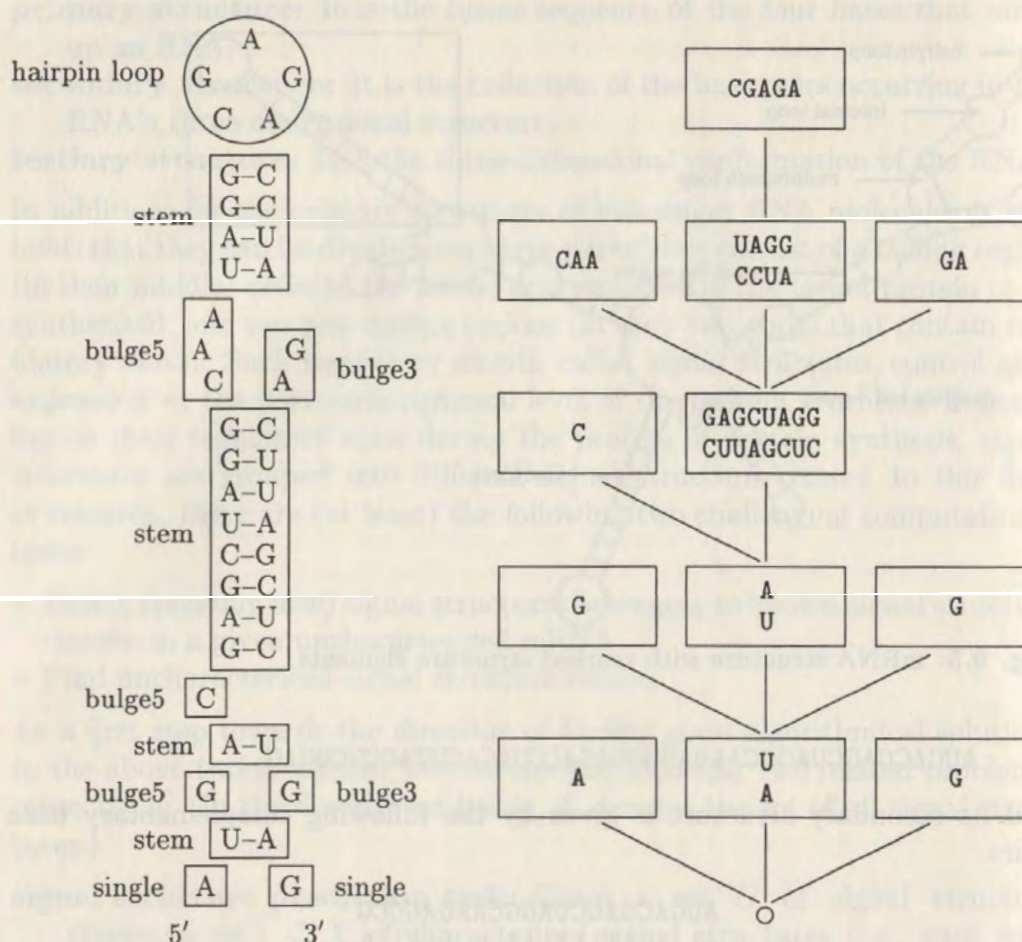


Fig. 9.6. The representation and the tree structure of a SECIS signal structure.

where ‘dummy’ is a constant denoting the root for each signal structure, and $se_{1,1}, \dots, se_{1,12}$ are further identifiers denoting the structure elements of s_1 . (We note that the identifiers $se_{1,1}, \dots, se_{1,12}$ are assigned to the descendants of the root of the tree given in Figure 9.6 according to preorder tree traversal.) The structure elements of the given signal structures are further described in the background knowledge. Depending on their types, structure elements are either helical (e.g., stems) or single (e.g., hairpin loop, bulge5, bulge3). A pattern of type single is described by the background predicate

$\text{single}(se_id, type, sequence)$

where the argument se_id of type ‘name’ is an identifier denoting the structure element, $type$ of type ‘discrete’ gives the structure element’s pattern, and $sequence$ of type ‘list’ defines the primary structure of the structure element. Patterns of type helical are described by the background predicate

$\text{helical}(se_id, type, sequence_1, sequence_2)$

where the arguments have similar semantics as in the case of predicate single. For instance, the structure elements of the signal structure in our example could be described by the background atoms:

```

single(se1,1, single, [a]),
helical(se1,2, stem, [u], [a]),
single(se1,3, bulge5, [g]),
helical(se1,4, stem, [a], [u]),
single(se1,5, bulge5, [c]),
helical(se1,6, stem, [g, a, g, c, u, a, g, g], [c, u, u, a, g, c, u, c]),
single(se1,7, bulge5, [c, a, a]),
helical(se1,8, stem, [u, a, g, g], [c, c, u, a]),
single(se1,9, hairpin, [c, g, a, g, a]),
single(se1,10, bulge3, [g, a]),
single(se1,11, bulge3, [g]),
single(se1,12, single, [g]).

```

9.6.2 Results

For our experiments we have chosen a small domain consisting of 66 signal structures over 5 signal structure classes. For predictive learning, the optimal number of neighbors (k) has been estimated by *leave-one-out* cross-validation. The results for the different values of k are given in the following table (for more details see also [9.8]).

Table 9.1. Results for estimating the optimal number of neighbors (k).

#instances	#correctly classified for											best accuracy (%)
	$k =$	1	2	3	4	5	6	7	8	9	10	
66	63	63	59	60	62	60	60	61	60	61	1	95.4

Clustering experiments were conducted with RDBC. Using all 66 instances for clustering, RDBC returned ten clusters (see Figure 9.7). While some of the actual classes were rediscovered by RDBC (*s_tar*, *s_ire*, *s_histone*, and *s_secise2*), the clustering of class *s_secise1* appears to be hard. Whereas the mixed cluster C_8 made up of *s_secise1*, and *s_secise2* may be caused by a folding which those instances have in common, cluster C_6 is hard to explain. In addition the misplaced instances *s_secise1*(5) and *s_secise1*(10) in cluster C_2 would better fit into cluster C_5 .

To further evaluate the quality of RDBC's results, we used the computed clusters for prediction. Two experiments were conducted and the prediction accuracy was determined via a *leave-one-out*-approach. In both experiments class information was not used during the clustering stage. In the first experiment all instances except the test instance were available for building the

clusters. The test instance's class was then determined by a majority vote over instances of the most similar of the obtained clusters (according to the distance measure). In contrast to that, for the second setup we included the test instance during clustering. As in the first experiment, we employed majority voting for determining the test instance's class². Table 9.2 gives the accuracies obtained by our two experiments.

Table 9.2. RDBC's accuracies on the mRNA domain.

RDBC with test instance	RDBC without test instance
87.9%	83.3%

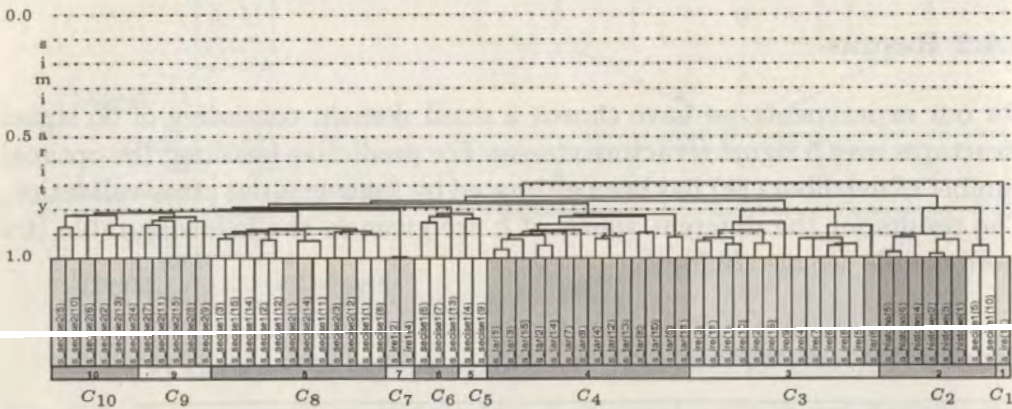


Fig. 9.7. Sample cluster hierarchy on the mRNA data. Shaded bars at the bottom mark the clusters found by RDBC.

9.7 Conclusion

First-order distance-based learning and clustering methods offer a lot of potential for applications involving complex and structured objects. For learning, distance-based “lazy” methods are known to often outperform their non-lazy counterparts; for clustering, a distance-based formulation is naturally at the heart of cluster quality. As shown in this chapter and the work on which

² In nine cases the majority vote resulted in a draw between two equally frequent classes. We resolved this by assigning the class of the instance most similar to our test instance. For this nearest-neighbor voting only instances of the cluster belonging to one of the two equally frequent classes were taken into account.

it is based, it is possible to define useful distance measures for first-order objects, making the strengths of distance-based methods available to first-order problems. Whenever an explicit generalization is not needed, the methods presented here are very strong contenders in terms of solution quality.

However, a few words of caution are also in order with respect to scalability. Firstly, instance-based learning methods must store all (or, when optimized variants are used, at least some) of their instances until classification time. Thus, the compression effect that can be obtained with explicit hypotheses is not achieved here. Secondly, computing distances between first-order objects is not as cheap an operation as computing a Euclidean distance (see [9.8] for details). While still faster than most generalization algorithms, this distance computation must be carried out at classification time, so such methods are not suitable if very rapid classification is a primary concern. As to the clustering methods, once the distance matrix is computed, they largely inherit the known computational properties of standard distance-based clustering algorithms; however, convergence of k -means may be somewhat slower since no exact cluster centers are computed.

In sum, all three of the methods described here are most suitable for domains with no more than several thousands of objects; more research is needed to scale them to the very large databases sometimes found in KDD.

References

- 9.1 G. Bisson. Conceptual Clustering in a First-Order Logic Representation. In *Proceedings of the Tenth European Conference on Artificial Intelligence*, pages 458–462. John Wiley and Sons, Chichester, 1992.
- 9.2 U. Bohnebeck, T. Horváth, and S. Wrobel. Term comparisons in first-order similarity measures. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, pages 65–79. Springer, Berlin, 1998.
- 9.3 W. Dillon and M. Goldstein. *Multivariate analysis*, pages 157–208. John Wiley and Sons, Chichester, 1984.
- 9.4 S. Džeroski, S. Schulze-Kremer, K. Heidtke, K. Siems, and D. Wettschereck. Diterpene structure elucidation from ^{13}C NMR spectra with machine learning. In N. Lavrač, E. Keravnou, and B. Zupan, editors, *Intelligent Data Analysis in Medicine and Pharmacology*, pages 207–225. Kluwer, Boston, 1997.
- 9.5 W. Emde and D. Wettschereck. Relational Instance-Based Learning. In *Proceedings of the Thirteen International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann, San Francisco, CA, 1996.
- 9.6 J. Hartigan. *Clustering Algorithms*, pages 58–73. John Wiley and Sons, Chichester, 1975.
- 9.7 T. Horváth, Z. Alexin, T. Gyimóthy, and S. Wrobel. Application of different learning methods to Hungarian part-of-speech tagging. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming*, pages 128–139. Springer, Berlin, 1999.
- 9.8 T. Horváth, S. Wrobel, and U. Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1/2): 53–80, 2001.

- 9.9 A. Hutchinson. Metrics on Terms and Clauses. In *Proceedings of the Ninth European Conference on Machine Learning*, pages 138–145. Springer, Berlin, 1997.
- 9.10 L. Kaufmann and P. J. Rousseeuw. Clustering by means of medoids. In Y. Dodge, editor, *Statistical Data Analysis based on the L_1 Norm*, pages 405–416. Elsevier, Amsterdam, 1987.
- 9.11 L. Kaufmann and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley and Sons, Chichester, 1990.
- 9.12 M. Kirsten and S. Wrobel. Relational Distance-Based Clustering. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, pages 261–270. Springer, Berlin, 1998.
- 9.13 M. Kirsten and S. Wrobel. Extending k -means clustering to first-order representations. In *Proceedings of the Tenth International Conference on Inductive Logic Programming*, pages 112–129. Springer, Berlin, 2000.
- 9.14 P. Mahalanobis. On the generalized distance in statistics. *Proceedings of the Indian National Institute Science*, 2: 49–55, Calcutta, 1936.
- 9.15 J. McQueen. Some methods of classification and analysis of multivariate observations. In *Proceedings of Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–293, 1967.
- 9.16 S.-H. Nienhuys-Cheng. Distance Between Herbrand Interpretations: A Measure for Approximations to a Target Concept. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 213–226. Springer, Berlin, 1997.
- 9.17 S.-H. Nienhuys-Cheng. Distances and limits on Herbrand interpretations. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, pages 250–260. Springer, Berlin, 1998.
- 9.18 J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. *Proceedings of the Eighth International Conference on Inductive Logic Programming*, pages 271–280. Springer, Berlin, 1998.
- 9.19 M. Sebag. Distance Induction in First Order Logic. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, pages 264–272. Springer, Berlin, 1997.
- 9.20 B. A. Shapiro and K. Zhang. Comparing Multiple RNA Secondary Structures Using Tree Comparisons. *Computer Applications in Biosciences*, 6(4): 309–318, 1990.
- 9.21 A. Srinivasan, S. Muggleton, and R. D. King. Comparing the use of background knowledge by inductive logic programming systems. In *Proceedings of the Fifth International Workshop on Inductive Logic Programming*, pages 199–230. Department of Computer Science, Katholieke Universiteit Leuven, Belgium, 1995.
- 9.22 E. Ukkonen. Algorithms for Approximate String Matching. *Information and Control*, 64: 100–118, 1985.
- 9.23 D. Wettschereck and D. Aha. Weighting Features. In *Proceedings of the First International Conference on Case-Based Reasoning*, pages 347–358. Springer, Berlin, 1995.
- 9.24 K. Zhang and D. Shasha. Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. *SIAM Journal on Computing*, 18(6): 1245–1262, 1989.
- 9.25 M. Zuker and P. Stiegler. Optimal Computer Folding of Large RNA Sequences Using Thermodynamics and Auxiliary Information. *Nucleic Acids Research*, 9(1): 133–148, 1981.

APPENDIX MANNOR01

```

leader 12461cam a2200601 a 4500
001 991001433729704436
005 20240613121242.0
008 050922s2005 nyua b 101 0 eng d
020 ##$a1595931805
020 ##$a9781595931801
029 1#$aAU@ $b000040174333
029 1#$aNZ1 $b10533817
035 ##$z1041272-o1cmu_inst
035 ##$a(OCOLC)61759050
035 ##$a(OCOLC)ocm61759050
040 ##$aCUIY $beng $cCUIY $dVDXCP $dC$A $dUBA $dOCLCA $dOCLCF $dOCLCO $dOCLCQ $dOCL $dOCLCO $dAU@ $dOCLCA $dOCLCQ $dOCLCO $dAAA $dOCLCO $dOCLCQ $dOCLCO
050 #4$aQ325.5 $b.I57 2005
111 2#$aInternational Conference on Machine Learning $n(22nd : $n2005 : $cBonn, Germany)
245 10$aProceedings, Twenty-second International Conference on Machine Learning / $cedited by Luc De Raedt and Stefan Wrobel.
246 14$aProceedings of the 22nd International Conference on Machine Learning
246 1#$iAt head of cover title: $aICML 2005, Bonn, Germany
246 1#$iAlso inonw as: $aICML '05
260 ##$a[New York, N.Y.?] : $bACM, $c02005.
300 ##$axxxviii, 1072 pages : $billustrations ; $c28 cm
336 ##$atext $btxt $2rdacontent
337 ##$aunmediated $bn $2rdamedia
338 ##$avolume $bnc $2rdacarrier
596 ##$a8 9
500 ##$a" ... annual conference of the International Machine Learning Society (IMLS) ..."--Page xiii
500 ##$aHeld on August 7-11, 2005.
504 ##$aIncludes bibliographical references and indexes.
505 0#$aExploration and apprenticeship learning in reinforcement learning / Pieter Abbeel / Andrew Y. Ng -- Active learning for hidden Markov models :
505 0#$aDietterich / Angelo Restificar / Alan Fern -- An efficient method for simplifying support vector machines / DucDung Nguyen / Tu Bao Ho -- Predi
650 #0$aMachine learning $vCongresses.
650 #6$aApprentissage automatique $vCongrès.
650 #7$aMachine learning $2fast
650 #7$aInteligência artificial (congressos) $2larpcal
650 #7$aAprendizado computacional (congressos) $2larpcal
655 #7$aConference papers and proceedings $2fast
710 2#$aAssociation for Computing Machinery.
710 2#$aInternational Machine Learning Society.
700 1#$aRaedt, Luc de, $d1964-
700 1#$aWrobel, Stefan.
901 ##$aocm61759050
904 41$uhttp://portal.acm.org/toc.cfm?id=1102351&coll=portal&dl=ACM&type=proceeding&idx=SERIES10714&part=Proceedings&WantType=Proceedings&title=ACM%20I
916 ##$a20071102
917 ##$a20070629
918 ##$aACQASSIST
919 ##$a20170411
920 ##$aCATASSIST
994 ##$aC0 $bPMC
999 ##$aQ325.5 .I58 2005 $wLC $c1 $iF156951 $lBY-REQUEST $mOFFSITE $rY $sY $tBOOK $u6/29/2007 $zGIFT-E&S
999 ##$aURL $wALPHANUM $c1 $i1041272-2001 $lWEB $mE-RESOURCE $rN $sY $tE-TEXT $u8/31/2016 FORM=MARC

```

Search anything

Library Catalog ▾



ADVANCED SEARCH

Chat with us



CONFERENCE PROCEEDING

Proceedings, Twenty-second International Conference on Machine Learning

International Conference on Machine Learning (22nd : 2005 : Bonn, Germany); Association for Computing Machinery; International Machine Learning Society.; Raedt, Luc de, 1964-; Wrobel, Stefan.

©2005

📄 Available at Offsite Repository BY-REQUEST (Q325.5 .I58 2005) >

TOP

FIND IN LIBRARY

LINKS

DETAILS

SEND TO

VIRTUAL BROWSE < BACK TO LOCATIONS

Find in Library

Please sign in to check if there are any request options.

🔑 Sign in

LOCATION ITEMS

Offsite Repository

Available , BY-REQUEST ; Q325.5 .I58 2005
(1 copy, 1 available, 0 requests)



Item in place

Sign in for loan information



Links

Report a Problem 📄 >

Signed in and still can't find what you're looking for? Let us know

Display source record 📄 >

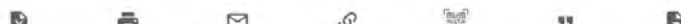
Details

Title	Proceedings, Twenty-second International Conference on Machine Learning
Creator	International Conference on Machine Learning (22nd : 2005 : Bonn, Germany) > Association for Computing Machinery. > International Machine Learning Society. > Raedt, Luc de, 1964- > Wrobel, Stefan. >
Subject	Apprentissage automatique > Aprendizado computacional (congressos) > Inteligência artificial (congressos) > Machine learning >
Genre	Congresses > Congrès > Conference papers and proceedings >
Contents	Exploration and apprenticeship learning in reinforcement learning / Pieter Abbeel / Andrew Y. Ng -- Active learning for hidden Markov models : objective functions and algorithms / Brigham Anderson / Andrew Moore -- Tempering for Bayesian C & RT / Nicos Angelopoulos / James Cussens -- Fast condensed nearest neighbor rule / Fabrizio Angiulli -- Predictive low-rank decomposition for kernel methods / Francis R. Bach / Michael I. Jordan -- Multi-way distributional clustering via pairwise interactions / Ron Bekkerman / Ran El-Yaniv / Andrew McCallum -- Error limiting reductions between classification tasks / Alina Beygelzimer / Varsha Dani / Tom Hayes / John Langford / Bianca Zadrozny -- Multi-instance tree learning / Hendrik Blockeel / David Page / Ashwin Srinivasan -- Action respecting embedding / Michael Bowling / Ali Ghodsi / Dana Wilkinson -- Clustering through ranking on manifolds / Markus Breitenbach / Gregory Z. Grudic -- Reducing overfitting in process model induction / Will Bridewell / Narges Bani Asadi / Pat Langle / Ljupco Todorovski -- Learning to rank using gradient descent / Chris Burges / Tal Shaked / Erin Renshaw / Ari Lazier / Matt Deeds / Nicole Hamilton / Greg Hullender -- Learning class-discriminative dynamic Bayesian networks / John Burge / Terran Lane -- Recognition and reproduction of gestures using a probabilistic framework combining PCA, ICA and HMM / Sylvain Calinon / Aude Billard -- Predicting probability distributions for surf height using an ensemble of mixture density networks / Michael Carney / Padraig Cunningham / Jim Dowling / Ciaran Lee -- Hedged learning : regret-minimization with learning experts / Yu-Han Chang / Leslie Pack Kaelbling -- Variational Bayesian image modelling / Li Cheng / Feng Jiao / Dale Schuurmans / Shaojun Wang -- Preference learning with Gaussian processes / Wei Chu / Zoubin Ghahramani -- New approaches to support vector ordinal regression / Wei Chu / S. Sathya Keerthi -- A general regression technique for learning transductions / Corinna Cortes / Mehryar Mohri / Jason Weston -- Learning to compete, compromise, and cooperate in repeated general-sum games / Jacob W. Crandall / Michael A. Goodrich -- Learning as search optimization : approximate large margin methods for structured prediction / Hal Daume III / Daniel Marcu -- Multimodal oriented discriminant analysis / Fernando De la Torre / Takeo Kanade -- A practical generalization of Fourier-based learning / Adam Drake / Dan Ventura -- Combining model-based and instance-based learning for first order transduction / Kurt Driessens / Soren Driessens -- Reinforcement learning with Gaussian processes /

Regression / Mark Herbster / John Elsom -- Reinforcement learning with Gaussian processes / Yaakov Engel / Shie Mannor / Ron Meir -- Experimental comparison between bagging and Monte Carlo ensemble classification / Roberto Eposito / Lorenza Saitta -- Supervised clustering with support vector machines / Thomas Finley / Thorsten Joachims -- Optimal assignment kernels for attributed molecular graphs / Holger Frohlich / Jorg K. Wegner / Florian Sieker / Andreas Zell -- Closed-form dual perturb and combine for tree-based models / Pierre Geurts / Louis Wehenkel -- Hierarchic Bayesian models for kernel learning / Mark Girolami / Simon Rogers -- Online feature selection for pixel classification / Karen Glöer / Damian Eads / James Theiler -- Learning strategies for story comprehension : a reinforcement learning approach / Eugene Grois / David C. Wilkins -- Near-optimal sensor placements in Gaussian processes / Carlos Guestrin / Andreas Krause / Ajit Paul Singh -- Robust one-class clustering using hybrid global and local search / Gunjan Gupta / Joydeep Ghosh -- Statistical and computational analysis of locality preserving projection / Xiaofei He / Deng Cai / Wanti Min -- Intrinsic dimensionality estimation of submanifolds in $\mathbb{R}^d$ / Matthias Hein / Jean-Yves Audibert -- Bayesian hierarchical clustering / Katherine A. Heller / Zoubin Ghahramani -- Online learning over graphs / Mark Herbster / Massimiliano Pontil / Lisa Wainer -- Adapting two-class support vector classification methods to many class problems / Simon I. Hill / Arnaud Doucet -- A martingale framework for concept change detection in time-varying data streams / Shen-Shyang Ho -- Multi-class protein fold recognition using adaptive codes / Eugene Ie / Jason Weston / William Stafford Noble / Christina Leslie -- Learning approximate preconditions for methods in hierarchical plans / Oktay Ilghami / Hector Munoz-Avila / Dana S. Nau / David W. Aha -- Evaluating machine learning for information extraction / Neil Ireson / Fabio Ciravegna / Mary Elaine Califf / Dayne Freitag / Nicholas Kushmerick / Alberto Lavelli -- Learn to weight terms in information retrieval using category information / Rong Jin / Joyce Y. Chai / Luo Si -- A smoothed boosting algorithm using probabilistic output codes / Rong Jin / Jian Zhang -- Efficient discriminative learning of Bayesian network classifier via boosted augmented naive Bayes / Yushi Jing / Vladimir Pavlovic / James M. Rehg -- A support vector method for multivariate performance measures / Thorsten Joachims -- Error bounds for correlation clustering / Thorsten Joachims / John Hopcroft -- Interactive learning of mappings from visual percepts to actions / Sebastien Jodogne / Justus H. Piater -- A causal approach to hierarchical decomposition of factored MDPs / Anders Jonsson / Andrew Barto -- A comparison of tight generalization error bounds / Matti Kaariainen / John Langford -- Generalized LARS as an effective feature selection tool for text classification with SVMs / S. Sathya Keerthi -- Ensembles of biased classifiers / Rinat Khossainov / Andreas Hess / Nicholas Kushmerick -- Computational aspects of Bayesian partition models / Mikko Koivisto / Kismet Sood -- Learning the structure of Markov logic networks / Stanley Kok / Pedro Domingos -- Using additive expert ensembles to cope with concept drift / Jeremy Z. Kolter / Marcus A. Maloof -- Semi-supervised graph clustering : a kernel approach / Brian Kulis / Sugato Basu / Inderjit Dhillon / Raymond Mooney -- A brain computer interface with online feedback based on magnetoencephalography / Thomas Navin Lal / Michael Schroder / N. Jeremy Hill / Hubert Preissl / Thilo Hinterberger / Jurgen Mellinger / Martin Bogdan / Wolfgang Rosenstiel / Thomas Hofmann / Niels Birbaumer / Bernhard Scholkopf -- Relating reinforcement learning performance to classification performance / John Langford / Bianca Zadrozny -- PAC-Bayes risk bounds for sample-compressed Gibbs classifiers / Francois Laviolette / Mario Marchand -- Heteroscedastic Gaussian process regression / Quoc V. Le / Alex J. Smola / Stephane Canu -- Predicting relative performance of classifiers from samples / Rui Leite / Pavel Brazdil -- Logistic regression with an auxiliary data source / Xuejun Liao / Ya Xue / Lawrence Carin -- Predicting protein folds with structural repeats using a chain graph model / Eric P. Xing / Yan Liu / Jaime Carbonell -- Unsupervised evidence integration / Philip M. Long / Vinay Varadan / Sarah Gilman / Mark Treshock / Rocco A. Servedio -- Naive Bayes models for probability estimation / Daniel Lowd / Pedro Domingos -- ROC confidence bands : an empirical evaluation / Sofus A. Macskassy / Foster Provost / Saharon Rosset -- Modeling word burstiness using the Dirichlet distribution / Rasmus E. Madsen / David Kauchak / Charles Elkan -- Proto-value functions : developmental reinforcement learning / Sridhar Mahadevan -- The cross entropy method for classification / Dori Peleg / Shie Mannor / Reuven Rubinstein -- Bounded real-time dynamic programming : RTDP with monotone upper bounds and performance guarantees / Geoffrey J. Gordon / Maxim Likhachev / H. Brendan McMahan -- Comparing clusterings -- an axiomatic view / Marina Meila -- Weighted decomposition kernels / Sauro Menchetti / Fabrizio Costa / Paolo Frasconi -- High speed obstacle avoidance using monocular vision and reinforcement learning / Jeff Michels / Ashutosh Saxena / Andrew Y. Ng -- Dynamic preferences in multi-criteria reinforcement learning / Sriaram Natarajan / Prasad Tadepalli -- Learning first-order probabilistic models with combining rules / Eric Altendorf / Prasad Tadepalli / Sriaram Natarajan / Thomas G. Dietterich / Angelo Restifcar / Alan Fern -- An efficient method for simplifying support vector machines / DucDung Nguyen / Tu Bao Ho -- Predicting good probabilities with supervised learning / Alexandru Niculescu-Mizil / Rich Caruana -- Recycling data for multi-agent learning / Santiago Ontanon / Enric Plaza -- A graphical model for chord progressions embedded in a psychoacoustic space / Jean-Francois Palement / David Barber / Samy Bengio / Douglas Eck -- Q-learning of sequential attention for visual object recognition from informative local descriptors / Gerald Fritz / Lucas Paletta / Christin Seifert -- Discriminative versus generative parameter and structure learning of Bayesian network classifiers / Jeff Bilmes / Franz Pernkopf -- Optimizing abstaining classifiers using ROC analysis / Tadeusz Pietraszek -- Independent subspace analysis using geodesic spanning trees / Barnabas Poczos / Andras Lorincz -- A model for handling approximate, noisy or incomplete labeling in text classification / Ganesh Ramakrishnan / Krishna Prasad Chitrapura / Raghu Krishnapuram / Pushpak Bhattacharyya -- Healing the relevance vector machine through augmentation / Carl Edward Rasmussen / Joaquin Quinero-Candela -- Supervised versus multiple instance learning : an empirical comparison / Soumya Ray / Mark Craven -- Generalized skewing for functions with continuous and nominal attributes / Soumya Ray / David Page -- Fast maximum margin matrix factorization for collaborative prediction / Jasson D.M. Rennie / Nati Srebro -- Coarticulation : an approach for generating concurrent plans in Markov decision processes / Khashayar Rohanimanesh / Sridhar Mahadevan

Other title	Proceedings of the 22nd International Conference on Machine Learning At head of cover title: ICML 2005, Bonn, Germany Also inonw as: ICML '05
Related titles	Available in other form: Proceedings, twenty-second International Conference on Machine Learning >
Publisher	New York, N.Y. : ACM
Creation Date	©2005
Format	xxviii, 1072 pages : illustrations ; 28 cm
Bibliography Note	Includes bibliographical references and indexes.
General Note	"... annual conference of the International Machine Learning Society (IMLS) ..."--Page xiii Held on August 7-11, 2005.
ISBN	1595931805 9781595931801
Source	Library Catalog

Send to



Virtual Browse

gs

1

Machine learning: proceedings of the nineteenth ...

c2002

Proceedings: Twenty-First International Conference on Machine- ...

@2004

Proceedings, Twenty-second International Conference on ...

@2005

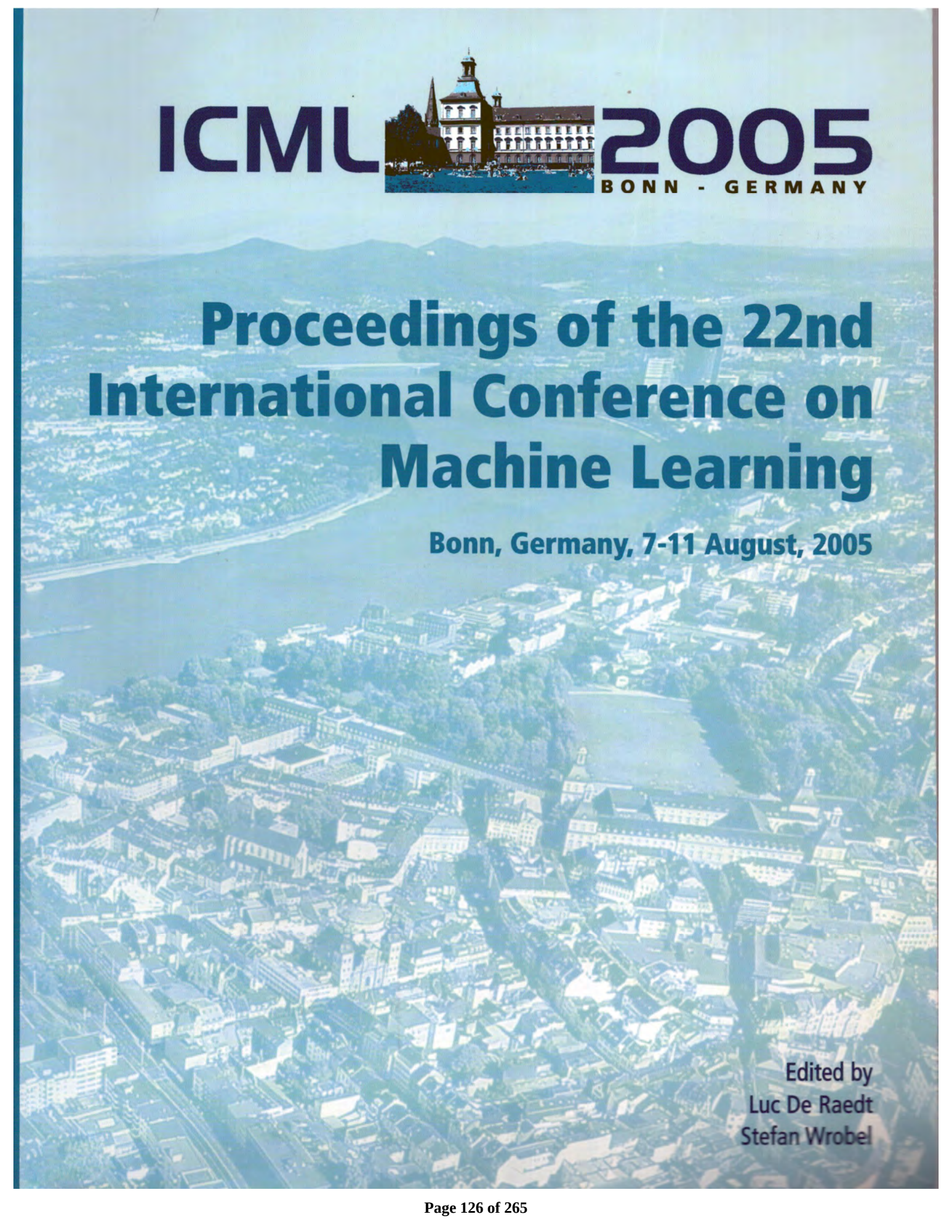
Proceedings of the 23rd International Conference on Machine ...

@2006

Proceedings: Twenty fourth International Conference on Machine ...

c2007

APPENDIX MANNOR02



ICML  **2005**
BONN - GERMANY

Proceedings of the 22nd International Conference on Machine Learning

Bonn, Germany, 7-11 August, 2005

Edited by
Luc De Raedt
Stefan Wrobel

**Proceedings of the 22nd International
Conference on Machine Learning**

ICML

F156951

Luc De Raet
Stefan Wrobel

ENGR&SCI

Q

325.5

.158

2005



Proceedings of the 22nd International Conference on Machine Learning

This proceedings contains a collection of original papers presented at the Twenty-second International Conference on Machine Learning (ICML-2005) held in Bonn, Germany, August 7-11, 2005. The annual ICML conference gathers scientists interested in the latest advances in the science of machine learning. The papers in this volume have been carefully selected by the program committee. They address theoretical issues, experimental investigation as well as applications, and cover a wide range of topics from all subfields of machine learning, including support vector machines, bayesian methods, evaluation of learning algorithms, reinforcement learning, statistical relational learning, unsupervised learning, boosting, etc. The selected papers are relevant to those in academia and industry keen on practicing and extending the state of the art in machine learning.

ISBN 1-59593-180-5



ICML 2005,

Twenty-Second International
Conference on Machine Learning

Edited by Luc De Raedt and Stefan Wrobel

Proceedings, Twenty-Second International Conference on Machine Learning

Edited by Luc De Raedt and Stefan Wrobel

Q
325.5
• I58
2005

per Copyright © 2005

These materials are owned by their respective copyright owners.
All rights reserved.

ISBN 1-59593-180-5

Printed in the United States of America

Table of Contents

Preface.....	xiii
ICML 2005 Organization	xv
Program Committee.....	xvi
External Reviewers.....	xvii
Board of the International Machine Learning Society 2005	xviii
Sponsors.....	xix
Invited Talks	xxiii
Workshop and Tutorial Summaries	xxix

ICML 2005 Papers

Exploration and Apprenticeship Learning in Reinforcement Learning	1
<i>Pieter Abbeel, Andrew Y. Ng</i>	
Active Learning for Hidden Markov Models: Objective Functions and Algorithms.....	9
<i>Brigham Anderson, Andrew W. Moore</i>	
Tempering for Bayesian C&RT	17
<i>Nicos Angelopoulos, James Cussens</i>	
Fast Condensed Nearest Neighbor Rule	25
<i>Fabrizio Angiulli</i>	
Predictive low-rank decomposition for kernel methods.....	33
<i>Francis R. Bach, Michael I. Jordan</i>	
Multi-Way Distributional Clustering via Pairwise Interactions	41
<i>Ron Bekkerman, Ran El-Yaniv, Andrew McCallum</i>	
Error Limiting Reductions Between Classification Tasks.....	49
<i>Alina Beygelzimer, Varsha Dani, Tom Hayes, John Langford, Bianca Zadrozny</i>	
Multi-Instance Tree Learning	57
<i>Hendrik Blockeel, David Page, Ashwin Srinivasan</i>	
Action Respecting Embedding	65
<i>Michael Bowling, Ali Ghodsi, Dana Wilkinson</i>	
Clustering Through Ranking On Manifolds.....	73
<i>Markus Breitenbach, Gregory Z. Grudic</i>	
Reducing Overfitting in Process Model Induction.....	81
<i>Will Bridewell, Narges Bani Asani, Pat Langley, Ljupco Todorovski</i>	
Learning to Rank using Gradient Descent.....	89
<i>Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, Greg Hullender</i>	
Learning Class-Discriminative Dynamic Bayesian Networks.....	97
<i>John Burge, Terran Lane</i>	

University Libraries
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM	105
<i>Sylvain Calinon, Aude Billard</i>	
Predicting Probability Distributions for Surf Height Using an Ensemble of Mixture Density Networks.....	113
<i>Michael Carney, Padraig Cunningham, Jim Dowling, Ciaran Lee</i>	
Hedged learning: Regret minimization with learning experts	121
<i>Yu-Han Chang, Leslie Pack Kaelbling</i>	
Variational Bayesian Image Modelling	129
<i>Li Cheng, Feng Jiao, Dale Schuurmans, Shaojun Wang</i>	
Preference Learning with Gaussian Processes	137
<i>Wei Chu, Zoubin Ghahramani</i>	
New Approaches to Support Vector Ordinal Regression.....	145
<i>Wei Chu, S. Sathya Keerthi</i>	
A General Regression Technique for Learning Transductions	153
<i>Corinna Cortes, Mehryar Mohri, Jason Weston</i>	
Learning to Compete, Compromise, and Cooperate in Repeated General-Sum Games	161
<i>Jacob W. Crandall, Michael A. Goodrich</i>	
Learning as Search Optimization: Approximate Large Margin Methods for Structured Prediction	169
<i>Hal Daume III, Daniel Marcu</i>	
Multimodal Oriented Discriminant Analysis	177
<i>Fernando De la Torre, Takeo Kanade</i>	
A Practical Generalization of Fourier-based Learning	185
<i>Adam Drake, Dan Ventura</i>	
Combining Model-Based and Instance-Based Learning for First Order Regression	193
<i>Kurt Driessens, Saso Dzeroski</i>	
Reinforcement learning with Gaussian processes	201
<i>Yaakov Engel, Shie Mannor, Ron Meir</i>	
Experimental Comparison between Bagging and Monte Carlo Ensemble Classification	209
<i>Roberto Esposito, Lorenza Saitta</i>	
Supervised Clustering with Support Vector Machines.....	217
<i>Thomas Finley, Thorsten Joachims</i>	
Optimal Assignment Kernels For Attributed Molecular Graphs	225
<i>Holger Fröhlich, Jörg K. Wegner, Florian Sieker, Andreas Zell</i>	
Closed-form dual perturb and combine for tree-based models	233
<i>Pierre Geurts, Louis Wehenkel</i>	
Hierarchical Bayesian Models for Kernel Learning.....	241
<i>Mark Girolami, Simon Rogers</i>	

Online Feature Selection for Pixel Classification	249
<i>Karen Glocer, Damian Eads, James Theiler</i>	
Learning Strategies for Story Comprehension: A Reinforcement Learning Approach	257
<i>Eugene Grois, David C. Wilkins</i>	
Near-Optimal Sensor Placements in Gaussian Processes	265
<i>Carlos Guestrin, Andreas Krause, Ajit Paul Singh</i>	
Robust One-Class Clustering Using Hybrid Global and Local Search	273
<i>Gunjan Gupta, Joydeep Ghosh</i>	
Statistical and Computational Analysis of Locality Preserving Projection	281
<i>Xiaofei He, Deng Cai, Wanli Min</i>	
Intrinsic Dimensionality Estimation of Submanifolds in $\Re^d$	289
<i>Matthias Hein, Jean-Yves Audibert</i>	
Bayesian Hierarchical Clustering	297
<i>Katherine A. Heller, Zoubin Ghahramani</i>	
Online Learning over Graphs	305
<i>Mark Herbster, Massimiliano Pontil, Lisa Wainer</i>	
Adapting Two-Class Classification Methods to Many Class Problems	313
<i>Simon I. Hill, Arnaud Doucet</i>	
A Martingale Framework for Concept Change Detection in Time-Varying Data Streams	321
<i>Shen-Shyang Ho</i>	
Multi-Class protein fold detection using adaptive codes	329
<i>Eugene Ie, Jason Weston, William Stafford Noble, Christina Leslie</i>	
Learning Approximate Preconditions for Methods in Hierarchical Plans	337
<i>Okhtay Ilghami, Héctor Muñoz-Avila, Dana S. Nau, David W. Aha</i>	
Evaluating Machine Learning for Information Extraction	345
<i>Neil Ireson, Fabio Ciravegna, Mary Elaine Califf, Dayne Freitag, Nicholas Kushmerick, Alberto Lavelli</i>	
Learn to Weight Terms in Information Retrieval Using Category Information	353
<i>Rong Jin, Joyce Y. Chai, Luo Si</i>	
A Smoothed Boosting Algorithm Using Probabilistic Output Codes	361
<i>Rong Jin, Jian Zhang</i>	
Efficient discriminative learning of Bayesian network classifier via Boosted Augmented Naive Bayes	369
<i>Yushi Jing, Vladimir Pavlovic, James M. Rehg</i>	
A Support Vector Method for Multivariate Performance Measures	377
<i>Thorsten Joachims</i>	
Error Bounds for Correlation Clustering	385
<i>Thorsten Joachims, John Hopcroft</i>	

Interactive Learning of Mappings from Visual Percepts to Actions	393
<i>Sébastien Jodogne, Justus H. Piater</i>	
A Causal Approach to Hierarchical Decomposition of Factored MDPs.....	401
<i>Anders Jonsson, Andrew Barto</i>	
A Comparison of Tight Generalization Error Bounds	409
<i>Matti Kääriäinen, John Langford</i>	
Generalized LARS as an Effective Feature Selection Tool for Text Classification With SVMs	417
<i>S. Sathiya Keerthi</i>	
Ensembles of Biased Classifiers.....	425
<i>Rinat Khoussainov, Andreas Heß, Nicholas Kushmerick</i>	
Computational Aspects of Bayesian Partition Models	433
<i>Mikko Koivisto, Kismat Sood</i>	
Learning the Structure of Markov Logic Networks.....	441
<i>Stanley Kok, Pedro Domingos</i>	
Using Additive Expert Ensembles to Cope with Concept Drift	449
<i>Jeremy Kolter, Marcus Maloof</i>	
Semi-supervised Graph Clustering: A Kernel Approach	457
<i>Brian Kulis, Sugato Basu, Inderjit S. Dhillon, Raymond J. Mooney</i>	
A Brain Computer Interface with Online Feedback based on Magnetoencephalography.....	465
<i>Thomas Navin Lal, Michael Schröder, N. Jeremy Hill, Hubert Preissl, Thilo Hinterberger, Jürgen Mellinger, Martin Bogdan, Wolfgang Rosenstiel, Thomas Hofmann, Niels Birbaumer, Bernhard Schölkopf</i>	
Relating Reinforcement Learning Performance to Classification Performance	473
<i>John Langford, Bianca Zadrozny</i>	
PAC-Bayes Risk Bounds for Sample-Compressed Gibbs Classifiers	481
<i>François Laviolette, Mario Marchand</i>	
Heteroscedastic Gaussian Process Regression.....	489
<i>Quoc V. Le, Alex J. Smola, Stéphane Canu</i>	
Predicting Relative Performance of Classifiers from Samples	497
<i>Rui Leite, Pavel Brazdil</i>	
Logistic Regression with an Auxiliary Data Source	505
<i>Xuejun Liao, Ya Xue, Lawrence Carin</i>	
Predicting Protein Folds with Structural Repeats Using a Chain Graph Model.....	513
<i>Yan Liu, Eric P. Xing, Jaime Carbonell</i>	
Unsupervised Evidence Integration	521
<i>Philip M. Long, Vinay Varadan, Sarah Gilman, Mark Treshock, Rocco A. Servedio</i>	
Naive Bayes Models for Probability Estimation.....	529
<i>Daniel Lowd, Pedro Domingos</i>	

ROC Confidence Bands : An Empirical Evaluation	537
<i>Sofus A. Macskassy, Foster Provost, Saharon Rosset</i>	
Modeling Word Burstiness Using the Dirichlet Distribution	545
<i>Rasmus E. Madsen, David Kauchak, Charles Elkan</i>	
Proto-Value Functions: Developmental Reinforcement Learning	553
<i>Sridhar Mahadevan</i>	
The cross entropy method for classification	561
<i>Shie Mannor, Dori Peleg, Reuven Y. Rubinstein</i>	
Bounded Real-Time Dynamic Programming: RTDP with monotone upper bounds and performance guarantees.....	569
<i>H. Brendan McMahan, Maxim Likhachev, Geoffrey J. Gordon</i>	
Comparing Clusterings - An Axiomatic View.....	577
<i>Marina Meila</i>	
Weighted Decomposition Kernels	585
<i>Sauro Menchetti, Fabrizio Costa, Paolo Frasconi</i>	
High Speed Obstacle Avoidance using Monocular Vision and Reinforcement learning	593
<i>Jeff Michels, Ashutosh Saxena, Andrew Y. Ng</i>	
Dynamic Preferences in Multi-Criteria Reinforcement Learning	601
<i>Sriiraaam Natarajan, Prasad Tadepalli</i>	
Learning First-Order Probabilistic Models with Combining Rules	609
<i>Sriiraaam Natarajan, Prasad Tadepalli, Eric Altendorf, Thomas G. Dietterich, Alan Fern, Angelo Restificar</i>	
An Efficient Method for Simplifying Support Vector Machines	617
<i>DucDung Nguyen, Tu Bao Ho</i>	
Predicting Good Probabilities With Supervised Learning	625
<i>Alexandru Niculescu-Mizil, Rich Caruana</i>	
Recycling Data for Multi-Agent Learning.....	633
<i>Santiago Ontañón, Enric Plaza</i>	
A Graphical Model for Chord Progressions Embedded in a Psychoacoustic Space	641
<i>Jean-François Paiement, Douglas Eck, Samy Bengio, David Barber</i>	
Q-Learning of Sequential Attention for Visual Object Recognition from Informative Local Descriptors	649
<i>Lucas Paletta, Gerald Fritz, Christin Seifert</i>	
Discriminative versus Generative Parameter and Structure Learning of Bayesian Network Classifiers.....	657
<i>Franz Pernkopf, Jeff Bilmes</i>	
Optimizing Abstaining Classifiers using ROC Analysis	665
<i>Tadeusz Pietraszek</i>	
Independent Subspace Analysis Using Geodesic Spanning Trees	673
<i>Barnabás Póczos, András Lörincz</i>	

A Model for Handling Approximate, Noisy or Incomplete Labeling in Text Classification.....	681
<i>Ganesh Ramakrishnan, Krishna Prasad Chitrapura, Raghu Krishnapuram, Pushpak Bhattacharyya</i>	
Healing the Relevance Vector Machine through Augmentation.....	689
<i>Carl Edward Rasmussen, Joaquin Quiñero-Candela</i>	
Supervised versus Multiple Instance Learning: An Empirical Comparison.....	697
<i>Soumya Ray, Mark Craven</i>	
Generalized Skewing for Functions with Continuous and Nominal Attributes.....	705
<i>Soumya Ray, David Page</i>	
Fast Maximum Margin Matrix Factorization for Collaborative Prediction.....	713
<i>Jason D. M. Rennie, Nati Srebro</i>	
Coarticulation: An Approach for Generating Concurrent Plans in Markov Decision Processes.....	721
<i>Khashayar Rohanimanesh, Sridhar Mahadevan</i>	
Why Skewing Works: Learning Difficult Boolean Functions with Greedy Tree Learners.....	729
<i>Bernard Rosell, Lisa Hellerstein, Soumya Ray, David Page</i>	
Integer Linear Programming Inference for Conditional Random Fields.....	737
<i>Dan Roth, Wen-Tau Yih</i>	
Learning Hierarchical Multi-Category Text Classification Models.....	745
<i>Juho Rousu, Craig Saunders, Sandor Szedmak, John Shawe-Taylor</i>	
Expectation Maximization Algorithms for Conditional Likelihoods.....	753
<i>Jarkko Salojärvi, Kai Puolamäki, Samuel Kaski</i>	
Estimating and computing density based distance metrics.....	761
<i>Sajama, Alon Orlitsky</i>	
Supervised dimensionality reduction using mixture models.....	769
<i>Sajama, Alon Orlitsky</i>	
Object Correspondence as a Machine Learning Problem.....	777
<i>Bernhard Schölkopf, Florian Steinke, Volker Blanz</i>	
Analysis and Extension of Spectral Methods for Nonlinear Dimensionality Reduction.....	785
<i>Fei Sha, Lawrence K. Saul</i>	
Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision.....	793
<i>Amnon Shashua, Tamir Hazan</i>	
Fast Inference and Learning in Large-State-Space HMMs.....	801
<i>Sajid M. Siddiqi, Andrew W. Moore</i>	
New D-Separation Identification Results for Learning Continuous Latent Variable Models.....	809
<i>Ricardo Silva, Richard Scheines</i>	
Identifying Useful Subgoals in Reinforcement Learning by Local Graph Partitioning.....	817
<i>Özgür Simsek, Alicia P. Wolfe, Andrew G. Barto</i>	

Beyond the Point Cloud: from Transductive to Semi-supervised Learning	825
<i>Vikas Sindhwani, Partha Niyogi, Mikhail Belkin</i>	
Active Learning for Sampling in Time-Series Experiments With Application to Gene Expression Analysis....	833
<i>Rohit Singh, Nathan Palmer, David Gifford, Bonnie Berger, Ziv Bar-Joseph</i>	
Compact approximations to Bayesian predictive distributions	841
<i>Edward Snelson, Zoubin Ghahramani</i>	
Large Scale Genomic Sequence SVM Classifiers	849
<i>Sören Sonnenburg, Gunnar Rätsch, Bernhard Schölkopf</i>	
A Theoretical Analysis of Model-Based Interval Estimation	857
<i>Alexander L. Strehl, Michael L. Littman</i>	
Explanation-Augmented SVM: an Approach to Incorporating Domain Knowledge into SVM Learning	865
<i>Qiang Sun, Gerald DeJong</i>	
Unifying the Error-Correcting and Output-Code AdaBoost within the Margin Framework	873
<i>Yijun Sun, Sinisa Todorovic, Jian Li, Dapeng Wu</i>	
Finite Time Bounds for Sampling Based Fitted Value Iteration	881
<i>Csaba Szepesvári, Rémi Munos</i>	
TD(λ) Networks: Temporal-Difference Networks with Eligibility Traces	889
<i>Brian Tanner, Richard S. Sutton</i>	
Learning Structured Prediction Models: A Large Margin Approach	897
<i>Ben Taskar, Vassil Chatalbashev, Daphne Koller, Carlos Guestrin</i>	
Learning Discontinuities with Products-of-Sigmoids for Switching between Local Models	905
<i>Marc Toussaint, Sethu Vijayakumar</i>	
Core Vector Regression for Very Large Regression Problems	913
<i>Ivor W. Tsang, James T. Kwok, Kimo T. Lai</i>	
Propagating Distributions on a Hypergraph by Dual Information Regularization	921
<i>Koji Tsuda</i>	
Hierarchical Dirichlet Model for Document Classification	929
<i>Sriharsha Veeramachaneni, Diego Sosa, Paolo Avesani</i>	
Implicit Surface Modelling as an Eigenvalue Problem	937
<i>Christian Walder, Olivier Chapelle, Bernhard Schölkopf</i>	
New Kernels for Protein Structural Motif Discovery and Function Classification	945
<i>Chang Wang, Stephen D. Scott</i>	
Exploiting Syntactic, Semantic and Lexical Regularities in Language Modeling via Directed Markov Random Fields	953
<i>Shaojun Wang, Shaomin Wang, Russell Greiner, Dale Schuurmans, Li Cheng</i>	
Bayesian Sparse Sampling for On-line Reward Optimization	961
<i>Tao Wang, Daniel J. Lizotte, Michael Bowling, Dale Schuurmans</i>	

Learning Predictive Representations from a History	969
<i>Eric Wiewiora</i>	
Incomplete-Data Classification using Logistic Regression	977
<i>David Williams, Xuejun Liao, Ya Xue, Lawrence Carin</i>	
Learning Predictive State Representations in Dynamical Systems Without Reset.....	985
<i>Britton Wolfe, Michael R. James, Satinder Singh</i>	
Linear Asymmetric Classifier for Cascade Detectors	993
<i>Jianxin Wu, Matthew D. Mullin, James M. Rehg</i>	
Building Sparse Large Margin Classifiers.....	1001
<i>Mingrui Wu, Bernhard Schölkopf, Gökhan Bakir</i>	
Dirichlet Enhanced Relational Learning.....	1009
<i>Zhao Xu, Volker Tresp, Kai Yu, Shipeng Yu, Hans-Peter Kriegel</i>	
Learning Gaussian Processes from Multiple Tasks.....	1017
<i>Kai Yu, Volker Tresp, Anton Schwaighofer</i>	
Augmenting Naive Bayes for Ranking	1025
<i>Harry Zhang, Liangxiao Jiang, Jiang Su</i>	
A New Mallows Distance Based Metric For Comparing Clusterings	1033
<i>Ding Zhou, Jia Li, Hongyuan Zha</i>	
Learning from Labeled and Unlabeled Data on a Directed Graph	1041
<i>Dengyong Zhou, Jiayuan Huang, Bernhard Schölkopf</i>	
2D Conditional Random Fields for Web Information Extraction	1049
<i>Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, Wei-Ying Ma</i>	
Harmonic mixtures: combining mixture models and graph-based methods for inductive and scalable semi-supervised learning.....	1057
<i>Xiaojin Zhu, John Lafferty</i>	
Large Margin Non-Linear Embedding	1065
<i>Alexander Zien, Joaquin Quiñero-Candela</i>	

The cross entropy method for classification

Shie Mannor

Dept. of Electrical and Computer Engineering, McGill University, Montreal, Canada

SHIE@ECE.MCGILL.CA

Dori Peleg

Dept. of Electrical Engineering, Technion Institute of Technology, Haifa 32000, Israel

DORIP@TECHUNIX.TECHNION.AC.IL

Reuven Rubinstein

Faculty of Industrial Engineering and Management, Technion Institute of Technology, Haifa 32000, Israel

IERRR01@IE.TECHNION.AC.IL

Abstract

We consider support vector machines for binary classification. As opposed to most approaches we use the number of support vectors (the " L_0 norm") as a regularizing term instead of the L_1 or L_2 norms. In order to solve the optimization problem we use the cross entropy method to search over the possible sets of support vectors. The algorithm consists of solving a sequence of efficient linear programs. We report experiments where our method produces generalization errors that are similar to support vector machines, while using a considerably smaller number of support vectors.

1. Introduction

Many classification algorithms are focused on finding a classifier which minimizes the sum of the training error (or some related cost functional) and a complexity penalty. The intuition is that a classifier which is too complex might be tailored to the data and result in overfitting. The exact choice of a complexity penalty is often motivated by generalization abilities of the resulting classifier. However, this choice is also restricted by guaranteeing that the optimization problem is tractable. In Support Vector Machines (SVMs) one typically considers an L_2 complexity term (e.g., Cristianini & Shawe-Taylor, 2000; Vapnik, 1998), which can be justified in terms of minimizing the margin. This yields a quadratic optimization problem which can be solved remarkably efficiently up to a few thousand samples (see Platt, 1998; Keerthi et al., 2001). The main problem with using the L_2

complexity term is that it does not encourage sparse solutions. That is, the resulting classifier involves a significant fraction of the training points. Moreover, it was observed (e.g., Downs et al., 2001) that the solution produced by SVMs can even be made more sparse without changing the resulting classifier. Other kernel-based methods (e.g., Schölkopf & Smola, 2002) attempt to guarantee that the obtained classifier is sparse using different ideas. Most notably, the ν -SVM formulation (Chang & Lin, 2002) attempts to control the fraction of Support Vectors (SVs) by introducing a parameter used to tradeoff the number of SVs with the size of errors. Still, regularization is done using L_2 norm. In the context of classification, using the L_1 norm for regularization was used in Bradley and Mangasarian (1998) and Zhu et al. (2003).

A classifier with a small number of SVs is termed "sparse". Such classifiers enjoy several merits. First, sparse classifiers typically have better generalization error (e.g., Floyd & Warmuth, 1995 and more recently within the luckiness framework of Herbrich and Williamson (2002) and via compression bounds of Herbrich, 2002). Second, such classifiers are considered more robust to noise in the data (in the context of L_1 classification see Zhu et al., 2003, for a discussion). Third, the resulting classifier can be computed more quickly on test data, as it involves less terms. Improving the complexity of classification (over the test data) was considered Burges (1996); Burges and Schölkopf (1997); Downs et al. (2001), who considered simplifying a classifier obtained from solving an SVM. Typically, some SVs are discarded with little (or no, as in Downs et al., 2001) effect on the resulting classifier. Our approach is different. Instead of trying to simplify a solution obtained from solving an SVM, we directly solve an optimization problem aimed at producing a sparse classifier.

In this paper we consider the classical soft SVM prob-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

lem, but regularize using the number of SVs (the so-called L_0 norm). As a result we obtain a discontinuous and non-convex optimization problem. We formulate the problem as a search problem where one looks for the set of SVs. We apply the Cross Entropy (CE) method for efficiently searching for the set of SVs. The CE method provides a sequence of candidate sets of SVs. For each such candidate set, finding the weights of the SVs amounts to solving a reasonably sized linear program provided that the size of the candidate set is not too large. The CE method is a state-of-the-art heuristic method for solving combinatorial optimization problems and continuous multi-extremal problems (Rubinstein & Kroese, 2004). The basic idea behind the CE method is to transform a combinatorial optimization problem into a stochastic optimization problem, and then solve the stochastic optimization method adaptively. In many problems of practical interest, such as the travelling salesperson problem and the Max-cut problem, the CE method was observed to quickly reach a solution close to optimal (for a discussion and examples, see Rubinstein & Kroese, 2004). We note that it is not claimed that the CE method is capable of solving every instance of some NP-hard problem efficiently, but rather that it typically performs well. For a convergence proof of the CE method see Rubinstein and Kroese. (2004). We will use the CE-method to search over possible subsets of SVs.

We are concerned with the classical classification setup, i.e., given a training set of pairs $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^n$, where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ are input patterns and $y^{(i)} \in \{-1, 1\}$ are the corresponding labels, find a classifier $f(\mathbf{x})$ which will classify new patterns with as few errors as possible. We are also given a kernel $K: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$. For a vector $\alpha \in \mathbb{R}^n$ let $\|\alpha\|_0 = \sum_{i=1}^n 1_{\{\alpha_i \neq 0\}}$ denote the " L_0 norm" that counts how many elements are not zero. The objective of this paper is to solve the problem

$$\begin{aligned} & \text{minimize} \quad \|\alpha\|_0 + C \sum_{i=1}^n \phi(y_i f(\mathbf{x}_i)) \\ & \text{subject to} \quad 0 \leq \alpha \leq C, \end{aligned} \quad (1)$$

with variables $\alpha \in \mathbb{R}^n$ and $b \in \mathbb{R}$, and where the hinge loss $\phi: \mathbb{R} \rightarrow \mathbb{R}_+$ is defined as $\phi(z) = 1 - z$ if $z \leq 1$, and zero otherwise; C is a hyperparameter; and the classifier f is defined by:

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

The remainder of the paper is organized as follows. In Section 2 we review the CE method and provide the

essential details needed here. In Section 3 we describe the classification algorithm based on the CE method. In Section 4 we report experimental results with an artificial data set and with several common data sets. In Section 5 we discuss the results and offer some conclusions.

2. The Cross Entropy Method

In this section we recapitulate the principles of the CE method. The reader is referred to Rubinstein and Kroese. (2004); de-Boer et al. (2005) and references therein for context, extensions, and applications. As mentioned, the basic idea behind the CE method is to transform the original (combinatorial) optimization problem to an associated stochastic optimization problem, and then to tackle the stochastic problem efficiently by an adaptive sampling algorithm. By doing so one constructs a random sequence of solutions which converges (probabilistically) to the optimal or at least a reasonable solution. Once the associated stochastic optimization is defined, the CE method alternates the following two phases:

1. Generation of a sample of random data (trajectories, vectors, etc.) according to a specified random mechanism.
2. Update of the parameters of the random mechanism, on the basis of the data, in order to produce a "better" sample in the next iteration.

We now present a proto-typical version of the CE method. Suppose we wish to minimize some cost function $S(\mathbf{z})$ over all $\mathbf{z}$ in some set $\mathcal{Z}$. Let us denote the minimum by γ^* , thus

$$\gamma^* = \min_{\mathbf{z} \in \mathcal{Z}} S(\mathbf{z}). \quad (2)$$

We randomize our deterministic problem by defining a family of auxiliary pdfs $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ on $\mathcal{Z}$ and we associate with Eq. (2) the following estimation problem for a given scalar γ :

$$\mathbb{P}_{\mathbf{u}}(S(\mathbf{Z}) \leq \gamma) = \mathbb{E}_{\mathbf{u}} [I_{\{S(\mathbf{Z}) \leq \gamma\}}],$$

where $\mathbf{u}$ is some known (initial) parameter. We consider the event "cost is low" to be the rare event $I_{\{S(\mathbf{Z}) \leq \gamma\}}$ of interest. To estimate this event, the CE method generates a sequence of tuples $\{(\hat{\gamma}_t, \hat{\mathbf{v}}_t)\}$, that converge (with high probability) to a small neighborhood of the optimal tuple $(\gamma^*, \mathbf{v}^*)$, where γ^* is the solution of the program (2) and $\mathbf{v}^*$ is a pdf that emphasizes values in $\mathcal{Z}$ with a low cost. We note that

typically the optimal $\mathbf{v}^*$ is degenerated as it concentrates on the optimal solution (or a small neighborhood thereof). Let ρ denote the fraction of the best samples used to find the threshold γ . The process that is based on sampled data is termed the *stochastic counterpart* since it is based on stochastic samples of data. The number of samples in each stage of the stochastic counterpart is denoted by N , which is a predefined parameter. The following is a standard CE procedure for minimization borrowed from de-Boer et al. (2005). We initialize by setting $\hat{\mathbf{v}}_0 = \mathbf{v}_0 = \mathbf{u}$ and choose a not very small ρ , say $10^{-2} \leq \rho$. We then proceed iteratively as follows:

1. **Adaptive updating of γ_t .** For a fixed $\mathbf{v}_{t-1}$, let γ_t be a $\rho 100\%$ -percentile of $S(\mathbf{Z})$ under $\mathbf{v}_{t-1}$. That is, γ_t satisfies $\mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{Z}) \leq \gamma_t) \geq \rho$ and $\mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{Z}) \geq \gamma_t) \geq 1 - \rho$ where $\mathbf{Z} \sim f(\cdot; \mathbf{v}_{t-1})$. A simple estimator $\hat{\gamma}_t$ of γ_t can be obtained by taking a random sample $\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(N)}$ from the pdf $f(\cdot; \mathbf{v}_{t-1})$, calculating the performances $S(\mathbf{Z}^{(\ell)})$ for all ℓ , ordering them from smallest to biggest as $S_{(1)} \leq \dots \leq S_{(N)}$ and finally evaluating the $\rho 100\%$ sample percentile as $\hat{\gamma}_t = S_{(\lceil \rho N \rceil)}$.

2. **Adaptive updating of $\mathbf{v}_t$.** For a fixed γ_t and $\mathbf{v}_{t-1}$, derive $\mathbf{v}_t$ from the solution of the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}} I_{\{S(\mathbf{Z}) \leq \gamma_t\}} \log f(\mathbf{Z}; \mathbf{v}). \quad (3)$$

The stochastic counterpart of (3) is as follows: for fixed $\hat{\gamma}_t$ and $\hat{\mathbf{v}}_{t-1}$, derive $\hat{\mathbf{v}}_t$ from the following program:

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{\ell=1}^N I_{\{S(\mathbf{Z}^{(\ell)}) \leq \hat{\gamma}_t\}} \log f(\mathbf{Z}^{(\ell)}; \mathbf{v}). \quad (4)$$

We note that if f belongs to the Natural Exponential Family (e.g., Gaussian, Bernoulli), then Eq. (4) has a closed form solution (see de-Boer et al., 2005). The CE optimization algorithm is summarized in Algorithm 2.1. In this paper we will assume that f belongs to a Bernoulli family. In our case $\mathcal{Z} = \{0, 1\}^n$ and $\mathbf{v}$ is an n dimensional vector of numbers between 0 and 1. The update formula of the k th element in $\mathbf{v}$ (Eq. (4)) in this case simply becomes:

$$\hat{v}_t(k) = \frac{\sum_{\ell=1}^N I_{\{S(\mathbf{Z}^{(\ell)}) \leq \hat{\gamma}_t\}} I_{\{Z_k^{(\ell)}=1\}}}{\sum_{\ell=1}^N I_{\{S(\mathbf{Z}^{(\ell)}) \leq \hat{\gamma}_t\}}}.$$

This formula has the interpretation that it counts how many times a value of 1 (in $I_{\{Z_k^{(\ell)}=1\}}$) led to a significant result (matches with the indicator $I_{\{S(\mathbf{Z}^{(\ell)}) \leq \hat{\gamma}_t\}}$),

how many times a value of 0 led to a significant result, and normalize the value of the parameter accordingly. Instead of the updating the parameter vector $\mathbf{v}$ directly via the solution of Eq. (4) we use the following smoothed version

$$\hat{\mathbf{v}}_t = \beta \tilde{\mathbf{v}}_t + (1 - \beta) \hat{\mathbf{v}}_{t-1}, \quad (5)$$

where $\tilde{\mathbf{v}}_t$ is the parameter vector obtained from the solution of (4), and β is a smoothing parameter, with $0.7 < \beta < 1$. The reason for using the smoothed (5) instead of the original updating rule is to smooth out the values of $\hat{\mathbf{v}}_t$, and to reduce the probability that some component $\hat{v}_{t,i}$ of $\hat{\mathbf{v}}_t$ will be zeros or unities at an early stage, and the algorithm will get stuck in a local maxima. Note that for $0 < \beta < 1$ we always have that $\hat{v}_{t,i} > 0$, while for $\beta = 1$ one might have (even at the first iterations) that either $\hat{v}_{t,i} = 0$ or $\hat{v}_{t,i} = 1$ for some indices i . As a result, the algorithm may converge to a wrong solution.

Algorithm 2.1 The CE Method for Stochastic Optimization

1. Choose some $\hat{\mathbf{v}}_0$. Set $t = 1$ (level counter).
2. Generate a sample $\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(N)}$ from the density $f(\cdot; \mathbf{v}_{t-1})$ and compute the sample $\rho 100\%$ -percentile $\hat{\gamma}_t$ of the sample scores.
3. Use the same sample $\mathbf{Z}^{(1)}, \dots, \mathbf{Z}^{(N)}$ and solve the stochastic program (4). Denote the solution by $\tilde{\mathbf{v}}_t$.
4. Apply (5) to smooth out the vector $\tilde{\mathbf{v}}_t$.
5. If for some $t \geq d$, say $d = 3$, $\hat{\gamma}_t = \hat{\gamma}_{t-1} = \dots = \hat{\gamma}_{t-d}$ then **stop**; otherwise set $t = t + 1$ and reiterate from step 2.

Figure 1. A proto-typical CE algorithm.

It is found empirically (Rubinstein & Kroese, 2004) that the CE method is robust with respect to the choice of its parameters N, ρ and β , as long as ρ is not too small, $\beta < 1$, and N is large enough. Typically those parameters satisfy that $0.01 \leq \rho \leq 0.1$, $0.5 \leq \beta \leq 0.9$, and $N \geq 3n$, where n is the number of parameters. Improvements of Algorithm 2.1 include the Fully Adaptive CE (FACE) variant, where the parameters N and ρ are updated online. See de-Boer et al. (2005) for more details.

3. The CE classifier algorithm

Problem (1) is a difficult non-convex discontinuous optimization problem due to the 'zero norm' of α . In

order to circumvent the discontinuity we introduce a vector of n binary variables $\sigma \in \{0, 1\}^n$ and define $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. We replace α with $\Sigma\hat{\alpha}$, i.e., the vector σ multiplies the n dimensional vector $\hat{\alpha}$ componentwise. Thus (1) is transformed to the problem

$$\begin{aligned} & \text{minimize} && \|\Sigma\hat{\alpha}\|_0 + C1^\top \xi \\ & \text{subject to} && Y(KY\Sigma\hat{\alpha} + b1) \geq 1 - \xi \\ & && \xi \geq 0 \\ & && 0 \leq \Sigma\hat{\alpha} \leq C \\ & && \sigma \in \{0, 1\}^n, \end{aligned} \quad (6)$$

with variables $\hat{\alpha}, \sigma, b$ and $\xi \in \mathbb{R}^n$, where 1 is a vector of ones, K is the kernel matrix satisfying $K_{ij} = K(x_i, x_j)$, and $Y = \text{diag}(y_1, \dots, y_n)$. Since any component of α can be written as $1 \cdot \hat{\alpha}$ if $\alpha > 0$ or $0 \cdot \hat{\alpha}$ if $\alpha = 0$ then problems (1) and (6) are equivalent.

The zero norm counts the number of non-zero elements in a vector. Therefore $\|\Sigma\hat{\alpha}\|_0 = \sum_{i=1}^n \sigma_i$ equals the number of SVs (in practice $\hat{\alpha}$ is never exactly zero). Denote by $\tilde{\alpha}$ the vector of non-zero elements of vector $\Sigma\hat{\alpha}$. Denote $\tilde{K}$ as the kernel matrix K without the columns indicated by the zeros of σ and $\tilde{Y}$ as the diagonal matrix of labels without the labels indicated by the zeros of σ . For a given vector σ , the problem (6) is reduced to:

$$\begin{aligned} & \text{minimize} && 1^\top \xi \\ & \text{subject to} && Y(\tilde{K}\tilde{\alpha} + b1) \geq 1 - \xi \\ & && \xi \geq 0 \\ & && 0 \leq \tilde{\alpha} \leq C, \end{aligned} \quad (7)$$

with variables $\tilde{\alpha}, \xi, b$. Notice that the dimension of $\tilde{\alpha}$ is equal to the number of SVs, i.e., $\sum_{i=1}^n \sigma_i$. This is typically a much smaller number than the number of training patterns n . Problem (7) is a Linear Programming (LP) problem which can be solved with highly efficient optimization algorithms. The number of variables is $(\sum_{i=1}^n \sigma_i) + n + 1$ and the number of constraints is $2n + 2(\sum_{i=1}^n \sigma_i)$. Therefore the algorithm is affected by the number of features only when the kernel matrix is initially calculated. We propose to solve problem (1) by using the CE algorithm for the combinatorial optimization problem of selecting the set of SVs. We use Bernoulli distribution for $f(\cdot; v)$, with the variable σ playing the role of z in the CE method. The objective function is

$$S(\sigma) = 1^\top \sigma + C1^\top \xi,$$

where $1^\top \xi$ was calculated by solving (7). We chose $\rho = 0.03, \beta = 0.7, d = 3, N = 5n$, where n is the number of training patterns. The values of parameters ρ, β, d are standard¹ and the value of N was determined via trial

¹We found that the CE algorithm is not sensitive to

and error to be the lowest number which consistently produces the same results as with a larger N .

Consider the following optimization problem, known as the "one norm SVM":

$$\begin{aligned} & \text{minimize} && 1^\top \alpha + C1^\top \xi \\ & \text{subject to} && Y(KY\alpha + b1) \geq 1 - \xi \\ & && \xi \geq 0 \\ & && 0 \leq \alpha \leq C, \end{aligned} \quad (8)$$

Problem (8) is an upper bound on the "zero norm" problem (1). This approach was introduced by Smola et al., (2000, pp. 141-142) and produces less sparse solutions than those generated by the "zero norm" problem. From a computational point of view, this problem can be solved efficiently. In order to reduce the complexity of our algorithm, we first run a one norm SVM, and then only consider the SVs that were computed by the one norm as potential SVs. It is reasonable to assume that the majority of SVs of the zero norm problem are SVs of the one-norm problem because these SVs are representative in some sense. A similar approach was adopted in Burges and Schölkopf (1997). Thus we suggest to use the solution of problem (8) as an initial solution for problem (6).

Three modifications were made to the standard CE algorithm:

1. The solution $\tilde{\alpha}$ of (7) may have nearly zero elements. To remove redundancy and to make the term of number of SVs in the target function $S(\sigma)$ slightly less rigid, we decided that if the value of $\tilde{\alpha}_j < 10^{-6}$ then the corresponding σ_j was set to zero.
2. In a CE iteration frequently identical σ vectors are generated (especially near convergence). Given the same σ , the solution of the convex problem (7) is identical. Thus computing (7) again is superfluous. To avoid this redundancy we used a cache for the results of (7). We note that caching can be done in an extremely simple way. Each σ (a vector of ones and zeros) is assigned a binary number corresponding to its value. The σ 's are sorted in ascending order. For each σ problem (7) is computed only if its value does not appear in the (sorted) list of already computed values².
3. Before the CE algorithm is executed the non SVs of the solution of Problem (8) are excluded from

these parameters as long as they are chosen from the ranges specified at the end of Section 2.

²After the target function is calculated without redundancy, the σ vectors are sorted according to the value of their target function as explained in (2.1).

consideration as potential SVs. All the training points (including the "removed" SVs) are still taken into account when solving (7). Note that this step guarantees sparser solution than those generated by Smola et al. (2000).

In summary, the training phase of the CE classifier algorithm consists of solving typically several thousand LP problems. We note that the number of variables used by each LP is independent of the number of features that only affects the computation of the kernel matrix, which is performed once.

4. Experiments

The CE classifier algorithm was compared to the SVM and the one-norm SVM (Smola et al., 2000) algorithms on one synthetic dataset and five real world two class problem datasets taken from the UCI repository (see Table 1).

4.1. Synthetic data

The synthetic dataset was generated in the following manner. The probability of $y = 1$ or -1 was equal. Features x_1, x_2 were drawn as $x|y \sim \mathcal{N}(y\mathbf{1}, \Lambda)$, with the covariance matrix $\Lambda = VDV^T$ and $\mathbf{1}$ a vector of ones. The matrix D was diagonal with the values 1, 2 and V a randomly generated unitary matrix. The number of training patterns was 300 and the test set consisted of 1000 patterns.

The linear kernel was used for all algorithms. The value of hyperparameter C for each algorithm was set as the minimizer of the errors on the test set. The SVM algorithm achieved 24.5% with 56.7% of the training patterns. The one-norm SVM algorithm achieved 24.5% with 0.667% of the training patterns. The CE algorithm achieved 24.3% with 0.333% of the training patterns. Figures 2-4 illustrate the sparsity of the one-norm SVM and CE algorithm in comparison to the SVM algorithm. The class of each training pattern is marked by a circle or a square. If the shape is full, this means that the training pattern acts as a support vector. Note that the CE algorithm utilizes only one training pattern out of 300 for the best results.

4.2. Real-world data

Each dataset was divided ten times into a training set of 150 patterns and a test set of the remaining patterns. Three kernels were used:

1. Linear: $K_{i,j} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$.
2. Polynomial of degree 5: $K_{i,j} = (\langle \mathbf{x}_i, \mathbf{x}_j \rangle + 1)^5$.

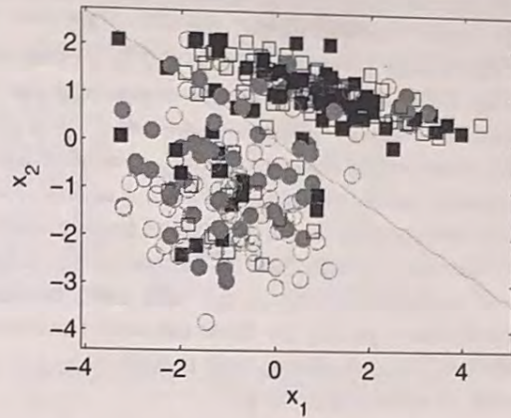


Figure 2. SVM classifier for synthetic problem (56.7% SVs).

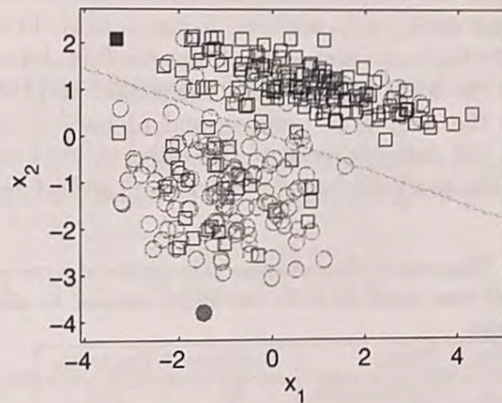


Figure 3. One-norm SVM classifier for synthetic problem (0.667% SVs).

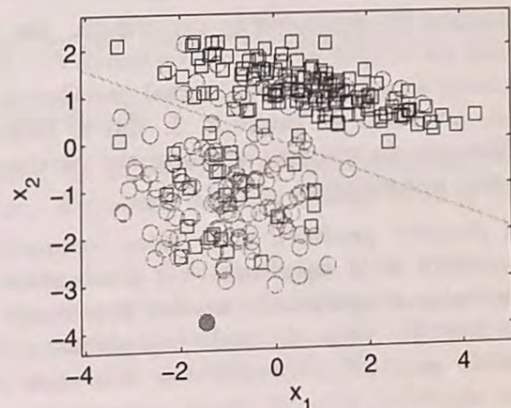


Figure 4. CE classifier for synthetic problem (0.333% SVs).

3. Gaussian, or radial basis functions (RBF): $K_{i,j} = \exp(-(\langle \mathbf{x}_i, \mathbf{x}_j \rangle^2)/(2\sigma_{RBF}^2))$.

For the linear and polynomial kernels the 5-fold cross validation error was calculated for $C = \lambda/(1 - \lambda)$, where λ had values between 0.1 and 0.9 in increments of 0.1. The RBF kernel is strongly dependant on its hyperparameter σ_{RBF} . Therefore we set $C = 1$ and the 5-fold cross validation error was calculated for 9 linearly spaced values of σ_{RBF} between 1 and 5. Nine values were used to match the number of hyperparameter values for the other kernels. This particular range of value of hyperparameter σ_{RBF} was used because we observed that typically for these datasets the kernel matrix shifted from an almost unit matrix ($\sigma_{RBF} = 1$) to a matrix of ones ($\sigma_{RBF} = 5$).

The training features of all the datasets, algorithms and kernels were preprocessed to have zero mean. In the case of the linear kernel only zero mean preprocessing was used. In the case of the polynomial kernel all datasets but the first dataset were preprocessed to have a unit variance (in addition to zero mean). In the case of the Gaussian kernel for all but the first dataset and only the SVM algorithm for the second and third datasets the data were preprocessed to have a unit variance (all datasets were preprocessed to have zero mean). The test examples were processed accordingly.

Table 1. The real-world datasets. The value of hyperparameter C was equal to λ for the Bupa dataset to allow convergence.

No.	Name	Features	Patterns
1	Ionosphere	34	351
2	Pima	8	768
3	Wdbc	30	569
4	Bupa	6	345
5	Sonar	60	208

The algorithms are compared by two criteria: the error rate and the number of Support Vectors (SVs). Results for linear kernel, polynomial kernel, and Gaussian kernel are presented in Tables 2, 3, and 4, respectively. The criteria was averaged over the partitions into training and test sets.

The CE classifier produced error rates comparable to the standard SVM algorithm. Yet it consistently did so by using a significantly smaller percentage of SVs. On average, across the real-world datasets, the CE classifier used 29%, 70%, 40% less SVs than the one-norm algorithm with the linear, polynomial and Gaussian kernels, respectively. It is interesting to note that this behavior was observed for both a "local" kernel (such as the RBF kernel) and a "global" kernel (such as the linear kernel), indicating that this phenomena cannot be attributed to the locality of the kernel.

5. Discussion

In this paper we considered solving SVM type classification problems with zero norm regularization. Since the optimization problem is non-convex and even discontinuous we formulated it as a combinatorial optimization problem and applied the CE method to solve it. The suggested method was shown to be able to cope well with several rather difficult problems. It produced test errors that were statistically similar to SVMs with L_2 regularization. The main benefit of the method is the sparsity of the solution which was significant in all the instances we experienced with.

A natural question at this point would be why did we employ the CE method and not some other heuristic search method such as genetic algorithms or simulated annealing for searching over the set of SVs. There are two broad types of heuristic search methods. The first type is based on having a proximity relation that facilitates efficient local search. Examples of this type include simulated annealing (Aarts & Korst, 1989) and guided local search (Voudouris & Tsang, 1999). The second type includes global methods that are typically based on iteratively creating collections of solutions, improving the quality of the collection from iteration to iteration. The first type of search algorithms is not suitable for the zero-norm minimization problem because the problem has many discontinuities, and algorithms that are based on local search are likely to get stuck. Algorithms that employ some penalty function on traversed areas (such as the guided local search algorithm) are not likely to perform well since there are many discontinuities leading to an overly complicated penalty function. The CE method belongs to the second type of algorithms. As opposed to other algorithms of this type, such as genetic algorithms, the CE method naturally fits the classification problem and is easy to use. The CE method does not require tedious tweaking or definition of unnatural operators as genetic algorithms do. Finally, similarly to simulated annealing the CE method is guaranteed to converge to the optimal solution in the large sample limit. This provides some theoretical guarantees on the performance of the algorithm.

The computational cost of running the CE based classifier is determined by the size of the LP problems and by the available computational resources. Currently, there are excellent and highly optimized LP solvers that are readily available. In the problems we describe above, a typical run of the CE method required solving a few thousand small linear programs, which can be done rather efficiently. It is worth noting that the CE classification algorithm is "embarrassingly paral-

Table 2. The error and SV percentage (in parentheses) for the real-world datasets with a linear kernel. The number after the $\pm$ represents the standard deviation of either the error or the percentage of SVs. The lowest error and sparsest solution for every dataset are in boldface.

No.	SVM	One-norm SVM	CE
1	14.7 $\pm$ 2.0 (36.4 $\pm$ 8.9)	13.0$\pm$1.8 (15.1 $\pm$ 2.5)	14.6 $\pm$ 1.8 (7.7$\pm$2.6)
2	24.3$\pm$1.4 (51.2 $\pm$ 6.2)	24.6 $\pm$ 1.1 (4.9 $\pm$ 0.5)	24.8 $\pm$ 1.5 (3.9$\pm$1.2)
3	5.7$\pm$1.2 (10.1 $\pm$ 2.5)	5.9 $\pm$ 1.4 (4.8 $\pm$ 1.1)	5.9 $\pm$ 0.8 (3.7$\pm$1.4)
4	32.6 $\pm$ 2.1 (71.9 $\pm$ 3.8)	32.5$\pm$1.7 (4.0 $\pm$ 0.0)	33.4 $\pm$ 2.8 (3.1$\pm$0.6)
5	25.9 $\pm$ 3.7 (53.7 $\pm$ 7.9)	25.5$\pm$4.7 (14.7 $\pm$ 2.4)	25.5$\pm$4.7 (10.3 $\pm$ 1.9)

Table 3. The error and SV percentage for the real-world datasets with a polynomial kernel of degree 5.

No.	SVM	One-norm SVM	CE
1	15.3 $\pm$ 2.7 (36.1 $\pm$ 3.7)	13.7 $\pm$ 2.6 (20.5 $\pm$ 8.4)	12.5$\pm$1.3 (7.1 $\pm$ 1.1)
2	33.2 $\pm$ 1.5 (48.8 $\pm$ 5.2)	30.6 $\pm$ 1.8 (29.5 $\pm$ 4.6)	30.2$\pm$2.4 (11.2 $\pm$ 6.6)
3	6.0 $\pm$ 2.1 (21.9 $\pm$ 2.7)	8.5 $\pm$ 2.8 (15.1 $\pm$ 3.2)	5.6$\pm$1.3 (2.5 $\pm$ 0.7)
4	33.7$\pm$5.2 (58.0 $\pm$ 6.0)	36.3 $\pm$ 2.2 (33.9 $\pm$ 3.5)	37.9 $\pm$ 4.4 (14.4 $\pm$ 9.9)
5	15.9$\pm$4.7 (70.3 $\pm$ 1.7)	20.3 $\pm$ 7.0 (51.1 $\pm$ 6.8)	23.3 $\pm$ 5.3 (6.9 $\pm$ 1.6)

Table 4. The error and SV percentage (in parentheses) for the real-world datasets with a Gaussian kernel and $C = 1$.

No.	SVM	One-norm SVM	CE
1	9.8 $\pm$ 2.3 (76.3 $\pm$ 2.2)	6.2$\pm$1.5 (19.3 $\pm$ 3.1)	6.6 $\pm$ 2.3 (14.1 $\pm$ 2.6)
2	27.5 $\pm$ 1.7 (67.9 $\pm$ 5.1)	25.2$\pm$3.0 (12.9 $\pm$ 4.9)	25.4 $\pm$ 3.3 (8.5 $\pm$ 1.7)
3	7.5 $\pm$ 0.8 (42.4 $\pm$ 3.4)	4.6$\pm$1.5 (14.4 $\pm$ 1.5)	4.7 $\pm$ 1.4 (9.7 $\pm$ 1.2)
4	34.4$\pm$3.0 (93.4 $\pm$ 1.6)	36.9 $\pm$ 3.9 (28.3 $\pm$ 25.5)	36.9 $\pm$ 4.6 (10.4 $\pm$ 4.3)
5	46.7 $\pm$ 6.3 (100.0 $\pm$ 0.0)	24.3$\pm$3.5 (41.7 $\pm$ 6.2)	24.5 $\pm$ 3.7 (22.5 $\pm$ 2.5)

lel" since the sampling method and calculation of the score (solving the LP) can be efficiently divided across processors. As a result, a speedup which is linear in the number of available CPUs can be obtained.

We observed in experiments that using the CE method as a second phase after running one norm SVM results in a sparser solution with a similar error. The CE method can therefore be considered as a post-processing phase, where classifiers that are obtained using a "standard" method are being made sparser. We expect that even greater gain would be obtained for regression problems. For such problems, the fraction of SVs that is typically obtained by kernel methods is large. The issue of using the CE method to further sparsify regressors is left for future research.

The classification algorithm we suggested operates on a single data set. In order to speed-up the computation for large data sets, it may be of interest to use active set methods (see Mangasarian & Musicant, 2001 for consideration of active set methods in SVMs). The idea is to partition the classification problem to multiple smaller optimization problems. After each of these problems is solved separately a global solution is obtained by merging the solutions of the smaller

problems. This may be done, perhaps, by interleaving activation of one norm SVMs with CE method type iterations on different parts of the data set. This issue is left for future research.

Acknowledgements

This research was partially supported by NSERC.

References

- Aarts, E., & Korst, J. (1989). *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons.
- Bradley, P. S., & Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. *Proceedings of the 15th International Conference on Machine Learning* (pp. 82-90).
- Burges, C. J. C. (1996). Simplified support vector decision rules. *Proceedings of the 13th International Conference on Machine Learning* (pp. 71-77).
- Burges, C. J. C., & Schölkopf, B. B. (1997). Improving speed and accuracy of support vector learning ma-

- chines. *Advances in Neural Information Processing Systems*, 9 (pp. 375 – 381).
- Chang, C.-C., & Lin, C.-J. (2002). Training nu-
support vector regression: theory and algorithms.
Neural Computation, 14, 1959–1977.
- Cristianini, N., & Shawe-Taylor, J. (2000). *An In-
troduction to Support Vector Machines and Other
Kernel-Based Learning Methods*. Cambridge, Eng-
land: Cambridge University Press.
- de-Boer, P., Kroese, D., Mannor, S., & Rubinstein,
R. (2005). A tutorial on the cross-entropy method.
Annals of Operations Research, 134, 19–67.
- Downs, T., Gates, K., & Masters, A. (2001). Exact
simplification of support vector solutions. *Journal
of Machine Learning Research*, 2, 293–297.
- Floyd, S., & Warmuth, M. (1995). Sample compres-
sion, learnability, and the Vapnik-Chervonenkis di-
mension. *Machine Learning*, 21, 269–304.
- Herbrich, R. (2002). *Learning kernel classifiers: The-
ory and algorithms*. Boston: MIT Press.
- Herbrich, R., & Williamson, R. C. (2002). Algorithmic
luckiness. *Journal of Machine Learning Research*, 3,
175–212.
- Keerthi, S. S., Shevade, S. K., Bhattacharyya, C., &
Murthy, K. R. K. (2001). Improvements to platts
SMO algorithm for SVM classifier design. *Neural
Computation*, 13, 637 –649.
- Mangasarian, O., & Musicant, D. (2001). Active
set support vector machine classification. *Advances
in Neural Information Processing Systems*, 13 (pp.
577–583).
- Platt, J. (1998). Fast training of support vector ma-
chines using sequential minimal optimization. In
B. Schölkopf, C. Burges and A. Smola (Eds.), *Ad-
vances in kernel methods - support vector learning*.
MIT press.
- Rubinstein, R., & Kroese., D. (2004). *The cross-
entropy method: A unified approach to combinator-
ial optimization, Monte-Carlo simulation, and ma-
chine learning*. Springer-Verlag.
- Schölkopf, B., & Smola, A. J. (2002). *Learning with
kernels*. MIT Press.
- Smola, A. J., Bartlett, P., Schölkopf, B., & Schuur-
mans, D. (2000). *Advances in large margin classi-
fiers*. MIT Press.
- Vapnik, V. N. (1998). *Statistical Learning Theory*.
New York: Wiley Interscience.
- Voudouris, C., & Tsang, E. (1999). Guided local
search. *European Journal of Operational Research*,
113, 469–499.
- Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R.
(2003). 1-norm support vector machines. *Advances
in Neural Information Processing Systems*, 16.

APPENDIX FRIEDRICH01



Keyword



Search...

Search

Advanced search Start Over

Share

Bookmark

Report an Issue

MARC View

LEADER 02119nas a2200493 a 4500

001 50324863

003 SIRSI

005 20260214090101.0

006 m o d

007 cr n

008 980511c19899999nyusx pss 0 0eng c

010 a| 2004233017

019 a| DO NOT EDIT this 360 MARC RECORD

022 a| 1872-8286 y| 0925-2312

035 a| (WaSeSS)ssj0017129

040 a| OH1 b| eng c| OH1 d| OCL d| OCLCQ d| GUA d| OCLCQ d| VRC d| U9S d| WAU d| OCLCQ d| OCLCF d| OCLCO d| VT2 d| OCLCQ d| OCLCO d| OCLCQ d| Z5A d| OCLCO d| NJR d| OCLCO d| OCLCA d| BUF d| U3W d| EZC d| OCLCO d| AU@ d| OCLCO d| WYU d| OCLCO d| OCLCA d| TKN d| OCLCO d| OCLCA d| NJT d| OCLCO d| OCL d| OCLCQ d| UAB d| OCLCQ d| WaSeSS

042 a| pcc

050 1 4 a| QA76.87

130 0 a| Neurocomputing (Online)

222 0 a| Neurocomputing b| (Online)

245 1 0 a| Neurocomputing h| [electronic resource].

260 a| [New York] : b| Elsevier Science B.V.

310 a| Twenty-eight no. a year, b| <2004->

321 a| Nine no. a year, b| <1995->

362 1 a| Print began in 1989.

500 a| Some issues also have distinctive theme titles.

515 a| Some nos. issued in combined form.

530 a| Also issued in print.

538 a| Mode of access: World Wide Web.

588 a| Description based on: Vol. 7, issue 1 (Jan. 1995); title from contents screen (ScienceDirect, viewed Jan. 16, 2004).

650 0 a| Neural computers v| Periodicals.

650 0 a| Artificial intelligence v| Periodicals.

650 0 a| Neural computers x| Periodicals.

650 0 a| Artificial intelligence x| Periodicals.

650 0 a| Neural networks (Computer science)

650 2 a| Neural Networks (Computer) v| periodicals.

650 2 a| Neural Networks, Computer

650 2 a| Neural Networks (Computer) v| Periodicals.

655 0 a| Electronic journals.

776 1 t| Neurocomputing x| 0925-2312 w| (DLC)sn 93002487 w| (OCoLC)24351769

856 4 0 u| http://SK8ES4MC2L.search.serialssolutions.com/?V=1.0&L=SK8ES4MC2L&S=JCS&C=NEURAMS&T=marc

910 a| CONSER online

023 0 a| 0925-2312 2| j

949 a| Electronic resource w| ASIS c| 1 i| 50324863-1001 I| ONLINE m| ONLINE r| Y s| Y t| ONLINE u| 2/28/2026 o| .STAFF. 360 MARC



Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#) (814) 865-6368
[Libraries Intranet \(Staff Only\)](#)
[Accessibility Help](#)
[Website Feedback](#)
[Policies and Guidelines](#)
[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#)
[Twitter](#)
[Instagram](#)



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)[Share](#)[Bookmark](#)[Report an Issue](#)

Neurocomputing [electronic resource].

Uniform Title

Neurocomputing (Online)

Published

[New York] : Elsevier Science B.V.

Access Online

[serialsolutions.com](#)

Dates of Publication and/or Sequential Designation

Print began in 1989.

Subject(s)

[Neural computers—Periodicals](#)
[Artificial Intelligence—Periodicals](#)
[Neural networks \(Computer science\)](#)
[Neural Networks \(Computer\)—periodicals](#)
[Neural Networks, Computer](#)
[Neural Networks \(Computer\)—Periodicals](#)

Genre(s)

[Electronic journals](#)

ISSN

1872-8286

Grouping ISSN

0925-2312

Publication Frequency

Twenty-eight no. a year, <2004->
Nine no. a year, <1995->

Note

Some issues also have distinctive theme titles.

Special Numbering

Some nos. issued in combined form.

Other Forms

Also issued in print.

Technical Details

Mode of access: World Wide Web.

ASK
A LIBRARIAN[View MARC record](#) | catkey: 50324863

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#) (814) 865-6368
[Libraries Intranet \(Staff Only\)](#)
[Accessibility Help](#)
[Website Feedback](#)
[Policies and Guidelines](#)
[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#)
[Twitter](#)
[Instagram](#)



PennState
University Libraries

RENEW MATERIALS

Give Today

My Account



Services Research About Ask

🕒 Pattee and Paterno: View Hours

Online Journals, Books and more

🔍 All **📖 Journals Only** 📖 Books Only 📺 Videos Only

Title begins with



Search

[Show more search options](#) ▼

Showing results 1 through 1 of 1

for the search: "NEURAMS"

Refine Results: [All](#) | [Journals Only](#) | [Books Only](#) | [Videos Only](#)

Limit by: [Peer Reviewed](#) | [Open Access](#)



Neurocomputing (Amsterdam)

ISSN: 0925-2312

🎓 Peer Reviewed

1989 to 1994 in [Backfile Package - Computer Science \(Legacy\) \[YCS\]](#)

1989 to Present in [ScienceDirect Journals](#)

Search inside this journal

Search

[Back to Top](#)



PennState
University Libraries



SERVICES

[Borrow & Renew](#)

[Course Reserves](#)

[Rooms & Spaces](#)

[Technology & Equipment](#)

[Adaptive Technology & Services](#)

[Interlibrary Loan](#)

[Teaching Support for Instructors](#)

RESEARCH

[Try These First](#)

[Citation Guides](#)

[ScholarSphere](#)

[Collections](#)

[Electronic Theses and Dissertations](#)

[Research Data Services](#)

ABOUT

[Hours](#)

[Directory](#)

[Libraries and Departments](#)

[Jobs](#)

[Message from the Dean](#)

[Diversity](#)

[News and Events](#)

[Your Donation at Work](#)

[Libraries Calendar](#)

ASK A LIBRARIAN

[By Chat](#)

[By Phone](#)

[In Person](#)

[By Email](#)

[Schedule a Research Appt.](#)

[FAQs \(or How Do I?...\)](#)

[Tutorials](#)

[The Search Bar](#)

[Thank a Library Worker](#)

[Libraries Home](#) | [Libraries Intranet \(Employees Only\)](#) | [Accessibility Help](#) | [Website Feedback](#) | [Policies and Guidelines](#) | [Acknowledgement of Land](#) | (814) 865-6368

[Hotline](#) | [Privacy Statements](#) | [Non Discrimination](#) | [Accessibility](#) | [Legal Statements](#)

Copyright ©2025 The Pennsylvania State University.

All rights reserved. Except where otherwise noted, this work is subject to a [Creative Commons Attribution 4.0 license](#). Details and exceptions.

Last Updated on 03-03-2026 © Ex Libris, a ProQuest Company, 2000-2026

APPENDIX FRIEDRICH02

Evolutionary tuning of multiple SVM parameters

Frauke Friedrichs, Christian Igel*

Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany

Available online 7 January 2005

Abstract

The problem of model selection for support vector machines (SVMs) is considered. We propose an evolutionary approach to determine multiple SVM hyperparameters: The covariance matrix adaptation evolution strategy (CMA-ES) is used to determine the kernel from a parameterized kernel space and to control the regularization. Our method is applicable to optimize non-differentiable kernel functions and arbitrary model selection criteria. We demonstrate on benchmark datasets that the CMA-ES improves the results achieved by grid search already when applied to few hyperparameters. Further, we show that the CMA-ES is able to handle much more kernel parameters compared to grid-search and that tuning of the scaling and the rotation of Gaussian kernels can lead to better results in comparison to standard Gaussian kernels with a single bandwidth parameter. In particular, more flexibility of the kernel can reduce the number of support vectors.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Support vector machines; Model selection; Evolutionary algorithms

1. Introduction

Support vector machines (SVMs, e.g., [5,22,24]) are learning machines based on two key elements: a general purpose linear learning algorithm and a problem specific kernel that computes the inner product of input data points in a feature space. The choice of the kernel function is the crucial step in handling a learning task with an SVM. For example, it is important to achieve a distribution of the data in the feature

*Corresponding author. Tel.: +49 2343225558; fax: +49 2343214209.

E-mail address: christian.igel@neuroinformatik.rub.de (C. Igel).

space that reflects the affiliation to the class labels. Often a parameterized family of kernel functions is considered and the problem reduces to finding an appropriate parameter vector for the given problem. In case of noisy, non-separable data one also has to choose a regularization parameter, which controls the trade-off between minimizing the training error and the complexity of the decision function. The kernel parameters together with the regularization parameter are called the hyperparameters of the SVM.

In practice the hyperparameters are usually determined by grid search. That is, the hyperparameters are varied with a fixed step-size through a wide range of values and the performance of every combination is assessed using some performance measure. Because of the computational complexity, grid search is only suitable for the adjustment of very few parameters. Perhaps, the most elaborate systematic technique for choosing multiple hyperparameters are gradient descent methods [3,4,8,15]. These algorithms iterate the following procedure: The SVM is trained using the current hyperparameter vector, the gradient of some generalization error bound w.r.t. the hyperparameters is calculated, and a step is performed in the parameter space based on this gradient. However, this approach has some significant drawbacks. The kernel function has to be differentiable. The score function for assessing the performance of the hyperparameters (or at least an accurate approximation of this function) also has to be differentiable with respect to kernel and regularization parameters. This excludes reasonable measures such as the number of support vectors. In [3, Section 6.2] separability of the dataset is assumed when computing the derivative, which is a very restrictive assumption. Iterative gradient-based algorithms, which usually rely on smoothed approximations of a score function, do not ensure that the search direction points exactly to an optimum of the original, often discontinuous generalization performance measure.

We propose an evolutionary method for hyperparameter selection that does not suffer from the limitations described above. Evolutionary algorithms have been successfully applied to model selection for neural networks [11,18,25]. This includes the recent applications of genetic algorithms for feature selection of SVMs [6,7,14,17]. We use the covariance matrix adaptation evolution strategy (CMA-ES, [10]) to search for an appropriate hyperparameter vector. The fitness function that is optimized directly corresponds to some generalization performance measure. We apply our method to tuning Gaussian kernels, where not only the scaling but also the orientation is adapted.

We give a short description of SVMs in Section 2 and of the CMA-ES in Section 3. The parameterization of general Gaussian kernels is introduced in Section 4. We present experimental results in Section 5 and draw our conclusions in Section 6.

2. Support vector machines

We consider L_1 -norm soft margin SVMs for the discrimination of two classes. Let $(\mathbf{x}_i, y_i)$, $1 \leq i \leq \ell$, be the training examples, where $y_i \in \{-1, 1\}$ is the label associated with input pattern $\mathbf{x}_i \in X$. The main idea of SVMs is to map the input vectors to a

feature space F and to classify the transformed data by a linear function. The transformation $\phi : X \rightarrow F$ is implicitly done by a kernel $K : X \times X \rightarrow \mathbb{R}$, which computes an inner product in the feature space, i.e., $K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$. The linear function for classification in the feature space is chosen according to a generalization error bound considering a target margin and the margin slack vector, i.e., the amounts by which individual training patterns fail to meet that margin (cf. [5,22,24]). This leads to the SVM decision function $f(\mathbf{x}) = \text{sign}(\sum_{i=1}^{\ell} y_i \alpha_i^* K(\mathbf{x}_i, \mathbf{x}) + b)$, where the coefficients α_i^* are the solution of the following quadratic optimization problem: Maximize $W(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j)$ subject to $\sum_{i=1}^{\ell} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$ for $i = 1, \dots, \ell$. The optimal value for b can then be computed based on the solution $\boldsymbol{\alpha}^*$. The vectors $\mathbf{x}_i$ with $\alpha_i > 0$ are called support vectors. The regularization parameter C controls the trade-off between maximizing the target margin and minimizing the L_1 -norm of the margin slack vector of the training data.

3. Evolution strategies

Evolution strategies (ES, [2,20,23]) are one of the main branches of evolutionary algorithms, i.e., a class of iterative, direct, randomized optimization methods mimicking principles of neo-Darwinian evolution theory. Here, we use the highly efficient CMA-ES [9,10]. A set of μ individuals that form the parent population is maintained. Each individual has a genotype that encodes a candidate solution for the optimization problem at hand, in this study an m -dimensional real-valued object variable vector representing the SVM hyperparameters. The fitness of an individual is equal to the objective function value at the point in the search space it represents. In each iteration of the algorithm, $\lambda > \mu$ new individuals, the offspring, are generated by partially stochastic variations of parent individuals. The fitness of the offspring is computed and the μ best of the offspring form the next parent population. This loop of variation and selection is repeated until a termination criterion is met.

In the following, we describe the covariance matrix adaptation ES (CMA-ES) proposed in [9,10], which performs efficient real-valued optimization. Each individual represents an m -dimensional real-valued object variable vector. These variables are altered by two variation operators, intermediate recombination and additive Gaussian mutation. The former corresponds to computing the center of mass of the μ individuals in the parent population. Mutation is realized by adding a normally distributed random vector with zero mean. In the CMA-ES, the complete covariance matrix of the Gaussian mutation distribution is adapted during evolution to improve the search strategy. More formally, the object parameters $\mathbf{g}_k^{(t)}$ of offspring $k = 1, \dots, \lambda$ created in generation t are given by

$$\mathbf{g}_k^{(t)} = \langle \tilde{\mathbf{g}} \rangle^{(t)} + \sigma^{(t)} \mathbf{B}^{(t)} \mathbf{D}^{(t)} \mathbf{z}_k^{(t)},$$

where $\langle \tilde{\mathbf{g}} \rangle^{(t)} = (1/\mu) \sum_{i=1}^{\mu} \tilde{\mathbf{g}}_i^{(t)}$ is the center of mass of the parent population in generation t and the $\mathbf{z}_k^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are independent realizations of an m -dimensional normally distributed random vector with zero mean and covariance matrix equal to

the identity matrix $\mathbf{I}$. The covariance matrix $\mathbf{C}^{(t)}$ of the random vectors

$$\sigma^{(t)} \mathbf{B}^{(t)} \mathbf{D}^{(t)} \mathbf{z}_k^{(t)} \sim \mathcal{N}(\mathbf{0}, \mathbf{C}^{(t)})$$

is a symmetric positive $m \times m$ matrix with

$$\mathbf{C}'^{(t)} = \mathbf{C}^{(t)} / \sigma^{(t)2} = \mathbf{B}^{(t)} \mathbf{D}^{(t)} (\mathbf{B}^{(t)} \mathbf{D}^{(t)})^T.$$

The columns of the orthogonal $m \times m$ matrix $\mathbf{B}^{(t)}$ are the normalized eigenvectors of $\mathbf{C}'^{(t)}$ and $\mathbf{D}^{(t)}$ is a $m \times m$ diagonal matrix with the square roots of the corresponding eigenvalues.

The strategy parameters, both the matrix $\mathbf{C}'^{(t)}$ and the so-called global step-size $\sigma^{(t)}$, are updated online using the covariance matrix adaptation (CMA) method. The key idea of the CMA is to alter the mutation distribution in a deterministic way such that the probability to reproduce steps in the search space that have led to the current population is increased. This enables the algorithm to detect correlations between object variables and to become invariant under orthogonal transformations of the search space (apart from the initialization). In order to use the information from previous generations efficiently, the search path of the population over a number of past generations is taken into account.

In the CMA-ES, rank-based (μ, λ) -selection is used for environmental selection. That is, the μ best of the λ offspring form the next parent population. After selection, the strategy parameters are updated:

$$\mathbf{s}^{(t+1)} = (1 - c) \mathbf{s}^{(t)} + c_u \underbrace{\sqrt{\mu} \mathbf{B}^{(t)} \mathbf{D}^{(t)} \langle \mathbf{z} \rangle_\mu^{(t)}}_{\frac{\sqrt{\mu}}{\sigma^{(t)}} (\langle \hat{\mathbf{g}} \rangle^{(t+1)} - \langle \hat{\mathbf{g}} \rangle^{(t)})},$$

$$\mathbf{C}'^{(t+1)} = (1 - c_{\text{cov}}) \mathbf{C}'^{(t)} + c_{\text{cov}} \mathbf{s}^{(t+1)} (\mathbf{s}^{(t+1)})^T.$$

Herein, $\mathbf{s}^{(t+1)} \in \mathbb{R}^m$ is the evolution path—a weighted sum of the centers of the population over the generations starting from $\mathbf{s}^{(1)} = \mathbf{0}$. The factor $\sqrt{\mu}$ compensates for the loss of variance due to computing the center of mass. The parameter $c \in]0, 1]$ controls the time horizon of the adaptation of $\mathbf{s}$; we set $c = 1/\sqrt{m}$. The constant $c_u = \sqrt{c(2-c)}$ normalizes the variance of $\mathbf{s}$ (viewed as a random variable) as $1^2 = (1-c)^2 + c_u^2$. The expression $\langle \mathbf{z} \rangle_\mu^{(t)} = (1/\mu) \sum_{i=1}^{\mu} \mathbf{z}_{i,\lambda}^{(t)}$ is the average of the realizations of the random vector that led to the new parent population, where $i : \lambda$ denotes the index of the offspring having the i th best fitness value of all offspring in the current generation ($\hat{\mathbf{g}}_i^{(t+1)} = \mathbf{g}_{i,\lambda}^{(t)}$, $i = 1, \dots, \mu$). The parameter $c_{\text{cov}} \in [0, 1[$ controls the update of $\mathbf{C}'^{(t)}$ and we set it to $c_{\text{cov}} = 2/(m^2 + m)$. This update rule shifts $\mathbf{C}'^{(t)}$ towards the $m \times m$ matrix $\mathbf{s}^{(t+1)} (\mathbf{s}^{(t+1)})^T$ making mutation steps in the direction of $\mathbf{s}^{(t+1)}$ more likely. The evolution path does not only represent the last (adaptive) step of the parent population, but a time average over all previous adaptive steps. The influence of previous steps decays exponentially, where the decay rate is controlled by c .

The adaptation of the global step-size parameter σ is done separately on a shorter timescale (a single parameter can be estimated based on less samples compared to the complete covariance matrix). We keep track of a second evolution path $\mathbf{s}_\sigma$ without the scaling by $\mathbf{D}$:

$$\mathbf{s}_\sigma^{(t+1)} = (1 - c_\sigma)\mathbf{s}_\sigma^{(t)} + c_{u_\sigma}\sqrt{\mu}\mathbf{B}^{(t)}\langle\mathbf{z}\rangle_\mu^{(t+1)},$$

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{\|\mathbf{s}_\sigma^{(t+1)}\| - \hat{\lambda}_n}{d\hat{\lambda}_n}\right),$$

where $\hat{\lambda}_n$ is the expected length of a m -dimensional, normally distributed random vector with covariance matrix $\mathbf{I}$. The damping parameter $d \geq 1$ decouples the adaptation rate from the strength of the variation. We set $d = \sqrt{m}$ and start from $\mathbf{s}_\sigma^{(1)} = \mathbf{0}$. The parameter $c_\sigma \in]0, 1]$ controls the update of $\mathbf{s}_\sigma$. Here, we use $c_\sigma = c$. Setting $c_{u_\sigma} = \sqrt{c_\sigma(2 - c_\sigma)}$ normalizes the variance of $\mathbf{s}_\sigma$. The evolution path $\mathbf{s}_\sigma$ is the sum of normally distributed random variables. Because of the normalization, its expected length would be $\hat{\lambda}_n$ if there were no selection. Hence, the update rule basically increases the global step size if the steps leading to selected individuals have been larger than expected and decreases the step size in the opposite case.

The CMA-ES needs only small population sizes. In the following, they are chosen according to the heuristics $\lambda = \max(5, \min(m, 4 + \lfloor 3 \ln m \rfloor))$ and $\mu = \max(1, \lfloor \lambda/4 \rfloor)$ based on [10], where m is the number of object variables. The initializations of $\mathbf{C}^{(1)}$ and $\sigma^{(1)}$ allow for incorporation of prior knowledge about the scaling of the search space (see Section 5.1). Note that all other parameters of the algorithm can be set to the carefully determined default values given in [9,10]. Of course, tuning for example the cumulation by changing c and c_σ for a particular class of cost functions can increase the search performance, but without additional knowledge about the objective function we stick to the robust default values, see [10] for a detailed discussion.

4. Encoding general Gaussian kernels

We consider SVMs with general Gaussian kernels

$$K_{\mathbf{A}}(\mathbf{x}, \mathbf{z}) = e^{-(\mathbf{x}-\mathbf{z})^T \mathbf{A}(\mathbf{x}-\mathbf{z})},$$

where $\mathbf{A}$ is a symmetric positive definite matrix. We use three different parameterizations of these kernels yielding three nested spaces of kernel functions. Firstly, there are the ordinary Gaussian kernels $K_{\mathbf{I}}$, where $\mathbf{I}$ is the unit matrix and $\tau > 0$ the only adjustable parameter. Secondly, we allow independent scalings of the components of the input vectors and consider the kernel family $K_{\mathbf{D}}$, where $\mathbf{D}$ is a diagonal matrix with positive entries. Finally, we allow arbitrary symmetric positive definite matrices $\mathbf{A}$, i.e., the input space can be scaled and rotated. This kernel family is denoted simply by $K_{\mathbf{A}}$.

The individuals in the ES encode the regularization parameter C and the kernel parameters. The ES obeys restrictions of parameters to $\mathbb{R}^+$ when necessary (see below). However, when encoding K_A kernels, we have to ensure that after variation the genotype still corresponds to a feasible (i.e., symmetric, positive definite) matrix. We use the fact that for any symmetric and positive definite $n \times n$ matrix A there exists an orthogonal $n \times n$ matrix T and a diagonal $n \times n$ matrix D with positive entries such that $A = T^T D T$ and

$$T = \prod_{i=1}^{n-1} \prod_{j=i+1}^n R(\alpha_{ij}),$$

see [21]. The $n \times n$ matrices $R(\alpha_{ij})$ are elementary rotation matrices which are equal to the unit matrix except for $[R(\alpha_{ij})]_{ii} = [R(\alpha_{ij})]_{jj} = \cos \alpha_{ij}$ and $[R(\alpha_{ij})]_{ji} = -[R(\alpha_{ij})]_{ij} = \sin \alpha_{ij}$. Each genotype encodes the $n + (n^2 - n)/2 + 1$ parameters

$$(C', d_1, \dots, d_n, \alpha_{1,2}, \alpha_{1,3}, \dots, \alpha_{1,n}, \alpha_{2,3}, \alpha_{2,4}, \dots, \alpha_{2,n}, \dots, \alpha_{n-1,n}),$$

and we set $D = \text{diag}(|d_1|, \dots, |d_n|)$ and $C = |C'|$.

5. Experimental evaluation

5.1. Experiments

For the evaluation of our hyperparameter optimization method we used the common medical benchmark datasets *Breast-Cancer*, *Diabetes*, *Heart*, and *Thyroid* with input dimension n equal to 9, 8, 13, and 5, and total number of patterns 277, 768, 270, and 215, respectively. The data were preprocessed and partitioned as in [19]. Each component of the input data is normalized to zero mean and unit standard deviation. There are 100 partitions of each dataset into disjoint training and test sets. In [16], appropriate SVM hyperparameters for the Gaussian kernel $K_{\tau I}$ were determined using a two-stage grid-search and the following score function: For each hyperparameter combination, five SVMs are built using the training sets of the first five data partitions and the average of the classification rates on the corresponding five test sets determines the score value (Test-5) of this parameter vector. The hyperparameter vector with the best score is selected and its performance is measured by calculating the score function using all 100 partitions (Test-100).¹

We used this scenario as a basis for our evolutionary hyperparameter adaptation approach: We took the results of the grid search performed in [16] as initial values for the CMA-ES and used the score function described above to determine the fitness. The initializations of τ and C were approximately 0.00661 and 280, 0.00090 and 274, 0.00120 and 8.8, and 0.05863 and 401 for *Breast-Cancer*, *Diabetes*, *Heart*,

¹We have chosen this evaluation procedure in order to compare our results one-to-one with [16]. As in [16,19], the data for computing Test-5 and Test-100 overlap and hence Test-100 is only a weak indicator for the generalization performance. However, we tested other evaluation criteria which gave qualitatively the same results.

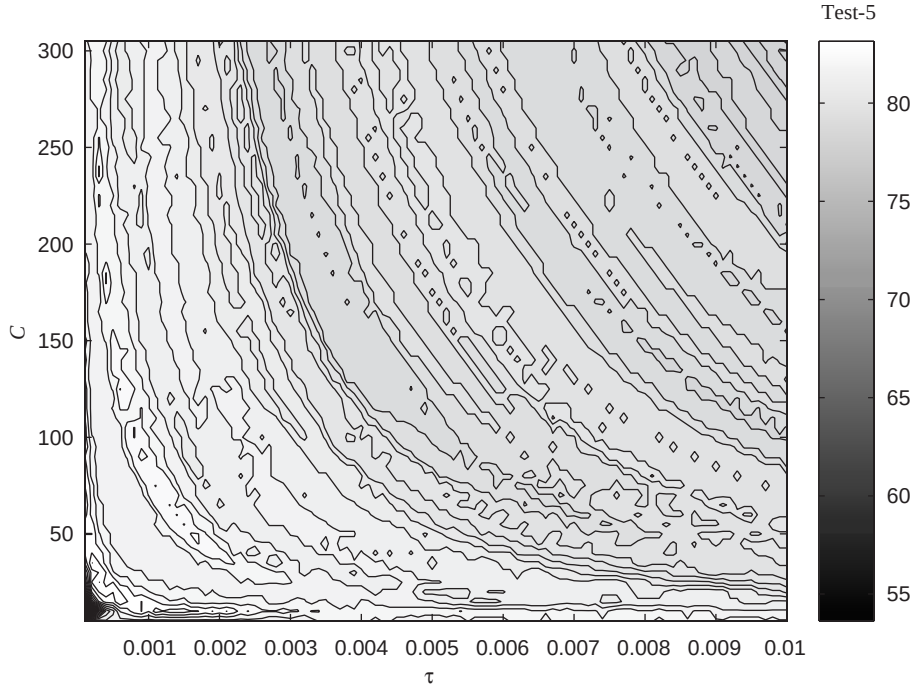


Fig. 1. Score function (average classification rates on the five test sets, Test-5, in percent) for the *Heart* dataset using $K_{\tau l}$ with different combinations of C and τ .

and *Thyroid*, respectively. Grid-search reveals that the hyperparameter optimization problem is indeed multi-modal, see Fig. 1 for an example. It is not necessary to determine the starting points for the ES by grid search. Typically, we start the CMA-ES from a randomly chosen point or in an area which was roughly determined by a very coarse grid search.

For each benchmark dataset and each of the three kernel families, 20 evolutionary optimizations over 250 generations were performed. In each trial, the hyperparameter vector with the best fitness value (Test-5) was regarded as the solution. The performance of each solution was assessed by calculating the score using all 100 data partitions (Test-100). The SVMs were trained using $\text{SVM}^{\text{light}}$ [13]. We set the parameters of the initial CMA mutation distribution to $\sigma^{(1)} = 1$ and $\mathbf{C}^{(1)} = \text{diag}(0.001, 0.00001, \dots, 0.00001)$, because the regularization parameter C should be varied stronger compared to the kernel parameters in the beginning.

5.2. Results

We compared the start values of Test-5 and Test-100 with the results obtained by the ES, see Table 1. Except for a few cases, we achieved significantly (t -test, $p < 0.05$) better results by evolutionary optimization. The ES could even improve the

Table 1
Results averaged over 20 trials $\pm$ standard deviations

Data	Kernel	$\hat{t}$	λ	Test-5	Test-100	#SV-100
C	init			73.77	74.51	113.52
	$K_{\tau I}$	10 ± 15	5	$74.00 \pm 0.08^*$	$74.56 \pm 0.04^*$	114.58 ± 1
	K_D	83 ± 42	10	$76.91 \pm 0.96^*$	$75.17 \pm 0.65^*$	$113.10 \pm 0.89^*$
	K_A	179 ± 59	15	$77.47 \pm 0.53^*$	$75.38 \pm 0.42^*$	$112.70 \pm 0.68^*$
D	init			76.34	76.67	247.83
	$K_{\tau I}$	102 ± 78	5	$76.84 \pm 0.11^*$	76.67 ± 0.04	251.63 ± 8.14
	K_D	149 ± 56	9	$78.10 \pm 0.22^*$	$76.88 \pm 0.25^*$	$239.31 \pm 2.84^*$
	K_A	181 ± 56	14	$78.07 \pm 0.19^*$	76.73 ± 0.32	$235.73 \pm 3.43^*$
H	init			83.80	84.79	106.33
	$K_{\tau I}$	10 ± 17	5	83.81 ± 0.04	84.74 ± 0.04	$103.37 \pm 0.97^*$
	K_D	97 ± 41	11	$85.71 \pm 0.43^*$	$84.98 \pm 0.36^*$	$76.52 \pm 2.10^*$
	K_A	153 ± 47	17	$85.86 \pm 0.17^*$	$85.14 \pm 0.33^*$	$75.51 \pm 1.5^*$
T	init			96.27	95.83	16.36
	$K_{\tau I}$	3 ± 5	5	$96.56 \pm 0.08^*$	95.74 ± 0.2	15.99 ± 1.01
	K_D	16 ± 35	6	$97.29 \pm 0.1^*$	$96.01 \pm 0.04^*$	$15.46 \pm 0.12^*$
	K_A	5 ± 3	12	$97.33 \pm 0^*$	$96.01 \pm 0.05^*$	$15.42 \pm 0.18^*$

The first column specifies the benchmark: *Breast-Cancer* (C), *Diabetes* (D), *Heart* (H), and *Thyroid* (T). The second column indicates whether the results refer to the initial grid search values (init, [16]) or to the evolutionary optimized kernels $K_{\tau I}$, K_D , or K_A . The $\hat{t}$ values are the generations needed to evolve the final solution on average and λ is the number of offspring per generation (μ is chosen according to the heuristic given in Section 3). Thus, $\hat{t} \cdot \lambda$ indicates the average computational costs to find the final parameters, of course, good values are already found earlier in the evolutionary process. The percentages of correctly classified patterns on the 5 and 100 test sets are given by Test-5 and Test-100. The average numbers of support vectors over the 100 training sets is #SV-100. Results statistically significantly better compared to grid-search (init) are marked with.

The average numbers of support vectors over the 100 training sets is #SV-100.

* (significantly better compared to grid-search (init), two-sided t -test, $p < 0.05$).

performance when adapting $K_{\tau I}$ in some cases, although the initial values were already tuned for $K_{\tau I}$ by intensive grid search.

We also used a t -test to compare the performance of the resulting SVMs when the evolutionary search was performed in the three spaces of kernel functions. The scaled kernels K_D as well as the scaled and rotated kernels K_A led to significantly ($p < 0.05$) better results compared to the ordinary Gaussian kernel $K_{\tau I}$ (except for *Diabetes*: Test-100 with kernel K_A). The major improvement was already achieved by allowing scaling. However, in one case (*Breast-Cancer*: Test-5) the kernel K_A with the additional rotation parameters gave significantly ($p < 0.05$) better results than just scaling.

There is another remarkable advantage of the scaled kernel K_D and the scaled and rotated kernel K_A : The number of support vectors decreases. Evolutionary

adaptation of kernels K_D and K_A led to SVMs having significantly less support vectors compared to the ordinary kernel $K_{\tau I}$ optimized by the ES or grid-search, see Table 1. Again, in one case (*Diabetes*) the kernel K_A with additional rotation parameters yielded a significantly ($p < 0.05$) better result than just scaling.

6. Conclusions

The CMA evolution strategy is a powerful, general method for SVM hyperparameter selection. It can handle a large number of kernel parameters and does neither require differentiable kernels and model selection criteria nor separability of the data. As typical selection criteria exhibit multiple local optima (e.g., see Fig. 1 and [8]), we claim that evolutionary optimization is generally better suited for SVM model selection than gradient-based methods.

It is often argued that evolutionary optimization of kernels is too time consuming for real-world applications. Given a certain amount of time for designing the classifier, there is a maximum number of objective function evaluations—and the question is how to spend these evaluations. Grid-search, for example, scales exponentially in the number of objective parameters. In contrast, a simple evolution strategy scales linearly on functions that are monotone with respect to the distance to the optimum [1,12]. The objective functions in case of SVM optimization seem to be well suited for evolution strategies (see Fig. 1), which can follow a global trend that is superposed by local minima. Thus, in problems which are currently addressed by grid-search the number of objective parameters and the accuracy of the optimization can be increased when using an evolutionary algorithm.

We demonstrated that extended Gaussian kernels with scaling (and rotating) parameters can lead to a significantly better performance than standard Gaussian kernels. To our knowledge, for the first time a method that adapts the orientation of Gaussian kernels—i.e., that can detect correlations in the input data space *relevant for the kernel machine*—was presented. Our experiments show that by increasing the flexibility of the kernels the number of support vectors can be reduced. Future work will address other, especially non-differentiable kernels and optimization criteria.

Acknowledgements

We thank the authors of [16] for providing their grid-search results and the reviewers for their careful study of the manuscript.

References

- [1] H.-G. Beyer, The Theory of Evolution Strategies, Springer, Berlin, 2001.
- [2] H.-G. Beyer, H.-P. Schwefel, Evolution strategies: a comprehensive introduction, Nat. Comput. 1 (1) (2002) 3–52.

- [3] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, *Mach. Learn.* 46 (1) (2002) 131–159.
- [4] K.-M. Chung, W.-C. Kao, C.-L. Sun, C.-J. Lin, Radius margin bounds for support vector machines with RBF kernel, *Neural Comput.* 15 (11) (2003) 2643–2681.
- [5] N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, 2000.
- [6] D.R. Eads, D. Hill, S. Davis, S.J. Perkins, J. Ma, R.B. Porter, J.P. Theiler, Genetic algorithms and support vector machines for time series classification, in: B. Bosacchi, D.B. Fogel, J.C. Bezdek (Eds.), *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation V*, Proceedings of the SPIE, vol. 4787, 2002, pp. 74–85.
- [7] H. Fröhlich, O. Chapelle, B. Schölkopf, Feature selection for support vector machines by means of genetic algorithms, in: *Proceedings of the 15th IEEE International Conference on Tools with AI (ICTAI 2003)*, IEEE Computer Society, 2003, pp. 142–148.
- [8] C. Gold, P. Sollich, Model selection for support vector machine classification, *Neurocomputing* 55 (1–2) (2003) 221–249.
- [9] N. Hansen, A. Ostermeier, Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: the $(\mu/\mu, \lambda)$ -CMA-ES, in: *Proceedings of the Fifth European Congress on Intelligent Techniques and Soft Computing (EUFIT'97)*, Aachen, Germany, Verlag Mainz, Wissenschaftsverlag, 1997, pp. 650–654.
- [10] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, *Evolution. Comput.* 9 (2) (2001) 159–195.
- [11] C. Igel, S. Wiegand, F. Friedrichs, Evolutionary optimization of neural systems: the use of self-adaptation, in: M.G. de Bruin, D.H. Mache, J. Szabados (Eds.), *Trends and Applications in Constructive Approximation*, International Series of Numerical Mathematics, vol. 151, Birkhäuser Verlag, Basel, 2005, in press.
- [12] J. Jägersküpper, Analysis of a simple evolutionary algorithm for minimization in euclidian spaces, in: *Proceedings of the 30th International Colloquium on Automata, Languages, and Programming (ICALP 2003)*, Lecture Notes in Computer Science, vol. 2719, Springer, Berlin, 2003, pp. 1068–1079.
- [13] T. Joachims, Making large-scale SVM learning practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), *Advances in Kernel Methods—Support Vector Learning*, MIT Press, Cambridge, 1999, pp. 169–184 (Chapter 11).
- [14] K. Jong, E. Marchiori, A. van der Vaart, Analysis of proteomic pattern data for cancer detection, in: G.R. Raidl, S. Cagnoni, J. Branke, D.W. Corne, R. Drechsler, Y. Jin, C.G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G.D. Smith, G. Squillero (Eds.), *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, vol. 3005, Springer, Berlin, 2004, pp. 41–51.
- [15] S.S. Keerthi, Efficient tuning of SVM hyperparameters using radius/margin bound and iterative algorithms, *IEEE Trans. Neural Networks* 13 (5) (2002) 1225–1229.
- [16] P. Meinicke, T. Twellmann, H. Ritter, Discriminative densities from maximum contrast estimation, in: S. Becker, S. Thrun, K. Obermayer (Eds.), *Advances in Neural Information Processing Systems*, vol. 15, MIT Press, Cambridge, 2002, pp. 985–992.
- [17] M.T. Miller, A.K. Jerebko, J.D. Malley, R.M. Summers, Feature selection for computer-aided polyp detection using genetic algorithms, in: A.V. Clough, A.A. Amini (Eds.), *Medical Imaging 2003: Physiology and Function: Methods, Systems, and Applications*, Proceedings of the SPIE, vol. 5031, 2003, pp. 102–110.
- [18] S. Nolfi, Evolution and learning in neural networks, in: M.A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, second ed., MIT Press, Cambridge, 2002, pp. 415–418.
- [19] G. Rätsch, T. Onoda, K.-R. Müller, Soft margins for adaboost, *Mach. Learn.* 42 (3) (2001) 287–332.
- [20] I. Rechenberg, *Evolutionstrategie'94*, Werkstatt Bionik und Evolutionstechnik, Frommann-Holzboog, Stuttgart, 1994.
- [21] G. Rudolph, On correlated mutations in evolution strategies, in: R. Männer, B. Manderick (Eds.), *Parallel Problem Solving from Nature 2 (PPSN II)*, Elsevier, Amsterdam, 1992, pp. 105–114.
- [22] B. Schölkopf, A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, 2002.

- [23] H.-P. Schwefel, *Evolution and Optimum Seeking*, Sixth-Generation Computer Technology Series, Wiley, New York, 1995.
- [24] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, Berlin, 1995.
- [25] X. Yao, Evolving artificial neural networks, *Proc. IEEE* 87 (9) (1999) 1423–1447.



Frauke Friedrichs received her Diploma in Biology in 2001 and her Diploma in Mathematics in 2003 from the Ruhr-University Bochum, Germany. In 2003 she joined the Department of Theoretical Biology at the Institut für Neuroinformatik in Bochum. Since 2004 she is a Ph.D. student at the Department of Genetic Epidemiology which belongs to the Institute for Arteriosclerosis Research in Münster, Germany.



Christian Igel received the Diploma degree in Computer Science from the University of Dortmund, Germany, in 1997. Since then he has been with the Institut für Neuroinformatik (INI) at the Ruhr-University Bochum, Germany. He received the doctoral degree from the Technical Faculty of the University of Bielefeld, Germany, in 2002. In 2003, he was appointed Juniorprofessor for the Optimization of Adaptive Systems at the INI. His research focuses on biological and technical aspects of neural and evolutionary information processing.

APPENDIX HORVATH01



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)

Share

[Bookmark](#)[Report an Issue](#)

MARC View

LEADER 01904nas a2200433 a 4500

001 49493627

003 SIRSI

005 20251114090051.0

006 m o d

007 cr n

008 980126c19869999maumx pss 0 0eng c

010 a| 2008233030

016 7 a| 1475529-4 2| DE-600

019 a| DO NOT EDIT this 360 MARC RECORD

022 a| 1573-0565 l| 0885-6125 y| 0885-6125

030 a| MALEEZ

035 a| (WaSeSS)ssj0002686

037 b| Kluwer Academic Publishers, Spuiboulevard 50, P.O. Box 17, 3300 AA Dordrecht, The Netherlands

040 a| F#A b| eng c| F#A d| OCL d| OCLCQ d| MUQ d| OCLCQ d| OCLCS d| GUA d| AZU d| OCLCQ d| OCLCO d| OCLCQ d| OCLCF d| OCLCQ d| TOF d| AU@ d| VT2 d| NJT d| AUD d| DLC d| OCLCO d| OCLCQ d| UAB d| DXU d| OCLCL d| OCLCE d| WaSeSS

042 a| pcc

050 0 0 a| Q325.5

082 0 4 a| 006.31 1| <https://id.oclc.org/worldcat/ddc/E44XrjQ697wjXyvKQFWbgh9gD>

130 0 a| Machine learning (Online)

245 1 0 a| Machine learning h| [electronic resource].

260 a| Boston : b| Kluwer Academic Publishers

310 a| Twelve issues yearly

362 1 a| Began with vol. 1, no. 1 (Mar. 1986).

500 a| Published: Springer, <2007->

515 a| Some issues combined.

538 a| Mode of access: World Wide Web.

588 a| Description based on first issue; title from journal information screen (SpringerLink, viewed Jan. 7, 2008).

588 a| Latest issue consulted: Vol. 108, issue 10 (Oct. 2019); (SpringerLink, viewed Oct. 1, 2019).

650 0 a| Machine learning v| Periodicals.

655 0 a| Electronic journals.

776 0 8 i| Print version: t| Machine learning x| 0885-6125 w| (DLC) 86655869 w| (OCoLC)12681958

856 4 0 u| <http://SK8ES4MC2L.search.serialssolutions.com/?V=1.0&L=SK8ES4MC2L&S=JCs&C=MACHLEA&T=marc>

910 a| CONSER online

850 a| DLC

949 a| Electronic resource w| ASIS c| 1 i| 49493627-1001 l| ONLINE m| ONLINE r| Y s| Y t| ONLINE u| 11/29/2025 o| .STAFF. 360 MARC

ASK
A LIBRARIAN 

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)**PENN STATE
UNIVERSITY LIBRARIES**[Libraries Home](#)

(814) 865-6368

[Libraries Intranet \(Staff Only\)](#)[Accessibility Help](#)[Website Feedback](#)[Policies and Guidelines](#)[Staff Directory](#)**CONNECT WITH
PENN STATE UNIVERSITY
LIBRARIES**[Facebook](#)[Twitter](#)[Instagram](#)



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)[< Previous](#) | 5 of 31,255 | [Next >](#)[Share](#)[Bookmark](#)[Report an Issue](#)

Machine learning [electronic resource].

Uniform Title

Machine learning (Online)

Published

Boston : Kluwer Academic Publishers

Access Online

[serialssolutions.com](#)

Dates of Publication and/or Sequential Designation

Began with vol. 1, no. 1 (Mar. 1986).

Subject(s)

[Machine learning—Periodicals](#)

Genre(s)

[Electronic journals](#)

ISSN

1573-0565

Publication Frequency

Twelve issues yearly

Note

Published: Springer, <2007->

Special Numbering

Some issues combined.

Technical Details

Mode of access: World Wide Web.

ASK
A LIBRARIAN[View MARC record](#) | catkey: 49493627

PennState

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#) (814) 865-6368
[Libraries Intranet \(Staff Only\)](#)
[Accessibility Help](#)
[Website Feedback](#)
[Policies and Guidelines](#)
[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#)
[Twitter](#)
[Instagram](#)



PennState
University Libraries

RENEW MATERIALS

Give Today

My Account



Services Research About Ask

⌚ Pattee and Paterno: View Hours

Online Journals, Books and more

All Journals Only Books Only Videos Only

Title begins with

Search

[Show more search options](#)

Showing results 1 through 1 of 1

for the search: "MACHLEA"

Refine Results: [All](#) | [Journals Only](#) | [Books Only](#) | [Videos Only](#)

Limit by: [Peer Reviewed](#) | [Open Access](#)



Machine learning

ISSN: 0885-6125

Peer Reviewed

[Springer Nature - Springer Journals Archive Collection - Springer](#)

01/01/1986 to 02/28/1996 in [Computer Science](#)

01/01/1997 to 12/31/2017 in [Advanced Technologies & Aerospace Database](#)

01/01/1997 to Present in [Springer Nature BTAA Reading TA](#)

[SpringerLink Contemporary \(1997 - Present\)](#)

1997 to Present in [SpringerLINK Contemporary \(BTAA\)](#)

01/01/2023 to 1 year ago in [Advanced Technologies & Aerospace Database](#)

Search inside this journal

Search

[Back to Top](#)



PennState
University Libraries



SERVICES

Borrow & Renew

Course Reserves

Rooms & Spaces

Technology & Equipment

Adaptive Technology & Services

Interlibrary Loan

Teaching Support for Instructors

RESEARCH

Try These First

Citation Guides

ScholarSphere

Collections

Electronic Theses and Dissertations

Research Data Services

ABOUT

Hours

Directory

Libraries and Departments

Jobs

Message from the Dean

Diversity

News and Events

Your Donation at Work

Libraries Calendar

ASK A LIBRARIAN

By Chat

By Phone

In Person

By Email

Schedule a Research Appt.

FAQs (or How Do I?...)

Tutorials

The Search Bar

Thank a Library Worker

[Libraries Home](#) | [Libraries Intranet \(Employees Only\)](#) | [Accessibility Help](#) | [Website Feedback](#) | [Policies and Guidelines](#) | [Acknowledgement of Land](#) | (814) 865-6368

[Hotline](#) | [Privacy Statements](#) | [Non Discrimination](#) | [Accessibility](#) | [Legal Statements](#)

Copyright ©2025 The Pennsylvania State University.

All rights reserved. Except where otherwise noted, this work is subject to a [Creative Commons Attribution 4.0 license](#). Details and exceptions.

Last Updated on 12-09-2025 © Ex Libris, a ProQuest Company, 2000-2025

APPENDIX HORVATH02



Relational Instance-Based Learning with Lists and Terms

TAMÁS HORVÁTH

tamas.horvath@gmd.de

German National Research Center for Information Technology, AiS.KD, Schloß Birlinghoven, D-53754 Sankt Augustin, Germany

STEFAN WROBEL

wrobel@iws.cs.uni-magdeburg.de

Otto-von-Guericke-Universität Magdeburg, School of Computer Science, IWS, P.O.Box 4120,
D-39106 Magdeburg, Germany

UTA BOHNEBECK

bohnebec@informatik.uni-bremen.de

University of Bremen, Center for Computing Technologies, P.O.Box 330 440, D-28834 Bremen, Germany

Editor: Lorenza Saitta

Abstract. The similarity measures used in first-order IBL so far have been limited to the function-free case. In this paper we show that a lot of power can be gained by allowing lists and other terms in the input representation and designing similarity measures that work directly on these structures. We present an improved similarity measure for the first-order instance-based learner RIBL that employs the concept of *edit distances* to efficiently compute distances between lists and terms, discuss its computational and formal properties, and empirically demonstrate its additional power on a problem from the domain of biochemistry. The paper also includes a thorough reconstruction of RIBL's overall algorithm.

Keywords: inductive logic programming, relational instance-based learning

1. Introduction

For propositional learning problems instance-based learning (IBL) algorithms have always been a very popular and well studied choice (Aha, 1997; Aha, Kibler & Albert, 1991; Wettschereck & Dietterich, 1995). An instance-based algorithm does not compute a generalization of the examples at learning time, but simply stores the examples or variants thereof and then arrives at a classification based on comparing a new instance to the previously stored instances. Since generalization is delayed until classification time, such algorithms are also referred to as *lazy learning* algorithms (Aha, 1997; Aha, Kibler & Albert, 1991). Even the simplest instance-based learning algorithm, the *k*-nearest neighbor (kNN) classifier, is known to outperform other approaches for many application problems. More advanced variants address the problem of sensitivity to irrelevant features by learning feature weights (Wettschereck, Mohri & Aha, 1997) and the problem of overly large instance sets by storing selected examples or generalized prototypes (Aha, Kibler & Albert, 1991; Salzberg, 1991).

In first-order relational learning, as it is studied e.g. in the field of Inductive Logic Programming (ILP) (Muggleton & De Raedt, 1994; Wrobel, 1996) the use of instance-based learning techniques was first described in Emde and Wettschereck (1996), and implemented in the first version of the Relational Instance-Based Learning system RIBL. Even though this original version of RIBL was limited to function-free relational representations, it was applied very successfully to practical problems such as the classification of Diterpene molecules based on their nuclear magnetic resonance (NMR) spectra (Džeroski et al., 1996).

In this paper we present the most recent version of RIBL which has been extended to now handle relational representations with lists and other functional terms as well. To this end we define an improved similarity measure for RIBL that employs the concept of *edit distances* to efficiently compute distances between lists and terms. With experiments in a real-world domain (signal structure prediction in mRNA-molecules) we show that the new similarity measure allows significant additional power to be gained. The paper also includes a thorough reconstruction of the basic original RIBL algorithm and can thus serve as a precise reference to RIBL which has previously only been described in the conference paper (Emde & Wettschereck, 1996).

The paper is organized as follows: In the next section we briefly review the basics of IBL and propositional similarity measures. In Section 3, we then give a detailed reconstruction of the RIBL algorithm that supplements the original paper of (Emde & Wettschereck, 1996). Section 4 then presents the main technical contribution of this paper, the new similarity measure for constants, lists and terms. We discuss its computational properties and prove that the measure is a semi-metric. Section 5 is devoted to the empirical evaluation on the mRNA signal structure prediction problem first presented in Bohnebeck, Horváth and Wrobel (1998a). In Section 6 we discuss related work. Section 7 contains our conclusions and pointers to future work.

2. Instance-based learning and RIBL

In function learning from examples, a learner is given a set of examples E from an instance space $\mathcal{I}$ along with a class label or target value $c(e) \in Y$ for each $e \in E$. If Y contains only a few discrete values, one often speaks of a classification learning problem; if Y has only two values, the task is referred to as concept learning, and one of the target function values (e.g., $c(e) = 1$) is taken to mean “ e is a member of the target concept”, and the other value (e.g., $c(e) = 0$) is taken to mean “ e is not a member of the target concept”. If Y is a continuous set such as $\mathbb{R}$, the problem is also named a regression learning problem. From these examples, most inductive learners would induce a hypothesis h that is a generalization of the examples, e.g., a decision tree, and use it to classify new, unseen examples, i.e., the learner’s prediction c' for a new instance $e_{new} \in \mathcal{I}$ would be

$$c'(e_{new}) := h(e_{new}).$$

An *instance-based* learner, on the other hand, in its simplest form just stores all of the examples E . It then bases its prediction on those k examples from E that are “closest” to e_{new} (where k is a user-given or system determined parameter). To this end, the instance-based

learner possesses (or is given) a *distance measure* DIST

$$\text{DIST} : \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{R}$$

that assigns a real number to each pair of instances such that “closer” pairs of instances are assigned smaller numbers. In the following, we will assume that DIST is bounded between 0 and 1, so we can equivalently speak of the *similarity measure* SIM defined as

$$\text{SIM}(x, y) := 1 - \text{DIST}(x, y).$$

The learner uses SIM (or DIST) to determine the k nearest neighbors $N(e_{\text{new}}) = \{e_{n_1}, \dots, e_{n_k}\} \subseteq E$ of e_{new} (breaking ties arbitrarily), and lets these nearest neighbors “vote” to arrive at a prediction for e_{new} . Usually, each neighbor’s vote is weighted according to its distance, so that closer neighbors have larger influence.

For a discrete prediction problem with class values $Y = \{c_1, \dots, c_m\}$, we select the class which individually has received the highest weighted vote. More precisely, let $N_{c_j}(e_{\text{new}}) := \{e \in N(e_{\text{new}}) \mid c(e) = c_j\}$ denote the neighbors of e_{new} voting for class c_j . Then the voting procedure is:

$$h(e_{\text{new}}) := \operatorname{argmax}_{c_j \in C} \sum_{e \in N_{c_j}(e_{\text{new}})} \text{SIM}(e_{\text{new}}, e).$$

For a continuous prediction problem, instead of selecting the value that received the highest vote, we predict the weighted average of the neighbors’ class values:

$$h(e_{\text{new}}) := \frac{\sum_{e \in N(e_{\text{new}})} \text{SIM}(e_{\text{new}}, e) \cdot c(e)}{\sum_{e \in N(e_{\text{new}})} \text{SIM}(e_{\text{new}}, e)}.$$

For obvious reasons, this simplest instance-based learner is also called a *k-nearest neighbor classifier* (kNN). The value k is usually determined by the learner using cross validation on the training set, choosing the value of k that performs best.

Since an instance-based learner delays generalization until classification time, such algorithms are also referred to as *lazy learning* algorithms. Based on the kNN algorithm, various more sophisticated approaches have been studied to address some of the open issues with kNN. First, for large example sets, storing all the examples might require too much storage and too much time for finding the nearest neighbors. This problem can be addressed by storing only selected examples (Aha, Kibler & Albert, 1991) or using generalized prototypes or rectangles in instance space (Salzberg, 1991). Second, the problem of sensitivity to irrelevant features has been addressed by different weight learning schemes (Wettschereck, Mohri & Aha, 1997).

2.1. Propositional similarity measures

Since the similarity measure is the central element of an instance-based learner, let us have a closer look at its required properties. These are easier to express for distance measures.

Ideally, a distance measure should be a *metric*:

$$\text{DIST}(x, y) \geq 0 \quad (1)$$

$$\text{DIST}(x, y) = 0 \text{ iff } x = y \quad (2)$$

$$\text{DIST}(x, y) = \text{DIST}(y, x) \quad (3)$$

$$\text{DIST}(x, y) \leq \text{DIST}(x, z) + \text{DIST}(z, y) \quad (4)$$

for every $x, y, z \in \mathcal{I}$. A metric DIST is said to be *trivial* if for every $x, y \in \mathcal{I}$

$$\text{DIST}(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise.} \end{cases}$$

DIST is a *semi-metric* if Condition (4) (the *triangle inequality*) does not hold. Relaxing Condition (2) to $\text{DIST}(x, x) = 0$ for $\forall x \in \mathcal{I}$, we get the definition of *pseudo-metric*. A metric, semi-metric or pseudo-metric DIST is *r-bounded* ($r \in \mathbb{R}$) if r is an upper bound on the values of DIST . In this paper, all distance (and thus similarity) measures considered are 1-bounded.

Depending on how instances in $\mathcal{I}$ are represented, defining a distance measure (or similarity measure, respectively) that satisfies the above conditions is more or less difficult. For propositional representations including both numeric and discrete attributes, one usually defines distance measures on individual attributes depending on their type.

Let $\text{TYPE}(A)$ denote the type of an attribute A . For a numerical attribute A with value range $[a, b]$ ($\text{TYPE}(A) := \text{number}(a, b)$), we define the similarity of two attribute values r_1 and r_2 as

$$\text{SIM}_{att}(r_1, r_2, A) = 1 - \frac{|r_1 - r_2|}{b - a}.$$

For a discrete attribute A ($\text{TYPE}(A) := \text{discrete}$), we define the similarity of two attribute values d_1 and d_2 by

$$\text{SIM}_{att}(d_1, d_2, A) = \begin{cases} 1 & \text{if } d_1 = d_2 \\ 0 & \text{otherwise.} \end{cases}$$

From these, the overall similarity of two instances is computed as the average of the similarities of all attributes. More precisely, let $A_1, \dots, A_n$ denote the attributes describing the instances, and let $i[j]$ denote the value of attribute A_j for a particular instance i . Then define the similarity of two instances $i_1, i_2 \in \mathcal{I}$ as

$$\text{SIM}(i_1, i_2) := \frac{\sum_{j=1, \dots, n} \text{SIM}(i_1[j], i_2[j], A_j)}{n}.$$

We note that if desired, attribute weights may be assigned or learned (Wettschereck, Mohri & Aha, 1997); in this case, the weighted average is used in the above formula.

3. First-order instance-based learning in RIBL

RIBL, as first introduced in Emde and Wettschereck (1996), is a first-order instance-based learner that applies the basic principles of IBL as described above to an instance space consisting of first-order descriptions. More precisely, RIBL uses a k -nearest-neighbor algorithm that determines the value of k through cross-validation on the training set. It differs from the basic algorithm firstly through its first-order learning task and similarity measure, and secondly through its use of a special weight learning scheme for predicates and arguments. Due to space reasons, we do not describe this weight learning approach here, and refer the reader to Emde and Wettschereck (1996) for details. Instead, in this section, we give a detailed definition of the learning task addressed by RIBL and then provide a thorough reconstruction of the first-order similarity measure that supplements the short description in Emde and Wettschereck (1996).

3.1. Learning task

RIBL turns the basic instance-based learning problem into a first-order version by allowing each instance to be described not only by numerical and discrete attributes, but also by attributes of type object ($\text{TYPE}(A) = \text{object}$) containing object identifiers that point to further information in a separately specified body of *background knowledge* $\mathcal{B}$, a database consisting of first-order ground atoms (first-order facts). Let us first illustrate this with an example.

Assume we want to send a culinary present to a friend, and want to classify a number of prepackaged present sets from different vendors according to whether they are *suitable* for our friend or not. Based on a few manually classified examples, we want further offers to be classified automatically by RIBL. Assume that about each set, we know its price and delivery mode (personal, mail, or pick-up). Each instance will be represented by three attributes: its identifier, price, and delivery mode. In addition, we know the contents of each package which for simplicity we assume consist of different red wines and cheeses. We also know something about different vineyards. Thus, our first package might be represented as follows:

$e : (\text{set1}, 125, \text{personal})$ with $c(e) = \text{suitable}$,

whereas a new package to be classified might be represented by

$e_{\text{new}} : (\text{set25}, 195, \text{mail})$ without classification, i.e., $c(e_{\text{new}}) = ?$.

In these instances the first argument (set1 and set25, respectively) is of type *object* and refers to further information in the background knowledge $\mathcal{B}$, such as:

```
% set ID, cheese type, weight, and origin of cheese
cheese(set1, camembert, 150, france)
cheese(set25, roquefort, 200, france)
cheese(set25, ricotta, 100, italy)
```

```
% set ID, vineyard ID, vintage year, and bottle size
wine(set1, mouton, 1988, 0.75)
wine(set1, gallo, 1995, 0.5)
wine(set25, mouton, 1995, 0.75)
```

```
% vineyard ID, popularity, size, country
vineyard(gallo, famous, large, usa)
vineyard(mouton, famous, small, france)
```

Note that $\mathcal{B}$ contains both general background knowledge and knowledge specific to individual instances. This particular example uses the function-free representation of Emde and Wettschereck (1996); below, we also allow arguments of type `constant` (with custom defined distances), `list` (which may contain lists), and arguments of type `term` (which may contain arbitrary first-order terms). Thus, the exact task considered in this paper is defined as follows.

Given:

- a set of examples E , each described by a set of attributes $\{A_1, \dots, A_n\}$ of type `number`, `discrete`, `object`, `constant`, `list`, or `term`,
- a target value¹ $c(e)$ for every $e \in E$,
- background knowledge $\mathcal{B}$ consisting of first-order atoms with arguments of type `number`, `discrete`, `object`, `list`, or `term`, and
- a set of unseen instances E_{new} each described by the same attributes as E .

Find: A prediction $c'(e)$ of the target value for each $e \in E_{new}$.

3.2. Similarity Computation in RIBL

To compute the similarity of two first-order instances, naturally we want to take into account not only the attributes of the instances, but also the available background knowledge. To this end, RIBL uses a recursive descent strategy similar to the one employed in Bisson (1992b). We first illustrate the idea using our wine and cheese running example (see also figure 1). So assume we want to compare e and e_{new} . For the arguments expressing price and delivery mode, we simply rely on the standard distances for numerical and discrete attributes. To compare the object identifiers *set1* and *set25*, however, we first collect all we know about these two objects from the background knowledge. In our example we obtain the two sets $\mathcal{L}_{set1}$ and $\mathcal{L}_{set25}$ of logical atoms

$$\begin{aligned} \mathcal{L}_{set1} = \{ & \text{cheese}(\text{set1}, \text{camembert}, 150, \text{france}), \\ & \text{wine}(\text{set1}, \text{mouton}, 1988, 0.75), \\ & \text{wine}(\text{set1}, \text{gallo}, 1995, 0.5) \}, \end{aligned}$$

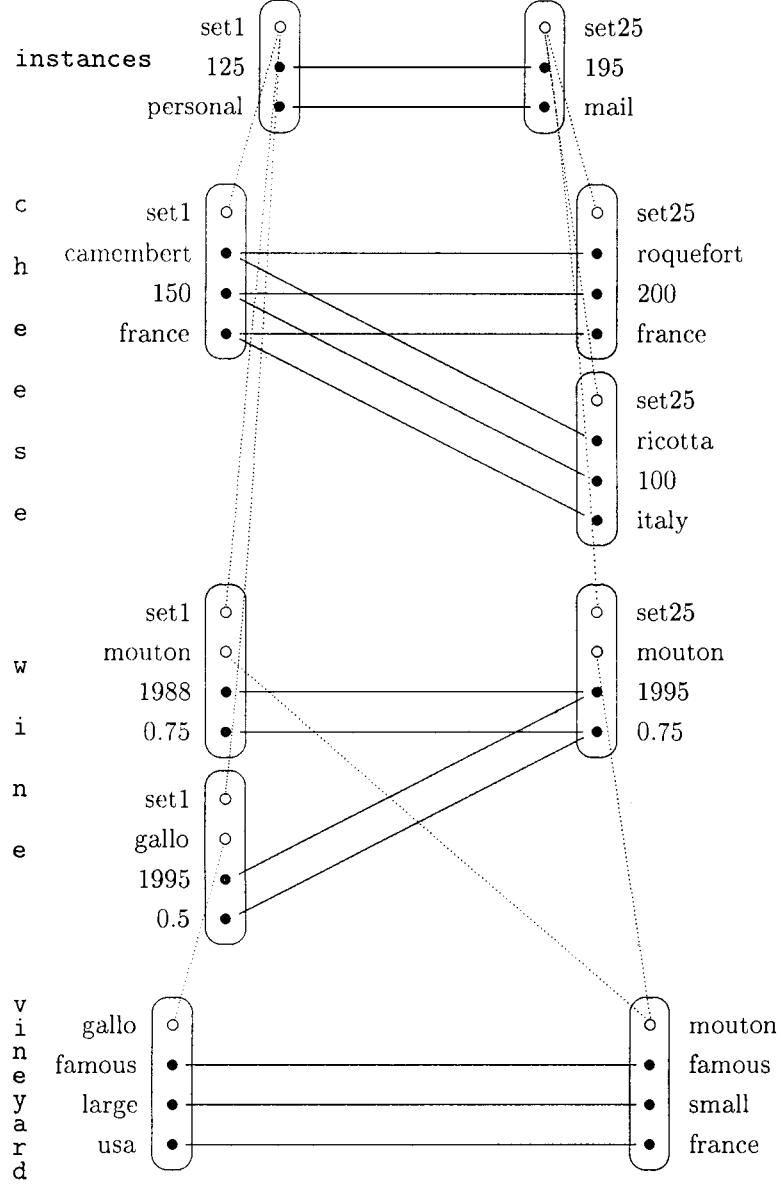


Figure 1. The recursive scheme for computing the similarity between e and e_{new} .

$$\mathcal{L}_{set25} = \{\text{cheese}(\text{set25}, \text{roquefort}, 200, \text{france}) \\ \text{cheese}(\text{set25}, \text{ricotta}, 100, \text{italy}) \\ \text{wine}(\text{set25}, \text{mouton}, 1995, 0.75)\},$$

respectively.

We now have to compute the similarity between these two sets of atoms. We do this separately for each predicate that is used, and then average across all of them. So taking the atoms using “wine” as an example, we get the subsets

$$\begin{aligned}\mathcal{L}'_{set1} &= \{\text{wine}(\text{set1}, \text{mouton}, 1988, 0.75), \text{wine}(\text{set1}, \text{gallo}, 1995, 0.5)\}, \\ \mathcal{L}'_{set25} &= \{\text{wine}(\text{set25}, \text{mouton}, 1995, 0.75)\},\end{aligned}$$

respectively. For each atom in the smaller set, which in this case is the set $\mathcal{L}'_{set25}$ for e_{new} , we determine which of the atoms in the other set is most similar. Thus in our example we compute the similarity between

$$\text{wine}(\text{set25}, \text{mouton}, 1995, 0.75) \text{ and } \text{wine}(\text{set1}, \text{mouton}, 1988, 0.75),$$

and between

$$\text{wine}(\text{set25}, \text{mouton}, 1995, 0.75) \text{ and } \text{wine}(\text{set1}, \text{gallo}, 1995, 0.5).$$

To do this, we apply the same principle recursively: We treat the two atoms as instances to be compared (however, we ignore the first argument to avoid infinite recursion). Thus in our example, we use the basic similarity measures on the third and fourth argument and collect further background knowledge about the objects *mouton* and *gallo*, i.e., for the second pair of atoms we need to compare the sets of literals

$$\begin{aligned}\{\text{vineyard}(\text{gallo}, \text{famous}, \text{large}, \text{usa})\} \text{ and} \\ \{\text{vineyard}(\text{mouton}, \text{famous}, \text{small}, \text{france})\}.\end{aligned}$$

Once we know the most similar partner for each literal of the smaller set, we sum up all these similarities and normalize with the size of the *larger* set. This is done to ensure that with an increasing number of extra literals in one of the sets, similarity becomes smaller (if one of the sets is empty, similarity is 0 for this pair of atom sets). In our example the most similar atom for $\text{wine}(\text{set25}, \text{mouton}, 1995, 0.75)$ is $\text{wine}(\text{set1}, \text{mouton}, 1988, 0.75)$, the similarity of which is divided by two since the larger set contained two atoms. A similar computation is made for the cheese literals, arriving at the overall similarity for *set1* and *set25* by adding the cheese and wine literal set similarities, and dividing by two.

In our example, we have used all of the available background knowledge; for larger problems, a depth bound is used to ensure limited effort for the recursive computation.

We precisely define the similarity computation as follows. Let $\mathcal{B}(e, 1)$ denote the set of background facts containing at least one object from the arguments of an instance e , and let $\mathcal{B}(e, d+1) \subseteq \mathcal{B} \setminus \bigcup_{i=1}^d \mathcal{B}(e, i)$ denote the set of facts containing at least one object occurring in a fact from $\mathcal{B}(e, d)$. Note that the above language includes propositional representations as special cases. For instance, in our example, $\mathcal{B}(e_{new}, 1)$ is the set $\mathcal{L}_{set25}$.

The similarity function used by RIBL to compare two instances e_1 and e_2 with attributes $A_1, \dots, A_n$ with respect to background knowledge $\mathcal{B}$ is defined as

$$\text{SIM}(\mathcal{B}, e_1, e_2) := \frac{\sum_{j=1, \dots, n} \text{SIMILARITY}(\mathcal{B}, e_1, e_2, 0, e_1, e_2, j)}{n}. \quad (5)$$

The algorithmic definition of the term SIMILARITY in (5) is given in Algorithm 1. In the algorithm we generally use $e[i]$ to denote the i -th argument value of an atom e , or the i -th element of a list e . Furthermore, by a q -atom we mean an atom with predicate symbol q . The algorithm computes, in the context of comparing e_1 and e_2 , the normalized similarity between the i th arguments of the atoms l_1 and l_2 , where l_1 and l_2 are both of depth d with respect to e_1 and e_2 . (For the sake of consistency, we define the depth of an instance to be 0, and also refer to instances as atoms.) We use the similarity measure of Section 2 for arguments of type `number`, and `discrete`, respectively (see Lines 3, and 5 in Algorithm 1). In order to compare objects o_1 and o_2 (i.e., arguments of type `object`), the algorithm first checks whether d has already reached the value of the depth parameter. If yes, it outputs 1, or 0 depending on the truth of $o_1 = o_2$ (Line 8). Otherwise, we first collect from $\mathcal{B}$ the set $\mathcal{L}_i$ of literals with depth $d + 1$ containing o_i (Line 12) for $i = 1, 2$. The similarity between o_1 and o_2 is defined by the similarity between the sets $\mathcal{L}_1$ and $\mathcal{L}_2$. The comparison of two sets of facts is based on a similarity measure between two facts. In RIBL the similarity between two facts is reduced again to the similarity between arguments thus making the algorithm recursive. The formal definition of similarity between facts is given by SIM_{lit} in Line 18.

Now we can turn back to the problem of similarity between the sets $\mathcal{L}_1$ and $\mathcal{L}_2$ induced by o_1 and o_2 , respectively. Let $\mathcal{P}_i$ denote the set of pairs (q', k) for which there is a q' -atom in $\mathcal{L}_i$ such that its k th argument is o_i for $i = 1, 2$. For each argument position (q', k) from $\mathcal{P}_1 \cup \mathcal{P}_2$ we collect the q' -atoms from $\mathcal{L}_1$ (resp. $\mathcal{L}_2$) with k th argument o_1 (resp. o_2). From these two sets let $\mathcal{L}'_{\max}$ denote the set that has greater or equal cardinality, and $\mathcal{L}'_{\min}$ the other one (Lines 16–17). For each atom from $\mathcal{L}'_{\min}$ we choose its most similar atom from $\mathcal{L}'_{\max}$, and take the sum of the similarities of these pairs of atoms divided by the cardinality of $\mathcal{L}'_{\max}$ (fraction in Line 18). Finally, the similarity between o_1 and o_2 is computed as the sum of the above similarities for every $(q', k) \in \mathcal{P}_1 \cup \mathcal{P}_2$ divided by the cardinality of $\mathcal{P}_1 \cup \mathcal{P}_2$ (Line 20).

4. A similarity measure for lists and terms

In order to directly compute similarity of instances with non-flat components (lists, terms), we have chosen to rely on the idea of *edit distances* between objects. Intuitively, with an edit distance, we are given a set of edit operations (insert, delete, change) with an associated cost function δ that tells us how expensive it is to delete or insert an element of our alphabet Σ , or how expensive it is to change one element into another. The edit distance is given by the smallest cost sequence of such operations that turns the first object into the second, and there are efficient algorithms for computing it (see below).

Input: l_1, l_2 are q -atoms of depth d , $1 \leq i \leq \rho(q)$.
Output: Normalized similarity between the i th arguments of l_1, l_2 , respectively.

```

1: let  $o_1 = l_1[i], o_2 = l_2[i]$ 
2: if  $\text{Type}(q[i]) = \text{number}(a, b)$  then
3:   return  $1 - \frac{|o_1 - o_2|}{b - a}$ 
4: else if  $\text{Type}(q[i]) = \text{discrete}$  then
5:   return 1 if  $o_1 = o_2$ , and 0 otherwise
6: else if  $\text{Type}(q[i]) = \text{object}$  then
7:   if  $d = \text{DEPTH}$  then
8:     return 1 if  $o_1 = o_2$ , and 0 otherwise
9:   else if  $o_1 = o_2$  then
10:    return 1
11:  else
12:    let  $\mathcal{L}_j = \{l \in \mathcal{B}(e_j, d+1) : o_j \text{ occurs in } l\}$ 
13:    for  $j = 1, 2$  let
       $\mathcal{P}_j = \{(q', k) : q' \text{ is a background predicate, } 1 \leq k \leq \rho(q'),$ 
       $\text{there exists a } q'\text{-atom } l \in \mathcal{L}_j \text{ such that } l[k] = o_j\}$ 

14:     $S = 0$ 
15:    for all  $(q', k) \in \mathcal{P}_1 \cup \mathcal{P}_2$  do
16:      let  $\mathcal{L}'_j = \{l \in \mathcal{L}_j : l \text{ is a } q'\text{-atom, and } l[k] = o_j\}$  for  $j = 1, 2$ 
17:      let  $\mathcal{L}'_{\min}$  be the smaller set from  $\mathcal{L}'_1, \mathcal{L}'_2$ , and let  $\mathcal{L}'_{\max}$  denote
      the other one
18:       $S = S + \frac{\sum_{l_x \in \mathcal{L}'_{\min}} \max_{l_y \in \mathcal{L}'_{\max}} \text{SIM}_{\text{lit}}(\mathcal{B}, e_1, e_2, d+1, l_x, l_y)}{|\mathcal{L}'_{\max}|}$ 
      where
      
$$\text{SIM}_{\text{lit}}(\mathcal{B}, e_1, e_2, d+1, l_x, l_y) =$$


$$\frac{1}{|I|} \sum_{i' \in I} \text{SIMILARITY}(\mathcal{B}, e_1, e_2, d+1, l_x, l_y, i')$$

      with  $I = \{u : 1 \leq u \leq \rho(q'), l_x[u] \neq o_1 \vee l_y[u] \neq o_2\}$ 
19:    end for
20:    return  $\frac{S}{|\mathcal{P}_1 \cup \mathcal{P}_2|}$ 
21:  end if
22: end if

```

Algorithm 1: $\text{SIMILARITY}(\mathcal{B}, e_1, e_2, d, l_1, l_2, i)$

In the following, we will precisely specify the edit distance measure used in the current version of RIBL (henceforth referred to as RIBL 2.0). Even though both constants and lists could be considered as terms, we distinguish them from each other for the sake of efficiency (see Section 4.4).

4.1. Constants

An argument type `constant` is given by a set of constant symbols Σ_{constant} and a 1-bounded semi-metric δ on Σ_{constant} . The *distance* between $a, b \in \Sigma_{\text{constant}}$ is then defined directly by δ , i.e.,

$$\text{DIST}_{\text{constant}}(a, b) = \delta(a, b).$$

We note that the argument type `discrete` can be considered as a special case of this type.

4.2. Lists

Let $\mathcal{L}(\Sigma_{\text{list}})$ be the set of finite lists over the alphabet Σ_{list} , and let $\Sigma_{\text{list}, \perp} = \Sigma_{\text{list}} \cup \{\perp\}$, where $\perp$ denotes the *empty list*. $|l|$ denotes the *length* of the list $l \in \mathcal{L}(\Sigma_{\text{list}})$. Consider the following operations on the elements of $\mathcal{L}(\Sigma_{\text{list}})$:

- *insert*(l, i, a): inserts the symbol $a \in \Sigma_{\text{list}}$ before the i -th symbol in l ,
- *delete*(l, i): deletes the i -th symbol in l ,
- *change*(l, i, a): changes the i -th symbol in l to a new symbol a

for every $a \in \Sigma_{\text{list}}$, $l \in \mathcal{L}(\Sigma_{\text{list}})$ and $i = 1, \dots, |l|$. Thus, an operation can be considered as an application of a *rewriting rule* $a \rightarrow b$ for $a, b \in \Sigma_{\text{list}, \perp}$, i.e., insert, delete and change correspond to the application of a rewriting rule $\perp \rightarrow a$, $a \rightarrow \perp$ and $a \rightarrow b$, respectively ($a, b \neq \perp$).

Let $l_1, l_2 \in \mathcal{L}(\Sigma_{\text{list}})$. We say that l_2 can be *directly derived* from l_1 (denoted by $l_1 \Rightarrow l_2$) if there exists an operation that produces l_2 from l_1 . The *derivation* (denoted by $l_1 \Rightarrow_{\sigma_1 \dots \sigma_n}^* l_2$) is the *reflexive, transitive closure* of $\Rightarrow$. The index $\sigma_1 \dots \sigma_n$ denotes the sequence of the applied rewriting rules. A `list` type is given by a 1-bounded metric space $(\Sigma_{\text{list}, \perp}, \delta)$. Here, $\delta(a \rightarrow b)$ is considered as the *cost* of the rewriting rule $a \rightarrow b$ for every $a, b \in \Sigma_{\text{list}, \perp}$. Since δ is a metric, it can be given by a symmetric *cost matrix* with zero elements in its diagonal. The *edit* or *Levenshtein distance*² between $l_1, l_2 \in \mathcal{L}(\Sigma_{\text{list}})$ is defined by

$$\begin{aligned} & \text{EDITDIST}_{\text{list}}(l_1, l_2) \\ &= \min \left\{ \sum_{i=1}^n \delta(\sigma_i) : \exists \sigma_1, \dots, \sigma_n \text{ such that } l_1 \Rightarrow_{\sigma_1 \dots \sigma_n}^* l_2 \right\}. \end{aligned}$$

Since δ is a distance metric on $\Sigma_{\text{list}, \perp}$, $\text{EDITDIST}_{\text{list}}$ is a distance metric on $\mathcal{L}(\Sigma_{\text{list}})$. The next theorem is a special case of Theorems 2 and 3 in Wagner and Fischer (1974):

Theorem 1. Let $l_1 = [a_1, \dots, a_n], l_2 = [b_1, \dots, b_m] \in \mathcal{L}(\Sigma_{\text{list}})$, and $(\Sigma_{\text{list}, \perp}, \delta)$ be a metric space. Let $D_{i,j}$ denote the edit distance between $[a_1, \dots, a_i]$, and $[b_1, \dots, b_j]$

Table 1. The distance matrix $D_{i,j}$.

	$\perp$	c	u	u	a	g	c	u	c
$\perp$	0	1	2	3	4	5	6	7	8
c	1	0	1	2	3	4	5	6	7
c	2	1	1	2	3	4	4	5	6
u	3	2	1	1	2	3	4	4	5
a	4	3	2	2	1	2	3	4	5

for $i = 0, \dots, n$, and $j = 0, \dots, m$. Then

$$D_{0,0} = 0, \quad D_{0,j} = \sum_{k=1}^j \delta(\perp \rightarrow b_k), \quad D_{i,0} = \sum_{k=1}^i \delta(a_k \rightarrow \perp), \quad \text{and}$$

$$D_{i,j} = \min(D_{i-1,j-1} + \delta(a_i \rightarrow b_j),$$

$$D_{i-1,j} + \delta(a_i \rightarrow \perp), D_{i,j-1} + \delta(\perp \rightarrow b_j))$$

for every $i, j > 0$.

By Theorem 1, $\text{EDITDIST}_{\text{list}}(l_1, l_2)$ can be computed in *time* and *space* $O(mn)$.

Example 2. Let $\Sigma_{\text{list}} = \{a, c, g, u\}$, and δ be the trivial metric on $\Sigma_{\text{list}, \perp}$. Let $l_1 = [c, c, u, a]$, and $l_2 = [c, u, u, a, g, c, u, c]$. The value of $D_{i,j}$ for $i = 0, \dots, 4$, and $j = 0, \dots, 8$ is given in Table 1. The edit distance is $\text{EDITDIST}_{\text{list}}(l_1, l_2) = D_{4,8} = 5$.

An improved algorithm for computing the edit distance between lists and the next theorem can be found in Ukkonen (1985).

Theorem 3. The edit distance $d = \text{EDITDIST}_{\text{list}}(l_1, l_2)$ between lists l_1 and l_2 can be computed in *time* and *space*

$$O(d \cdot \min(|l_1|, |l_2|)) \text{ and } O(\min(d, |l_1|, |l_2|)),$$

respectively.

We define distance between lists in RIBL as:

$$\text{DIST}_{\text{list}}(l_1, l_2) = \frac{\text{EDITDIST}_{\text{list}}(l_1, l_2)}{\max(|l_1|, |l_2|)}. \quad (6)$$

Proposition 4. If $(\Sigma_{\text{list}, \perp}, \delta)$ is a 1-bounded metric space then $\text{DIST}_{\text{list}}$ is a 1-bounded semi-metric on $\mathcal{L}(\Sigma_{\text{list}})$.

Proof: Since $\text{EDITDIST}_{\text{list}}$ is a metric on $\mathcal{L}(\Sigma_{\text{list}})$, $\text{DIST}_{\text{list}}$ satisfies Conditions (1)–(3) defined in Section 2.1. We show that (6) violates the triangle inequality. Consider the alphabet $\Sigma_{\text{list}} = \{a, b\}$ with the trivial metric δ . Then

$$\begin{aligned} \text{DIST}_{\text{list}}([a, b], [b, a]) &= 2/2 \\ &> \text{DIST}_{\text{list}}([a, b], [a, b, a]) \\ &\quad + \text{DIST}_{\text{list}}([a, b, a], [b, a]) \\ &= 1/3 + 1/3. \end{aligned}$$

As δ is a 1-bounded metric on $\Sigma_{\text{list}, \perp}$,

$$\text{EDITDIST}_{\text{list}}(l_1, l_2) \leq \max(|l_1|, |l_2|)$$

for every $l_1, l_2 \in \mathcal{L}(\Sigma_{\text{list}})$. Hence, $\text{DIST}_{\text{list}}$ is also 1-bounded. $\square$

4.3. Terms

In RIBL 2.0, terms are considered as *labeled ordered trees* and hence, the distance between terms is computed as the *edit distance* between labeled ordered trees.

Let Σ_{term} be an alphabet. We note that Σ_{term} is not a *ranked* alphabet, i.e., no arity function is assigned to Σ_{term} . The set of all *finite* terms with nodes labeled by the elements of Σ_{term} including the *empty* term ($\perp$) is denoted by $\mathcal{T}(\Sigma_{\text{term}})$.

For a given term $t \in \mathcal{T}(\Sigma_{\text{term}})$, $|t|$ denotes the number of nodes of t , and $t[i]$ ($1 \leq i \leq |t|$) denotes the i th node of t according to the preorder traversal of t . Let t_1 , and t_2 be terms from $\mathcal{T}(\Sigma_{\text{term}})$. A *mapping* from t_1 to t_2 , denoted by $M_{t_1 \rightarrow t_2}$, is a subset of $\{1, \dots, |t_1|\} \times \{1, \dots, |t_2|\}$ such that for every $(i_1, j_1), (i_2, j_2) \in M$ it holds that

- (i) $i_1 = i_2$ iff $j_1 = j_2$,
- (ii) $i_1 < i_2$ iff $j_1 < j_2$,
- (iii) $t_1[i_1]$ is an ancestor (resp. descendant) of $t_2[i_2]$ iff $t_1[j_1]$ is an ancestor (resp. descendant) of $t_2[j_2]$.

A mapping $M_{t_1 \rightarrow t_2}$ can be considered as a one-to-one correspondence between subsets of nodes of t_1 and t_2 , respectively, such that by deleting the 'untouched' nodes from t_1 and t_2 , respectively, the remaining terms are identical up to relabeling. For a detailed discussion of mapping the reader is referred to Tai (1979). If there is no confusion we define a mapping by using the nodes' labels instead of their positions according to the preorder walk. Furthermore, if t_1 and t_2 follow from the context we write M instead of $M_{t_1 \rightarrow t_2}$.

Example 5. Consider the terms

$$t_1 = a(b, c(d, e(f, g), h)) \quad \text{and} \quad t_2 = a'(c(d, e(f, g), h), b').$$

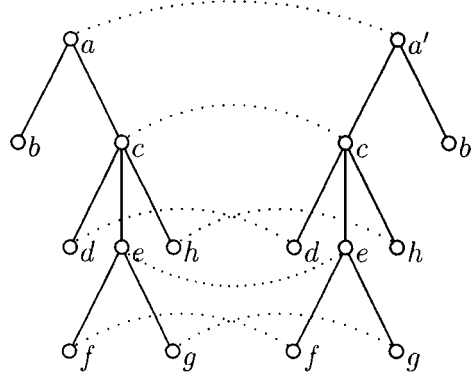


Figure 2. Mapping between two terms.

Then $M_1 = \{(a, a'), (b, c)\}$ is a mapping from t_1 to t_2 , as the corresponding positions $\{(1, 1), (2, 2)\}$ according to preorder traversal satisfy conditions (i)–(iii). Note that the trees $a(b)$ and $a'(c)$ obtained from t_1 by deleting the nodes c, d, e, f, g, h and from t_2 by deleting b', d, e, f, g, h , respectively, are identical up to relabeling.

$M_2 = \{(a, a'), (c, c), (d, d), (e, e), (h, h), (f, f), (g, g)\}$ is also a mapping between t_1 , and t_2 (see figure 2).

A mapping M between terms t_1 , and t_2 defines a transformation from t_1 into t_2 by the following sequence of *edit* operations:

- *delete* each untouched node from t_1 (i.e., each node x of t_1 such that there is no pair $(x, y) \in M$ for any y), then
- *insert* each untouched node of t_2 with its corresponding label (i.e., each node y of t_2 such that there is no pair $(x, y) \in M$ for any x), and then
- for every $(x, y) \in M$ such that x, y have different labels, *replace* the label of x by the label of y .

The above edit operations on terms can also be considered as applications of *term rewriting rules* $a \rightarrow b$ ($a, b \in \Sigma_{\text{term}, \perp}$ and $a \rightarrow b \neq \perp \rightarrow \perp$). Let δ be a 1-bounded metric on $\Sigma_{\text{term}, \perp}$. As for lists, δ is considered as a *cost matrix* for the term rewriting rules. The *cost* of a mapping M between t_1, t_2 is the sum of the costs of the applied term rewriting rules transforming t_1 into t_2 . The *edit distance* between t_1, t_2 , denoted by $\text{EDITDIST}_{\text{term}}(t_1, t_2)$, is the cost of the mapping with the smallest cost with respect to δ . Since $(\Sigma_{\text{term}, \perp}, \delta)$ is a metric space, $(\mathcal{T}(\Sigma_{\text{term}}), \text{EDITDIST}_{\text{term}})$ is also a metric space.

The algorithm for computing the edit distance between terms and the following theorem can be found in Zhang and Shasha (1989) (we denote by $\text{depth}(t)$, and $\text{leaves}(t)$ the depth, and the number of leaves of the term t , respectively):

Theorem 6. *The edit distance between terms t_1 and t_2 can be computed in time*

$$O(|t_1| \times |t_2| \times \min(\text{depth}(t_1), \text{leaves}(t_1)) \times \min(\text{depth}(t_2), \text{leaves}(t_2)))$$

and space $O(|t_1| \times |t_2|)$.

In contrast to lists, in the next example we show that $\max(|t_1|, |t_2|)$ is *not* an upper bound on $\text{EDITDIST}_{\text{term}}$.

Example 7. Let δ be the trivial metric on $\{\perp, a, b, c\}$, and consider the terms $t_1 = a(a \cdots (a)) \cdots$ (i.e., a *chain* of a 's) and $t_2 = b(c, \dots, c)$ with $|t_1| = |t_2| = n$. The edit distance between the two terms is $2n - 2$.

Although $|t_1| + |t_2|$ is an upper bound on $\text{EDITDIST}_{\text{term}}(t_1, t_2)$ for every 1-bounded metric space $(\Sigma_{\text{term}}, \perp, \delta)$, and $t_1, t_2 \in \mathcal{T}(\Sigma_{\text{term}})$, we normalize with another function measuring the *structural difference* of t_1 , and t_2 and giving a tighter upper bound.

Let $M_{t_1 \rightarrow t_2}$ be a mapping. By the definition of mapping, M is a set of ordered pairs. We denote by $\mathcal{U}_1(M)$ (resp. $\mathcal{U}_2(M)$) the set of nodes of t_1 (resp. t_2) *untouched* by M . The *size* of a mapping is defined by

$$\text{size}(M) = |M| + |\mathcal{U}_1(M)| + |\mathcal{U}_2(M)|. \quad (7)$$

The *optimal mapping* $M_{t_1 \rightarrow t_2}^{\text{OPT}}$ is a mapping with the smallest size.

In RIBL distance between terms is defined by

$$\text{DIST}_{\text{term}}(t_1, t_2) = \frac{\text{EDITDIST}_{\text{term}}(t_1, t_2)}{\text{size}(M_{t_1 \rightarrow t_2}^{\text{OPT}})}. \quad (8)$$

Example 8. Let $t_1 = a(b, c(d))$, and $t_2 = a(b(d))$ be terms.

$$\text{EDITDIST}_{\text{term}}(t_1, t_2) = 2$$

using the trivial metric. The optimal mapping is

$$M_{t_1 \rightarrow t_2}^{\text{OPT}} = \{(a, a), (c, b), (d, d)\}.$$

Since $\mathcal{U}_1(M_{t_1 \rightarrow t_2}^{\text{OPT}}) = \{b\}$, and $\mathcal{U}_2(M_{t_1 \rightarrow t_2}^{\text{OPT}}) = \emptyset$,

$$\text{size}(M_{t_1 \rightarrow t_2}^{\text{OPT}}) = 3 + 1 + 0$$

by (7). Applying (8) we get

$$\text{DIST}_{\text{term}}(t_1, t_2) = 2/4.$$

In RIBL, $\text{size}(M_{t_1 \rightarrow t_2}^{\text{OPT}})$ is computed as follows: let t_1/a (resp. t_2/b) denote the terms obtained from t_1 (resp. t_2) by replacing the label of each node with a (resp. b). Then

$$\text{size}(M_{t_1 \rightarrow t_2}^{\text{OPT}}) = \text{EDITDIST}_{\text{term}}(t_1/a, t_2/b)$$

using the trivial metric on $\{\perp, a, b\}$.

Proposition 9. *If $(\Sigma_{\text{term}, \perp}, \delta)$ is a 1-bounded metric space then $\text{DIST}_{\text{term}}$ is a 1-bounded semi-metric on $\mathcal{T}(\Sigma_{\text{term}})$.*

Proof: Since $\text{DIST}_{\text{term}}$ is a metric on $\mathcal{T}(\Sigma_{\text{term}})$, and the size of a mapping between terms is symmetric, (8) satisfy Conditions (1)–(3). However, it does not satisfy the triangle inequality, e.g. let $(\{\perp, a, b\}, \delta)$ be a trivial metric space, and consider the terms $a(b)$, and $b(a)$. Then

$$\begin{aligned} \text{DIST}_{\text{term}}(a(b), b(a)) &= 2/2 \\ &> \text{DIST}_{\text{term}}(a(b), a(b(a))) \\ &\quad + \text{DIST}_{\text{term}}(a(b(a)), b(a)) \\ &= 1/3 + 1/3. \end{aligned}$$

It remains to be proven that $\text{DIST}_{\text{term}}$ is 1-bounded. Let t_1, t_2 be terms, and $M_{t_1 \rightarrow t_2}$ be any mapping. Since δ is 1-bounded,

$$\text{size}(M_{t_1 \rightarrow t_2}) \geq \text{EDITDIST}_{\text{term}}(t_1, t_2). \quad (9)$$

As M is any arbitrary mapping from t_1 to t_2 , (9) holds for $M_{t_1 \rightarrow t_2}^{\text{OPT}}$ as well. $\square$

4.4. Lists as terms

The standard term representation of lists is given by the function symbol $./2$, and the empty list $\perp$ (or $[]$ in the standard Prolog notation). However, in this representation, $\text{DIST}_{\text{list}}$ and $\text{DIST}_{\text{term}}$ may differ on the same lists, e.g.

$$\begin{aligned} 2/2 &= \text{DIST}_{\text{list}}([a, b], [b, a]) \\ &\neq \text{DIST}_{\text{term}}(.(a, .(b, \perp)), .(b, .(a, \perp))) \\ &= 2/5 \end{aligned}$$

using the trivial metric on $\{\perp, a, b\}$.

As an alternative, let $l = [a_1, \dots, a_n]$ be a list in $\mathcal{L}(\Sigma_{\text{list}})$. The *chain term* representation of l is $a_1(a_2(\dots(a_n)\dots))$. Let l_1^c , and l_2^c denote the chain representations of $l_1 = [a_1, \dots, a_n]$,

and $l_2 = [b_1, \dots, b_m]$, respectively. Since $\text{EDITDIST}_{\text{list}}(l_1, l_2) = \text{EDITDIST}_{\text{term}}(l_1^c, l_2^c)$, and $\max(|l_1|, |l_2|) = \text{size}(M_{l_1^c \rightarrow l_2^c}^{\text{OPT}})$,

$$\text{DIST}_{\text{list}}(l_1, l_2) = \text{DIST}_{\text{term}}(l_1^c, l_2^c)$$

for every metric used in both representations.

Applying Theorem 6 to l_1^c , and l_2^c , the edit distance of lists in the chain term representation can be computed in $O(mn)$. The reason why we handle lists and terms separately is that the time and space complexity for computing edit distance between lists given in Ukkonen (1985) is always as good as the complexity of computing their edit distance in the chain term representation, and the norm function of lists can be computed in linear time.

4.5. Defining δ for the alphabets

In Sections 4.2 and 4.3, we have treated list and term distances as basic (non-recursive) distance functions which rely on metrics δ for comparing individual symbols from the alphabet. Clearly, one possibility is to use user-defined metrics which allows the encoding of additional domain knowledge of the user. Another attractive possibility is to use RIBL's basic idea again, and compute the editing cost of two symbols by using RIBL's similarity measure.

To do this, the elements of the corresponding alphabet must be considered as objects that are further described in the background knowledge. Each object is considered as an instance (without a target value). In contrast to the original IBL prediction problem, we are interested only in the similarities between such instances. These similarity values are then used for determining the editing costs between the corresponding symbols.

The editing costs δ on an alphabet Σ can be computed automatically by the following two steps:

1. we compute a *semi-metric* δ' on Σ by recursively recalling RIBL's similarity measure, i.e.,

$$\delta'(a, b) := 1 - \text{SIMILARITY}(\mathcal{B}, (a), (b), 0, (a), (b), 1)$$

(we use notation (x) for denoting the instance corresponding to x),

2. and then we transform it into a *metric* δ by

$$\delta(a, b) = \min(1, \delta'(a, b) + C) \tag{10}$$

for every $a, b \in \Sigma_{\text{term}, \perp}$, where C is the maximum of

$$\frac{\delta'(x, y) - \delta'(x, z) - \delta'(z, y)}{2}$$

for every $x, y, z \in \Sigma$ violating the triangle inequality.

5. Empirical evaluation

In this section we discuss our experiments with the revised version of RIBL (RIBL 2.0) in a real-world domain, the *mRNA signal structure prediction* problem. The experiments are designed to show that the addition of lists and terms enables the easy construction of background knowledge representing complex objects, and that this representation leads to excellent results on a significant real-world problem, greatly improving on simpler, RIBL 1.0-style background knowledge. In fact, the mRNA application can also be considered a success in terms of the application domain (Bohnebeck et al., 1998b).

5.1. mRNA signal structures

Protein synthesis involves a *translation* step of copying specific regions (the genes) of DNA into a complementary strand of RNA. RNA molecules that *mediate* between the spatially distinct localizations of information storage (i.e., the DNA) and protein synthesis are called *messenger RNA* (mRNA). Beside the genetic information on the amino acid sequence of proteins, mRNA contains two regions, called *untranslated regions*, that do not code for protein. These untranslated regions contain continuous parts, called *signal structures*, that control gene expression at the posttranscriptional level. Depending on their biological functions (i.e., regulatory roles during protein synthesis), signal structures are grouped into different signal structure *classes*.

In general, RNA is composed of a linear sequence of the four bases *adenine* (A), *cytosine* (C), *guanine* (G), and *uracil* (U). This sequence is its *primary structure*. In contrast to DNA, RNA is a *single-stranded*, and relatively short molecule. The linear sequence of an RNA folds back on itself thus giving the RNA a *three-dimensional* conformation. The three-dimensional conformation contains local regions of short complementary base-pairing, i.e., the two stable Watson-Crick base pairs A-U, G-C, and the weak G-U. The *secondary structure* of an RNA is defined as the collection of the base pairs occurring in its three-dimensional structure (see Figure 3). Such a secondary structure can be viewed as a *tree* (Shapiro & Zhang, 1990) with nodes, also called *structure elements*, corresponding to continuous regions that belong to basic *patterns*.³ This observation holds for signal structures as well (i.e., the secondary structure of a signal structure can be considered as a tree), as their primary structures are (continuous) subsequences of the RNA's primary structure (see also the right upper part of Figure 3 for a signal structure).

Since the biological function of a given signal structure class is based on the common characteristic binding site of its instances, signal structures belonging to the same class are not necessarily identical but *very similar* to each other. They can slightly vary in their topology, in the sizes of their structure elements, and in their primary structures (see, e.g., Figure 4 for instances of the Iron Responsive Element signal structure class).

5.2. The learning task and data

In this field of research, one of the challenging problems for computational biology is to *detect* instances of known signal structure classes on uncharacterized mRNA. In order to

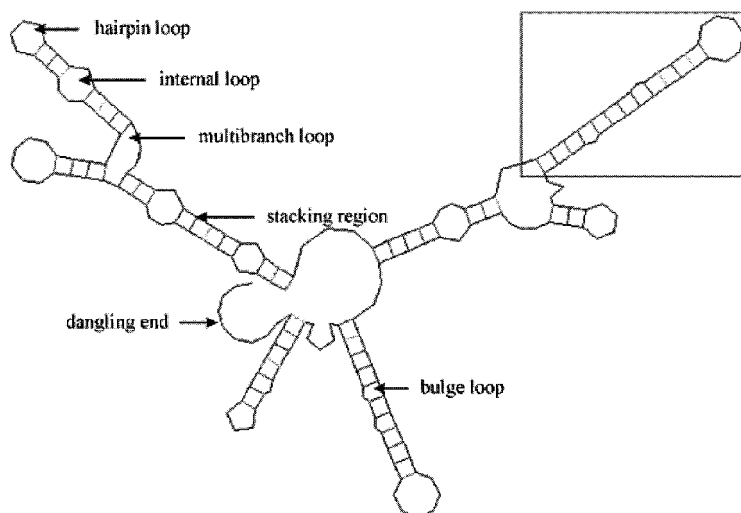


Figure 3. mRNA structure with marked structure elements.

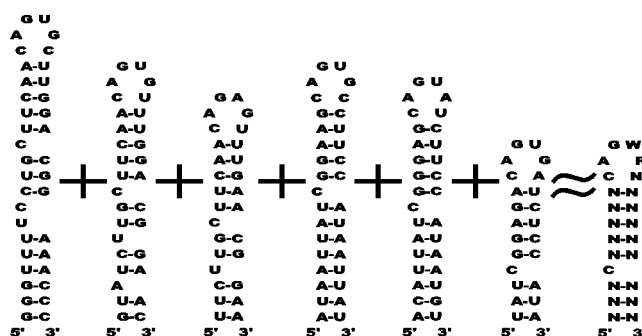


Figure 4. Example for IRE signal structure instances and the corresponding consensus structure described in the IUB-IUPAC code.

solve this problem, one has to consider the related problem of *predicting* the class of an uncharacterized signal structure. That is, we are *given*

- a set of signal structures E , each identified by a single attribute of type object, and background knowledge $\mathcal{B}$ describing the structure of each instance with structure elements as determined by biological analysis software,
- each signal structure in E is associated with a target value representing its signal structure class,
- a set of uncharacterized (i.e., without class information) signal structures E_{new} , each described in the same way as E .

The aim is to *find* a prediction of the signal structure class (target value) for each $e \in E_{new}$.

For our experiments, two different example sets were used. The first example set consists of 66 signal structures, composed of 15 IRE (Iron Responsive Element) (Klausner, Rouault & Harford, 1993), 15 TAR (Trans Activating Region) (McCarthy & Kollmus, 1995), 2×15 SECIS (Selenocysteine Insertion Sequence) (Low & Berry, 1996), and 6 histone stemloops (Belasco & Brawerman, 1993). This example set was used for detailed comparative experiments with different representations and background knowledge. To confirm the scalability of the accuracy figures that were obtained, a second experiment was carried out on a larger example set containing 400 signal structures by using the term representation (Bohnebeck et al., 1998b).

5.3. Background knowledge and representations

As mentioned above, training and unseen signal structure instances were described in each representation by a single attribute of type `object` pointing to facts on structure information in the background knowledge. Thus, the difference between the experiments is in the complexity and representation of the background knowledge.

5.3.1. Flat representations. In these representations we have used neither lists, nor terms. The most simple description of signal structures is the *chain* representation, in which they are considered as simple sequences of structure elements. Depending on its type (i.e., is it a *single* sequence, or a *helical* consisting of two attached sequences), the i th structure element of a signal structure is described by a background fact of the form

$$\begin{aligned} &<se_type>(struct_id, i, size, bases) \text{ or} \\ &<se_type>(struct_id, i, size, bases, bases) \end{aligned}$$

where $<se_type>$ is a background predicate symbol from the set {stem, hairpin, bulge5, bulge3, single} defining the type of the i th structure element (i is of type `number`). Argument *size* of type `number` stores the length of the primary structure of the corresponding structure element. Finally, *bases* of type `discrete` is a special symbol representing the nucleotid sequence of the i th structure element (i.e., the whole sequence is coded by a single symbol in this representation).

The other flat representation we investigated is the *neighbor* represent, which is the *chain* representation extended by additional information on the neighborhoods of structure elements: for each element, both neighboring elements were represented explicitly.

5.3.2. Representations with lists. The two representations used here are basically the same as in the previous case, but instead of type `discrete`, we used type `list` (over the alphabet $\{A, C, G, U\}$) for describing sequences in arguments *bases*.

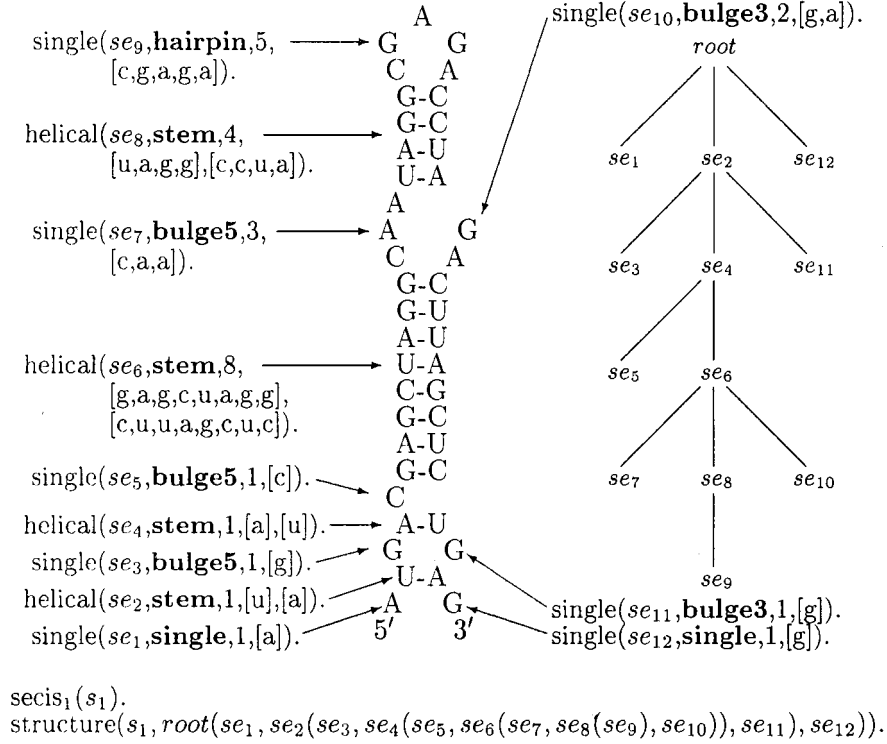


Figure 5. The representation and the tree structure of a SECIS signal structure.

5.3.3. Representation with terms. In this representation (see also Figure 5) we described the tree structure of signal structures by using background predicates of the form

structure(struct_id, topology)
single(se_id, type, size, bases)
helical(se_id, type, size, bases, bases).

Argument *topology* of type *term* is the tree structure of the signal structure identified by *struct_id*. Its nodes are structure elements identified by the argument *se_id*, and are further described by the last two predicates. Depending on its type, a structure element is represented by either the *single*/4 or the *helical*/5 predicate. Their second argument of type *constant* is used to describe the type of the structure element identified by argument *se_id* (this information was stored by using different predicate symbols in the flat representations). Finally, the last arguments *size*, and *bases* are the same as in the case of representations with lists.

5.4. Defining δ for the different types

In this particular application, we have used a manual definition of δ for constants (referred to as δ_{constant} in the following), the trivial edit costs for the list alphabet $\{A, C, G, U\}$, and

the recursively computed editing cost as defined in Section 4 for the nodes of terms. The values of δ_{constant} were as follows:

	$\perp$	single	bulge3	bulge5	hairpin	stem
$\perp$	0	0.03	0.03	0.03	0.90	0.03
single		0	0.03	0.03	0.18	1.0
bulge3			0	0.06	0.12	1.0
bulge5				0	0.12	1.0
hairpin					0	1.0
stem						0

These are heuristic costs and still under development. Two properties are essential. First, helical and single-stranded regions are morphologically substantially different and should therefore not be mapped. Secondly the hairpin loop takes up an exceptional position because its insertion or deletion produces also a quite substantial morphological difference.

5.5. Results and discussion

In order to compare the different levels of representation and background knowledge, experiments were performed using the smaller of the two example sets. Each experiment was performed using leave-one-out cross-validation, determining the best value of k on the training instances, and then classifying the held-out instance. The following table shows the results, where times were taken on a Sun Ultrasparc with OS Solaris 2.5.

background knowledge/ representation	depth	accuracy (%)	k	cpu time (sec)
chain	1	77.2	1	0.5
chain with lists	1	78.8	1	4.3
neighbor	2	83.3	3	2.8
neighbor with lists	2	83.3	1	15.5
term	1	95.4	1	32.8

As can be seen in this table, making the list and term background knowledge available has had significant influence on the obtained accuracy, ultimately reaching a level which can be considered as a success in the domain (Bohnebeck et al., 1998b). To confirm the scalability of this result, the final experiment was repeated in the larger example set containing 400 signal structures, yielding a similar level of accuracy:

background knowledge/ representation	depth	accuracy (%)	k	cpu time (sec)
term	1	96.2	3	9861.0

As for differences between experiments, besides the remarkable overall increase in accuracy, we see that in this particular domain, the information contained in the base sequences represented by lists did not produce a marked difference in accuracy. To understand why this is so, recall that in the *chain* representations a structure was basically considered as a linear ordered set of its structure elements. The linear ordering was described by an attribute of type *number* thus not supporting the direct reference to a next or previous structure element. This was improved by the *neighbor* representation. The whole topology was described adequately by the *term* representation. In other words, as suspected by the domain experts, the mRNA signal structure problem appears to be inherently structural in nature, and the topology of the molecule, which in RIBL 2.0 can easily be represented using terms, seems to play an important role in prediction.

The experiments also show a marked increase in computation time when terms are used. This is due to the optimizing nature of the edit distance measure which considers the optimal mapping of one term to another. Clearly, this leads to very effective use of the information contained in terms, but may mean that running time becomes problematic for very large applications. This is thus one of the concerns for future work (order-of-magnitude optimizations appear possible e.g. through the use of distance caching), and we are planning to work on a re-implementation of the system.

6. Related work

As an IBL system, RIBL is of course related to the large body of work on instance-based learning, e.g., Aha et al. (1991), Salzberg (1991), Wettschereck and Dietterich (1995). For all variants of IBL that are exclusively based on similarity measures, these variants could be used as a basis for RIBL without problems. For feature weighting approaches (Wettschereck et al., 1997), as pointed out above, the more powerful representation of RIBL requires different weighting schemes, which have been developed by Emde and Wettschereck (1996). Finally, IBL approaches that rely on generalized prototypes or generalized regions in instance space (Salzberg, 1991) do not carry over directly, since in a first-order space, different generalization operators would have to be used. Of course, it would be an interesting topic of further work to see whether for example restricted first-order LGGs (least-general generalizations) are suitable for this purpose.

As for the distance measure used in RIBL, it is most closely related to the recursive computations used in the KBG system (Bisson, 1992a); there, however, they were not used as the basis of an instance-based learning approach, but rather to select objects for generalization. Furthermore, RIBL's similarity measure can be regarded as a generalization of propositional similarity measures, which is not the case in Bisson (1992a).

Recently, several other measures for first-order distances have appeared which we briefly want to contrast with RIBL's measure. Two different ways are distinguished in the literature of distance definitions for first-order logic: *semantical* (Sebag, 1997) and *syntactical* (Hutchinson, 1997; Nienhuys-Cheng, 1997; Nienhuys-Cheng, 1998; Ramon & Bruynooghe, 1998) distances. A semantical distance maps the first-order space into another space by redescribing it in terms of a number of given or learned elementary concepts. Thus, the form of description of the instance is not directly used in similarity computation.

RIBL belongs to the second direction, the syntactical measures, which compute distances directly from the representation of instances. In contrast to RIBL, where distance between atoms is recursively computed based on other background knowledge atoms, most other syntactical first-order distances are based on introducing an appropriate basic metric or pseudo-metric on ground expressions (i.e., on terms and atoms), and then using this basic metric to induce a metric on clauses. Thus, it appears useful to compare such basic metrics on terms with RIBL's term distance metric which is based on edit distances.

6.1. Distances on terms

First, we discuss the relevant results on distances between expressions. By using simple examples, we also show that the behavior of the term editing distance differs fundamentally from the term distances of Hutchinson (1997), Nienhuys-Cheng (1997).

In Hutchinson (1997), a family of pseudo-metrics between terms t_1 and t_2 is defined by

$$D_1(t_1, t_2) = S(\theta_1) + S(\theta_2)$$

where θ_i is the *most general unifier* of t_i and $\text{lgg}(t_1, t_2)$ (i.e., the *least general generalization* (Plotkin, 1970)) for $i = 1, 2$, and S is a *size* function on substitutions (see Hutchinson, 1997 for the details).

For two terms with small edit distance, the pseudo-distance with the size function S on substitutions given in Hutchinson (1997) may be unreasonably large. Another property of this pseudo-metric is that the distance between the standard term representations ($n : s^n(\perp)$) of two different natural numbers is always a constant depending on S , while it is a function of n and m , in our case (see Section 6.1.1).

In Nienhuys-Cheng (1997), the 1-bounded recursive metric D_2 is defined as follows:

$$D_2(f(s_1, \dots, s_n), g(t_1, \dots, t_m)) = \begin{cases} 0 & \text{if } f = g, n = m \text{ and } t_i = s_i \quad \text{for } i = 1, \dots, n \\ \frac{1}{2n} \sum_{i=1}^n D_2(s_i, t_i) & \text{if } f = g \text{ and } n = m, s_i \neq t_i \quad \text{for some } 1 \leq i \leq n \\ 1 & \text{otherwise} \end{cases}$$

The idea is that the positions of the differences are implicitly weighted (i.e., distance on lower position has higher weight). Considering the term representations of lists, (Nienhuys-Cheng, 1997) does not take into account the *suffix* of the longer list, i.e., the distance between l_1 and l_2 is equal to the distance between l_1 and $l_2 + l$, for all lists l_1, l_2, l with $|l_1| \leq |l_2|$. Furthermore, it may happen that terms with big and other terms with small edit distances may have very close distances by Nienhuys-Cheng (1997). In the case of distance between different natural numbers, the result is independent of the larger number (see Section 6.1.1).

6.1.1. Quantitative comparison of metrics on examples. Since D_1 is a pseudo-metric with no upper bound, and D_2 is a 1-bounded metric on terms, we compare D_1 and D_2 with $\text{EDITDIST}_{\text{term}}$ and $\text{DIST}_{\text{term}}$, respectively. We consider the following three cases:

Table 2. Comparison of different distances on terms.

Terms	D_1	EDITDIST _{term}	D_2	DIST _{term}
$(t_{1,1}, t_{1,2})$	$2n + 1$	2	$\frac{1+2^{5-2n}}{3}$	$\frac{2}{2n-1}$
$(t_{2,1}, t_{2,2})$	2	n	$\frac{1+2^{3-2n}}{3}$	$\frac{n}{2n-1}$
$(t_{3,1}, t_{3,2})$	2	$n - m$	2^{-m}	$\frac{n-m}{n+1}$

1. Let $f, a_1, \dots, a_n$ ($n > 1$) be different symbols and consider the terms

$$\begin{aligned} t_{1,1} &= f(a_1, f(a_2, \dots f(a_{n-1}, a_n) \dots)) \\ t_{1,2} &= f(a_2, f(a_3, \dots f(a_{n-1}, a_n) \dots)). \end{aligned}$$

2. Let f, a_1, a_2 be different symbols and consider the terms

$$\begin{aligned} t_{2,1} &= f(a_1, f(a_1, \dots f(a_1, a_1) \dots)) \\ t_{2,2} &= f(a_2, f(a_2, \dots f(a_2, a_2) \dots)) \end{aligned}$$

with $|t_{2,1}| = |t_{2,2}| = 2n - 1$ ($n > 1$).

3. Let $s^n(\perp)$ denote the n applications of the unary function symbol s (successor) to $\perp$, i.e., $s^n(\perp)$ denotes the integer n ($n > 0$). Let $t_{3,1} = s^n(\perp)$ and $t_{3,2} = s^m(\perp)$ for some $n > m \geq 0$ (all the distances are 0 for $n = m$).

Table 2 contains a comparison of the different term metrics on the above terms (for D_1 we used the size function S of Hutchinson (1997) with $wt_{\perp} = 0$ and $wt_{a_i} = 1$ for $i = 1, \dots, n$ and the trivial metrics for the edit distances).

6.2. Distances between sets of literals

One of the most frequently used extension of a distance between expressions to a distance between sets of expressions is the Hausdorff-metric (Hutchinson, 1997; Nienhuys-Cheng, 1997) defining the distance between sets S_1 , and S_2 by

$$\max \left(\max_{x \in S_1} \min_{y \in S_2} \text{DIST}(x, y), \max_{x \in S_2} \min_{y \in S_1} \text{DIST}(x, y) \right),$$

where DIST is a metric. Although it can be computed in $O(|S_1| \cdot |S_2|)$ for finite sets, and remains a metric, by reducing the distance between sets to a distance between two chosen points it “loses” the information on the whole sets.

Following the approach of Eiter and Mannila (1997), the distance between two sets of atoms is considered as a *weighted matching* problem in Ramon and Bruynooghe (1998), where the weights are defined by the distance between the literals. Although this distance

has a number of nice properties, its complexity is cubic in contrast to RIBL's quadratic complexity (see Lines 17–18 in Algorithm 1).

7. Conclusion

In this paper, we have presented the most recent version of the relational instance-based learning system RIBL (RIBL 2.0). After giving a detailed reconstruction of the original RIBL algorithm and similarity measure which was limited to function-free representations, we presented an enhanced similarity measure that can directly handle lists and terms in instance descriptions. As shown by experiments in the domain of mRNA signal structure classification, significant additional power is gained by adding list and term descriptions to the background knowledge. Lists and terms have also proven useful in experiments in other domains, such as part-of-speech tagging for the Hungarian language (Horváth et al., 1999).

While these experiments, and previous experiments with the first version of RIBL, indicate that RIBL's similarity measure is general enough to produce excellent results on a variety of domains, we must of course emphasize that there can be no “best” similarity measure across all problems and goals, so that in other situations, other similarity measures might be preferable. For example, in contrast to Hutchinson (1997), Nienhuys-Cheng (1997), RIBL's distance measure is *stable*, meaning that small changes on a term do not cause big distances. This is usually desirable, but may not always be appropriate. Finally, as with most ILP systems, learning success for RIBL also depends on how background knowledge is represented; here, the addition of lists and terms increases flexibility and allows the user to better fit the background knowledge to the problem domain.

As for future work, above we have already remarked upon further potential for complexity optimization. Firstly, RIBL 2.0 currently uses a non-optimized implementation that does not use indexing nor caching, so that order of magnitude improvements appear possible by eliminating recomputation of previously computed distances. Secondly, further improvements are possible within the algorithm for similarity computation which currently checks all literal matches to find the best partner for each literal. Here it would be possible to use a stochastic approach instead, along the lines of the stochastic matching principle introduced by Sebag and Rouveirol (1997), Sebag (1998). Thirdly, the size of an optimal mapping between terms t_1 , and t_2 is currently computed in RIBL by $\text{EDITDIST}_{\text{term}}(t_1/a, t_2/b)$ using the trivial metric on $\{\perp, a, b\}$. As this is a special case of Theorem 6, it might be possible to find a faster algorithm here.

Furthermore, there are some areas where it appears attractive to further enhance the current similarity measure. Firstly, throughout this paper terms have been considered as labeled ordered trees, i.e., the arities of the function symbols (labels) have not been taken into account. In certain situations, it might be desirable to be able to differentiate between different arities or even automatically extend a given distance measure to varying arities. Secondly, in this paper, lists and terms have been defined above alphabets consisting of only *constants*. It appears interesting to extend the similarity metric to *arbitrary* lists and terms, i.e., where the alphabet belongs to an *aggregate* type, so that non-constant functors would also be allowed.

Acknowledgments

We would like to thank Werner Emde and Dietrich Wettschereck for allowing work to continue on RIBL. We are grateful to Tobias Scheffer for proofreading and commentary and to Gábor Keresztes and Joachim Selbig for discussions on the application. We thank Mathias Kirsten for technical support of the implementation. Uta Bohnbeck's biology colleagues in Bremen, Werner Sälter and Dietmar Blohm, were essential to the mRNA application. The first and the second authors were partially supported by the ESPRIT IV Long Term Research Project ILP II (No. 20237).

Notes

1. In RIBL, examples are actually represented by first-order atoms, and the target value can be provided as an additional argument of this atom, or by using several different example predicates representing different classes. We omit these variants for simplicity.
2. Some authors (e.g. Aho, 1990) use different definitions for the edit and for the Levenshtein distances, respectively.
3. The most frequently used structure elements are *stacking regions*, *hairpin loops*, *internal loops*, *bulge loops*, *multibranch loops*, and *dangling ends* (Zuker, 1989).

References

- Aha, D. (1997). *Lazy learning*. Dordrecht: Kluwer Academic Publishers.
- Aha, D., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1), 37–66.
- Aho, A. (1990). Algorithms for finding patterns in strings. In J. van Leeuwen (Ed.), *Handbook of theoretical computer science, algorithms and complexity*, (Vol. A). Amsterdam: Elsevier.
- Belasco, J. G. & Brawerman G. (1993). *Control of Messenger RNA Stability*. Oxford: Academic Press.
- Bisson, G. (1992a). Conceptual clustering in a first-order logic representation. In *Proceedings of the Tenth European Conference on Artificial Intelligence* (pp. 458–462). Chichester: John Wiley and Sons.
- Bisson, G. (1992b). Learning in FOL with a similarity measure. In *Proceedings of the Tenth National Conference on Artificial Intelligence* (pp. 82–87). Cambridge: The MIT Press.
- Bohnebeck, U., Horváth, T., & Wrobel, S. (1998a). Term comparisons in first-order similarity measures. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, (pp. 65–79). Berlin: Springer. Vol. 1446 of *Lecture Notes in Artificial Intelligence*.
- Bohnebeck, U., Sälter, W., Horváth, T., Wrobel, S., & Blohm, D. (1998b). Measuring similarity of RNA structures by relational instance-based learning: A first step toward detecting RNA signal structures in silico. In *Proceedings of the German Conference on Bioinformatics*. Technical Report, Universität Köln.
- Džeroski, S., Schulze-Kremer, S., Heidtke, K., Siems, K., & Wettschereck, D. (1996). Diterpene structure elucidation from ¹³C NMR spectra with machine learning. In N. Lavrač, E. Keravnou, & B. Zupan (Eds.), *Intelligent data analysis in medicine and pharmacology*, Dordrecht: Kluwer Academic Publishers.
- Eiter, T. & Mannila, H. (1997). Distance measures for point sets and their computation. *Acta Informatica*, 34(2), 109–133.
- Emde, W. & Wettschereck, D. (1996). Relational instance-based learning. In *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 122–130). San Mateo: Morgan Kaufmann.
- Horváth, T., Alexin, Z., Gyimóthy, T., & Wrobel, S. (1999). Application of different learning methods to Hungarian part-of-speech tagging. In *Proceedings of the Ninth International Workshop on Inductive Logic Programming*, (pp. 128–139). Berlin: Springer. Vol. 1634 of *Lecture Notes in Artificial Intelligence*.
- Hutchinson, A. (1997). Metrics on terms and clauses. In *Proceedings of the Ninth European Conference on Machine Learning*, (pp. 138–145). Berlin: Springer. Vol. 1224 of *Lecture Notes in Artificial Intelligence*.

- Klausner, R. D., Rouault, T. A., & Harford J. B. (1993). Regulating the fate of mRNA: The control of cellular iron metabolism. *Cell*, 72, 19–28.
- Low, S. L. & Berry, M. J. (1996). Knowing when not to stop: Selenocysteine incorporation in eukaryotes. *Trends in Biochemistry Sciences*, 21, 203–208.
- McCarthy, J. E. G. & Kollmus, H. (1995). Cytoplasmic mRNA-protein interactions in eukaryotic gene expression. *Trends in Biochemistry Sciences*, 20, 191–197.
- Muggleton, S. & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20, 629–679.
- Nienhuys-Cheng, S.-H. (1997). Distance Between Herbrand Interpretations: A measure for approximations to a target concept. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, (pp. 213–226). Berlin: Springer. Vol. 1297 of *Lecture Notes in Artificial Intelligence*.
- Nienhuys-Cheng, S.-H. (1998). Distances and limits on Herbrand interpretations. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, (pp. 250–260). Berlin: Springer. Vol. 1446 of *Lecture Notes in Artificial Intelligence*.
- Plotkin, G. (1970). A note on inductive generalization. In B. Meltzer & D. Michie (Eds.), *Machine intelligence*, (Vol. 5). Edinburgh University Press.
- Ramon, J. & Bruynooghe, M. (1998). A framework for defining distances between first-order logic objects. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, (pp. 271–280). Berlin: Springer. Vol. 1446 of *Lecture Notes in Artificial Intelligence*.
- Salzberg, S. (1991). A nearest hyperrectangle learning method. *Machine Learning*, 6, 251–276.
- Sebag, M. (1997). Distance induction in first order logic. In *Proceedings of the Seventh International Workshop on Inductive Logic Programming*, (pp. 264–272). Berlin: Springer. Vol. 1297 of *Lecture Notes in Artificial Intelligence*.
- Sebag, M. (1998). A Stochastic Simple Similarity. In *Proceedings of the Eighth International Conference on Inductive Logic Programming*, (pp. 95–105). Berlin: Springer. Vol. 1446 of *Lecture Notes in Artificial Intelligence*.
- Sebag, M. & Rouveirol, C. (1997). Tractable induction and classification in first order logic via stochastic matching. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. (pp. 888–893). San Francisco: Morgan Kaufmann.
- Shapiro, B. A. & Zhang, K. (1990). Comparing multiple RNA secondary structures using tree comparisons. *Computer Applications in Biosciences*, 6(4), 309–318.
- Tai, K.-C. (1979). The tree-to-tree correction problem. *Journal of the ACM*, 26(3), 422–433.
- Ukkonen, E. (1985). Algorithms for approximate string matching. *Information and Control*, 64, 100–118.
- Wagner, R. A. & Fischer, M. J. (1974). The string-to-string correction problem. *Journal of the ACM*, 21(1), 168–173.
- Wettschereck, D. & Dietterich, T. G. (1995). An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, 19(1), 5–27.
- Wettschereck, D., Mohri, T., & Aha, D. W. (1997). A review and comparative evaluation of feature weighting methods for lazy learning algorithms. *AI Review Journal*, 11, 273–314.
- Wrobel, S. (1996). Inductive logic programming. In G. Brewka (Ed.), *Advances in knowledge representation and reasoning*. Studies in Logic, Language and Information. Stanford, CSLI-Publishers.
- Zhang, K. & Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6), 1245–1262.
- Zuker, M. (1989). On finding all suboptimal foldings of an RNA Molecule, *Science* 244, 48–52.

Received March 31, 1999

Accepted September 8, 1999

Final manuscript June 11, 2000

APPENDIX CARUANA01



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)

Share

Bookmark

Report an Issue

MARC View

LEADER 04308nas a2200625 a 4500

001 1295069

003 SIRSI

005 20151215075954.0

008 980904c19899999cauar 1 a0eng d

010 a| 90644910 z| sn89033004

019 a| *Itemize*

019 a| MARS

022 0 a| 1049-5258

035 a| LIAS1523086

035 a| (OCoLC)19891678

040 c| PSt d| WaOLN d| UtOrBLW

050 0 0 a| QA76.87 b| .A38

245 0 0 a| Advances in neural information processing systems.

246 1 3 a| NIPS

264 1 a| San Mateo, CA : b| Morgan Kaufmann Publishers, c| [1989-]

264 4 c| ©1989-

300 a| volumes : b| illustrations ; c| 24 cm

310 a| Annual

336 a| text b| txt 2| rdacontent

337 a| unmediated b| n 2| rdamedia

338 a| volume b| nc 2| rdacarrier

362 0 a| 1-

500 a| Vols. 1- contain the collected papers of the 1988- IEEE Conference on Neural Information Processing Systems--Natural and Synthetic.

500 a| Published: Cambridge, Mass. : MIT Press, 1995-

515 a| None held 2001.

515 a| Vol. 14 (2002) issued in 2 vols.

650 0 a| Neural computers v| Periodicals.

650 0 a| Neural networks (Computer science) v| Periodicals.

711 2 a| IEEE Conference on Neural Information Processing Systems--Natural and Synthetic.

936 a| 8, Proceedings of the 1995 Conference LIC.

949 a| QP363.3.A43 v.1 w| LCPER c| 1 i| 21520703003 d| 6/30/2001 e| 9/7/1995 l| CATO-PARK m| UP-ANNEX n| 25 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0703 1| 003

949 a| QP363.3.A43 v.2 w| LCPER c| 1 i| 21520703002 d| 9/13/2000 e| 3/1/2001 l| CATO-PARK m| UP-ANNEX n| 26 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0703 1| 002

949 a| QP363.3.A43 v.3 w| LCPER c| 1 i| 21520703001 d| 7/22/2013 e| 3/7/2000 l| CATO-PARK m| UP-ANNEX n| 18 q| 2 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0703 1| 001

949 a| QP363.3.A43 v.4 w| LCPER c| 1 i| 21510702008 d| 9/2/2005 e| 8/31/2005 l| CATO-PARK m| UP-ANNEX n| 17 q| 1 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2151 0| 0702 1| 008

949 a| QP363.3.A43 v.5 w| LCPER c| 1 i| 21510702007 d| 7/26/2005 e| 12/14/2000 l| CATO-PARK m| UP-ANNEX n| 9 q| 1 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2151 0| 0702 1| 007

949 a| QP363.3.A43 v.6 w| LCPER c| 1 i| 21510702006 d| 9/13/2000 e| 5/6/1998 l| CATO-PARK m| UP-ANNEX n| 6 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2151 0| 0702 1| 006

949 a| QP363.3.A43 v.7 w| LCPER c| 1 i| 21520404007 d| 2/3/2003 e| 9/4/2002 l| CATO-PARK m| UP-ANNEX n| 13 q| 2 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0404 1| 007

949 a| QP363.3.A43 v.8 w| LCPER c| 1 i| 21520403009 d| 8/9/2004 e| 5/7/2004 l| CATO-PARK m| UP-ANNEX n| 6 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0403 1| 009

949 a| QP363.3.A43 v.10 w| LCPER c| 1 i| 21520403007 d| 9/13/2000 e| 5/14/2001 l| CATO-PARK m| UP-ANNEX n| 2 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0403 1| 007

949 a| QP363.3.A43 v.9 w| LCPER c| 1 i| 21520403008 d| 10/3/2003 e| 5/12/1998 l| CATO-PARK m| UP-ANNEX n| 1 q| 2 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0403 1| 008

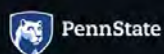
949 a| QP363.3.A43 v.11 w| LCPER c| 1 i| 21520404006 d| 7/15/2003 e| 7/7/2003 l| CATO-PARK m| UP-ANNEX n| 4 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0404 1| 006

949 a| QP363.3.A43 v.12 w| LCPER c| 1 i| 21520404005 d| 3/19/2004 e| 12/16/2003 l| CATO-PARK m| UP-ANNEX n| 10 q| 1 r| Y s| Y t| SERIAL u| 4/13/2001 x| C z| 2152 0| 0404 1| 005

949 a| QP363.3.A43 v.13 2000 w| LCPER c| 1 i| 21520404004 d| 12/17/2003 e| 12/16/2003 l| CATO-PARK m| UP-ANNEX n| 6 r| Y s| Y t| SERIAL u| 10/15/2001 x| C z| 2152 0| 0404 1| 004



949		a	QP363.3.A43 v.14 2001 v.1 w	LCPER c	1 i	21520404003 d	1/19/2010 e	1/4/2010 l	CATO-PARK m	UP-ANNEX n	6 r	Y s	Y t
		SERIAL u	6/10/2003 x	C z	2152 0	0404 1	003						
949		a	QP363.3.A43 v.14 2001 v.2 w	LCPER c	1 i	21520404002 d	2/3/2010 e	1/4/2010 l	CATO-PARK m	UP-ANNEX n	6 q	1 r	Y s
		SERIAL u	6/10/2003 x	C z	2152 0	0404 1	002						Y t
949		a	QP363.3.A43 v.15 2002 w	LCPER c	1 i	21520404001 d	9/1/2004 e	8/9/2004 l	CATO-PARK m	UP-ANNEX n	3 r	Y s	Y t
		SERIAL u	11/5/2003 x	C z	2152 0	0404 1	001						
949		a	QP363.3.A43 v.16 2003 w	LCPER c	1 i	21510802013 d	1/25/2010 e	12/14/2009 l	CATO-PARK m	UP-ANNEX n	13 r	Y s	Y t
		SERIAL u	11/3/2004 x	C z	2151 0	0802 1	013						
949		a	QP363.3.A43 v.17 2004 w	LCPER c	1 i	21510802012 d	7/2/2009 e	5/4/2009 l	CATO-PARK m	UP-ANNEX n	12 r	Y s	Y t
		SERIAL u	7/1/2005 x	C z	2151 0	0802 1	012						
949		a	QP363.3.A43 v.18 2005 w	LCPER c	1 i	21510802011 d	7/2/2009 e	5/4/2009 l	CATO-PARK m	UP-ANNEX n	8 r	Y s	Y t
		SERIAL u	8/11/2006 x	C z	2151 0	0802 1	011						
949		a	QP363.3.A43 v.19 2006 w	LCPER c	1 i	000061068902 d	3/4/2024 e	1/8/2010 l	PATERNO-4 m	UP-PAT n	1 r	Y s	Y t
		SERIAL u	1/10/2008										
852	0	1	a	PSt b	UP-ANNEX c	ANNEX h	QP363.3 i	.A43					
866	0	8	0 a	v.1-v.18(2005)									
852	0		a	PSt b	UP-PAT c	PATERNO-4 h	QP363.3 i	.A43					
866	0	8	0 a	v.19(2006)									



Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#) (814) 865-6368
[Libraries Intranet \(Staff Only\)](#)
[Accessibility Help](#)
[Website Feedback](#)
[Policies and Guidelines](#)
[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#)
[Twitter](#)
[Instagram](#)

Keyword  Search...Search [Advanced search](#) [Start Over](#)[Share](#) [Bookmark](#)[Report an Issue](#)

Advances in neural information processing systems

Additional Titles

NIPS

Published

San Mateo, CA : Morgan Kaufmann Publishers, [1989-]

Copyright Date

©1989-

Physical Description

volumes : illustrations ; 24 cm

Additional Creators

[IEEE Conference on Neural Information Processing Systems--Natural and Synthetic](#)

Availability

[Browse Nearby on Shelf](#)[I Want It](#)

Pattee Library and Paterno Library Stacks (1 item)

Call number	Material	Location
Paterno - 4th Floor: Holdings Summary		
v.19(2006)		
QP363.3.A43 v.19 2006	Book	Paterno - 4th Floor

Library Storage (19 items)

Call number	Material	Location
Annexed Material: Holdings Summary		
v.1-v.18(2005)		
QP363.3.A43 v.18 2005	Book	Annexed Material
QP363.3.A43 v.17 2004	Book	Annexed Material
QP363.3.A43 v.16 2003	Book	Annexed Material
QP363.3.A43 v.15 2002	Book	Annexed Material
QP363.3.A43 v.14 2001 v.2	Book	Annexed Material
QP363.3.A43 v.14 2001 v.1	Book	Annexed Material
QP363.3.A43 v.13 2000	Book	Annexed Material
QP363.3.A43 v.12	Book	Annexed Material
QP363.3.A43 v.11	Book	Annexed Material
QP363.3.A43 v.10	Book	Annexed Material
QP363.3.A43 v.9	Book	Annexed Material
QP363.3.A43 v.8	Book	Annexed Material
QP363.3.A43 v.7	Book	Annexed Material
QP363.3.A43 v.6	Book	Annexed Material
QP363.3.A43 v.5	Book	Annexed Material
QP363.3.A43 v.4	Book	Annexed Material
QP363.3.A43 v.3	Book	Annexed Material
QP363.3.A43 v.2	Book	Annexed Material
QP363.3.A43 v.1	Book	Annexed Material

 ASK
A LIBRARIAN

Dates of Publication and/or Sequential Designation

1-

Subject(s)[Neural computers—Periodicals](#)[Neural networks \(Computer science\)—Periodicals](#)**ISSN**

1049-5258

Publication Frequency

Annual

Note

Vols. 1- contain the collected papers of the 1988- IEEE Conference on Neural Information Processing Systems--Natural and Synthetic.

Published: Cambridge, Mass. : MIT Press, 1995-

Special Numbering

None held 2001.

Vol. 14 (2002) issued in 2 vols.

[View MARC record](#) | catkey: 1295069**PennState**

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)**PENN STATE
UNIVERSITY LIBRARIES**[Libraries Home](#)

(814) 865-6368

[Libraries Intranet \(Staff
Only\)](#)[Accessibility Help](#)[Website Feedback](#)[Policies and Guidelines](#)[Staff Directory](#)**CONNECT WITH
PENN STATE UNIVERSITY
LIBRARIES**[Facebook](#)[Twitter](#)[Instagram](#)

APPENDIX CARUANA02

PSU LIBRARIES ANNEX

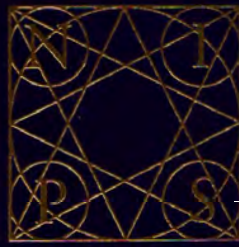


2152 04 04 004



ADVANCES IN
NEURAL
INFORMATION
PROCESSING
SYSTEMS 13

EDITED BY
TODD K. LEEN,
THOMAS G. DIETTERICH,
AND VOLKER TRESP



EDITORS
LEEN
DIETTERICH
TRESP

ADVANCES IN
NEURAL INFORMATION
PROCESSING SYSTEMS

13

Engin.
QP363
.3
.A43
v.13
2000

The MIT Press
Massachusetts Institute of Technology
Cambridge, Massachusetts 02142
<http://mitpress.mit.edu>



LEEAH 0-262-12241-3

ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 13

ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 13

Proceedings of the 2000 Conference

edited by

Todd K. Leen, Thomas G. Dietterich and Volker Tresp

A Bradford Book
The MIT Press
Cambridge, Massachusetts
London, England

ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 12

© 2001 Massachusetts Institute of Technology

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means (including photocopying, recording or information storage and retrieval) without permission in writing from the publisher.

This book was printed and bound in the United States of America.

ISSN: 1049-5258

ISBN: 0-262-12241-3

Edited by
Todd K. Leen, Thomas G. Dietterich, and Robert E. Long

Volume 12

Published by the MIT Press, Cambridge, MA 02142, USA
and by Cambridge University Press, The Edinburgh Building, Shaftesbury Road, Cambridge CB2 2RU, UK

1999

Printed in the United States of America
David S. Touretzky, MIT, Cambridge, MA 02142, USA

1998

Published by the MIT Press, Cambridge, MA 02142, USA
Michael C. Mozer, Michael J. Jordan, and Thomas G. Dietterich, MIT, 1997

1997

Published by the MIT Press, Cambridge, MA 02142, USA
Michael J. Jordan, Michael C. Mozer, and Thomas G. Dietterich, MIT, 1996

1996

Published by the MIT Press, Cambridge, MA 02142, USA
Michael J. Jordan, Michael C. Mozer, and Thomas G. Dietterich, MIT, 1995

1995

Published by the MIT Press, Cambridge, MA 02142, USA
Michael J. Jordan, Michael C. Mozer, and Thomas G. Dietterich, MIT, 1994

1994

Published by the MIT Press, Cambridge, MA 02142, USA
Michael J. Jordan, Michael C. Mozer, and Thomas G. Dietterich, MIT, 1993
Cambridge, Massachusetts
London, England

Contents

Preface	xv
NIPS Committees	xvii
Reviewers	xix

Part I Cognitive Science

<i>Who Does What? A Novel Algorithm to Determine Function Localization,</i> Ranit Aharonov-Barki, Isaac Meilijson and Eytan Ruppin	3
<i>A Productive, Systematic Framework for the Representation of Visual Structure,</i> Shimon Edelman and Nathan Intrator	10
<i>The Interplay of Symbolic and Subsymbolic Processes in Anagram Problem Solving,</i> David B. Grimes and Michael C. Mozer	17
<i>Hippocampally-Dependent Consolidation in a Hierarchical Model of Neocortex,</i> Szabolcs Káli and Peter Dayan	24
<i>Position Variance, Recurrence and Perceptual Learning,</i> Zhaoping Li and Peter Dayan	31
<i>The Use of MDL to Select among Computational Models of Cognition,</i> In J. Myung, Mark A. Pitt, Shaobo Zhang and Vijay Balasubramanian	38
<i>Active Inference in Concept Learning,</i> Jonathan D. Nelson and Javier R. Movellan .	45
<i>The Early Word Catches the Weights,</i> Mark A. Smith, Garrison W. Cottrell and Karen L. Anderson	52
<i>Structure Learning in Human Causal Induction,</i> Joshua B. Tenenbaum and Thomas L. Griffiths	59
<i>Adaptive Object Representation with Hierarchically-Distributed Memory Sites,</i> Bosco S. Tjan	66

Part II Neuroscience

<i>What Can a Single Neuron Compute?,</i> Blaise Agüera y Arcas, Adrienne L. Fairhall and William Bialek	75
<i>Dendritic Compartmentalization Could Underlie Competition and Attentional Biasing of Simultaneous Visual Stimuli,</i> Kevin A. Archie and Bartlett W. Mel . . .	82
<i>Place Cells and Spatial Navigation Based on 2D Visual Feature Extraction, Path Integration, and Reinforcement Learning,</i> Angelo Arleo, Fabrizio Smeraldi, Stéphane Hug and Wulfram Gerstner	89
<i>Modelling Spatial Recall, Mental Imagery and Neglect,</i> Suzanna Becker and Neil Burgess	96

<i>Stability and Noise in Biochemical Switches</i> , William Bialek	103
<i>Temporally Dependent Plasticity: An Information Theoretic Account</i> , Gal Chechik and Naftali Tishby	110
<i>A New Model of Spatial Representation in Multimodal Brain Areas</i> , Sophie Deneve, Jean-Rene Duhamel and Alexandre Pouget	117
<i>Multiple Timescales of Adaptation in a Neural Code</i> , Adrienne L. Fairhall, Geoffrey D. Lewen, William Bialek and Robert R. de Ruyter van Steveninck	124
<i>Dopamine Bonuses</i> , Sham Kakade and Peter Dayan	131
<i>Finding the Key to a Synapse</i> , Thomas Natschl�ger and Wolfgang Maass	138
<i>Processing of Time Series by Neural Circuits with Biologically Realistic Synaptic Dynamics</i> , Thomas Natschl�ger, Wolfgang Maass, Eduardo D. Sontag and Anthony Zador . . .	145
<i>Spike-Timing-Dependent Learning for Oscillatory Networks</i> , Silvia Scarpetta, Zhaoping Li and John Hertz	152
<i>Universality and Individuality in a Neural Code</i> , Elad Schneidman, Naama Brenner, Naftali Tishby, Robert R. de Ruyter van Steveninck and William Bialek	159
<i>Natural Sound Statistics and Divisive Normalization in the Auditory System</i> , Odelia Schwartz and Eero P. Simoncelli	166
<i>Development of Hybrid Systems: Interfacing a Silicon Neuron to a Leech Heart Interneuron</i> , Mario F. Simoni, Gennady S. Cymbalyuk, Michael Q. Sorensen, Ronald L. Calabrese and Stephen P. DeWeerth	173
<i>Whence Sparseness?</i> , Carl van Vreeswijk	180

Part III Theory

<i>Efficient Learning of Linear Perceptrons</i> , Shai Ben-David and Hans Ulrich Simon .	189
<i>Algorithmic Stability and Generalization Performance</i> , Olivier Bousquet and Andr� Elisseeff	196
<i>Competition and Arbors in Ocular Dominance</i> , Peter Dayan	203
<i>From Margin to Sparsity</i> , Thore Graepel, Ralf Herbrich and Robert C. Williamson .	210
<i>Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks</i> , Richard H.R. Hahnloser and H. Sebastian Seung	217
<i>A PAC-Bayesian Margin Bound for Linear Classifiers: Why SVMs work</i> , Ralf Herbrich and Thore Graepel	224
<i>On Reversing Jensen's Inequality</i> , Tony Jebara and Alex Pentland	231
<i>Second Order Approximations for Probability Models</i> , Hilbert J. Kappen and Wim Wiegerinck	238

<i>Some New Bounds on the Generalization Error of Combined Classifiers,</i> Vladimir Koltchinskii, Dmitriy Panchenko and Fernando Lozano	245
<i>Sparsity of Data Representation of Optimal Kernel Machine and Leave-one-out Estimator,</i> Adam Kowalczyk	252
<i>Foundations for a Circuit Complexity Theory of Sensory Processing,</i> Robert A. Legenstein and Wolfgang Maass	259
<i>A Tighter Bound for Graphical Models,</i> Martijn A. R. Leisink and Hilbert J. Kappen	266
<i>Learning Curves for Gaussian Processes Regression: A Framework for Good Approximations,</i> Dörthe Malzahn and Manfred Opper	273
<i>Weak Learners and Improved Rates of Convergence in Boosting,</i> Shie Mannor and Ron Meir	280
<i>Learning Continuous Distributions: Simulations With Field Theoretic Priors,</i> Ilya Nemenman and William Bialek	287
<i>Occam's Razor,</i> Carl Edward Rasmussen and Zoubin Ghahramani	294
<i>The Kernel Trick for Distances,</i> Bernhard Schölkopf	301
<i>Regularization with Dot-Product Kernels,</i> Alex J. Smola, Zoltán L. Óvári and Robert C. Williamson	308
<i>Analysis of Bit Error Probability of Direct-Sequence CDMA Multiuser Demodulators,</i> Toshiyuki Tanaka	315
<i>Error-correcting Codes on a Bethe-like Lattice,</i> Renato Vicente, David Saad and Yoshiyuki Kabashima	322
<i>Algebraic Information Geometry for Learning Machines with Singularities,</i> Sumio Watanabe	329
<i>Computing with Finite and Infinite Networks,</i> Ole Winther	336
<i>Stagewise Processing in Error-correcting Codes and Image Restoration,</i> K. Y. Michael Wong and Hidetoshi Nishimori	343
<i>Learning Winner-take-all Competition Between Groups of Neurons in Lateral Inhibitory Networks,</i> Xiaohui Xie, Richard H.R. Hahnloser and H. Sebastian Seung	350
<i>Convergence of Large Margin Separable Linear Classification,</i> Tong Zhang	357

Part IV Algorithms and Architecture

<i>A Support Vector Method for Clustering,</i> Asa Ben-Hur, David Horn, Hava T. Siegelmann and Vladimir Vapnik	367
<i>A Variational Mean-Field Theory for Sigmoidal Belief Networks,</i> Chiranjib Bhattacharyya and S. Sathiya Keerthi	374
<i>Direct Classification with Indirect Data,</i> Timothy X Brown	381
<i>Model Complexity, Goodness of Fit and Diminishing Returns,</i> Igor V. Cadez and Padhraic Smyth	388

<i>A Linear Programming Approach to Novelty Detection,</i> Colin Campbell and Kristin P. Bennett	395
<i>Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping,</i> Rich Caruana, Steve Lawrence and Lee Giles	402
<i>Incremental and Decremental Support Vector Machine Learning,</i> Gert Cauwenberghs and Tomaso Poggio	409
<i>Vicinal Risk Minimization,</i> Olivier Chapelle, Jason Weston, Léon Bottou and Vladimir Vapnik	416
<i>Gaussianization,</i> Scott Saobing Chen and Ramesh A. Gopinath	423
<i>The Missing Link - A Probabilistic Model of Document Content and Hypertext Connectivity,</i> David Cohn and Thomas Hofmann	430
<i>Improved Output Coding for Classification Using Continuous Relaxation,</i> Koby Crammer and Yoram Singer	437
<i>Sparse Representation for Gaussian Process Models,</i> Lehel Csató and Manfred Opper	444
<i>Explaining Away in Weight Space,</i> Peter Dayan and Sham Kakade	451
<i>An Adaptive Metric Machine for Pattern Classification,</i> Carlotta Domeniconi, Jing Peng and Dimitrios Gunopulos	458
<i>High-temperature Expansions for Learning Models of Nonnegative Data,</i> Oliver B. Downs	465
<i>Incorporating Second-Order Functional Knowledge for Better Option Pricing,</i> Charles Dugas, Yoshua Bengio, François Bédille, Claude Nadeau and René Garcia	472
<i>Discovering Hidden Variables: A Structure-Based Approach,</i> Gal Elidan, Noam Lotner, Nir Friedman and Daphne Koller	479
<i>Accumulator Networks: Suitors of Local Probability Propagation,</i> Brendan J. Frey and Anitha Kannan	486
<i>Sequentially Fitting "Inclusive" Trees for Inference in Noisy-OR Networks,</i> Brendan J. Frey, Relu Patrascu, Tommi S. Jaakkola and Jodi Moran	493
<i>A New Approximate Maximal Margin Classification Algorithm,</i> Claudio Gentile	500
<i>Propagation Algorithms for Variational Bayesian Learning,</i> Zoubin Ghahramani and Matthew J. Beal	507
<i>The Kernel Gibbs Sampler,</i> Thore Graepel and Ralf Herbrich	514
<i>'N-Body' Problems in Statistical Learning,</i> Alexander G. Gray and Andrew W. Moore	521
<i>Large Scale Bayes Point Machines,</i> Ralf Herbrich and Thore Graepel	528
<i>Beyond Maximum Likelihood and Density Estimation: A Sample-Based Criterion for Unsupervised Learning of Complex Models,</i> Sepp Hochreiter and Michael C. Mozer	535

<i>Ensemble Learning and Linear Response Theory for ICA,</i> Pedro A.d.F.R. Højen-Sørensen, Ole Winther and Lars Kai Hansen	542
<i>Generalizable Singular Value Decomposition for Ill-posed Datasets,</i> Ulrik Kjems, Lars Kai Hansen and Stephen C. Strother	549
<i>Algorithms for Non-negative Matrix Factorization,</i> Daniel D. Lee and H. Sebastian Seung	556
<i>Text Classification using String Kernels,</i> Huma Lodhi, John Shawe-Taylor, Nello Cristianini and Chris Watkins	563
<i>Constrained Independent Component Analysis,</i> Wei Lu and Jagath C. Rajapakse . .	570
<i>Active Support Vector Machine Classification,</i> Olvi L. Mangasarian and David R. Musicant	577
<i>The Unscented Particle Filter,</i> Rudoiph van der Merwe, Arnaud Doucet, Nando de Freitas and Eric Wan	584
<i>A Mathematical Programming Approach to the Kernel Fisher Algorithm,</i> Sebastian Mika, Gunnar Rätsch and Klaus-Robert Müller	591
<i>Automatic Choice of Dimensionality for PCA,</i> Thomas P. Minka	598
<i>On Iterative Krylov-Dogleg Trust-Region Steps for Solving Neural Networks</i> <i>Nonlinear Least Squares Problems,</i> Eiji Mizutani and James W. Demmel	605
<i>An Information Maximization Approach to Overcomplete and Recurrent</i> <i>Representations,</i> Oren Shriki, Haim Sompolsky and Daniel D. Lee	612
<i>Sparse Greedy Gaussian Process Regression,</i> Alex J. Smola and Peter Bartlett . . .	619
<i>Kernel Expansions with Unlabeled Examples,</i> Martin Szummer and Tommi S. Jaakkola	626
<i>Sparse Kernel Principal Component Analysis,</i> Michael E. Tipping	633
<i>Data Clustering by Markovian Relaxation and the Information Bottleneck</i> <i>Method,</i> Naftali Tishby and Noam Slonim	640
<i>Active Learning for Parameter Estimation in Bayesian Networks,</i> Simon Tong and Daphne Koller	647
<i>Mixtures of Gaussian Processes,</i> Volker Tresp	654
<i>Tree-Based Modeling and Estimation of Gaussian Processes on Graphs with</i> <i>Cycles,</i> Martin J. Wainwright, Erik B. Sudderth and Alan S. Willsky	661
<i>Feature Selection for SVMs,</i> Jason Weston, Sayan Mukherjee, Olivier Chapelle, Massimiliano Pontil, Tomaso Poggio and Vladimir Vapnik	668
<i>On a Connection between Kernel PCA and Metric Multidimensional Scaling,</i> Christopher K. I. Williams	675
<i>Using the Nyström Method to Speed Up Kernel Machines,</i> Christopher K. I. Williams and Matthias Seeger	682
<i>Generalized Belief Propagation,</i> Jonathan S. Yedidia, William T. Freeman and Yair Weiss	689

<i>A Gradient-Based Boosting Algorithm for Regression Problems,</i> Richard S. Zemel and Toniann Pitassi	696
<i>Regularized Winnow Methods,</i> Tong Zhang	703

Part V Implementation

<i>A Silicon Primitive for Competitive Learning,</i> David Hsu, Miguel Figueroa and Chris Diorio	713
<i>Smart Vision Chip Fabricated Using Three Dimensional Integration Technology,</i> Hiroyuki Kurino, M. Nakagawa, Kang Wook Lee, Tomonori Nakamura, Yuusuke Yamada, Ki Tae Park and Mitsumasa Koyanagi	720
<i>Homeostasis in a Silicon Integrate and Fire Neuron,</i> Shih-Chii Liu and Bradley A. Minch	727
<i>Fast Training of Support Vector Classifiers,</i> Fernando Pérez-Cruz, Pedro L. Alarcón-Diana, Angel Navia-Vázquez and Antonio Artés-Rodríguez	734
<i>Four-legged Walking Gait Control Using a Neuromorphic Chip Interfaced to a</i> <i>Support Vector Learning Algorithm,</i> Susanne Still, Bernhard Schölkopf, Klaus Hepp and Rodney J. Douglas	741

Part VI Speech, Handwriting and Signal Processing

<i>New Approaches Towards Robust and Adaptive Speech Recognition</i> (invited paper), Hervé Bourlard, Samy Bengio and Katrin Weber	751
<i>Speech Denoising and Dereverberation Using Probabilistic Models,</i> Hagai Attias, John C. Platt, Alex Acero and Li Deng	758
<i>Combining ICA and Top-Down Attention for Robust Speech Recognition,</i> Un-Min Bae and Soo-Young Lee	765
<i>Learning Joint Statistical Models for Audio-Visual Fusion and Segregation,</i> John W. Fisher III, Trevor Darrell, William T. Freeman and Paul Viola	772
<i>Factored Semi-Tied Covariance Matrices,</i> Mark J.F. Gales	779
<i>Higher-Order Statistical Properties Arising from the Non-Stationarity of Natural</i> <i>Signals,</i> Lucas Parra, Clay Spence and Paul Sajda	786
<i>One Microphone Source Separation,</i> Sam T. Roweis	793
<i>Minimum Bayes Error Feature Selection for Continuous Speech Recognition,</i> George Saon and Mukund Padmanabhan	800
<i>Periodic Component Analysis: An Eigenvalue Method for Representing Periodic</i> <i>Structure in Speech,</i> Lawrence K. Saul and Jont B. Allen	807
<i>FaceSync: A Linear Operator for Measuring Synchronization of Video Facial</i> <i>Images and Audio Tracks,</i> Malcolm Slaney and Michele Covell	814

<i>Noise Suppression Based on Neurophysiologically-motivated SNR Estimation for Robust Speech Recognition,</i> Jürgen Tchorz, Michael Kleinschmidt and Birger Kollmeier	821
--	-----

Part VII Visual Processing

<i>Shape Context: A New Descriptor for Shape Matching and Object Recognition,</i> Serge Belongie, Jitendra Malik and Jan Puzicha	831
<i>Emergence of Movement Sensitive Neurons' Properties by Learning a Sparse Code for Natural Moving Images,</i> Rafal Bogacz, Malcolm W. Brown and Christophe Giraud-Carrier	838
<i>The Manhattan World Assumption: Regularities in Scene Statistics which Enable Bayesian Inference,</i> James M. Coughlan and Allen L. Yuille	845
<i>Feature Correspondence: A Markov Chain Monte Carlo Approach,</i> Frank Dellaert, Steven M. Seitz, Sebastian Thrun and Charles Thorpe	852
<i>Keeping Flexible Active Contours on Track using Metropolis Updates,</i> Trausti T. Kristjansson and Brendan J. Frey	859
<i>Color Opponency Constitutes a Sparse Representation for the Chromatic Structure of Natural Scenes,</i> Te-Won Lee, Thomas Wachtler and Terrence J. Sejnowski	866
<i>Learning Segmentation by Random Walks,</i> Marina Meilă and Jianbo Shi	873
<i>Partially Observable SDE Models for Image Sequence Recognition Tasks,</i> Javier R. Movellan, Paul Mineiro and Ruth J. Williams	880
<i>Learning Sparse Image Codes using a Wavelet Pyramid Architecture,</i> Bruno A. Olshausen, Phil Sallee and Michael S. Lewicki	887
<i>Learning and Tracking Cyclic Human Motion,</i> Dirk Ormoneit, Hedvig Sidenbladh, Michael J. Black and Trevor Hastie	894
<i>Redundancy and Dimensionality Reduction in Sparse-Distributed Representations of Natural Objects in Terms of Their Local Features,</i> Penio S. Penev	901
<i>Rate-coded Restricted Boltzmann Machines for Face Recognition,</i> Yee Whye Teh and Geoffrey E. Hinton	908
<i>Divisive and Subtractive Mask Effects: Linking Psychophysics and Biophysics,</i> Barbara Zenger and Christof Koch	915

Part VIII Applications

<i>From Mixtures of Mixtures to Adaptive Transform Coding,</i> Cynthia Archer and Todd K. Leen	925
<i>A Neural Probabilistic Language Model,</i> Yoshua Bengio, Réjean Ducharme and Pascal Vincent	932

<i>A Comparison of Image Processing Techniques for Visual Speech Recognition Applications</i> , Michael S. Gray, Terrence J. Sejnowski and Javier R. Movellan . . .	939
<i>Support Vector Novelty Detection Applied to Jet Engine Vibration Spectra</i> , Paul Hayton, Bernhard Schölkopf, Lionel Tarassenko and Paul Anuzis	946
<i>Recognizing Hand-written Digits Using Hierarchical Products of Experts</i> , Guy Mayraz and Geoffrey E. Hinton	953
<i>Sex with Support Vector Machines</i> , Baback Moghaddam and Ming-Hsuan Yang . .	960
<i>Probabilistic Semantic Video Indexing</i> , Milind R. Naphade, Igor Kozintsev and Thomas Huang	967
<i>Interactive Parts Model: An Application to Recognition of On-line Cursive Script</i> , Predrag Neskovic, Philip C. Davis and Leon N. Cooper	974
<i>Learning Switching Linear Models of Human Motion</i> , Vladimir Pavlović, James M. Rehg and John MacCormick	981
<i>Bayes Networks on Ice: Robotic Search for Antarctic Meteorites</i> , Liam Pedersen, Dimi Apostolopoulos and Red Whittaker	988
<i>The Use of Classifiers in Sequential Inference</i> , Vasin Punyakanok and Dan Roth . .	995
<i>Machine Learning for Video-Based Rendering</i> , Arno Schödl and Irfan Essa	1002
<i>Bayesian Video Shot Segmentation</i> , Nuno Vasconcelos and Andrew Lippman . . .	1009

Part IX Control, Navigation and Planning

<i>Programmable Reinforcement Learning Agents</i> , David Andre and Stuart J. Russell .	1019
<i>Exact Solutions to Time-Dependent MDPs</i> , Justin A. Boyan and Michael L. Littman	1026
<i>Decomposition of Reinforcement Learning for Admission Control of Self-Similar Call Arrival Processes</i> , Jakob Carlström	1033
<i>Reinforcement Learning with Function Approximation Converges to a Region</i> , Geoffrey J. Gordon	1040
<i>Hierarchical Memory-Based Reinforcement Learning</i> , Natalia Hernandez-Gardiol and Sridhar Mahadevan	1047
<i>Automated State Abstraction for Options using the U-Tree Algorithm</i> , Anders Jonsson and Andrew G. Barto	1054
<i>Robust Reinforcement Learning</i> , Jun Morimoto and Kenji Doya	1061
<i>Kernel-Based Reinforcement Learning in Average-Cost Problems: An Application to Optimal Portfolio Choice</i> , Dirk Ormoneit and Peter Glynn	1068
<i>Using Free Energies to Represent Q-values in a Multiagent Reinforcement Learning Task</i> , Brian Sallans and Geoffrey E. Hinton	1075
<i>Balancing Multiple Sources of Reward in Reinforcement Learning</i> , Christian R. Shelton	1082

APRICODD: Approximate Policy Construction Using Decision Diagrams, Robert St-Aubin, Jesse Hoey and Craig Boutilier	1089
Index of Authors	1097
Keyword Index	1101

The 1997 Systems II (SI) conference, held in Colorado from November 22 through December 2, 1997, was a landmark event in the history of AI. It was the first time that a conference in the field of AI had a theme, "Systems II: The Next Generation of AI Systems". The conference was a celebration of the progress made in the field of AI over the past decade, and a look ahead to the future. The conference was organized by a committee of leading researchers in the field, and was held in a beautiful location in the mountains of Colorado. The conference was a success, and it was a pleasure to be part of it.

The conference was a landmark event in the history of AI. It was the first time that a conference in the field of AI had a theme, "Systems II: The Next Generation of AI Systems". The conference was a celebration of the progress made in the field of AI over the past decade, and a look ahead to the future. The conference was organized by a committee of leading researchers in the field, and was held in a beautiful location in the mountains of Colorado. The conference was a success, and it was a pleasure to be part of it.

The conference was a landmark event in the history of AI. It was the first time that a conference in the field of AI had a theme, "Systems II: The Next Generation of AI Systems". The conference was a celebration of the progress made in the field of AI over the past decade, and a look ahead to the future. The conference was organized by a committee of leading researchers in the field, and was held in a beautiful location in the mountains of Colorado. The conference was a success, and it was a pleasure to be part of it.

The conference was a landmark event in the history of AI. It was the first time that a conference in the field of AI had a theme, "Systems II: The Next Generation of AI Systems". The conference was a celebration of the progress made in the field of AI over the past decade, and a look ahead to the future. The conference was organized by a committee of leading researchers in the field, and was held in a beautiful location in the mountains of Colorado. The conference was a success, and it was a pleasure to be part of it.

The conference was a landmark event in the history of AI. It was the first time that a conference in the field of AI had a theme, "Systems II: The Next Generation of AI Systems". The conference was a celebration of the progress made in the field of AI over the past decade, and a look ahead to the future. The conference was organized by a committee of leading researchers in the field, and was held in a beautiful location in the mountains of Colorado. The conference was a success, and it was a pleasure to be part of it.

Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping

Rich Caruana
CALD, CMU
5000 Forbes Ave.
Pittsburgh, PA 15213
caruana@cs.cmu.edu

Steve Lawrence
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
lawrence@research.nj.nec.com

Lee Giles
Information Sciences
Penn State University
University Park, PA 16801
giles@ist.psu.edu

Abstract

The conventional wisdom is that backprop nets with excess hidden units generalize poorly. We show that nets with excess capacity generalize well when trained with backprop and early stopping. Experiments suggest two reasons for this: 1) Overfitting can vary significantly in different regions of the model. Excess capacity allows better fit to regions of high non-linearity, and backprop often avoids overfitting the regions of low non-linearity. 2) Regardless of size, nets learn task subcomponents in similar sequence. Big nets pass through stages similar to those learned by smaller nets. Early stopping can stop training the large net when it generalizes comparably to a smaller net. We also show that conjugate gradient can yield worse generalization because it overfits regions of low non-linearity when learning to fit regions of high non-linearity.

1 Introduction

It is commonly believed that large multi-layer perceptrons (MLPs) generalize poorly: nets with too much capacity overfit the training data. Restricting net capacity prevents overfitting because the net has insufficient capacity to learn models that are too complex. This belief is consistent with a VC-dimension analysis of net capacity vs. generalization: the more free parameters in the net the larger the VC-dimension of the hypothesis space, and the less likely the training sample is large enough to select a (nearly) correct hypothesis [2].

Once it became feasible to train large nets on real problems, a number of MLP users noted that the overfitting they expected from nets with excess capacity did not occur. Large nets appeared to generalize as well as smaller nets — sometimes better. The earliest report of this that we are aware of is Martin and Pittman in 1991: “*We find only marginal and inconsistent indications that constraining net capacity improves generalization*” [7].

We present empirical results showing that MLPs with excess capacity often do not overfit. On the contrary, we observe that large nets often generalize better than small nets of sufficient capacity. Backprop appears to use excess capacity to better fit regions of high non-linearity, while still fitting regions of low non-linearity smoothly. (This desirable behavior can disappear if a fast training algorithm such as conjugate gradient is used instead of backprop.) Nets with excess capacity trained with backprop appear first to learn models similar to models learned by smaller nets. If early stopping is used, training of the large net can be halted when the large net’s model is similar to models learned by smaller nets.

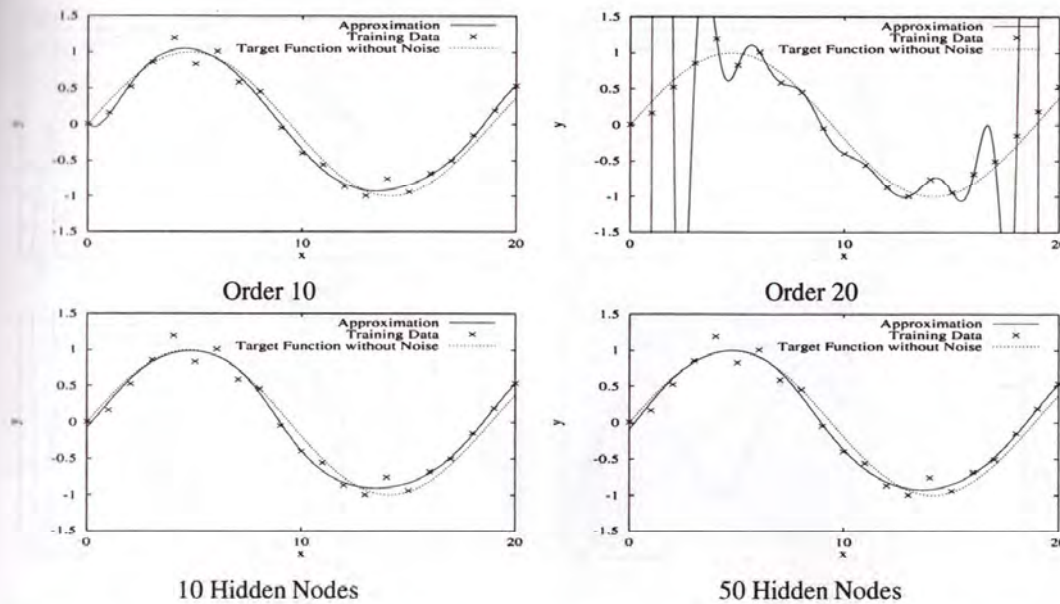


Figure 1: Top: Polynomial fit to data from $y = \sin(x/3) + \nu$. Order 20 overfits. Bottom: Small and large MLPs fit to same data. The large MLP does not overfit significantly more than the small MLP.

2 Overfitting

Much has been written about overfitting and the bias/variance tradeoff in neural nets and other machine learning models [2, 12, 4, 8, 5, 13, 6]. The top of Figure 1 illustrates polynomial overfitting. We created a training dataset by evaluating $y = \sin(x/3) + \nu$ at $0, 1, 2, \dots, 20$ where ν is a uniformly distributed random variable between -0.25 and 0.25 . We fit polynomial models with orders 2-20 to the data. Underfitting occurs with order 2. The fit is good with order 10. As the order (and number of parameters) increases, however, significant overfitting (poor generalization) occurs. At order 20, the polynomial fits the training data well, but interpolates poorly.

The bottom of Figure 1 shows MLPs fit to the data. We used a single hidden layer MLP, backpropagation (BP), and 100,000 stochastic updates. The learning rate was reduced linearly to zero from an initial rate of 0.5 (reducing the learning rate improves convergence, and linear reduction performs similarly to other schedules [3]). This schedule and number of updates trains the MLPs to completion. (We examine early stopping in Section 4.) As with polynomials, the smallest net with one hidden unit (HU) (4 weights) underfits the data. The fit is good with two HU (7 weights). Unlike polynomials, however, networks with 10 HU (31 weights) and 50 HU (151 weights) also yield good models. MLPs with seven times as many parameters as data points trained with BP do not significantly overfit this data. The experiments in Section 4 confirm that this bias of BP-trained MLPs towards smooth models is not limited to the simple 2-D problem used here.

3 Local Overfitting

Regularization methods such as weight decay typically assume that overfitting is a global phenomenon. But overfitting can vary significantly in different regions of a model. Figure 2 shows polynomial fits for data generated from the following equation:

$$y = \begin{cases} -\cos(x) + \nu & 0 \leq x < \pi \\ \cos(3(x - \pi)) + \nu & \pi \leq x \leq 2\pi \end{cases} \quad (\text{Equation 1})$$

Five equally spaced points were generated in the first region, and 15 in the second region, so that the two regions have different data densities and different underlying functions. Overfitting is different in the two regions. In Figure 2 the order 6 model fits the left region

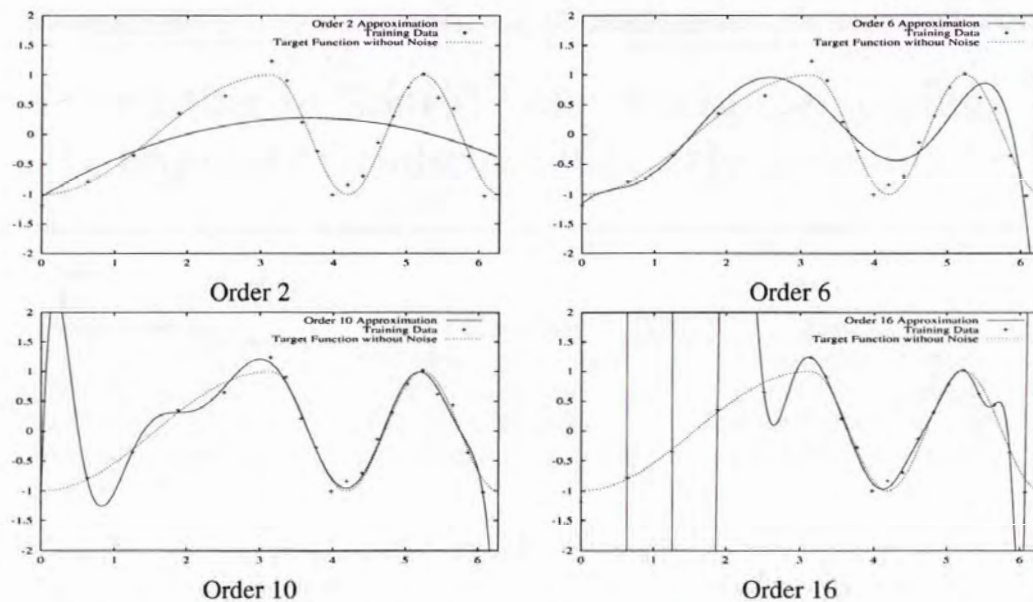


Figure 2: Polynomial approximation of data from Equation 1 as the order of the model is increased from 2 to 16. The overfitting behavior differs in the left and right hand regions.

well, but larger models overfit it. The order 6 model underfits the region on the right, and the order 10 model fits it better. No model performs well on both regions. Figure 3 shows MLPs trained on the same data (20,000 batch updates, learning rate linearly reduced to zero starting at 0.5). Small nets underfit. Larger nets, however, fit the entire function well *without significant overfitting in the left region*.

The ability of MLPs to fit both regions of low and high non-linearity well (without overfitting) depends on the training algorithm. Conjugate gradient (CG) is the most popular second order method. CG results in lower *training* error for this problem, but overfits significantly. Figure 4 shows results for 10 trials for BP and CG. Large BP nets generalize better on this problem — even the optimal size CG net is prone to overfitting. The degree of overfitting varies in different regions. When the net is large enough to fit the region of high non-linearity, overfitting is often seen in the region of low non-linearity.

4 Generalization, Network Capacity, and Early Stopping

The results in Sections 2 and 3 suggest that BP nets are less prone to overfitting than expected. But MLPs can and do overfit. This section examines overfitting vs. net size on seven problems: NETtalk [10], 7 and 12 bit parity, an inverse kinematic model for a robot arm (thanks to Sebastian Thrun for the simulator), Base 1 and Base 2: two sonar modeling problems using data collected from a robot wondering hallways at CMU, and vision data used to learn to steer an autonomous car [9]. These problems exhibit a variety of characteristics. Some are Boolean. Others are continuous. Some have noise. Others are noise-free. Some have many inputs or outputs. Others have few inputs or outputs.

4.1 Results

For each problem we used small training sets (100–1000 points, depending on the problem) so that overfitting was possible. We trained fully connected feedforward MLPs with one hidden layer whose size varied from 2 to 800 HU (about 500–100,000 parameters). All the nets were trained with BP using stochastic updates, learning rate 0.1, and momentum 0.9.

We used early stopping for regularization because it doesn't interfere with backprop's ability to control capacity locally. Early stopping combined with backprop is so effective that very large nets can be trained without significant overfitting. Section 4.2 explains why.

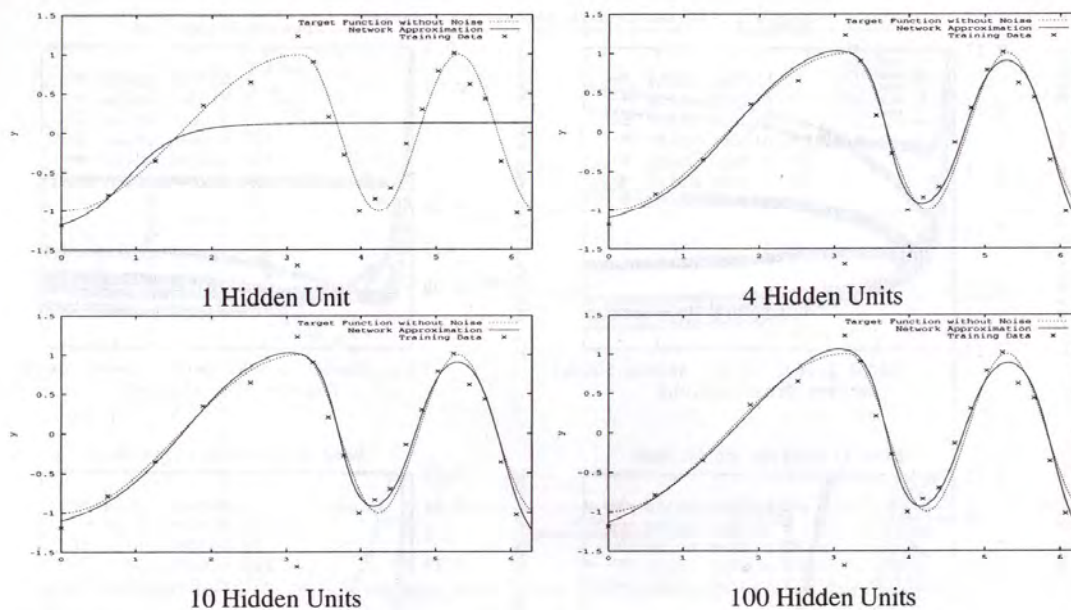


Figure 3: MLP approximation using backpropagation (BP) training of data from Equation 1 as the number of hidden units is increased. No significant overfitting can be seen.

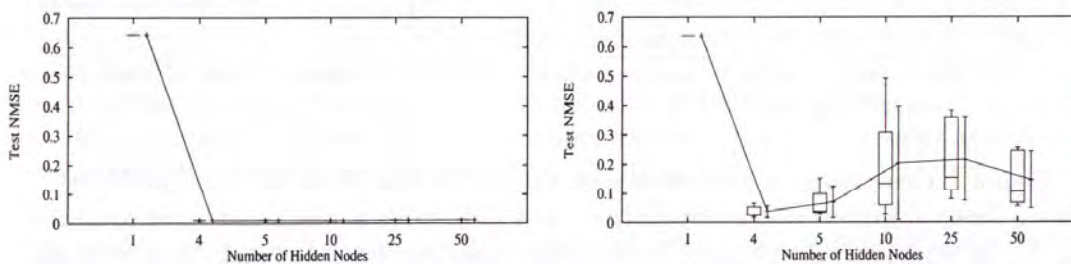


Figure 4: Test Normalized Mean Squared Error for MLPs trained with BP (left) and CG (right). Results are shown with both box-whiskers plots and the mean plus and minus one standard deviation.

Figure 5 shows generalization curves for four of the problems. Examining the results for all seven problems, we observe that on only three (Base 1, Base 2, and ALVINN), do nets that are too large yield worse generalization than smaller networks, but the loss is surprisingly small. Many trials were required before statistical tests confirmed that the differences between the optimal size net and the largest net were significant. Moreover, the results suggest that generalization is hurt more by using a net that is a little too small than by using one that is far too large, i.e., it is better to make nets too large than too small.

For most tasks and net sizes, we trained well beyond the point where generalization performance peaked. Because we had complete generalization curves, we noticed something unexpected. On some tasks, small nets overtrained considerably. The NETtalk graph in Figure 5 is a good example. Regularization (e.g., early stopping) is critical for nets of all sizes — not just ones that are *too big*. Nets with restricted capacity can overtrain.

4.2 Why Excess Capacity Does Not Hurt

BP nets initialized with small weights can develop large weights only after the number of updates is large. Thus BP nets consider hypotheses with small weights before hypotheses with large weights. Nets with large weights have more representational power, so simple hypotheses are explored before complex hypotheses.

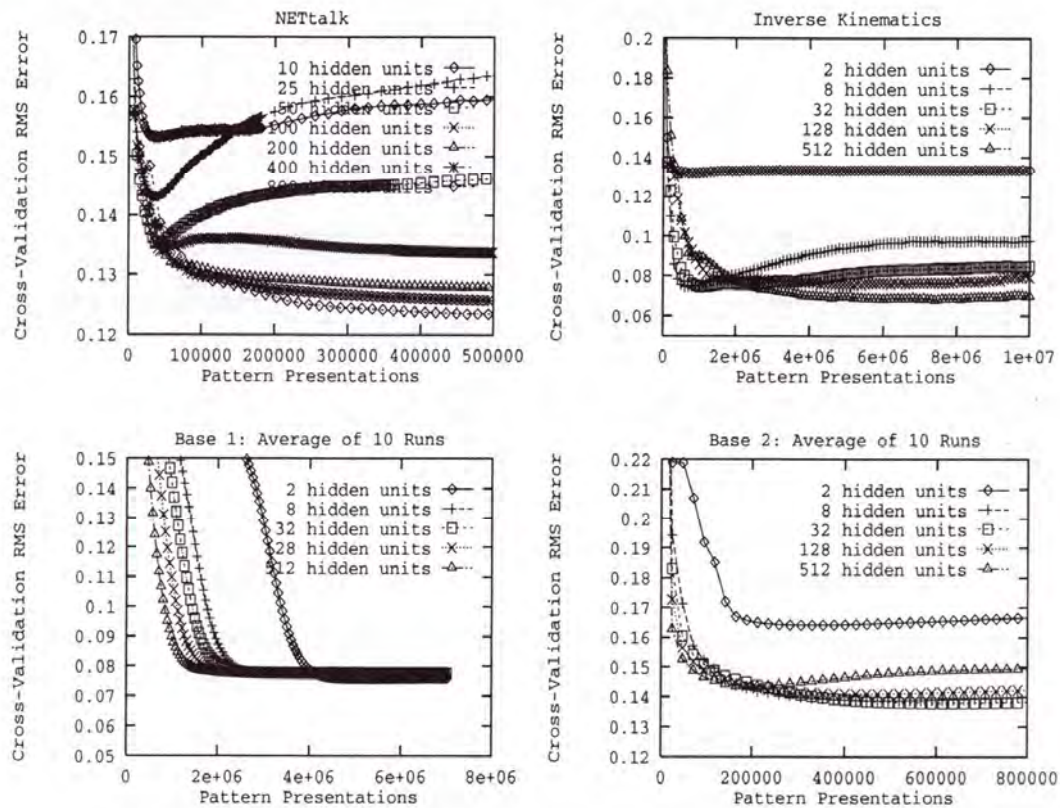


Figure 5: Generalization performance vs. net size for four of the seven test problems.

We analyzed *what* nets of different size learn *while* they are trained. We compared the input/output behavior of nets at different stages of learning on large samples of test patterns. We compare the input/output behavior of two nets by computing the squared error between the predictions made by the two nets. If two nets make the same predictions for all test cases, they have learned the same model (even though each model is represented differently), and the squared error between the two models is zero. If two nets make different predictions for test cases, they have learned different models, and the squared error between them is large. This is not the error the models make predicting the true labels, but the difference between predictions made by two different models. Two models can have poor generalization (large error on true labels), but have near zero error compared to each other if they are similar models. But two models with good generalization (low error on true labels) must have low error compared to each other.

The first graph in Figure 5 shows learning curves for nets with 10, 25, 50, 100, 200, and 400 HU trained on NETtalk. For each size, we saved the net from the epoch that generalized best on a large test set. This gives us the best model of each size found by backprop. We then trained a BP net with 800 HU, and after each epoch compared this net's model with the best models saved for nets of 10–400 HU. This lets us compare the sequence of models learned by the 800 HU net to the best models learned by smaller nets.

Figure 6 shows this comparison. The horizontal axis is the number of backprop passes applied to the 800 HU net. The vertical axis is the error between the 800 HU net model and the best model for each smaller net. The 800 HU net starts off distant from the good smaller models, then becomes similar to the good models, and then diverges from them. This is expected. What is interesting is that the 800 HU net first becomes closest to the best

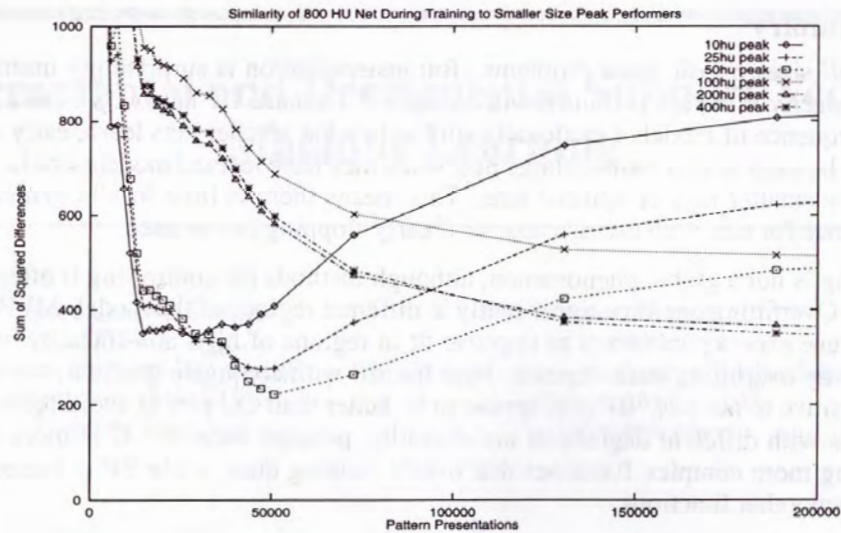


Figure 6: I/O similarity during training between an 800 hidden unit net and smaller nets (10, 25, 50, 100, 200, and 400 hidden units) trained on NETtalk.

10 HU net, then closest to the 25 HU net, then closest to the 50 HU net, etc. As it is trained, the 800 HU net learns a sequence of models similar to the models learned by smaller nets. If early stopping is used, training of the 800 HU net can be stopped when it behaves similar to the best model that could be learned with nets of 10, 25, 50, . . . HU. Large BP nets learn models similar to those learned by smaller nets. *If a BP net with too much capacity would overfit, early stopping could stop training when the model was similar to a model that would have been learned by a smaller net of optimal size.*

The error between models is about 200–400, yet the *generalization* error is about 1600. The models are much closer to each other than any of them are to the true model. With early stopping, what counts is the closest approach of each model to the target function, not where models end up late in training. With early stopping there is little disadvantage to using models that are too large because their learning trajectories are similar to those followed by smaller nets of more optimal size.

5 Related Work

Our results show that models learned by backprop are biased towards “smooth” solutions. As nets with excess capacity are trained, they first explore smoother models similar to the models smaller nets would have learned. Weigend [11] performed an experiment that showed BP nets learn a problem’s eigenvectors in sequence, learning the 1st eigenvector first, then the 2nd, etc. His result complements our analysis of what nets of different sizes learn: if large nets learn an eigenvector sequence similar to smaller nets, then the models learned by the large net will pass through intermediate stages similar to what is learned by small nets (but iff nets of different sizes learn the eigenvectors equally well, which is an assumption we do not need to make.)

Theoretical work by [1] supports our results. Bartlett notes: “the VC-bounds seem loose; neural nets often perform successfully with training sets that are considerably smaller than the number of weights.” Bartlett shows (for classification) that the number of training samples only needs to grow according to A^{2l} (ignoring log factors) to avoid overfitting, where A is a bound on the total weight magnitudes and l is the number of layers in the network. This result suggests that a net with smaller weights will generalize better than a similar net with large weights. Examining the weights from BP and CG nets shows that BP training typically results in smaller weights.

6 Summary

Nets of *all* sizes overfit some problems. But generalization is surprisingly insensitive to excess capacity if the net is trained with backprop. Because BP nets with excess capacity learn a sequence of models functionally similar to what smaller nets learn, early stopping can often be used to stop training large nets when they have learned models similar to those learned by smaller nets of optimal size. This means there is little loss in generalization performance for nets with excess capacity if early stopping can be used.

Overfitting is not a global phenomenon, although methods for controlling it often assume that it is. Overfitting can vary significantly in different regions of the model. MLPs trained with BP use excess parameters to improve fit in regions of high non-linearity, while not significantly overfitting other regions. Nets trained with conjugate gradient, however, are more sensitive to net size. BP nets appear to be better than CG nets at avoiding overfitting in regions with different degrees of non-linearity, perhaps because CG is more effective at learning more complex functions that overfit training data, while BP is biased toward learning smoother functions.

References

- [1] Peter L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In *Advances in Neural Information Processing Systems*, volume 9, page 134. The MIT Press, 1997.
- [2] E.B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.
- [3] C. Darken and J.E. Moody. Note on learning rate schedules for stochastic optimization. In *Advances in Neural Information Processing Systems*, volume 3, pages 832–838. Morgan Kaufmann, 1991.
- [4] S. Geman et al. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
- [5] A. Krogh and J.A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4, pages 950–957. Morgan Kaufmann, 1992.
- [6] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal Brain Damage. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605, San Mateo, 1990. (Denver 1989), Morgan Kaufmann.
- [7] G.L. Martin and J.A. Pittman. Recognizing hand-printed letters and digits using backpropagation learning. *Neural Computation*, 3:258–267, 1991.
- [8] J.E. Moody. The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems. In *Advances in Neural Information Processing Systems*, volume 4, pages 847–854. Morgan Kaufmann, 1992.
- [9] D.A. Pomerleau. Alvin: An autonomous land vehicle in a neural network. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 305–313, San Mateo, 1989. (Denver 1988), Morgan Kaufmann.
- [10] T. Sejnowski and C. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168, 1987.
- [11] A. Weigend. On overfitting and the effective number of hidden units. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 335–342. Lawrence Erlbaum Associates, 1993.
- [12] A.S. Weigend, D.E. Rumelhart, and B.A. Huberman. Generalization by weight-elimination with application to forecasting. In *Advances in Neural Information Processing Systems*, volume 3, pages 875–882. Morgan Kaufmann, 1991.
- [13] D. Wolpert. On bias plus variance. *Neural Computation*, 9(6):1211–1243, 1997.

APPENDIX XU01



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)

Share

Bookmark

Report an Issue

MARC View

LEADER 02532nas a22005417a 4500

001 50272593

003 SIRSI

005 20260214090101.0

006 m o d

007 cr n

008 000713d19902011nyubr p s 0 a0eng c

010 a| 2008212909

016 7 a| 012433305 2| Uk

019 a| DO NOT EDIT this 360 MARC RECORD

022 a| 1941-0093 y| 1045-9227

035 a| (WaSeSS)ssj0014506

037 b| IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331

040 a| NTD b| eng c| NTD d| OCLCQ d| VRC d| OCLCQ d| NSD d| U95 d| OCLCQ d| DLC d| OCLCQ d| OCLCF d| VT2 d| OCLCQ d| DEFHM d| OCLCO d| OCLCA d| BUF d| EZC d| OCLCO d| U3W d| OCLCA d| AU@ d| OCLCO d| WYU d| OCLCO d| TKN d| OCLCO d| OCLCA d| UKMGB d| OCLCQ d| SXB d| OCLCL d| OCLCA d| OCLCL d| OCLCQ d| WaSeSS

042 a| pcc a| nsdp

050 1 4 a| TK7800 b| .I5 NN

060 0 0 a| W1 b| IE44625

082 1 0 a| 006 2| 14 1| <https://id.oclc.org/worldcat/ddc/E379xBCBchbdmrqYKbR6BJFPFX>

130 0 a| IEEE transactions on neural networks (Online)

210 0 a| IEEE trans. neural netw. b| (Online)

222 0 a| IEEE transactions on neural networks b| (Online)

245 1 0 a| IEEE transactions on neural networks h| [electronic resource] / c| a publication of the IEEE Neural Networks Council.

246 1 a| Transactions on neural networks

246 1 a| Neural networks

260 a| New York, NY : b| Institute of Electrical and Electronics Engineers, c| ©:1990-

310 a| Bimonthly, b| 1991-

321 a| Quarterly, b| 1990-

362 0 a| Vol. 1, no. 1 (Mar. 1990)-

362 1 a| Ceased in 2011.

500 a| Title from IEEEExplore homepage (viewed July 13, 2000).

538 a| Mode of access: World Wide Web.

550 a| Issued by: IEEE Neural Networks Council.

650 0 a| Neural networks (Computer science) v| Periodicals.

650 2 a| Nervous System

650 2 a| Neural Networks (Computer)

655 0 a| Electronic journals.

655 2 a| Periodical.

710 2 a| IEEE Neural Networks Council. 1| <https://id.oclc.org/worldcat/entity/E39QQPVP7TtCPXxp4JvWg86pqF>.

776 0 8 i| Print version: t| IEEE transactions on neural networks x| 1045-9227 w| (DLC) 90640964 w| (OCLC)20284540

785 0 0 t| IEEE transaction on neural networks and learning systems x| 2162-2388 w| (DLC) 2011200426 w| (OCLC)738499288

856 4 0 u| <http://SK8ES4MC2L.search.serialssolutions.com/?V=1.0&L=SK8ES4MC2L&S=JCS&C=IEEETRANONN&T=marc>

910 a| CONSER online

023 0 a| 1045-9227 2| 1

949 a| Electronic resource w| ASIS c| 1 i| 50272593-1001 i| ONLINE m| ONLINE r| Y s| Y t| ONLINE u| 2/28/2026 o| .STAFF. 360 MARC

ASK
A LIBRARIAN

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)**PENN STATE
UNIVERSITY LIBRARIES**

[Libraries Home](#) (814) 865-6368
[Libraries Intranet \(Staff Only\)](#)
[Accessibility Help](#)
[Website Feedback](#)
[Policies and Guidelines](#)
[Staff Directory](#)

**CONNECT WITH
PENN STATE UNIVERSITY
LIBRARIES**

[Facebook](#)
[Twitter](#)
[Instagram](#)



Keyword Search...

Search

Advanced search Start Over

Share

Bookmark

Report an Issue



IEEE transactions on neural networks [electronic resource] / a publication of the IEEE Neural Networks Council

Uniform Title	IEEE transactions on neural networks (Online)
Additional Titles	IEEE trans. neural netw. (Online), Transactions on neural networks, and Neural networks
Published	New York, NY : Institute of Electrical and Electronics Engineers, ©1990-
Additional Creators	IEEE Neural Networks Council

Access Online

serialssolutions.com

Continued By

[IEEE transaction on neural networks and learning systems](#)

Dates of Publication and/or Sequential Designation

Vol. 1, no. 1 (Mar. 1990)-
Ceased in 2011.

Subject(s)

[Neural networks \(Computer science\)—Periodicals](#)
[Nervous System](#)
[Neural Networks \(Computer\)](#)

Genre(s)

[Electronic journals](#)

ISSN

1941-0093

Grouping ISSN

1045-9227

Publication Frequency

Bimonthly, 1991-
Quarterly, 1990-

Note

Title from IEEEExplore homepage (viewed July 13, 2000).

Technical Details

Mode of access: World Wide Web.

Issuing Body

Issued by: IEEE Neural Networks Council.

ASK
A LIBRARIAN[View MARC record](#) | catkey: 50272593

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#) (814) 865-6368
[Libraries Intranet \(Staff Only\)](#)
[Accessibility Help](#)
[Website Feedback](#)
[Policies and Guidelines](#)
[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#)
[Twitter](#)
[Instagram](#)



PennState
University Libraries

RENEW MATERIALS

Give Today

My Account



Services Research About Ask

🕒 Pattee and Paterno: View Hours

Online Journals, Books and more

👉 All ☒ Journals Only ☐ Books Only ☐ Videos Only

Title begins with



Search

[Show more search options](#) ▼

Showing results 1 through 1 of 1

for the search: "IEEETRANONN"

Refine Results: [All](#) | [Journals Only](#) | [Books Only](#) | [Videos Only](#)

Limit by: [Peer Reviewed](#) | [Open Access](#)

IEEE transactions on neural networks

ISSN: 1045-9227

1990 to 2011 in [IEEE Electronic Library \(IEL\)](#)

Search inside this journal

Search

[Back to Top](#)



PennState
University Libraries



SERVICES

Borrow & Renew

Course Reserves

Rooms & Spaces

Technology & Equipment

Adaptive Technology & Services

Interlibrary Loan

Teaching Support for Instructors

RESEARCH

Try These First

Citation Guides

ScholarSphere

Collections

Electronic Theses and Dissertations

Research Data Services

ABOUT

Hours

Directory

Libraries and Departments

Jobs

Message from the Dean

Diversity

News and Events

Your Donation at Work

Libraries Calendar

ASK A LIBRARIAN

By Chat

By Phone

In Person

By Email

Schedule a Research Appt.

FAQs (or How Do I?...)

Tutorials

The Search Bar

Thank a Library Worker

[Libraries Home](#) | [Libraries Intranet \(Employees Only\)](#) | [Accessibility Help](#) | [Website Feedback](#) | [Policies and Guidelines](#) | [Acknowledgement of Land](#) | (814) 865-6368

[Hotline](#) | [Privacy Statements](#) | [Non Discrimination](#) | [Accessibility](#) | [Legal Statements](#)

Copyright ©2025 The Pennsylvania State University.

All rights reserved. Except where otherwise noted, this work is subject to a [Creative Commons Attribution 4.0 license](#). Details and exceptions.

Last Updated on 03-03-2026 © Ex Libris, a ProQuest Company, 2000-2026

APPENDIX XU02

Survey of Clustering Algorithms

Rui Xu, *Student Member, IEEE* and Donald Wunsch II, *Fellow, IEEE*

Abstract—Data analysis plays an indispensable role for understanding various phenomena. Cluster analysis, primitive exploration with little or no prior knowledge, consists of research developed across a wide variety of communities. The diversity, on one hand, equips us with many tools. On the other hand, the profusion of options causes confusion. We survey clustering algorithms for data sets appearing in statistics, computer science, and machine learning, and illustrate their applications in some benchmark data sets, the traveling salesman problem, and bioinformatics, a new field attracting intensive efforts. Several tightly related topics, proximity measure, and cluster validation, are also discussed.

Index Terms—Adaptive resonance theory (ART), clustering, clustering algorithm, cluster validation, neural networks, proximity, self-organizing feature map (SOFM).

I. INTRODUCTION

WE ARE living in a world full of data. Every day, people encounter a large amount of information and store or represent it as data, for further analysis and management. One of the vital means in dealing with these data is to classify or group them into a set of categories or clusters. Actually, as one of the most primitive activities of human beings [14], classification plays an important and indispensable role in the long history of human development. In order to learn a new object or understand a new phenomenon, people always try to seek the features that can describe it, and further compare it with other known objects or phenomena, based on the similarity or dissimilarity, generalized as proximity, according to some certain standards or rules. “Basically, classification systems are either supervised or unsupervised, depending on whether they assign new inputs to one of a finite number of discrete supervised classes or unsupervised categories, respectively [38], [60], [75]. In supervised classification, the mapping from a set of input data vectors ($\mathbf{x} \in \mathbb{R}^d$, where d is the input space dimensionality), to a finite set of discrete class labels ($y \in 1, \dots, C$, where C is the total number of class types), is modeled in terms of some mathematical function $y = y(\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ is a vector of adjustable parameters. The values of these parameters are determined (optimized) by an inductive learning algorithm (also termed inducer), whose aim is to minimize an empirical risk functional (related to an inductive principle) on a finite data set of input–output examples, $(\mathbf{x}_i, y_i), i = 1, \dots, N$, where N is the finite cardinality of the available representative data set [38],

[60], [167]. When the inducer reaches convergence or terminates, an induced classifier is generated [167].

In unsupervised classification, called clustering or exploratory data analysis, no labeled data are available [88], [150]. The goal of clustering is to separate a finite unlabeled data set into a finite and discrete set of “natural,” hidden data structures, rather than provide an accurate characterization of unobserved samples generated from the same probability distribution [23], [60]. This can make the task of clustering fall outside of the framework of unsupervised predictive learning problems, such as vector quantization [60] (see Section II-C), probability density function estimation [38] (see Section II-D), [60], and entropy maximization [99]. It is noteworthy that clustering differs from multidimensional scaling (perceptual maps), whose goal is to depict all the evaluated objects in a way that minimizes the topographical distortion while using as few dimensions as possible. Also note that, in practice, many (predictive) vector quantizers are also used for (nonpredictive) clustering analysis [60].

Nonpredictive clustering is a subjective process in nature, which precludes an absolute judgment as to the relative efficacy of all clustering techniques [23], [152]. As pointed out by Backer and Jain [17], “in cluster analysis a group of objects is split up into a number of more or less homogeneous subgroups on the basis of an often subjectively chosen measure of similarity (i.e., chosen subjectively based on its ability to create “interesting” clusters), such that the similarity between objects within a subgroup is larger than the similarity between objects belonging to different subgroups”¹.

Clustering algorithms partition data into a certain number of clusters (groups, subsets, or categories). There is no universally agreed upon definition [88]. Most researchers describe a cluster by considering the internal homogeneity and the external separation [111], [124], [150], i.e., patterns in the same cluster should be similar to each other, while patterns in different clusters should not. Both the similarity and the dissimilarity should be examinable in a clear and meaningful way. Here, we give some simple mathematical descriptions of several types of clustering, based on the descriptions in [124].

Given a set of input patterns $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_j, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_j = (x_{j1}, x_{j2}, \dots, x_{jd})^T \in \mathbb{R}^d$ and each measure x_{ji} is said to be a feature (attribute, dimension, or variable).

- (Hard) partitional clustering attempts to seek a K -partition of $\mathbf{X}, C = \{C_1, \dots, C_K\}$ ($K \leq N$), such that
 - 1) $C_i \neq \phi, i = 1, \dots, K$;
 - 2) $\bigcup_{i=1}^K C_i = \mathbf{X}$;
 - 3) $C_i \cap C_j = \phi, i, j = 1, \dots, K$ and $i \neq j$.

Manuscript received March 31, 2003; revised September 28, 2004. This work was supported in part by the National Science Foundation and in part by the M. K. Finley Missouri Endowment.

The authors are with the Department of Electrical and Computer Engineering, University of Missouri-Rolla, Rolla, MO 65409 USA (e-mail: rxu@umr.edu; dwunsch@ece.umar.edu).

Digital Object Identifier 10.1109/TNN.2005.845141

¹The preceding quote is taken verbatim from verbiage suggested by the anonymous associate editor, a suggestion which we gratefully acknowledge.

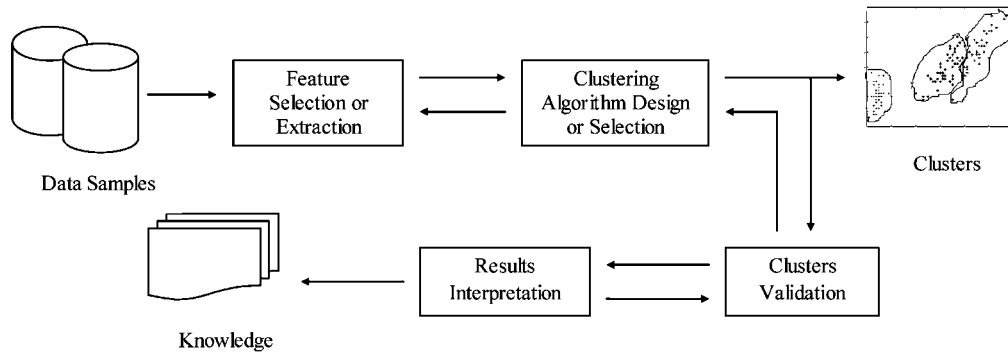


Fig. 1. Clustering procedure. The typical cluster analysis consists of four steps with a feedback pathway. These steps are closely related to each other and affect the derived clusters.

-) Hierarchical clustering attempts to construct a tree-like nested structure partition of $\mathbf{X}, H = \{H_1, \dots, H_Q\}$ ($Q \leq N$), such that $C_i \in H_m, C_j \in H_l$, and $m > l$ imply $C_i \in C_j$ or $C_i \cap C_j = \emptyset$ for all $i, j \neq i, m, l = 1, \dots, Q$.

For hard partitional clustering, each pattern only belongs to one cluster. However, a pattern may also be allowed to belong to all clusters with a degree of membership, $u_{i,j} \in [0, 1]$, which represents the membership coefficient of the j th object in the i th cluster and satisfies the following two constraints:

$$\sum_{i=1}^c u_{i,j} = 1, \quad \forall j \quad \text{and} \quad \sum_{j=1}^N u_{i,j} < N, \quad \forall i$$

as introduced in fuzzy set theory [293]. This is known as fuzzy clustering, reviewed in Section II-G.

Fig. 1 depicts the procedure of cluster analysis with four basic steps.

- 1) *Feature selection or extraction.* As pointed out by Jain *et al.* [151], [152] and Bishop [38], feature selection chooses distinguishing features from a set of candidates, while feature extraction utilizes some transformations to generate useful and novel features from the original ones. Both are very crucial to the effectiveness of clustering applications. Elegant selection of features can greatly decrease the workload and simplify the subsequent design process. Generally, ideal features should be of use in distinguishing patterns belonging to different clusters, immune to noise, easy to extract and interpret. We elaborate the discussion on feature extraction in Section II-L, in the context of data visualization and dimensionality reduction. More information on feature selection can be found in [38], [151], and [250].
- 2) *Clustering algorithm design or selection.* The step is usually combined with the selection of a corresponding proximity measure and the construction of a criterion function. Patterns are grouped according to whether they resemble each other. Obviously, the proximity measure directly affects the formation of the resulting clusters. Almost all clustering algorithms are explicitly or implicitly connected to some definition of proximity measure. Some algorithms even work directly on the proximity matrix, as defined in Section II-A. Once a proximity measure is chosen, the construction of a

clustering criterion function makes the partition of clusters an optimization problem, which is well defined mathematically, and has rich solutions in the literature. Clustering is ubiquitous, and a wealth of clustering algorithms has been developed to solve different problems in specific fields. However, there is no clustering algorithm that can be universally used to solve all problems. "It has been very difficult to develop a unified framework for reasoning about it (clustering) at a technical level, and profoundly diverse approaches to clustering" [166], as proved through an impossibility theorem. Therefore, it is important to carefully investigate the characteristics of the problem at hand, in order to select or design an appropriate clustering strategy.

- 3) *Cluster validation.* Given a data set, each clustering algorithm can always generate a division, no matter whether the structure exists or not. Moreover, different approaches usually lead to different clusters; and even for the same algorithm, parameter identification or the presentation order of input patterns may affect the final results. Therefore, effective evaluation standards and criteria are important to provide the users with a degree of confidence for the clustering results derived from the used algorithms. These assessments should be objective and have no preferences to any algorithm. Also, they should be useful for answering questions like how many clusters are hidden in the data, whether the clusters obtained are meaningful or just an artifact of the algorithms, or why we choose some algorithm instead of another. Generally, there are three categories of testing criteria: external indices, internal indices, and relative indices. These are defined on three types of clustering structures, known as partitional clustering, hierarchical clustering, and individual clusters [150]. Tests for the situation, where no clustering structure exists in the data, are also considered [110], but seldom used, since users are confident of the presence of clusters. External indices are based on some prespecified structure, which is the reflection of prior information on the data, and used as a standard to validate the clustering solutions. Internal tests are not dependent on external information (prior knowledge). On the contrary, they examine the clustering structure directly from the original data. Relative criteria place

the emphasis on the comparison of different clustering structures, in order to provide a reference, to decide which one may best reveal the characteristics of the objects. We will not survey the topic in depth and refer interested readers to [74], [110], and [150]. However, we will cover more details on how to determine the number of clusters in Section II-M. Some more recent discussion can be found in [22], [37], [121], [180], and [181]. Approaches for fuzzy clustering validity are reported in [71], [104], [123], and [220].

- 4) *Results interpretation.* The ultimate goal of clustering is to provide users with meaningful insights from the original data, so that they can effectively solve the problems encountered. Experts in the relevant fields interpret the data partition. Further analyzes, even experiments, may be required to guarantee the reliability of extracted knowledge.

Note that the flow chart also includes a feedback pathway. Cluster analysis is not a one-shot process. In many circumstances, it needs a series of trials and repetitions. Moreover, there are no universal and effective criteria to guide the selection of features and clustering schemes. Validation criteria provide some insights on the quality of clustering solutions. But even how to choose the appropriate criterion is still a problem requiring more efforts.

Clustering has been applied in a wide variety of fields, ranging from engineering (machine learning, artificial intelligence, pattern recognition, mechanical engineering, electrical engineering), computer sciences (web mining, spatial database analysis, textual document collection, image segmentation), life and medical sciences (genetics, biology, microbiology, paleontology, psychiatry, clinic, pathology), to earth sciences (geography, geology, remote sensing), social sciences (sociology, psychology, archeology, education), and economics (marketing, business) [88], [127]. Accordingly, clustering is also known as numerical taxonomy, learning without a teacher (or unsupervised learning), typological analysis and partition. The diversity reflects the important position of clustering in scientific research. On the other hand, it causes confusion, due to the differing terminologies and goals. Clustering algorithms developed to solve a particular problem, in a specialized field, usually make assumptions in favor of the application of interest. These biases inevitably affect performance in other problems that do not satisfy these premises. For example, the K -means algorithm is based on the Euclidean measure and, hence, tends to generate hyperspherical clusters. But if the real clusters are in other geometric forms, K -means may no longer be effective, and we need to resort to other schemes. This situation also holds true for mixture-model clustering, in which a model is fit to data in advance.

Clustering has a long history, with lineage dating back to Aristotle [124]. General references on clustering techniques include [14], [75], [77], [88], [111], [127], [150], [161], [259]. Important survey papers on clustering techniques also exist in the literature. Starting from a statistical pattern recognition viewpoint, Jain, Murty, and Flynn reviewed the clustering algorithms and other important issues related to cluster analysis [152], while Hansen and Jaumard described the clustering problems under a mathematical programming scheme [124]. Kolatch and He investigated appli-

cations of clustering algorithms for spatial database systems [171] and information retrieval [133], respectively. Berkhin further expanded the topic to the whole field of data mining [33]. Murtagh reported the advances in hierarchical clustering algorithms [210] and Baraldi surveyed several models for fuzzy and neural network clustering [24]. Some more survey papers can also be found in [25], [40], [74], [89], and [151]. In addition to the review papers, comparative research on clustering algorithms is also significant. Rauber, Paralic, and Pampalk presented empirical results for five typical clustering algorithms [231]. Wei, Lee, and Hsu placed the emphasis on the comparison of fast algorithms for large databases [280]. Scheunders compared several clustering techniques for color image quantization, with emphasis on computational time and the possibility of obtaining global optima [239]. Applications and evaluations of different clustering algorithms for the analysis of gene expression data from DNA microarray experiments were described in [153], [192], [246], and [271]. Experimental evaluation on document clustering techniques, based on hierarchical and K -means clustering algorithms, were summarized by Steinbach, Karypis, and Kumar [261].

In contrast to the above, the purpose of this paper is to provide a comprehensive and systematic description of the influential and important clustering algorithms rooted in statistics, computer science, and machine learning, with emphasis on new advances in recent years.

The remainder of the paper is organized as follows. In Section II, we review clustering algorithms, based on the natures of generated clusters and techniques and theories behind them. Furthermore, we discuss approaches for clustering sequential data, large data sets, data visualization, and high-dimensional data through dimension reduction. Two important issues on cluster analysis, including proximity measure and how to choose the number of clusters, are also summarized in the section. This is the longest section of the paper, so, for convenience, we give an outline of Section II in bullet form here:

II. Clustering Algorithms

- A. Distance and Similarity Measures
(See also Table I)
- B. Hierarchical
 - Agglomerative
 - Single linkage, complete linkage, group average linkage, median linkage, centroid linkage, Ward's method, balanced iterative reducing and clustering using hierarchies (BIRCH), clustering using representatives (CURE), robust clustering using links (ROCK) ...
 - Divisive
 - divisive analysis (DIANA), monothetic analysis (MONA) ...
- C. Squared Error-Based (Vector Quantization)
 - K -means, iterative self-organizing data analysis technique (ISODATA), genetic K -means algorithm (GKA), partitioning around medoids (PAM) ...
- D. pdf Estimation via Mixture Densities
 - Gaussian mixture density decomposition (GMDD), AutoClass ...
- E. Graph Theory-Based
 - Chameleon, Delaunay triangulation graph (DTG), highly connected subgraphs (HCS), clustering iden-

TABLE I
SIMILARITY AND DISSIMILARITY MEASURE FOR QUANTITATIVE FEATURES

Measures	Forms	Comments	Examples and Applications
Minkowski distance	$D_{ij} = \left(\sum_{l=1}^d x_{il} - x_{jl} ^n \right)^{1/n}$	Metric. Invariant to any translation and rotation only for $n=2$ (Euclidean distance). Features with large values and variances tend to dominate over other features.	Fuzzy c -means with measures based on Minkowski family [130].
Euclidean distance	$D_{ij} = \left(\sum_{l=1}^d x_{il} - x_{jl} ^{1/2} \right)^2$	The most commonly used metric. Special case of Minkowski metric at $n=2$. Tend to form hyperspherical clusters.	K -means algorithm [191]
City-block distance	$D_{ij} = \sum_{l=1}^d x_{il} - x_{jl} $	Special case of Minkowski metric at $n=1$. Tend to form hyperrectangular clusters.	Fuzzy ART [57]
Sup distance	$D_{ij} = \max_{1 \leq l \leq d} x_{il} - x_{jl} $	Special case of Minkowski metric at $n \rightarrow \infty$.	Fuzzy c -means with sup norm [39].
Mahalanobis distance	$D_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{S}^{-1} (\mathbf{x}_i - \mathbf{x}_j)$, where $\mathbf{S}$ is the within-group covariance matrix.	Invariant to any nonsingular linear transformation. $\mathbf{S}$ is calculated based on all objects. Tend to form hyperellipsoidal clusters. When features are not correlated, squared Mahalanobis distance is equivalent to squared Euclidean distance. May cause some computational burden.	Ellipsoidal ART [13], Hyperellipsoidal clustering algorithm [194].
Pearson correlation	$D_{ij} = (1 - r_{ij})/2$, where $r_{ij} = \frac{\sum_{l=1}^d (x_{il} - \bar{x}_i)(x_{jl} - \bar{x}_j)}{\sqrt{\sum_{l=1}^d (x_{il} - \bar{x}_i)^2 \sum_{l=1}^d (x_{jl} - \bar{x}_j)^2}}$	Not a metric. Derived from correlation coefficient. Unable to detect the magnitude of differences of two variables.	Widely used as the measure for analyzing gene expression data [80].
Point symmetry distance	$D_{ir} = \min_{\substack{j=1, \dots, N \\ \text{and } j \neq i}} \frac{\ (\mathbf{x}_i - \mathbf{x}_r) + (\mathbf{x}_j - \mathbf{x}_r)\ }{\ (\mathbf{x}_i - \mathbf{x}_r)\ + \ (\mathbf{x}_j - \mathbf{x}_r)\ }$	Not a metric. Compute the distance between an object $\mathbf{x}_i$ and a reference point $\mathbf{x}_r$. D_{ir} is minimized when a symmetric pattern exists.	SBKM (Symmetry-based K -means) [264].
Cosine similarity	$S_{ij} = \cos \alpha = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\ \mathbf{x}_i\ \ \mathbf{x}_j\ }$	Independent of vector length. Invariant to rotation, but not to linear transformations.	The most commonly used measure in document clustering [261].

tification via connectivity kernels (CLICK), cluster affinity search technique (CAST) ...

- F. Combinatorial Search Techniques-Based
 - Genetically guided algorithm (GGA), TS clustering, SA clustering ...
- G. Fuzzy
 - Fuzzy c -means (FCM), mountain method (MM), possibilistic c -means clustering algorithm (PCM), fuzzy c -shells (FCS) ...
- H. Neural Networks-Based
 - Learning vector quantization (LVQ), self-organizing feature map (SOFM), ART, simplified ART (SART), hyperellipsoidal clustering network (HEC), self-splitting competitive learning network (SPLL) ...
- I. Kernel-Based
 - Kernel K -means, support vector clustering (SVC) ...
- J. Sequential Data
 - Sequence Similarity
 - Indirect sequence clustering
 - Statistical sequence clustering
- K. Large-Scale Data Sets (See also Table II)
 - CLARA, CURE, CLARANS, BIRCH, DBSCAN, DENCLUE, WaveCluster, FC, ART ...
- L. Data visualization and High-dimensional Data
 - PCA, ICA, Projection pursuit, Isomap, LLE, CLIQUE, OptiGrid, ORCLUS ...
- M. How Many Clusters?

Applications in two benchmark data sets, the traveling salesman problem, and bioinformatics are illustrated in Section III. We conclude the paper in Section IV.

II. CLUSTERING ALGORITHMS

Different starting points and criteria usually lead to different taxonomies of clustering algorithms [33], [88], [124], [150], [152], [171]. A rough but widely agreed frame is to classify clustering techniques as hierarchical clustering and partitional clustering, based on the properties of clusters generated [88], [152]. Hierarchical clustering groups data objects with a sequence of partitions, either from singleton clusters to a cluster including all individuals or vice versa, while partitional clustering directly divides data objects into some prespecified number of clusters without the hierarchical structure. We follow this frame in surveying the clustering algorithms in the literature. Beginning with the discussion on proximity measure, which is the basis for most clustering algorithms, we focus on hierarchical clustering and classical partitional clustering algorithms in Section II-B–D. Starting from part E, we introduce and analyze clustering algorithms based on a wide variety of theories and techniques, including graph theory, combinatorial search techniques, fuzzy set theory, neural networks, and kernels techniques. Compared with graph theory and fuzzy set

TABLE II
COMPUTATIONAL COMPLEXITY OF CLUSTERING ALGORITHMS

Cluster algorithm	Complexity	Capability of tackling high dimensional data
K -means	$O(NKd)$ (time) $O(N + K)$ (space)	No
Fuzzy c -means	Near $O(N)$	No
Hierarchical clustering*	$O(N^2)$ (time) $O(N^2)$ (space)	No
CLARA	$O(K(40+K)^2 + K(N-K))$ (time)	No
CLARANS	Quadratic in total performance	No
BIRCH	$O(N)$ (time)	No
DBSCAN	$O(N \log N)$ (time)	No
CURE	$O(N_{sample}^2 \log N_{sample})$ (time) $O(N_{sample})$ (space)	Yes
WaveCluster	$O(N)$ (time)	No
DENCLUE	$O(N \log N)$ (time)	Yes
FC	$O(N)$ (time)	Yes
CLIQUE	Linear with the number of objects, Quadratic with the number of dimensions	Yes
OptiGrid	Between $O(Nd)$ and $O(Nd \log N)$	Yes
ORCLUS	$O(K_0^3 + K_0Nd + K_0^2d^3)$ (time) $O(K_0d^2)$ (space)	Yes

*Include single-linkage, complete-linkage, average-linkage, etc.

+Based on the heuristic for drawing a sample from the entire data set in [161].

theory, which had already been widely used in cluster analysis before the 1980s, the other techniques have been finding their applications in clustering just in the recent decades. In spite of the short history, much progress has been achieved. Note that these techniques can be used for both hierarchical and partition clustering. Considering the more frequent requirement of tackling sequential data sets, large-scale, and high-dimensional data sets in many current applications, we review clustering algorithms for them in the following three parts. We focus particular attention on clustering algorithms applied in bioinformatics. We offer more detailed discussion on how to identify appropriate number of clusters, which is particularly important in cluster validity, in the last part of the section.

A. Distance and Similarity Measures

It is natural to ask what kind of standards we should use to determine the closeness, or how to measure the distance (dissimilarity) or similarity between a pair of objects, an object and a cluster, or a pair of clusters. In the next section on hierarchical clustering, we will illustrate linkage metrics for measuring proximity between clusters. Usually, a prototype is used to represent a cluster so that it can be further processed like other objects. Here, we focus on reviewing measure approaches between individuals due to the previous consideration.

A data object is described by a set of features, usually represented as a multidimensional vector. The features can be quantitative or qualitative, continuous or binary, nominal or ordinal, which determine the corresponding measure mechanisms.

A distance or dissimilarity function on a data set $\mathbf{X}$ is defined to satisfy the following conditions.

- 1) Symmetry. $D(\mathbf{x}_i, \mathbf{x}_j) = D(\mathbf{x}_j, \mathbf{x}_i)$;
- 2) Positivity. $D(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ for all $\mathbf{x}_i$ and $\mathbf{x}_j$.

If conditions

- 3) Triangle inequality.

$$D(\mathbf{x}_i, \mathbf{x}_j) \leq D(\mathbf{x}_i, \mathbf{x}_k) + D(\mathbf{x}_k, \mathbf{x}_j) \text{ for all } \mathbf{x}_i, \mathbf{x}_j, \text{ and } \mathbf{x}_k$$

and (4) Reflexivity. $D(\mathbf{x}_i, \mathbf{x}_j) = 0$ iff $\mathbf{x}_i = \mathbf{x}_j$ also hold, it is called a metric.

Likewise, a similarity function is defined to satisfy the conditions in the following.

- 1) Symmetry. $S(\mathbf{x}_i, \mathbf{x}_j) = S(\mathbf{x}_j, \mathbf{x}_i)$;
- 2) Positivity. $0 \leq S(\mathbf{x}_i, \mathbf{x}_j) \leq 1$, for all $\mathbf{x}_i$ and $\mathbf{x}_j$.

If it also satisfies conditions

- 3)

$$S(\mathbf{x}_i, \mathbf{x}_j)S(\mathbf{x}_j, \mathbf{x}_k) \leq [S(\mathbf{x}_i, \mathbf{x}_j) + S(\mathbf{x}_j, \mathbf{x}_k)]S(\mathbf{x}_i, \mathbf{x}_k) \\ \text{for all } \mathbf{x}_i, \mathbf{x}_j \text{ and } \mathbf{x}_k$$

and (4) $S(\mathbf{x}_i, \mathbf{x}_j) = 1$ iff $\mathbf{x}_i = \mathbf{x}_j$, it is called a similarity metric.

For a data set with N input patterns, we can define an $N \times N$ symmetric matrix, called proximity matrix, whose (i, j) th element represents the similarity or dissimilarity measure for the i th and j th patterns $(i, j = 1, \dots, N)$.

Typically, distance functions are used to measure continuous features, while similarity measures are more important for qualitative variables. We summarize some typical measures for continuous features in Table I. The selection of different measures is problem dependent. For binary features, a similarity measure is commonly used (dissimilarity measures can be obtained by simply using $D_{ij} = 1 - S_{ij}$). Suppose we use two binary subscripts to count features in two objects. n_{00} and n_{11} represent the number of simultaneous absence or presence of features in two objects, and n_{01} and n_{10} count the features present only in one object. Then two types of commonly used similarity measures for data points $\mathbf{x}_i$ and $\mathbf{x}_j$ are illustrated in the following.

$$S_{ij} = \frac{n_{11} + n_{00}}{n_{11} + n_{00} + w(n_{10} + n_{01})} \\ w = 1, \quad \text{simple matching coefficient} \\ w = 2, \quad \text{Rogers and Tanimoto measure.} \\ w = 1/2, \quad \text{Gower and Legendre measure.}$$

These measures compute the match between two objects directly. Unmatched pairs are weighted based on their contribution to the similarity.

$$S_{ij} = \frac{n_{11}}{n_{11} + w(n_{10} + n_{01})} \\ w = 1, \quad \text{Jaccard coefficient} \\ w = 2, \quad \text{Sokal and Sneath measure.} \\ w = 1/2, \quad \text{Gower and Legendre measure.}$$

These measures focus on the co-occurrence features while ignoring the effect of co-absence.

For nominal features that have more than two states, a simple strategy needs to map them into new binary features [161], while a more effective method utilizes the matching criterion

$$S_{ij} = \frac{1}{d} \sum_{l=1}^d S_{ijl}$$

where

$$S_{ijl} = \begin{cases} 0, & \text{if } i \text{ and } j \text{ do not match} \\ 1, & \text{if } i \text{ and } j \text{ match} \end{cases}$$

[88]. Ordinal features order multiple states according to some standard and can be compared by using continuous dissimilarity measures discussed in [161]. Edit distance for alphabetic sequences is discussed in Section II-J. More discussion on sequences and strings comparisons can be found in [120] and [236].

Generally, for objects consisting of mixed variables, we can map all these variables into the interval (0, 1) and use measures like the Euclidean metric. Alternatively, we can transform them into binary variables and use binary similarity functions. The drawback of these methods is the information loss. A more powerful method was described by Gower in the form of $S_{ij} = (\sum_{l=1}^d \eta_{ijl} S_{ijl}) / (\sum_{l=1}^d \eta_{ijl})$, where S_{ijl} indicates the similarity for the l th feature and η_{ijl} is a 0–1 coefficient based on whether the measure of the two objects is missing [88], [112].

B. Hierarchical Clustering

Hierarchical clustering (HC) algorithms organize data into a hierarchical structure according to the proximity matrix. The results of HC are usually depicted by a binary tree or dendrogram. The root node of the dendrogram represents the whole data set and each leaf node is regarded as a data object. The intermediate nodes, thus, describe the extent that the objects are proximal to each other; and the height of the dendrogram usually expresses the distance between each pair of objects or clusters, or an object and a cluster. The ultimate clustering results can be obtained by cutting the dendrogram at different levels. This representation provides very informative descriptions and visualization for the potential data clustering structures, especially when real hierarchical relations exist in the data, like the data from evolutionary research on different species of organisms. HC algorithms are mainly classified as agglomerative methods and divisive methods. Agglomerative clustering starts with N clusters and each of them includes exactly one object. A series of merge operations are then followed out that finally lead all objects to the same group. Divisive clustering proceeds in an opposite way. In the beginning, the entire data set belongs to a cluster and a procedure successively divides it until all clusters are singleton clusters. For a cluster with N objects, there are $2^{N-1} - 1$ possible two-subset divisions, which is very expensive in computation [88]. Therefore, divisive clustering is not commonly used in practice. We focus on the agglomerative clustering in the following discussion and some of divisive clustering applications for binary data can be found in [88]. Two divisive clustering algorithms, named MONA and DIANA, are described in [161].

The general agglomerative clustering can be summarized by the following procedure.

- 1) Start with N singleton clusters. Calculate the proximity matrix for the N clusters.
- 2) Search the minimal distance

$$D(C_i, C_j) = \min_{\substack{1 \leq m, l \leq N \\ m \neq l}} D(C_m, C_l)$$

where $D(*, *)$ is the distance function discussed before, in the proximity matrix, and combine cluster C_i and C_j to form a new cluster.

- 3) Update the proximity matrix by computing the distances between the new cluster and the other clusters.
- 4) Repeat steps 2)–3) until all objects are in the same cluster.

Based on the different definitions for distance between two clusters, there are many agglomerative clustering algorithms. The simplest and most popular methods include single linkage [256] and complete linkage technique [258]. For the single linkage method, the distance between two clusters is determined by the two closest objects in different clusters, so it is also called nearest neighbor method. On the contrary, the complete linkage method uses the farthest distance of a pair of objects to define inter-cluster distance. Both the single linkage and the complete linkage method can be generalized by the recurrence formula proposed by Lance and Williams [178] as

$$D(C_l, (C_i, C_j)) = \alpha_i D(C_l, C_i) + \alpha_j D(C_l, C_j) + \beta D(C_i, C_j) + \gamma |D(C_l, C_i) - D(C_l, C_j)|$$

where $D(*, *)$ is the distance function and $\alpha_i, \alpha_j, \beta$, and γ are coefficients that take values dependent on the scheme used.

The formula describes the distance between a cluster l and a new cluster formed by the merge of two clusters i and j . Note that when $\alpha_i = \alpha_j = 1/2, \beta = 0$, and $\gamma = -1/2$, the formula becomes

$$D(C_l, (C_i, C_j)) = \min(D(C_l, C_i), D(C_l, C_j))$$

which corresponds to the single linkage method. When $\alpha_i = \alpha_j = \gamma = 1/2$ and $\beta = 0$, the formula is

$$D(C_l, (C_i, C_j)) = \max(D(C_l, C_i), D(C_l, C_j))$$

which corresponds to the complete linkage method.

Several more complicated agglomerative clustering algorithms, including group average linkage, median linkage, centroid linkage, and Ward's method, can also be constructed by selecting appropriate coefficients in the formula. A detailed table describing the coefficient values for different algorithms is offered in [150] and [210]. Single linkage, complete linkage and average linkage consider all points of a pair of clusters, when calculating their inter-cluster distance, and are also called graph methods. The others are called geometric methods since they use geometric centers to represent clusters and determine their distances. Remarks on important features and properties of these methods are summarized in [88]. More inter-cluster

distance measures, especially the mean-based ones, were introduced by Yager, with further discussion on their possible effect to control the hierarchical clustering process [289].

The common criticism for classical HC algorithms is that they lack robustness and are, hence, sensitive to noise and outliers. Once an object is assigned to a cluster, it will not be considered again, which means that HC algorithms are not capable of correcting possible previous misclassification. The computational complexity for most of HC algorithms is at least $O(N^2)$ and this high cost limits their application in large-scale data sets. Other disadvantages of HC include the tendency to form spherical shapes and reversal phenomenon, in which the normal hierarchical structure is distorted.

In recent years, with the requirement for handling large-scale data sets in data mining and other fields, many new HC techniques have appeared and greatly improved the clustering performance. Typical examples include CURE [116], ROCK [117], Chameleon [159], and BIRCH [295].

The main motivations of BIRCH lie in two aspects, the ability to deal with large data sets and the robustness to outliers [295]. In order to achieve these goals, a new data structure, clustering feature (CF) tree, is designed to store the summaries of the original data. The CF tree is a height-balanced tree, with each internal vertex composed of entries defined as $[\mathbf{CF}_i, \text{child}_i]$, $i = 1, \dots, B$, where $\mathbf{CF}_i$ is a representation of the cluster i and is defined as $\mathbf{CF}_i = (N_i, \mathbf{LS}, \mathbf{SS})$, where N_i is the number of data objects in the cluster, $\mathbf{LS}$ is the linear sum of the objects, and $\mathbf{SS}$ is the squared sum of the objects, child_i is a pointer to the i th child node, and B is a threshold parameter that determines the maximum number of entries in the vertex, and each leaf composed of entries in the form of $[\mathbf{CF}_i]$, $i = 1, \dots, L$, where L is the threshold parameter that controls the maximum number of entries in the leaf. Moreover, the leaves must follow the restriction that the diameter $((\sum_{l=1}^{N_i} \sum_{m=1}^{N_i} (\mathbf{x}_l - \mathbf{x}_m)^2 / N_i(N_i - 1))^{1/2})$ of each entry in the leaf is less than a threshold T . The CF tree structure captures the important clustering information of the original data while reducing the required storage. Outliers are eliminated from the summaries by identifying the objects sparsely distributed in the feature space. After the CF tree is built, an agglomerative HC is applied to the set of summaries to perform global clustering. An additional step may be performed to refine the clusters. BIRCH can achieve a computational complexity of $O(N)$.

Noticing the restriction of centroid-based HC, which is unable to identify arbitrary cluster shapes, Guha, Rastogi, and Shim developed a HC algorithm, called CURE, to explore more sophisticated cluster shapes [116]. The crucial feature of CURE lies in the usage of a set of well-scattered points to represent each cluster, which makes it possible to find rich cluster shapes other than hyperspheres and avoids both the chaining effect [88] of the minimum linkage method and the tendency to favor clusters with similar sizes of centroid. These representative points are further shrunk toward the cluster centroid according to an adjustable parameter in order to weaken the effects of outliers. CURE utilizes random sample (and partition) strategy to reduce computational complexity. Guha *et al.* also proposed another agglomerative HC algorithm, ROCK, to group data with qualitative attributes [117]. They used a novel measure

“link” to describe the relation between a pair of objects and their common neighbors. Like CURE, a random sample strategy is used to handle large data sets. Chameleon is constructed from graph theory and will be discussed in Section II-E.

Relative hierarchical clustering (RHC) is another exploration that considers both the internal distance (distance between a pair of clusters which may be merged to yield a new cluster) and the external distance (distance from the two clusters to the rest), and uses the ratio of them to decide the proximities [203]. Leung *et al.* showed an interesting hierarchical clustering based on scale-space theory [180]. They interpreted clustering using a blurring process, in which each datum is regarded as a light point in an image, and a cluster is represented as a blob. Li and Biswas extended agglomerative HC to deal with both numeric and nominal data. The proposed algorithm, called similarity-based agglomerative clustering (SBAC), employs a mixed data measure scheme that pays extra attention to less common matches of feature values [183]. Parallel techniques for HC are discussed in [69] and [217], respectively.

C. Squared Error—Based Clustering (Vector Quantization)

In contrast to hierarchical clustering, which yields a successive level of clusters by iterative fusions or divisions, partitional clustering assigns a set of objects into K clusters with no hierarchical structure. In principle, the optimal partition, based on some specific criterion, can be found by enumerating all possibilities. But this brute force method is infeasible in practice, due to the expensive computation [189]. Even for a small-scale clustering problem (organizing 30 objects into 3 groups), the number of possible partitions is 2×10^{14} . Therefore, heuristic algorithms have been developed in order to seek approximate solutions.

One of the important factors in partitional clustering is the criterion function [124]. The sum of squared error function is one of the most widely used criteria. Suppose we have a set of objects $\mathbf{x}_j \in \mathbb{R}^d$, $j = 1, \dots, N$, and we want to organize them into K subsets $C = \{C_1, \dots, C_K\}$. The squared error criterion then is defined as

$$J(\mathbf{\Gamma}, \mathbf{M}) = \sum_{i=1}^K \sum_{j=1}^N \gamma_{ij} \|\mathbf{x}_j - \mathbf{m}_i\|^2$$

where

$\mathbf{\Gamma}$ = a partition matrix;

$\{\gamma_{ij}\}$

$$\gamma_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_j \in \text{cluster } i \\ 0 & \text{otherwise} \end{cases} \quad \text{with } \sum_{i=1}^K \gamma_{ij} = 1 \forall j;$$

$\mathbf{M}$ = cluster prototype or centroid (means) matrix;

$[\mathbf{m}_1, \dots, \mathbf{m}_K]$

$\mathbf{m}_i$ = sample mean for the i th cluster;

$(1/N_i) \sum_{j=1}^N \gamma_{ij} \mathbf{x}_j$

N_i = number of objects in the i th cluster.

Note the relation between the sum of squared error criterion and the scatter matrices defined in multiclass discriminant analysis [75],

$$S_T = S_W + S_B$$

where

$$\begin{aligned} S_T &= \text{total scatter matrix;} \\ S_W &= \text{within-class scatter matrix;} \\ \sum_{i=1}^K \sum_{j=1}^N \gamma_{ij}(\mathbf{x}_j - \mathbf{m}_i)(\mathbf{x}_j - \mathbf{m}_i)^T \\ S_B &= \text{between-class scatter matrix;} \text{ and} \\ \sum_{i=1}^K N_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T \\ \mathbf{m} &= \text{mean vector for the whole data set.} \\ (1/N) \sum_{i=1}^K N_i \mathbf{m}_i \end{aligned}$$

It is not difficult to see that the criterion based on the trace of S_W is the same as the sum of squared error criterion. To minimize the squared error criterion is equivalent to minimizing the trace of S_W or maximizing the trace of S_B . We can obtain a rich class of criterion functions based on the characteristics of S_W and S_B [75].

The K -means algorithm is the best-known squared error-based clustering algorithm [94], [191].

- 1) Initialize a K -partition randomly or based on some prior knowledge. Calculate the cluster prototype matrix $\mathbf{M} = [\mathbf{m}_1, \dots, \mathbf{m}_K]$.
- 2) Assign each object in the data set to the nearest cluster C_w , i.e.

$$\begin{aligned} \mathbf{x}_j \in C_w, \quad & \text{if } \|\mathbf{x}_j - \mathbf{m}_w\| < \|\mathbf{x}_j - \mathbf{m}_i\| \\ & \text{for } j = 1, \dots, N, \quad i \neq w, \quad \text{and } i = 1, \dots, K. \end{aligned}$$

- 3) Recalculate the cluster prototype matrix based on the current partition.
- 4) Repeat steps 2)–3) until there is no change for each cluster.

The K -means algorithm is very simple and can be easily implemented in solving many practical problems. It can work very well for compact and hyperspherical clusters. The time complexity of K -means is $O(NKd)$. Since K and d are usually much less than N , K -means can be used to cluster large data sets. Parallel techniques for K -means are developed that can largely accelerate the algorithm [262]. The drawbacks of K -means are also well studied, and as a result, many variants of K -means have appeared in order to overcome these obstacles. We summarize some of the major disadvantages with the proposed improvement in the following.

- 1) There is no efficient and universal method for identifying the initial partitions and the number of clusters K . The convergence centroids vary with different initial points. A general strategy for the problem is to run the algorithm many times with random initial partitions. Peña, Lozano, and Larrañaga compared the random method with other three classical initial partition methods by Forgy [94], Kaufman [161], and MacQueen [191], based on the effectiveness, robustness, and convergence speed criteria [227]. According to their experimental results, the random and Kaufman's method work much better than the other two under the first two criteria and by further considering the convergence speed, they recommended Kaufman's method. Bradley and Fayyad presented a refinement algorithm that first utilizes K -means M times to M random subsets from the original data [43]. The set formed from the union of the solution (centroids of the K clusters)

of the M subsets is clustered M times again, setting each subset solution as the initial guess. The starting points for the whole data are obtained by choosing the solution with minimal sum of squared distances. Likas, Vlassis, and Verbeek proposed a global K -means algorithm consisting of a series of K -means clustering procedures with the number of clusters varying from 1 to K [186]. After finding the centroid for only one cluster existing, at each $k, k = 2, \dots, K$, the previous $k - 1$ centroids are fixed and the new centroid is selected by examining all data points. The authors claimed that the algorithm is independent of the initial partitions and provided accelerating strategies. But the problem on computational complexity exists, due to the requirement for executing K -means N times for each value of k .

An interesting technique, called ISODATA, developed by Ball and Hall [21], deals with the estimation of K . ISODATA can dynamically adjust the number of clusters by merging and splitting clusters according to some predefined thresholds (in this sense, the problem of identifying the initial number of clusters becomes that of parameter (threshold) tweaking). The new K is used as the expected number of clusters for the next iteration.

- 2) The iteratively optimal procedure of K -means cannot guarantee convergence to a global optimum. The stochastic optimal techniques, like simulated annealing (SA) and genetic algorithms (also see part II.F), can find the global optimum with the price of expensive computation. Krishna and Murty designed new operators in their hybrid scheme, GKA, in order to achieve global search and fast convergence [173]. The defined biased mutation operator is based on the Euclidean distance between an object and the centroids and aims to avoid getting stuck in a local optimum. Another operator, the K -means operator (KMO), replaces the computationally expensive crossover operators and alleviates the complexities coming with them. An adaptive learning rate strategy for the online mode K -means is illustrated in [63]. The learning rate is exclusively dependent on the within-group variations and can be adjusted without involving any user activities. The proposed enhanced LBG (ELBG) algorithm adopts a roulette mechanism typical of genetic algorithms to become near-optimal and therefore, is not sensitive to initialization [222].
- 3) K -means is sensitive to outliers and noise. Even if an object is quite far away from the cluster centroid, it is still forced into a cluster and, thus, distorts the cluster shapes. ISODATA [21] and PAM [161] both consider the effect of outliers in clustering procedures. ISODATA gets rid of clusters with few objects. The splitting operation of ISODATA eliminates the possibility of elongated clusters typical of K -means. PAM utilizes real data points (medoids) as the cluster prototypes and avoids the effect of outliers. Based on the same consideration, a K -medoids algorithm is presented in [87]

by searching the discrete 1-medians as the cluster centroids.

- 4) The definition of “means” limits the application only to numerical variables. The K -medoids algorithm mentioned previously is a natural choice, when the computation of means is unavailable, since the medoids do not need any computation and always exist [161]. Huang [142] and Gupta *et al.* [118] defined different dissimilarity measures to extend K -means to categorical variables. For Huang’s method, the clustering goal is to minimize the cost function $J(\Gamma, \mathbf{Q}) = \sum_{i=1}^K \sum_{j=1}^N \gamma_{ij} D(\mathbf{x}_j, \mathbf{Q}_i)$, where

$$D(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^d \delta(x_{il}, x_{jl})$$

and

$$\delta(x_{il}, x_{jl}) = \begin{cases} 1 & x_{il} \neq x_{jl} \\ 0 & x_{il} = x_{jl} \end{cases}$$

with a set of d -dimensional vectors $\mathbf{Q} = \{\mathbf{Q}_1, \dots, \mathbf{Q}_K\}$, where $\mathbf{Q}_j = (Q_{j1}, \dots, Q_{jd})$. Each vector $\mathbf{Q}_j$ is known as a mode and is defined to minimize the sum of distances $\sum_{i=1}^N D(\mathbf{x}_i, \mathbf{Q}_j)$. The proposed K -modes algorithm operates in a similar way as K -means.

Several recent advances on K -means and other squared-error based clustering algorithms with their applications can be found in [125], [155], [222], [223], [264], and [277].

D. Mixture Densities-Based Clustering (pdf Estimation via Mixture Densities)

In the probabilistic view, data objects are assumed to be generated according to several probability distributions. Data points in different clusters were generated by different probability distributions. They can be derived from different types of density functions (e.g., multivariate Gaussian or t -distribution), or the same families, but with different parameters. If the distributions are known, finding the clusters of a given data set is equivalent to estimating the parameters of several underlying models. Suppose the prior probability (also known as mixing probability) $P(C_i)$ for cluster $C_i, i = 1, \dots, K$ (here, K is assumed to be known and methods for estimating K are discussed in Section II-M) and the conditional probability density $p(\mathbf{x} | C_i, \boldsymbol{\theta}_i)$ (also known as component density), where $\boldsymbol{\theta}_i$ is the unknown parameter vector, are known. Then, the mixture probability density for the whole data set is expressed as

$$p(\mathbf{x} | \boldsymbol{\theta}) = \sum_{i=1}^K p(\mathbf{x} | C_i, \boldsymbol{\theta}_i) P(C_i)$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_K)$, and $\sum_{i=1}^K P(C_i) = 1$. As long as the parameter vector $\boldsymbol{\theta}$ is decided, the posterior probability for assigning a data point to a cluster can be easily calculated with Bayes’s theorem. Here, the mixtures can be constructed with any types of components, but more commonly, multivariate

Gaussian densities are used due to its complete theory and analytical tractability [88], [297].

Maximum likelihood (ML) estimation is an important statistical approach for parameter estimation [75] and it considers the best estimate as the one that maximizes the probability of generating all the observations, which is given by the joint density function

$$p(\{\mathbf{x}_1, \dots, \mathbf{x}_N\} | \boldsymbol{\theta}) = \prod_{j=1}^N p(\mathbf{x}_j | \boldsymbol{\theta})$$

or, in a logarithm form

$$l(\boldsymbol{\theta}) = \sum_{j=1}^N \ln p(\mathbf{x}_j | \boldsymbol{\theta}).$$

The best estimate can be achieved by solving the log-likelihood equations $(\partial l(\boldsymbol{\theta})) / (\partial \boldsymbol{\theta}_i) = 0$.

Unfortunately, since the solutions of the likelihood equations cannot be obtained analytically in most circumstances [90], [197], iteratively suboptimal approaches are required to approximate the ML estimates. Among these methods, the expectation-maximization (EM) algorithm is the most popular [196]. EM regards the data set as incomplete and divides each data point $\mathbf{x}_j$ into two parts $\mathbf{x}_j = \{\mathbf{x}_j^g, \mathbf{x}_j^m\}$, where $\mathbf{x}_j^g$ represents the observable features and $\mathbf{x}_j^m = (x_{j1}^m, \dots, x_{jK}^m)$ is the missing data, where x_{ji}^m chooses value 1 or 0 according to whether $\mathbf{x}_j$ belongs to the component i or not. Thus, the complete data log-likelihood is

$$l(\boldsymbol{\theta}) = \sum_{j=1}^N \sum_{i=1}^K x_{ji}^m \log [P(C_i) p(\mathbf{x}_j^g | \boldsymbol{\theta}_i)].$$

The standard EM algorithm generates a series of parameter estimates $\{\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \dots, \boldsymbol{\theta}^T\}$, where T represents the reaching of the convergence criterion, through the following steps:

- 1) initialize $\boldsymbol{\theta}^0$ and set $t = 0$;
- 2) e-step: Compute the expectation of the complete data log-likelihood

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t) = E[\log p(\mathbf{x}^g, \mathbf{x}^m | \boldsymbol{\theta}) | \mathbf{x}^g, \boldsymbol{\theta}^t];$$

- 3) m-step: Select a new parameter estimate that maximizes the Q -function, $\boldsymbol{\theta}^{t+1} = \arg \max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^t)$;
- 4) Increase $t = t + 1$; repeat steps 2)–3) until the convergence condition is satisfied.

The major disadvantages for EM algorithm are the sensitivity to the selection of initial parameters, the effect of a singular covariance matrix, the possibility of convergence to a local optimum, and the slow convergence rate [96], [196]. Variants of EM for addressing these problems are discussed in [90] and [196].

A valuable theoretical note is the relation between the EM algorithm and the K -means algorithm. Celeux and Govaert proved that classification EM (CEM) algorithm under a spherical Gaussian mixture is equivalent to the K -means algorithm [58].

Fraley and Raftery described a comprehensive mixture-model based clustering scheme [96], which was implemented as a software package, known as MCLUST [95]. In this case, the component density is multivariate Gaussian, with a mean vector μ and a covariance matrix Σ as the parameters to be estimated. The covariance matrix for each component can further be parameterized by virtue of eigenvalue decomposition, represented as $\Sigma = \lambda \Pi \Lambda \Pi^T$, where λ is a scalar, Π is the orthogonal matrix of eigenvectors, and Λ is the diagonal matrix based on the eigenvalues of Σ [96]. These three elements determine the geometric properties of each component. After the maximum number of clusters and the candidate models are specified, an agglomerative hierarchical clustering was used to ignite the EM algorithm by forming an initial partition, which includes at most the maximum number of clusters, for each model. The optimal clustering result is achieved by checking the Bayesian information criterion (BIC) value discussed in Section II-M. GMDD is also based on multivariate Gaussian densities and is designed as a recursive algorithm that sequentially estimates each component [297]. GMDD views data points that are not generated from a distribution as noise and utilizes an enhanced model-fitting estimator to construct each component from the contaminated model. AutoClass considers more families of probability distributions (e.g., Poisson and Bernoulli) for different data types [59]. A Bayesian approach is used in AutoClass to find out the optimal partition of the given data based on the prior probabilities. Its parallel realization is described in [228]. Other important algorithms and programs include Multimix [147], EM based mixture program (EMMIX) [198], and Snob [278].

E. Graph Theory-Based Clustering

The concepts and properties of graph theory [126] make it very convenient to describe clustering problems by means of graphs. Nodes V of a weighted graph G correspond to data points in the pattern space and edges E reflect the proximities between each pair of data points. If the dissimilarity matrix is defined as

$$D_{ij} = \begin{cases} 1 & \text{if } D(\mathbf{x}_i, \mathbf{x}_j) < d_0 \\ 0 & \text{otherwise} \end{cases}$$

where d_0 is a threshold value, the graph is simplified to an unweighted threshold graph. Both the single linkage HC and the complete linkage HC can be described on the basis of the threshold graph. Single linkage clustering is equivalent to seeking maximally connected subgraphs (components) while complete linkage clustering corresponds to finding maximally complete subgraphs (cliques) [150]. Jain and Dubes illustrated and discussed more applications of graph theory (e.g., Hubert's algorithm and Johnson's algorithm) for hierarchical clustering in [150]. Chameleon [159] is a newly developed agglomerative HC algorithm based on the k -nearest-neighbor graph, in which an edge is eliminated if both vertices are not within the k closest points related to each other. At the first step, Chameleon divides the connectivity graph into a set of subclusters with the minimal edge cut. Each subgraph should contain enough nodes in order for effective similarity computation. By combining both the relative interconnectivity and relative closeness, which

make Chameleon flexible enough to explore the characteristics of potential clusters, Chameleon merges these small subsets and, thus, comes up with the ultimate clustering solutions. Here, the relative interconnectivity (or closeness) is obtained by normalizing the sum of weights (or average weight) of the edges connecting the two clusters over the internal connectivity (or closeness) of the clusters. DTG is another important graph representation for HC analysis. Cherng and Lo constructed a hypergraph (each edge is allowed to connect more than two vertices) from the DTG and used a two-phase algorithm that is similar to Chameleon to find clusters [61]. Another DTG-based application, known as AMOEBA algorithm, is presented in [86].

Graph theory can also be used for nonhierarchical clusters. Zahn's clustering algorithm seeks connected components as clusters by detecting and discarding inconsistent edges in the minimum spanning tree [150]. Hartuv and Shamir treated clusters as HCS, where "highly connected" means the connectivity (the minimum number of edges needed to disconnect a graph) of the subgraph is at least half as great as the number of the vertices [128]. A minimum cut (mcut) procedure, which aims to separate a graph with a minimum number of edges, is used to find these HCSs recursively. Another algorithm, called CLICK, is based on the calculation of the minimum weight cut to form clusters [247]. Here, the graph is weighted and the edge weights are assigned a new interpretation, by combining probability and graph theory. The edge weight between node i and j is defined as shown in

$$e_{ij} = \log \frac{\Pr ob(i, j \text{ belong to the same cluster} | S_{ij})}{\Pr ob(i, j \text{ does not belong to the same cluster} | S_{ij})}$$

where S_{ij} represents the similarity between the two nodes. CLICK further assumes that the similarity values within clusters and between clusters follow Gaussian distributions with different means and variances, respectively. Therefore, the previous equation can be rewritten by using Bayes' theorem as

$$e_{ij} = \log \frac{p_0 \sigma_B}{(1-p_0) \sigma_W} + \frac{(S_{ij} - \mu_B)^2}{2\sigma_B^2} - \frac{(S_{ij} - \mu_W)^2}{2\sigma_W^2}$$

where p_0 is the prior probability that two objects belong to the same cluster and $\mu_B, \sigma_B^2, \mu_W, \sigma_W^2$ are the means and variances for between-cluster similarities and within-clusters similarities, respectively. These parameters can be estimated either from prior knowledge, or by using parameter estimation methods [75]. CLICK recursively checks the current subgraph, and generates a kernel list, which consists of the components satisfying some criterion function. Subgraphs that include only one node are regarded as singletons, and are separated for further manipulation. Using the kernels as the basic clusters, CLICK carries out a series of singleton adoptions and cluster merge to generate the resulting clusters. Additional heuristics are provided to accelerate the algorithm performance.

Similarly, CAST considers a probabilistic model in designing a graph theory-based clustering algorithm [29]. Clusters are modeled as corrupted clique graphs, which, in ideal conditions, are regarded as a set of disjoint cliques. The effect of noise is incorporated by adding or removing edges from the ideal

model, with a probability α . Proofs were given for recovering the uncorrupted graph with a high probability. CAST is the heuristic implementation of the original theoretical version. CAST creates clusters sequentially, and each cluster begins with a random and unassigned data point. The relation between a data point i and a cluster C_0 being built is determined by the affinity, defined as $a(i) = \sum_{j \in C_0} S_{ij}$, and the affinity threshold parameter t . When $a(i) \geq t|C_0|$, it means that the data point is highly related to the cluster and vice versa. CAST alternately adds high affinity data points or deletes low affinity data points from the cluster until no more changes occur.

F. Combinatorial Search Techniques-Based Clustering

The basic object of search techniques is to find the global or approximate global optimum for combinatorial optimization problems, which usually have NP-hard complexity and need to search an exponentially large solution space. Clustering can be regarded as a category of optimization problems. Given a set of data points $\mathbf{x}_j \in \mathbb{R}^d, j = 1, \dots, N$, clustering algorithms aim to organize them into K subsets $\{C_1, \dots, C_K\}$ that optimize some criterion function. The possible partition for N points into K clusters is given by the formula [189]

$$P(N, K) = \frac{1}{K!} \sum_{m=1}^K (-1)^{K-m} C_K^m m^N.$$

As shown before, even for small N and K , the computational complexity is extremely expensive, not to mention the large-scale clustering problems frequently encountered in recent decades. Simple local search techniques, like hill-climbing algorithms, are utilized to find the partitions, but they are easily stuck in local minima and therefore cannot guarantee optimality. More complex search methods (e.g., evolutionary algorithms (EAs) [93], SA [165], and Tabu search (TS) [108] are known as stochastic optimization methods, while deterministic annealing (DA) [139], [234] is the most typical deterministic search technique) can explore the solution space more flexibly and efficiently.

Inspired by the natural evolution process, evolutionary computation, which consists of genetic algorithms (GAs), evolution strategies (ESs), evolutionary programming (EP), and genetic programming (GP), optimizes a population of structure by using a set of evolutionary operators [93]. An optimization function, called the fitness function, is the standard for evaluating the optimizing degree of the population, in which each individual has its corresponding fitness. Selection, recombination, and mutation are the most widely used evolutionary operators. The selection operator ensures the continuity of the population by favoring the best individuals in the next generation. The recombination and mutation operators support the diversity of the population by exerting perturbations on the individuals. Among many EAs, GAs [140] are the most popular approaches applied in cluster analysis. In GAs, each individual is usually encoded as a binary bit string, called a chromosome. After an initial population is generated according to some heuristic rules or just randomly, a series of operations, including selection, crossover and mutation, are iteratively applied to the population until the stop condition is satisfied.

Hall, Özyurt, and Bezdek proposed a GGA that can be regarded as a general scheme for center-based (hard or fuzzy) clustering problems [122]. Fitness functions are reformulated from the standard sum of squared error criterion function in order to adapt the change of the construction of the optimization problem (only the prototype matrix is needed)

$$J(\mathbf{M}) = \sum_{j=1}^N \min(D_{1j}, D_{2j}, \dots, D_{Kj}) \quad \text{for hard clustering}$$

$$J(\mathbf{M}) = \sum_{j=1}^N \left(\sum_{i=1}^K D_{ij}^{1/(1-m)} \right)^{1-m} \quad \text{for fuzzy clustering}$$

where $D_{ij}, i = 1, \dots, K, j = 1, \dots, N$, is the distance between the i th cluster and the j th data object, and m is the fuzzification parameter.

GGA proceeds with the following steps.

- 1) Choose appropriate parameters for the algorithm. Initialize the population randomly with P individuals, each of which represents a $K \times d$ prototype matrix and is encoded as gray codes. Calculate the fitness value for each individual.
- 2) Use selection (tournament selection) operator to choose parental members for reproduction.
- 3) Use crossover (two-point crossover) and mutation (bit-wise mutation) operator to generate offspring from the individuals chosen in step 2).
- 4) Determine the next generation by keeping the individuals with the highest fitness.
- 5) Repeat steps 2)–4) until the termination condition is satisfied.

Other GAs-based clustering applications have appeared based on a similar framework. They are different in the meaning of an individual in the population, encoding methods, fitness function definition, and evolutionary operators [67], [195], [273]. The algorithm CLUSTERING in [273] includes a heuristic scheme for estimating the appropriate number of clusters in the data. It also uses a nearest-neighbor algorithm to divide data into small subsets, before GAs-based clustering, in order to reduce the computational complexity. GAs are very useful for improving the performance of K -means algorithms. Babu and Murty used GAs to find good initial partitions [15]. Krishna and Murty combined GA with K -means and developed GKA algorithm that can find the global optimum [173]. As indicated in Section II-C, the algorithm ELBG uses the roulette mechanism to address the problems due to the bad initialization [222]. It is worthwhile to note that ELBG are equivalent to another algorithm, fully automatic clustering system (FACS) [223], in terms of quantization level detection. The difference lies in the input parameters employed (ELBG adopts the number of quantization levels, while FACS uses the desired distortion error). Except the previous applications, GAs can also be used for hierarchical clustering. Lozano and Larrañaga discussed the properties of ultrametric distance [127] and reformulated the hierarchical clustering as an optimization

problem that tries to find the closest ultrametric distance for a given dissimilarity with Euclidean norm [190]. They suggested an order-based GA to solve the problem. Clustering algorithms based on ESs and EP are described and analyzed in [16] and [106], respectively.

TS is a combinatorial search technique that uses the tabu list to guide the search process consisting of a sequence of moves. The tabu list stores part or all of previously selected moves according to the specified size. These moves are forbidden in the current search and are called tabu. In the TS clustering algorithm developed by Al-Sultan [9], a set of candidate solutions are generated from the current solution with some strategy. Each candidate solution represents the allocations of N data objects in K clusters. The candidate with the optimal cost function is selected as the current solution and appended to the tabu list, if it is not already in the tabu list or meets the aspiration criterion, which can overrule the tabu restriction. Otherwise, the remaining candidates are evaluated in the order of their cost function values, until all these conditions are satisfied. When all the candidates are tabu, a new set of candidate solutions are created followed by the same search process. The search process proceeds until the maximum number of iterations is reached. Sung and Jin's method includes more elaborate search processes with the packing and releasing procedures [266]. They also used a secondary tabu list to keep the search from trapping into the potential cycles. A fuzzy version of TS clustering can be found in [72].

SA is also a sequential and global search technique and is motivated by the annealing process in metallurgy [165]. SA allows the search process to accept a worse solution with a certain probability. The probability is controlled by a parameter, known as temperature T and is usually expressed as $\exp(\Delta J/T)$, where ΔJ is the change of the energy (cost function). The temperature T goes through an annealing schedule from initial high to ultimate low values, which means that SA attempts to explore solution space more completely at high temperatures while favors the solutions that lead to lower energy at low temperatures. SA-based clustering was reported in [47] and [245]. The former illustrated an application of SA clustering to evaluate different clustering criteria and the latter investigated the effects of input parameters to the clustering performance.

Hybrid approaches that combine these search techniques are also proposed. A tabu list is used in a GA clustering algorithm to preserve the variety of the population and avoid repeating computation [243]. An application of SA for improving TS was reported in [64]. The algorithm further reduces the possible moves to local optima.

The main drawback that plagues the search techniques-based clustering algorithms is the parameter selection. More often than not, search techniques introduce more parameters than other methods (like K -means). There are no theoretic guidelines to select the appropriate and effective parameters. Hall *et al.* provided some methods for setting parameters in their GAs-based clustering framework [122], but most of these criteria are still obtained empirically. The same situation exists for TS and SA clustering [9], [245]. Another problem is the computational complexity paid for the convergence to global optima. High computational requirement limits their applications in large-scale data sets.

G. Fuzzy Clustering

Except for GGA, the clustering techniques we have discussed so far are referred to as hard or crisp clustering, which means that each object is assigned to only one cluster. For fuzzy clustering, this restriction is relaxed, and the object can belong to all of the clusters with a certain degree of membership [293]. This is particularly useful when the boundaries among the clusters are not well separated and ambiguous. Moreover, the memberships may help us discover more sophisticated relations between a given object and the disclosed clusters.

FCM is one of the most popular fuzzy clustering algorithms [141]. FCM can be regarded as a generalization of ISODATA [76] and was realized by Bezdek [35]. FCM attempts to find a partition (c fuzzy clusters) for a set of data points $\mathbf{x}_j \in \mathbb{R}^d, j = 1, \dots, N$ while minimizing the cost function

$$J(\mathbf{U}, \mathbf{M}) = \sum_{i=1}^c \sum_{j=1}^N (u_{i,j})^m D_{ij}$$

where

- $\mathbf{U}=[u_{i,j}]_{c \times N}$ is the fuzzy partition matrix and $u_{i,j} \in [0, 1]$ is the membership coefficient of the j th object in the i th cluster;
- $\mathbf{M}=[\mathbf{m}_1, \dots, \mathbf{m}_c]$ is the cluster prototype (mean or center) matrix;
- $m \in [1, \infty)$ is the fuzzification parameter and usually is set to 2 [129];
- $D_{ij}=D(\mathbf{x}_j, \mathbf{m}_i)$ is the distance measure between $\mathbf{x}_j$ and $\mathbf{m}_i$.

We summarize the standard FCM as follows, in which the Euclidean or L_2 norm distance function is used.

- 1) Select appropriate values for m, c , and a small positive number ε . Initialize the prototype matrix $\mathbf{M}$ randomly. Set step variable $t = 0$.
- 2) Calculate (at $t = 0$) or update (at $t > 0$) the membership matrix $\mathbf{U}$ by

$$u_{ij}^{(t+1)} = 1 / \left(\sum_{l=1}^c (D_{lj}/D_{ij})^{1/(1-m)} \right) \quad \text{for } i = 1, \dots, c \quad \text{and} \quad j = 1, \dots, N.$$

- 3) Update the prototype matrix $\mathbf{M}$ by

$$\mathbf{m}_i^{(t+1)} = \left(\sum_{j=1}^N (u_{ij}^{(t+1)})^m \mathbf{x}_j \right) / \left(\sum_{j=1}^N (u_{ij}^{(t+1)})^m \right) \quad \text{for } i = 1, \dots, c.$$

- 4) Repeat steps 2)–3) until $\|\mathbf{M}^{(t+1)} - \mathbf{M}^{(t)}\| < \varepsilon$.

Numerous FCM variants and other fuzzy clustering algorithms have appeared as a result of the intensive investigation on the distance measure functions, the effect of weighting exponent on fuzziness control, the optimization approaches for fuzzy partition, and improvements of the drawbacks of FCM [84], [141].

Like its hard counterpart, FCM also suffers from the presence of noise and outliers and the difficulty to identify the initial partitions. Yager and Filev proposed a MM in order to estimate the

centers of clusters [290]. Candidate centers consist of a set of vertices that are formed by building a grid on the pattern space. The mountain function for a vertex $\mathbf{v}_i$ is defined as

$$M(\mathbf{v}_i) = \sum_{j=1}^N e^{-\alpha D(\mathbf{x}_j, \mathbf{v}_i)}$$

where $D(\mathbf{x}_j, \mathbf{v}_i)$ is the distance between the j th data object and the i th node, and α is a positive constant. Therefore, the closer a data object is to a vertex, the more the data object contributes to the mountain function. The vertex $\mathbf{v}_{m_1}$ with the maximum value of mountain function $M_{\mathbf{v}_{m_1}}$ is selected as the first center. A procedure, called mountain destruction, is performed to get rid of the effects of the selected center. This is achieved by subtracting the mountain function value for each of the rest vertices with an amount dependent on the current maximum mountain function value and the distance between the vertex and the center. The process iterates until the ratio between the current maximum and $M_{\mathbf{v}_{m_1}}$ is below some threshold. The connection of MM with several other fuzzy clustering algorithms was further discussed in [71]. Gath and Geva described an initialization strategy of unsupervised tracking of cluster prototypes in their 2-layer clustering scheme, in which FCM and fuzzy ML estimation are effectively combined [102].

Kersten suggested that city block distance (or L_1 norm) could improve the robustness of FCM to outliers [163]. Furthermore, Hathaway, Bezdek, and Hu extended FCM to a more universal case by using Minkowski distance (or L_p norm, $p \geq 1$) and seminorm ($0 < p < 1$) for the models that operate either directly on the data objects or indirectly on the dissimilarity measures [130]. According to their empirical results, the object data based models, with L_1 and L_2 norm, are recommended. They also pointed out the possible improvement of models for other L_p norm with the price of more complicated optimization operations. PCM is another approach for dealing with outliers [175]. Under this model, the memberships are interpreted by a possibilistic view, i.e., “the compatibilities of the points with the class prototypes” [175]. The effect of noise and outliers is abated with the consideration of typicality. In this case, the first condition for the membership coefficient described in Section I is relaxed to $\max_i u_{i,j} > 0, \forall j$. Accordingly, the cost function is reformulated as

$$J(\mathbf{U}, \mathbf{M}) = \sum_{i=1}^c \sum_{j=1}^N (u_{i,j})^m D_{ij} + \sum_{i=1}^c \eta_i \sum_{j=1}^N (1 - u_{i,j})^m$$

where η_i are some positive constants. The additional term tends to give credits to memberships with large values. A modified version in order to find appropriate clusters is proposed in [294]. Davé and Krishnapuram further elaborated the discussion on fuzzy clustering robustness and indicated its connection with robust statistics [71]. Relations among some widely used fuzzy clustering algorithms were discussed and their similarities to some robust statistical methods were also reviewed. They reached a unified framework as the conclusion for the previous discussion and proposed generic algorithms for robust clustering.

The standard FCM alternates the calculation of the membership and prototype matrix, which causes a computational burden for large-scale data sets. Kolen and Hutcheson accelerated the computation by combining updates of the two matrices [172]. Hung and Yang proposed a method to reduce computational time by identifying more accurate cluster centers [146]. FCM variants were also developed to deal with other data types, such as symbolic data [81] and data with missing values [129].

A family of fuzzyc-shells algorithms has also appeared to detect different types of cluster shapes, especially contours (lines, circles, ellipses, rings, rectangles, hyperbolas) in a two-dimensional data space. They use the “shells” (curved surfaces [70]) as the cluster prototypes instead of points or surfaces in traditional fuzzy clustering algorithms. In the case of FCS [36], [70], the proposed cluster prototype is represented as a d -dimensional hyperspherical shell $\mathbf{m}(\mathbf{v}, r)$ ($d = 2$ for circles), where $\mathbf{v} \in \mathbb{R}^d$ is the center, and $r \in \mathbb{R}$ is the radius. A distance function is defined as $D(\mathbf{x}_i, \mathbf{m}_j) = (||\mathbf{x}_i - \mathbf{v}_j|| - r_j)$ to measure the distance from a data object $\mathbf{x}_i$ to the prototype $\mathbf{m}_j$. Similarly, other cluster shapes can be achieved by defining appropriate prototypes and corresponding distance functions, example including fuzzy c -spherical shells (FCSS) [176], fuzzy c -rings (FCR) [193], fuzzy c -quadratic shells (FCQS) [174], and fuzzy c -rectangular shells (FCRS) [137]. See [141] for further details.

Fuzzy set theories can also be used to create hierarchical cluster structure. Geva proposed a hierarchical unsupervised fuzzy clustering (HUFC) algorithm [104], which can effectively explore data structure at different levels like HC, while establishing the connections between each object and cluster in the hierarchy with the memberships. This design makes HUFC overcome one of the major disadvantages of HC, i.e., HC cannot reassign an object once it is designated into a cluster. Fuzzy clustering is also closely related to neural networks [24], and we will see more discussions in the following section.

H. Neural Networks-Based Clustering

Neural networks-based clustering has been dominated by SOFMs and adaptive resonance theory (ART), both of which are reviewed here, followed by a brief discussion of other approaches.

In competitive neural networks, active neurons reinforce their neighborhood within certain regions, while suppressing the activities of other neurons (so-called on-center/off-surround competition). Typical examples include LVQ and SOFM [168], [169]. Intrinsically, LVQ performs supervised learning, and is not categorized as a clustering algorithm [169], [221]. But its learning properties provide an insight to describe the potential data structure using the prototype vectors in the competitive layer. By pointing out the limitations of LVQ, including sensitivity to initiation and lack of a definite clustering object, Pal, Bezdek, and Tsao proposed a general LVQ algorithm for clustering, known as GLVQ [221] (also see [157] for its improved version GLVQ-F). They constructed the clustering problem as an optimization process based on minimizing a loss function, which is defined on the locally weighted error between the input pattern and the winning prototype. They also showed the relations between LVQ and the online K -means

algorithm. Soft LVQ algorithms, e.g., fuzzy algorithms for LVQ (FALVQ), were discussed in [156].

The objective of SOFM is to represent high-dimensional input patterns with prototype vectors that can be visualized in a usually two-dimensional lattice structure [168], [169]. Each unit in the lattice is called a neuron, and adjacent neurons are connected to each other, which gives the clear topology of how the network fits itself to the input space. Input patterns are fully connected to all neurons via adaptable weights, and during the training process, neighboring input patterns are projected into the lattice, corresponding to adjacent neurons. In this sense, some authors prefer to think of SOFM as a method to displaying latent data structure in a visual way rather than a clustering approach [221]. Basic SOFM training goes through the following steps.

- 1) Define the topology of the SOFM; Initialize the prototype vectors $\mathbf{m}_i(0), i = 1, \dots, K$ randomly.
- 2) Present an input pattern $\mathbf{x}$ to the network; Choose the winning node J that is closest to $\mathbf{x}$, i.e., $J = \arg_j \min\{\|\mathbf{x} - \mathbf{m}_j\|\}$.
- 3) Update prototype vectors

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x} - \mathbf{m}_i(t)]$$

where $h_{ci}(t)$ is the neighborhood function that is often defined as

$$h_{ci}(t) = \alpha(t) \exp\left(\frac{-\|\mathbf{r}_c - \mathbf{r}_i\|^2}{2\sigma^2(t)}\right)$$

where $\alpha(t)$ is the monotonically decreasing learning rate, $\mathbf{r}$ represents the position of corresponding neuron, and $\sigma(t)$ is the monotonically decreasing kernel width function, or

$$h_{ci}(t) = \begin{cases} \alpha(t), & \text{if node } c \text{ belongs to the neighborhood} \\ & \text{of the winning node } J \\ 0, & \text{otherwise.} \end{cases}$$

- 4) Repeat steps 2)–3) until no change of neuron position that is more than a small positive number is observed.

While SOFM enjoy the merits of input space density approximation and independence of the order of input patterns, a number of user-dependent parameters cause problems when applied in real practice. Like the K -means algorithm, SOFM need to predefine the size of the lattice, i.e., the number of clusters, which is unknown for most circumstances. Additionally, trained SOFM may be suffering from input space density misrepresentation [132], where areas of low pattern density may be over-represented and areas of high density under-represented. Kohonen reviewed a variety of variants of SOFM in [169], which improve drawbacks of basic SOFM and broaden its applications. SOFM can also be integrated with other clustering approaches (e.g., K -means algorithm or HC) to provide more effective and faster clustering. [263] and [276] illustrate two such hybrid systems.

ART was developed by Carpenter and Grossberg, as a solution to the plasticity and stability dilemma [51], [53], [113]. ART can learn arbitrary input patterns in a stable, fast, and

self-organizing way, thus, overcoming the effect of learning instability that plagues many other competitive networks. ART is not, as is popularly imagined, a neural network architecture. It is a learning theory, that resonance in neural circuits can trigger fast learning. As such, it subsumes a large family of current and future neural networks architectures, with many variants. ART1 is the first member, which only deals with binary input patterns [51], although it can be extended to arbitrary input patterns by a variety of coding mechanisms. ART2 extends the applications to analog input patterns [52] and ART3 introduces a new mechanism originating from elaborate biological processes to achieve more efficient parallel search in hierarchical structures [54]. By incorporating two ART modules, which receive input patterns (ART_a) and corresponding labels (ART_b), respectively, with an inter-ART module, the resulting ARTMAP system can be used for supervised classifications [56]. The match tracking strategy ensures the consistency of category prediction between two ART modules by dynamically adjusting the vigilance parameter of ART_a . Also see fuzzy ARTMAP in [55]. A similar idea, omitting the inter-ART module, is known as LAPART [134].

The basic ART1 architecture consists of two-layer nodes, the feature representation field F_1 and the category representation field F_2 . They are connected by adaptive weights, bottom-up weight matrix $\mathbf{W}^{12}$ and top-down weight matrix $\mathbf{W}^{21}$. The prototypes of clusters are stored in layer F_2 . After it is activated according to the winner-takes-all competition, an expectation is reflected in layer F_1 , and compared with the input pattern. The orienting subsystem with the specified vigilance parameter ρ ($0 \leq \rho \leq 1$) determines whether the expectation and the input are closely matched, and therefore controls the generation of new clusters. It is clear that the larger ρ is, the more clusters are generated. Once weight adaptation occurs, both bottom-up and top-down weights are updated simultaneously. This is called resonance, from which the name comes. The ART1 algorithm can be described as follows.

- 1) Initialize weight matrices $\mathbf{W}^{12}$ and $\mathbf{W}^{21}$ as $W_{ij}^{12} = \alpha_j$, where α_j are sorted in a descending order and satisfies $0 < \alpha_j < (1)/(\beta + |\mathbf{x}|)$ for $\beta > 0$ and any binary input pattern $\mathbf{x}$, and $W_{ji}^{21} = 1$;
- 2) For a new pattern $\mathbf{x}$, calculate the input from layer F_1 to layer F_2 as

$$T_j = \sum_{i=1}^d W_{ij}^{12} x_i = \begin{cases} |\mathbf{x}| \alpha_j & \text{if } j \text{ is an uncommitted node} \\ & \text{(first activated)} \\ \frac{|\mathbf{x} \cap \mathbf{W}_j^{21}|}{\beta + |\mathbf{W}_j^{21}|} & \text{if } j \text{ is a committed node} \end{cases}$$

where $\cap$ represents the logic AND operation.

- 3) Activate layer F_2 by choosing node J with the winner-takes-all rule $T_J = \max_j\{T_j\}$.
- 4) Compare the expectation from layer F_2 with the input pattern. If $\rho \leq (|\mathbf{x} \cap \mathbf{W}_J^{21}|/|\mathbf{x}|)$, go to step 5a), otherwise go to step 5b).

- 5)
 - a) Update the corresponding weights for the active node as $\mathbf{W}_j^{12}(\text{new}) = (|\mathbf{x} \cap \mathbf{W}_j^{21}(\text{old})|) / (\beta + |\mathbf{W}_j^{21}(\text{old})|)$ and $\mathbf{W}_j^{21}(\text{new}) = \mathbf{x} \cap \mathbf{W}_j^{21}(\text{old})$;
 - b) Send a reset signal to disable the current active node by the orienting subsystem and return to step 3).
- 6) Present another input pattern, return to step 2) until all patterns are processed.

Note the relation between ART network and other clustering algorithms described in traditional and statistical language. Moore used several clustering algorithms to explain the clustering behaviors of ART1 and therefore induced and proved a number of important properties of ART1, notably its equivalence to varying K -means clustering [204]. She also showed how to adapt these algorithms under the ART1 framework. In [284] and [285], the ease with which ART may be used for hierarchical clustering is also discussed.

Fuzzy ART (FA) benefits the incorporation of fuzzy set theory and ART [57]. FA maintains similar operations to ART1 and uses the fuzzy set operators to replace the binary operators, so that it can work for all real data sets. FA exhibits many desirable characteristics such as fast and stable learning and atypical pattern detection. Huang *et al.* investigated and revealed more properties of FA classified as template, access, reset, and the number of learning epochs [143]. The criticisms for FA are mostly focused on its inefficiency in dealing with noise and the deficiency of hyperrectangular representation for clusters in many circumstances [23], [24], [281]. Williamson described Gaussian ART (GA) to overcome these shortcomings [281], in which each cluster is modeled with Gaussian distribution and represented as a hyperellipsoid geometrically. GA does not inherit the offline fast learning property of FA, as indicated by Anagnostopoulos *et al.* [13], who proposed different ART architectures: hypersphere ART (HA) [12] for hyperspherical clusters and ellipsoid ART (EA) [13] for hyperellipsoidal clusters, to explore a more efficient representation of clusters, while keeping important properties of FA. Baraldi and Alpaydin proposed SART following their general ART clustering networks frame, which is described through a feedforward architecture combined with a match comparison mechanism [23]. As specific examples, they illustrated symmetric fuzzy ART (SFART) and fully self-organizing SART (FOSART) networks. These networks outperform ART1 and FA according to their empirical studies [23].

In addition to these, many other neural network architectures are developed for clustering. Most of these architectures utilize prototype vectors to represent clusters, e.g., cluster detection and labeling network (CDL) [82], HEC [194], and SPLL [296]. HEC uses a two-layer network architecture to estimate the regularized Mahalanobis distance, which is equated to the Euclidean distance in a transformed whitened space. CDL is also a two-layer network with an inverse squared Euclidean metric. CDL requires the match between the input patterns and the prototypes above a threshold, which is dynamically adjusted. SPLL emphasizes initiation independent and adaptive generation of clusters. It begins with a random prototype in the input space and iteratively chooses and divides prototypes until no further split is available. The divisibility of a prototype is based on the consideration that each prototype represents only one natural

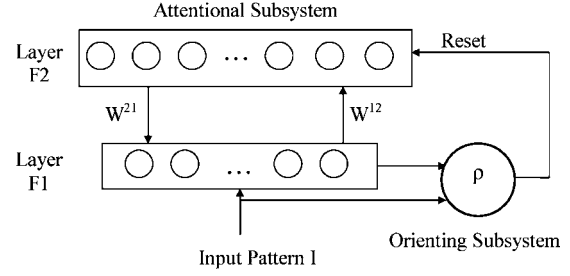


Fig. 2. ART1 architecture. Two layers are included in the attentional subsystem, connected via bottom-up and top-down adaptive weights. Their interactions are controlled by the orienting subsystem through a vigilance parameter.

cluster, instead of the combinations of several clusters. Simpson employed hyperbox fuzzy sets to characterize clusters [100], [249]. Each hyperbox is delineated by a min and max point, and data points build their relations with the hyperbox through the membership function. The learning process experiences a series of expansion and contraction operations, until all clusters are stable.

I. Kernel-Based Clustering

Kernel-based learning algorithms [209], [240], [274] are based on Cover's theorem. By nonlinearly transforming a set of complex and nonlinearly separable patterns into a higher-dimensional feature space, we can obtain the possibility to separate these patterns linearly [132]. The difficulty of curse of dimensionality can be overcome by the kernel trick, arising from Mercer's theorem [132]. By designing and calculating an inner-product kernel, we can avoid the time-consuming, sometimes even infeasible process to explicitly describe the nonlinear mapping and compute the corresponding points in the transformed space.

In [241], Schölkopf, Smola, and Müller depicted a kernel- K -means algorithm in the online mode. Suppose we have a set of patterns $\mathbf{x}_j \in \mathbb{R}^d, j = 1, \dots, N$ and a nonlinear map $\Phi: \mathbb{R}^d \rightarrow F$. Here, F represents a feature space with arbitrarily high dimensionality. The object of the algorithm is to find K centers so that we can minimize the distance between the mapped patterns and their closest center

$$\begin{aligned}
 \|\Phi(\mathbf{x}) - \mathbf{m}_l\|^2 &= \left\| \Phi(\mathbf{x}) - \sum_{j=1}^N \tau_{lj} \Phi(\mathbf{x}_j) \right\|^2 \\
 &= k(\mathbf{x}, \mathbf{x}) - 2 \sum_{j=1}^N \tau_{lj} k(\mathbf{x}, \mathbf{x}_j) + \sum_{i,j=1}^N \tau_{li} \tau_{lj} k(\mathbf{x}_i, \mathbf{x}_j)
 \end{aligned}$$

where $\mathbf{m}_l$ is the center for the l th cluster and lies in a span of $\Phi(\mathbf{x}_1), \dots, \Phi(\mathbf{x}_N)$, and $k(\mathbf{x}, \mathbf{x}_j) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_j)$ is the inner-product kernel.

Define the cluster assignment variable

$$C_{jl} = \begin{cases} 1, & \text{if } \mathbf{x}_j \text{ belongs to cluster } l \\ 0, & \text{otherwise.} \end{cases}$$

Then the kernel- K -means algorithm can be formulated as the following.

- 1) Initialize the centers $\mathbf{m}_l$ with the first i , ($i \geq K$), observation patterns;
- 2) Take a new pattern $\mathbf{x}_{i+1}$ and calculate $C_{(i+1)h}$ as shown in the equation at the bottom of the page.
- 3) Update the mean vector $\mathbf{m}_h$ whose corresponding $C_{(i+1)h}$ is 1

$$\mathbf{m}_h^{\text{new}} = \mathbf{m}_h^{\text{old}} + \xi (\Phi(\mathbf{x}_{i+1}) - \mathbf{m}_h^{\text{old}})$$

where $\xi = C_{(i+1)h} / \sum_{j=1}^{i+1} C_{jh}$.

- 4) Adapt the coefficients τ_{hj} for each $\phi(\mathbf{x}_j)$ as

$$\tau_{hj}^{\text{new}} = \begin{cases} \tau_{hj}^{\text{old}}(1 - \xi), & \text{for } j \neq i + 1 \\ \xi, & \text{for } j = i + 1 \end{cases}$$

- 5) Repeat steps 2)–4) until convergence is achieved.

Two variants of kernel- K -means were introduced in [66], motivated by SOFM and ART networks. These variants consider effects of neighborhood relations, while adjusting the cluster assignment variables, and use a vigilance parameter to control the process of producing mean vectors. The authors also illustrated the application of these approaches in case based reasoning systems.

An alternative kernel-based clustering approach is in [107]. The problem was formulated to determine an optimal partition Γ to minimize the trace of within-group scatter matrix in the feature space

$$\begin{aligned} \Gamma &= \arg \min_{\Gamma} \text{Tr}(S_W^{\Phi}) \\ &= \arg \min_{\Gamma} \text{Tr} \left\{ \frac{1}{N} \sum_{i=1}^K \sum_{j=1}^N \gamma_{ij} (\Phi(\mathbf{x}_j) - \mathbf{m}_i) \right. \\ &\quad \left. \times (\Phi(\mathbf{x}_j) - \mathbf{m}_i)^T \right\} \\ &= \arg \max_{\Gamma} \sum_{i=1}^K \xi_i R(\mathbf{x} | C_i) \end{aligned}$$

where

$$\xi_i = N_i / N, R(\mathbf{x} | C_i) = (1/N_i^2) \sum_{l=1}^N \sum_{j=1}^N \gamma_{il} \gamma_{ij} k(\mathbf{x}_l, \mathbf{x}_j),$$

and N_i is the total number of patterns in the i th cluster.

Note that the kernel function utilized in this case is the radial basis function (RBF) and $R(\mathbf{x} | C_i)$ can be interpreted as a measure of the denseness for the i th cluster.

Ben-Hur *et al.* presented a new clustering algorithm, SVC, in order to find a set of contours used as the cluster boundaries in the original data space [31], [32]. These contours can be formed by mapping back the smallest enclosing sphere in the transformed feature space. RBF is chosen in this algorithm,

and, by adjusting the width parameter of RBF, SVC can form either agglomerative or divisive hierarchical clusters. When some points are allowed to lie outside the hypersphere, SVC can deal with outliers effectively. An extension, called multiple spheres support vector clustering, was proposed in [62], which combines the concept of fuzzy membership.

Kernel-based clustering algorithms have many advantages.

- 1) It is more possible to obtain a linearly separable hyperplane in the high-dimensional, or even infinite feature space.
- 2) They can form arbitrary clustering shapes other than hyperellipsoid and hypersphere.
- 3) Kernel-based clustering algorithms, like SVC, have the capability of dealing with noise and outliers.
- 4) For SVC, there is no requirement for prior knowledge to determine the system topological structure. In [107], the kernel matrix can provide the means to estimate the number of clusters.

Meanwhile, there are also some problems requiring further consideration and investigation. Like many other algorithms, how to determine the appropriate parameters, for example, the width of Gaussian kernel, is not trivial. The problem of computational complexity may become serious for large data sets.

The process of constructing the sum-of-squared clustering algorithm [107] and K -means algorithm [241] presents a good example to reformulate more powerful nonlinear versions for many existing linear algorithms, provided that the scalar product can be obtained. Theoretically, it is important to investigate whether these nonlinear variants can keep some useful and essential properties of the original algorithms and how Mercer kernels contribute to the improvement of the algorithms. The effect of different types of kernel functions, which are rich in the literature, is also an interesting topic for further exploration.

J. Clustering Sequential Data

Sequential data are sequences with variable length and many other distinct characteristics, e.g., dynamic behaviors, time constraints, and large volume [120], [265]. Sequential data can be generated from: DNA sequencing, speech processing, text mining, medical diagnosis, stock market, customer transactions, web data mining, and robot sensor analysis, to name a few [78], [265]. In recent decades, sequential data grew explosively. For example, in genetics, the recent statistics released on October 15, 2004 (Release 144.0) shows that there are 43 194 602 655 bases from 38 941 263 sequences in GenBank database [103] and release 45.0 of SWISSPROT on October 25, 2004 contains 59 631 787 amino acids in 163 235 sequence entries [267]. Cluster analysis explores potential patterns hidden in the large number of sequential data in the context of unsupervised learning and therefore provides a crucial way to meet the current challenges. Generally, strategies for sequential clustering mostly fall into three categories.

$$C_{(i+1)h} = \begin{cases} 1, & \text{if } \|\Phi(\mathbf{x}_{i+1}) - \mathbf{m}_h\|^2 < \|\Phi(\mathbf{x}_{i+1}) - \mathbf{m}_j\|^2, \quad \forall j \neq h \\ 0, & \text{otherwise.} \end{cases}$$

1) *Sequence Similarity*: The first scheme is based on the measure of the distance (or similarity) between each pair of sequences. Then, proximity-based clustering algorithms, either hierarchical or partitional, can group sequences. Since many sequential data are expressed in an alphabetic form, like DNA or protein sequences, conventional measure methods are inappropriate. If a sequence comparison is regarded as a process of transforming a given sequence to another with a series of substitution, insertion, and deletion operations, the distance between the two sequences can be defined by virtue of the minimum number of required operations. A common analysis processes is alignment, illustrated in Fig. 3. The defined distance is known as edit distance or Levenshtein distance [120], [236]. These edit operations are weighted (punished or rewarded) according to some prior domain knowledge and the distance herein is equivalent to the minimum cost to complete the transformation. In this sense, the similarity or distance between two sequences can be reformulated as an optimal alignment problem, which fits well in the framework of dynamic programming.

Given two sequences, $\mathbf{x} = (x_1 \dots x_i \dots x_N)$ and $\mathbf{y} = (y_1 \dots y_j \dots y_M)$, the basic dynamic programming-based sequence alignment algorithm, also known as the Needleman-Wunsch algorithm, can be depicted by the following recursive equation [78], [212]:

$$S(i, j) = \max \begin{cases} S(i-1, j-1) + e(x_i, y_j) \\ S(i-1, j) + e(x_i, \phi) \\ S(i, j-1) + e(\phi, y_j) \end{cases}$$

where $S(i, j)$ is defined as the best alignment score between sequence segment $(x_1 \dots x_i), (1 \leq i \leq N)$ of $\mathbf{x}$ and $(y_1 \dots y_j), (1 \leq j \leq M)$ of $\mathbf{y}$, and $e(x_i, y_j), e(x_i, \phi)$, or $e(\phi, y_j)$ represent the cost for aligning x_i to y_j , aligning x_i to a gap (denoted as ϕ), or aligning y_j to a gap, respectively. The computational results for each position at i and j are recorded in an array with a pointer that stores current optimal operations and provides an effective path in backtracking the alignment. The Needleman-Wunsch algorithm considers the comparison of the whole length of two sequences and therefore performs a global optimal alignment. However, it is also important to find local similarity among sequences in many circumstances. The Smith-Waterman algorithm achieves that by allowing the beginning of a new alignment during the recursive computation, and the stop of an alignment anywhere in the dynamic programming matrix [78], [251]. This change is summarized in the following:

$$S(i, j) = \min \begin{cases} S(i-1, j-1) + e(x_i, y_j) \\ S(i-1, j) + e(x_i, \phi) \\ S(i, j-1) + e(\phi, y_j) \\ 0. \end{cases}$$

For both the global and local alignment algorithms, the computation complexity is $O(NM)$, which is very expensive, especially for a clustering problem that requires an all-against-all pairwise comparison. A wealth of speed-up methods has been developed to improve the situation [78], [120]. We will see more discussion in Section III-E in the context of biological sequences analysis. Other examples include applications for speech recognition [236] and navigation pattern mining [131].

```

C L U S - - - - - T E R I - N G
M M S M I I I I I M D D M I M D
C L A S S I F I C A T - - I O N -
M: Match; D: Deletion;
I: Insertion; S: Substitution

```

Fig. 3. Illustration of a sequence alignment. Series of edit operations is performed to change the sequence CLUSTERING into the sequence CLASSIFICATION.

2) *Indirect Sequence Clustering*: The second approach employs an indirect strategy, which begins with the extraction of a set of features from the sequences. All the sequences are then mapped into the transformed feature space, where classical vector space-based clustering algorithms can be used to form clusters. Obviously, feature extraction becomes the essential factor that decides the effectiveness of these algorithms. Guralnik and Karypis discussed the potential dependency between two sequential patterns and suggested both the global and the local approaches to prune the initial feature sets in order to better represent sequences in the new feature space [119]. Morzy *et al.* utilized the sequential patterns as the basic element in the agglomerative hierarchical clustering and defined a co-occurrence measure, as the standard of fusion of smaller clusters [207]. These methods greatly reduce the computational complexities and can be applied to large-scale sequence databases. However, the process of feature selection inevitably causes the loss of some information in the original sequences and needs extra attention.

3) *Statistical Sequence Clustering*: Typically, the first two approaches are used to deal with sequential data composed of alphabets, while the third paradigm, which aims to construct statistical models to describe the dynamics of each group of sequences, can be applied to numerical or categorical sequences. The most important method is hidden Markov models (HMMs) [214], [219], [253], which first gained its popularity in the application of speech recognition [229]. A discrete HMM describes an unobservable stochastic process consisting of a set of states, each of which is related to another stochastic process that emits observable symbols. Therefore, the HMM is completely specified by the following.

- 1) A finite set $V = \{V_1, \dots, V_Q\}$ with Q states.
- 2) A discrete set $O = \{O_1, \dots, O_M\}$ with M observation symbols.
- 3) A state transition distribution $A = \{\alpha_{ij}\}$, where

$$\alpha_{ij} = P(j\text{th state at time } t+1 \mid i\text{th state at time } t).$$

- 4) A symbol emission distribution $B = \{\beta_{it}\}$, where

$$\beta_{it} = P[V_i \text{ at } t \mid i\text{th state at } t].$$

- 5) An initial state distribution $\pi = \{\pi_i\}$, where

$$\pi_i = P[i\text{th state at } t=1].$$

After an initial state is selected according to the initial distribution π , a symbol is emitted with emission distribution E . The next state is decided by the state transition distribution T and it also generates a symbol based on E . The process repeats until reaching the last state. Note that the procedure generates

a sequence of symbol observations instead of states, which is where the name “hidden” comes from. HMMs are well founded theoretically [229]. Dynamic programming and EM algorithm are developed to solve the three basic problems of HMMs as the following.

- 1) *Likelihood* (forward or backward algorithm). Compute the probability of an observation sequence given a model.
- 2) *State interpretation* (Viterbi algorithm). Find an optimal state sequence by optimizing some criterion function given the observation sequence and the model.
- 3) *Parameter estimation* (Baum–Welch algorithm). Design suitable model parameters to maximize the probability of observation sequence under the model.

The equivalence between an HMM and a recurrent back-propagation network was elucidated in [148], and a universal framework was constructed to describe both the computational and the structural properties of the HMM and the neural network.

Smyth proposed an HMM-based clustering model, which, similar to the theories introduced in mixture densities-based clustering, assumes that each cluster is generated based on some probability distribution [253]. Here, HMMs are used rather than the common Gaussian or t -distribution. In addition to the form of finite mixture densities, the mixture model can also be described by means of an HMM with the transition matrix

$$A = \begin{pmatrix} A_1 & & & 0 \\ & A_2 & & \\ & & \dots & \\ 0 & & & A_K \end{pmatrix}$$

where $A_i, i = 1, \dots, K$ is the transition distribution for the i th cluster. The initial distribution of the HMM is determined based on the prior probability for each cluster. The basic learning process starts with a parameter initialization scheme to form a rough partition with the log-likelihood of each sequence serving as the distance measure. The partition is further refined by training the overall HMM over all sequences with the classical EM algorithm. A Monte-Carlo cross validation method was used to estimate the possible number of clusters. An application with a modified HMM model that considers the effect of context for clustering facial display sequences is illustrated in [138]. Oates *et al.* addressed the initial problem by regrouping the sequences with the agglomerative hierarchical clustering, which operates on the proximity matrix determined by the dynamic time warping (DTW) technique [214]. The area formed between one original sequence and a new sequence, generated by warping the time dimension of another original sequence, reflects the similarity of the two sequences. Li and Biswas suggested several objective criterion functions based on posterior probability and information theory for structural selection of HMMs and cluster validity [182]. More recent advances on HMMs and other related topics are reviewed in [30].

Other model-based sequence clustering includes mixtures of first-order Markov chain [255] and a linear model like autoregressive moving average (ARMA) model [286]. Usually, they

are combined with EM for parameter estimation [286]. Smyth [255] and Cadez *et al.* [50] further generalize a universal probabilistic framework to model mixed data measurement, which includes both conventional static multivariate vectors and dynamic sequence data.

The paradigm models clusters directly from original data without additional process that may cause information loss. They provide more intuitive ways to capture the dynamics of data and more flexible means to deal with variable length sequences. However, determining the number of model components remains a complicated and uncertain process [214], [253]. Also, the model selected is required to have sufficient complexity, in order to interpret the characteristics of data.

K. Clustering Large-Scale Data Sets

Scalability becomes more and more important for clustering algorithms with the increasing complexity of data, mainly manifesting in two aspects: enormous data volume and high dimensionality. Examples, illustrated in the sequential clustering section, are just some of the many applications that require this capability. With the further advances of database and Internet technologies, clustering algorithms will face more severe challenges in handling the rapid growth of data. We summarize the computational complexity of some typical and classical clustering algorithms in Table II with several newly proposed approaches specifically designed to deal with large-scale data sets. Several points can be generalized through the table.

- 1) Obviously, classical hierarchical clustering algorithms, including single-linkage, complete linkage, average linkage, centroid linkage and median linkage, are not appropriate for large-scale data sets due to the quadratic computational complexities in both execution time and store space.
- 2) K -means algorithm has a time complexity of $O(NKd)$ and space complexity of $O(N + K)$. Since N is usually much larger than both K and d , the complexity becomes near linear to the number of samples in the data sets. K -means algorithm is effective in clustering large-scale data sets, and efforts have been made in order to overcome its disadvantages [142], [218].
- 3) Many novel algorithms have been developed to cluster large-scale data sets, especially in the context of data mining [44], [45], [85], [135], [213], [248]. Many of them can scale the computational complexity linearly to the input size and demonstrate the possibility of handling very large data sets.
 - a) Random sampling approach, e.g., CLARA clustering large applications (CLARA) [161] and CURE [116]. The key point lies that the appropriate sample sizes can effectively maintain the important geometrical properties of clusters. Furthermore, Chernoff bounds can provide estimation for the lower bound of the minimum sample size, given the low probability that points in each cluster are missed in the sample set [116]. CLARA represents each cluster with a

medoid while CURE chooses a set of well-scattered and center-shrunk points.

- b) Randomized search approach, e.g., clustering large applications based on randomized search (CLARANS) [213]. CLARANS sees the clustering as a search process in a graph, in which each node corresponds to a set of K medoids. It begins with an arbitrary node as the current node and examines a set of neighbors, defined as the node consisting of only one different data object, to seek a better solution, i.e., any neighbor, with a lower cost, becomes the current node. If the maximum number of neighbors, specified by the user, has been reached, the current node is accepted as a winning node. This process iterates several times as specified by users. Though CLARANS achieves better performance than algorithms like CLARA, the total computational time is still quadratic, which makes CLARANS not quite effective in very large data sets.
 - c) Condensation-based approach, e.g., BIRCH [295]. BIRCH generates and stores the compact summaries of the original data in a CF tree, as discussed in Section II-B. This new data structure efficiently captures the clustering information and largely reduces the computational burden. BIRCH was generalized into a broader framework in [101] with two algorithms realization, named as BUBBLE and BUBBLE-FM.
 - d) Density-based approach, e.g., density based spatial clustering of applications with noise (DBSCAN) [85] and density-based clustering (DENCLUE) [135]. DBSCAN requires that the density in a neighborhood for an object should be high enough if it belongs to a cluster. DBSCAN creates a new cluster from a data object by absorbing all objects in its neighborhood. The neighborhood needs to satisfy a user-specified density threshold. DBSCAN uses a R^* -tree structure for more efficient queries. DENCLUE seeks clusters with local maxima of the overall density function, which reflects the comprehensive influence of data objects to their neighborhoods in the corresponding data space.
 - e) Grid-based approach, e.g., WaveCluster [248] and fractal clustering (FC) [26]. WaveCluster assigns data objects to a set of units divided in the original feature space, and employs wavelet transforms on these units, to map objects into the frequency domain. The key idea is that clusters can be easily distinguished in the transformed space. FC combines the concepts of both incremental clustering and fractal dimension. Data objects are incrementally added to the clusters, specified through an initial process, and represented as cells in a grid, with the condition that the fractal dimension of cluster needs to keep relatively stable.
- 4) Most algorithms listed previously lack the capability of dealing with data with high dimensionality. Their performances degenerate with the increase of dimensionality. Some algorithms, like FC and DENCLUE, have

shown some successful applications in such cases, but these are still far from completely effective.

In addition to the aforementioned approaches, several other techniques also play significant roles in clustering large-scale data sets. Parallel algorithms can more effectively use computational resources, and greatly improve overall performance in the context of both time and space complexity [69], [217], [262]. Incremental clustering techniques do not require the storage of the entire data set, and can handle it in a one-pattern-at-a-time way. If the pattern displays enough closeness to a cluster according to some predefined criteria, it is assigned to the cluster. Otherwise, a new cluster is created to represent the object. A typical example is the ART family [51]–[53] discussed in Section II-H. Most incremental clustering algorithms are dependent on the order of the input patterns [51], [204]. Bradley, Fayyad, and Reina proposed a scalable clustering framework, considering seven relevant important characteristics in dealing with large databases [44]. Applications of the framework were illustrated for the K -means algorithm and EM mixture models [44], [45].

L. Exploratory Data Visualization and High-Dimensional Data Analysis Through Dimensionality Reduction

For most of the algorithms summarized in Table II, although they can deal with large-scale data, they are not sufficient for analyzing high-dimensional data. The term, “curse of dimensionality,” which was first used by Bellman to indicate the exponential growth of complexity in the case of multivariate function estimation under a high dimensionality situation [28], is generally used to describe the problems accompanying high dimensional spaces [34], [132]. It is theoretically proved that the distance between the nearest points is no different from that of other points when the dimensionality of the space is high enough [34]. Therefore, clustering algorithms that are based on the distance measure may no longer be effective in a high dimensional space. Fortunately, in practice, many high-dimensional data usually have an intrinsic dimensionality that is much lower than the original dimension [60]. Dimension reduction is important in cluster analysis, which not only makes the high-dimensional data addressable and reduces the computational cost, but provides users with a clearer picture and visual examination of the data of interest. However, dimensionality reduction methods inevitably cause some loss of information, and may damage the interpretability of the results, even distort the real clusters.

One natural strategy for dimensionality reduction is to extract important components from original data, which can contribute to the division of clusters. Principle component analysis (PCA) or Karhunen-Loève transformation is one of the typical approaches, which is concerned with constructing a linear combination of a set of vectors that can best describe the variance of data. Given the $N \times D$ input pattern matrix $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_j, \dots, \mathbf{x}_N]^T$, the linear mapping $\mathbf{Y} = \mathbf{X}\mathbf{V}$ projects $\mathbf{x}_i$ ($1 \leq i \leq N$) into a low-dimensional ($L(L < D)$) subspace, where $\mathbf{Y}$ is the resulting $N \times L$ matrix and $\mathbf{V}$ is the $D \times L$ projection matrix whose columns are the eigenvectors that correspond to the L largest eigenvalues of the $D \times D$ covariance matrix $\mathbf{\Sigma}$, calculated from the whole data set (hence, the column vectors of $\mathbf{V}$ are orthonormal). PCA estimates the matrix $\mathbf{V}$ while minimizing the sum of squares of the error

of approximating the input vectors. In this sense, PCA can be realized through a three-layer neural network, called an auto-associative multilayer perceptron, with linear activation functions [19], [215]. In order to extract more complicated nonlinear data structure, nonlinear PCA was developed and one of the typical examples is kernel PCA. As methods discussed in Section II-I, kernel PCA first maps the input patterns into a feature space. The similar steps are then applied to solve the eigenvalue problem with the new covariance matrix in the feature space. In another way, extra hidden layers with nonlinear activation functions can be added into the auto-associative network for this purpose [38], [75].

PCA is appropriate for Gaussian distributions since it relies on second-order relationships in the covariance matrix. Other linear transforms, like independent component analysis (ICA) and projection pursuit, which use higher order statistical information, are more suited for non-Gaussian distributions [60], [151]. The basic goal of ICA is to find the components that are most statistically independent from each other [149], [154]. In the context of blind source separation, ICA aims to separate the independent source signals from the mixed observation signal. This problem can be formulated in several different ways [149], and one of the simplest form (without considering noise) is represented as $\mathbf{x} = \mathbf{A}\mathbf{s}$, where $\mathbf{x}$ is the N -dimensional observable vector, $\mathbf{s}$ is the L -dimensional source vector assumed to be statistically independent, and $\mathbf{A}$ is a nonsingular $N \times L$ mixing matrix. ICA can also be realized by virtue of multilayer perceptrons, and [158] illustrates one of such examples. The proposed ICA network includes whitening, separation, and basis vectors estimation layers, with corresponding learning algorithms. The authors also indicated its connection to the auto-associative multilayer perceptron. Projection pursuit is another statistical technique for seeking low-dimensional projection structures for multivariate data [97], [144]. Generally, projection pursuit regards the normal distribution as the least interesting projections and optimizes some certain indices that measure the degree of nonnormality [97]. PCA can be considered as a special example of projection pursuit, as indicated in [60]. More discussions on the relations among PCA, ICA, projection pursuit, and other relevant techniques are offered in [149] and [158].

Different from PCA, ICA, and projection pursuit, Multidimensional scaling (MDS) is a nonlinear projection technique [75], [292]. The basic idea of MDS lies in fitting original multivariate data into a low-dimensional structure while aiming to maintain the proximity information. The distortion is measured through some criterion functions, e.g., in the sense of sum of squared error between the real distance and the projection distance. The isometric feature mapping (Isomap) algorithm is another nonlinear technique, based on MDS [270]. Isomap estimates the geodesic distance between a pair of points, which is the shortest path between the points on a manifold, by virtue of the measured input-space distances, e.g., the Euclidean distance usually used. This extends the capability of MDS to explore more complex nonlinear structures in the data. Locally linear embedding (LLE) algorithm addresses the nonlinear dimensionality reduction problem from a different starting point [235]. LLE emphasizes the local linearity of the manifold and assumes that the local relations in the original data space

(D -dimensional) are also preserved in the projected low-dimensional space (L -dimensional). This is represented through a weight matrix, describing how each point is related to the reconstruction of another data point. Therefore, the procedure for dimensional reduction can be constructed as the problem that finding L -dimensional vectors $\mathbf{y}_i$ so that the criterion function $\sum_i |\mathbf{y}_i - \sum_j w_{ij} \mathbf{y}_j|$ is minimized. Another interesting nonlinear dimensionality reduction approach, known as Laplace eigenmap algorithm, is presented in [27].

As discussed in Section II-H, SOFM also provide good visualization for high-dimensional input patterns [168]. SOFM map input patterns into a one or usually two dimensional lattice structure, consisting of nodes associated with different clusters. An application for clustering of a large set of documental data is illustrated in [170], in which 6 840 568 patent abstracts were projected onto a SOFM with 1 002 240 nodes.

Subspace-based clustering addresses the challenge by exploring the relations of data objects under different combinations of features. clustering in quest (CLIQUE) [3] employs a bottom-up scheme to seek dense rectangular cells in all subspaces with high density of points. Clusters are generated as the connected components in a graph whose vertices stand for the dense units. The resulting minimal description of the clusters is obtained through the merge of these rectangles. OptiGrid [136] is designed to obtain an optimal grid-partitioning. This is achieved by constructing the best cutting hyperplanes through a set of projections. The time complexity for OptiGrid is in the interval of $O(Nd)$ and $O(Nd \log N)$. ORCLUS (arbitrarily ORiented projected CLuster generation) [2] defines a generalized projected cluster as a densely distributed subset of data objects in a subspace, along with a subset of vectors that represent the subspace. The dimensionality of the subspace is prespecified by users as an input parameter, and several strategies are proposed in guidance of its selection. The algorithm begins with a set of randomly selected K_0 seeds with the full dimensionality. This dimensionality and the number of clusters are decayed according to some factors at each iteration, until the number of clusters reaches the predefined values. Each repetition consists of three basic operations, known as assignment, vector finding, and merge. ORCLUS has the overall time complexity of $O(K_0^3 + K_0 Nd + K_0^2 d^3)$ and space complexity of $O(K_0 d^2)$. Obviously, the scalability to large data sets relies on the number of initial seeds K_0 . A generalized subspace clustering model, pCluster was proposed in [279]. These pClusters are formed by a depth-first clustering algorithm. Several other interesting applications, including a Clindex (CLustering for INDEXing) scheme and wavelet transform, are shown in [184] and [211], respectively.

M. How Many Clusters?

The clustering process partitions data into an appropriate number of subsets. Although for some applications, users can determine the number of clusters, K , in terms of their expertise, under more circumstances, the value of K is unknown and needs to be estimated exclusively from the data themselves. Many clustering algorithms ask K to be provided as an input parameter, and it is obvious that the quality of resulting clusters is largely dependent on the estimation of K . A division with

too many clusters complicates the result, therefore, makes it hard to interpret and analyze, while a division with too few clusters causes the loss of information and misleads the final decision. Dubes called the problem of determining the number of clusters “the fundamental problem of cluster validity” [74].

A large number of attempts have been made to estimate the appropriate K and some of representative examples are illustrated in the following.

- 1) *Visualization of the data set.* For the data points that can be effectively projected onto a two-dimensional Euclidean space, which are commonly depicted with a histogram or scatterplot, direct observations can provide good insight on the value of K . However, the complexity of most real data sets restricts the effectiveness of the strategy only to a small scope of applications.
- 2) *Construction of certain indices (or stopping rules).* These indices usually emphasize the compactness of intra-cluster and isolation of inter-cluster and consider the comprehensive effects of several factors, including the defined squared error, the geometric or statistical properties of the data, the number of patterns, the dissimilarity (or similarity), and the number of clusters. Milligan and Cooper compared and ranked 30 indices according to their performance over a series of artificial data sets [202]. Among these indices, the Caliński and Harabasz index [74] achieve the best performance and can be represented as

$$CH(K) = \frac{\text{Tr}(S_B)}{K-1} \bigg/ \frac{\text{Tr}(S_W)}{N-K}$$

where N is the total number of patterns and $\text{Tr}(S_B)$ and $\text{Tr}(S_W)$ are the trace of the between and within class scatter matrix, respectively. The K that maximizes the value of $CH(K)$ is selected as the optimal. It is worth noting that these indices may be data dependent. The good performance of an index for certain data does not guarantee the same behavior with different data. As pointed out by Everitt, Landau, and Leese, “it is advisable not to depend on a single rule for selecting the number of groups, but to synthesize the results of several techniques” [88].

- 3) *Optimization of some criterion functions under probabilistic mixture-model framework.* In a statistical framework, finding the correct number of clusters (components) K , is equivalent to fitting a model with observed data and optimizing some criterion [197]. Usually, the EM algorithm is used to estimate the model parameters for a given K , which goes through a predefined range of values. The value of K that maximizes (or minimizes) the defined criterion is regarded as optimal. Smyth presented a Monte-Carlo cross-validation method, which randomly divides data into training and test sets M times according to a certain fraction β ($\beta = 0.5$ works well from the empirical results) [252]. The K is selected either directly based on the criterion function or some posterior probabilities calculated.

A large number of criteria, which combine concepts from information theory, have been proposed in the literature. Typical examples include,

- Akaike’s information criterion (AIC) [4], [282]

$$\text{AIC}(K) = \frac{-2(N-1-N_k-K/2)l(\hat{\theta})}{N} + 3N_p$$

where N is the total number of patterns, N_k is the number of parameters for each cluster, N_p is the total number of parameters estimated, and $l(\hat{\theta})$ is the maximum log-likelihood. K is selected with the minimum value of $\text{AIC}(K)$.

- Bayesian inference criterion (BIC) [226], [242]

$$\text{BIC}(K) = l(\hat{\theta}) - (N_p/2) \log(N).$$

K is selected with the maximum value of $\text{BIC}(K)$.

More criteria, such as minimum description length (MDL) [114], [233], minimum message length (MML) [114], [216], cross validation-based information criterion (CVIC) [254] and covariance inflation criterion (CIC) [272], with their characteristics, are summarized in [197]. Like the previous discussion for validation index, there is no criterion that is superior to others in general case. The selection of different criteria is still dependent on the data at hand.

- 4) *Other heuristic approaches based on a variety of techniques and theories.* Girolami performed eigenvalue decomposition on the kernel matrix in the high-dimensional feature space and used the dominant K components in the decomposition summation as an indication of the possible existence of K clusters [107]. Kothari and Pitts described a scale-based method, in which the distance from a cluster centroid to other clusters in its neighborhood is considered (added as a regularization term in the original squared error criterion, Section II-C) [160]. The neighborhood of clusters work as a scale parameter and the K that is persistent in the largest interval of the neighborhood parameter is regarded as the optimal.

Besides the previous methods, constructive clustering algorithms can adaptively and dynamically adjust the number of clusters rather than use a prespecified and fixed number. ART networks generate a new cluster, only when the match between the input pattern and the expectation is below some prespecified confidence value [51]. A functionally similar mechanism is used in the CDL network [82]. The robust competitive clustering algorithm (RCA) describes a competitive agglomeration process that progresses in stages, and clusters that lose in the competition are discarded, and absorbed into other clusters [98]. This process is generalized in [42], which attains the number of clusters by balancing the effect between the complexity and the fidelity. Another learning scheme, SPLN iteratively divides cluster prototypes from a single prototype until no more prototypes satisfy the split criterion [296]. Several other constructive clustering algorithms, including the FACS and plastic neural gas, can be accessed in [223] and [232], respectively. Obviously, the problem of determining the number

of clusters is converted into a parameter selection problem, and the resulting number of clusters is largely dependent on parameter tweaking.

III. APPLICATIONS

We illustrate applications of clustering techniques in three aspects. The first is for two classical benchmark data sets that are widely used in pattern recognition and machine learning. Then, we show an application of clustering for the traveling salesman problem. The last topic is on bioinformatics. We deal with classical benchmarks in Sections III-A and III-B and the traveling salesman problem in Section III-C. A more extensive discussion of bioinformatics is in Sections III-D and III-E.

A. Benchmark Data Sets—IRIS

The iris data set [92] is one of the most popular data sets to examine the performance of novel methods in pattern recognition and machine learning. It can be downloaded from the UCI Machine Learning Repository at <http://www.ics.uci.edu/~mllearn/MLRepository.html>. There are three categories in the data set (i.e., iris setosa, iris versicolor and iris virginical), each having 50 patterns with four features [i.e., sepal length (SL), sepal width (SW), petal length (PL), and petal width (PW)]. Iris setosa can be linearly separated from iris versicolor and iris virginical, while iris versicolor and iris virginical are not linearly separable (see Fig. 4(a), in which only three features are used). Fig. 4(b) depicts the clustering result with a standard K -means algorithm. It is clear to see that K -means can correctly differentiate iris setosa from the other two iris plants. But for iris versicolor and virginical, there exist 16 misclassifications. This result is similar to those (around 15 errors) obtained from other classical clustering algorithms [221]. Table III summarizes some of the clustering results reported in the literature. From the table, we can see that many newly developed approaches can greatly improve the clustering performance on iris data set (around 5 misclassifications); some even can achieve 100% accuracy. Therefore, the data can be well classified with appropriate methods.

B. Benchmark Data Sets—MUSHROOM

Unlike the iris data set, all of the features of the mushroom data set, which can also be accessible at the UCI Machine Learning Repository, are nominal rather than numerical. These 23 species of gilled mushrooms are categorized as either edible or poisonous. The total number of instances is 8 124 with 4 208 being edible and 3 916 poisonous. The 22 features are summarized in Table IV with corresponding possible values. Table V illustrates some experimental results in the literature. As indicated in [117] and [277], traditional clustering strategies, like K -means and hierarchical clustering, work poorly on the data set. The accuracy for K -means is just around 69% [277] and the clusters formed by classical HC are mixed with nearly similar proportion of both edible and poisonous objects [117]. The results reported in the newly developed algorithms, which are specifically used for tackling categorical or mixture data, greatly improve the situation [117], [183]. The algorithm ROCK

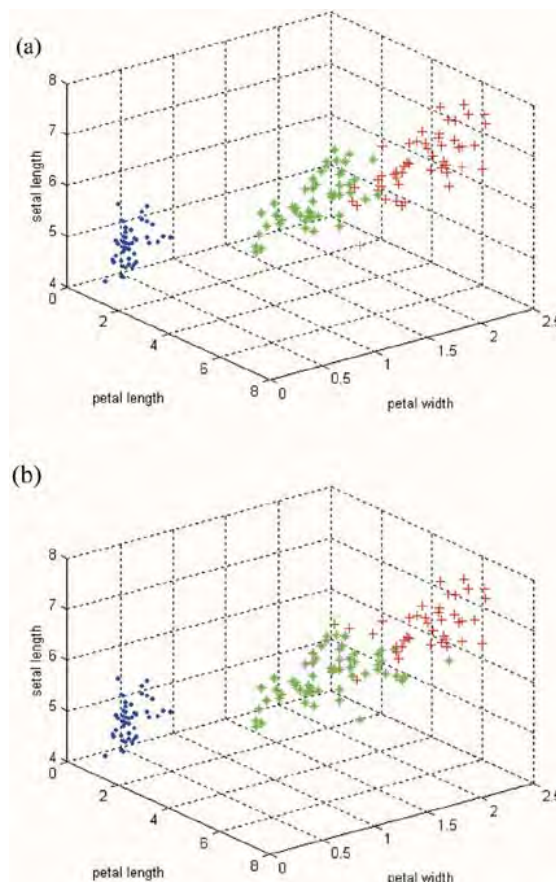


Fig. 4. (a) Iris data sets. There are three iris categories, each having 50 samples with 4 features. Here, only three features are used: PL, PW, and SL. (b) K -means clustering result with 16 classification errors observed.

TABLE III
SOME CLUSTERING RESULTS FOR THE IRIS DATA SET

Algorithm	Number of errors	Percentage of errors
GLVQ [221]	17	11.3%
FCM [129]	16	10.6%
GFMM [100]	0~7	0~4.7%
Mercer kernel based clustering algorithm [107]	3	2%
SVC [31]	4	2.7%
CDL [82]	6	4%
HC [203]	13~17	8.7~11.3%
RHC [203]	5~6	3.3~4%
FA [23]	6.77~46.4	4.5~30.9%

GLVQ: general learning vector quantization;
GFMM: general fuzzy min-max neural network;
SVC: support vector clustering; FCM: fuzzy c-means;
CDL: cluster detection and labeling network;
HC: hierarchical clustering; RHC: relative hierarchical clustering;
FA: fuzzy adaptive resonance theory.

divides objects into 21 clusters with most of them (except one) consisting of only one category, which increases the accuracy almost to 99%. The algorithm SBAC works on a subset of 200 randomly selected objects, 100 for each category and the general results show the correct partition of 3 clusters (two for edible mushrooms, one for poisonous ones). In both studies, the

TABLE IV
FEATURES FOR THE MUSHROOM DATA SET

Features	Potential values
Cap-shape	bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
Cap-surface	fibrous=f, grooves=g, scaly=y, smooth=s
Cap-color	brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y
bruises	bruises=t, no=f
odor	almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
Gill-attachment	attached=a, descending=d, free=f, notched=n
Gill-spacing	close=c, crowded=w, distant=d
Gill-size	broad=b, narrow=n
Gill-color	black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
Stalk-shape	enlarging=e, tapering=t
Stalk-root	bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
Stalk-surface-above-ring	fibrous=f, scaly=y, silky=k, smooth=s
Stalk-surface-below-ring	fibrous=f, scaly=y, silky=k, smooth=s
Stalk-color-above-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
Stalk-color-below-ring	brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
Veil-type	partial=p, universal=u
Veil-color	brown=n, orange=o, white=w, yellow=y
Ring-number	none=n, one=o, two=t
Ring-type	cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
Spore-print-color	black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
population	abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
habitat	grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

constitution of each feature for generated clusters is also illustrated and it is observed that some features, like cap-shape and ring-type, represent themselves identically for both categories and, thus, suggest poor performance of traditional approaches. Meanwhile, feature odor shows good discrimination for the different types of mushrooms. Usually, value almond, anise, or none indicates the edibility of mushrooms, while value pungent, foul, or fishy means the high possibility of presence of poisonous contents in the mushrooms.

C. Traveling Salesman Problem

The traveling salesman problem (TSP) is one of the most studied examples in an important class of problems known as NP-complete problems. Given a complete undirected graph $G = (V, E)$, where V is a set of vertices and E is a set of edges each relating two vertices with an associated nonnegative integer cost, the most general form of the TSP is equivalent to finding any Hamiltonian cycle, which is a tour over G that begins and ends at the same vertex and visits other vertices exactly once. The more common form of the problem is the

TABLE V
SOME CLUSTERING RESULTS FOR THE MUSHROOM DATA SET

Algorithm	Percentage of accuracy
K -means [277] *	69%
COP- K -means [277] +	96%
Hierarchical clustering [117]	Fail to discriminate the two types of mushroom
ROCK [117]	About 99%
SBAC [183] +	Correctly classify instances into 3 groups
ARTMAP [56]	Over 99% with 1000 training samples
AutoClass [183] +	Correctly classify instances into 4 groups

* A subset with 50 instances and 21 features was used.

+ A subset with 200 instances was used.

COP: constraint-partitioning;

ROCK: robust clustering using links;

SBAC: similarity-based agglomerative clustering.

optimization problem of trying to find the shortest Hamiltonian cycle, and in particular, the most common is the Euclidean version, where the vertices and edges all lie in the plane. Mulder and Wunsch applied a divide-and-conquer clustering technique, with ART networks, to scale the problem to a million cities [208]. The divide and conquer paradigm gives the flexibility to hierarchically break large problems into arbitrarily small clusters depending on what tradeoff between accuracy and speed is desired. In addition, the subproblems provide an excellent opportunity to take advantage of parallel systems for further optimization. As the first stage of the algorithm, the ART network is used to sort the cities into clusters. The vigilance parameter is used to set a maximum distance from the current pattern. A vigilance parameter between 0 and 1 is used as a percentage of the global space to determine the vigilance distance. Values were chosen based on the desired number and size of individual clusters. The clusters were then each passed to a version of the Lin-Kernighan (LK) algorithm [187]. The last step combines the subtours back into one complete tour. Tours with good quality for city levels up to 1 000 000 were obtained within 25 minutes on a 2 GHz AMD Athlon MP processor with 512 M of DDR RAM. Fig. 5 shows the visualizing results for 1 000, 10 000, and 1 000 000 cities, respectively.

It is worthwhile to emphasize the relation between the TSP and very large-scale integrated (VLSI) circuit clustering, which partitions a sophisticated system into smaller and simpler subcircuits to facilitate the circuit design. The object of the partitions is to minimize the number of connections among the components. One strategy for solving the problem is based on geometric representations, either linear or multidimensional [8]. Alpert and Kahng considered a solution to the problem as the “inverse” of the divide-and-conquer TSP method and used a linear tour of the modules to form the subcircuit partitions [7]. They adopted the spacefilling curve heuristic for the TSP to construct the tour so that connected modules are still close in the generated tour. A dynamic programming method was used to generate the resulting partitions. More detailed discussion on VLSI circuit clustering can be found in the survey by Alpert and Kahng [7].

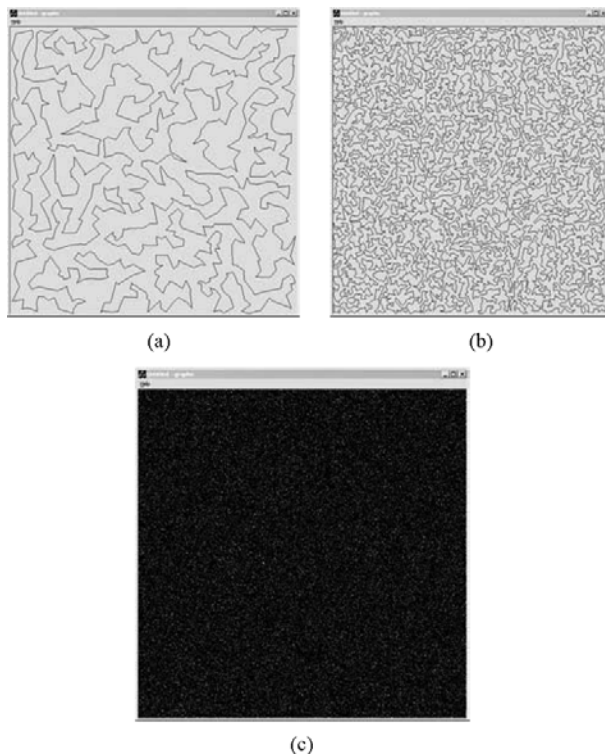


Fig. 5. Clustering divide-and-conquer TSP resulting tours for (a) 1 k, (b) 10 k, (c) 1 M cities. The clustered LK algorithm achieves a significant speedup and shows good scalability.

D. Bioinformatics—Gene Expression Data

Recently, advances in genome sequencing projects and DNA microarray technologies have been achieved. The first draft of the human genome sequence project was completed in 2001, several years earlier than expected [65], [275]. The genomic sequence data for other organisms (e.g., *Drosophila melanogaster* and *Escherichia coli*) are also abundant. DNA microarray technologies provide an effective and efficient way to measure gene expression levels of thousands of genes simultaneously under different conditions and tissues, which makes it possible to investigate gene activities from the angle of the whole genome [79], [188]. With sequences and gene expression data in hand, to investigate the functions of genes and identify their roles in the genetic process become increasingly important. Analyzes under traditional laboratory techniques are time-consuming and expensive. They fall far behind the explosively increasing generation of new data. Among the large number of computational methods used to accelerate the exploration of life science, clustering can reveal the hidden structures of biological data, and is particularly useful for helping biologists investigate and understand the activities of uncharacterized genes and proteins and further, the systematic architecture of the whole genetic network. We demonstrate the applications of clustering algorithms in bioinformatics from two aspects. The first part is based on the analysis of gene expression data generated from DNA microarray technologies. The second part describes clustering processes that directly work on linear DNA or protein sequences. The assumption is that functionally similar genes or proteins usually share similar patterns or primary sequence structures.

DNA microarray technologies generate many gene expression profiles. Currently, there are two major microarray technologies based on the nature of the attached DNA: cDNA with length varying from several hundred to thousand bases, or oligonucleotides containing 20–30 bases. For cDNA technologies, a DNA microarray consists of a solid substrate to which a large amount of cDNA clones are attached according to a certain order [79]. Fluorescently labeled cDNA, obtained from RNA samples of interest through the process of reverse transcription, is hybridized with the array. A reference sample with a different fluorescent label is also needed for comparison. Image analysis techniques are then used to measure the fluorescence of each dye, and the ratio reflects relative levels of gene expression. For a high-density oligonucleotide microarray, oligonucleotides are fixed on a chip through photolithography or solid-phase DNA synthesis [188]. In this case, absolute gene expression levels are obtained. After the normalization of the fluorescence intensities, the gene expression profiles are represented as a matrix $E = (e_{ij})$, where e_{ij} is the expression level of the i th gene in the j th condition, tissue, or experimental stage. Gene expression data analysis consists of a three-level framework based on the complexity, ranging from the investigation of single gene activities to the inference of the entire genetic network [20]. The intermediate level explores the relations and interactions between genes under different conditions, and attracts more attention currently. Generally, cluster analysis of gene expression data is composed of two aspects: clustering genes [80], [206], [260], [268], [283], [288] or clustering tissues or experiments [5], [109], [238].

Results of gene clustering may suggest that genes in the same group have similar functions, or they share the same transcriptional regulation mechanism. Cluster analysis, for grouping functionally similar genes, gradually became popular after the successful application of the average linkage hierarchical clustering algorithm for the expression data of budding yeast *Saccharomyces cerevisiae* and reaction of human fibroblasts to serum by Eisen *et al.* [80]. They used the Pearson correlation coefficient to measure the similarity between two genes, and provided a very informative visualization of the clustering results. Their results demonstrate that functionally similar genes tend to reside in the same clusters formed by their expression pattern, even under a relatively small set of conditions. Herwig *et al.* developed a variant of K -means algorithm to cluster a set of 2 029 human cDNA clones and adopted mutual information as the similarity measure [230]. Tomayo *et al.* [268] made use of SOFM to cluster gene expression data and its application in hematopoietic differentiation provided new insight for further research. Graph theories based clustering algorithms, like CAST [29] and CLICK [247], showed very promising performances in tackling different types of gene expression data. Since many genes usually display more than one function, fuzzy clustering may be more effective in exposing these relations [73]. Gene expression data is also important to elucidate the genetic regulation mechanism in a cell. By examining the corresponding DNA sequences in the control regions of a cluster of co-expressed genes, we may identify potential short and consensus sequence patterns, known as motifs, and further investigate their interaction with transcriptional binding factors,

leading to different gene activities. Spellman *et al.* clustered 800 genes according to their expression during the yeast cell cycle [260]. Analyses of 8 major gene clusters unravel the connection between co-expression and co-regulation. Tavazoie *et al.* partitioned 3 000 genes into 30 clusters with the K -means algorithm [269]. For each cluster, 600 base pairs upstream sequences of the genes were searched for potential motifs. 18 motifs were found from 12 clusters in their experiments and 7 of them can be verified according to previous empirical results in the literature. A more comprehensive investigation can be found in [206].

As to another application, clustering tissues or experiments are valuable in identifying samples that are in the different disease states, discovering, or predicting different cancer types, and evaluating the effects of novel drugs and therapies [5], [109], [238]. Golub *et al.* described the restriction of traditional cancer classification methods, which are mostly dependent on morphological appearance of tumors, and divided cancer classification into two challenges: class discovery and class prediction. They utilized SOFM to discriminate two types of human acute leukemias: acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) [109]. According to their results, two subsets of ALL, with different origin of lineage, can be well separated. Alon *et al.* performed a two-way clustering for both tissues and genes and revealed the potential relations, represented as visualizing patterns, among them [6]. Alizadeh *et al.* demonstrated the effectiveness of molecular classification of cancers by their gene expression profiles and successfully distinguished two molecularly distinct subtypes of diffuse large B-cell lymphoma, which cause high percentage failure in clinical treatment [5]. Furthermore, Scherf *et al.* constructed a gene expression database to study the relationship between genes and drugs for 60 human cancer cell lines, which provides an important criterion for therapy selection and drug discovery [238]. Other applications of clustering algorithms for tissue classification include: mixtures of multivariate Gaussian distributions [105], ellipsoidal ART [287], and graph theory-based methods [29], [247]. In most of these applications, important genes that are tightly related to the tumor types are identified according to their expression differentiation under different cancerous categories, which are in accord with our prior recognition of roles of these genes, to a large extent [5], [109]. For example, Alon *et al.* found that 5 of 20 statistically significant genes were muscle genes, and the corresponding muscle indices provided an explanation for false classifications [6].

Fig. 7 illustrates an application of hierarchical clustering and SOFM for gene expression data. This data set is on the diagnostic research of small round blue-cell tumors (SRBCT's) of childhood and consists of 83 samples from four categories, known as Burkitt lymphomas (BL), the Ewing family of tumors (EWS), neuroblastoma (NB), and rhabdomyosarcoma (RMS), and 5 non-SRBCT samples [164]. Gene expression levels of 6 567 genes were measured using cDNA microarray for each sample, 2 308 of which passed the filter and were kept for further analyzes. These genes are further ranked according to the scores calculated by some criterion functions [109]. Generally, these criterion functions attempt to seek a subset of genes that contribute most to the discrimination of different cancer types.

This can be regarded as a feature selection process. However, problems like how many genes are really required, and whether these genes selected are really biologically meaningful, are still not answered satisfactorily. Hierarchical clustering was performed by the program CLUSTER and the results were visualized by the program TreeView, developed by Eisen in Stanford University. Fig. 7(a) and (b) depicts the clustering results for both the top 100 genes, selected by the Fisher scores, and the samples. Graphic visualization is achieved by associating each data point with a certain color according to the corresponding scale. Some clustering patterns are clearly displayed in the image. Fig. 7(c) depicts a 5-by-5 SOFM topology for all genes, with each cluster represented by the centroid (mean) for each feature (sample). 25 clusters are generated and the number of genes in each cluster is also indicated. The software package GeneCluster, developed by Whitehead Institute/MIT Center for Genome Research (WICGR), was used in this analysis.

Although clustering techniques have already achieved many impressive results in the analysis of gene expression data, there are still many problems that remain open. Gene expression data sets usually are characterized as

- 1) small set samples with high-dimensional features;
- 2) high redundancy;
- 3) inherent noise;
- 4) sparsity of the data.

Most of the published data sets include usually less than 20 samples for each tumor type, but with as many as thousands of gene measures [80], [109], [238], [268]. This is partly caused by the lag of experimental condition (e.g., sample collection), in contrast to the rapid advancement of microarray and sequencing technologies. In order to evaluate existing algorithms more reasonably and develop more effective new approaches, more data with enough samples or more conditional observations are needed. But from the trend of gene chip technologies, which also follows Moore's law for semiconductor chips [205], the current status will still exist for a long time. This problem is more serious in the application of gene expression data for cancer research, in which clustering algorithms are required to be capable of effectively finding potential patterns under a large number of irrelevant factors, as a result of the introduction of too many genes. At the same time, feature selection, which is also called informative gene selection in the context, also plays a very important role. Without any doubt, clustering algorithms should be feasible in both time and space complexity. Due to the nature of the manufacture process of the microarray chip, noise can be inevitably introduced into the expression data during different stages. Accordingly, clustering algorithms should have noise and outlier detection mechanisms in order to remove their effects. Furthermore, different algorithms usually form different clusters for the same data set, which is a general problem in cluster analysis. How to evaluate the quality of the generated clusters of genes, and how to choose appropriate algorithms for a specified application, are particularly crucial for gene expression data research, because sometimes, even biologists cannot identify the real patterns from the artifacts of the clustering algorithms, due to the limitations of biological

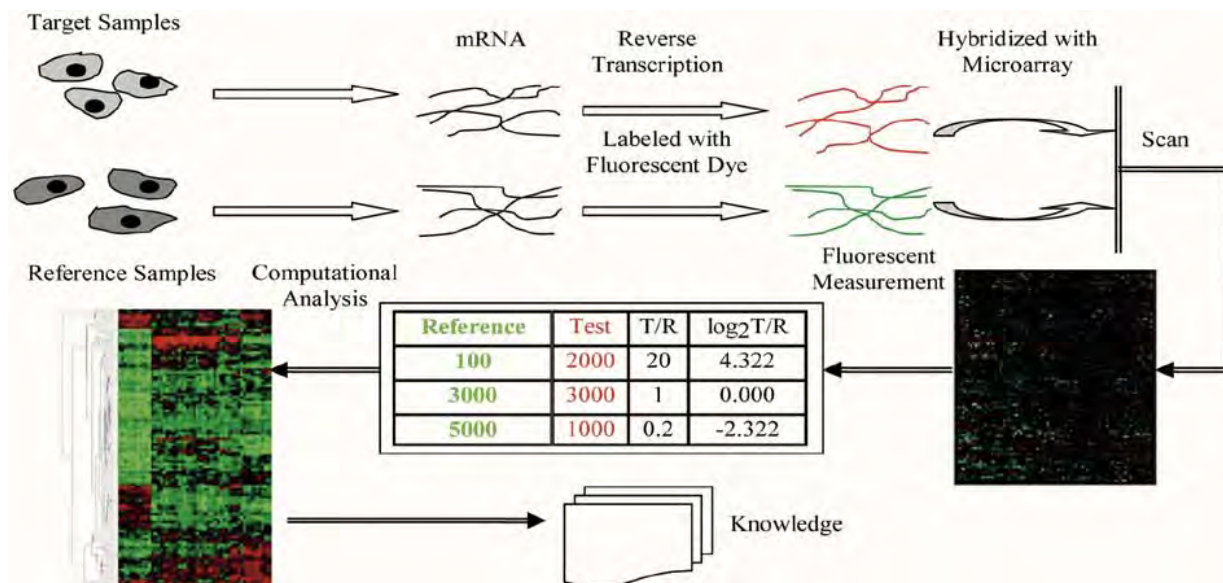


Fig. 6. Basic procedure of cDNA microarray technology [68]. Fluorescently labeled cDNAs, obtained from target and reference samples through reverse transcription, are hybridized with the microarray, which is comprised of a large amount of cDNA clones. Image analysis measures the ratio of the two dyes. Computational methods, e.g., hierarchical clustering, further disclose the relations among genes and corresponding conditions.

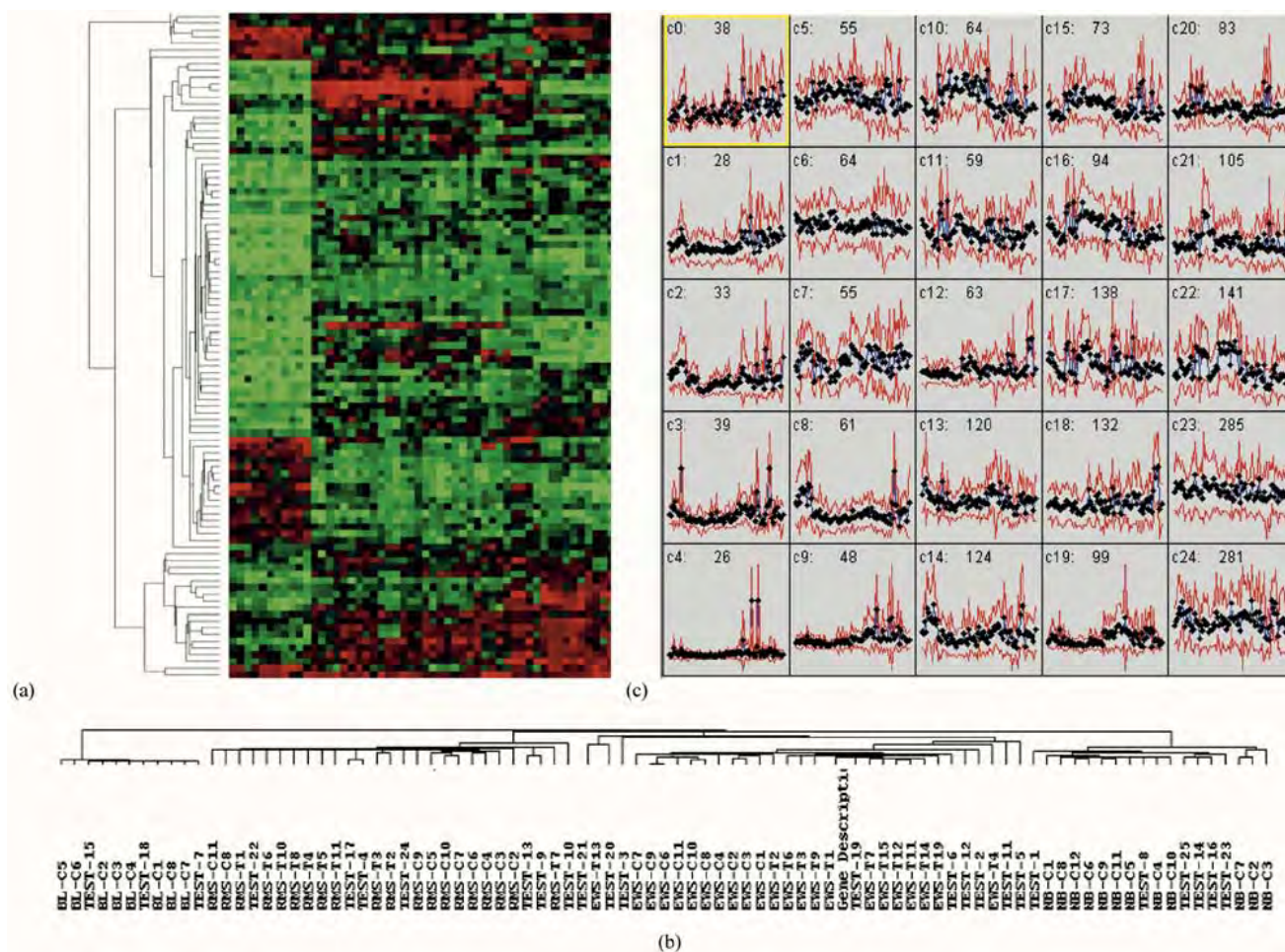


Fig. 7. Hierarchical and SOFM clustering of SRBCT's gene expression data set. (a) Hierarchical clustering result for the 100 selected genes under 83 tissue samples. The gene expression matrix is visualized through a color scale. (b) Hierarchical clustering result for the 83 tissue samples. Here, the dimension is 100 as 100 genes are selected like in (a). (c) SOFM clustering result for the 2308 genes. A 5×5 SOFM is used and 25 clusters are formed. Each cluster is represented by the average values.

knowledge. Some recent results can be accessed in [29], [247], and [291].

E. Bioinformatics—DNA or Protein Sequences Clustering

DNA (deoxyribonucleic acid) is the hereditary material existing in all living cells. A DNA molecule is a double helix consisting of two strands, each of which is a linear sequence composed of four different nucleotides—adenine, guanine, thymine, and cytosine, abbreviated as the letters A, G, T, and C, respectively. Each letter in a DNA sequence is also called a base. Proteins determine most of cells' structures, functions, properties, and regulatory mechanisms. The primary structure of a protein is also a linear and alphabetic chain with the difference that each unit represents an amino acid, which has twenty types in total. Proteins are encoded by certain segments of DNA sequences through a two-stage process (transcription and translation). These segments are known as genes or coding regions. Investigation of the relations between DNA and proteins, as well as their own functions and properties, is one of the important research directions in both genetics and bioinformatics.

The similarity between newly sequenced genes or proteins and annotated genes or proteins usually offers a cue to identify their functions. Searching corresponding databases for a new DNA or protein sequence has already become routine in genetic research. In contrast to sequence comparison and search, cluster analysis provides a more effective means to discover complicated relations among DNA and protein sequences. We summarize the following clustering applications for DNA and protein sequences:

- 1) function recognition of uncharacterized genes or proteins [119];
- 2) structure identification of large-scale DNA or protein databases [237], [257];
- 3) redundancy decrease of large-scale DNA or protein databases [185];
- 4) domain identification [83], [115];
- 5) expressed sequence tag (EST) clustering [49], [200].

As described in Section II-J, classical dynamic programming algorithms for global and local sequence alignment are too intensive in computational complexity. This becomes worse because of the existence of a large volume of nucleic acids and amino acids in the current DNA or protein databases, e.g., bacteria genomes are from 0.5 to 10 Mbp, fungi genomes range from 10 to 50 Mbp, while the human genome is around 3 310 Mbp [18] (Mbp means million base pairs). Thus, conventional dynamic programming algorithms are computationally infeasible. In practice, sequence comparison or proximity measure is achieved via some heuristics. Well-known examples include BLAST and FASTA with many variants [10], [11], [224]. The key idea of these methods is to identify regions that may have potentially high matches, with a list of prespecified high-scoring words, at an early stage. Therefore, further search only needs to focus on these regions with expensive but accurate algorithms. Recognizing the benefit coming from the separation of word matching and sequence alignment to computational burden reduction, Miller, Gurd, and Brass described three algorithms focusing on specific problems [199]. The implementation of the

scheme for large database vs. database comparison exhibits an apparent improvement in computation time. Kent and Zahler designed a three-pass algorithm, called wobble aware bulk aligner (WABA) [162], for aligning large-scale genomic sequences of different species, which employs a seven-state pairwise hidden Markov model [78] for more effective alignments. In [201], Miller summarized the current research status of genomic sequence comparison and suggested valuable directions for further research efforts.

Many clustering techniques have been applied to organize DNA or protein sequence data. Some directly operate on a proximity measure; some are based on feature extraction, while others are constructed on statistical models. Somervuo and Kohonen illustrated an application of SOFM to cluster protein sequences in SWISSPROT database [257]. FASTA was used to calculate the sequence similarity. The resulting two-dimensional SOFM provides a visualized representation of the relations within the entire sequence database. Based on the similarity measure of gapped BLAST, Sasson *et al.* utilized an agglomerative hierarchical clustering paradigm to cluster all protein sequences in SWISSPROT [237]. The effects of four merging rules, different from the interpretation of cluster centers, on the resulting protein clusters were examined. The advantages as well as the potential risk of the concept, transitivity, were also elucidated in the paper. According to the transitivity relation, two sequences that do not show high sequence similarity by virtue of direct comparison, may be homologous (having a common ancestor) if there exists an intermediate sequence similar to both of them. This makes it possible to detect remote homologues that can not be observed by similarity comparison. However, unrelated sequences may be clustered together due to the effects of these intermediate sequences [237]. Bolten *et al.* addressed the problem with the construction a directed graph, in which each protein sequence corresponds to a vertex and edges are weighted based on the alignment score between two sequences and self alignment score of each sequence [41]. Clusters were formed through the search of strongly connected components (SCCs), each of which is a maximal subset of vertices and for each pair of vertices u and v in the subset, there exist two directed paths from u to v and vice versa. A minimum normalized cut algorithm for detecting protein families and a minimum spanning tree (MST) application for seeking domain information were presented in [1] and [115], respectively. In contrast with the aforementioned proximity-based methods, Guralnik and Karypis transformed protein or DNA sequences into a new feature space, based on the detected subpatterns working as the sequence features, and clustered with the K -means algorithm [119]. The method is immune from all-against-all expensive sequence comparison and suitable for analyzing large-scale databases. Krogh demonstrated the power of hidden Markov models (HMMs) in biological sequences modeling and clustering of protein families [177]. Fig. 8 depicts a typical structure of HMM, in which match states (abbreviated with letter M), insert states (I) and delete states (D) are represented as rectangles, diamonds, and circles, respectively [78], [177]. These states correspond to substitution, insertion, and deletion in edit operations. For convenience, a begin state and an end state are added to the

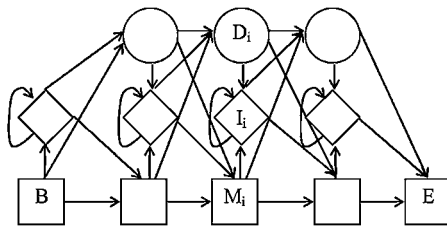


Fig. 8. HMM architecture [177]. There are three different states, match (M), insert (I), and delete (D), corresponding to substitution, insertion, and deletion operation, respectively. A begin (B) and end (E) state are also introduced to represent the start and end of the process. This process goes through a series of states according to the transition probability, and emits either 4-letter nucleotide or 20-letter amino acid alphabet based on the emission probability.

model, denoted by letter B and E. Letters, either from the 4-letter nucleotide alphabet or from 20-letter amino acid alphabet, are generated from match and insert states according to some emission probability distributions. Delete states do not produce any symbols, and are used to skip the match states. K HMMs are required in order to describe K clusters, or families (subfamilies), which are regarded as a mixture model and proceeded with an EM learning algorithm similar to single HMM case. An example for clustering subfamilies of 628 globins shows the encouraging results. Further discussion can be found in [78] and [145].

IV. CONCLUSION

As an important tool for data exploration, cluster analysis examines unlabeled data, by either constructing a hierarchical structure, or forming a set of groups according to a prespecified number. This process includes a series of steps, ranging from preprocessing and algorithm development, to solution validity and evaluation. Each of them is tightly related to each other and exerts great challenges to the scientific disciplines. Here, we place the focus on the clustering algorithms and review a wide variety of approaches appearing in the literature. These algorithms evolve from different research communities, aim to solve different problems, and have their own pros and cons. Though we have already seen many examples of successful applications of cluster analysis, there still remain many open problems due to the existence of many inherent uncertain factors. These problems have already attracted and will continue to attract intensive efforts from broad disciplines. We summarize and conclude the survey with listing some important issues and research trends for cluster algorithms.

- 1) There is no clustering algorithm that can be universally used to solve all problems. Usually, algorithms are designed with certain assumptions and favor some type of biases. In this sense, it is not accurate to say "best" in the context of clustering algorithms, although some comparisons are possible. These comparisons are mostly based on some specific applications, under certain conditions, and the results may become quite different if the conditions change.
- 2) New technology has generated more complex and challenging tasks, requiring more powerful clustering

algorithms. The following properties are important to the efficiency and effectiveness of a novel algorithm.

- I) generate arbitrary shapes of clusters rather than be confined to some particular shape;
- II) handle large volume of data as well as high-dimensional features with acceptable time and storage complexities;
- III) detect and remove possible outliers and noise;
- IV) decrease the reliance of algorithms on users-dependent parameters;
- V) have the capability of dealing with newly occurring data without relearning from the scratch;
- VI) be immune to the effects of order of input patterns;
- VII) provide some insight for the number of potential clusters without prior knowledge;
- VIII) show good data visualization and provide users with results that can simplify further analysis;
- IX) be capable of handling both numerical and nominal data or be easily adaptable to some other data type.

Of course, some more detailed requirements for specific applications will affect these properties.

- 3) At the preprocessing and post-processing phase, feature selection/extraction (as well as standardization and normalization) and cluster validation are as important as the clustering algorithms. Choosing appropriate and meaningful features can greatly reduce the burden of subsequent designs and result evaluations reflect the degree of confidence to which we can rely on the generated clusters. Unfortunately, both processes lack universal guidance. Ultimately, the tradeoff among different criteria and methods is still dependent on the applications themselves.

ACKNOWLEDGMENT

The authors would like to thank the Eisen Laboratory in Stanford University for use of their CLUSTER and TreeView software and Whitehead Institute/MIT Center for Genome Research for use of their GeneCluster software. They would also like to thank S. Mulder for the part on the traveling salesman problem and also acknowledge extensive comments from the reviewers and the anonymous associate editor.

REFERENCES

- [1] F. Abascal and A. Valencia, "Clustering of proximal sequence space for the identification of protein families," *Bioinformatics*, vol. 18, pp. 908–921, 2002.
- [2] C. Aggarwal and P. Yu, "Redefining clustering for high-dimensional applications," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 2, pp. 210–225, Feb. 2002.
- [3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1998, pp. 94–105.
- [4] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. AC-19, no. 6, pp. 716–722, Dec. 1974.
- [5] A. Alizadeh *et al.*, "Distinct types of diffuse large B-cell Lymphoma identified by gene expression profiling," *Nature*, vol. 403, pp. 503–511, 2000.
- [6] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proc. Nat. Acad. Sci. USA*, pp. 6745–6750, 1999.

- [7] C. Alpert and A. Kahng, "Multi-way partitioning via spacefilling curves and dynamic programming," in *Proc. 31st ACM/IEEE Design Automation Conf.*, 1994, pp. 652–657.
- [8] —, "Recent directions in netlist partitioning: A survey," *VLSI J.*, vol. 19, pp. 1–81, 1995.
- [9] K. Al-Sultan, "A Tabu search approach to the clustering problem," *Pattern Recognit.*, vol. 28, no. 9, pp. 1443–1451, 1995.
- [10] S. Altschul *et al.*, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, pp. 3389–3402, 1997.
- [11] S. Altschul *et al.*, "Basic local alignment search tool," *J. Molec. Biol.*, vol. 215, pp. 403–410, 1990.
- [12] G. Anagnostopoulos and M. Georgiopoulos, "Hypersphere ART and ARTMAP for unsupervised and supervised incremental learning," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks (IJCNN'00)*, vol. 6, Como, Italy, pp. 59–64.
- [13] —, "Ellipsoid ART and ARTMAP for incremental unsupervised and supervised learning," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks (IJCNN'01)*, vol. 2, Washington, DC, 2001, pp. 1221–1226.
- [14] M. Anderberg, *Cluster Analysis for Applications*. New York: Academic, 1973.
- [15] G. Babu and M. Murty, "A near-optimal initial seed value selection in K -means algorithm using a genetic algorithm," *Pattern Recognit. Lett.*, vol. 14, no. 10, pp. 763–769, 1993.
- [16] —, "Clustering with evolution strategies," *Pattern Recognit.*, vol. 27, no. 2, pp. 321–329, 1994.
- [17] E. Backer and A. Jain, "A clustering performance measure based on fuzzy set decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 1, pp. 66–75, Jan. 1981.
- [18] P. Baldi and S. Brunak, *Bioinformatics: The Machine Learning Approach*, 2nd ed. Cambridge, MA: MIT Press, 2001.
- [19] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, pp. 53–58, 1989.
- [20] P. Baldi and A. Long, "A Bayesian framework for the analysis of microarray expression data: Regularized t-test and statistical inferences of gene changes," *Bioinformatics*, vol. 17, pp. 509–519, 2001.
- [21] G. Ball and D. Hall, "A clustering technique for summarizing multivariate data," *Behav. Sci.*, vol. 12, pp. 153–155, 1967.
- [22] S. Bandyopadhyay and U. Maulik, "Nonparametric genetic clustering: Comparison of validity indices," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 1, pp. 120–125, Feb. 2001.
- [23] A. Baraldi and E. Alpaydin, "Constructive feedforward ART clustering networks—Part I and II," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 645–677, May 2002.
- [24] A. Baraldi and P. Blonda, "A survey of fuzzy clustering algorithms for pattern recognition—Part I and II," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 6, pp. 778–801, Dec. 1999.
- [25] A. Baraldi and L. Schenato, "Soft-to-hard model transition in clustering: A review," Tech. Rep. TR-99-010, 1999.
- [26] D. Barabási and P. Chen, "Using the fractal dimension to cluster datasets," in *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2000, pp. 260–264.
- [27] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002, vol. 14.
- [28] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton Univ. Press, 1961.
- [29] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns," *J. Comput. Biol.*, vol. 6, pp. 281–297, 1999.
- [30] Y. Bengio, "Markovian models for sequential data," *Neural Comput. Surv.*, vol. 2, pp. 129–162, 1999.
- [31] A. Ben-Hur, D. Horn, H. Siegelmann, and V. Vapnik, "Support vector clustering," *J. Mach. Learn. Res.*, vol. 2, pp. 125–137, 2001.
- [32] —, "A support vector clustering method," in *Proc. Int. Conf. Pattern Recognition*, vol. 2, 2000, pp. 2724–2727.
- [33] P. Berkhin. (2001) Survey of clustering data mining techniques. [Online]. Available: http://www.acrrue.com/products/rp_cluster_review.pdf
<http://citeseer.nj.nec.com/berkhin02survey.html>
- [34] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is nearest neighbor meaningful," in *Proc. 7th Int. Conf. Database Theory*, 1999, pp. 217–235.
- [35] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.
- [36] J. Bezdek and R. Hathaway, "Numerical convergence and interpretation of the fuzzy c -shells clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 787–793, Sep. 1992.
- [37] J. Bezdek and N. Pal, "Some new indexes of cluster validity," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 28, no. 3, pp. 301–315, Jun. 1998.
- [38] C. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford Univ. Press, 1995.
- [39] L. Bobrowski and J. Bezdek, "c-Means clustering with the l_1 and l_∞ norms," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 545–554, May-Jun. 1991.
- [40] H. Bock, "Probabilistic models in cluster analysis," *Comput. Statist. Data Anal.*, vol. 23, pp. 5–28, 1996.
- [41] E. Bolten, A. Schliep, S. Schneckener, D. Schomburg, and R. Schrader, "Clustering protein sequences—Structure prediction by transitive homology," *Bioinformatics*, vol. 17, pp. 935–941, 2001.
- [42] N. Boujemaa, "Generalized competitive clustering for image segmentation," in *Proc. 19th Int. Meeting North American Fuzzy Information Processing Soc. (NAFIPS'00)*, Atlanta, GA, 2000, pp. 133–137.
- [43] P. Bradley and U. Fayyad, "Refining initial points for K -means clustering," in *Proc. 15th Int. Conf. Machine Learning*, 1998, pp. 91–99.
- [44] P. Bradley, U. Fayyad, and C. Reina, "Scaling clustering algorithms to large databases," in *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, 1998, pp. 9–15.
- [45] —, "Clustering very large databases using EM mixture models," in *Proc. 15th Int. Conf. Pattern Recognition*, vol. 2, 2000, pp. 76–80.
- [46] —, "Clustering very large databases using EM mixture models," in *Proc. 15th Int. Conf. Pattern Recognition*, vol. 2, 2000, pp. 76–80.
- [47] D. Brown and C. Huntley, "A practical application of simulated annealing to clustering," *Pattern Recognit.*, vol. 25, no. 4, pp. 401–412, 1992.
- [48] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discov.*, vol. 2, pp. 121–167, 1998.
- [49] J. Burke, D. Davison, and W. Hide, "d2_Cluster: A validated method for clustering EST and full-length cDNA sequences," *Genome Res.*, vol. 9, pp. 1135–1142, 1999.
- [50] I. Cadez, S. Gaffney, and P. Smyth, "A general probabilistic framework for clustering individuals and objects," in *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2000, pp. 140–149.
- [51] G. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vis. Graph. Image Process.*, vol. 37, pp. 54–115, 1987.
- [52] —, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Appl. Opt.*, vol. 26, no. 23, pp. 4919–4930, 1987.
- [53] —, "The ART of adaptive pattern recognition by a self-organizing neural network," *IEEE Computer*, vol. 21, no. 3, pp. 77–88, Mar. 1988.
- [54] —, "ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures," *Neural Netw.*, vol. 3, no. 23, pp. 129–152, 1990.
- [55] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, and D. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, 1992.
- [56] G. Carpenter, S. Grossberg, and J. Reynolds, "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Netw.*, vol. 4, no. 5, pp. 169–181, 1991.
- [57] G. Carpenter, S. Grossberg, and D. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Netw.*, vol. 4, pp. 759–771, 1991.
- [58] G. Celeux and G. Govaert, "A classification EM algorithm for clustering and two stochastic versions," *Comput. Statist. Data Anal.*, vol. 14, pp. 315–332, 1992.
- [59] P. Cheeseman and J. Stutz, "Bayesian classification (AutoClass): Theory and results," in *Advances in Knowledge Discovery and Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Menlo Park, CA: AAAI Press, 1996, pp. 153–180.
- [60] V. Cherkassky and F. Mulier, *Learning From Data: Concepts, Theory, and Methods*. New York: Wiley, 1998.
- [61] J. Cherng and M. Lo, "A hypergraph based clustering algorithm for spatial data sets," in *Proc. IEEE Int. Conf. Data Mining (ICDM'01)*, 2001, pp. 83–90.
- [62] J. Chiang and P. Hao, "A new kernel-based fuzzy clustering approach: Support vector clustering with cell growing," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 4, pp. 518–527, Aug. 2003.
- [63] C. Chinrungrueng and C. Séquin, "Optimal adaptive K -means algorithm with dynamic adjustment of learning rate," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 157–169, Jan. 1995.

- [64] S. Chu and J. Roddick, "A clustering algorithm using the Tabu search approach with simulated annealing," in *Data Mining II—Proceedings of Second International Conference on Data Mining Methods and Databases*, N. Ebecken and C. Brebbia, Eds., Cambridge, U.K., 2000, pp. 515–523.
- [65] I. H. G. S. Consortium, "Initial sequencing and analysis of the human genome," *Nature*, vol. 409, pp. 860–921, 2001.
- [66] J. Corchado and C. Fyfe, "A comparison of kernel methods for instantiating case based reasoning systems," *Comput. Inf. Syst.*, vol. 7, pp. 29–42, 2000.
- [67] M. Cowgill, R. Harvey, and L. Watson, "A genetic algorithm approach to cluster analysis," *Comput. Math. Appl.*, vol. 37, pp. 99–108, 1999.
- [68] C. Cummings and D. Relman, "Using DNA microarray to study host-microbe interactions," *Genomics*, vol. 6, no. 5, pp. 513–525, 2000.
- [69] E. Dahlhaus, "Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition," *J. Algorithms*, vol. 36, no. 2, pp. 205–240, 2000.
- [70] R. Davé, "Adaptive fuzzy c -shells clustering and detection of ellipses," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 643–662, Sep. 1992.
- [71] R. Davé and R. Krishnapuram, "Robust clustering methods: A unified view," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 2, pp. 270–293, May 1997.
- [72] M. Delgado, A. Skármeta, and H. Barberá, "A Tabu search approach to the fuzzy clustering problem," in *Proc. 6th IEEE Int. Conf. Fuzzy Systems*, vol. 1, 1997, pp. 125–130.
- [73] D. Dembélé and P. Kastner, "Fuzzy c -means method for clustering microarray data," *Bioinformatics*, vol. 19, no. 8, pp. 973–980, 2003.
- [74] *Handbook of Pattern Recognition and Computer Vision*, C. Chen, L. Pau, and P. Wang, Eds., World Scientific, Singapore, 1993, pp. 3–32. R. Dubes, "Cluster analysis and related issue".
- [75] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
- [76] J. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters," *J. Cybern.*, vol. 3, no. 3, pp. 32–57, 1974.
- [77] B. Duran and P. Odell, *Cluster Analysis: A Survey*. New York: Springer-Verlag, 1974.
- [78] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
- [79] M. Eisen and P. Brown, "DNA arrays for analysis of gene expression," *Methods Enzymol.*, vol. 303, pp. 179–205, 1999.
- [80] M. Eisen, P. Spellman, P. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," in *Proc. Nat. Acad. Sci. USA*, vol. 95, 1998, pp. 14 863–14 868.
- [81] Y. El-Sonbaty and M. Ismail, "Fuzzy clustering for symbolic data," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 2, pp. 195–204, May 1998.
- [82] T. Eltoft and R. deFigueiredo, "A new neural network for cluster-detection-and-labeling," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 1021–1035, Sep. 1998.
- [83] A. Enright and C. Ouzounis, "GeneRAGE: A robust algorithm for sequence clustering and domain detection," *Bioinformatics*, vol. 16, pp. 451–457, 2000.
- [84] S. Eschrich, J. Ke, L. Hall, and D. Goldgof, "Fast accurate fuzzy clustering through data reduction," *IEEE Trans. Fuzzy Syst.*, vol. 11, no. 2, pp. 262–270, Apr. 2003.
- [85] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, 1996, pp. 226–231.
- [86] V. Estivill-Castro and I. Lee, "AMOEB: Hierarchical clustering based on spatial proximity using Delaunay diagram," in *Proc. 9th Int. Symp. Spatial Data Handling (SDH'99)*, Beijing, China, 1999, pp. 7a.26–7a.41.
- [87] V. Estivill-Castro and J. Yang, "A fast and robust general purpose clustering algorithm," in *Proc. 6th Pacific Rim Int. Conf. Artificial Intelligence (PRICAI'00)*, R. Mizoguchi and J. Slaney, Eds., Melbourne, Australia, 2000, pp. 208–218.
- [88] B. Everitt, S. Landau, and M. Leese, *Cluster Analysis*. London: Arnold, 2001.
- [89] D. Fasulo, "An analysis of recent work on clustering algorithms," Dept. Comput. Sci. Eng., Univ. Washington, Seattle, WA, Tech. Rep. 01-03-02, 1999.
- [90] M. Figueiredo and A. Jain, "Unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 381–396, Mar. 2002.
- [91] D. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Mach. Learn.*, vol. 2, pp. 139–172, 1987.
- [92] R. Fisher, "The use of multiple measurements in taxonomic problems," *Annu. Eugenics*, pt. II, vol. 7, pp. 179–188, 1936.
- [93] D. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 3–14, Jan. 1994.
- [94] E. Forgy, "Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications," *Biometrics*, vol. 21, pp. 768–780, 1965.
- [95] C. Fraley and A. Raftery, "MCLUST: Software for model-based cluster analysis," *J. Classificat.*, vol. 16, pp. 297–306, 1999.
- [96] —, "Model-Based clustering, discriminant analysis, and density estimation," *J. Amer. Statist. Assoc.*, vol. 97, pp. 611–631, 2002.
- [97] J. Friedman, "Exploratory projection pursuit," *J. Amer. Statist. Assoc.*, vol. 82, pp. 249–266, 1987.
- [98] H. Frigui and R. Krishnapuram, "A robust competitive clustering algorithm with applications in computer vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 450–465, May 1999.
- [99] B. Fritzke, (1997) Some competitive learning methods. [Online]. Available: <http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper>
- [100] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 769–783, May 2000.
- [101] V. Ganti, R. Ramakrishnan, J. Gehrke, A. Powell, and J. French, "Clustering large datasets in arbitrary metric spaces," in *Proc. 15th Int. Conf. Data Engineering*, 1999, pp. 502–511.
- [102] I. Gath and A. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 773–781, Jul. 1989.
- [103] *GenBank Release Notes 144.0*.
- [104] A. Geva, "Hierarchical unsupervised fuzzy clustering," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 6, pp. 723–733, Dec. 1999.
- [105] D. Ghosh and A. Chinnaiyan, "Mixture modeling of gene expression data from microarray experiments," *Bioinformatics*, vol. 18, no. 2, pp. 275–286, 2002.
- [106] A. Ghozeil and D. Fogel, "Discovering patterns in spatial data using evolutionary programming," in *Proc. 1st Annu. Conf. Genetic Programming*, 1996, pp. 512–520.
- [107] M. Girolami, "Mercer kernel based clustering in feature space," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 780–784, May 2002.
- [108] F. Glover, "Tabu search, part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.
- [109] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531–537, 1999.
- [110] A. Gordon, "Cluster validation," in *Data Science, Classification, and Related Methods*, C. Hayashi, N. Ohsumi, K. Yajima, Y. Tanaka, H. Bock, and Y. Bada, Eds. New York: Springer-Verlag, 1998, pp. 22–39.
- [111] —, *Classification*, 2nd ed. London, U.K.: Chapman & Hall, 1999.
- [112] J. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, vol. 27, pp. 857–872, 1971.
- [113] S. Grossberg, "Adaptive pattern recognition and universal encoding II: Feedback, expectation, olfaction, and illusions," *Biol. Cybern.*, vol. 23, pp. 187–202, 1976.
- [114] P. Grünwald, P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri, "Minimum encoding approaches for predictive modeling," in *Proc. 14th Int. Conf. Uncertainty in AI (UAI'98)*, 1998, pp. 183–192.
- [115] X. Guan and L. Du, "Domain identification by clustering sequence alignments," *Bioinformatics*, vol. 14, pp. 783–788, 1998.
- [116] S. Guha, R. Rastogi, and K. Shim, "CURE: An efficient clustering algorithm for large databases," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 1998, pp. 73–84.
- [117] —, "ROCK: A robust clustering algorithm for categorical attributes," *Inf. Syst.*, vol. 25, no. 5, pp. 345–366, 2000.
- [118] S. Gupata, K. Rao, and V. Bhatnagar, " K -means clustering algorithm for categorical attributes," in *Proc. 1st Int. Conf. Data Warehousing and Knowledge Discovery (DaWaK'99)*, Florence, Italy, 1999, pp. 203–208.
- [119] V. Guralnik and G. Karypis, "A scalable algorithm for clustering sequential data," in *Proc. 1st IEEE Int. Conf. Data Mining (ICDM'01)*, 2001, pp. 179–186.
- [120] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge, U.K.: Cambridge Univ. Press, 1997.
- [121] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Cluster validity methods: Part I & II," *SIGMOD Record*, vol. 31, no. 2–3, 2002.
- [122] L. Hall, I. Özyurt, and J. Bezdek, "Clustering with a genetically optimized approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 103–112, 1999.

- [123] R. Hammah and J. Curran, "Validity measures for the fuzzy cluster analysis of orientations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1467–1472, Dec. 2000.
- [124] P. Hansen and B. Jaumard, "Cluster analysis and mathematical programming," *Math. Program.*, vol. 79, pp. 191–215, 1997.
- [125] P. Hansen and N. Mladenovi , "J-means: A new local search heuristic for minimum sum of squares clustering," *Pattern Recognit.*, vol. 34, pp. 405–413, 2001.
- [126] F. Harary, *Graph Theory*. Reading, MA: Addison-Wesley, 1969.
- [127] J. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.
- [128] E. Hartuv and R. Shamir, "A clustering algorithm based on graph connectivity," *Inf. Process. Lett.*, vol. 76, pp. 175–181, 2000.
- [129] R. Hathaway and J. Bezdek, "Fuzzy c -means clustering of incomplete data," *IEEE Trans. Syst., Man, Cybern.*, vol. 31, no. 5, pp. 735–744, 2001.
- [130] R. Hathaway, J. Bezdek, and Y. Hu, "Generalized fuzzy c -means clustering strategies using L_p norm distances," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 576–582, Oct. 2000.
- [131] B. Hay, G. Wets, and K. Vanhoof, "Clustering navigation patterns on a website using a sequence alignment method," in *Proc. Intelligent Techniques for Web Personalization: 17th Int. Joint Conf. Artificial Intelligence*, vol. s.l, 2001, pp. 1–6, 200.
- [132] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.
- [133] Q. He, "A review of clustering algorithms as applied to IR," Univ. Illinois at Urbana-Champaign, Tech. Rep. UIUC-1999/6+IRG, 1999.
- [134] M. Healy, T. Caudell, and S. Smith, "A neural architecture for pattern sequence verification through inferencing," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 9–20, Jan. 1993.
- [135] A. Hinneburg and D. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. 4th Int. Conf. Knowledge Discovery and Data Mining (KDD'98)*, 1998, pp. 58–65.
- [136] —, "Optimal grid-clustering: Toward breaking the curse of dimensionality in high-dimensional clustering," in *Proc. 25th VLDB Conf.*, 1999, pp. 506–517.
- [137] F. Hoepfner, "Fuzzy shell clustering algorithms in image processing: Fuzzy c -rectangular and 2-rectangular shells," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 599–613, Nov. 1997.
- [138] J. Hoey, "Clustering contextual facial display sequences," in *Proc. 5th IEEE Int. Conf. Automatic Face and Gesture Recognition (FGR'02)*, 2002, pp. 354–359.
- [139] T. Hofmann and J. Buhmann, "Pairwise data clustering by deterministic annealing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 1, pp. 1–14, Jan. 1997.
- [140] J. Holland, *Adaption in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
- [141] F. H ppner, F. Klawonn, and R. Kruse, *Fuzzy Cluster Analysis: Methods for Classification, Data Analysis, and Image Recognition*. New York: Wiley, 1999.
- [142] Z. Huang, "Extensions to the K -means algorithm for clustering large data sets with categorical values," *Data Mining Knowl. Discov.*, vol. 2, pp. 283–304, 1998.
- [143] J. Huang, M. Georgiopoulos, and G. Heileman, "Fuzzy ART properties," *Neural Netw.*, vol. 8, no. 2, pp. 203–213, 1995.
- [144] P. Huber, "Projection pursuit," *Ann. Statist.*, vol. 13, no. 2, pp. 435–475, 1985.
- [145] R. Hughey and A. Krogh, "Hidden Markov models for sequence analysis: Extension and analysis of the basic method," *CABIOS*, vol. 12, no. 2, pp. 95–107, 1996.
- [146] M. Hung and D. Yang, "An efficient fuzzy c -means clustering algorithm," in *Proc. IEEE Int. Conf. Data Mining*, 2001, pp. 225–232.
- [147] L. Hunt and J. Jorgensen, "Mixture model clustering using the MULTIMIX program," *Australia and New Zealand J. Statist.*, vol. 41, pp. 153–171, 1999.
- [148] J. Hwang, J. Vlontzos, and S. Kung, "A systolic neural network architecture for hidden Markov models," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12, pp. 1967–1979, Dec. 1989.
- [149] A. Hyv rinen, "Survey of independent component analysis," *Neural Comput. Surv.*, vol. 2, pp. 94–128, 1999.
- [150] A. Jain and R. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
- [151] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, 2000.
- [152] A. Jain, M. Murty, and P. Flynn, "Data clustering: A review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.
- [153] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 11, pp. 1370–1386, Nov. 2004.
- [154] C. Jutten and J. Herault, "Blind separation of sources, Part I: An adaptive algorithms based on neuromimetic architecture," *Signal Process.*, vol. 24, no. 1, pp. 1–10, 1991.
- [155] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient K -means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2000.
- [156] N. Karayiannis, "A methodology for construction fuzzy algorithms for learning vector quantization," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 505–518, May 1997.
- [157] N. Karayiannis, J. Bezdek, N. Pal, R. Hathaway, and P. Pai, "Repairs to GLVQ: A new family of competitive learning schemes," *IEEE Trans. Neural Netw.*, vol. 7, no. 5, pp. 1062–1071, Sep. 1996.
- [158] J. Karhunen, E. Oja, L. Wang, R. Vig rio, and J. Joutsensalo, "A class of neural networks for independent component analysis," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 486–504, May 1997.
- [159] G. Karypis, E. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *IEEE Computer*, vol. 32, no. 8, pp. 68–75, Aug. 1999.
- [160] R. Kathari and D. Pitts, "On finding the number of clusters," *Pattern Recognit. Lett.*, vol. 20, pp. 405–416, 1999.
- [161] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.
- [162] W. Kent and A. Zahler, "Conservation, regulation, syteny, and introns in a large-scale *C. Briggsae*—*C. elegans* genomic alignment," *Genome Res.*, vol. 10, pp. 1115–1125, 2000.
- [163] P. Kersten, "Implementation issues in the fuzzy c -medians clustering algorithm," in *Proc. 6th IEEE Int. Conf. Fuzzy Systems*, vol. 2, 1997, pp. 957–962.
- [164] J. Khan, J. Wei, M. Ringn r, L. Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. Antonescu, C. Peterson, and P. Meltzer, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature Med.*, vol. 7, no. 6, pp. 673–679, 2001.
- [165] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
- [166] J. Kleinberg, "An impossibility theorem for clustering," in *Proc. 2002 Conf. Advances in Neural Information Processing Systems*, vol. 15, 2002, pp. 463–470.
- [167] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. 14th Int. Joint Conf. Artificial Intelligence*, 1995, pp. 338–345.
- [168] T. Kohonen, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.
- [169] —, *Self-Organizing Maps*, 3rd ed. New York: Springer-Verlag, 2001.
- [170] T. Kohonen, S. Kaski, K. Lagus, J. Saloj rvi, J. Honkela, V. Paatero, and A. Saarela, "Self organization of a massive document collection," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 574–585, May 2000.
- [171] E. Kolatch. (2001) Clustering algorithms for spatial databases: A Survey. [Online]. Available: <http://citeseer.nj.nec.com/436843.html>
- [172] J. Kolen and T. Hutcheson, "Reducing the time complexity of the fuzzy c -means algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 263–267, Apr. 2002.
- [173] K. Krishna and M. Murty, "Genetic K -means algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 3, pp. 433–439, Jun. 1999.
- [174] R. Krishnapuram, H. Frigui, and O. Nasraoui, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part I and II," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 1, pp. 29–60, Feb. 1995.
- [175] R. Krishnapuram and J. Keller, "A possibilistic approach to clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 2, pp. 98–110, Apr. 1993.
- [176] R. Krishnapuram, O. Nasraoui, and H. Frigui, "The fuzzy c spherical shells algorithm: A new approach," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 663–671, Sep. 1992.
- [177] A. Krogh, M. Brown, I. Mian, K. Sj lander, and D. Haussler, "Hidden Markov models in computational biology: Applications to protein modeling," *J. Molec. Biol.*, vol. 235, pp. 1501–1531, 1994.
- [178] G. Lance and W. Williams, "A general theory of classification sorting strategies: 1. Hierarchical systems," *Comput. J.*, vol. 9, pp. 373–380, 1967.
- [179] M. Law and J. Kwok, "Rival penalized competitive learning for model-based sequence clustering," in *Proc. 15th Int. Conf. Pattern Recognition*, vol. 2, 2000, pp. 195–198.

- [180] Y. Leung, J. Zhang, and Z. Xu, "Clustering by scale-space filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1396–1410, Dec. 2000.
- [181] E. Levine and E. Domany, "Resampling method for unsupervised estimation of cluster validity," *Neural Comput.*, vol. 13, pp. 2573–2593, 2001.
- [182] C. Li and G. Biswas, "Temporal pattern generation using hidden Markov model based unsupervised classification," in *Advances in Intelligent Data Analysis*, ser. Lecture Notes in Computer Science, D. Hand, K. Kok, and M. Berthold, Eds. New York: Springer-Verlag, 1999, vol. 1642.
- [183] —, "Unsupervised learning with mixed numeric and nominal data," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 4, pp. 673–690, Jul.-Aug. 2002.
- [184] C. Li, H. Garcia-Molina, and G. Wiederhold, "Clustering for approximate similarity search in high-dimensional spaces," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 4, pp. 792–808, Jul.-Aug. 2002.
- [185] W. Li, L. Jaroszewski, and A. Godzik, "Clustering of highly homologous sequences to reduce the size of large protein databases," *Bioinformatics*, vol. 17, pp. 282–283, 2001.
- [186] A. Likas, N. Vlassis, and J. Verbeek, "The global K -means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, 2003.
- [187] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operat. Res.*, vol. 21, pp. 498–516, 1973.
- [188] R. Lipshutz, S. Fodor, T. Gingeras, and D. Lockhart, "High density synthetic oligonucleotide arrays," *Nature Genetics*, vol. 21, pp. 20–24, 1999.
- [189] G. Liu, *Introduction to Combinatorial Mathematics*. New York: McGraw-Hill, 1968.
- [190] J. Lozano and P. Larrañaga, "Applying genetic algorithms to search for the best hierarchical clustering of a dataset," *Pattern Recognit. Lett.*, vol. 20, pp. 911–918, 1999.
- [191] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp.*, vol. 1, 1967, pp. 281–297.
- [192] S. C. Madeira and A. L. Oliveira, "Bicustering algorithms for biological data analysis: A survey," *IEEE/ACM Trans. Computat. Biol. Bioinformatics*, vol. 1, no. 1, pp. 24–45, Jan. 2004.
- [193] Y. Man and I. Gath, "Detection and separation of ring-shaped clusters using fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 8, pp. 855–861, Aug. 1994.
- [194] J. Mao and A. Jain, "A self-organizing network for hyperellipsoidal clustering (HEC)," *IEEE Trans. Neural Netw.*, vol. 7, no. 1, pp. 16–29, Jan. 1996.
- [195] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognit.*, vol. 33, pp. 1455–1465, 2000.
- [196] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. New York: Wiley, 1997.
- [197] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.
- [198] G. McLachlan, D. Peel, K. Basford, and P. Adams, "The EMMIX software for the fitting of mixtures of normal and t-components," *J. Statist. Software*, vol. 4, 1999.
- [199] C. Miller, J. Gurd, and A. Brass, "A RAPID algorithm for sequence database comparisons: Application to the identification of vector contamination in the EMBL databases," *Bioinformatics*, vol. 15, pp. 111–121, 1999.
- [200] R. Miller *et al.*, "A comprehensive approach to clustering of expressed human gene sequence: The sequence tag alignment and consensus knowledge base," *Genome Res.*, vol. 9, pp. 1143–1155, 1999.
- [201] W. Miller, "Comparison of genomic DNA sequences: Solved and unsolved problems," *Bioinformatics*, vol. 17, pp. 391–397, 2001.
- [202] G. Milligan and M. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, pp. 159–179, 1985.
- [203] R. Mollineda and E. Vidal, "A relative approach to hierarchical clustering," in *Pattern Recognition and Applications, Frontiers in Artificial Intelligence and Applications*, M. Torres and A. Sanfeliu, Eds. Amsterdam, The Netherlands: IOS Press, 2000, vol. 56, pp. 19–28.
- [204] B. Moore, "ART1 and pattern clustering," in *Proc. 1988 Connectionist Models Summer School*, 1989, pp. 174–185.
- [205] S. Moore, "Making chips to probe genes," *IEEE Spectr.*, vol. 38, no. 3, pp. 54–60, Mar. 2001.
- [206] Y. Moreau, F. Smet, G. Thijs, K. Marchal, and B. Moor, "Functional bioinformatics of microarray data: From expression to regulation," *Proc. IEEE*, vol. 90, no. 11, pp. 1722–1743, Nov. 2002.
- [207] T. Morzy, M. Wojciechowski, and M. Zakrzewicz, "Pattern-oriented hierarchical clustering," in *Proc. 3rd East Eur. Conf. Advances in Databases and Information Systems*, 1999, pp. 179–190.
- [208] S. Mulder and D. Wunsch, "Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks," *Neural Netw.*, vol. 16, pp. 827–832, 2003.
- [209] K. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 181–201, Mar. 2001.
- [210] F. Murtagh, "A survey of recent advances in hierarchical clustering algorithms," *Comput. J.*, vol. 26, no. 4, pp. 354–359, 1983.
- [211] F. Murtagh and M. Berry, "Overcoming the curse of dimensionality in clustering by means of the wavelet transform," *Comput. J.*, vol. 43, no. 2, pp. 107–120, 2000.
- [212] S. Needleman and C. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Molec. Biol.*, vol. 48, pp. 443–453, 1970.
- [213] R. Ng and J. Han, "CLARANS: A method for clustering objects for spatial data mining," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 5, pp. 1003–1016, Sep.-Oct. 2002.
- [214] T. Oates, L. Firoiu, and P. Cohen, "Using dynamic time warping to bootstrap HMM-based clustering of time series," in *Sequence Learning*, ser. LNAI 1828, R. Sun and C. Giles, Eds. Berlin, Germany: Springer-Verlag, 2000, pp. 35–52.
- [215] E. Oja, "Principal components minor components, and linear neural networks," *Neural Netw.*, vol. 5, pp. 927–935, 1992.
- [216] J. Oliver, R. Baxter, and C. Wallace, "Unsupervised learning using MML," in *Proc. 13th Int. Conf. Machine Learning (ICML'96)*, Lorenza, Saitta, 1996, pp. 364–372.
- [217] C. Olson, "Parallel algorithms for hierarchical clustering," *Parallel Comput.*, vol. 21, pp. 1313–1325, 1995.
- [218] C. Ordonez and E. Omiecinski, "Efficient disk-based K-means clustering for relational databases," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 8, pp. 909–921, Aug. 2004.
- [219] L. Owsley, L. Atlas, and G. Bernard, "Self-organizing feature maps and hidden Markov models for machine-tool monitoring," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2787–2798, Nov. 1997.
- [220] N. Pal and J. Bezdek, "On cluster validity for the fuzzy c -means model," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 370–379, Aug. 1995.
- [221] N. Pal, J. Bezdek, and E. Tsao, "Generalized clustering networks and Kohonen's self-organizing scheme," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 549–557, Jul. 1993.
- [222] G. Patané and M. Russo, "The enhanced-LBG algorithm," *Neural Netw.*, vol. 14, no. 9, pp. 1219–1237, 2001.
- [223] —, "Fully automatic clustering system," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1285–1298, Nov. 2002.
- [224] W. Pearson, "Improved tools for biological sequence comparison," *Proc. Nat. Acad. Sci.*, vol. 85, pp. 2444–2448, 1988.
- [225] D. Peel and G. McLachlan, "Robust mixture modeling using the t-distribution," *Statist. Comput.*, vol. 10, pp. 339–348, 2000.
- [226] D. Pelleg and A. Moore, "X-means: Extending K -means with efficient estimation of the number of clusters," in *Proc. 17th Int. Conf. Machine Learning (ICML'00)*, 2000, pp. 727–734.
- [227] J. Peña, J. Lozano, and P. Larrañaga, "An empirical comparison of four initialization methods for the K -means algorithm," *Pattern Recognit. Lett.*, vol. 20, pp. 1027–1040, 1999.
- [228] C. Pizzuti and D. Talia, "P-AutoClass: Scalable parallel clustering for mining large data sets," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 3, pp. 629–641, May-Jun. 2003.
- [229] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
- [230] Ralf-Herwig, A. Poustka, C. Müller, C. Bull, H. Lehrach, and J. O'Brien, "Large-scale clustering of cDNA-fingerprinting data," *Genome Res.*, pp. 1093–1105, 1999.
- [231] A. Rauber, J. Paralic, and E. Pampalk, "Empirical evaluation of clustering algorithms," *J. Inf. Org. Sci.*, vol. 24, no. 2, pp. 195–209, 2000.
- [232] S. Ridella, S. Rovetta, and R. Zunino, "Plastic algorithm for adaptive vector quantization," *Neural Comput. Appl.*, vol. 7, pp. 37–51, 1998.
- [233] J. Rissanen, "Fisher information and stochastic complexity," *IEEE Trans. Inf. Theory*, vol. 42, no. 1, pp. 40–47, Jan. 1996.
- [234] K. Rose, "Deterministic annealing for clustering, compression, classification, regression, and related optimization problems," *Proc. IEEE*, vol. 86, no. 11, pp. 2210–2239, Nov. 1998.

- [235] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [236] D. Sankoff and J. Kruskal, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Stanford, CA: CSLI Publications, 1999.
- [237] O. Sasson, N. Linial, and M. Linial, "The metric space of proteins—Comparative study of clustering algorithms," *Bioinformatics*, vol. 18, pp. s14–s21, 2002.
- [238] U. Scherf, D. Ross, M. Waltham, L. Smith, J. Lee, L. Tanabe, K. Kohn, W. Reinhold, T. Myers, D. Andrews, D. Scudiero, M. Eisen, E. Sausville, Y. Pommier, D. Botstein, P. Brown, and J. Weinstein, "A gene expression database for the molecular pharmacology of cancer," *Nature Genetics*, vol. 24, no. 3, pp. 236–244, 2000.
- [239] P. Scheunders, "A comparison of clustering algorithms applied to color image quantization," *Pattern Recognit. Lett.*, vol. 18, pp. 1379–1384, 1997.
- [240] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.
- [241] B. Schölkopf, A. Smola, and K. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computat.*, vol. 10, no. 5, pp. 1299–1319, 1998.
- [242] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.
- [243] G. Scott, D. Clark, and T. Pham, "A genetic clustering algorithm guided by a descent algorithm," in *Proc. Congr. Evolutionary Computation*, vol. 2, Piscataway, NJ, 2001, pp. 734–740.
- [244] P. Sebastiani, M. Ramoni, and P. Cohen, "Sequence learning via Bayesian clustering by dynamics," in *Sequence Learning*, ser. LNAI 1828, R. Sun and C. Giles, Eds. Berlin, Germany: Springer-Verlag, 2000, pp. 11–34.
- [245] S. Selim and K. Alsultan, "A simulated annealing algorithm for the clustering problems," *Pattern Recognit.*, vol. 24, no. 10, pp. 1003–1008, 1991.
- [246] R. Shamir and R. Sharan, "Algorithmic approaches to clustering gene expression data," in *Current Topics in Computational Molecular Biology*, T. Jiang, T. Smith, Y. Xu, and M. Zhang, Eds. Cambridge, MA: MIT Press, 2002, pp. 269–300.
- [247] R. Sharan and R. Shamir, "CLICK: A clustering algorithm with applications to gene expression analysis," in *Proc. 8th Int. Conf. Intelligent Systems for Molecular Biology*, 2000, pp. 307–316.
- [248] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: A multi-resolution clustering approach for very large spatial databases," in *Proc. 24th VLDB Conf.*, 1998, pp. 428–439.
- [249] P. Simpson, "Fuzzy min-max neural networks—Part 2: Clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 32–45, Feb. 1993.
- [250] *Handbook of Pattern Recognition and Computer Vision*, C. Chen, L. Pau, and P. Wang, Eds., World Scientific, Singapore, 1993, pp. 61–124.
- [251] J. Sklansky and W. Siedlecki, "Large-scale feature selection".
- [252] T. Smith and M. Waterman, "New stratigraphic correlation techniques," *J. Geology*, vol. 88, pp. 451–457, 1980.
- [253] P. Smyth, "Clustering using Monte Carlo cross-validation," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, 1996, pp. 126–133.
- [254] —, "Clustering sequences with hidden Markov models," in *Advances in Neural Information Processing*, M. Mozer, M. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, vol. 9, pp. 648–654.
- [255] —, "Model selection for probabilistic clustering using cross validated likelihood," *Statist. Comput.*, vol. 10, pp. 63–72, 1998.
- [256] —, "Probabilistic model-based clustering of multivariate and sequential data," in *Proc. 7th Int. Workshop on Artificial Intelligence and Statistics*, 1999, pp. 299–304.
- [257] P. Sneath, "The application of computers to taxonomy," *J. Gen. Microbiol.*, vol. 17, pp. 201–226, 1957.
- [258] P. Somervuo and T. Kohonen, "Clustering and visualization of large protein sequence databases by means of an extension of the self-organizing map," in *LNAI 1967*, 2000, pp. 76–85.
- [259] T. Sorensen, "A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons," *Biologiske Skrifter*, vol. 5, pp. 1–34, 1948.
- [260] H. Späth, *Cluster Analysis Algorithms*. Chichester, U.K.: Ellis Horwood, 1980.
- [261] P. Spellman, G. Sherlock, M. Ma, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher, "Comprehensive identification of cell cycle-regulated genes of the Yeast *Saccharomyces Cerevisiae* by microarray hybridization," *Mol. Biol. Cell*, vol. 9, pp. 3273–3297, 1998.
- [262] "Tech. Rep. 00–034," Univ. Minnesota, Minneapolis, 2000.
- [263] K. Stoffel and A. Belkoniene, "Parallel K -means clustering for large data sets," in *Proc. EuroPar'99 Parallel Processing*, 1999, pp. 1451–1454.
- [264] M. Su and H. Chang, "Fast self-organizing feature map algorithm," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 721–733, May 2000.
- [265] M. Su and C. Chou, "A modified version of the K -means algorithm with a distance based on cluster symmetry," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 6, pp. 674–680, Jun. 2001.
- [266] R. Sun and C. Giles, "Sequence learning: Paradigms, algorithms, and applications," in *LNAI 1828*, Berlin, Germany, 2000.
- [267] C. Sung and H. Jin, "A Tabu-search-based heuristic for clustering," *Pattern Recognit.*, vol. 33, pp. 849–858, 2000.
- [268] SWISS-PROT Protein Knowledgebase Release 45.0 Statistics.
- [269] P. Tamayo, D. Slonim, J. Mesirov, Q. Zhu, S. Kitaraweean, E. Dmitrovsky, E. Lander, and T. Golub, "Interpreting patterns of gene expression with self-organizing maps: Methods and application to hematopoietic differentiation," *Proc. Nat. Acad. Sci.*, pp. 2907–2912, 1999.
- [270] S. Tavazoie, J. Hughes, M. Campbell, R. Cho, and G. Church, "Systematic determination of genetic network architecture," *Nature Genetics*, vol. 22, pp. 281–285, 1999.
- [271] J. Tenenbaum, V. Silva, and J. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.
- [272] R. Tibshirani, T. Hastie, M. Eisen, D. Ross, D. Botstein, and P. Brown, "Clustering methods for the analysis of DNA microarray data," Dept. Statist., Stanford Univ., Stanford, CA, Tech. Rep..
- [273] R. Tibshirani and K. Knight, "The covariance inflation criterion for adaptive model selection," *J. Roy. Statist. Soc. B*, vol. 61, pp. 529–546, 1999.
- [274] L. Tseng and S. Yang, "A genetic approach to the automatic clustering problem," *Pattern Recognit.*, vol. 34, pp. 415–424, 2001.
- [275] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.
- [276] J. Venter et al., "The sequence of the human genome," *Science*, vol. 291, pp. 1304–1351, 2001.
- [277] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 586–600, May 2000.
- [278] K. Wagstaff, S. Rogers, and S. Schroedl, "Constrained K -means clustering with background knowledge," in *Proc. 8th Int. Conf. Machine Learning*, 2001, pp. 577–584.
- [279] C. Wallace and D. Dowe, "Intrinsic classification by MML—The SNOB program," in *Proc. 7th Australian Joint Conf. Artificial Intelligence*, 1994, pp. 37–44.
- [280] H. Wang, W. Wang, J. Yang, and P. Yu, "Clustering by pattern similarity in large data sets," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2002, pp. 394–405.
- [281] C. Wei, Y. Lee, and C. Hsu, "Empirical comparison of fast clustering algorithms for large data sets," in *Proc. 33rd Hawaii Int. Conf. System Sciences*, Maui, HI, 2000, pp. 1–10.
- [282] J. Williamson, "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Netw.*, vol. 9, no. 5, pp. 881–897, 1996.
- [283] M. Windham and A. Culter, "Information ratios for validating mixture analysis," *J. Amer. Statist. Assoc.*, vol. 87, pp. 1188–1192, 1992.
- [284] S. Wu, A. W.-C. Liew, H. Yan, and M. Yang, "Cluster analysis of gene expression data based on self-splitting and merging competitive learning," *IEEE Trans. Inf. Technol. Biomed.*, vol. 8, no. 1, pp. 5–15, Jan. 2004.
- [285] D. Wunsch, "An optoelectronic learning machine: Invention, experimentation, analysis of first hardware implementation of the ART1 neural network," Ph.D. dissertation, Univ. Washington, Seattle, WA, 1991.
- [286] D. Wunsch, T. Caudell, C. Capps, R. Marks, and R. Falk, "An optoelectronic implementation of the adaptive resonance neural network," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 673–684, Jul. 1993.
- [287] Y. Xiong and D. Yeung, "Mixtures of ARMA models for model-based time series clustering," in *Proc. IEEE Int. Conf. Data Mining*, 2002, pp. 717–720.
- [288] R. Xu, G. Anagnostopoulos, and D. Wunsch, "Tissue classification through analysis of gene expression data using a new family of ART architectures," in *Proc. Int. Joint Conf. Neural Networks (IJCNN'02)*, vol. 1, 2002, pp. 300–304.
- [289] Y. Xu, V. Olman, and D. Xu, "Clustering gene expression data using graph-theoretic approach: An application of minimum spanning trees," *Bioinformatics*, vol. 18, no. 4, pp. 536–545, 2002.

- [289] R. Yager, "Intelligent control of the hierarchical agglomerative clustering process," *IEEE Trans. Syst., Man, Cybern.*, vol. 30, no. 6, pp. 835–845, 2000.
- [290] R. Yager and D. Filev, "Approximate clustering via the mountain method," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 8, pp. 1279–1284, 1994.
- [291] K. Yeung, D. Haynor, and W. Ruzzo, "Validating clustering for gene expression data," *Bioinformatics*, vol. 17, no. 4, pp. 309–318, 2001.
- [292] F. Young and R. Hamer, *Multidimensional Scaling: History, Theory, and Applications*. Hillsdale, NJ: Lawrence Erlbaum, 1987.
- [293] L. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, pp. 338–353, 1965.
- [294] J. Zhang and Y. Leung, "Improved possibilistic C-means clustering algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 2, pp. 209–217, Apr. 2004.
- [295] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," in *Proc. ACM SIGMOD Conf. Management of Data*, 1996, pp. 103–114.
- [296] Y. Zhang and Z. Liu, "Self-splitting competitive learning: A new on-line clustering paradigm," *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 369–380, Mar. 2002.
- [297] X. Zhuang, Y. Huang, K. Palaniappan, and Y. Zhao, "Gaussian mixture density modeling, decomposition, and applications," *IEEE Trans. Image Process.*, vol. 5, no. 9, pp. 1293–1302, Sep. 1996.



Rui Xu (S'00) received the B.E. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, Hubei, China, in 1997, and the M.E. degree in electrical engineering from Sichuan University, Chengdu, Sichuan, in 2000. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Missouri-Rolla.

His research interests include machine learning, neural networks, pattern classification and clustering, and bioinformatics.

Mr. Xu is a Student Member of the IEEE Computational Intelligence Society, Engineering in Medicine and Biology Society, and the International Society for Computational Biology.



Donald C. Wunsch II (S'87–M'92–SM'94–F'05) received the B.S. degree in applied mathematics from the University of New Mexico, Albuquerque, and the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the University of Washington, Seattle.

He is the Mary K. Finley Missouri Distinguished Professor of Computer Engineering, University of Missouri-Rolla, where he has been since 1999. His prior positions were Associate Professor and Director of the Applied Computational Intelligence

Laboratory, Texas Tech University, Lubbock; Senior Principal Scientist, Boeing; Consultant, Rockwell International; and Technician, International Laser Systems. He has well over 200 publications, and has attracted over \$5 million in research funding. He has produced eight Ph.D. recipients—four in electrical engineering, three in computer engineering, and one in computer science.

Dr. Wunsch has received the Halliburton Award for Excellence in Teaching and Research, and the National Science Foundation CAREER Award. He served as a Voting Member of the IEEE Neural Networks Council, Technical Program Co-Chair for IJCNN'02, General Chair for IJCNN'03, International Neural Networks Society Board of Governors Member, and is now President of the International Neural Networks Society.