

# Artificial Intelligence

## A Modern Approach

### *Second Edition*

Stuart J. Russell and Peter Norvig

*Contributing writers:*

John F. Canny  
Douglas D. Edwards  
Jitendra M. Malik  
Sebastian Thrun



*Pearson Education International*

If you purchased this book within the United States or Canada you should be aware that it has been wrongfully imported without the approval of the Publisher or the Author.

Vice President and Editorial Director, ECS: Marcia J. Horton  
Publisher: Alan R. Apt  
Associate Editor: Toni Dianne Holm  
Editorial Assistant: Patrick Lindner  
Vice President and Director of Production and Manufacturing, ESM: David W. Riccardi  
Executive Managing Editor: Vince O'Brien  
Assistant Managing Editor: Camille Trentacoste  
Production Editor: Irwin Zucker  
Manufacturing Manager: Trudy Piscioti  
Manufacturing Buyer: Lisa McDowell  
Director, Creative Services: Paul Belfanti  
Creative Director: Carole Anson  
Art Editor: Greg Dulles  
Art Director: Heather Scott  
Assistant to Art Director: Geoffrey Cassar  
Cover Designers: Stuart Russell and Peter Norvig  
Cover Image Creation: Stuart Russell and Peter Norvig; Tamara Newnam and Patrice Van Acker  
Interior Designer: Stuart Russell and Peter Norvig  
Marketing Manager: Pamela Shaffer  
Marketing Assistant: Barrie Reinhold



© 2003, 1995 by Pearson Education, Inc.  
Pearson Education, Inc.,  
Upper Saddle River, New Jersey 07458

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, express or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America

10 9 8 7 6 5 4 3

**ISBN 0-13-080302-2**

Pearson Education Ltd., *London*  
Pearson Education Australia Pty. Ltd., *Sydney*  
Pearson Education Singapore, Pte. Ltd.  
Pearson Education North Asia Ltd., *Hong Kong*  
Pearson Education Canada, Inc., *Toronto*  
Pearson Educación de México, S.A. de C.V.  
Pearson Education—Japan, *Tokyo*  
Pearson Education Malaysia, Pte. Ltd.  
Pearson Education, Inc., *Upper Saddle River, New Jersey*

**Trend Mico, Inc.**  
**EX1008-2**

20.6    KERNEL MACHINES

SUPPORT VECTOR  
MACHINE  
  
KERNEL MACHINE

Our discussion of neural networks left us with a dilemma. Single-layer networks have a simple and efficient learning algorithm, but have very limited expressive power—they can learn only linear decision boundaries in the input space. Multilayer networks, on the other hand, are much more expressive—they can represent general nonlinear functions—but are very hard to train because of the abundance of local minima and the high dimensionality of the weight space. In this section, we will explore a relatively new family of learning methods called **support vector machines** (SVMs) or, more generally, **kernel machines**. To some extent, kernel machines give us the best of both worlds. That is, these methods use an efficient training algorithm *and* can represent complex, nonlinear functions.

The full treatment of kernel machines is beyond the scope of the book, but we can illustrate the main idea through an example. Figure 20.27(a) shows a two-dimensional input space defined by attributes  $\mathbf{x} = (x_1, x_2)$ , with positive examples ( $y = +1$ ) inside a circular region and negative examples ( $y = -1$ ) outside. Clearly, there is no linear separator for this problem. Now, suppose we re-express the input data using some computed features—i.e., we map each input vector  $\mathbf{x}$  to a new vector of feature values,  $F(\mathbf{x})$ . In particular, let us use the three features

$$f_1 = x_1^2, \quad f_2 = x_2^2, \quad f_3 = \sqrt{2}x_1x_2. \tag{20.16}$$

We will see shortly where these came from, but, for now, just look at what happens. Figure 20.27(b) shows the data in the new, three-dimensional space defined by the three features; the data are *linearly separable* in this space! This phenomenon is actually fairly general: if data are mapped into a space of sufficiently high dimension, then they will always be linearly separable. Here, we used only three dimensions,<sup>14</sup> but if we have  $N$  data points then, except in special cases, they will always be separable in a space of  $N - 1$  dimensions or more (Exercise 20.21).

MARGIN

So, is that it? Do we just produce loads of computed features and then find a linear separator in the corresponding high-dimensional space? Unfortunately, it's not that easy. Remember that a linear separator in a space of  $d$  dimensions is defined by an equation with  $d$  parameters, so we are in serious danger of overfitting the data if  $d \approx N$ , the number of data points. (This is like overfitting data with a high-degree polynomial, which we discussed in Chapter 18.) For this reason, kernel machines usually find the *optimal* linear separator—the one that has the largest **margin** between it and the positive examples on one side and the negative examples on the other. (See Figure 20.28.) It can be shown, using arguments from computational learning theory (Section 18.5), that this separator has desirable properties in terms of robust generalization to new examples.

QUADRATIC  
PROGRAMMING

Now, how do we find this separator? It turns out that this is a **quadratic programming** optimization problem. Suppose we have examples  $\mathbf{x}_i$  with classifications  $y_i = \pm 1$  and we want to find an optimal separator in the input space; then the quadratic programming problem

<sup>14</sup> The reader may notice that we could have used just  $f_1$  and  $f_2$ , but the 3D mapping illustrates the idea better.

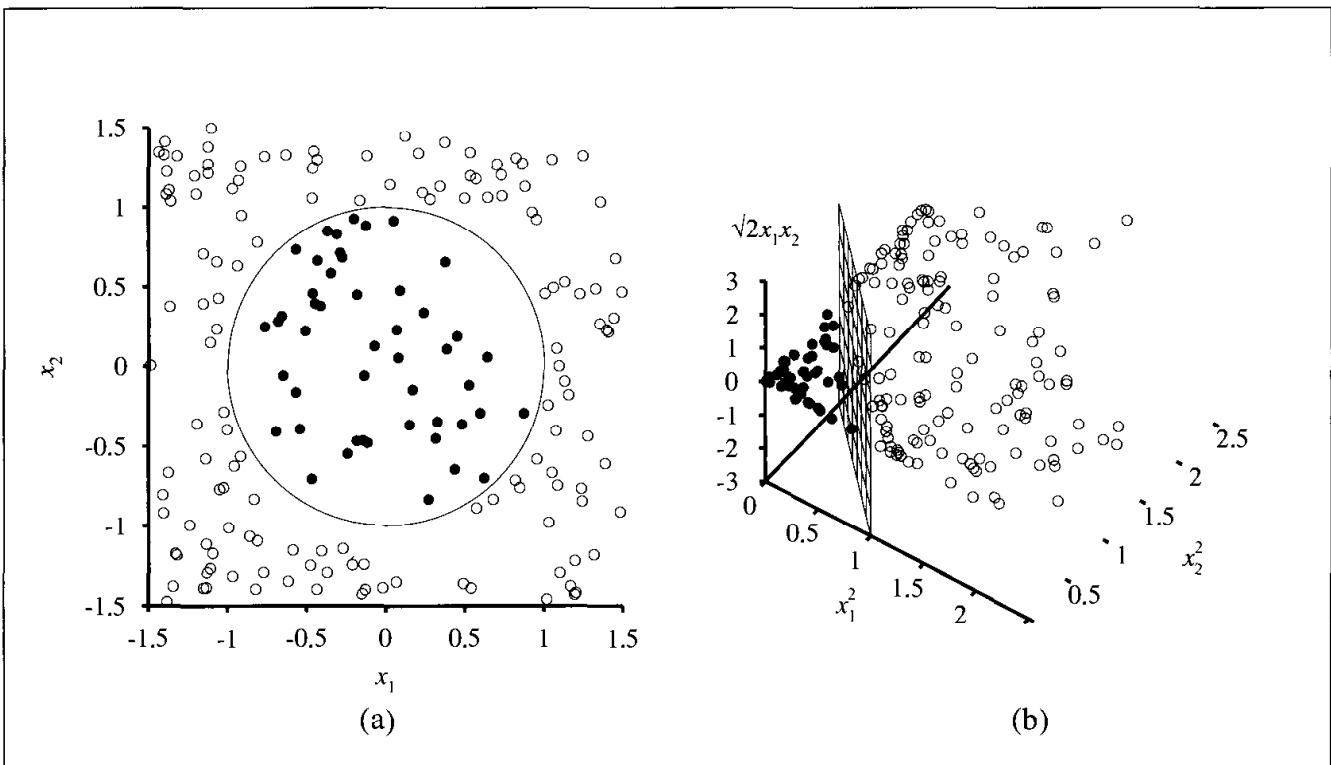


**Figure 20.27** (a) A two-dimensional training with positive examples as black circles and negative examples as white circles. The true decision boundary,  $x_1^2 + x_2^2 \leq 1$ , is also shown. (b) The same data after mapping into a three-dimensional input space  $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$ . The circular decision boundary in (a) becomes a linear decision boundary in three dimensions.

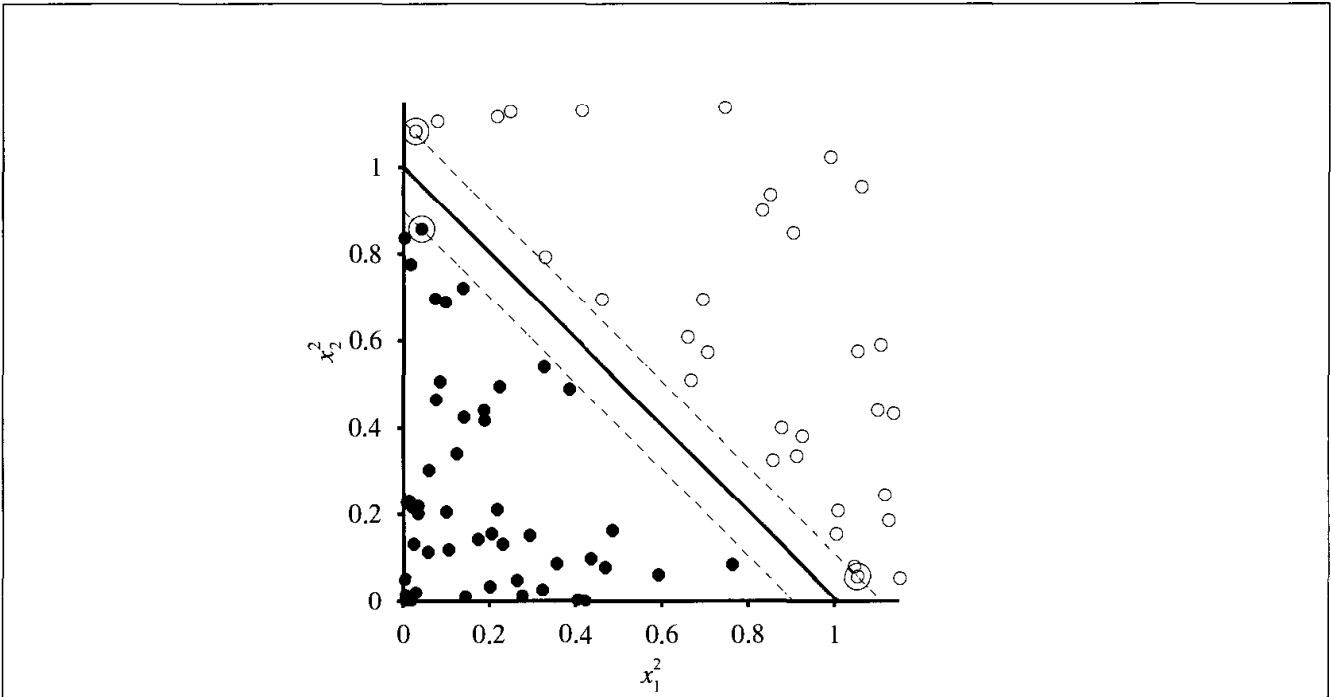


**Figure 20.28** A close-up, projected onto the first two dimensions, of the optimal separator shown in Figure 20.27(b). The separator is shown as a heavy line, with the closest points—the **support vectors**—marked with circles. The **margin** is the separation between the positive and negative examples.

is to find values of the parameters  $\alpha_i$  that maximize the expression

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j) \quad (20.17)$$

subject to the constraints  $\alpha_i \geq 0$  and  $\sum_i \alpha_i y_i = 0$ . Although the derivation of this expression is not crucial to the story, it does have two important properties. First, the expression has a single global maximum that can be found efficiently. Second, *the data enter the expression only in the form of dot products of pairs of points*. This second property is also true of the equation for the separator itself; once the optimal  $\alpha_i$ s have been calculated, it is

$$h(\mathbf{x}) = \text{sign} \left( \sum_i \alpha_i y_i (\mathbf{x} \cdot \mathbf{x}_i) \right). \quad (20.18)$$

A final important property of the optimal separator defined by this equation is that the weights  $\alpha_i$  associated with each data point are *zero* except for those points closest to the separator—the so-called **support vectors**. (They are called this because they “hold up” the separating plane.) Because there are usually many fewer support vectors than data points, the effective number of parameters defining the optimal separator is usually much less than  $N$ .

Now, we would not usually expect to find a linear separator in the input space  $\mathbf{x}$ , but it is easy to see that we can find linear separators in the high-dimensional feature space  $F(\mathbf{x})$  simply by replacing  $\mathbf{x}_i \cdot \mathbf{x}_j$  in Equation (20.17) with  $F(\mathbf{x}_i) \cdot F(\mathbf{x}_j)$ . This by itself is not remarkable—replacing  $\mathbf{x}$  by  $F(\mathbf{x})$  in *any* learning algorithm has the required effect—but the dot product has some special properties. It turns out that  $F(\mathbf{x}_i) \cdot F(\mathbf{x}_j)$  can often be computed without first computing  $F$  for each point. In our three-dimensional feature space defined by Equation (20.16), a little bit of algebra shows that

$$F(\mathbf{x}_i) \cdot F(\mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2.$$

The expression  $(\mathbf{x}_i \cdot \mathbf{x}_j)^2$  is called a **kernel function**, usually written as  $K(\mathbf{x}_i, \mathbf{x}_j)$ . In the kernel machine context, this means a function that can be applied to pairs of input data to evaluate dot products in some corresponding feature space. So, we can restate the claim at the beginning of this paragraph as follows: we can find linear separators in the high-dimensional feature space  $F(\mathbf{x})$  simply by replacing  $\mathbf{x}_i \cdot \mathbf{x}_j$  in Equation (20.17) with a kernel function  $K(\mathbf{x}_i, \mathbf{x}_j)$ . Thus, we can learn in the high-dimensional space but we compute only kernel functions rather than the full list of features for each data point.

The next step, which should by now be obvious, is to see that there’s nothing special about the kernel  $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^2$ . It corresponds to a particular higher-dimensional feature space, but other kernel functions correspond to other feature spaces. A venerable result in mathematics, **Mercer’s theorem** (1909), tells us that any “reasonable”<sup>15</sup> kernel function corresponds to *some* feature space. These feature spaces can be very large, even for innocuous-looking kernels. For example, the **polynomial kernel**,  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^d$ , corresponds to a feature space whose dimension is exponential in  $d$ . Using such kernels in Equation (20.17), then, *optimal linear separators can be found efficiently in feature spaces with billions (or, in some cases, infinitely many) dimensions*. The resulting linear separators,

<sup>15</sup> Here, “reasonable” means that the matrix  $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$  is positive definite; see Appendix A.



SUPPORT VECTOR

MERCER'S THEOREM

POLYNOMIAL  
KERNEL

