mp3

MPEG2

H.263

H.264

Wavelets

SPIHT

JPEG-LS

CELP

PPM

BWT

VQ

TCQ

CALIC

MELP

KHALID SAYOOD

Introduction to DATA COMPRESSION

THIRD EDITION

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000001
IPR2024-00921 (Amazon.com, Inc., Amazon.com Services LLC v. Nokia Technologies Oy)

THIRD EDITION

Introduction to Data Compression

Khalid Sayood
University of Nebraska

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier

MORGAN KAUFMANN

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000002

Senior Acquisitions Editor	Rick Adams
Publishing Services Manager	Simon Crump
Assistant Editor	Rachel Roumeliotis
Cover Design	Cate Barr
Composition	Integra Software Services Pvt. Ltd.
Copyeditor	Jessika Bella Mura
Proofreader	Jacqui Brownstein
Indexer	Northwind Editorial Sevices
Interior printer	Maple Vail Book Manufacturing Group
Cover printer	Phoenix Color

Morgan Kaufmann Publishers is an imprint of Elsevier.
500 Sansome Street, Suite 400, San Francisco, CA 94111

This book is printed on acid-free paper.

©2006 by Elsevier Inc. All rights reserved.

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, scanning, or otherwise—without prior written permission of the publisher.

Permissions may be sought directly from Elsevier's Science & Technology Rights Department in Oxford, UK: phone: (+44) 1865 843830, fax: (+44) 1865 853333, e-mail: permissions@elsevier.co.uk. You may also complete your request on-line via the Elsevier homepage (http://elsevier.com) by selecting "Customer Support" and then "Obtaining Permissions."

Library of Congress Cataloging-in-Publication Data
Sayood, Khalid.
Introduction to data compression/Khalid Sayood.—3rd ed.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-12-620862-7
ISBN-10: 0-12-620862-X
1. Data compression (Telecommunication) 2. Coding theory. I. Title
TK5102.92.S39 2005
005.74′6—dc22
2005052759

ISBN 13: 978-0-12-620862-7
ISBN 10: 0-12-620862-X

For information on all Morgan Kaufmann publications,
visit our Web site at www.mkp.com or www.books.elsevier.com

Printed in the United States of America
05 06 07 08 09 5 4 3 2 1

Working together to grow
libraries in developing countries
www.elsevier.com | www.bookaid.org | www.sabre.org
ELSEVIER BOOK AID International Sabre Foundation

code tree that would have a shorter average length than our original "optimum" tree. This would contradict our statement about the optimality of the original tree.

In order to satisfy conditions 1, 2, and 3, the two least probable letters would have to be assigned codewords of maximum length l_m. Furthermore, the leaves corresponding to these letters arise from the same intermediate node. This is the same as saying that the codewords for these letters are identical except for the last bit. Consider the common prefix as the codeword for the composite letter of a reduced alphabet. Since the code for the reduced alphabet needs to be optimum for the code of the original alphabet to be optimum, we follow the same procedure again. To satisfy the necessary conditions, the procedure needs to be iterated until we have a reduced alphabet of size one. But this is exactly the Huffman procedure. Therefore, the necessary conditions above, which are all satisfied by the Huffman procedure, are also sufficient conditions.

3.2.3 Length of Huffman Codes ★

We have said that the Huffman coding procedure generates an optimum code, but we have not said what the average length of an optimum code is. The length of any code will depend on a number of things, including the size of the alphabet and the probabilities of individual letters. In this section we will show that the optimal code for a source $\mathcal{S}$, hence the Huffman code for the source $\mathcal{S}$, has an average code length $\bar{l}$ bounded below by the entropy and bounded above by the entropy plus 1 bit. In other words,

$$H(\mathcal{S}) \leq \bar{l} < H(\mathcal{S}) + 1. \tag{3.1}$$

In order for us to do this, we will need to use the Kraft-McMillan inequality introduced in Chapter 2. Recall that the first part of this result, due to McMillan, states that if we have a uniquely decodable code $\mathcal{C}$ with K codewords of length $\{l_i\}_{i=1}^{K}$, then the following inequality holds:

$$\sum_{i=1}^{K} 2^{-l_i} \leq 1. \tag{3.2}$$

Example 3.2.2:

Examining the code generated in Example 3.2.1 (Table 3.5), the lengths of the codewords are $\{1, 2, 3, 4, 4\}$. Substituting these values into the left-hand side of Equation (3.2), we get

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-4} = 1$$

which satisfies the Kraft-McMillan inequality.

If we use the minimum variance code (Table 3.9), the lengths of the codewords are $\{2, 2, 2, 3, 3\}$. Substituting these values into the left-hand side of Equation (3.2), we get

$$2^{-2} + 2^{-2} + 2^{-2} + 2^{-3} + 2^{-3} = 1$$

which again satisfies the inequality. ♦

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000004

The second part of this result, due to Kraft, states that if we have a sequence of positive integers $\{l_i\}_{i=1}^K$, which satisfies (3.2), then there exists a uniquely decodable code whose codeword lengths are given by the sequence $\{l_i\}_{i=1}^K$.

Using this result, we will now show the following:

1. The average codeword length $\bar{l}$ of an optimal code for a source $\mathcal{S}$ is greater than or equal to $H(\mathcal{S})$.
2. The average codeword length $\bar{l}$ of an optimal code for a source $\mathcal{S}$ is strictly less than $H(\mathcal{S})+1$.

For a source $\mathcal{S}$ with alphabet $\mathcal{A} = \{a_1, a_2, \ldots a_K\}$, and probability model $\{P(a_1), P(a_2), \ldots, P(a_K)\}$, the average codeword length is given by

$$\bar{l} = \sum_{i=1}^{K} P(a_i) l_i.$$

Therefore, we can write the difference between the entropy of the source $H(\mathcal{S})$ and the average length as

$$\begin{aligned}
H(\mathcal{S}) - \bar{l} &= -\sum_{i=1}^{K} P(a_i)\log_2 P(a_i) - \sum_{i=1}^{K} P(a_i) l_i \\
&= \sum_{i=1}^{K} P(a_i)\left(\log_2\left[\frac{1}{P(a_i)}\right] - l_i\right) \\
&= \sum_{i=1}^{K} P(a_i)\left(\log_2\left[\frac{1}{P(a_i)}\right] - \log_2[2^{l_i}]\right) \\
&= \sum_{i=1}^{K} P(a_i)\log_2\left[\frac{2^{-l_i}}{P(a_i)}\right] \\
&\leq \log_2\left[\sum_{i=1}^{K} 2^{-l_i}\right].
\end{aligned}$$

The last inequality is obtained using Jensen's inequality, which states that if $f(x)$ is a concave (convex cap, convex $\cap$) function, then $E[f(X)] \leq f(E[X])$. The log function is a concave function.

As the code is an optimal code $\sum_{i=1}^{K} 2^{-l_i} \leq 1$, therefore

$$H(\mathcal{S}) - \bar{l} \leq 0. \tag{3.3}$$

We will prove the upper bound by showing that there exists a uniquely decodable code with average codeword length $H(\mathcal{S})+1$. Therefore, if we have an optimal code, this code must have an average length that is less than or equal to $H(\mathcal{S})+1$.

Given a source, alphabet, and probability model as before, define

$$l_i = \left\lceil \log_2 \frac{1}{P(a_i)} \right\rceil$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000005

where $\lceil x \rceil$ is the smallest integer greater than or equal to x. For example, $\lceil 3.3 \rceil = 4$ and $\lceil 5 \rceil = 5$. Therefore,

$$\lceil x \rceil = x + \epsilon \qquad \text{where } 0 \le \epsilon < 1.$$

Therefore,

$$\log_2 \frac{1}{P(a_i)} \le l_i < \log_2 \frac{1}{P(a_i)} + 1. \tag{3.4}$$

From the left inequality of (3.4) we can see that

$$2^{-l_i} \le P(a_i).$$

Therefore,

$$\sum_{i=1}^{K} 2^{-l_i} \le \sum_{i=1}^{K} P(a_i) = 1$$

and by the Kraft-McMillan inequality there exists a uniquely decodable code with codeword lengths $\{l_i\}$. The average length of this code can be upper-bounded by using the right inequality of (3.4):

$$\bar{l} = \sum_{i=1}^{K} P(a_i) l_i < \sum_{i=1}^{K} P(a_i) \left[\log_2 \frac{1}{P(a_i)} + 1 \right]$$

or

$$\bar{l} < H(\mathcal{S}) + 1.$$

We can see from the way the upper bound was derived that this is a rather loose upper bound. In fact, it can be shown that if $p_{\max}$ is the largest probability in the probability model, then for $p_{\max} \ge 0.5$, the upper bound for the Huffman code is $H(\mathcal{S}) + p_{\max}$, while for $p_{\max} < 0.5$, the upper bound is $H(\mathcal{S}) + p_{\max} + 0.086$. Obviously, this is a much tighter bound than the one we derived above. The derivation of this bound takes some time (see [23] for details).

3.2.4 Extended Huffman Codes ★

In applications where the alphabet size is large, $p_{\max}$ is generally quite small, and the amount of deviation from the entropy, especially in terms of a percentage of the rate, is quite small. However, in cases where the alphabet is small and the probability of occurrence of the different letters is skewed, the value of $p_{\max}$ can be quite large and the Huffman code can become rather inefficient when compared to the entropy.

Example 3.2.3:

Consider a source that puts out *iid* letters from the alphabet $\mathcal{A} = \{a_1, a_2, a_3\}$ with the probability model $P(a_1) = 0.8$, $P(a_2) = 0.02$, and $P(a_3) = 0.18$. The entropy for this source is 0.816 bits/symbol. A Huffman code for this source is shown in Table 3.10.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000006

TABLE 3.10 Huffman code for the alphabet $\mathcal{A}$.

Letter	Codeword
a_1	0
a_2	11
a_3	10

The average length for this code is 1.2 bits/symbol. The difference between the average code length and the entropy, or the redundancy, for this code is 0.384 bits/symbol, which is 47% of the entropy. This means that to code this sequence we would need 47% more bits than the minimum required. ♦

We can sometimes reduce the coding rate by blocking more than one symbol together. To see how this can happen, consider a source S that emits a sequence of letters from an alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$. Each element of the sequence is generated independently of the other elements in the sequence. The entropy for this source is given by

$$H(S) = -\sum_{i=1}^{m} P(a_i) \log_2 P(a_i).$$

We know that we can generate a Huffman code for this source with rate R such that

$$H(S) \leq R < H(S) + 1. \tag{3.5}$$

We have used the looser bound here; the same argument can be made with the tighter bound. Notice that we have used "rate R" to denote the number of bits per symbol. This is a standard convention in the data compression literature. However, in the communication literature, the word "rate" often refers to the number of bits per second.

Suppose we now encode the sequence by generating one codeword for every n symbols. As there are m^n combinations of n symbols, we will need m^n codewords in our Huffman code. We could generate this code by viewing the m^n symbols as letters of an *extended alphabet*

$$\mathcal{A}^{(n)} = \{\overbrace{a_1 a_1 \ldots a_1}^{n \text{ times}}, a_1 a_1 \ldots a_2, \ldots, a_1 a_1 \ldots a_m, a_1 a_1 \ldots a_2 a_1, \ldots, a_m a_m \ldots a_m\}$$

from a source $S^{(n)}$. Let us denote the rate for the new source as $R^{(n)}$. Then we know that

$$H(S^{(n)}) \leq R^{(n)} < H(S^{(n)}) + 1. \tag{3.6}$$

$R^{(n)}$ is the number of bits required to code n symbols. Therefore, the number of bits required per symbol, R, is given by

$$R = \frac{1}{n} R^{(n)}.$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000007

The number of bits per symbol can be bounded as

$$\frac{H(S^{(n)})}{n} \le R < \frac{H(S^{(n)})}{n} + \frac{1}{n}.$$

In order to compare this to (3.5), and see the advantage we get from encoding symbols in blocks instead of one at a time, we need to express $H(S^{(n)})$ in terms of $H(S)$. This turns out to be a relatively easy (although somewhat messy) thing to do.

$$\begin{aligned}
H(S^{(n)}) &= -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1}, a_{i_2}, \ldots a_{i_n}) \log[P(a_{i_1}, a_{i_2}, \ldots a_{i_n})] \\
&= -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1})P(a_{i_2})\ldots P(a_{i_n}) \log[P(a_{i_1})P(a_{i_2})\ldots P(a_{i_n})] \\
&= -\sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1})P(a_{i_2})\ldots P(a_{i_n}) \sum_{j=1}^{n} \log[P(a_{i_j})] \\
&= -\sum_{i_1=1}^{m} P(a_{i_1}) \log[P(a_{i_1})] \left\{ \sum_{i_2=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_2})\ldots P(a_{i_n}) \right\} \\
&\quad -\sum_{i_2=1}^{m} P(a_{i_2}) \log[P(a_{i_2})] \left\{ \sum_{i_1=1}^{m}\sum_{i_3=1}^{m}\cdots\sum_{i_n=1}^{m} P(a_{i_1})P(a_{i_3})\ldots P(a_{i_n}) \right\} \\
&\quad \vdots \\
&\quad -\sum_{i_n=1}^{m} P(a_{i_n}) \log[P(a_{i_n})] \left\{ \sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\cdots\sum_{i_{n-1}=1}^{m} P(a_{i_1})P(a_{i_2})\ldots P(a_{i_{n-1}}) \right\}
\end{aligned}$$

The $n-1$ summations in braces in each term sum to one. Therefore,

$$\begin{aligned}
H(S^{(n)}) &= -\sum_{i_1=1}^{m} P(a_{i_1}) \log[P(a_{i_1})] - \sum_{i_2=1}^{m} P(a_{i_2}) \log[P(a_{i_2})] - \cdots - \sum_{i_n=1}^{m} P(a_{i_n}) \log[P(a_{i_n})] \\
&= nH(S)
\end{aligned}$$

and we can write (3.6) as

$$H(S) \le R \le H(S) + \frac{1}{n}. \tag{3.7}$$

Comparing this to (3.5), we can see that by encoding the output of the source in longer blocks of symbols we are *guaranteed* a rate closer to the entropy. Note that all we are talking about here is a bound or guarantee about the rate. As we have seen in the previous chapter, there are a number of situations in which we can achieve a rate *equal* to the entropy with a block length of one!

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000008

We have already shown that $\lfloor \bar{T}_X(\mathbf{x}) \rfloor_{l(\mathbf{x})} > F_X(\mathbf{x}-1)$. Therefore, all we need to do is show that

$$F_X(\mathbf{x}) - \lfloor \bar{T}_X(\mathbf{x}) \rfloor_{l(\mathbf{x})} > \frac{1}{2^{l(\mathbf{x})}}.$$

This is true because

$$\begin{aligned} F_X(\mathbf{x}) - \lfloor \bar{T}_X(\mathbf{x}) \rfloor_{l(\mathbf{x})} &> F_X(\mathbf{x}) - \bar{T}_X(\mathbf{x}) \\ &= \frac{P(\mathbf{x})}{2} \\ &> \frac{1}{2^{l(\mathbf{x})}}. \end{aligned}$$

This code is prefix free, and by taking the binary representation of $\bar{T}_X(\mathbf{x})$ and truncating it to $l(x) = \lceil \log \frac{1}{P(x)} \rceil + 1$ bits, we obtain a uniquely decodable code.

Although the code is uniquely decodable, how efficient is it? We have shown that the number of bits $l(\mathbf{x})$ required to represent $F_X(\mathbf{x})$ with enough accuracy such that the code for different values of $\mathbf{x}$ are distinct is

$$l(\mathbf{x}) = \left\lceil \log \frac{1}{P(\mathbf{x})} \right\rceil + 1.$$

Remember that $l(\mathbf{x})$ is the number of bits required to encode the *entire* sequence $\mathbf{x}$. So, the average length of an arithmetic code for a sequence of length m is given by

$$\begin{aligned} l_{A^m} &= \sum P(\mathbf{x}) l(\mathbf{x}) && (4.14) \\ &= \sum P(\mathbf{x}) \left[\left\lceil \log \frac{1}{P(\mathbf{x})} \right\rceil + 1 \right] && (4.15) \\ &< \sum P(\mathbf{x}) \left[\log \frac{1}{P(\mathbf{x})} + 1 + 1 \right] && (4.16) \\ &= -\sum P(\mathbf{x}) \log P(\mathbf{x}) + 2 \sum P(\mathbf{x}) && (4.17) \\ &= H(X^m) + 2. && (4.18) \end{aligned}$$

Given that the average length is always greater than the entropy, the bounds on $l_{A^{(m)}}$ are

$$H(X^{(m)}) \le l_{A^{(m)}} < H(X^{(m)}) + 2.$$

The length per symbol, l_A, or rate of the arithmetic code is $\frac{l_{A^{(m)}}}{m}$. Therefore, the bounds on l_A are

$$\frac{H(X^{(m)})}{m} \le l_A < \frac{H(X^{(m)})}{m} + \frac{2}{m}. \qquad (4.19)$$

We have shown in Chapter 3 that for *iid* sources

$$H(X^{(m)}) = mH(X). \qquad (4.20)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000009

Therefore,

$$H(X) \leq l_A < H(X) + \frac{2}{m}. \tag{4.21}$$

By increasing the length of the sequence, we can guarantee a rate as close to the entropy as we desire.

4.4.2 Algorithm Implementation

In Section 4.3.1 we developed a recursive algorithm for the boundaries of the interval containing the tag for the sequence being encoded as

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1) \tag{4.22}$$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n) \tag{4.23}$$

where x_n is the value of the random variable corresponding to the nth observed symbol, $l^{(n)}$ is the lower limit of the tag interval at the nth iteration, and $u^{(n)}$ is the upper limit of the tag interval at the nth iteration.

Before we can implement this algorithm, there is one major problem we have to resolve. Recall that the rationale for using numbers in the interval $[0, 1)$ as a tag was that there are an infinite number of numbers in this interval. However, in practice the number of numbers that can be uniquely represented on a machine is limited by the maximum number of digits (or bits) we can use for representing the number. Consider the values of $l^{(n)}$ and $u^{(n)}$ in Example 4.3.5. As n gets larger, these values come closer and closer together. This means that in order to represent all the subintervals uniquely we need increasing precision as the length of the sequence increases. In a system with finite precision, the two values are bound to converge, and we will lose all information about the sequence from the point at which the two values converged. To avoid this situation, we need to rescale the interval. However, we have to do it in a way that will preserve the information that is being transmitted. We would also like to perform the encoding *incrementally*—that is, to transmit portions of the code as the sequence is being observed, rather than wait until the entire sequence has been observed before transmitting the first bit. The algorithm we describe in this section takes care of the problems of synchronized rescaling and incremental encoding.

As the interval becomes narrower, we have three possibilities:

1. The interval is entirely confined to the lower half of the unit interval $[0, 0.5)$.
2. The interval is entirely confined to the upper half of the unit interval $[0.5, 1.0)$.
3. The interval straddles the midpoint of the unit interval.

We will look at the third case a little later in this section. First, let us examine the first two cases. Once the interval is confined to either the upper or lower half of the unit interval, it is forever confined to that half of the unit interval. The most significant bit of the binary representation of all numbers in the interval $[0, 0.5)$ is 0, and the most significant bit of the binary representation of all numbers in the interval $[0.5, 1]$ is 1. Therefore, once the interval gets restricted to either the upper or lower half of the unit interval, the most significant bit of

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1013, Page 000010