IMAGE SEGMENTATION AND COMPRESSION USING HIDDEN MARKOV MODELS

THE KLUWER INTERNATIONAL SERIES
IN ENGINEERING AND COMPUTER SCIENCE

IMAGE SEGMENTATION AND COMPRESSION USING HIDDEN MARKOV MODELS

by

Jia Li
The Pennsylvania State University

Robert M. Gray
Stanford University

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

Library of Congress Cataloging-in-Publication Data

Image segmentation and compression using hidden Markov models / by Jia Li, Robert M. Gray.
p.cm.—(The Kluwer international series in engineering and computer science ; SECS 571)
Includes bibliographical references and index.
ISBN 978-1-4613-7027-7 ISBN 978-1-4615-4497-5 (eBook)
DOI 10.1007/978-1-4615-4497-5
1. Image processing—Digital techniques. 2. Markov processes. I. Li, Jia, 1974- II. Gray, Robert M., 1943- III. Series.

TK5102.9. I52 2000
621.36'7—dc21

00-056157

Originally published by Kluwer Academic Publishers, New York in 2000
Softcover reprint of the hardcover 1st edition 2000

Printed on acid-free paper.

Contents

Preface

Image segmentation is a process for dividing an image into its constituent parts. With applications ranging widely from remote sensing to medical image analysis, segmentation is often a key step for extracting information from images. In computer aided diagnosis, segmentation of certain medical images into regions of different tissues enables automatic measurements. Additionally, segmentation techniques are used to detect abnormalities such as tumors in medical images. Many new applications of segmentation are arising with the rapid expansion of the Internet, including content-based image retrieval that finds images with parts similar to specified examples.

Our work on segmentation started with a project on document image analysis. Based on a set of document images provided by Hewlett-Packard Laboratories, we designed an algorithm to segment document images into regions of four classes: background, text, artificial graph, and photograph. While attempting to improve accuracy, we observed that a key issue is how localized the system should be to best decide the class of an area. In general, a larger area provides more distinguishable features and is consequently easier to classify, provided it contains only one class. Even for human beings, it is often difficult to classify a small region without context. On the other hand, classification based on large blocks is crude since a large block is more likely to contain multiple classes. To overcome the conflict of over-localization and crude classification, we concentrated on a multiscale architecture, which begins classification with large blocks and no context information and then, if necessary, moves to smaller blocks using context extracted from larger blocks. The extent of localization for deciding classes is thus adjusted

adaptively. Furthermore, context information is used to compensate the ambiguity of the class caused by over-localization.

Encouraged by the performance of the document image segmentation algorithm, we attempted to extend the context-dependent classifier to a trainable system that could be tuned automatically given any set of images and their manual segmentation. Thanks to discussions with Dr. Amir Najmi, we began applying hidden Markov models (HMMs) similar to those successfully used in speech recognition to the image classification problem, at the time unaware of earlier work on hidden Markov random fields and of Markov meshes, which provide a particularly good match to the successful speech techniques. A parametric classification approach based on hidden Markov models proved to be a promising framework for context-dependent classification.

HMMs have earned their popularity from successful applications to speech recognition and genomic sequence search. An HMM is a conditionally independent process on a Markov chain. Given the state of the Markov chain at any discrete unit of time, a single observation is generated from the state according to a probability distribution depending only on the state. To infer states from observations, the optimal decision needs to be made jointly for all the units of time due to the statistical dependence among the states, which reflects the context-dependent spirit. The Markov assumption can be considered as a balance between sufficient statistical dependence among units of time and the tractability of the model.

Under a two dimensional HMM model, an image is represented by an array of feature vectors, each formed by grouping statistics of pixel intensities in a block at the corresponding location. The underlying state process of the 2-D HMM is a second order Markov mesh, a special case of a Markov random field which incorporates an ordering of pixels in a way that allows 1-D algorithms to be extended while taking advantage of true 2-D structure. Although the second order Markov mesh is not as general as many Markov random fields used in image processing, the tradeoff of the simple structure is the availability of analytic formulas for estimating model parameters. The EM algorithm, a well-known algorithm for maximum likelihood estimation based on incomplete data, yields converging analytic formulas that improve the estimation of the 2-D HMM iteratively.

We extended the 2-D HMM to a multiresolution model. With a 2-D multiresolution HMM, an image is taken to be a collection of feature vectors at several resolutions. The underlying state process is a multiscale

Markov mesh. The motivation of the extension is two-fold. The multiresolution model represents context hierarchically so that more global information can be incorporated into classification. In addition, it has been shown in many works that combining features extracted at multiple resolutions often improves segmentation.

To gain better understanding of the validity of the 2-D HMM, we applied techniques of hypothesis testing. Professor Richard A. Olshen at Stanford University introduced to us various hypothesis testing methods. The tests provide insights into how improvements on the model can be made. The last topic studied in this book is the design of joint compression and classification systems using the 2-D HMM, which is potentially important in multimedia communication for extracting explicit information about content from compressed image formats.

A large portion of the material in this book has appeared or will appear in journals and conference proceedings. Our goal has been to present this diverse material in a unified and thorough manner in order to make the methods, algorithms, and results accessible to a wider audience.

JIA LI AND ROBERT M. GRAY

Acknowledgments

We would like to acknowledge many individuals at Stanford University for their help on the research reported in this book and on the preparation of the manuscript. Both research and writing benefited from many discussions with Professor Richard A. Olshen (Biostatistics) and Dr. Amir Najmi (formerly of Electrical Engineering), who shared their great scientific insights and editorial talents with us. We also thank Professors Thomas M. Cover (Electrical Engineering) and G. Leonard Tyler (Electrical Engineering) for many suggestions on the manuscript. The research and manuscript were much improved thanks to numerous comments by reviewers of papers derived from the work and by colleagues who heard our presentations at conferences and provided useful feedback. We would like to express our gratitude to the Stanford database research group for providing computing facilities. We thank Jennifer Evans and C. Anne Murray, the editor and editorial assistant at Kluwer Academic Publishers, for making the publication of the book go very smoothly.

This book grew out of Jia Li's Ph.D research and dissertation, done within the Signal Compression and Classification Group in the Information Systems Laboratory of the Department of Electrical Engineering of Stanford University. We would like to thank all the members of the group for providing an excellent research environment and for their friendship and support. We are thankful to the financial support of the National Science Foundation and Hewlett-Packard, Inc. We are grateful to members of the Document Image Decoding Group at Xerox Palo Alto Research Center for their encouragement and support. We also acknowledge the Institute for Electrical and Electronic Engineers

(IEEE) for their generous permission to use material published in their *Transactions* in this book as detailed in specific citations in the text.

Jia Li would like to express her deepest appreciation to her family. She thanks her husband, James Z. Wang, for always being supportive while conducting his own demanding Ph.D research at Stanford University. She is immensely grateful to her parents, Shao-han Zhou and Ping-sen Li, for their love, encouragement, and education.

To our families

Chapter 1

INTRODUCTION

Better to light a candle than to curse the darkness.

—Chinese Proverb

1.1 IMAGE SEGMENTATION AND COMPRESSION

It is said that an image is worth more than a thousand words. Images play various and significant roles in our daily life. In medicine, many diagnoses are based on biomedical images derived from x-rays, computerized tomography (CT), ultra-sound (US), magnetic resonance, and other imaging modalities. Environmental scientists use aerial and satellite imagery to study pollution patterns. For entertainment, television bringing live pictures to our homes is a part of modern life. Classical images, drawings and paintings, have been giving human beings the enjoyment of art since the dawn of civilization.

In the current age of information technology, the issues of distributing images efficiently and making better use of them are of substantial concern. To achieve these goals, we rely on computers. Images are first digitized so that computers can read them. Image processing algorithms are then applied to instruct computers to handle the images automatically. Among the large variety of image processing techniques, in this book we focus on classification and compression.

Classification is a critical step in image understanding. There are two types of image classification. One is to label an entire image as a certain class, for example, distinguishing whether a photograph captures an

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

outdoor or indoor scene. The other is to divide an image into regions of different types. For example, in computer aided diagnosis, it is helpful to segment medical images into different tissues so that a specified measurement can be done automatically. The second type of classification is usually referred to as image segmentation. We are primarily concerned with supervised segmentation, in which case a segmentation system is designed based on training images that are manually segmented. Since supervised segmentation is framed as a statistical classification problem in this book, we often refer to segmentation as classification if the meaning is clear from the context.

The technique of data compression attempts to represent good quality images with as few bits as possible so that they can be transmitted or stored efficiently. For obvious reasons, compression is the technical core for making images more accessible.

1.2 OVERVIEW

In Chapter 2, background for statistical classification is provided. After reviewing important classification techniques, we describe approaches to applying those techniques to image segmentation. In particular, model-based segmentation using Markov random fields and multiresolution models is discussed. Chapter 3 is about vector quantization (VQ), a key technique for lossy data compression. VQ is reviewed as a general method for finding representative points for a set with different purposes. In particular, we describe the underlying ground shared by VQ and data classification and discuss how to benefit from this connection.

Chapter 4 presents an image classification algorithm based on 2-D hidden Markov models (HMMs), which is proposed to incorporate context information into classification. First, we introduce 1-D HMMs, which are used widely in speech recognition. We then provide the mathematical formulation of the basic assumptions of 2-D HMMs. Next, an iterative model estimation algorithm is derived from the general EM algorithm, and its computational complexity is analyzed. It is shown that computation is reduced exponentially by the forward-backward algorithm, an efficient recursive algorithm for estimating 1-D HMMs. However, since the forward-backward algorithm cannot provide polynomial-time computation in the 2-D case, an approximation algorithm is described to further reduce computational complexity. The classification algorithm based on 2-D HMMs is applied to aerial and document images.

In order to classify based upon more global context information, the 2-D HMM is extended to multiresolution in Chapter 5. The extension

allows an image to be represented by features at several resolutions, corresponding to global to local context information. Furthermore, the multiresolution model provides a hierarchical structure for progressive classification, which can speed up classification based on 2-D HMMs significantly. Comparisons are made between the multiresolution model and the single resolution model through experiments on aerial images.

An important issue is the modeling accuracy of 2-D HMMs. Obviously, the performance of algorithms using 2-D HMMs depends on the validity of those models. Although good results obtained in Chapter 4 and 5 intuitively justify the models, we examine formally the validity of the 2-D HMMs by testing of hypothesis in Chapter 6.

In Chapter 7, an algorithm for designing vector quantizers aimed at simultaneous good compression and classification is developed. A vector quantizer for the combined purpose generates indices that are mapped into both representative codewords and classes for original vectors at the receiving end. This type of quantization has the potential for several applications in the rapidly growing area of multimedia communication. For example, in image databases, in order to retrieve efficiently a particular image type of interest, it is preferable to code information about image types explicitly in the compressed bit streams of pixel intensities. Recent work, in particular the study of Bayes vector quantization (BVQ), has led to ways of optimizing vector quantizers for joint compression and classification. The approach taken by BVQ is to define a new distortion measure as a weighted sum of compression distortion and the penalty of misclassification, the latter being the Bayes risk. Our algorithm defines a new penalty for misclassification based on 2-D HMMs. This algorithm is applied to aerial images, and compared with BVQ. Chapter 8 concludes the book and provides directions to future research.

Chapter 2

STATISTICAL CLASSIFICATION

Real knowledge is to know the extent of one's ignorance.

—Confucious

Classification is the grouping of similar objects, a concept we encounter frequently in daily life. Knowledge itself is categorized into disciplines, enabling closely related topics to be studied efficiently. We restrict our interest to scientific classification, which is about discovering the optimal rule to classify a set of objects with respect to their true classes. Techniques for classification originated from biological taxonomy, dating back to Aristotle. In recent decades, classification has emerged as a prominent research field largely due to rapidly advancing computers. It is studied in a number of disciplines, such as statistics, electrical engineering, and computer science. As a result, classification is widely used in problems ranging from medical diagnosis to speech recognition and image understanding.

To formalize a classification problem, we assume that objects belong to one of M classes. The task is to predict class identities based on observed features or properties of the objects. Classification is often categorized into supervised classification and unsupervised classification. For supervised classification, a class prediction rule is formed by learning from a training set of objects with known features and class identities. Such a training set is not available for unsupervised classification, in which case the prediction rule is knowledge based. We will focus on supervised classification.

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

2.1 BAYES OPTIMAL CLASSIFICATION

Suppose features of an object are components of a vector X in space $\mathcal{X}$. The class Y belongs to a finite set $\mathcal{Y} = \{1, ..., M\}$. The training data $\{(x_i, y_i) : i = 1, 2, ..., L\}$ are independent samples drawn from the joint distribution of X and Y. Based on the training data, the aim is to develop classification rule $\kappa(x)$ that predicts class identities from the feature vectors of new test observations. The performance of $\kappa(x)$ is measured by the *Bayes risk*. Suppose the cost of labeling X as class k when the true class is j is $C_{j,k}$. The Bayes risk is defined as

$$B(\kappa) = \sum_{j=1}^{M} \sum_{k=1}^{M} C_{j,k} P(\kappa(X) = k \text{ and } Y = j) \ .$$

To minimize the Bayes risk, note that

$$\begin{aligned} B(\kappa) &= \sum_{j=1}^{M} \sum_{k=1}^{M} C_{j,k} P(\kappa(X) = k \text{ and } Y = j) \\ &= E_{X,Y} C_{Y,\kappa(X)} \\ &= E_X E_{Y|X} C_{Y,\kappa(X)} \\ &= E_X \sum_{j=1}^{M} P(Y = j \mid X) C_{j,\kappa(X)} \ . \end{aligned}$$

It is thus sufficient to minimize the conditional risk $E_{Y|X=x} C_{Y,\kappa(x)}$ for each x. The optimal classification rule, referred to as the Bayes classifier, is given by

$$\kappa(x) = \min_k{}^{-1} \sum_{j=1}^{M} P(Y = j \mid X = x) C_{j,k} \ .$$

In the special case when the Bayes risk is the probability of misclassification, which arises from the loss function

$$C_{i,j} = \begin{cases} 1 & i \neq j \\ 0 & i = j \ , \end{cases}$$

the Bayes classifier is

$$\kappa(x) = \max_k{}^{-1} P(Y = k \mid X = x) \ ,$$

which is determined by majority vote. This classifier is often called a *maximum a posteriori* (MAP) classifier. If the prior probability is uniform, this becomes a *maximum likelihood* (ML) classifier.

2.2 ALGORITHMS

Statistical classification is a rich research field with many algorithms developed and applied to various problems [66, 79, 53, 20, 77]. We review briefly several techniques in this section.

One branch of techniques aims at estimating the posterior class probability $P(Y = j \mid X = x)$. Recall that the Bayes classifier is

$$\kappa(x) = \min_k{}^{-1} \sum_{j=1}^{M} P(Y = j \mid X = x) C_{j,k} \; .$$

Accurate estimation of $P(Y = j \mid X = x)$, therefore, results in $\kappa(x)$ close to the Bayes classifier. This type of technique includes classification trees, linear and nonparametric logistic regression, and the k-nearest neighbor rule.

Classification trees owe their popularity largely to CART^R [20], developed by Breiman, Friedman, Olshen and Stone. The basic idea is to partition a feature space using a tree structure. Every leaf (terminal node) of the tree, corresponding to a cell of the partition, is assigned a class. A feature vector is predicted as the class of the cell in which the vector lies. CART grows a tree to increase the purity of leaves, which is often measured by Gini Index, $\sum_{j=1}^{M} p_j(1 - p_j)$, where p_j is the probability of being class j for feature vectors in a particular node. Entropy is another common measure of impurity. Classes are assigned by suitably weighted majority vote in each terminal node. Decision trees trained by CART yield simple rules, which often have clear physical meanings. This property is preferable especially for medical diagnosis since a decision tree explicitly describes which symptoms point to a certain decease.

For k-nearest neighbor (K-NN) classification, a training set is stored. For any new test feature vector, the k closest feature vectors in the training set are identified. A majority vote on the corresponding classes of the k neighboring vectors yields the class of the test vector. An important special case of K-NN is 1-NN (nearest neighbor classification). Cover and Hart [33] proved that for a suitably smooth underlying distribution, the NN rule achieves asymptotic probability of error no greater than $R^*(2 - MR^*/(M - 1))$, where R^* is the Bayes probability of error, and M is the number of classes.

Another group of classification techniques is based on density estimation within each class, since maximizing $P(Y = j \mid X = x)$ over j is equivalent to maximizing $f_j(x)\pi_j$, where $f_j(x)$ is the probability density function within class j and π_j is the a priori probability of class j. Both

the kernel method [18, 62] and the mixture model method [93, 47] belong to this category.

There are algorithms called prototype methods, which represent data by a set of points, or prototypes. A class is assigned to each prototype by majority vote on the associated class distribution of the prototype. A test feature vector is identified as the class of its closest prototype. K-means [66, 67, 16, 113] and learning vector quantization (LVQ) [75, 76, 77] are prototype methods. Other classification algorithms include partition-based classifiers: maximum likelihood (ML), linear discriminant, etc.

For the k-means algorithm, suppose observations are $\{x_i : 1 \leq i \leq L\}$. The goal of the k-means algorithm is to partition the observations into k groups with means $\hat{x}_1, \hat{x}_2, ..., \hat{x}_k$ such that

$$\sum_{i=1}^{L} \min_{1 \leq j \leq k} (x_i - \hat{x}_j)^2$$

is minimized. This algorithm is closely related to vector quantization, as we shall discuss in the next chapter. Kohonen *et al.* modified the k-means algorithm and proposed a variety of LVQ algorithms, including the LVQ1 considered next. Assume that the training sequence is $\{(x_i, y_i) : i = 1, 2, ..., L\}$, where x_i is the feature vector and y_i is the class. Suppose that LVQ1 starts with an initial set of centroids $\{\hat{x}_1, \hat{x}_2, ..., \hat{x}_k\}$. Each centroid $\hat{x}_i$ is assigned with class $\hat{y}_i$ by majority vote. Let (x_t, y_t) be a sample from the training data and let $\hat{x}_i(t)$ be the ith centroid computed up to sample t. Assume that $\hat{x}_c(t)$ is the nearest centroid to x_t, that is,

$$c = \min_i{}^{-1} \|x_t - \hat{x}_i(t)\| .$$

The centroids are updated by the following equation

$$\hat{x}_i(t+1) = \begin{cases} \hat{x}_i(t) + \alpha(t)\,[x_t - \hat{x}_i(t)] & i = c \text{ and } y_t = \hat{y}_i \\ \hat{x}_i(t) - \alpha(t)[x_t - \hat{x}_i(t)] & i = c \text{ and } y_t \neq \hat{y}_i \\ \hat{x}_i(t) & i \neq c\,. \end{cases}$$

The learning rate parameter $\alpha(t)$ satisfies $0 < \alpha(t) < 1$, and it may be constant or decrease monotonically with t. Implementations of various LVQ algorithms are available in the LVQ_PAK software package [78].

Statistical classification is applied to image segmentation in a variety of ways depending on different abstractions of images. One approach is to generate a feature vector for each pixel in an image. For every

pixel, statistics of pixel intensities in a window centered around it are computed as its features. A lower complexity version of this approach divides an image into blocks and generates a feature vector for each block. An image is then represented by a sequence of feature vectors. Various classification algorithms can be applied. See [100, 107, 110] for examples.

In the above approach, feature vectors are considered independent samples of a distribution, from which performance usually suffers because significant information as to context is ignored. More sophisticated algorithms take into account the statistical dependence among feature vectors when forming classification rules. Due to complexity, models are proposed to structure the statistical dependence. To segment a test image based on models estimated from training data, optimal classes are searched according to the maximum a posteriori criterion.

The idea of using context information has given rise to algorithms based on Markov random fields [73, 58], which are described in the next section.

2.3 MARKOV RANDOM FIELDS

An image is regarded as a random matrix X. Let $\mathbb{N}_m = \{(i,j) : 0 \leq i,j \leq m-1\}$ denote the $m \times m$ integer lattice; then $X = \{X_{i,j} : (i,j) \in \mathbb{N}_m\}$ denotes pixel intensities, or some other quantities depending on particular applications. Define a *neighborhood system* $\mathfrak{X} = \{\mathfrak{X}_{i,j} : (i,j) \in \mathbb{N}_m\}$, where $\mathfrak{X}_{i,j} \subset \mathbb{N}_m$ is the neighbor set of (i,j). A joint probability distribution of $\{X_{i,j}\}$ is a *Markov random field* (MRF) over $(\mathbb{N}_m, \mathfrak{X})$ if for every (i,j) and x,

$$\begin{aligned} & P(X_{i,j} = x_{i,j} \mid X_{k,l} = x_{k,l} : (k,l) \neq (i,j)) \\ = \; & P(X_{i,j} = x_{i,j} \mid X_{k,l} = x_{k,l} : (k,l) \in \mathfrak{X}_{i,j}) \,, \end{aligned}$$

that is, given all the pixels in the neighborhood of a pixel, this pixel is statistically independent of pixels outside the neighborhood.

A general theory defines MRFs on graphs. A graph $G = (T, E)$, where $T = \{t_1, t_2, ..., t_N\}$ is the finite set of vertices, and E is the set of edges. An example is provided in Fig. 2.1. Two points are called *neighbors* if they are connected by an edge. Thus E determines a neighborhood system $\mathfrak{X}_t = \{\tau : \tau \text{ is a neighbor of } t\}$. A subset $C \subset T$ is a *clique* if every pair of distinct vertices in C are neighbors. Each vertex in the graph is assigned with a random label X_t from a finite set. The

distribution of $\{X_t : t \in T\}$ is an MRF over G if

$$P(X_t = x_t \mid X_\tau = x_\tau : \tau \neq t) = P(X_t = x_t \mid X_\tau = x_\tau : \tau \in \mathfrak{X}_t) .$$

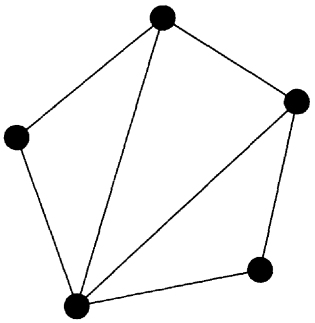


Figure 2.1. An example graph

For images, common neighborhood systems are homogeneous with the form

$$\mathfrak{X}_{i,j} = \{(k,l) \in \mathbb{N}_m : 0 < (k-i)^2 + (l-j)^2 \leq c\} .$$

For $c = 1, 2$, the neighborhood systems and their corresponding cliques are shown in Fig. 2.2.

The extension of Markovian dependence from 1-D to a general setting by the concept of MRFs is essentially due to Dobrushin [45]. Many others also worked in the direction of generalizing the Markovian property. An early contribution was that of Abend, Harley, and Kanal [1, 72] on pattern recognition, which will be described in the next section.

A considerable amount of work [58, 65, 141, 35] has been done on applying MRFs to image segmentation. In practice, to retain feasible computation, further constraints are put on the conditional distribution given all the neighboring pixels. One example is the Gaussian Markov random field for which the conditional distribution is Gaussian with parameters depending on neighboring pixels. Many applications use MRFs to model the pixel representation of images, which may not be the best for modeling the dependencies occurring in real images.

2.4 MARKOV MESH

Abend, Harley, and Kanal proposed the *Markov mesh* model, for which the Markovian dependence is *causal* (see [13, 112] for more on

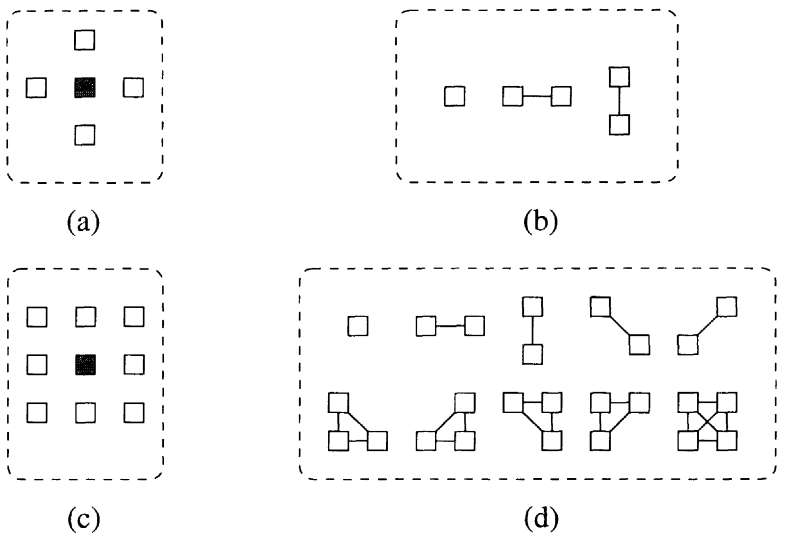


Figure 2.2. Example neighborhood systems of MRFs for images: (a) Neighborhood system with $c = 1$, (b) Cliques for neighborhood system with $c = 1$, (c) Neighborhood system with $c = 2$, (d) Cliques for neighborhood system with $c = 2$

causal MRFs). The assumption of the model is that, for all (i, j) and x,

$$\begin{aligned} & P(X_{i,j} = x_{i,j} \mid X_{k,l} = x_{k,l} : (k,l) \in A_{i,j}) \\ = \; & P(X_{i,j} = x_{i,j} \mid X_{k,l} = x_{k,l} : (k,l) \in B_{i,j}) \; . \end{aligned}$$

The set $A_{i,j}$ represents the "past" at (i, j); and the set $B_{i,j}$ is the neighborhood of (i, j) in the "past." In particular, $B_{i,j} \subset A_{i,j} \subset \{(k, l) : k < i \text{ or } l < j\}$. Two common Markov meshes are the 2nd and 3rd order Markov meshes, for which

$$\begin{aligned} \text{both:} \quad A_{i,j} &= \{(k,l) : k < i \text{ or } l < j\} \\ \text{2nd order:} \quad B_{i,j} &= \{(i-1,j), (i,j-1)\} \\ \text{3rd order:} \quad B_{i,j} &= \{(i-1,j), (i,j-1), (i-1,j-1)\} \; . \end{aligned}$$

For a block (i, j) at the boundary, if $i - 1 < 0$ or $j - 1 < 0$, $B_{i,j}$ is reduced to $\{(i, j - 1)\}$ or $\{(i - 1, j)\}$ correspondingly. Of particular interest to us is the 2nd order Markov mesh, which is assumed to be the underlying state process for the 2-D hidden Markov model described in Chapter 4. The 2nd and 3rd order Markov meshes are special Markov random fields with neighborhood systems shown in Fig. 2.3. The proof

for the 2nd order case is presented below. To simplify notation, define

$$
\begin{aligned}
& Q_{i,j}\left(\begin{bmatrix} & x_{i-1,j} \\ x_{i,j-1} & x_{i,j} \end{bmatrix}\right) \\
= \; & P(X_{i,j} = x_{i,j} \mid X_{i-1,j} = x_{i-1,j}, X_{i,j-1} = x_{i,j-1}) \ ,
\end{aligned}
$$

$$
\begin{aligned}
& Q_{i,j}\left(\begin{bmatrix} & x_{i-1,j} & x_{i-1,j+1} \\ x_{i,j-1} & x_{i,j} & x_{i,j+1} \\ x_{i+1,j-1} & x_{i+1,j} & \end{bmatrix}\right) \\
= \; & P(X_{i,j} = x_{i,j} \mid X_{i-1,j} = x_{i-1,j}, X_{i,j-1} = x_{i,j-1}) \times \\
& P(X_{i+1,j} = x_{i+1,j} \mid X_{i,j} = x_{i,j}, X_{i+1,j-1} = x_{i+1,j-1}) \times \\
& P(X_{i,j+1} = x_{i,j+1} \mid X_{i-1,j+1} = x_{i-1,j+1}, X_{i,j} = x_{i,j}) \quad .
\end{aligned}
$$

Also define

$$
\tilde{\mathbb{N}}_{i,j} = \{(k,l) : (k,l) \in \mathbb{N}, (k,l) \neq (i,j), (i+1,j), (i,j+1)\} \ .
$$

We can then derive

$$
\begin{aligned}
& P(X_{i,j} = x_{i,j} \mid X_{k,l} = x_{k,l} : (k,l) \neq (i,j), (k,l) \in \mathbb{N}) \\
= \; & \frac{P(X_{i',j'} = x_{i',j'} : (i',j') \in \mathbb{N})}{P(X_{k,l} = x_{k,l} : (k,l) \neq (i,j), (k,l) \in \mathbb{N})} \\
= \; & \frac{1}{\sum_{m=1}^{M} Q_{i,j}\left(\begin{bmatrix} & x_{i-1,j} & x_{i-1,j+1} \\ x_{i,j-1} & m & x_{i,j+1} \\ x_{i+1,j-1} & x_{i+1,j} & \end{bmatrix}\right)} \times \\
& \frac{\prod_{(i',j') \in \mathbb{N}} Q_{i',j'}\left(\begin{bmatrix} & x_{i'-1,j'} \\ x_{i',j'-1} & x_{i',j'} \end{bmatrix}\right)}{\prod_{(k,l) \in \tilde{\mathbb{N}}_{i,j}} Q_{k,l}\left(\begin{bmatrix} & x_{k-1,l} \\ x_{k,l-1} & x_{k,l} \end{bmatrix}\right)} \\
= \; & \frac{Q_{i,j}\left(\begin{bmatrix} & x_{i-1,j} & x_{i-1,j+1} \\ x_{i,j-1} & x_{i,j} & x_{i,j+1} \\ x_{i+1,j-1} & x_{i+1,j} & \end{bmatrix}\right)}{\sum_{m=1}^{M} Q_{i,j}\left(\begin{bmatrix} & x_{i-1,j} & x_{i-1,j+1} \\ x_{i,j-1} & m & x_{i,j+1} \\ x_{i+1,j-1} & x_{i+1,j} & \end{bmatrix}\right)}
\end{aligned}
$$

The above equation shows that the only random vectors that affect the conditional probability of $X_{i,j}$ are

$$
\{X_{i-1,j}, X_{i-1,j+1}, X_{i,j-1}, X_{i,j+1}, X_{i+1,j-1}, X_{i+1,j}\} \ .
$$

We thus have proved the neighborhood system for the 2nd order Markov mesh is as shown in Fig. 2.3(a).

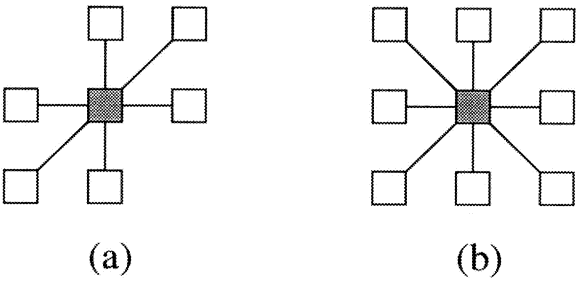


Figure 2.3. Neighborhood systems of the 2nd and 3rd order Markov meshes: (a) The 2nd order Markov mesh, (b) The 3rd order Markov mesh

2.5 MULTIRESOLUTION IMAGE CLASSIFICATION

Recent years have seen substantial interest and activity devoted to algorithms for multiresolution processing [55, 127]. One reason for this focus on image segmentation is that multiresolution processing seems to imitate the decision procedure of the human visual system (HVS) [95]. For example, when the HVS segments a picture shown in Fig. 2.4 into a foreground region (a fox) and a background region, the foreground can be located roughly by a brief glance, which is similar to viewing a low resolution image. As is shown in Fig. 2.4(b), the crude decision leaves only a small unsure area around the boundary. Further careful examination of details at the boundary results in the final decision as to what is important in the image. Both global and local information are used by the HVS, which distributes effort unevenly by looking at more ambiguous regions at higher resolutions than it devotes to other regions.

Many multiresolution approaches to image classification reflect the effort to combine global and local information. One straightforward approach is to design classifiers based on features extracted from several resolutions. Images at multiple resolutions are usually obtained by wavelet transforms [37, 95]. With the original image being the highest resolution, lower resolutions are simply the low frequency bands of wavelet transforms. See [127, 89] for examples.

Another approach exploiting multiresolution information is to form multiresolution models. Examples include the multiscale autoregressive

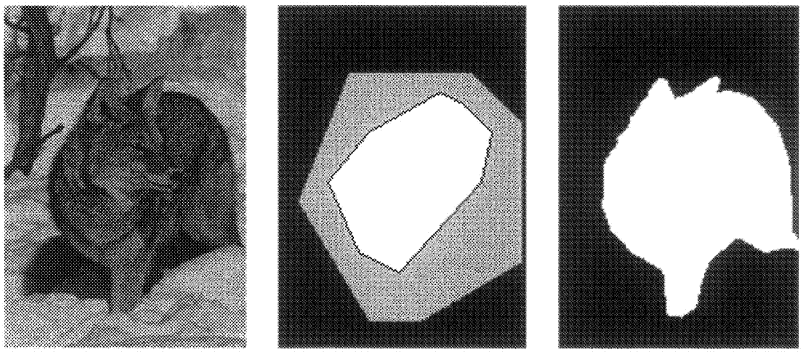


Figure 2.4. The segmentation process of the human visual system: (a) Original image, (b) A rough segmentation with the gray region being undecided, (c) The refined segmentation

model proposed by Basseville *et al.* [7] and the multiscale random field model by Bouman and Shapiro [17], both described in detail in Section 5.2.

As was mentioned, the HVS is fast as well as accurate, at least by standards of automated technologies, whereas multiresolution features or models do not necessarily benefit classification speed because information is combined from several resolutions in order to make a decision regarding a local region. Motivated by the observation that the HVS achieves fast segmentation by viewing higher resolutions selectively for ambiguous regions, we propose a multiresolution classification structure shown in Fig. 2.6.

As shown in the structure, the classifier starts with the crudest resolution and the initial context information $CI(1)$ is generated. It then enters an iterative process with every loop corresponding to classification at one resolution, and exits the iteration when the entire image is classified or the resolution exceeds R. With the resolution increased by one, the width and height of a region represented by one feature vector are reduced by half. At every resolution, if a feature vector strongly suggests the class of the region represented by it, the region is labeled as a specific class; otherwise, the region is undetermined. Classification at a higher resolution is performed if undetermined regions exist. At the beginning of classification at a particular resolution r, the context information $CI(r)$ is inherited from the context information obtained at the previous resolution, i.e., $CI(r-1)$. However, instead of being

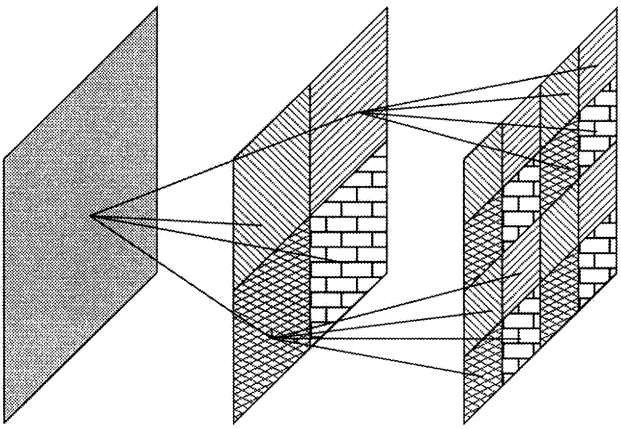


Figure 2.5. The quadtree expansion to higher resolutions

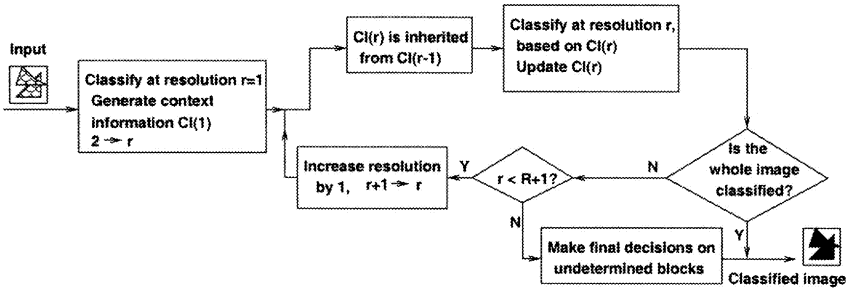


Figure 2.6. A multiresolution classification structure

fixed through the classification at resolution r, $CI(r)$ is updated with every newly classified region. By such a strategy, the classifier has more context information when it classifies ambiguous regions at higher resolutions. This structure is successfully used in a few applications [84, 86].

Chapter 3

VECTOR QUANTIZATION

Intelligence is that faculty of mind, by which order is perceived in a situation previously considered disordered.

—Haneef Fatmi

Vector quantization is both a mathematical model and a technique for data compression, the goal of which is to minimize the transmission and storage rate for a communication system while retaining the best allowable fidelity to the original. Since the use of resources such as bandwidth will generally expand to meet the resources available, it will always be beneficial to apply data compression even though the cost of bandwidth and storage capacity drops steadily. One example is information flow on the World Wide Web (WWW). With the remarkably growing popularity of the WWW, demands on data distribution rise much faster than the speed of transmission channels. Data compression thus remains at the technical core of many communication systems. A brief look at a video transmission system demonstrates this fact. According to the H.263 video coding standard, every video image in the QCIF format contains 176×144 pixels. Each pixel is described by one luminance component and two chrominance components, each stored originally in 1 byte. Since the chrominance components are subsampled every 2×2 pixels, one image requires $176 \times 144 \times 1.5$ bytes, i.e., 304128 bits, to specify. For a video sequence with 30 frames per second, the transmission rate should be above $304128 \times 30 \approx 9.12 \times 10^6$ bits per second. Since for the most up-to-date modem, the transmission rate is about 56k bits per second,

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

in order to view real time video sent by a web server, compression ratios of at least $9.12 \times 10^6/5.6 \times 10^4 \approx 163$ to 1 are needed.

According to whether a receiver can recover compressed data without error, data compression systems are categorized into lossless compression (entropy coding) and lossy compression. Entropy coding is required for text compression and especially for computer programs and financial information [12, 121, 34]. Moreover, it is often cascaded after lossy compression to further reduce bit rates. Speech and image compression [4, 21, 60] stress lossy compression because the number of bits saved by lossless compression is too limited, and human ears and eyes can tolerate a considerable amount of distortion. For most gray-scale natural photographs with 8 bits per-pixel, lossless compression cannot reduce the bit rate by more than two thirds of the original, whereas a picture compressed to 0.5 bits per-pixel by state of the art algorithms [123, 118, 140] usually looks almost the same as the original.

Vector quantization, a technique for lossy compression, is a generalization of scalar quantization to multiple dimensions. The idea of vector quantization dates back to Shannon [122] in his development of the information rate of a source subject to a fidelity criterion. The source coding system constructed to prove the rate-distortion theorem segments an input signal into consecutive nonoverlapping blocks (vectors) and maps the blocks independently into consecutive nonoverlapping channel symbols. For a sufficiently large block size, there exists such a code with rate arbitrarily close to the rate-distortion lower bound. More recent work [25, 69, 21] concentrated on building vector quantization systems with feasible complexity. The extension of scalar quantization to higher dimensions has allowed many new ideas and concepts to arise, some of which will be reviewed in this chapter.

3.1 INTRODUCTION

A vector quantizer Q of dimension k is a mapping from a subset $\mathcal{A}$ of k-dimensional Euclidean space $\Re^k$ to a finite set of $\mathcal{C}$ containing N k-dimensional vectors. The set $\mathcal{C} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_N\}$ is called the *codebook*, and the $\hat{x}_i$ *codewords*. The number of codewords in $\mathcal{C}$, N, is referred to as *codebook size*. The *rate* of a fixed-rate vector quantizer is measured by $r = \log_2 N/k$, the number of bits per vector component used to represent an input vector. A quantizer can be considered a compound function of an encoder α and a decoder β, that is, $Q(x) = \beta(\alpha(x))$, $x \in \mathcal{A}$. The encoder α maps a vector $x \in \mathcal{A}$ to an integer in set $\{1, 2, ..., N\}$. The encoder is thus associated with a partition of $\mathcal{A} = \bigcup_{i=1}^{N} \mathcal{A}_i$, where

$\mathcal{A}_i = \{x : \alpha(x) = i\}$. The decoder β is a mapping from $\{1, ..., N\}$ to $\mathcal{C} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_N\}$ given by $\beta(i) = \hat{x}_i$. In particular for image compression, a typical vector quantization system, shown in Fig. 3.1, divides an image into nonoverlapping blocks with equal size $m \times n$. Pixel intensities in each block form a vector with dimension $m \times n$. According to a certain order, the image is converted to a sequence of vectors input to the quantizer. At the decoder end, the quantized image blocks are grouped to reconstruct the image.

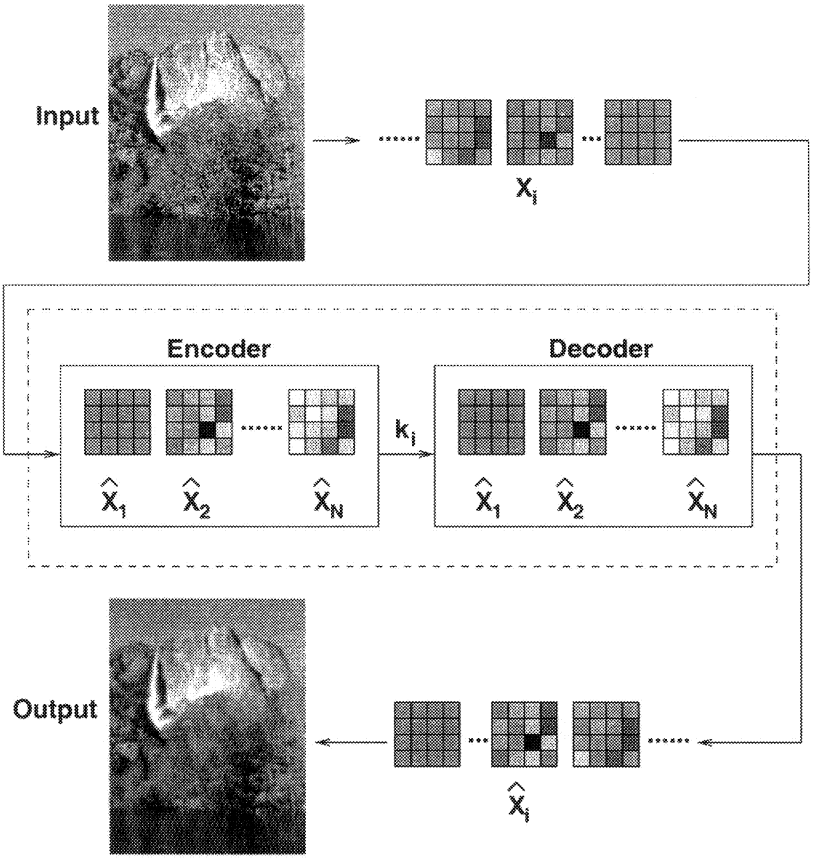


Figure 3.1. Vector quantization for image compression

An input vector to a quantizer Q is usually modeled as a random vector X with distribution f_X. The principal goal of a quantizer Q is to search for an encoder with partition $\{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_N\}$ and a decoder with codebook $\{\hat{x}_1, \hat{x}_2, ..., \hat{x}_N\}$ such that the average distortion between the

input vector and the quantized vector is minimized. With the distortion between two vectors x_1, x_2 defined as $d(x_1, x_2)$, the average distortion of quantizer Q is

$$\begin{aligned} D(\alpha, \beta) &= Ed(x, Q(x)) \\ &= \int_{\mathcal{A}} d(x, Q(x)) f_X(x) dx \ . \end{aligned}$$

Necessary conditions for the optimality of a quantizer are specified by the Lloyd conditions [88].

- For a given codebook $\mathcal{C}$, the optimal encoder satisfies

$$\alpha(x) = \min_i{}^{-1} d(x, \hat{x}_i) \ ,$$

  which is the (Euclidean) nearest neighbor partition in the special case of mean squared error distortion.

- For a given encoder with partition $\{\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_N\}$, the optimal decoder satisfies

$$\beta(i) = \min_{\hat{x}}{}^{-1} \int_{\mathcal{A}_i} d(x, \hat{x}) f_X(x) dx \ ,$$

  which is the expectation of $x \in \mathcal{A}_i$ if MSE is the distortion. The Lloyd algorithm designs quantizers by successively alternating the above optimality conditions.

A quantizer designed by the Lloyd algorithm is heavily asymmetric in terms of complexity at the encoder and the decoder. The computation of the decoder is negligible regardless of the rate and the dimension since it can be implemented by simple table lookup. The encoder, however, has to compute the distortion between each codeword and an input vector in order to choose the codeword with the minimum distortion. Suppose the rate per dimension is r and the vector dimension is k; then there are 2^{rk} codewords. The encoding complexity, roughly proportional to 2^{rk}, therefore grows exponentially with both the rate and the dimension. For image compression, the encoding complexity poses the major constraint on image block sizes. Performance is confined because statistical dependence among pixels cannot be exploited adequately.

Different approaches are taken by numerous algorithms to overcome the complexity issue. Examples include tree structured vector quantization (TSVQ), hierarchical vector quantization (HVQ), and transform vector quantization. Quantization cells of TSVQ [87, 90, 30, 117]

are obtained by recursively splitting already designed quantization cells. The tree structured partition leads to linear encoding complexity in rk. HVQ [61, 27, 129, 24, 24] implements encoding as pure table lookup. In the next section, transform VQ is discussed in detail.

3.2 TRANSFORM VQ

Transform vector quantization [64, 115, 26, 2, 3] is a constrained vector quantization scheme aimed at avoiding the computational complexity of the Lloyd algorithm. A diagram for a transform VQ system is shown in Fig. 3.2. To quantize a vector X, the quantizer first takes a linear transform of the vector, then applies quantization separately on each component of the transformed vector, usually referred to as coefficients. At the receiving end, the quantized coefficients are inversely transformed to generate the quantized vector of X.

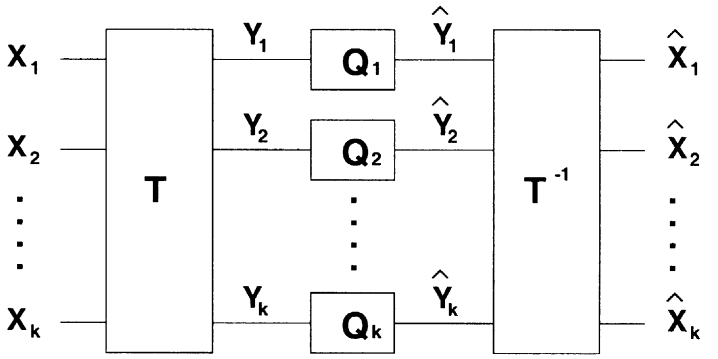


Figure 3.2. A transform vector quantizer

Although a transform vector quantizer may apply vector quantization to several coefficients grouped as one vector, scalar quantization is used much more often because a great amount of computation is saved in this way. The performance loss due to discarding vector quantization can be restrained by choosing a transform that reduces statistical redundancy among the components of a random vector via decorrelate. In addition, an appropriately chosen transform results in energy compaction for random vectors with highly correlated components, e.g., images and speech. In particular, a substantial fraction of the coefficients are likely to have near zero variances. As a result, an inverse transform close to the original vector is available even if these coefficients are replaced by their mean values.

Many international standard image and video compression algorithms use transform coding. Most state of the art algorithms are also applied to transformed data. One well-known standard compression algorithm for still images, JPEG, uses the DCT transform on 8×8 image blocks. Uniform scalar quantization and entropy coding are applied on the transformed image.

Wavelet transforms [37, 128, 95] are another important class of transforms used by many state of the art algorithms [123, 118, 140]. As with the DCT, some wavelet transforms span the original signal by orthonormal bases. Wavelet transforms are special in that the basis functions are formed by scaling and translating a single function.

3.3 VQ AS A CLUSTERING METHOD

Although the popularity of vector quantization in technology arises primarily from data compression, its applications are much broader. Mathematically, VQ addresses a fundamental problem, that is, how to select representative points of a set so that its properties are well retained. The goodness of the representative points is measured by a "loss" function. Different applications raise different definitions of "loss." For most data compression systems, in particular, the "loss" is the average mean squared error with respect to a certain probability measure on the set.

Another example of using representative points is the Monte Carlo method applied in particular to evaluate the expectations of functions. Suppose the pdf of random vector $X \in \Re^k$ is f. For simplicity, let us consider the case that f is the uniform density on C^k, where $C^k = [0,1]^k$. The expected value of $g(X)$ is

$$E(g(X)) = \int_{C^k} g(x)dx \ ,$$

which is assumed finite. The idea of Monte Carlo method is to change this analytic problem to statistical simulation. First, n independent samples x_1, ..., x_n of X are generated. The expectation of $g(X)$ is estimated by

$$\hat{E}(g(X)) = \frac{1}{n}\sum_{i=1}^{n} g(x_i) \ .$$

The sample points x_i thus serve as representative points of C^k. From the VQ point of view, a random codebook obeying the probability law f is selected to approximate the continuous random variable. This ran-

dom codebook is good with respect to f because $\hat{E}(g(X))$ converges to $E(g(X))$ with probability one as $n \to \infty$ by the strong law of large numbers. On the other hand, the Monte Carlo method is not efficient as the convergence rate in two senses is $O(n^{-1/2})$. The number-theoretic method [50, 49, 132, 48] searches for better representative points that give a faster convergence rate. The measure of "loss" used by the number-theoretic method is *discrepancy* [48]. Suppose F is the cdf of a uniform random vector on C^k, and F_n is an empirical cdf given a codebook $\mathcal{P} = \{x_1, ..., x_n\}$, that is, $F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \leq x)$, where $x_i \leq x$ means every component of x_i is less than or equal to the corresponding component of x. The discrepancy of the codebook is defined as

$$D(n, \mathcal{P}) = \sup_{x \in C^k} |F_n(x) - F(x)| .$$

For a hypothesis test, $D(n, \mathcal{P})$ is the Kolmogorov statistic [14] for the goodness of fit test of F. Define $\tilde{E}(g(X)) = \frac{1}{n}\sum_{i=1}^{n} g(x_i)$. Let components of vector x be $x_{(j)}$, that is, $x = (x_{(1)}, x_{(2)}, ..., x_{(k)})^t$. It is proved that [48]

$$|E(g(X)) - \tilde{E}(g(X))| \leq 2^k L D(n, \mathcal{P}) , \tag{3.1}$$

if the derivative

$$\frac{\partial^k g}{\partial x_{(1)} \cdots \partial x_{(k)}}$$

exists with all its lower derivatives bounded by L over C^k. The codebook $\mathcal{P}$ can be chosen so that $D(n, \mathcal{P}) = O(n^{-1}(\log n)^k)$. With this codebook, we achieve convergence rate $O(n^{-1}(\log n)^k)$, which is much better than $O(n^{-1/2})$. If the distribution of X, $f(x)$, $x \in C^k$, is not uniform, the expected value of $g(X)$ is estimated by $\tilde{E}(g(X)) = \frac{1}{n}\sum_{i=1}^{n} g(x_i) f(x_i)$. Expression (3.1) holds if the derivative of the product of f and g

$$\frac{\partial^k (gf)}{\partial x_{(1)} \cdots \partial x_{(k)}}$$

exists with all its lower derivatives bounded by L over C^k.

VQ is also naturally suited for classification by assigning a class to each quantization cell. The k-means clustering algorithm [66, 67, 16, 113], a well-known data clustering algorithm described in Section 2.2, is essentially the same as Lloyd vector quantization since the goal is simply to find a set of centroids yielding minimum average mean squared error.

In the next section, we review another classification algorithm based on vector quantization: Bayes vector quantization. Bayes vector quantization aims at joint compression and classification, which is revisited in Chapter 7.

3.4 BAYES VECTOR QUANTIZATION

The issue of joint compression and classification was considered and studied extensively by Oehler, Gray, Perlmutter, Olshen, *et al.* [100, 101, 108, 109, 106, 110, 111, 96, 22]. A vector quantizer aimed at combining compression and classification generates indices that are mapped into both representative codewords and classes for original vectors at the receiving end. This type of quantization has the potential for several applications in the rapidly growing area of multimedia communication systems. For example, in image databases, in order to retrieve efficiently a particular image type of interest, it may be required that the codes of images indicate both pixel intensities and image classes. In addition to practical motivation, there is theoretical interest in the study of such vector quantizers. It has been found that the two factors, compression and classification, are not always in conflict; a vector quantizer that minimizes the sum of misclassification penalty and a small portion of compression distortion may result in better classification than a pure classifier, and vice versa [111, 106].

The algorithm for simultaneous compression and classification developed by Oehler, Gray, Perlmutter, Olshen, *et al.* [100, 101, 108, 109, 106, 110, 111], is referred to as Bayes vector quantization (BVQ). The basic assumption is that a training sequence $\mathcal{L} = \{(x_i, y_i), i = 1, 2, ..., L\}$ is a realization of a random process $\{(X_i, Y_i), i = 1, 2, ...\}$ with (X_i, Y_i) obeying a common but unknown distribution P_{XY} on $(X, Y) \in A_X \times A_Y$. Typically P_X is absolutely continuous and is described by a pdf f_X on $\Re^k$, and P_Y is discrete, described by some pmf p_Y. For a testing sequence, we observe $\{x_i, i = 1, 2, ...\}$. The goal of BVQ is to achieve good tradeoff among average distortion, bit rate, and the Bayes risk entailed by guessing Y from the encoded X. A fixed rate Bayes vector quantizer thus consists of three components:

1. The encoder $\tilde{\alpha}$: $A_X \to \mathcal{Z}$, maps x_i to an index.

2. The decoder $\tilde{\beta}$: $\mathcal{Z} \to A_X$, maps the index into a representative vector $\hat{x}_i$ (codeword).

3. The classifier κ: $\mathcal{Z} \to A_Y$, maps the index into a class.

Unlike ordinary vector quantizers, which aim at minimizing compression distortion, the BVQ algorithm defines a new distortion measure as a weighted sum of compression distortion and a penalty for misclassification, which is usually the Bayes risk:

$$B(\tilde{\alpha}, \kappa) = \sum_{k=1}^{M} \sum_{j=1}^{M} C_{j,k} P(\kappa(\tilde{\alpha}(X)) = k \text{ and } Y = j) ,$$

where M is the number of classes, and $C_{j,k}$ is the cost of classifying $Y = k$ given that the true class is j. Normally, $C_{j,k} = 1$ when $j \neq k$, and 0 when $j = k$. In this case, the Bayes risk is simply the probability of classification error. The new distortion, referred to as the Lagrangian distortion, between an input x and an encoder output i is

$$d(x, \tilde{\beta}(i)) + \lambda \sum_{j=1}^{M} C_{j,\kappa(i)} P(Y = j \mid X = x) .$$

A Bayes vector quantizer attempts to minimize the average Lagrangian distortion

$$J(\tilde{\alpha}, \tilde{\beta}, \kappa) = D(\tilde{\alpha}, \tilde{\beta}) + \lambda B(\tilde{\alpha}, \kappa) .$$

An optimal Bayes vector quantizer satisfies the following conditions. A Lloyd-like descent algorithm can be applied to design Bayes vector quantizers by iterating the optimal conditions.

1. Optimal Decoder: Given $\tilde{\alpha}$, κ, the optimal decoder is

$$\tilde{\beta}(i) = \min_{\hat{x} \in \hat{A}_X}{}^{-1} E[d(X, \hat{x}) \mid \tilde{\alpha}(X) = i] ,$$

   which is the expected value of the input x in a quantization cell if mean squared error is used as compression distortion.

2. Optimal Classifier: Given $\tilde{\alpha}$, $\tilde{\beta}$, the optimal classifier is

$$\kappa(i) = \min_{k \in A_Y}{}^{-1} \left\{ \sum_{j=1}^{M} C_{j,k} P(Y = j \mid \tilde{\alpha}(X) = i) \right\} ,$$

   which is the Bayes optimal classifier given the encoded input.

3. Optimal Encoder: Given κ, $\tilde{\beta}$, the optimal encoder is

$$\tilde{\alpha}(x) = \min_{i}{}^{-1} \left\{ d(x, \tilde{\beta}(i)) + \lambda \sum_{j=1}^{M} C_{j,\kappa(i)} P(Y = j \mid X = x) \right\} .$$

To design a quantizer minimizing $D + \lambda B$, the conditional probability of being in a particular class given the vector is needed to evaluate the Bayes risk. Since the conditional probabilities are generally unknown in practice, an empirical distribution given by the relative frequencies of the classes in the training sequence is used. In the encoding process, as the empirical distribution obtained from the training data usually does not provide good estimation for data outside the training set, different approaches are taken to approximate Bayes risk. A simple solution, which is the approach taken by Oehler, *et al.* [99, 100, 101], is to replace Bayes risk by the compression distortion in the encoding process, which is referred to as the Bayes VQ with MSE encoding [106], since the mean squared error is normally taken as the compression distortion. The more sophisticated approach taken by Perlmutter [106, 111] generates a posterior estimation of the conditional probabilities of the classes in parallel with the design of the quantizer. This posterior estimation is used by the encoder to evaluate the Bayes risk. For some applications, Perlmutter [106, 111] showed that the Bayes VQ with posterior estimation achieves much more accurate classification than that by the Bayes VQ with MSE encoding. It is also found that adding a small portion of compression distortion to Bayes risk often benefits classification. This is consistent with the empirical fact that in CART, it is usually better to use the Gini criterion rather than Bayes risk for splitting, even though the ultimate goal of the exercise is to produce a classifier with small Bayes risk.

Chapter 4

TWO DIMENSIONAL HIDDEN MARKOV MODEL

The ability to simplify means to eliminate the unnecessary so that the necessary may speak.

—Hans Hofmann

For most block-based image classification algorithms, images are divided into blocks, and decisions are made independently for the class of each block. Choosing block sizes is consequently critical. We do not want to choose a block size too large since this obviously entails crude classification. On the other hand, if we choose a small block size, only very local properties belonging to the small block are examined in classification. The penalty then comes from losing information about surrounding regions. One solution to the conflict is forming a context-dependent classifier, an adaptive approach taken by a large variety of algorithms in signal processing, e.g., trellis coding [60]. How to incorporate "context" into classification is what is of interest to us. Previous work [55, 84] has looked into ways of taking advantage of context information to improve classification performance, such as selecting block sizes dynamically according to context. The improvement achieved demonstrates the potential of context to help classification. In this chapter, a two dimensional hidden Markov model (2-D HMM) is introduced as a general framework for context dependent classifiers.

We first review the basics of one dimensional HMMs that are used widely in speech recognition. We then discuss the extension to two dimensions. The main difficulty with applying the 2-D model to image

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

classification is computational complexity. Several approximation methods are developed to achieve computational feasibility.

4.1 BACKGROUND

The theory of hidden Markov models in one dimension (1-D HMMs) was developed in the 1960s by Baum, Eagon, Petrie, Soules, and Weiss [8, 9, 10, 11]. HMMs have earned their popularity in large part from successful application to speech recognition [5, 104, 114, 70, 31]. Underlying an HMM is a basic Markov chain [92]. In fact, an HMM is simply a "Markov Source" as defined by Shannon [120] and Gallager [57]: a conditionally independent process on a Markov chain or, equivalently, a Markov chain viewed through a memoryless noisy channel. Thus, at any discrete unit of time the system is assumed to exist in one of a finite set of states. Transitions between states take place according to a fixed probability depending only on the state of the system at the unit of time immediately preceding (1-step Markovian). In an HMM, at each unit of time a single observation is generated from the current state according to a probability distribution depending only on the state. Thus, in contrast to a Markov model, since the observation is a random function of the state, it is not in general possible to determine the current state by simply looking at the current observation. HMMs owe both their name and modeling power to the fact that these states represent abstract quantities that are themselves never observed. They correspond to "clusters" of contexts having similar probability distributions of the observation.

Suppose that there are M states $\{1, ..., M\}$ and that the probability of transition between states i and j is $a_{i,j}$. Hence the probability that at time t the system will be in the state j given that at time $t-1$ it was in state i is $a_{i,j}$. Define u_t as the observation of the system at time t. This observation is generated according to a probability distribution dependent only on the state at time t. Let $b_i(u_t)$ be the probability distribution of u_t in state i. If π_i is the probability of being in state i at time $t = 1$, then the likelihood of observing the sequence $\mathbf{u} = \{u_t\}_{t=1}^{T}$ is evaluated by summing over all possible state sequences, that is,

$$P(\mathbf{u}) = \sum_{s_1, s_2, .., s_T} \pi_{s_1} b_{s_1}(u_1) a_{s_1,s_2} b_{s_2}(u_2) \cdots a_{s_{T-1},s_T} b_{s_T}(u_T) \ ,$$

where s_t represents the state at time t. For simplicity, if it is clear from context, we will be sloppy with notation $P(\cdot)$. When the argument is continuous, $P(\cdot)$ refers to the probability density function. In most continuous density HMM systems used for speech recognition, the

density of the observation u_t in a particular state is assumed to be a Gaussian mixture distribution. Further generalization can be made by assuming single Gaussian distributions since a state with a number of mixture components can be split into substates with single Gaussian distributions. The density of the observation u_t in state i is thus

$$b_i(u_t) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma_i)}} e^{-\frac{1}{2}(u_t-\mu_i)^t \Sigma_i^{-1}(u_t-\mu_i)} ,$$

where k is the dimension of u_t, μ_i and Σ_i are the mean vector and covariance matrix respectively.

Estimation of 1-D HMM model parameters is usually performed according to the Baum-Welch algorithm [11] (later shown to be a special case of the EM algorithm [38]), which gives maximum likelihood estimation. Let $L_i(t)$ denote the conditional probability of being in state i at time t given the observations, and $H_{i,j}(t)$ denote the conditional probability of a transition from state i at time t to state j at time $t+1$ given the observations. The re-estimation formulae for the means, covariances, and the transition probabilities are

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T} L_i(t) u_t}{\sum_{t=1}^{T} L_i(t)}$$

$$\hat{\Sigma}_i = \frac{\sum_{t=1}^{T} L_i(t)(u_t - \hat{\mu}_i)(u_t - \hat{\mu}_i)^t}{\sum_{t=1}^{T} L_i(t)}$$

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} H_{i,j}(t)}{\sum_{t=1}^{T} L_i(t)} .$$

To apply the above estimation formulae, the probabilities $L_i(t)$ and $H_{i,j}(t)$ must be calculated. This is done efficiently by the so-called *forward-backward* algorithm [11]. Define the forward probability $\alpha_i(t)$ as the joint probability of observing the first t vectors u_τ, $\tau = 1, ..., t$, and being in state i at time t. This probability can be evaluated by the following recursive formula

$$\begin{aligned} \alpha_i(1) &= \pi_i b_i(u_1) \quad 1 \le i \le M \\ \alpha_i(t) &= b_i(u_t) \sum_{j=1}^{M} \alpha_j(t-1) a_{j,i} \quad 1 < t \le T,\ 1 \le i \le M . \end{aligned}$$

Define the backward probability $\beta_i(t)$ as the conditional probability of observing the vectors after time t, u_τ, $\tau = t+1, ..., T$, given the state at

time t is i. As with the forward probability, the backward probability can be evaluated using the following recursion

$$\begin{aligned}\beta_i(T) &= 1 \\ \beta_i(t) &= \sum_{j=1}^{M} a_{i,j} b_j(u_{t+1}) \beta_j(t+1) \quad 1 \le t < T \ .\end{aligned}$$

The probabilities $L_i(t)$ and $H_{i,j}(t)$ are solved by

$$\begin{aligned}L_i(t) &= P(s_t = i \mid \mathbf{u}) = \frac{P(\mathbf{u}, s_t = i)}{P(\mathbf{u})} \\ &= \frac{1}{P(\mathbf{u})} \alpha_i(t) \beta_i(t) \\ H_{i,j}(t) &= P(s_t = i, s_{t+1} = j \mid \mathbf{u}) \\ &= \frac{1}{P(\mathbf{u})} \alpha_i(t) a_{i,j} b_j(u_{t+1}) \beta_j(t+1) \ .\end{aligned}$$

For details, see any of the references on speech recognition [104, 114, 70, 139].

4.2 VITERBI TRAINING

An approximation to the maximum likelihood training provided by the Baum-Welch algorithm, often termed Viterbi training [139], is based on the assumption that each observation results from the single most likely state sequence that might have caused it. Denote the sequence of states $\mathbf{s} = \{s_t\}_{t=1}^{T}$. The state sequence with the maximum conditional probability given the observations is

$$\mathbf{s}^* = \max_{\mathbf{s}}{}^{-1} P(\mathbf{s} \mid \mathbf{u}) = \max_{\mathbf{s}}{}^{-1} P(\mathbf{s}, \mathbf{u}) \ .$$

The second equality follows as $\mathbf{u}$ is fixed for all possible state sequences. The Viterbi algorithm [130] is used to maximize $P(\mathbf{s}, \mathbf{u})$ as $\max_{\mathbf{s}} P(\mathbf{s}, \mathbf{u})$ can be computed by the recursive formulae

$$\begin{aligned}\theta_i(1) &= \pi_i b_i(u_1) \quad 1 \le i \le M \\ \theta_i(t) &= \max_j \{\theta_j(t-1) a_{j,i}\} b_i(u_t) \quad 1 < t \le T, \ 1 \le i \le M \\ \max_{\mathbf{s}} P(\mathbf{s}, \mathbf{u}) &= \max_j \theta_j(T) \ .\end{aligned}$$

The model parameters are then estimated by

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T} I(s_t^* = i) u_t}{\sum_{t=1}^{T} I(s_t^* = i)}$$

$$\hat{\Sigma}_i = \frac{\sum_{t=1}^{T} I(s_t^* = i)(u_t - \hat{\mu}_i)(u_t - \hat{\mu}_i)^t}{\sum_{t=1}^{T} I(s_t^* = i)}$$

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1} I(s_t^* = i) I(s_{t+1}^* = j)}{\sum_{t=1}^{T} I(s_t^* = i)} .$$

As usual, $I(\cdot)$ is the indicator function that equals one when the argument is true, and zero otherwise. Note that the estimation formulae above differ from the Baum-Welch formulae by substitution of $I(s_t^* = i)$ for $L_i(t)$ and $I(s_t^* = i)I(s_{t+1}^* = j)$ for $H_{i,j}(t)$. Thus, another way to view the Viterbi training is that the state sequence with the maximum a posteriori probability is assumed to be the real state sequence. With the real state sequence known, the probability of being in state i at time t, $L_i(t)$, is either 1 or 0 depending on whether the real state at t equals i, i.e., $L_i(t) = I(s_t^* = i)$. For the Baum-Welch algorithm, the assignment of observations to states is "soft" in the sense that each observation is assigned to each state with a weight $L_i(t)$. For the Viterbi training algorithm, however, the observations are uniquely assigned to the states according to the state sequence with the maximum a posteriori probability.

While more efficient computationally, Viterbi training does not in general result in maximum likelihood estimates. Note that an intermediate technique often used is to consider only the N most likely state sequences for each observation sequence for likelihood weighted training.

4.3 PREVIOUS WORK ON 2-D HMM

To apply the HMM to images, previous work extended the 1-D HMM to a pseudo 2-D HMM [80, 138]. The model is "pseudo 2-D" in the sense that it is not a fully connected 2-D HMM. The basic assumption is that there exists a set of "superstates" that are Markovian. Within each superstate there is a set of simple Markovian states. For 2-D images, first the superstate is chosen using a first order Markov transition probability based on the previous superstate. This superstate determines the simple Markov chain to be used by the entire row. A simple Markov chain is then used to generate observations in the row. Thus, superstates relate to rows and simple states to columns. In particular applications,

this model works better than the 1-D HMM [80], but we expect the pseudo 2-D HMM to be much more effective with regular images, such as documents. Since the effect of the state of a pixel on the state below it is distributed across the whole row, the pseudo 2-D model is too constrained for normal image classification.

The effort devoted to applying a truly 2-D HMM to image segmentation was first made by Devijver [39, 40, 41]. Devijver proposed to represent images as hidden Markov models with the state processes being Markov meshes, in particular, second and third order Markov meshes, the former being the focus of following sections. Applications to image segmentation, restoration, and compression were explored [41, 42, 43]. In [39], it was noted that the complexity of estimating the models or using them to perform maximum a posteriori classification is exponential in $w \times w$, the size of an image. The analytic solution for estimating the models was not discussed. Instead, computationally feasible algorithms [39, 40, 41] were developed by making additional assumptions regarding models or using locally optimal solutions. Worth noting is the deterministic relaxation algorithm [39] for searching maximum a posteriori states. The algorithm optimizes states iteratively by making local changes to current states in such a way as to increase the likelihood of the entire image. In Section 4.7, we derive analytic formulas for model estimation and show that computation is exponential in $2w$ by using a forward-backward-like algorithm. A suboptimal algorithm is described in Section 4.9 to achieve polynomial-time complexity.

Other work based on 2-D HMMs includes an algorithm for character recognition developed by Levin and Pieraccini [83], and an image decoding system over noisy channels constructed by Park and Miller [103]. In [103], 2-D HMMs with Markov meshes are used to model noisy channels, in which case underlying states, corresponding to true indices transmitted by an encoder, are observable from training data. Consequently, it is straightforward to estimate the models, whereas estimation is the main difficulty for situations when states are unobservable.

4.4 OUTLINE OF THE ALGORITHM

An outline of our algorithm is as follows:

1. Training

   (a) Divide training images into nonoverlapping blocks with equal size and extract a feature vector for each block.

   (b) Select the number of states for the 2-D HMM.

(c) Estimate model parameters based on the feature vectors and their hand-labeled classes.

2. Testing

(a) Generate feature vectors (same as step 1a) for the testing image.

(b) Search for the set of classes with maximum a posteriori probability given the feature vectors according to the trained 2-D HMM.

4.5 ASSUMPTIONS OF 2-D HMM

As in all block based classification systems, an image to be classified is divided into blocks and feature vectors are evaluated as statistics of the blocks. The image is then classified according to the feature vectors.

The 2-D HMM assumes that the feature vectors are generated by a Markov model which may change state once every block. Suppose there are M states, $\{1, ..., M\}$, the state of block (i, j) is denoted by $s_{i,j}$. The feature vector of block (i, j) is $u_{i,j}$ and the class is $c_{i,j}$. Denote $(i', j') < (i, j)$, or $(i, j) > (i', j')$, if $i' < i$, or $i' = i$ and $j' < j$, in which case we say that block (i', j') is before block (i, j). For example, in the left panel of Fig. 4.1, the blocks before (i, j) are the shaded blocks. This sense of order is the same as the raster order of row by row. We would like to point out, however, that this order is introduced only for stating the assumptions. In classification, blocks are not classified one by one in such an order. The classification algorithm attempts to find the optimal combination of classes jointly for many blocks at once. A one dimensional approach of joint classification, assuming a scanning order in classification, is usually suboptimal.

The first assumption is that

$$\begin{aligned} P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i', j') \in \Psi) &= a_{m,n,l} \ , \\ \text{where} \quad \Psi &= \{(i', j') : (i', j') < (i, j)\} \\ \text{and} \quad m = s_{i-1,j}, \ n &= s_{i,j-1}, \text{ and } l = s_{i,j} \ . \end{aligned} \tag{4.1}$$

The above assumption can be summarized by two points. First, the state $s_{i',j'}$ is a sufficient statistic for $(s_{i',j'}, u_{i',j'})$ for estimating transition probabilities, i.e., the u are conditionally memoryless. Second, the state transition is first order Markovian in a two dimensional sense. The probability of the system entering a particular state depends upon the state of the system at the adjacent observations in both horizontal and vertical directions. A transition from any state to any state is allowed. Shown in the left panel of Fig. 4.1, knowing the states of all the shaded

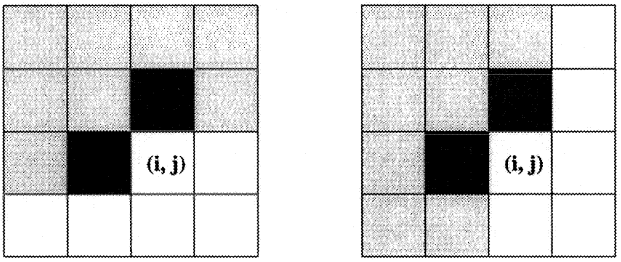


Figure 4.1. The Markovian property of transitions among states

blocks, we need only the states of the two adjacent blocks in the darker shade to calculate the transition probability to a next state. It is also assumed that there is a unique mapping from states to classes. Thus, the classes of the blocks are determined once the states are known.

The second assumption is that for every state, the feature vectors follow a Gaussian mixture distribution. Once the state of a block is known, the feature vector is conditionally independent of the other blocks. Since any state with an M-component Gaussian mixture can be split into M substates with single Gaussian distributions, the model restricts us to single Gaussian distributions. For a block with state s and feature vector u, the distribution has density

$$b_s(u) = \frac{1}{\sqrt{(2\pi)^k |\mathbf{\Sigma}_s|}} e^{-\frac{1}{2}(u-\mu_s)^t \mathbf{\Sigma}_s^{-1}(u-\mu_s)} , \tag{4.2}$$

where $\mathbf{\Sigma}_s$ is the covariance matrix and μ_s is the mean vector.

4.6 MARKOVIAN PROPERTIES

The Markovian assumption on state transitions can simplify significantly the evaluation of the probability of the states, i.e., $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$, where $\mathbb{N} = \{(i,j) : 0 \leq i < w, 0 \leq j < z\}$ is the set of indices of all the blocks in an image. To expand this probability efficiently by the conditional probability formula, we first prove that a rotated form of the two dimensional Markovian property holds given the two assumptions. Recall the definition: $(i',j') < (i,j)$ if $i' < i$ or $i' = i$, and $j' < j$. We then define a rotated relation of "$<$", denoted by "$\tilde{<}$", which specifies $(i',j') \tilde{<} (i,j)$, or $(i,j) \tilde{>} (i',j')$, if $j' < j$, or $j' = j$ and $i' < i$. An example is shown in the right panel of Fig. 4.1. To prove that

$$P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi}) = a_{m,n,l} ,$$

$$\text{where} \quad \tilde{\Psi} = \{(i',j') : (i',j') \tilde{<} (i,j)\}$$
$$\text{and} \quad m = s_{i-1,j},\ n = s_{i,j-1},\ \text{and}\ l = s_{i,j}\ ,$$

use the previous definition $\Psi = \{(i',j') : (i',j') < (i,j)\}$ and introduce the following notation:

$$\tilde{\Psi} \cup \Psi = \{(i',j') : (i',j') < (i,j) \text{ or } (i',j') \tilde{<} (i,j)\}\ ,$$
$$\tilde{\Psi} \cap \Psi = \{(i',j') : (i',j') < (i,j) \text{ and } (i',j') \tilde{<} (i,j)\}\ ,$$
$$\tilde{\Psi} - \Psi = \{(i',j') : (i',j') \tilde{<} (i,j) \text{ and } (i',j') > (i,j)\}\ .$$

The above three sets are shown in Fig. 4.2. Note that $\tilde{\Psi} = (\tilde{\Psi} \cap \Psi) \cup (\tilde{\Psi} - \Psi)$. Denote $\gamma_0 = P(s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi})$ and $\gamma_1 = P(s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi)$. We can then derive

$$\begin{aligned}
& P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi}) \\
= \ & \frac{1}{\gamma_0} P(s_{i,j}, s_{i',j'}, u_{i',j'}, s_{i'',j''}, u_{i'',j''} : \\
& (i',j') \in \tilde{\Psi} \cap \Psi, (i'',j'') \in \tilde{\Psi} - \Psi) \\
= \ & \frac{\gamma_1}{\gamma_0} P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \times \\
& P(s_{i'',j''}, u_{i'',j''} : (i'',j'') \in \tilde{\Psi} - \Psi \mid \\
& s_{i,j}, s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \qquad (4.3) \\
= \ & \frac{\gamma_1}{\gamma_0} P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}) \times \\
& P(s_{i'',j''}, u_{i'',j''} : (i'',j'') \in \tilde{\Psi} - \Psi \mid \\
& s_{i,j}, s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \qquad (4.4) \\
= \ & P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}) \times \\
& \frac{\gamma_1}{\gamma_0} P(s_{i'',j''}, u_{i'',j''} : (i'',j'') \in \tilde{\Psi} - \Psi \mid \\
& s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cap \Psi) \qquad (4.5) \\
= \ & P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}) \\
= \ & a_{m,n,l}
\end{aligned}$$

where $m = s_{i-1,j}$, $n = s_{i,j-1}$, and $l = s_{i,j}$. Equality (4.3) follows from the expansion of conditional probability. Equality (4.4) follows from the Markovian assumption. Equality (4.5) holds due to both the Markovian assumption and the assumption that the feature vector of a block is conditionally independent of other blocks given its state.

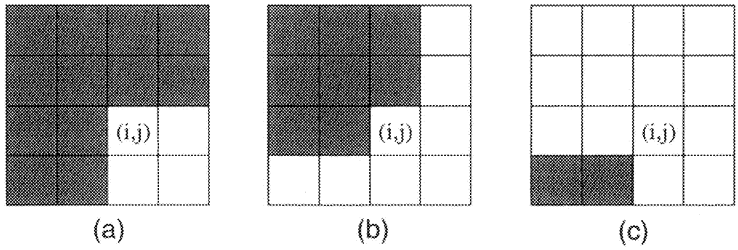


Figure 4.2. The sets of shaded blocks: (a) $\tilde{\Psi} \cup \Psi$, (b) $\tilde{\Psi} \cap \Psi$, (c) $\tilde{\Psi} - \Psi$

From the derivation, there follows an even stronger statement, that is,

$$P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cup \Psi) = P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}) \ . \quad (4.6)$$

The reason is that in the derivation, if we change $\tilde{\Psi} \cap \Psi$ to Ψ and $\tilde{\Psi}$ to $\tilde{\Psi} \cup \Psi$, all the equalities still hold. Since Equation (4.6) implies obviously the original Markovian assumption and its rotated version, we have shown the equivalence of the two assumptions:

$$\begin{aligned} P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i',j') \in \Psi) &= P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}) \ ; \\ P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i',j') \in \tilde{\Psi} \cup \Psi) &= P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}) \ . \end{aligned}$$

Note that the underlying state process is a 2nd order Markov mesh described in Section 2.3, where it is stated that this Markov mesh is causal because states in condition are the states of "past"—blocks above and to the left of a current block. The causality enables the derivation of an analytic iterative algorithm to estimate an HMM and to estimate states with the maximum a posteriori probability.

Now we are ready to simplify the expansion of $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$:

$$\begin{aligned} &P\{s_{i,j} : (i,j) \in \mathbb{N}\} \\ &\quad = P(T_0)P(T_1 \mid T_0) \cdots P(T_{w+z-2} \mid T_{w+z-3}, T_{w+z-4}, \ldots, T_0) \ , \end{aligned} \quad (4.7)$$

where T_i denotes the sequence of states for blocks on diagonal i, that is, $\{s_{i,0}, s_{i-1,1}, \cdots, s_{0,i}\}$, and w and z are the number of rows and columns respectively, as shown in Fig. 4.3.

We next show that $P(T_i \mid T_{i-1}, \ldots, T_0) = P(T_i \mid T_{i-1})$. Without loss of generality, suppose $T_i = \{s_{i,0}, s_{i-1,1}, \ldots, s_{0,i}\}$; then

$$T_{i-1} = \{s_{i-1,0}, s_{i-2,1}, \ldots, s_{0,i-1}\} \quad \text{and}$$

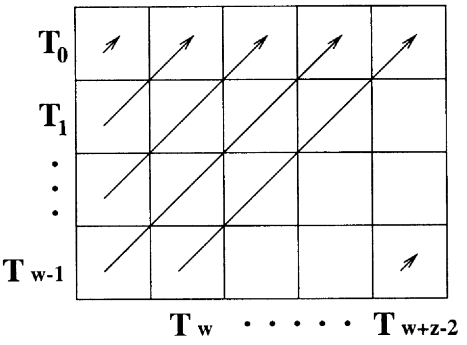


Figure 4.3. Blocks on the diagonals of an image

$$
\begin{aligned}
& P(T_i \mid T_{i-1}, \ldots, T_0) \\
= \; & P(s_{i,0}, s_{i-1,1}, \ldots, s_{0,i} \mid T_{i-1}, T_{i-2}, \ldots, T_0) \\
= \; & P(s_{i,0} \mid T_{i-1}, \ldots, T_0) P(s_{i-1,1} \mid s_{i,0}, T_{i-1}, \ldots, T_0) \\
& \cdots P(s_{0,i} \mid s_{1,i-1}, \ldots, s_{i,0}, T_{i-1}, \ldots, T_0) \\
= \; & P(s_{i,0} \mid s_{i-1,0}) P(s_{i-1,1} \mid s_{i-2,1}, s_{i-1,0}) \cdots P(s_{0,i} \mid s_{0,i-1}) \; .
\end{aligned}
$$

The last equality is obtained from Equation (4.6). Since all the states $s_{i,j}$ that appear in the conditions are in T_{i-1}, it is concluded that

$$P(T_i \mid T_{i-1}, \ldots, T_0) = P(T_i \mid T_{i-1}) \; .$$

Equation (4.7) simplifies to

$$
\begin{aligned}
& P\{s_{i,j} : (i,j) \in \mathbb{N}\} \\
= \; & P(T_0) P(T_1 \mid T_0) \cdots P(T_{w+z-2} \mid T_{w+z-3}) \; .
\end{aligned} \tag{4.8}
$$

The state sequence T_i thus serves as an "isolating" element in the expansion of $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$, which plays the role of a state at a single unit of time in the case of a one dimensional Markov model. As we shall see, this property is essential for developing the algorithm. We may notice that, besides diagonals, there exist other geometric forms that can serve as "isolating" elements, for example, state sequences on rows or columns. State sequences T_i on diagonals are preferred for computational reasons which will be explained in Section 4.9.

The task of the classifier is to estimate the 2-D HMM from training data and to classify images by finding the combination of states with the maximum a posteriori probability given the observed feature vectors.

4.7 PARAMETER ESTIMATION

For the assumed HMM, we need to estimate the following parameters: transition probabilities $a_{m,n,l}$, where $m, n, l = 1, ..., M$, and M is the total number of states, the mean vectors μ_m, and the covariance matrices $\mathbf{\Sigma}_m$ of the Gaussian distributions, $m = 1, ..., M$. We define set $\mathcal{M} = \{1, ..., M\}$. The parameters are estimated by the maximum likelihood (ML) criterion using the EM algorithm [38, 137, 11]. First, the EM algorithm as described in Dempster, Laird and Rubin [38] is introduced briefly. The algorithm is then applied to the particular case to derive a specific formula.

The EM algorithm provides an iterative computation of maximum likelihood estimation when the observed data are incomplete. The term "incomplete" reflects the fact that we need to estimate the distribution of $\mathbf{x}$, in sample space $\mathcal{X}$, but we can only observe $\mathbf{x}$ indirectly through $\mathbf{y}$, in sample space $\mathcal{Y}$. In many cases, there is a mapping $\mathbf{x} \to \mathbf{y}(\mathbf{x})$ from $\mathcal{X}$ to $\mathcal{Y}$, and $\mathbf{x}$ is only known to lie in a subset of $\mathcal{X}$, denoted by $\mathcal{X}(\mathbf{y})$, which is determined by the equation $\mathbf{y} = \mathbf{y}(\mathbf{x})$. We postulate a family of distribution $f(\mathbf{x} \mid \phi)$, with parameters $\phi \in \Omega$, on $\mathbf{x}$. The distribution of $\mathbf{y}$, $g(\mathbf{y} \mid \phi)$, can be derived as

$$g(\mathbf{y} \mid \phi) = \int_{\mathcal{X}(\mathbf{y})} f(\mathbf{x} \mid \phi) d\mathbf{x} \, .$$

The EM algorithm aims at finding a ϕ that maximizes $g(\mathbf{y} \mid \phi)$ given an observed $\mathbf{y}$.

Before describing the algorithm, we introduce a function [38]

$$Q(\phi' \mid \phi) = E(\log f(\mathbf{x} \mid \phi') \mid \mathbf{y}, \phi) \, ,$$

that is, the expected value of $\log f(\mathbf{x} \mid \phi')$ according to the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ and parameter ϕ. The expectation is assumed to exist for all pairs (ϕ', ϕ). In particular, it is assumed that $f(\mathbf{x} \mid \phi) > 0$ for $\phi \in \Omega$. The EM iteration $\phi^{(p)} \to \phi^{(p+1)}$ is defined in [38] as follows:

1. E-step: Compute $Q(\phi \mid \phi^{(p)})$.
2. M-step: Choose $\phi^{(p+1)}$ to be a value of $\phi \in \Omega$ that maximizes $Q(\phi \mid \phi^{(p)})$.

Define the following notation:

1. The set of observed feature vectors for the entire image is $\mathbf{u} = \{u_{i,j} : (i, j) \in \mathbb{N}\}$.

2. The set of states for the image is $\mathbf{s} = \{s_{i,j} : (i,j) \in \mathbb{N}\}$.

3. The set of classes for the image is $\mathbf{c} = \{c_{i,j} : (i,j) \in \mathbb{N}\}$.

4. The mapping from a state $s_{i,j}$ to its class is $C(s_{i,j})$, and the set of classes mapped from states $\mathbf{s}$ is denoted by $C(\mathbf{s})$.

Specific to our case, the complete data $\mathbf{x}$ are $\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$, and the incomplete data $\mathbf{y}$ are $\{c_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$. The function $f(\mathbf{x} \mid \phi')$ is

$$\begin{aligned} f(\mathbf{x} \mid \phi') &= P(\mathbf{s} \mid \phi')P(\mathbf{u} \mid \mathbf{s}, \phi') \\ &= P(\mathbf{s} \mid a'_{m,n,l} : m,n,l \in \mathcal{M})P(\mathbf{u} \mid \mathbf{s}, \mu'_m, \mathbf{\Sigma}'_m : m \in \mathcal{M}) \\ &= \prod_{(i,j)\in\mathbb{N}} a'_{s_{i-1,j},s_{i,j-1},s_{i,j}} \times \prod_{(i,j)\in\mathbb{N}} P(u_{i,j} \mid \mu'_{s_{i,j}}, \mathbf{\Sigma}'_{s_{i,j}}) \, . \end{aligned}$$

We then have

$$\begin{aligned} \log f(\mathbf{x} \mid \phi') &= \sum_{(i,j)\in\mathbb{N}} \log a'_{s_{i-1,j},s_{i,j-1},s_{i,j}} + \\ &\quad \sum_{(i,j)\in\mathbb{N}} \log P(u_{i,j} \mid \mu'_{s_{i,j}}, \mathbf{\Sigma}'_{s_{i,j}}) \end{aligned} \tag{4.9}$$

Given $\mathbf{y}$, $\mathbf{x}$ can only take finite number of values, corresponding to different sets of states $\mathbf{s}$ that have classes consistent with $\mathbf{y}$. The distribution of $\mathbf{x}$ is

$$\begin{aligned} P(\mathbf{x} \mid \mathbf{y}, \phi^{(p)}) &= \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) P(\mathbf{s} \mid \phi^{(p)}) P(\mathbf{u} \mid \mathbf{s}, \phi^{(p)}) \\ &= \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \times \prod_{(i,j)\in\mathbb{N}} a^{(p)}_{s_{i-1,j},s_{i,j-1},s_{i,j}} \times \\ &\quad \prod_{(i,j)\in\mathbb{N}} P(u_{i,j} \mid \mu^{(p)}_{s_{i,j}}, \mathbf{\Sigma}^{(p)}_{s_{i,j}}) \, , \end{aligned}$$

where α is a normalization constant, and $I(\cdot)$ is the obvious indicator function. From this point, we write $P(\mathbf{x} \mid \mathbf{y}, \phi^{(p)})$ as $P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)})$, assuming that all the $u_{i,j}$ in $\mathbf{x}$ are the same as those in $\mathbf{y}$, since otherwise the conditional probability of $\mathbf{x}$ given $\mathbf{y}$ is zero.

In the M-step, we set $\phi^{(p+1)}$ to the ϕ' that maximizes

$$E(\log f(\mathbf{x} \mid \phi') \mid \mathbf{y}, \phi^{(p)})$$

$$= \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \sum_{(i,j)\in\mathbb{N}} \log a'_{s_{i-1,j},s_{i,j-1},s_{i,j}} + \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \sum_{(i,j)\in\mathbb{N}} \log P(u_{i,j} \mid \mu'_{s_{i,j}}, \Sigma'_{s_{i,j}}) \qquad (4.10)$$

Equation (4.10) follows directly from (4.9). The two items in (4.10) can be maximized separately by choosing corresponding parameters. Consider the first term

$$\begin{aligned} & \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \sum_{(i,j)\in\mathbb{N}} \log a'_{s_{i-1,j},s_{i,j-1},s_{i,j}} \\ = & \sum_{\mathbf{s}} \Bigg[P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \times \\ & \sum_{m,n,l\in\mathcal{M}} \sum_{(i,j)\in\mathbb{N}} \log a'_{m,n,l} I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) \Bigg] \\ = & \sum_{m,n,l\in\mathcal{M}} \log a'_{m,n,l} \sum_{(i,j)\in\mathbb{N}} \sum_{\mathbf{s}} \Bigg[P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \times \\ & I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) \Bigg] \quad . \end{aligned} \qquad (4.11)$$

Introduce the following notation

$$H^{(p)}_{m,n,l}(i,j) = \sum_{\mathbf{s}} I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \ ,$$

which is the probability of being in state m at block $(i-1, j)$, state n at block $(i, j-1)$, and state l at block (i, j) given the observed feature vectors, classes, and model $\phi^{(p)}$. Expression (4.11) becomes

$$\sum_{m,n,l\in\mathcal{M}} \log a'_{m,n,l} \sum_{(i,j)\in\mathbb{N}} H^{(p)}_{m,n,l}(i,j) \quad .$$

The above function of $a'_{m,n,l}$ is concave. Therefore, to maximize it under the linear constraint

$$\sum_{l=1}^{M} a'_{m,n,l} = 1 \ , \text{ for all } m, n \in \mathcal{M} \ ,$$

use the Lagrangian multiplier and take derivatives with respect to $a'_{m,n,l}$. The conclusion is

$$a'_{m,n,l} \propto \sum_{(i,j)\in\mathbb{N}} H^{(p)}_{m,n,l}(i,j) \ ,$$

which in turn yields

$$a'_{m,n,l} = \frac{\sum_{(i,j)\in\mathbb{N}} H^{(p)}_{m,n,l}(i,j)}{\sum_{l'=1}^{M}\sum_{(i,j)\in\mathbb{N}} H^{(p)}_{m,n,l'}(i,j)} \ .$$

Now we discuss the maximization of the second term in (4.10).

$$\begin{aligned}
&\sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \sum_{(i,j)\in\mathbb{N}} \log P(u_{i,j} \mid \mu'_{s_{i,j}}, \Sigma'_{s_{i,j}}) \\
= &\sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \sum_{m=1}^{M} \sum_{(i,j)\in\mathbb{N}} \log P(u_{i,j} \mid \mu'_m, \Sigma'_m) I(m = s_{i,j}) \\
= &\sum_{m=1}^{M} \sum_{(i,j)\in\mathbb{N}} \sum_{\mathbf{s}} I(m = s_{i,j}) P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \log P(u_{i,j} \mid \mu'_m, \Sigma'_m) \ .
\end{aligned}$$

To simplify the above expression, let

$$L^{(p)}_m(i,j) = \sum_{\mathbf{s}} I(m = s_{i,j}) P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \ ,$$

which is the probability of being in state m at block (i,j) given the observed feature vectors, classes and model $\phi^{(p)}$. The above expression is then

$$\sum_{m=1}^{M} \sum_{(i,j)\in\mathbb{N}} L^{(p)}_m(i,j) \log P(u_{i,j} \mid \mu'_m, \Sigma'_m) \ .$$

It is known that for Gaussian distributions, the ML estimate of μ'_m is the sample average of the data, and the ML estimate of Σ'_m is the sample covariance matrix of the data [14]. Since in our case, the data are weighted by $L^{(p)}_m(i,j)$, the ML estimates of μ'_m and Σ'_m are

$$\begin{aligned}
\mu'_m &= \frac{\sum_{i,j} L^{(p)}_m(i,j) u_{i,j}}{\sum_{i,j} L^{(p)}_m(i,j)} \ , \\
\Sigma'_m &= \frac{\sum_{i,j} L^{(p)}_m(i,j)(u_{i,j} - \mu'_m)(u_{i,j} - \mu'_m)^t}{\sum_{i,j} L^{(p)}_m(i,j)} \ .
\end{aligned}$$

In summary, the algorithm iteratively improves the model estimation by the following two steps:

1. Given the current model estimation $\phi^{(p)}$, the observed feature vectors $u_{i,j}$, and classes $c_{i,j}$, the mean vectors and covariance matrices are updated by

$$\mu_m^{(p+1)} = \frac{\Sigma_{i,j} L_m^{(p)}(i,j) u_{i,j}}{\Sigma_{i,j} L_m^{(p)}(i,j)} \tag{4.12}$$

$$\Sigma_m^{(p+1)} = \frac{\Sigma_{i,j} L_m^{(p)}(i,j)(u_{i,j} - \mu_m^{(p+1)})(u_{i,j} - \mu_m^{(p+1)})^t}{\Sigma_{i,j} L_m^{(p)}(i,j)} . \tag{4.13}$$

The probability $L_m^{(p)}(i,j)$ is calculated by

$$L_m^{(p)}(i,j) = \sum_{\mathbf{s}} \Bigg[I(m = s_{i,j}) \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \times \prod_{(i',j') \in \mathbb{N}} a^{(p)}_{s_{i'-1,j'}, s_{i',j'-1}, s_{i',j'}} \times \prod_{(i',j') \in \mathbb{N}} P(u_{i',j'} \mid \mu^{(p)}_{s_{i',j'}}, \Sigma^{(p)}_{s_{i',j'}}) \Bigg] . \tag{4.14}$$

2. The transition probabilities are updated by

$$a_{m,n,l}^{(p+1)} = \frac{\sum_{i,j} H^{(p)}_{m,n,l}(i,j)}{\sum_{l'=1}^{M} \sum_{i,j} H^{(p)}_{m,n,l'}(i,j)} ,$$

where $H^{(p)}_{m,n,l}(i,j)$ is calculated by

$$\begin{aligned} H^{(p)}_{m,n,l}(i,j) \;=\; & \sum_{\mathbf{s}} \Bigg[I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) \times \\ & \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \times \prod_{(i',j') \in \mathbb{N}} a^{(p)}_{s_{i'-1,j'}, s_{i',j'-1}, s_{i',j'}} \times \\ & \prod_{(i',j') \in \mathbb{N}} P(u_{i',j'} \mid \mu^{(p)}_{s_{i',j'}}, \Sigma^{(p)}_{s_{i',j'}}) \Bigg] . \end{aligned} \tag{4.15}$$

In the case of a one dimensional HMM as used in speech recognition, the forward-backward algorithm is applied to calculate $L_m(k)$ and

$H_{m,l}(k)$ [139] efficiently. For a 2-D HMM, however, the computation of $L_m(i,j)$ and $H_{m,n,l}(i,j)$ is not feasible in view of the two dimensional transition probabilities. In the next section, we discuss why this is so and how to reduce the computational complexity.

4.8 COMPUTATIONAL COMPLEXITY

As shown in previous section, the calculation of the probabilities $H^{(p)}_{m,n,l}(i,j)$ and $L^{(p)}_m(i,j)$ is the key for the iterative estimation of the model parameters. If we compute $L^{(p)}_m(i,j)$ and $H^{(p)}_{m,n,l}(i,j)$ directly according to Equation (4.14) and (4.15), we need to consider all the combinations of states that yield the same classes as those in the training set. The enormous number of such combinations of states results in an infeasible computation. Let us take $L^{(p)}_m(i,j)$ as an example. Suppose there are M_0 states for each class and the number of blocks in an image is $w \times z$ as previously assumed, then the number of admissible combinations of states that satisfy $C(\mathbf{s}) = \mathbf{c}$ and $s_{i,j} = m$, is $M_0^{(w\times z-1)}$. When applying the HMM algorithm, although one image is often divided into many subimages such that w, or z, is the number of blocks in one column, or one row, in a subimage, we need to keep w and z sufficiently large to ensure that an adequate amount of context information is incorporated in classification. In the limit, if $w = z = 1$, the algorithm is simply a parametric classification algorithm performed independently on each block. It is normal to have $w = z = 8$. In this case, if each class has 4 states, the number of the combinations of states is $M_0^{(w\times z-1)} = 4^{63}$, which is prohibitive for a straightforward calculation of $L^{(p)}_m(i,j)$. A similar difficulty arises when estimating a one dimensional HMM. The problem is solved by a recursive calculation of forward and backward probabilities [139].

The idea of simplifying computation by the forward and backward probabilities can be extended to the two dimensional HMM. Recall Equation (4.8) in Section 4.5,

$$P\{s_{i,j} : (i,j) \in \mathbb{N}\} = P(T_0)P(T_1 \mid T_0)\cdots P(T_{w+z-2} \mid T_{w+z-3}) .$$

The fact that the state sequence T_i on a diagonal is an "isolating" element in the expansion of $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$ enables us to define the forward and backward probabilities and to evaluate them by recursive formulas.

Let us clarify notation first. In addition to the notation provided in the list in Section 4.7, the following definitions are needed:

1. The diagonal on which block (i, j) lies is denoted by $\Delta(i, j)$.

2. The feature vectors on diagonal d, $\{u_{i,j} : \Delta(i, j) = d\}$, is denoted by $\mathbf{u}(d)$.

3. The state sequence on diagonal d, $\{s_{i,j} : \Delta(i, j) = d\}$, is denoted by $\mathbf{s}(d)$.

4. For a state sequence T on diagonal d, its value at block (i, j) is $T(i, j)$.

The forward probability $\theta_T(d)$ for some model ϕ is defined as

$$\theta_T(d) = P\{\mathbf{s}(d) = T, \mathbf{u}(\tau) : \tau \leq d \mid \phi\} .$$

The forward probability $\theta_T(d)$ is the probability of observing the vectors lying on or above diagonal d and having state sequence T for blocks on diagonal d.

The backward probability $\beta_T(d)$ is defined as

$$\beta_T(d) = P\{\mathbf{u}(\tau) : \tau > d \mid \mathbf{s}(d) = T, \phi\} ,$$

which is the conditional probability of observing the vectors lying below diagonal d given the state sequence on diagonal d is T.

Similar to the case of 1-D HMM, we can derive recursive formulas for calculating $\theta_T(d)$ and $\beta_T(d)$:

$$\theta_{T_d}(d) = \sum_{T_{d-1}} \theta_{T_{d-1}}(d-1) P(T_d \mid T_{d-1}, \phi) P(\mathbf{u}(d) \mid T_d, \phi) \tag{4.16}$$

$$\beta_{T_d}(d) = \sum_{T_{d+1}} P(T_{d+1} \mid T_d, \phi) P(\mathbf{u}(d+1) \mid T_{d+1}, \phi) \beta_{T_{d+1}}(d+1) . \tag{4.17}$$

We can then compute $L_m(i, j)$ given model ϕ by

$$\begin{aligned} L_m(i, j) &= P(s_{i,j} = m \mid \mathbf{u}, \mathbf{c}, \phi) \\ &= \begin{cases} \sum_{T_d : T_d(i,j) = m} P(T_d \mid \mathbf{u}, \mathbf{c}, \phi) & C(m) = c_{i,j} \\ 0 & \text{otherwise} , \end{cases} \end{aligned}$$

where $d = \Delta(i, j)$, the diagonal of block (i, j). Consider the case $C(m) = c_{i,j}$. It is assumed in the derivation below that the summation over T_d

only covers T_d that yields consistent classes with the training data.

$$\begin{aligned} L_m(i,j) &= \sum_{T_d:T_d(i,j)=m} \frac{P(T_d, \mathbf{u} \mid \phi)}{P(\mathbf{u}, \mathbf{c} \mid \phi)} \\ &= \sum_{T_d:T_d(i,j)=m} \frac{\theta_{T_d}(\Delta(i,j))\beta_{T_d}(\Delta(i,j))}{P(\mathbf{u}, \mathbf{c} \mid \phi)} . \end{aligned} \tag{4.18}$$

In the following calculation of $H_{m,n,l}(i,j)$, the summations are always over state sequences T_d and T_{d+1} with the same classes as those in the training data.

$$\begin{aligned} \text{If } C(m) &= c_{i-1,j}, C(n) = c_{i,j-1}, C(l) = c_{i,j} , \\ H_{m,n,l}(i,j) &= P(s_{i-1,j} = m, s_{i,j-1} = n, s_{i,j} = l \mid \mathbf{u}, \mathbf{c}, \phi) \\ &= \sum_{T_d} \sum_{T_{d-1}} P(T_d, T_{d-1} \mid \mathbf{u}, \mathbf{c}, \phi) ; \\ \text{otherwise,} \quad & H_{m,n,l}(i,j) = 0 . \end{aligned}$$

We then consider the case $C(m) = c_{i-1,j}$, $C(n) = c_{i,j-1}$, and $C(l) = c_{i,j}$. In the equation below, the summations over T_d and T_{d-1} are constrained additionally to T_d satisfying $T_d(i,j) = l$ and T_{d-1} satisfying $T_{d-1}(i-1,j) = m$, $T_{d-1}(i,j-1) = n$.

$$\begin{aligned} H_{m,n,l}(i,j) = \sum_{T_d} \sum_{T_{d-1}} \Bigg[& \frac{\theta_{T_{d-1}}(\Delta(i,j)-1)}{P(\mathbf{u}, \mathbf{c} \mid \phi)} \times \\ & P(T_d \mid T_{d-1}, \phi) P(\mathbf{u}(d) \mid T_d, \phi) \beta_{T_d}(\Delta(i,j)) \Bigg] . \end{aligned} \tag{4.19}$$

Although the computation of $L_m(i,j)$ and $H_{m,n,l}(i,j)$ is significantly reduced using the forward and backward probabilities, computational complexity retains high due to the two dimensional aspects. Equation (4.16) and (4.17) for evaluating the forward and backward probabilities are summations over all state sequences on diagonal $d-1$, or $d+1$, with classes consistent with the training data. The number of state sequences to be accounted for increases exponentially with the number of blocks on a diagonal. The same difficulty exists when calculating $L_m(i,j)$ and $H_{m,n,l}(i,j)$. Consequently, an approximation is made in the calculation of $L_m(i,j)$ and $H_{m,n,l}(i,j)$ to avoid computing the backward and forward

probabilities. Recall the definitions in Section 4.7

$$H^{(p)}_{m,n,l}(i,j) = \sum_{\mathbf{s}} I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) ,$$

$$L^{(p)}_{m}(i,j) = \sum_{\mathbf{s}} I(m = s_{i,j}) P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) .$$

To simplify the calculation of $L_m(i,j)$ and $H_{m,n,l}(i,j)$, it is assumed that the single most likely state sequence accounts for virtually all the likelihood of the observations. We thus aim at finding the optimal state sequence that maximizes $P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)})$. This is the Viterbi training algorithm. The maximization of $P(\mathbf{s} \mid \mathbf{y})$ given the model $\phi^{(p)}$ can be solved by the Viterbi algorithm, described in the next section.

4.9 VARIABLE-STATE VITERBI ALGORITHM

The maximization of $P(\mathbf{s} \mid \mathbf{y})$ is equivalent to maximizing $P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$ constrained to $C(s_{i,j}) = c_{i,j}$ in training. When we apply the trained model to classify images (testing process), we also attempt to find states $\{s_{i,j} : (i,j) \in \mathbb{N}\}$ maximizing $P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$ (MAP rule). The states are then mapped into classes. In testing, since $c_{i,j}$ is to be decided, the previous constraint is removed. In the discussion, the unconstrained case, i.e., the testing situation, is considered, since in the constrained case the only difference is to shrink the search range of $s_{i,j}$ to states yielding classes $c_{i,j}$. Expand $P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$ as follows

$$\begin{aligned}
& P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\} \\
= \; & P\{s_{i,j} : (i,j) \in \mathbb{N}\} P\{u_{i,j} : (i,j) \in \mathbb{N} \mid s_{i,j} : (i,j) \in \mathbb{N}\} \\
= \; & P\{s_{i,j} : (i,j) \in \mathbb{N}\} \times \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} \mid s_{i,j}) \\
= \; & P(T_0) P(T_1 \mid T_0) P(T_2 \mid T_1) \cdots P(T_{w+z-2} \mid T_{w+z-3}) \times \\
& \prod_{(i,j) \in \mathbb{N}} P(u_{i,j} \mid s_{i,j}), \qquad (4.20)
\end{aligned}$$

where T_d denotes the sequence of states for blocks lying on diagonal d. The last equality comes from Equation (4.7).

Since T_d serves as an "isolating" element in the expansion of $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$, the Viterbi algorithm can be applied straightforwardly to find the states maximizing the likelihood $P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$. The

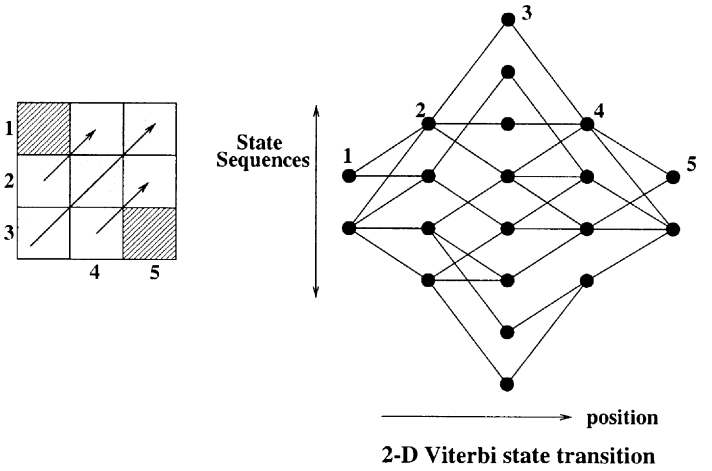


Figure 4.4. The variable-state Viterbi algorithm

difference from the normal Viterbi algorithm is that the number of possible sequences of states at every position in the Viterbi transition diagram increases exponentially with the increase of blocks in T_d. If there are M states, the amount of computation and memory are both in the order of M^ν, where ν is the number of states in T_d. Fig. 4.4 shows an example. Hence, this version of the Viterbi algorithm is referred to as a variable-state Viterbi algorithm.

The fact that in the two dimension case, only a sequence of states on a diagonal, rather than a single block, can serve as an "isolating" element in the expansion of $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$ causes computational infeasibility for the variable-state Viterbi algorithm. To reduce computation, at every position of the Viterbi transition diagram, the algorithm only uses N out of all the M^ν sequences of states, shown in Fig. 4.5. The paths are constrained to pass one of these N nodes. To choose the N sequences of states, the algorithm temporarily ignores inter-block statistical dependence. The posterior probability of a sequence of states on the diagonal is thus evaluated as a product of the posterior probability of every block. Then, the N sequences with the largest posterior probabilities are selected as the N nodes allowed in the Viterbi transition diagram. The implicit assumption in doing this is that the optimal state sequence (the node in the optimal path of the Viterbi transition diagram) yields high likelihood when the blocks are treated independently. It is also expected that when the optimal state sequence is not among

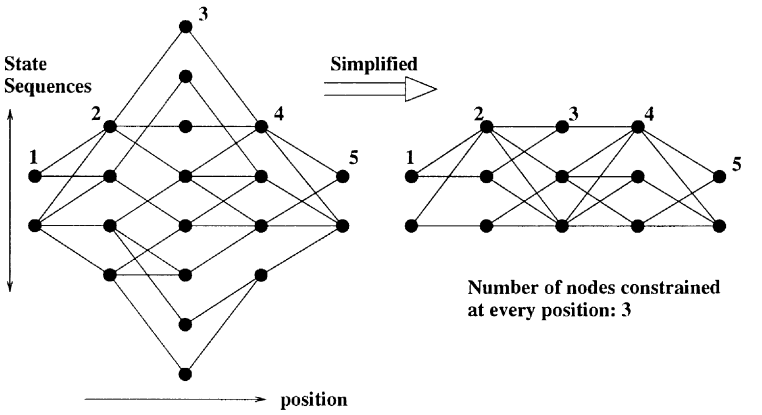


Figure 4.5. The path-constrained Viterbi algorithm

the N nodes, the chosen suboptimal state sequence coincides with the optimal sequence at most of the blocks. The sub-optimal version of the algorithm is referred to as the path-constrained variable-state Viterbi algorithm. This algorithm is different from the M-algorithm introduced for source coding by Jelinek and Anderson [71] since the N nodes are pre-selected to avoid calculating the posterior probabilities of all the M^ν state sequences.

As mentioned in Section 4.6, state sequences on rows or columns can also serve as "isolating" elements in the expansion of $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$. Diagonals are used for the expansion because intuition suggests that the pre-selection of N nodes by neglecting dependence among states degrades performance less than would doing the same for a row or a column. Remember that blocks on a diagonal are not geometrically as close as blocks on a row or a column.

A fast algorithm is developed for identifying such N sequences of states. It is unnecessary to calculate the posterior probabilities of all the M^ν sequences in order to find the largest N from them. In the following discussion, we consider the maximization of the joint log likelihood of states and feature vectors, since maximizing the posterior probability of the states given the feature vectors is equivalent to maximizing the joint log likelihood. Also, note that the log likelihood of a sequence of states is equal to the sum of the log likelihoods of the individual states because dependence among states is neglected in the pre-selection. Suppose there

are ν blocks on a diagonal, and each block exists in one of M states. The log likelihood of block i being in state m is $\gamma_{i,m}$. The pre-selection of the N nodes is simply to find N state sequences $\{s_i : i = 1, ..., \nu\}$ with the largest $\sum_{i=1}^{\nu} \gamma_{i,s_i}$. Suppose we want to find the state sequence $\max^{-1}_{s_i:i=1,...,\nu} \sum_{i=1}^{\nu} \gamma_{i,s_i}$; it is unnecessary to calculate $\sum_{i=1}^{\nu} \gamma_{i,s_i}$ for all the M^ν state sequences. We need only to find $\max^{-1}_{s_i} \gamma_{i,s_i}$ for each i, then the optimal state sequence is $\{\max^{-1}_{s_i} \gamma_{i,s_i} : i = 1, ..., \nu\}$. The idea can be extended to finding the N sequences with the largest log likelihoods.

To ensure that the path-constrained variable-state Viterbi algorithm yields results sufficiently close to the optimal solutions, the parameter N should be correspondingly larger when more blocks are contained in the 2-D Markov chain. Therefore, an image is usually divided into subimages to avoid too many blocks in one chain. Every subimage is assumed to be a 2-D Markov chain, but the dependence among subimages is not considered. On the other hand, to incorporate any preassigned amount of context information in classification, the subimages must contain a sufficient number of blocks. The selection of the parameters will be discussed in the section on experiments.

4.10 INTRA- AND INTER-BLOCK FEATURES

Choosing features is a critical issue in classification because features often set the limits of classification performance. For a classifier based on the 2-D HMM, both intra-block features and inter-block features are used. The intra-block features are defined using pixel intensities in a block to reflect its statistical properties. Features selected vary greatly for different applications. Widely used examples include moments in the spatial domain or frequency domain and coefficients of transformations, e.g., the discrete cosine transform (DCT).

The inter-block features are defined to represent relations between two blocks, for example, the difference between the average intensities of the two blocks. The use of the inter-block features is similar to that of delta and acceleration coefficients in speech recognition, in which there is ample empirical justification for the inclusion of these features [139]. The motivation for us to use inter-block features is to compensate for the strictness of the 2-D HMM. The 2-D HMM assumes constant state transition probabilities. In practice, however, we expect that a transition to a state may depend on mutual properties of two blocks. For instance, if the two blocks have close intensities, then they may be more likely to be in the same state. Since it is too complicated to estimate models

with transition probabilities being functions, we preserve the constant transition probabilities and offset this assumption somewhat by incorporating the mutual properties into feature vectors in such a way that they can influence the determination of states through posterior probabilities. In the 2-D HMM, since the states of adjacent blocks right above or to the left of a block determine the transition probability to a new state, mutual properties between the current block and these two neighboring blocks are used as inter-block features.

4.11 AERIAL IMAGE SEGMENTATION

4.11.1 FEATURES

The first application of the 2-D HMM algorithm is the segmentation into man-made and natural regions of aerial images. The images are 512×512 gray-scale images with 8 bits per-pixel (bpp). They are the aerial images of the San Francisco Bay area provided by TRW (formerly ESL, Inc.) [99]. The data set used contains six images, whose hand-labeled segmented images are used as the truth set of classes. The six images and their hand-labeled classes are shown in Fig. 4.6.

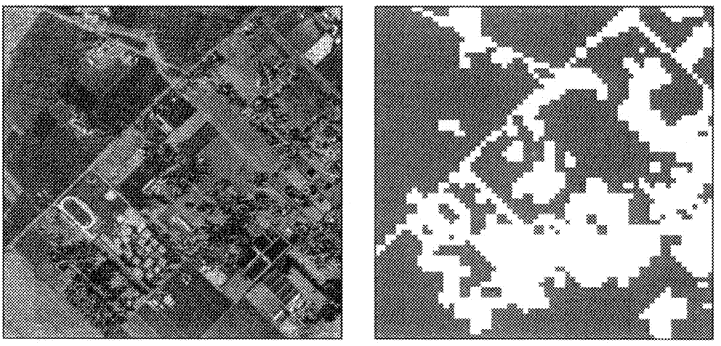
The images were divided into 4×4 blocks, and DCT coefficients or averages over some of them as described shortly were used as features. There are 6 such features. The reason to use DCT coefficients is that the different energy distributions in the frequency domain distinguish the two classes better. Denote the DCT coefficients for a 4×4 block by $\{D_{i,j} : i, j \in (0, 1, 2, 3)\}$, shown by Fig. 4.7. The definitions of the 6 features are:

1. $f_1 = D_{0,0}$; $f_2 = |D_{1,0}|$; $f_3 = |D_{0,1}|$;
2. $f_4 = \sum_{i=2}^{3} \sum_{j=0}^{1} |D_{i,j}|/4$;
3. $f_5 = \sum_{i=0}^{1} \sum_{j=2}^{3} |D_{i,j}|/4$;
4. $f_6 = \sum_{i=2}^{3} \sum_{j=2}^{3} |D_{i,j}|/4$.

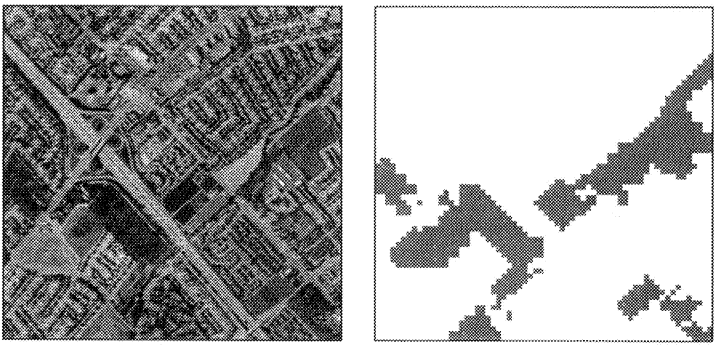
DCT coefficients at various frequencies reflect variation patterns in a block. They are more efficient than space domain pixel intensities for distinguishing the classes. Alternative features based on frequency properties include wavelet coefficients.

In addition to the intra-block features computed from pixels within a block, the spatial derivatives of the average intensity values of blocks were used as inter-block features. In particular, the spatial derivative

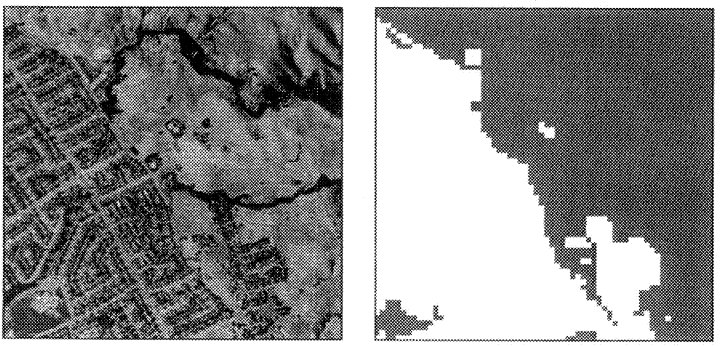
refers to the difference between the average intensity of a block and that of the block's upper neighbor or left neighbor.



(a)



(b)



(c)

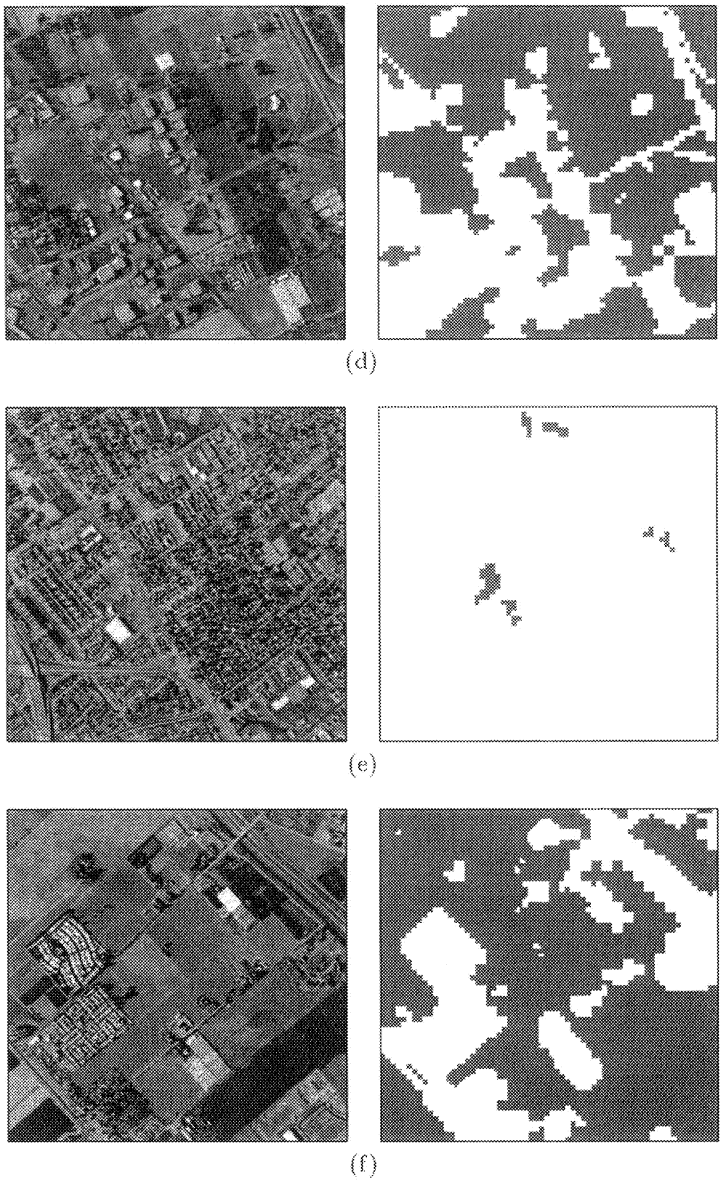


Figure 4.6. Aerial images: (a)~(f) Image 1~6. Left: Original 8 bpp images, Right: Hand-labeled classified images. White: man-made, Gray: natural

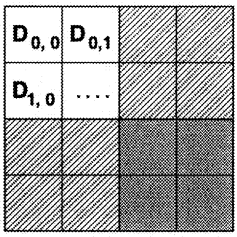


Figure 4.7. DCT coefficients of a 4 × 4 image block

4.11.2 RESULTS

Six-fold cross-validation [126] was used to evaluate the algorithms tested. For each iteration, one image was used as test data and the other five were used as training data. Hidden Markov models with different number of states were trained and tested. Experiments show that models with 4 to 6 states for the natural class, and 7 to 10 states for the man-made class yield very similar results. For the result to be given in this section, a model with 5 states for the natural class and 9 states for the man-made class was used. Setting too many states for each class results in worse classification for two reasons: the model closest to the truth might not be the most sophisticated; and more complicated models require a larger training set. With a fixed training set, the accuracy of estimation degrades with the increase of parameters.

When training and applying the HMM using the path-constrained 2-D Viterbi algorithm, an image was divided into square subimages each containing 16 blocks. The subimages were considered separate Markov chains. The number of nodes constrained at each position in the Viterbi transition diagram, N, was chosen as 32 for the result provided in this section. We experimented with several Ns. For N from 2 to 16, the performance is gradually enhanced. For N greater than 16, the results, with minor differences, start showing a convergence trend. The classification error rate with $N = 16$ is about 0.26% higher than that with $N = 32$. As classification time is spent mainly on the Viterbi searching process, and the Viterbi searching time increases at the order of the second power of the number of nodes at every transition step; the classification time is roughly proportional to N^2. Experiments were performed on a Pentium Pro 230MHz PC with the LINUX operating system. The average user CPU time to classify an aerial image was 18 seconds for $N = 8$, 59 seconds for $N = 16$, and 200 seconds for $N = 32$. A more detailed discussion of computational complexity is in Section 5.5.

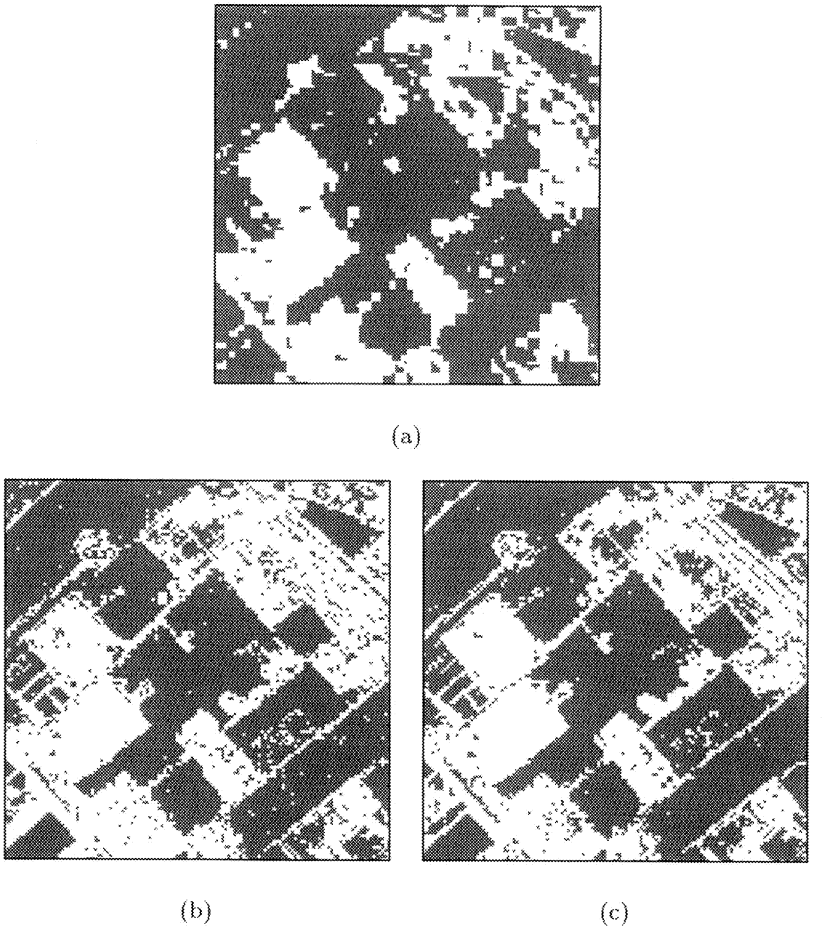


Figure 4.8. Comparison of the classification results of 2-D HMM, CART, and LVQ1 for an aerial image: (a) HMM with classification error rate 13.39%, (b) CART using both inter- and intra-block features with classification error rate 20.29%, (c) LVQ1 using both inter- and intra-block features with classification error rate 18.13%. White: man-made, Gray: natural

The result based on 2-D HMM was compared with those obtained by CART [20] and LVQ1 [77]. As described in Section 2.2, CART was developed for general decision tree design, which includes context dependent classification. As the goal here is to explore how much context improves classification by the 2-D HMM algorithm, CART was applied in a context independent manner to set a benchmark for comparison. In the training process, CART was used to partition feature vectors formed

for each image block. Images were then classified by tracing their feature vectors independently through the decision tree. Two types of decision trees were trained with CART. One was trained on both inter- and intra-block features; the other was trained on only intra-block features. These two classifiers are referred to as CART 1 and CART 2 respectively. CART 1 incorporates context information implicitly through inter-block features, but not as directly and extensively as does the 2-D HMM algorithm.

To compare with LVQ1 described in Section 2.2, we used programs provided by the LVQ_PAK software package [78]. As with CART 1, classification was based on both inter- and intra-block features. The total number of centroids for the two classes is 1024, and the number for each class is proportional to the empirical a priori probabilities of the classes. Other parameters were set by default.

The classification results for 2-D HMM, CART 1, CART 2, and LVQ1 are shown in Table 4.1. Suppose the man-made class is the target class, or positive class. Sensitivity is the true positive ratio, i.e., the probability of detecting positive given the truth is positive. Specificity is the true negative ratio, i.e., the probability of accepting negative given the truth is negative. Predictive value positive (PVP) is the probability of being truly positive given a positive detection of the classifier. The average percentage of classification error with CART 2 is 24.08%. CART 1 improves the error rate to 21.58%. LVQ1 achieves an error rate of 21.83%, which is close to the result of CART 1. The 2-D HMM algorithm further reduces the error rate to 18.80%. The classification results for Image 6, the image shown in Fig. 4.6(f), are given in Fig. 4.8. A visual difference to note is that the results of CART 1 and LVQ1 appear "noisy" due to scattered errors caused by classifying blocks independently.

The segmentation of aerial images was also studied by Oehler [99] and Perlmutter [106]. In both cases, the Bayes vector quantizer (BVQ) [99, 106, 100, 101] is used as a classifier. With the same set of images and six-fold cross-validation, the best result of simulations with different parameters provides an average classification error rate of roughly 21.5% [106], comparable to CART 1.

4.12 DOCUMENT IMAGE SEGMENTATION

4.12.1 RELATED WORK

The second application of the 2-D HMM algorithm is the segmentation of document images into text and photograph. Photograph refers to continuous-tone images such as scanned pictures; and text refers to

Algorithm	Iteration	sensitivity	specificity	PVP	P_e
2-D HMM	1	0.6250	0.9171	0.8146	0.1904
	2	0.8717	0.6141	0.9074	0.1765
	3	0.9188	0.6974	0.7114	0.2034
	4	0.5543	0.9201	0.8446	0.2405
	5	0.8152	0.9196	0.9986	0.1834
	6	0.8919	0.8533	0.7518	0.1339
	Ave	0.7795	0.8203	0.8381	0.1880
CART 1	1	0.7870	0.7660	0.6622	0.2263
	2	0.8594	0.6473	0.9136	0.1803
	3	0.9587	0.5083	0.6128	0.2899
	4	0.7676	0.7310	0.6908	0.2529
	5	0.8574	0.8705	0.9979	0.1425
	6	0.8867	0.7525	0.6408	0.2029
	Ave	0.8528	0.7126	0.7530	0.2158
CART 2	1	0.7281	0.7812	0.6598	0.2383
	2	0.7415	0.7611	0.9309	0.2548
	3	0.9505	0.4950	0.6044	0.3009
	4	0.7265	0.7279	0.6765	0.2727
	5	0.8304	0.8929	0.9982	0.1687
	6	0.8810	0.7457	0.6331	0.2093
	Ave	0.8097	0.7340	0.7505	0.2408
LVQ1	1	0.7139	0.8247	0.7035	0.2161
	2	0.8409	0.6666	0.9163	0.1918
	3	0.9505	0.4950	0.6044	0.2846
	4	0.7104	0.7824	0.7189	0.2492
	5	0.8120	0.8973	0.9983	0.1868
	6	0.8847	0.7857	0.6730	0.1813
	Ave	0.8187	0.7419	0.7691	0.2183

Table 4.1. Comparison of classification performance

normal text, tables, and artificial graphs generated by computer software [85]. We may refer to the normal text as text for simplicity if it is clear from context later on. Images experimented with are 8 bpp gray-scale images. An example image and its segmented image are shown in Fig. 4.12. This type of classification is useful in a printing process for separately rendering different local image types. It is also a tool for efficient extraction of data from image databases.

Previous work on gray-scale document image segmentation includes Chaddha [23], Williams [135], Perlmutter [107, 106], and Ohuchi [102]. Thresholding is used to distinguish image types in [23]. In [135], a

modified quadratic neural network [97] is used for classifying features. In [107, 106], the Bayes VQ algorithm is applied.

Another type of document image classification studied more often is the segmentation of half-tone (binary) document images [124, 52, 54, 131, 133, 119, 46]. For binary images, run-length statistics, e.g., the average horizontal run-length of black dots, are important for distinguishing text and photograph [124, 136, 46]. An algorithm proposed by Shih [124] combines techniques in Wong [136] and Fisher [52]. A run-length smoothing algorithm is first applied to merge black dots that are close to each other. Under the assumption that different image types are spaced sufficiently far apart and aligned well, the run-length smoothing algorithm decomposes a document image into a set of rectangular blocks. Features extracted for each block are fed into a classifier that applies a set of rules to decide the class of the block.

4.12.2 FEATURE EXTRACTION

For document images, wavelet transforms [116, 91] were used to form features. Wavelet transforms have proven useful for classification in many applications such as classifying texture [127, 89] and detecting abnormalities in medical images [44, 134]. In classification algorithms, the principal wavelet-based approach is to define features according to the statistical behavior of wavelet coefficients. Moments of wavelet coefficients are the most commonly used [127, 89, 44, 134]. In this application, however, direct attention is paid to the sample distribution or histogram pattern of wavelet coefficients. Features are defined according to the shape of these histograms.

Distribution of Wavelet Coefficients. It has been observed based on many document images that for photographs, wavelet coefficients in high frequency bands, i.e., LH, HL, and HH bands [128] as shown in Figure 4.9, tend to follow Laplacian distributions. Although this approximation is controversial in some applications, it will be seen to work quite well as a means of distinguishing continuous-tone images from graph and text by means of the goodness of fit. To visualize the different distribution patterns of the three image types, the histograms of Haar wavelet coefficients in the LH band for one photograph image, one graph image, and one text image are plotted in Figure 4.10.

In addition to the goodness of fit to Laplacian distributions, another important difference to note in Figure 4.10 is the continuity of the observed distributions. The histograms of the graph image and the text im-

age suggest that the coefficients are highly concentrated on a few discrete values, but the histogram of the photograph image shows much better continuity of distribution. In the special case shown in Figure 4.10, the histogram of the text image contains five bins far apart from each other because the text image has bi-level intensities. For the graph image, a very high percentage of data locate around zero. Although the concentration of data around zero is not absolute, the amount of nonzero data is negligible. For the photograph image, there does not exist any value that similarly dominates. Instead, the histogram peaks at zero and attenuates roughly exponentially. It is worth pointing out that in practice histograms of the three image types are usually not as extreme as the examples shown here. Ambiguity occurs more with graph because it is intermediate between photograph and text.

Based on the observations, two features are extracted according to the histograms of wavelet coefficients in high frequency bands. The first one is the goodness of match between the observed distribution and the Laplacian distribution. The second one is a measure of the likelihood of wavelet coefficients being composed by highly concentrated values.

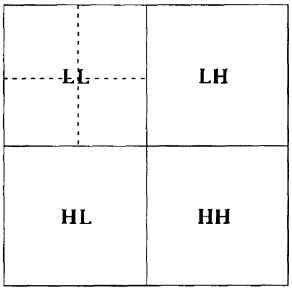


Figure 4.9. The decomposition of an image into four frequency bands by wavelet transforms

Goodness of Fit to the Laplacian Distribution. In order to measure the goodness of match between an observed distribution and a Laplacian distribution, the χ^2 test [125] normalized by the sample size N, denoted by $\bar{\chi}^2$, is used. The χ^2 test is a widely applicable measure of how well a set of observed data matches a specified theoretical distribution.

Given a random variable X, assume that its theoretical probability density function is $p(t)$, $t \in \Re$. The samples of X are $\{x_1, x_2, \ldots, x_N\}$.

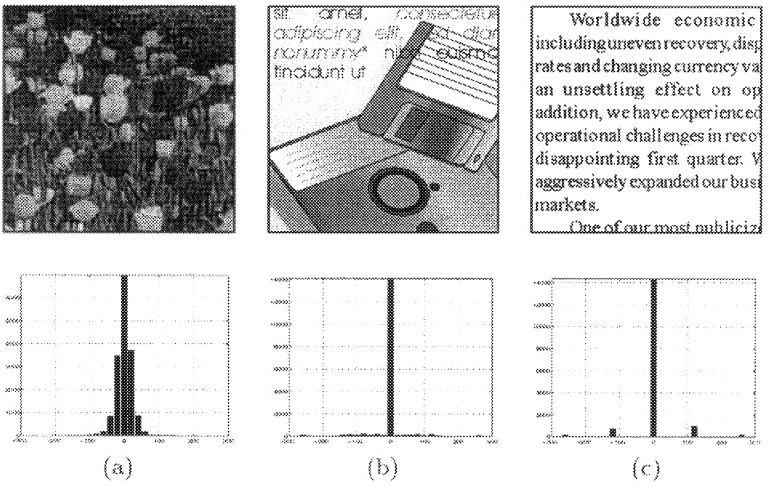


Figure 4.10. The histograms of Haar wavelet coefficients in the LH band: (a) photograph image, (b) graph image, (c) text image. First row: original images. Second row: histograms.

To test whether the samples are drawn from the distribution $p(t)$, the sample set is divided into ν categories. The number of x_j's in category i is denoted by m_i. Usually, the categories are consecutive intervals in the domain of the data. Specifically, if we denote category i by C_i, then

$$C_i = \{x : \alpha_i < x \leq \alpha_{i+1}\}\ ,\ i = 1, \ldots, \nu,\ \alpha_1 < \alpha_2 < \ldots < \alpha_{\nu+1}\ .$$

The relative frequency f_i for the ith category C_i is

$$f_i = m_i/N\ .$$

According to the theoretical distribution, the expected frequency F_i for C_i is

$$F_i = \int_{\alpha_i}^{\alpha_{i+1}} p(t)dt\ .$$

Thus, the expected count for x_i being in C_i is

$$M_i = NF_i\ .$$

The χ^2 test criterion is defined as

$$\chi^2 = \sum_{i=1}^{\nu}(m_i - M_i)^2/M_i\ .$$

Pearson [125] showed that if the observed data are randomly drawn from the theoretical distribution, then the test criterion follows the theoretical χ^2 distribution in large samples. If the observations come from some other distribution, the observed f_i tends to deviate from the expected F_i and the computed χ^2 becomes large.

If we test the null hypothesis [125] H_0: the observations are randomly drawn from a specified theoretical distribution, a computed χ^2 greater than $\chi^2_{0.050}$ causes rejection of H_0 at the 5% significance level [125]. The value of $\chi^2_{0.050}$ is calculated according to the theoretical χ^2 distribution. A key parameter for the χ^2 distribution is the number of degrees of freedom. In this case, the number of degrees of freedom is $\nu - 1$, where ν is the number of categories to which x_i's belong. If the variance of the theoretical distribution is estimated by the sample variance [125] of the observed data, the number of degrees of freedom is then $\nu - 2$.

There are many other methods to test the null hypothesis H_0, for example, the Kolmogorov test [14]. We have studied χ^2 in a greater depth because our goal is to obtain good features for distinguishing image types rather than to test the literal truthfulness of the hypothesis. The χ^2 statistic is chosen because the deviation from an assumed distribution in every data bin C_i is taken into consideration and this statistic is easy to compute.

To measure the goodness of match between an observed distribution and a Laplacian distribution, we need to determine the parameters of the Laplacian distribution. Recall that the pdf of a Laplacian distribution is

$$p_X(x) = \frac{\lambda}{2} e^{-\lambda|x|} \qquad -\infty < x < \infty \,,\ \lambda > 0 \,.$$

Since $VAR[X] = 2/\lambda^2$, the parameter λ is estimated by moment matching, i.e.,

$$\hat{\lambda} = \sqrt{\frac{2}{\hat{\sigma}^2}} \,,$$

where $\hat{\sigma}^2$ is the sample variance of X.

One classification feature is defined as χ^2 normalized by the sample size N, denoted by $\bar{\chi}^2$, which is defined by

$$\bar{\chi}^2 = \chi^2/N = \sum_{i=1}^{\nu} (f_i - F_i)^2/F_i \,.$$

The feature is χ^2 normalized by N because we are interested in how close f_i and F_i are instead of whether the null hypothesis H_0 should be accepted. When the sample size is large, the observed relative frequency converges to its true distribution, which cannot really be a Laplacian distribution since x_i is bounded. Therefore, χ^2 increases approximately linearly with the sample size.

Likelihood of Being a Highly Discrete Distribution. A criterion, denoted by L, is defined to measure the likelihood of wavelet coefficients being composed by highly concentrated values. For example, Figure 4.10(c) shows that data only lie in the five far apart bins, indicating a high L. The efficiency of L estimating the likelihood of wavelet coefficients obeying a highly discrete distribution depends on how completely concentrated peak values can be detected, and how robust the identification of these values is to local fluctuations.

Since for highly concentrated data, their absolute values have the same property, histograms of absolute values are used to calculate L to save computation. An example histogram shown in Figure 4.11 is analyzed first for gaining intuition regarding to a proper definition of L. It seems reasonable to assume that V_0 and V_1 are two concentration values. First of all, they are both peak values (maximum values) in a data range $[0, Z_1]$ and $[Z_1, Z_2]$. In addition, most of the data in either of the two data ranges are distributed in a narrow neighborhood of V_0 or V_1; and the number of data points vanishes towards the end points. On the other hand, although V_2 is a local maximum in $[Z_1, Z_2]$, it is not regarded as a concentration value since it appears to result from a fluctuation around V_1; V_3 and V_4 are not considered as concentration values because neither of them is surrounded by a tight cluster containing most data in the range $[Z_2, Z_3]$. To summarize from the example, a concentration value is expected to possess the following properties:

1. A concentration value is the maximum in a certain data range, say $[Z_1, Z_2]$. Such a data range is defined as a 'zone.'

2. Most of the data in the range $[Z_1, Z_2]$ are distributed in a narrow neighborhood of the peak value.

3. The amount of data is vanishingly small at the ends of $[Z_1, Z_2]$.

In order to locate concentration values, an essential step is to divide data into zones based on the histogram. Intuitively, we imagine a zone to be a range, isolated from the remainder of the data axis, in which data

clump. The histogram in a zone should have a peak value and vanish towards the ends of the zone. A more rigorous definition is provided to capture these ideas.

Suppose the complete data range is $[0, Z]$. The histogram in $[0, Z]$, normalized by the total number of samples, is represented by a non-negative unit integral function $h(t), t \in [0, Z]$. The interval $[0, Z]$ is partitioned into connected zones $[t_0, t_1], [t_1, t_2], \ldots$, and $[t_{r-1}, t_r]$, where $t_0 = 0$ and $t_r = Z$. An interval $[t_i, t_{i+1}]$ is a zone if it satisfies the following conditions:

1. If $t_i \neq 0$ and $t_{i+1} \neq Z$,
   (a) There exists a $t^* \in (t_i, t_{i+1})$, s.t. $h(t^*)$ is the maximum value for $t \in [t_i, t_{i+1}]$.
   (b) $h(t_i) \leq h(t)$, for $t \in (t_i, t^*)$
   $h(t_{i+1}) < h(t)$, for $t \in (t^*, t_{i+1})$
   (c) $h(t_i)/h(t^*) < \delta$, and $h(t_{i+1})/h(t^*) < \delta$, where δ is a threshold, which is set to be 0.05 in our implementation.
   (d) There does not exist $[\tau, \tau'] \subset [t_i, t_{i+1}]$ and $[\tau, \tau'] \neq [t_i, t_{i+1}]$, so that $[\tau, \tau']$ satisfies the above three conditions.
2. If $t_i = 0$ and $t_{i+1} \neq Z$, the four conditions are almost the same as those of the previous case, except for two relaxed conditions listed below:
   (a) The maximum point t^* is allowed to appear at the end point $t_i = 0$.
   (b) It is not required that $h(t_i)/h(t^*) < \delta$ and $h(t_i) \leq h(t)$, for $t \in (t_i, t^*)$.
3. If $t_{i+1} = Z$, the only condition for $[t_i, t_{i+1}]$ to be a zone is:
   (a) There does not exist $[\tau, \tau'] \subset [t_i, t_{i+1}]$ and $[\tau, \tau'] \neq [t_i, t_{i+1}]$, so that $[\tau, \tau']$ satisfies the previous definition of a zone.

In the above list of conditions, the first case applies to a normal zone, while the second and the third cases provide relaxed conditions for zones at the ends of the data range $[0, Z]$. For a normal zone, the first condition requires that the zone contains an internal maximum point. This maximum point is the candidate concentrated value in the zone. The second condition ensures that each end point of the zone is a local minimum in the entire data range and a global minimum in the interval between

the end point and the maximum point in the zone. The third condition requires that the empirical probability density at each end point of the zone be significantly smaller than the maximum in the zone. This requirement guards against local fluctuations of the distribution. In addition, the combination of the second and the third conditions forces the histogram to vanish towards the ends of the zone. The fourth condition guarantees the partition of $[0, Z]$ being the finest possible. The second and the third cases specify the conditions for a zone located at one end of the data range. Some requirements in the first case are discarded so that a zone partition always exists for a data range with continuous probability density function. Every zone in a partition may have a different level of concentration. In the special case when the distribution is sufficiently smooth, the entire data range is identified to be one zone, and the concentration level of the zone is nearly zero.

The algorithm for finding the partition of $[0, Z]$ is described in the Appendix 4.A. To define L, suppose that the partition has been obtained. Let us start with defining a concentration level for every zone. Then, L is calculated as a weighted sum of these concentration levels.

For a zone $[t_i, t_{i+1}]$, $i = 0, 1, \ldots, r-1$, denote the concentration level by β_i. Let the maximum point in the zone be t^*. The width of the neighborhood around t^*, in which data are considered to be sufficiently close to t^*, is set to w. Subject to the limited data range, the two end points of the neighborhood around t^*, denoted by τ_1 and τ_2, are thus

$$\tau_1 = \begin{cases} t^* - w & t_i < t^* - w \\ t_i & \text{otherwise} . \end{cases}$$

$$\tau_2 = \begin{cases} t^* + w & t^* + w < t_{i+1} \\ t_{i+1} & \text{otherwise} . \end{cases}$$

The probability of being in $[\tau_1, \tau_2]$ is $p_i = \int_{\tau_1}^{\tau_2} h(t)dt$, and the probability of being in $[t_i, t_{i+1}]$ is $p_i' = \int_{t_i}^{t_{i+1}} h(t)dt$. The concentration level β_i is then defined as a thresholded version of the ratio of p_i to p_i', which is

$$\beta_i = \begin{cases} p_i/p_i' & p_i/p_i' > \gamma \\ 0 & \text{otherwise} , \end{cases} \qquad \text{where } \gamma \text{ is a threshold} .$$

The quantity β_i is set to zero when it is below a threshold in order to increase robustness to noise. Simulations do not show significant difference, however, if the thresholding is not applied. Once the β_i's are

obtained, L is evaluated as

$$L = \sum_{i=0}^{r-1} p_i \beta_i \ .$$

Note that $0 \leq L \leq 1$. To ensure the existence of a highly concentrated value, the parameter w should be decreased when the sample size of the histogram becomes smaller. The reason is that, for a small block of pixels, their intensities tend to be close to each other simply due to the spatial continuity of natural images.

4.12.3 THE SELECTION OF WAVELET TRANSFORMS

When we choose a wavelet transform for computing features, several criteria are considered. First, and most importantly, a good transform should yield distinct features for different image types. Second, the transform is required to have a good localization property. After an image is decomposed by the wavelet transform, coefficients at the same spatial location as a particular block should not be heavily influenced by surrounding blocks. Otherwise, it will be difficult to classify at boundaries between image types. Provided that the two criteria are met, a wavelet transform with less computation is preferred.

In our system, the one-level Haar wavelet transform is used. As is shown by tests on many document images, the Haar transform provides distinct features for various image types. Comparison with wavelet transforms with longer filters, such as the Daubechies 8 and Daubechies 4 transforms [37], shows that the difference between those transforms in terms of providing good features is negligible. On the other hand, the Haar transform has the best spatial localization property since its wavelet filter is the shortest. For the same reason, this transform costs the least amount of computation.

4.12.4 RESULTS

The two features defined in the previous section were used as intra-block features. The first one is the goodness of fit between the empirical distribution of wavelet coefficients in high frequency bands and the Laplacian distribution. The second one is the likelihood of the wavelet coefficients having highly concentrated values, denoted by L. Inter-block features were also used. They are the spatial derivatives of average intensity values and L's. The block size used was 8×8. The HMM has 5

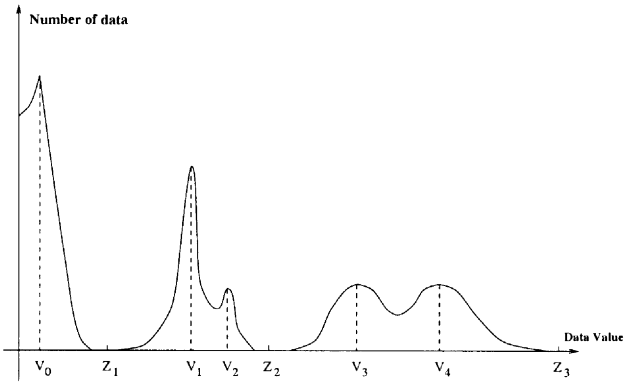


Figure 4.11. The concentration values of a histogram

states for each class. Experiments show that models with 2 to 5 states for each class yield similar results.

The result of HMM was compared with that of a classification tree generated by CART with both inter- and intra-block features. The image set is provided by Hewlett Packard, Inc. [107, 106]. They are RGB color images with approximate size 1600×1300 pixels. Each color component is 8 bpp. In the experiments, only the luminance component (i.e., gray-scale images) was used. For most images tested, both algorithms achieve very low classification error rates, about 2% on average. More differences between the two algorithms appear with the sample image shown in Fig. 4.12 because the photograph region in this image is very smooth at many places, so it resembles text. The classification results of both CART and the 2-D HMM algorithm are presented in Fig. 4.12. It is shown that the result using the HMM is much cleaner than that using CART, especially in the photograph regions. This is expected since the classification based on the HMM takes context into consideration. Consequently, some smooth blocks in the photograph regions, which locally resemble text blocks, are identified correctly as photograph. This advantage of the HMM algorithm is also demonstrated in Fig. 4.13 by another example test image.

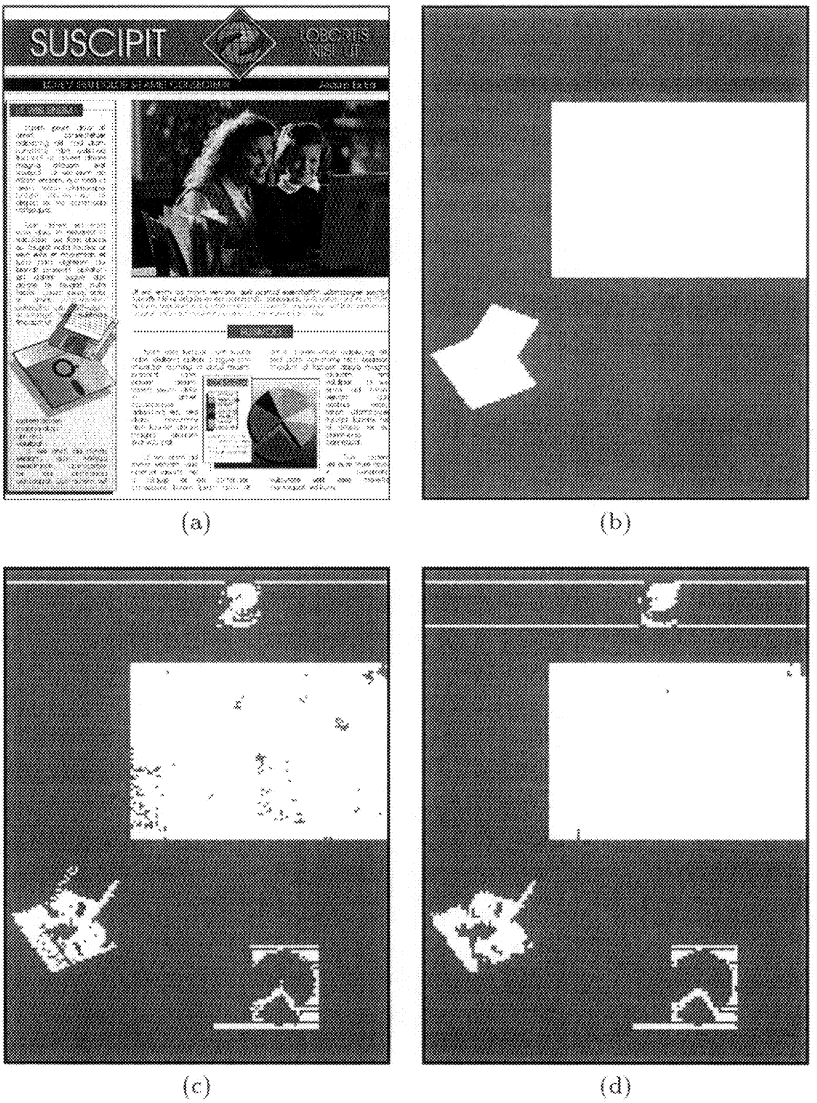


Figure 4.12. Test document image 1: (a) Original image, (b) Hand-labeled classified image, (c) CART classification result, (d) 2-D HMM classification result. White: photograph, Gray: text

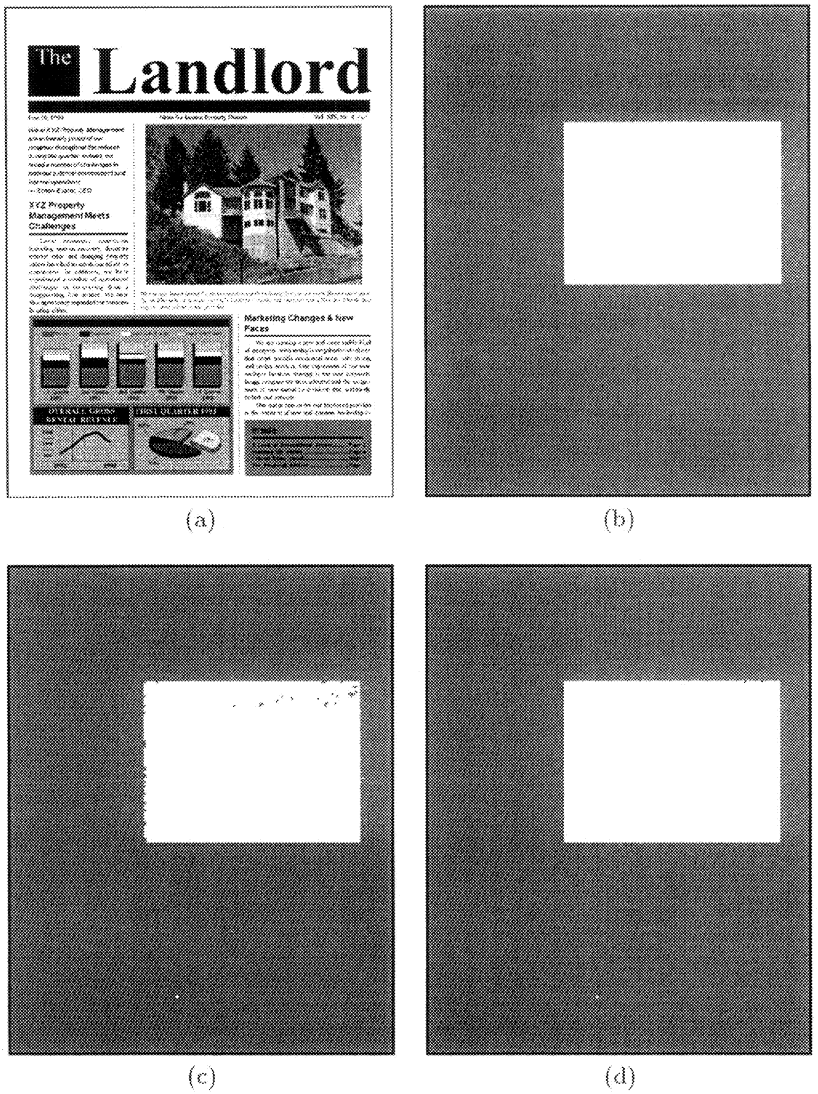


Figure 4.13. Test document image 2: (a) Original image, (b) Hand-labeled classified image, (c) CART classification result, (d) 2-D HMM classification result. White: photograph, Gray: text

APPENDIX 4.A: Histogram Partition

We describe the algorithm for finding the zone partition for a data range $[0, Z]$ given an empirical distribution. This completes the definition of the classification feature L, the likelihood of being a highly discrete distribution, in Section 4.12.2. According to the second constraint of the definition for a zone, the two end points of a zone have to be local minima except for the two zones $[0, t_1]$ and $[t_{r-1}, Z]$, that is, t_1, $\ldots$, t_{r-1} are local minimum points. So the problem is reduced to finding all of the local minima of $h(t)$. On the other hand, not every local minimum point is an end point of a zone due to the third constraint. The value at an end point of a zone is required to be sufficiently small compared with the maximum value in the zone. Consequently, we have to extract the local maxima in the process of seeking local minima and guarantee that the global maximum (one of the local maxima) in a zone is significantly greater than the values at the two end points. Thus, a partition is characterized by a sequence of interleaved local maximum and local minimum points, t_0^*, t_1, t_1^*, t_2, $\ldots$, t_{r-1}, t_{r-1}^*. Every local maximum point t_i^* is also a global maximum point in interval $[t_i, t_{i+1}]$.

In the following algorithm, we assume that $h(t)$ is a continuous function on $[0, Z]$, so that there exists a local maximum between two local minima and there exists a local minimum between two local maxima. The continuity constraint is always acceptable in practice because we can only obtain estimated samples of $h(t)$, and the function $h(t)$ can be estimated by a continuous interpolation of the samples.

1. Find the first local maximum point starting from 0 and set t_0^* to it.
2. Set True $\rightarrow$ pre_is_maximum, and True $\rightarrow$ seek_minimum.
3. Set $t_0^* \rightarrow T$, $0 \rightarrow j$.
4. If $T < Z$,
   (a) If seek_minimum = True
      i. Find $T < t < Z$, so that $h(t)$ is a local minimum.
      If such $h(t)$ does not exist, $Z \rightarrow t_{j+1}$, $Z \rightarrow T$, stop.
      ii. If pre_is_maximum = True
      { if $h(t)/h(t_j^*) < \delta$, then set $t \rightarrow t_{j+1}$, $j + 1 \rightarrow j$, False $\rightarrow$ pre_is_maximum. } ;
      Else (i.e., pre_is_maximum = False)
      { if $h(t) < h(t_j)$, $t \rightarrow t_j$}

iii. Set False $\rightarrow$ seek_minimum, $t \rightarrow T$.

(b) If seek_minimum = False

i. Find $T < t < Z$, so that $h(t)$ is a local maximum.
If such $h(t)$ does not exist, $Z \rightarrow t_{j+1}$, $Z \rightarrow T$, stop.

ii. If pre_is_maximum = True
{ if $h(t) > h(t_j^*)$, $t \rightarrow t_j^*$ } ;
Else (i.e., pre_is_maximum = False)
{ if $h(t_j)/h(t) < \delta$, then set $t \rightarrow t_j^*$, True $\rightarrow$ pre_is_maximum.
}

iii. Set True $\rightarrow$ seek_minimum, $t \rightarrow T$.

5. If $T < Z$, go back to 4;
else, stop.

The flag seek_minimum in the algorithm is alternated so that the sequence of all the interleaved local maxima and local minima, denoted by $\tilde{t^*}_0$, $\tilde{t}_1$, $\tilde{t^*}_1$, $\tilde{t}_2$, ..., $\tilde{t}_{r'-1}$, $\tilde{t^*}_{r'-1}$ can be found. Since this sequence may not satisfy all the constrains for a partition, we need to choose a subsequence that forms a partition. This is accomplished by effective control of the flag pre_is_maximum. The algorithm determines all the t_i and t_i^* in the order of minimum to maximum. The process, however, is not completely sequential in the sense that t_i, t_i^*, and t_{i+1}, which correspond to the end points, and maximum point of a zone, are adjusted simultaneously, instead of being determined one by one. Only when a zone $[t_i, t_{i+1}]$, with maximum point t_i^*, satisfying all the conditions is found, the left end point t_i and the maximum point t_i^* are fixed. The right end point t_{i+1} may still be changed when seeking for the next zone $[t_{i+1}, t_{i+2}]$. Nevertheless, the end point t_{i+1} is ensured to be replaced by a point with a lower probability density value if there is ever a replacement. This strategy guarantees the division between two zones to take place at a point with distribution density as low as possible.

To gain a better understanding, consider a complete cycle of obtaining a zone $[t_i, t_{i+1}]$ given that the zone $[t_{i-1}, t_i]$ has been found. At the beginning, the flag pre_is_maximum is set as false and the algorithm seeks a maximum point. According to the two flags seek_minimum and pre_is_maximum, the algorithm may be in any of four states. In Table 4.A.1, we list the points that might be initially set or changed at the current state, and the next possible state to enter. The algorithm transits between states according to Table 4.A.1. Except for determining the first zone $[0, t_1]$, which starts with both seek_minimum and

Current State (S, P)	Points to be adjusted	Possible Next State (S, P)
(F, F)	t_i^*	(T, F), (T, T)
(T, F)	t_i	(F, F)
(F, T)	t_i^*	(T, T)
(T, T)	t_{i+1}	(F, T), (F, F)

Table 4.A.1. Transitions in the 'zone' division algorithm and the points adjusted at every state. S: seek_minimum, P: pre_is_maximum, T: True, F: False

pre_is_maximum being true, the cycles for the other zones always start with both seek_minimum and pre_is_maximum being false. The special case for $[0, t_1]$ happens because the left end point is tied at 0.

Chapter 5

2-D MULTIRESOLUTION HMM

What is the most rigorous law of our being? Growth. No smallest atom of our moral, mental, or physical structure can stand still a year. It grows – it must grow; nothing can prevent it.

—Mark Twain

In Chapter 4, a two dimensional hidden Markov model is defined for image classification. By assuming that the underlying state process is a Markov mesh, context information is incorporated in classification. A joint decision on the classes of all the blocks in an image is needed to classify the image optimally based on the 2-D HMM. In real life, however, because of computational complexity, we have to divide an image into subimages and ignore the statistical dependence among the subimages. With the increase of model complexity, it is necessary to decrease the size of the subimages to preserve modest computational feasibility. Instead of using smaller subimages, a classifier based on the multiresolution model retains tractability by representing context information hierarchically.

With a 2-D multiresolution hidden Markov model (MHMM), an image is taken to be a collection of feature vectors at several resolutions. These feature vectors at a particular resolution are determined only by the image at that resolution. The feature vectors across all the resolutions are generated by a multiresolution Markov source. As with the 2-D HMM, the source exists in a state at any block at any resolution. Given the state of a block at each particular resolution, the feature vector is assumed to have a Gaussian distribution so that the unconditional distribution is a Gaussian mixture. The parameters of each Gaussian

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

distribution depend on both state and resolution. At any fixed resolution, as with the 2-D HMM, the probability of the source entering a particular state is governed by a second order Markov mesh [1]. Unlike the HMM, there are multiple Markov meshes at one resolution whose transition probabilities depend on the states of parent blocks.

5.1 BASIC ASSUMPTIONS OF 2-D MHMM

To classify an image, representations of the image at different resolutions are computed first. The original image corresponds to the highest resolution. Lower resolutions are generated by successively filtering out high frequency information. Wavelet transforms [37] naturally provide low resolution images in the low frequency band (the LL band). A sequence of images at several resolutions is shown in Fig. 5.1. As subsampling is applied for every reduced resolution, the image size decreases by a factor of two in each direction. As is shown by Fig. 5.1, the number of blocks in both rows and columns is successively diminished by half at each lower resolution. Obviously, a block at a lower resolution covers a spatially more global region of the image. As is indicated by Fig. 5.2, the block at the lower resolution is referred to as a parent block, and the four blocks at the same spatial location at the higher resolution are referred to as child blocks. We will always assume such a "quadtree" split in the sequel since the training and testing algorithms can be easily extended to other hierarchical structures.

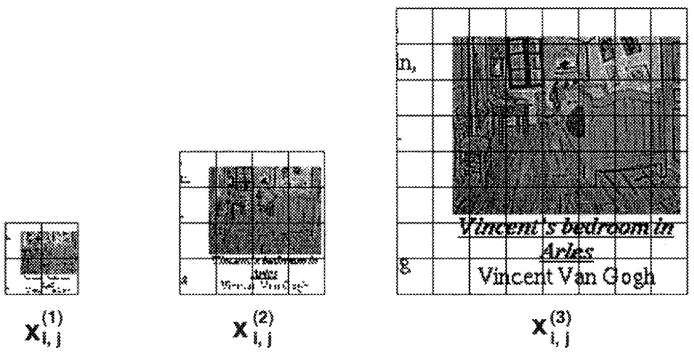


Figure 5.1. Multiple resolutions of an image

Denote the collection of resolutions by $\mathcal{R} = \{1, ..., R\}$, with $r = R$ being the finest resolution. Let the collection of block indices at resolution r be $\mathbb{N}^{(r)} = \{(i,j) : 0 \leq i < w/2^{R-r}, 0 \leq j < z/2^{R-r}\}$. Images

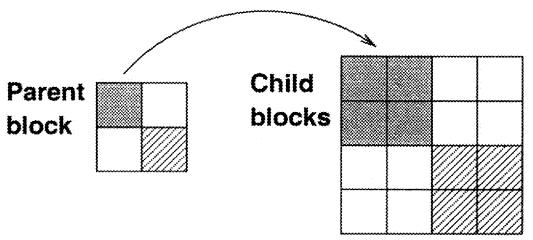


Figure 5.2. The image hierarchy across resolutions

are described by feature vectors at all the resolutions, denoted by $u_{i,j}^{(r)}$, $r \in \mathcal{R}$. Every feature vector is labeled with a class $c_{i,j}^{(r)}$. The underlying state of a feature vector is $s_{i,j}^{(r)}$. At each resolution r, the set of states is $\{1^{(r)}, 2^{(r)}, ..., M_r^{(r)}\}$. Note that different resolutions do not share states.

As with the single resolution model, each state at every resolution is uniquely mapped to one class. On the other hand, a block with a known class may exist in one of several states. Since a block at a lower resolution contains several blocks at a higher resolution, it may not be of a pure class. Therefore, except for the highest resolution, there is an extra "mixed" class in addition to the original classes. Denote the set of original classes by $\mathcal{G} = \{1, 2, ..., G\}$ and the "mixed" class by $G + 1$. Because of the unique mapping from states to classes, the state of a parent block may restrict the possible states for its child blocks. If the state of a parent block is mapped to a determined (non-mixed) class, the child blocks can exist only in states that map to the same class.

To structure statistical dependence among resolutions, a Markov chain with resolution playing a time-like role is assumed in the 2-D MHMM. Given the states and the features at the parent resolution, the states and the features at the current resolution are conditionally independent of the other previous resolutions, so that

$$\begin{aligned}
&P\{s_{i,j}^{(r)}, u_{i,j}^{(r)} : r \in \mathcal{R}, (i,j) \in \mathbb{N}^{(r)}\} \\
&= P\{s_{i,j}^{(1)}, u_{i,j}^{(1)} : (i,j) \in \mathbb{N}^{(1)}\} \times \\
&P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathbb{N}^{(2)} | \, s_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} \times \cdots \times \\
&P\{s_{i,j}^{(R)}, u_{i,j}^{(R)} : (i,j) \in \mathbb{N}^{(R)} | \, s_{k,l}^{(R-1)} : (k,l) \in \mathbb{N}^{(R-1)}\}. \qquad (5.1)
\end{aligned}$$

At the coarsest resolution, $r = 1$, feature vectors are assumed to be generated by a single resolution 2-D HMM. At a higher resolution, the

conditional distribution of a feature vector given its state is also assumed to be Gaussian. The parameters of the Gaussian distribution depend upon the state at the particular resolution.

Given the states at resolution $r-1$, statistical dependence among blocks at the finer resolution r is constrained to sibling blocks (child blocks descended from the same parent block). Equivalently, child blocks descended from different parent blocks are conditionally independent. In addition, given the state of a parent block, the states of its child blocks are independent of the states of their "uncle" blocks (non-parent blocks at the parent resolution). State transitions among sibling blocks are governed by the same Markovian property assumed for a single resolution 2-D HMM. The state transition probabilities, however, depend on the state of their parent block. To formulate these assumptions, denote the child blocks at resolution r of block (k,l) at resolution $r-1$ by

$$\mathbb{D}(k,l) = \{(2k,2l),(2k+1,2l),(2k,2l+1),(2k+1,2l+1)\} .$$

According to the assumptions,

$$\begin{aligned} & P\{s_{i,j}^{(r)} : (i,j) \in \mathbb{N}^{(r)} \mid s_{k,l}^{(r-1)} : (k,l) \in \mathbb{N}^{(r-1)}\} \\ = & \prod_{(k,l)\in\mathbb{N}^{(r-1)}} P\{s_{i,j}^{(r)} : (i,j) \in \mathbb{D}(k,l) \mid s_{k,l}^{(r-1)}\} , \end{aligned}$$

where $P\{s_{i,j}^{(r)} : (i,j) \in \mathbb{D}(k,l) | \, s_{k,l}^{(r-1)}\}$ can be evaluated by transition probabilities conditioned on $s_{k,l}^{(r-1)}$, denoted by $a_{m,n,l}(s_{k,l}^{(r-1)})$. We thus have a different set of transition probabilities $a_{m,n,l}$ for every possible state in the parent resolution. The influence of previous resolutions is exerted hierarchically through the probability of the states, which can be visualized in Fig. 5.3. The joint probability of states and feature vectors at all the resolutions in (5.1) is then derived as

$$\begin{aligned} & P\{s_{i,j}^{(r)}, u_{i,j}^{(r)} : r \in \mathcal{R}, (i,j) \in \mathbb{N}^{(r)}\} \\ = \; & P\{s_{i,j}^{(1)}, u_{i,j}^{(1)} : (i,j) \in \mathbb{N}^{(1)}\} \times \\ & \prod_{r=2}^{R} \prod_{(k,l)\in\mathbb{N}^{(r-1)}} \Big[P\{s_{i,j}^{(r)} : (i,j) \in \mathbb{D}(k,l) \mid s_{k,l}^{(r-1)}\} \times \\ & \prod_{(i,j)\in\mathbb{D}(k,l)} P\{u_{i,j}^{(r)} \mid s_{i,j}^{(r)}\} \Big] . \end{aligned}$$

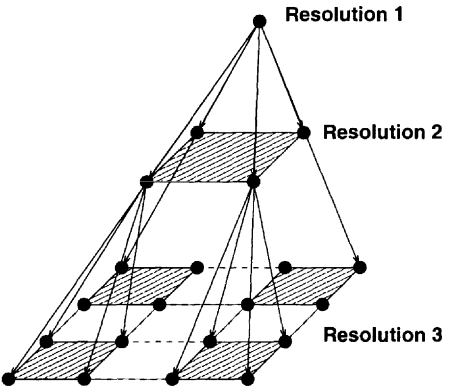


Figure 5.3. The hierarchical statistical dependence across resolutions

To summarize, a 2-D MHMM reflects both the inter-scale and intra-scale statistical dependence. The inter-scale dependence is modeled by the Markov chain over resolutions. The intra-scale dependence is modeled by the HMM. At the coarsest resolution, feature vectors are assumed to be generated by a 2-D HMM. At all the higher resolutions, feature vectors of sibling blocks are also assumed to be generated by 2-D HMMs. The HMMs vary according to the states of parent blocks. Therefore, if the next coarser resolution has M states, then there are, correspondingly, M HMMs at the current resolution. The motivation for having both the inter- and intra-scale dependence is discussed in Section 5.2. Experiments in Section 5.6 show the influence of both types of dependence.

5.2 RELATED WORK

A variety of multiresolution models have been proposed for the purpose of incorporating relatively global information into image classification. One early work in this direction is the multiscale autoregressive model proposed by Basseville *et al.* [7]. Suppose images are represented by R resolutions, with $r = 1$ being the coarsest resolution. Pixel intensities at resolution r are denoted by $X^{(r)} = \{X^{(r)}(i,j) : (i,j) \in \mathbb{N}^{(r)}\}$. Define a coarse-scale shift operator s to reference the *parent* node, and s^k to reference the *ancestor* node k levels higher. Specifically, $s^k(i,j) = (\lfloor i/2^k \rfloor, \lfloor j/2^k \rfloor)$. A homogeneous multiscale autoregressive model has

the property that

$$\begin{aligned} x^{(r)}(i,j) &= a_{r-1}x^{(r-1)}(s(i,j)) + a_{r-2}x^{(r-2)}(s^2(i,j)) \\ &\quad + \cdots + a_1 x^{(1)}(s^{r-1}(i,j)) + w(i,j), \quad a_k \in \Re \end{aligned}$$

where $w(i,j)$ is an independent white driving noise.

As with autoregressive models in other contexts, this model entails a rather constrained dependence, here across resolutions. Recent work has generalized the cross resolution dependence by introducing Gaussian mixture models [17] or hidden Markov models [36]. Bouman and Shapiro proposed the multiscale random field (MSRF) model for images. Suppose an image is described by a random field X. The pixel labels (or classes) at resolution r are $C^{(r)}$, $r = 1, ..., R$. The first assumption of the MSRF is the Markovian property across resolutions, i.e.,

$$P(C^{(r)} = c^{(r)} \mid C^{(l)} = c^{(l)}, l < r) = P(C^{(r)} = c^{(r)} \mid C^{(r-1)} = c^{(r-1)}) \, .$$

The second assumption is the exclusive dependence of X on $C^{(R)}$, that is,

$$P(X \in dx \mid C^{(r)} = c^{(r)}, r = 1, ..., R) = P(X \in dx \mid C^{(R)} = c^{(R)}) \, .$$

For segmentation, the models are restricted to two properties regarding $C^{(r)}$, $r = 1, ..., R$. First, the individual classes in $C^{(r)}$ are conditionally independent given the classes in $C^{(r-1)}$. Second, each class in $C^{(r)}$ depends only on classes in a neighborhood at the coarser resolution $r-1$.

There are three key differences between our 2-D MHMMs and the MSRF models for segmentation [17]. First, in the MSRF, features are observed solely at the finest resolution. The coarser resolutions figure only in prior probabilities of classes. For many applications of image classification [127, 89], it has been found that combining features extracted from several resolutions improves classification. In Section 5.6, experiments also demonstrate the gain in performance that owes to multiresolution features. Second, states and classes are not distinguished by the MSRF in that every class is considered as one state. At the finest resolution, the conditional distribution of feature vectors given a state is a Gaussian mixture. It is shown [139] that such an HMM is equivalent to a special case of the HMM we assumed, in which every class contains several states each corresponding to a component of the Gaussian mixture. On the other hand, in general, an HMM with multiple states in one class and Gaussian distributions conditioned on states cannot be converted to an HMM with a single state in every class and a Gaussian

mixture distribution given each state. Third, the MSRF assumes statistical dependence only across resolutions. In the 2-D MHMM, however, since sibling blocks are dependent given the state of their parent block, inter-scale and intra-scale dependence can be balanced flexibly. With only the inter-scale dependence, a multiresolution model implies that a parent node completely summarizes context information for its child nodes. However, this assumption need not be true in practice, even approximately. In fact, for many applications, information useful for distinguishing classes is embedded in relatively high frequency bands. As a result, when the resolution is sufficiently low, a parent node cannot provide any helpful context information.

A 2-D MHMM provides a mechanism to trade inter-scale and intra-scale dependence according to applications. For example, suppose the number of blocks at the finest resolution that a system intends to classify jointly is 8×8. If the HMM assumed for feature vectors at the coarsest resolution examines 2×2 blocks jointly, we need a 3 resolution model with quadtree split. If the HMM at the coarsest resolution examines 4×4 blocks jointly, we then need a 2 resolution model with quadtree split. Another set-up of a 2 resolution model might be to replace the quadtree split by a 4×4 split and assume an HMM on 2×2 blocks at the coarse resolution. The three possibilities of the MHMM are shown in Figure 5.4. All the parameters in the model structure set-up can be chosen conveniently as inputs to algorithms for training and testing.

Another multiresolution model based on HMMs is the model proposed for wavelet coefficients by Crouse *et al.* [36], where wavelet coefficients across resolutions are assumed to be generated by one-dimensional hidden Markov models with resolution being the time-like role in the Markov chain. If we view wavelet coefficients as special cases of features, the model in [36] considers features observed at multiple resolutions. However, intra-scale dependence is not pursued in depth in [36]. This wavelet-domain model is applied to image segmentation [28] and is extended to general features in [98].

The approach of applying models to image segmentation in [28] is different from that of Bouman and Shapiro [17] and ours. States in wavelet-domain HMMs are not related to classes. In particular, there are two states at every resolution, one representing a wavelet coefficient being large and the other small. To segment images, a separate HMM is trained for each class. A local region in an image is regarded as an instance of a random process described by one of the HMMs. To decide the class of the local region, likelihood is computed using the HMM of

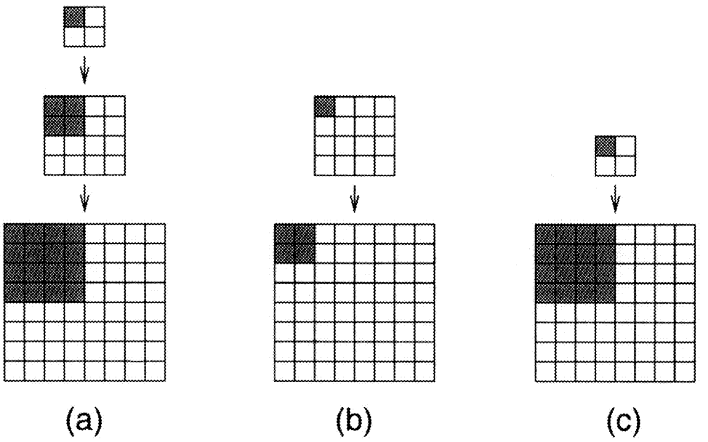


Figure 5.4. Three possible structures of MHMMs on an 8×8 grid of blocks: (a) a 3-level MHMM with quadtree split and the coarsest resolution modeled by an HMM on a 2×2 grid, (b) a 2-level MHMM with quadtree split and the coarsest resolution modeled by an HMM on a 4×4 grid, (c) a 2-level MHMM with 4×4 split and the coarsest resolution modeled by an HMM on a 2×2 grid.

each class, and the class yielding the maximum likelihood is selected. The whole image is then segmented by combining decisions for all the local regions. It is not straightforward for such an approach to account for the spatial dependence among classes in an image. Furthermore, the wavelet-domain HMMs alone do not provide a natural mechanism to incorporate segmentation results at multiple resolutions. A remedy, specifically context-based inter-scale fusion, is developed in [28] to address this issue. In Bouman and Shapiro [17] and our algorithm, however, an entire image is regarded as an instance of a 2-D random process characterized by one model, which reflects the transition properties among classes/states at all the resolutions as well as the dependence of feature vectors on classes/states. The set of classes or states with the maximum a posteriori probability is sought according to the model. Segmenting an image by combining features at multiple resolutions is inherent in our algorithm based on 2-D MHMMs. As the number of states and the way of extracting features are allowed to vary with resolution, it is flexible enough to incorporate multiscale information for classification using 2-D MHMMs.

In the field of computer vision, there has been much work on learning vision by image modeling [74, 81, 56]. Particularly, in [56], multiresolution modeling is applied to estimate motion from image frames.

Bayesian network techniques [15, 105, 68] have played an important role in learning models in computer vision. Theories on Bayesian network also provide guidance on how to construct models with tractable learning complexity. Exact inference on general Bayesian networks is NP-hard, as discussed by Cooper [32]. Computationally efficient algorithms for training a general Bayesian network are not always available. As we have constructed 2-D MHMMs by extending 1-D HMMs used in speech recognition [139], efficient algorithms for training and applying 2-D MHMMs are derived from the EM algorithm [38] and related techniques developed for speech recognition.

5.3 THE ALGORITHM

As with the single resolution 2-D HMM, the Viterbi training algorithm is applied to estimate the parameters of the 2-D MHMM. At every iteration, the combination of states at all the resolutions with the maximum a posteriori probability is searched by the Viterbi algorithm. These states are then assumed to be real states to update the estimation of parameters. Because of the multiple resolutions, a certain part of the training algorithm used for the single resolution HMM is replaced by a recursive procedure.

In this section, we extend the Viterbi training algorithm for 2-D HMMs to multiresolution models. Denote $\mathbf{s}^{(r)} = \{s_{i,j}^{(r)} : (i,j) \in \mathbb{N}^{(r)}\}$, $\mathbf{u}^{(r)} = \{u_{i,j}^{(r)} : (i,j) \in \mathbb{N}^{(r)}\}$, and $\mathbf{c}^{(r)} = \{c_{i,j}^{(r)} : (i,j) \in \mathbb{N}^{(r)}\}$. If the superscript (r) is omitted, for example, $\mathbf{s}$, it denotes the collection of $\mathbf{s}^{(r)}$ over all $r \in \mathcal{R}$.

The Viterbi training algorithm searches for $\mathbf{s}_{opt} = \max_{\mathbf{s}:C(\mathbf{s})=\mathbf{c}}^{-1} P(\mathbf{s} \mid \mathbf{u})$. As mentioned earlier, the set of original classes is $\mathcal{G} = \{1, 2, ..., G\}$. The "mixed" class is denoted by $G+1$. At the finest resolution R, $c_{i,j}^{(R)} \in \mathcal{G}$ is given by the training data. At a coarser resolution $r < R$, $c_{i,j}^{(r)}$ is determined by the recursive formula:

$$c_{i,j}^{(r)} = \begin{cases} g & c_{k,l}^{(r+1)} = g \text{ for all } (k,l) \in \mathbb{D}(i,j) \\ G+1 & \text{otherwise .} \end{cases}$$

Equivalent to the above recursion, $c_{i,j}^{(r)} = g$, $g \in \mathcal{G}$ if all the descending blocks of (i,j) at the finest resolution R are of class g. Otherwise, if different classes occur, $c_{i,j}^{(r)} = G+1$. By assigning $c_{i,j}^{(r)}$ in such a way, consistency on the mapping from states to classes at multiple resolutions

is enforced in that if $c_{i,j}^{(r)} = g$, $g \in \mathcal{G}$, the probability that $C(s_{k,l}^{(r+1)}) \neq g$, for any $(k,l) \in \mathbb{D}(i,j)$, is assigned 0.

To clarify matters, we present a case with two resolutions. By induction, the algorithm extends to models with more than two. Equivalent to the MAP rule, the optimal set of states maximizes the joint log likelihood of all the feature vectors and states under the constraint $C(\mathbf{s}) = \mathbf{c}$:

$$\begin{aligned}
& \log P\{s_{k,l}^{(r)}, u_{k,l}^{(r)} : r \in \{1,2\}, (k,l) \in \mathbb{N}^{(r)}\} \\
= \; & \log P\{s_{k,l}^{(1)}, u_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} + \\
& \log P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathbb{N}^{(2)} | \, s_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} \\
= \; & \log P\{s_{k,l}^{(1)}, u_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} + \\
& \sum_{(k,l) \in \mathbb{N}^{(1)}} \log P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l) | \, s_{k,l}^{(1)}\} \, .
\end{aligned}$$

The algorithm works backwards to maximize the above log likelihood. First, for each $s_{k,l}^{(1)}$ and each $(k,l) \in \mathbb{N}^{(1)}$, $\{\bar{s}_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l)\}$ is searched to maximize $\log P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l) | \, s_{k,l}^{(1)}\}$. Since given $s_{k,l}^{(1)}$, the child blocks at Resolution 2 are governed by a single resolution 2-D HMM with transition probabilities $a_{m,n,l}(s_{k,l}^{(1)})$, the variable-state Viterbi algorithm can be applied directly. In order to make clear that $\bar{s}_{i,j}^{(2)}$ depends on $s_{k,l}^{(1)}$, we often write $\bar{s}_{i,j}^{(2)}(s_{k,l}^{(1)})$. The next step is to maximize

$$\begin{aligned}
& \log P\{s_{k,l}^{(1)}, u_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} + \\
& \sum_{(k,l) \in \mathbb{N}^{(1)}} \log P\{\bar{s}_{i,j}^{(2)}(s_{k,l}^{(1)}), u_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l) | \, s_{k,l}^{(1)}\} \\
= \; & \sum_{\tau} \Bigg[\log P(T_{\tau}^{(1)} | T_{\tau-1}^{(1)}) + \sum_{(k,l):\Delta(k,l)=\tau} \Big(\log P(u_{k,l}^{(1)} | s_{k,l}^{(1)}) + \\
& \log P\{\bar{s}_{i,j}^{(2)}(s_{k,l}^{(1)}), u_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l) | \, s_{k,l}^{(1)}\} \Big) \Bigg] \, ; \qquad (5.2)
\end{aligned}$$

(5.2) follows from (4.20). As in (4.20), $T_{\tau}^{(1)}$ denotes the sequence of states for blocks on diagonal τ at Resolution 1. We can use the variable-state Viterbi algorithm again to search for the optimal $s_{i,j}^{(1)}$ since $T_{\tau}^{(1)}$ still serves as an "isolating" element in the expansion. The only difference

from the maximization of (4.20) is the extra term

$$\log P\{\bar{s}_{i,j}^{(2)}(s_{k,l}^{(1)}), u_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l) | \, s_{k,l}^{(1)}\} \, ,$$

which is already computed and stored as part of the first step.

Provided with the $\mathbf{s}_{opt}$, parameters are estimated by equations similar to (4.12), (4.13), and (4.14). For notational simplicity, the superscripts (p) and $(p+1)$ denoting iterations are replaced by (r) to denote the resolution, and the subscript '*opt*' of $\mathbf{s}_{opt}$ is omitted. States $s_{i,j}^{(r)}$, $(i,j) \in \mathbb{N}^{(r)}$, $r \in \mathcal{R}$, in equations (5.3), (5.4), and (5.5) are the optimal states searched by the MAP rule using model parameters estimated up to iteration p; parameters $\mu_m^{(r)}$, $\Sigma_m^{(r)}$, and $a_{m,n,l}^{(r)}(k)$ are results updated at iteration $p+1$. At each resolution r, $r \in \mathcal{R}$, the parameters are updated as follows:

$$\mu_m^{(r)} = \frac{\sum_{(i,j):(i,j)\in\mathbb{N}^{(r)}} I(m = s_{i,j}^{(r)}) u_{i,j}}{\sum_{(i,j):(i,j)\in\mathbb{N}^{(r)}} I(m = s_{i,j}^{(r)})} \tag{5.3}$$

$$\Sigma_m^{(r)} = \frac{\sum_{(i,j):(i,j)\in\mathbb{N}^{(r)}} I(m = s_{i,j}^{(r)})(u_{i,j} - \mu_m^{(r)})(u_{i,j} - \mu_m^{(r)})^t}{\sum_{(i,j):(i,j)\in\mathbb{N}^{(r)}} I(m = s_{i,j}^{(r)})} \tag{5.4}$$

$$a_{m,n,l}^{(r)}(k) = \frac{\gamma_1}{\gamma_2} \, , \tag{5.5}$$

where

$$\gamma_1 = \sum_{(i,j):(i,j)\in\mathbb{N}^{(r)}} I(m = s_{i-1,j}^{(r)}, n = s_{i,j-1}^{(r)}, l = s_{i,j}^{(r)}) I(k = s_{i',j'}^{(r-1)})$$

$$\gamma_2 = \sum_{l'=1}^{M} \sum_{(i,j):(i,j)\in\mathbb{N}^{(r)}} I(m = s_{i-1,j}^{(r)}, n = s_{i,j-1}^{(r)}, l' = s_{i,j}^{(r)}) I(k = s_{i',j'}^{(r-1)}) \, ,$$

and (i',j') is the parent block of (i,j) at resolution $r-1$. For quadtree split, $i' = \lfloor i/2 \rfloor$, $j' = \lfloor j/2 \rfloor$.

In the model testing process, that is, applying a 2-D MHMM to classify an image, the MAP states $\mathbf{s}_{opt}$ is searched first. Because the training algorithm guarantees the consistency on class mapping across resolutions, to derive classes from states, we only need to map the states at the finest resolution, $s_{i,j}^{(R)}$, $(i,j) \in \mathbb{N}^{(R)}$, into corresponding classes. The algorithm used to search $\mathbf{s}_{opt}$ in training is applied directly to testing. The only difference is that the constraint $C(\mathbf{s}_{opt}) = \mathbf{c}$ is removed since $\mathbf{c}$ is to be determined.

5.4 FAST ALGORITHMS

As states across resolutions are statistically dependent, to determine the optimal states according to the MAP rule, joint consideration of all resolutions is necessary. However, the hierarchical structure of the multiresolution model is naturally suited to progressive classification if we relax the MAP rule. Suboptimal fast algorithms are developed by discarding the joint consideration and searching for states in a layered fashion. States in the lowest resolution are determined only by feature vectors in this resolution. A classifier searches for the state of a child block in the higher resolution only if the class of its parent block is "mixed."

As one block at a lower resolution covers a larger region in the original image, making decisions at the lower resolution reduces computation. On the other hand, the existence of the "mixed" class warns the classifier of ambiguous areas that need examination at higher resolutions. For this reason, the degradation of classification due to the low resolution is avoided to a certain extent. Two fast algorithms are proposed.

5.4.1 FAST ALGORITHM 1

Use the two resolution case in the previous section as an example. To maximize $\log P\{s_{k,l}^{(r)}, u_{k,l}^{(r)} : r \in \{1,2\}, (k,l) \in \mathbb{N}^{(r)}\}$, Fast Algorithm 1 first searches for $\{\bar{s}_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\}$ that maximizes $\log P\{s_{k,l}^{(1)}, u_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\}$. For any $\bar{s}_{k,l}^{(1)}$, if it is mapped into the "mixed" class, the second step searches for $\{\bar{s}_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l)\}$ that maximizes $\log P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l) \mid \bar{s}_{k,l}^{(1)}\}$. Although the algorithm is "greedy" in the sense that it searches for the optimal states at each resolution, it does not give the overall optimal solution generally since the resolutions are statistically dependent.

5.4.2 FAST ALGORITHM 2

Fast Algorithm 2 trains a sequence of single resolution HMMs, each estimated using features and classes in a particular resolution. Except for the finest resolution, there is a "mixed" class. To classify an image, the first step is the same as that of Fast Algorithm 1: search for $\{\bar{s}_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\}$ that maximizes $\log P\{s_{k,l}^{(1)}, u_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\}$. In the second step, context information obtained from the first resolution is used, but differently from Fast Algorithm 1. Suppose $\bar{s}_{k,l}^{(1)}$ is mapped

into class "mixed," to decide $s_{i,j}^{(2)}$, $(i,j) \in \mathbb{D}(k,l)$, a neighborhood of (i,j), $\mathbb{B}(i,j)$ is formed, which contains $\mathbb{D}(k,l)$ as a subset. The algorithm then searches for the combination of states in $\mathbb{B}(i,j)$ that maximizes the a posteriori probability given features in this neighborhood according to the model at Resolution 2. Since the classes of some blocks in the neighborhood may have been determined by the states of their parent blocks, the possible states of those blocks are constrained to be mapped into the classes already known. The limited choices of these states, in turn, affect the selection of states for blocks whose classes are to be decided.

There are many choices for the neighborhood. In our experiments, particularly, the neighborhood is a 4×4 grid of blocks. For simplicity, the neighborhood of a block is not necessarily centered around the block. Blocks in the entire image are pre-divided into 4×4 groups. The neighborhood of each block is the group to which the block belongs.

5.5 COMPARISON OF COMPLEXITY WITH 2-D HMM

To show that the multiresolution HMM saves computation in comparison with the single resolution HMM, we analyze quantitatively the order of computational complexity for both cases. Assume that the Viterbi algorithm without path constraints is used to search for the MAP states so that we have a common ground for comparison.

For the single resolution HMM, recall that the Viterbi algorithm is used to maximize the joint log likelihood of all the states and features in an image according to (4.20):

$$\begin{aligned} & \log P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\} \\ = \; & \log P(T_0) + \log P(u_{0,0}|T_0) + \cdots + \\ & \sum_{\tau=1}^{w+z-2} \left(\log P(T_\tau | T_{\tau-1}) + \sum_{(i,j):\Delta(i,j)=\tau} P(u_{i,j}|s_{i,j}) \right), \end{aligned}$$

where T_τ is the sequence of states for blocks on diagonal τ, and w, or z is the number of rows, or columns in the image. For simplicity, assume that $w = z$. Every node in the Viterbi transition diagram (Fig. 4.4) corresponds to a state sequence T_τ, and every transition step corresponds to one diagonal τ. Therefore, there are in total $2w - 1$ transition steps

in the Viterbi algorithm. The number of blocks on diagonal τ is,

$$n(\tau) = \begin{cases} \tau + 1 & 0 \leq \tau \leq w-1 \\ 2w - \tau - 1 & w \leq \tau \leq 2w-2 \, . \end{cases}$$

The number of nodes at Step τ is $M^{n(\tau)}$, where M is the number of states.

For each node at Step τ, a node in the preceding step is chosen so that the path passing through the node yields the maximum likelihood up to Step τ. Suppose the amount of computation for calculating accumulated cost from one node in Step $\tau - 1$ to one node in Step τ is $\gamma(\tau)$. Since $\gamma(\tau)$ increases linearly with the number of blocks on diagonal τ, we write $\gamma(\tau) = c_1 n(\tau) + c_2$. The computation at step τ is thus $M^{n(\tau)} M^{n(\tau-1)} \gamma(\tau)$. The total computation for the Viterbi algorithm is

$$\begin{aligned} & \sum_{\tau=1}^{2w-2} M^{n(\tau)} M^{n(\tau-1)} \gamma(\tau) \\ = \; & ((2w-1)c_1 + 2c_2) \frac{M^{2w+1}}{M^2-1} - 2c_1 \frac{M^{2w+1}}{(M^2-1)^2} - \\ & (2c_2 - c_1) \frac{M}{M^2-1} + 2c_1 \frac{M}{(M^2-1)^2} - c_2 M \, . \end{aligned}$$

If M is sufficiently large so that $M^2 - 1 \approx M^2$ and $\frac{1}{M} \approx 0$, we simplify the above to

$$\begin{aligned} & \sum_{\tau=0}^{2w-2} M^{n(\tau)} M^{n(\tau-1)} \gamma(\tau) \\ \approx \; & ((2w-1)c_1 + 2c_2) M^{2w-1} - 2c_1 M^{2w-3} - c_2 M \, . \end{aligned}$$

The computation is thus of order $O(wM^{2w-1})$.

For the multiresolution model, considering the two resolution case, in the first step the Viterbi algorithm is applied to subimages $\mathbb{D}(k,l)$ to search for $\{s^{(2)}_{i,j} : (i,j) \in \mathbb{D}(k,l)\}$ that maximize $\log P\{s^{(2)}_{i,j}, u^{(2)}_{i,j} : (i,j) \in \mathbb{D}(k,l) | \, s^{(1)}_{k,l}\}$. For a fixed $(k,l) \in \mathbb{N}^{(1)}$ and a fixed state $s^{(1)}_{k,l}$, since $\mathbb{D}(k,l)$ is of size 2×2, the amount of computation needed for $\{\bar{s}^{(2)}_{i,j} : (i,j) \in \mathbb{D}(k,l)\}$ is of order $O(M_2^3)$, where M_2 is the number of states at Resolution 2. The total computation for the first step is then of order $M_2^3 (w/2)^2 M_1$, where M_1 is the number of states at Resolution 1. Since in the second step the Viterbi algorithm is applied to an image

of size $(w/2) \times (w/2)$, the computation for the second step is of order $(w/2)M_1^{w-1}$. If w is sufficiently large and M_1 and M_2 are about the same as M, the total computation for the multiresolution model is of order $O(wM^{w-1})$. Therefore, the multiresolution model reduces the amount of computation by order M^w.

Since computational order increases exponentially with w, the cardinality of the side of an image, we usually divide the image into subimages with side size w_0 and classify the subimages independently. The computational order for the single resolution HMM is then reduced to $O((\frac{w}{w_0})^2 w_0 M^{2w_0-1})$, which is $O(w^2 M^{2w_0-1})$ if w_0 is fixed. For the multiresolution HMM, the computational order of the second step becomes $(\frac{w}{w_0})^2 \frac{w_0}{2} M^{w_0-1}$, which does not dominate the computation in the first step if $w_0 - 1 \leq 4$. Hence the total computational order is $O(w^2 M^{\max\{w_0-1,4\}})$.

In practice, the path-constrained Viterbi algorithm, which preselects N nodes at each step for candidate paths, is applied to further reduce complexity. Since complexity varies tremendously when parameters including w_0 and N change, computational time will be compared based on experiments in the next section.

The comparison of complexity discussed above is for the testing process. In training, because more parameters need be estimated for the multiresolution model, a larger set of training data is required. As a result, the relative complexity of the multiresolution model in training is higher than in testing. In fact, if the number of states for each class is fixed across resolutions, with every increased resolution, the number of transition probabilities at the previous coarsest resolution increases by a fixed factor. Furthermore, Gaussian distributions and transition probabilities at the new coarsest resolution add parameters to be estimated. Therefore, the total number of parameters increases linearly with resolutions at a very high rate. Practically, however, such as in our applications, the number of states at a resolution usually declines when the resolution becomes coarser since images tend to be more homogeneous at coarser resolutions. In addition, the intra-scale dependence assumed in the 2-D MHMM allows adequate amount of context information to be used without driving the number of resolutions very high.

5.6 EXPERIMENTS

The algorithm was applied to the segmentation of man-made and natural regions of aerial images with the data set described in Chapter 4. Feature vectors were extracted at three resolutions. Images at

the two low resolutions were obtained by the Daubechies 4 [37] wavelet transform. This transform was chosen because it provides better low resolution images in comparison to the Haar wavelet and it also guarantees good spatial localization by sufficiently short wavelet filters. The images at Resolution 1 and 2 are, respectively, the LL bands of the 2-level and 1-level wavelet transforms. At each resolution, the image was divided into 4×4 blocks, and a feature vector was formed for every block. The features are defined the same as in Section 4.11.1.

Class	res 1	res 2	res 3
natural	5	5	5
man-made	5	5	9
mixed	4	2	0

(a)

Class	res 1	res 2	res 3
natural	1	1	5
man-made	1	1	9
mixed	1	1	0

(b)

Class	res 1	res 2	res 3
natural	2	2	5
man-made	2	2	9
mixed	2	2	0

(c)

Table 5.1. The number of states for each class at each resolution

The MHMM algorithm and its two fast versions were tested by six-fold cross-validation [126, 20]. Performance is evaluated by averaging over all iterations. Under a given computational complexity constraint, the number of states in each class can be chosen according to the principle of Minimum Description Length [6]. The automatic selection of those parameters has not been explored deeply in our current algorithm. Experiments, which will be described, show that with a fairly small number of states, the MHMM algorithm outperforms the single resolution HMM algorithm and other algorithms.

A 2-D MHMM tested is described by Table 5.1(a), which lists the number of states assigned for each class at each resolution. With the MHMM, the average classification error rate computed at the finest resolution is 16.02%. Detailed performance for each iteration is presented in Table 5.2, where the algorithm is referred to as the MAP algorithm since the goal is to maximize the a posteriori probability of states. Fig. 5.5 shows the classification result for Image 6, of which the original is shown in Fig. 4.6(f). The classification error rate for this image is 11.57%. Compared with the results in Chapter 4, the classified image by the

MHMM is both visually cleaner and closer to the hand-labeled classes in terms of classification error rates.

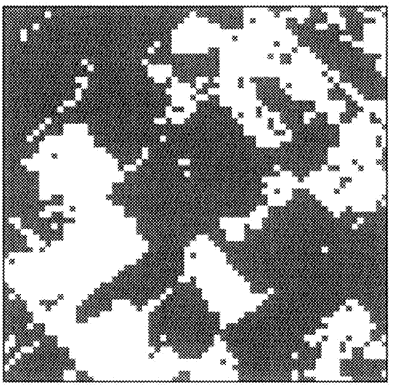


Figure 5.5. The classification result of 2-D MHMM for an aerial image: classification error rate 11.57%. White: man-made, Gray: natural

The two fast algorithms described in Section 5.4 were also applied based on models with the same number of states listed in Table 5.1(a). The performance is presented in Table 5.2. To compare the various algorithms applied to aerial image segmentation, the average classification performance obtained by six-fold cross-validation is listed in Table 5.3. The algorithms are referred to as shortened names as below:

1. CART 1: the decision tree classifier trained on both inter- and intra-block features by CART.
2. CART 2: the decision tree classifier trained on only intra-block features by CART.
3. LVQ1: version 1 of Kohonen's learning vector quantization algorithm based on both inter- and intra-block features.
4. HMM: the MAP classifier based on the single resolution 2-D HMM.
5. MHMM: the MAP classifier based on the multiresolution 2-D HMM.
6. Fast 1: Fast Algorithm 1 based on the multiresolution 2-D HMM.
7. Fast 2: Fast Algorithm 2 based on the multiresolution 2-D HMM.

On average, CART 1 and LVQ1 perform about equally well. The Bayes VQ algorithm was used in [111] to segment the aerial images.

Algorithm	Iteration	sensitivity	specificity	PVP	P_e
MAP	1	0.6409	0.9349	0.8515	0.1733
	2	0.8669	0.7040	0.9271	0.1636
	3	0.9482	0.7192	0.7326	0.1782
	4	0.6146	0.9360	0.8827	0.2051
	5	0.8743	0.8929	0.9983	0.1255
	6	0.8856	0.8837	0.7913	0.1157
	Ave.	0.8051	0.8451	0.8639	0.1602
Fast 1	1	0.7313	0.8581	0.7502	0.1886
	2	0.8932	0.6271	0.9123	0.1566
	3	0.9613	0.5034	0.6111	0.2914
	4	0.6404	0.8484	0.7679	0.2430
	5	0.8124	0.5893	0.9930	0.1906
	6	0.7806	0.8373	0.7049	0.1816
	Ave.	0.8032	0.7106	0.7899	0.2086
Fast 2	1	0.7188	0.8702	0.7634	0.1855
	2	0.9087	0.6532	0.9192	0.1392
	3	0.9624	0.5129	0.6159	0.2857
	4	0.6673	0.8546	0.7823	0.2277
	5	0.8490	0.6250	0.9939	0.1541
	6	0.7639	0.8531	0.7214	0.1766
	Ave.	0.8117	0.7282	0.7994	0.1948

Table 5.2. Classification performance of 2-D MHMM

Algorithm	sensitivity	specificity	PVP	P_e
CART 1	0.8528	0.7126	0.7530	0.2158
CART 2	0.8097	0.7340	0.7505	0.2408
LVQ1	0.8187	0.7419	0.7691	0.2183
HMM	0.7795	0.8203	0.8381	0.1880
MHMM	0.8051	0.8451	0.8639	0.1602
Fast 1	0.8032	0.7106	0.7899	0.2086
Fast 2	0.8117	0.7282	0.7994	0.1948

Table 5.3. The comparison of classification performance for aerial images

BVQ achieves an error rate of about 21.5%, nearly the same as that of CART. The HMM algorithm improves CART and LVQ1 by roughly 13%. The MHMM algorithm further improves the HMM by 15%. It is shown by Table 5.3 and 4.1 that the MHMM algorithm achieves lower

error rates for all the testing images than the HMM algorithm, CART, and LVQ1.

As is mentioned in Section 5.2, some multiresolution models consider only inter-scale statistical dependence. To test whether the intra-scale dependence assumed in the 2-D MHMM is redundant given the inter-scale dependence, a 2-D MHMM discarding the intra-scale dependence was evaluated by six-fold cross-validation. With this new model, given the state of a block, the states of its child blocks are assumed independent. The number of states assigned to each class at each resolution in the new 2-D MHMM as well as all the other parameters controlling the computational complexity of the algorithm are the same as those used for the MHMM described in Table 5.1(a). The average error rate achieved by the new model is 17.26%, whereas the average error rate with the previous model is 16.02%. The experiment thus demonstrates the additional improvement on classification by introducing the intra-scale dependence.

Comparison was made extensively with the quadtree MSRF developed by Bouman and Shapiro [17]. The quadtree model assumes that, at the finest resolution, the probability density function of every class is a Gaussian mixture, which is equivalent to an HMM with several states in one class each corresponding to a component of the mixture [139]. At all the coarse resolutions, since features do not exist and only the prior probabilities of classes are considered, each class can be viewed as one state. Consequently, we examined a 2-D MHMM with parameters shown in Table 5.1(b). As the quadtree model ignores intra-scale dependence, the 2-D MHMM was trained with the intra-scale dependence dismissed. Such a 2-D MHMM has the same underlying state process as the quadtree model. Since the MSRF assumes features observed only at the finest resolution, when applying the 2-D MHMM to classification, we blocked the effect of features at the two coarse resolutions and only used the prior probabilities of classes for computing the joint posteriori probability of states. The classification error rate obtained by cross-validation is 18.89%, higher than the error rate obtained with the HMM. For this 2-D MHMM, when features at the coarse resolutions are used, the error rate is reduced to 17.63%.

Although a more advanced MSRF, namely the pyramid graph model, is also explored in [17] for segmentation, comparison is constrained to the quadtree model because equivalence cannot be established between the pyramid graph model and a 2-D MHMM. The former assumes that the state of a block depends on the states of blocks in a neighborhood

at the next coarser scale, while the latter assumes dependence on the parent block and the sibling blocks at the same scale.

Experiments were performed on a Pentium Pro 230MHz PC with a LINUX operating system. For both the single resolution HMM and the MHMM, computational complexity depends on many parameters including the number of states in a model and parameters that control the extent of approximation taken by the path-constrained Viterbi algorithm. Instead of comparing computational time directly, we compare classification performance given roughly equal computational time. The average CPU time to classify a 512×512 aerial image with the HMM described previously is 200 seconds. With a 2-D MHMM described in Table 5.1(c) and somewhat arbitrarily chosen parameters required by the path-constrained Viterbi algorithm, the average user CPU time to classify one image is 192 seconds, slightly less than that with the HMM. The average classification error rate is 17.32%, 8% lower than the error rate achieved with the HMM. By using the more sophisticated model given by Table 5.1(a) and more computation, the error rate can be improved further to 16.02%. With the HMM, however, using more sophisticated models and more computation does not yield considerable improvement in performance.

The average user CPU time to classify one aerial image is 0.2 seconds for Fast Algorithm 1 and 7.3 seconds for Fast Algorithm 2, much faster than the MAP algorithms based on HMMs and MHMMs. The computation time of Fast Algorithm 1 is very close to that of CART, which is 0.16 second on average. In all cases, the classification time provided here does not include the common feature computation time, which is a few seconds. In the case of Fast Algorithm 1, the feature computation is the primary computational cost.

Chapter 6

TESTING MODELS

No amount of experimentation can ever prove me right; a single experiment can prove me wrong.

—Albert Einstein

We model images by two dimensional hidden Markov models in Chapter 4 and extend the models to multiresolution in Chapter 5. So far we have not asked ourselves how accurate the models are. It is obvious that the potential of the algorithms described in Chapter 4 and 5 ought to depend to some extent upon the validity of the hidden Markov models. Although good results are achieved by algorithms based on the HMMs, which intuitively justify the models, we examine the validity of the HMMs for images more formally by testing their goodness of fit in this chapter. The main reason for proposing the models is to balance their accuracy and computational complexity; that they are absolutely correct is not really an issue. The purpose of testing is thus more for gaining insight into how improvements can be made rather than for arguing the literal truthfulness of the models.

6.1 HYPOTHESIS TESTING

Hypothesis testing is a process to decide whether an observation indicates yes or no for a certain question. It is a common practice in scientific work. The procedure of showing support for claims by experimental results is essentially hypothesis testing. One area in which hypothesis testing is the key issue is the test of drug effect. When a new

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

drug is manufactured, a few questions are asked. What is the maximum safe dose? Is the drug effective for that which it aims to cure? Does the drug have any side effects? To answer such questions, a drug company normally applies the drug first to animals, and then to human beings to collect experimental results. Hypothesis testing techniques are applied to analyze the experimental records and hopefully can provide answers to those questions. In the case that conclusions cannot be reached convincingly, further experiments have to be carried out.

In this section we introduce the widely used formalization of hypothesis testing largely due to Neyman and Pearson. Readers are referred to [14] for detailed study of hypothesis testing. Suppose we observe X drawn from distribution P_θ, $\theta \in \Theta$. The *hypothesis* H specifies that $\theta \in \Theta_0$, a subset of Θ. We write $H : \theta \in \Theta_0$. The *alternative* K specifies that $\theta \in \Theta_1$, another subset of Θ disjoint from Θ_0. The union of Θ_0 and Θ_1 may not cover the entire set Θ. Based on X, we decide whether to *accept* H and claim that $\theta \in \Theta_0$, or reject H (accept K), and claim that $\theta \in \Theta_1$. "Acceptance" is used purely as a terminology here. In scientific problems, "acceptance" might mean more data needed rather than making a claim.

The rule of accepting H or K is specified by the *acceptance region* or its complement, *rejection region* of the X. The rejection region is also referred to as the *critical region*. We often decide whether H is true or false based on a statistic $T(x)$, a function of x. In most problems, we can choose T so that T tends to be small if H is true. Thus, the critical region is $\{x : T(x) \geq c\}$.

The performance of a test is measured by the probability of making correct judgments. There are two types of error we can commit: we can reject the hypothesis when it is true; or we can accept the hypothesis when it is false. The first of these errors is called *type I*, and the second *type II*. Associated are a few important definitions. The *power* of a test against the alternative θ is the probability of rejecting H when θ is true. We say that a test has *level* (*of significance*) α, or a test rejects H at level α, if for all $\theta \in \Theta_0$, the probability of rejecting θ when it is true is less than or equal to α. Since a test with level α is also of level $\alpha' > \alpha$, what is of interest is the minimum level of significance, defined as the *size* of the test. Obviously, the size of a test is the upper bound on the probability of type I error. Another important quantity of a test is the *observed size* or *p-value*. The p-value is the smallest level of significance α at which a test would reject H on the basis of the observed outcome x.

6.2 TEST OF NORMALITY

A 2-D HMM assumes that given its state, a feature vector is Gaussian distributed. The parameters of the Gaussian distribution depend on the state. In order to test the normality of feature vectors in a particular state, the states of the entire data set are searched according to the MAP rule using an estimated model. Feature vectors in each state are then collected as data to verify the assumption of normality. The test was performed on the aerial image data set described in Chapter 4. The model used is the same model trained for classification in Section 4.11.2, which has 5 states for the natural class and 9 states for the man-made class.

The test of normality is based on the well-known fact that a multivariate normal distribution with covariance proportional to the identity is uniform in direction. No matter the covariance, its projection onto any direction has a normal distribution. For a general Gaussian distribution, a translation followed by a linear transform can generate a random vector with the unit spherical normal distribution, perhaps in dimension lower than that of the original data. This process is usually referred to as decorrelation, or whitening. Recall that for a k-dimensional, nonsingular Gaussian random vector U with covariance matrix Σ and mean vector μ, the pdf is

$$f(u) = \frac{1}{\sqrt{(2\pi)^k \det(\Sigma)}} e^{-\frac{1}{2}(u-\mu)^t \Sigma^{-1}(u-\mu)} .$$

The decorrelation matrix is $\Sigma^{-\frac{1}{2}}$, the inverse of matrix $\Sigma^{\frac{1}{2}}$ which satisfies $\Sigma^{\frac{1}{2}} \cdot (\Sigma^{\frac{1}{2}})^t = \Sigma$. The linear transform of the mean subtracted u, $\Sigma^{-\frac{1}{2}}(u - \mu)$ obeys the unit normal distribution.

Normal probability plots (for details see [14]) are applied to check the normality of random variables. Suppose random variable Z is the projection of U onto a direction v, $\|v\| = 1$; and $\{Z_{(1)}, Z_{(2)}, ..., Z_{(n)}\}$ is the order statistics of sample set $\{Z_1, Z_2, ..., Z_n\}$. The normal probability plot is the plot of points $(Z_{(i)}, \Phi^{-1}(i/n))$, where $\Phi(\cdot)$ is the cdf of the normal distribution. If Z follows the normal distribution, the plot should be roughly a straight line through the origin at angle 45°.

For each state of the model, the normality of feature vectors was tested. The data were decorrelated and then projected onto a variety of directions. The normal probability plot for every projection was drawn. The projection directions include individual components of the vector, the average of all the components, differences between all the pairs of components, and 7 random directions. Since the feature vectors are 8

dimensional, 44 directions were tested in all. It would be cumbersome to show all the plots. Therefore, details are shown for one representative state.

Fig. 6.1(a) presents the normal probability plots of all the 8 components. Counted row-wise, the 7th and 8th plots in part (a) show typical "bad" fits to the normal distribution for projections onto individual components, whereas the 1st plot is a typical "good" fit. "Bad" plots are characterized by data that are truncated below and with heavy upper tails. Most plots resemble the 4th to 6th plots in part (a), which are slightly curved. Fig. 6.1(b) shows plots for the average of all the components (the first plot) and projections onto random directions. We see that the average and the projections onto the random directions fit better with the normal distribution than do the individual components. These owe to a "central limit effect" and are shown consistently by the other states. Fig. 6.2 presents the normal probability plots for differences between some pairs of components. Differences between components also tend to fit normality better than the individual components for all the states. A typical "good" fit is shown by the 3rd and the 7th plots, which are aligned with the ideal straight line in a large range. A typical "bad" fit is shown by the 2nd and 6th plots, which only deviate slightly from the straight line. But here, "bad" means heavy lower tails and truncated upper tails.

6.3 TEST OF THE MARKOVIAN ASSUMPTION

For 2-D HMMs, it is assumed that given the states of the two neighboring blocks (right to the left and above), the state of a block is conditionally independent of the other blocks in the "past." In particular, "past" means all the blocks above and to the left. See Section 4.5 for details. In this section, we test three cases of conditional independence given the states of block (m, n) and $(m-1, n+1)$: the independence of $\{(m-1, n), (m, n+1)\}$, $\{(m, n-1), (m, n+1)\}$, and $\{(m, n+1), (m-2, n+1)\}$. For notational simplicity, we refer to the three cases as Case 1, 2, and 3, which are shown in Fig. 6.3.

For 2-D MHMMs, two assumptions are tested. One is the Markovian assumption that given the state of a block at resolution r, the state of its parent block at resolution $r-1$ and the states of its child blocks at resolution $r+1$ are independent. A special case investigated is the conditional independence of block $(i_0, j_0) \in \mathbb{N}^{(r-1)}$ and one of its grandchild blocks (i_2, j_2), $i_2 = 4i_0$, $j_2 = 4j_0$, $(i_2, j_2) \in \mathbb{N}^{(r+1)}$. Figure 6.4(a) shows

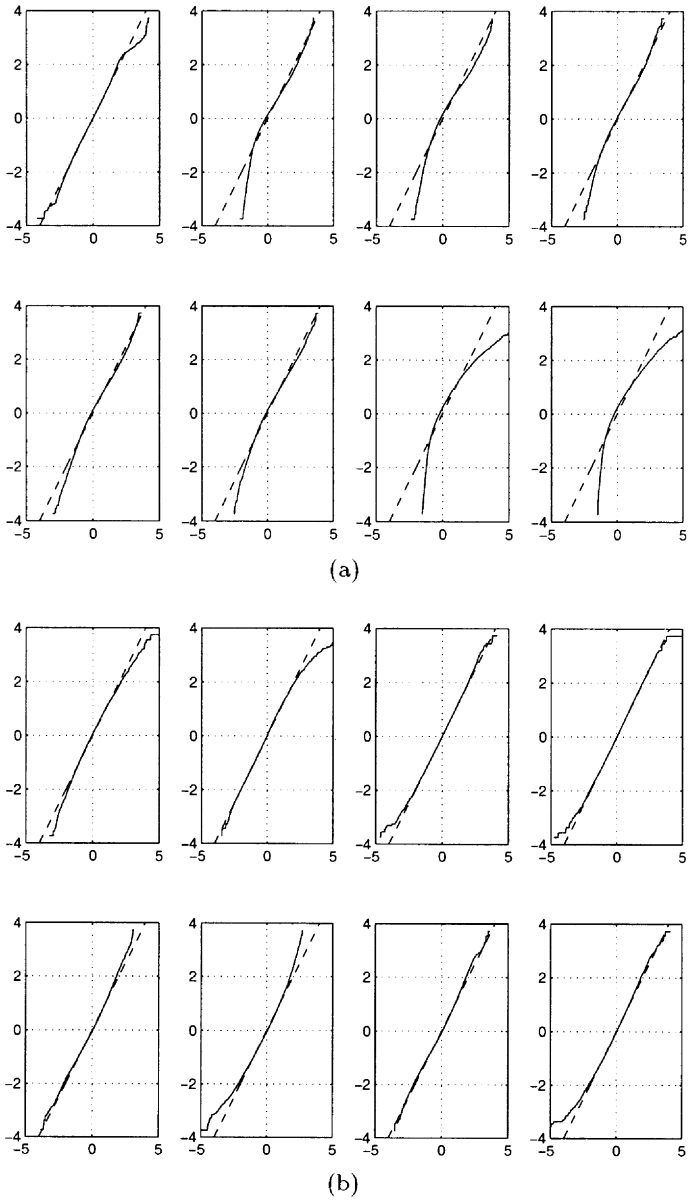


Figure 6.1. Normal probability plots for one state: (a) Each component, (b) The average of all the components and projections onto random directions

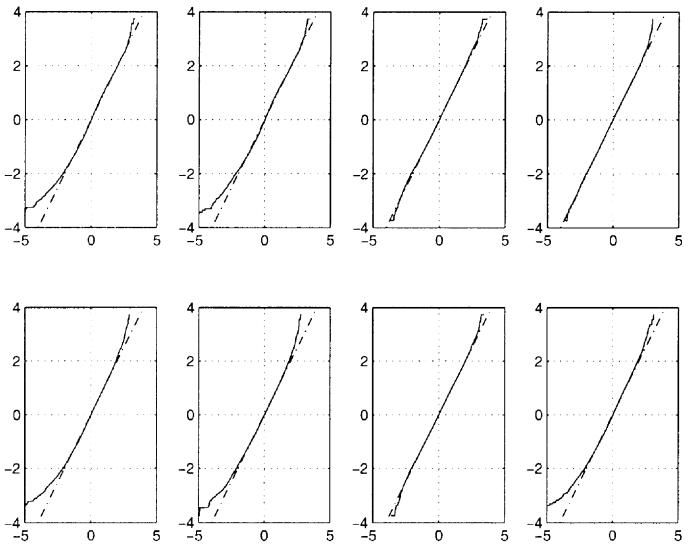


Figure 6.2. Normal probability plots for one state: differences between pairs of components

the conditioned and tested blocks. The other assumption tested is that given the states of parent blocks at resolution r, the states of non-sibling blocks at resolution $r+1$ are independent. In particular, a worst possible case is discussed, that is, given the states of two adjacent blocks at resolution r, the states of their two adjacent child blocks are independent. The spatial relation of those blocks is shown in Figure 6.4(b).

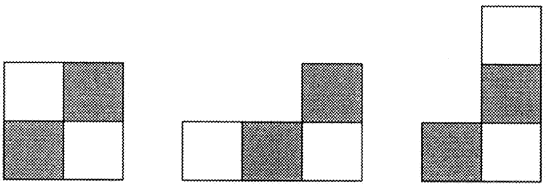


Figure 6.3. Tests of conditional independence. The states of gray blocks are conditional states; the states of white blocks are states upon which tests for independence are performed.

As with the test of normality in the previous section, the test of independence was performed on the aerial image data set. States were searched by the MAP rule. Then, the test was performed on those states. In the case of the HMM, the same model for the test of normality was

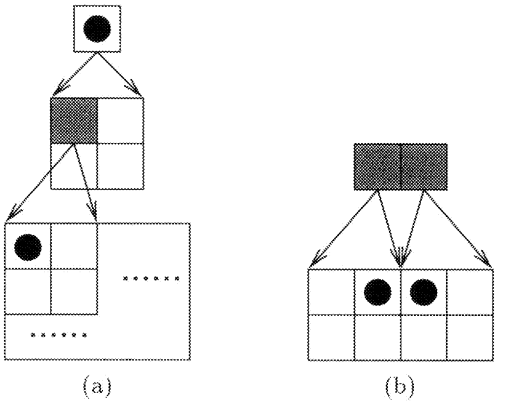


Figure 6.4. Tests of the MHMM assumptions: (a) the Markovian property across resolutions, (b) the conditional independence of child blocks descended from different parent blocks. The states of gray blocks are conditional states; the states of dotted blocks are states upon which tests for independence were performed.

used. For the MHMM, the 3 resolution model described by Table 5.1(c) was used.

To test the conditional independence, for each fixed pair of conditional states, a permutation χ^2 test [82, 19] was applied. The idea of a permutation test dates back to Fisher's exact test [51] for independence in a 2×2 contingency table [14]. In principle, Fisher's exact test can be generalized to testing independence of an $r \times c$ contingency table. The difficulty is with computational complexity, which seriously constrains the use of Fisher's exact test. Mehta and Patel [94] have taken a network approach to achieve computational feasibility. Boyett [19] proposed subsampling random permutations to reduce computation. The random permutation approach is taken here.

Suppose the independence test is for block $(m-1, n)$ and $(m, n+1)$ (Case 1 of the HMM). The entry $a_{i,j}$ in the contingency table is the number of occurrences for $(m-1, n)$ being in state i and $(m, n+1)$ being in state j. Denote the marginal counts by

$$r_i = \sum_{j=1}^{M} a_{i,j} \ , \quad c_j = \sum_{i=1}^{M} a_{i,j} \ , \quad \text{and} \quad n = \sum_{i=1}^{M} \sum_{j=1}^{M} a_{i,j} \ ,$$

where M is the total number of states. For each $a_{i,j}$, generate $a_{i,j}$ indices $\begin{pmatrix} i \\ j \end{pmatrix}$. A list is generated by assembling indices $\begin{pmatrix} i \\ j \end{pmatrix}$ in a certain

order. A permutation is obtained by randomly permuting the second number j while fixing the order of the first number i. For an example 2×2 contingency table

$$\begin{pmatrix} 2 & 1 \\ 3 & 2 \end{pmatrix},$$

a list as follows is generated

$$\begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 1 & 1 & 1 & 2 & 2 \end{pmatrix}.$$

Fixing the first row and permuting the second row may yield a list

$$\begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 2 & 1 & 2 & 1 & 1 & 1 \end{pmatrix}.$$

The permutation yields new counts for the number of $\begin{pmatrix} i \\ j \end{pmatrix}$ in the list, denoted as $\hat{a}_{i,j}$. Note that the marginal counts remain, that is, $r_i = \sum_{j=1}^{M} \hat{a}_{i,j}$, $c_j = \sum_{i=1}^{M} \hat{a}_{i,j}$. For the particular case of the above list, the new contingency table is

$$\begin{pmatrix} 1 & 2 \\ 4 & 1 \end{pmatrix}.$$

For both the original contingency table and those generated by random permutations, we compute Pearson's χ^2 statistic [14],

$$\chi^2 = n \sum_{i=1}^{M} \sum_{j=1}^{M} \frac{(a_{i,j} - r_i c_j / n)^2}{r_i c_j}. \tag{6.1}$$

The quantity $a_{i,j}$ is replaced by $\hat{a}_{i,j}$ for tables generated by permutations. Denote the χ^2 statistic of the original contingency table as χ^2_{obs}. The p-value for the original contingency table is

$$p = \frac{\text{number of contingency tables such that } \chi^2 \geq \chi^2_{obs}}{\text{number of permutations} + 1}.$$

The number of permutations used is 1000.

Since conditional independence is of concern, p-values were computed for each condition. The HMM in discussion has a total of 14 states, which yield 14×14 conditions, each corresponding to a pair of states for

neighboring blocks above and to the left. We thus have 196 p-values for each case of the independence tests shown in Fig. 6.3. The histograms of p-values for the three cases are plotted in Fig. 6.5. For Case 1, 2, and 3, the medians of the p-values are 0.055, 0.462, and 0.443 respectively. The percentage of p-values above 0.05 for Case 1, 2, and 3 is around 50%, 95%, and 90% correspondingly. Results show that Case 2 and 3 fit the conditional independence assumption about equally well, and much better than Case 1. This indication coincides with our intuition. We expect that the conditional independence assumption is less true for Case 1 since the two blocks under examination touch at a corner.

For contingency tables with small p-values, equivalent to relatively large χ^2_{obs}, it is of interest to elaborate upon the relation between $(a_{i,j} - r_i c_j/n)^2/(r_i c_j)$, the summand in Equation (6.1), and $(r_i c_j)/n$. If summands with small $(r_i c_j)/n$ contribute the most to χ^2_{obs}, the failure of the independence assumption is caused mainly by (i,j)'s that occur with low probability. The impreciseness of the model thus should have less effect. Fig. 6.6 plots a function

$$F(\gamma) = \sum_{(i,j):\frac{r_i c_j}{n}<\gamma} \frac{(a_{i,j} - r_i c_j/n)^2}{r_i c_j}$$

for an example contingency table in Case 1. The p-value, 0.01, for this contingency table is low. On the other hand, the majority of χ^2_{obs} (more than 82%) result from (i,j)'s with $r_i c_j < 0.1n$, which occur with very low probability. To obtain a global picture for each of the three cases, we selected all the contingency tables with p-values below 0.05 and computed statistics $F(1.0) = \sum_{(i,j):\frac{r_i c_j}{n}<1.0} \frac{(a_{i,j}-r_i c_j/n)^2}{r_i c_j}$. The histograms of $F(1.0)$ for the three cases are shown in Fig. 6.7. We see that for every contingency table in every case, $F(1.0)$ is greater than or equal to 0.5, which indicates that half or more of χ^2_{obs} is due to infrequently occurring (i,j)'s with $r_i c_j < n$.

To test the Markovian property across resolutions for the 2-D MHMM, p-values were computed for each of the 6 conditional states at Resolution 2. Among the 6 p-values, one is 0.89, another is 0.17, and all the other four are below 0.05, indicating strong dependence between Resolution 1 and 3. However, the Markovian property across resolutions is usually assumed to maintain computational tractability. For the testing of conditional independence of non-sibling blocks, there are $6 \times 6 = 36$ state pairs of parent blocks, each of which is a condition. The median of the 36 p-values is 0.44. About 70% of them are above 0.05. Therefore,

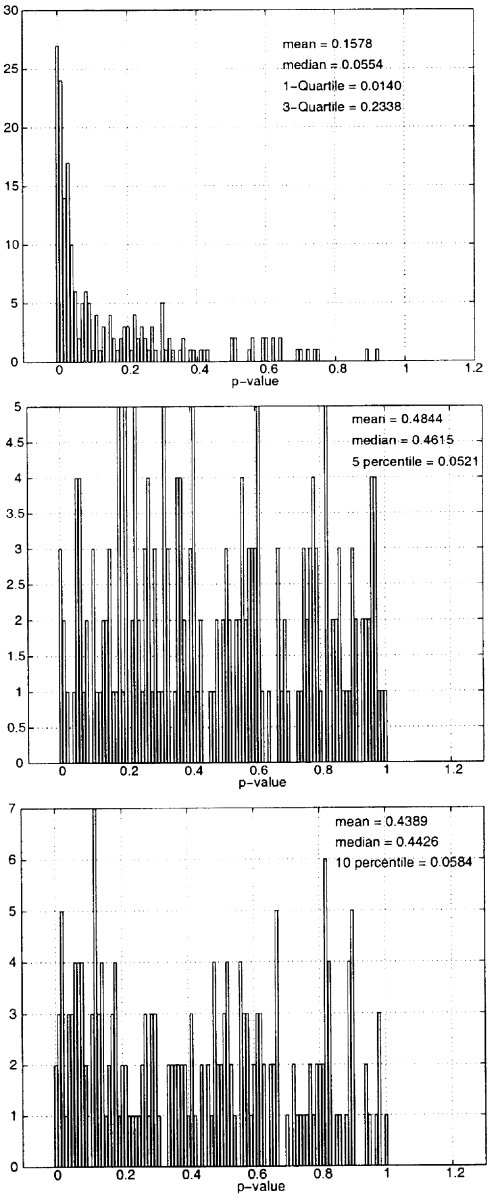


Figure 6.5. Histograms of p-values. Top to bottom: Cases 1 to 3

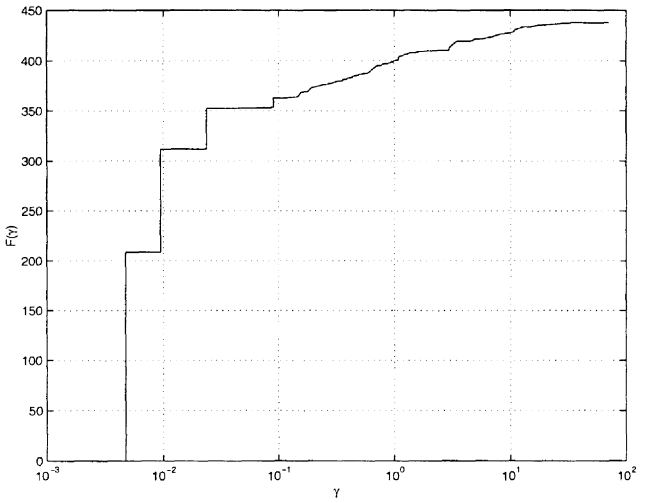


Figure 6.6. $F(\gamma) = \sum_{(i,j):\frac{r_i c_j}{n}<\gamma} \frac{(a_{i,j}-r_i c_j/n)^2}{r_i c_j}$ for a contingency table in Case 1

for most conditions, there is no strong evidence for dependence between blocks descended from different parents.

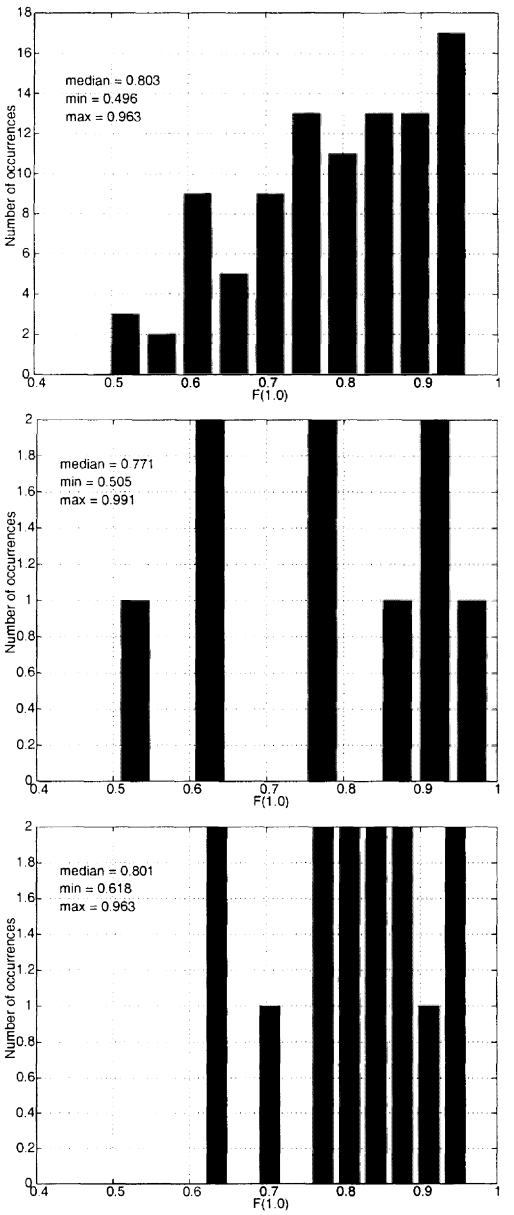


Figure 6.7. Histograms of statistics $F(1.0)$ for contingency tables with p-values below 0.05 in Case 1, 2 and 3. Top to bottom: Cases 1 to 3

Chapter 7

JOINT COMPRESSION AND CLASSIFICATION

All that is really necessary for survival of the fittest, it seems, is an interest in life, good, bad or peculiar.

—Grace Paley

As is described in Chapter 4, the classification algorithm based on 2-D HMMs improves classification by incorporation of context information. The purpose of this chapter is to develop a joint compression and classification algorithm based on 2-D HMMs so that the improved classification performance leads to improved compression and classification jointly. We call this algorithm the HMM VQ algorithm. Comparisons are made with BVQ, a joint compression and classification algorithm developed by Oehler, Gray, Perlmutter, Olshen, *et al.*, which is described in Section 3.4.

As with the BVQ algorithm, a new distortion measure is defined as a weighted sum of compression distortion and the penalty for misclassification. The penalty for misclassification used by BVQ is the Bayes risk. In order to take statistical dependence among image blocks into consideration, the HMM VQ algorithm defines the penalty for misclassification as the negative of the maximum joint log likelihood of states and feature vectors based on a 2-D HMM. Since the 2-D HMM assumes that the blocks of an image are statistically dependent, to classify the image optimally, decisions are made jointly for the blocks. Consequently, blocks with the same feature vector may be classified differently according to their context. A vector quantizer designed with a 2-D HMM has the

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

same property. Hard boundaries between quantization cells vanish due to context dependent encoding.

7.1 DISTORTION MEASURE

A training sequence of image data is represented by $\mathcal{L} = \{(x_{k,l}, y_{k,l}) : (k,l) \in \mathbb{N}\}$, where $\mathbb{N} = \{(k,l) : 0 \leq k < w, 0 \leq l < z\}$ denotes the collection of blocks in the training image. The random vector $X_{k,l}$ consists of the intensities of pixels in block (k,l), and $Y_{k,l}$ is the class of the block. The domain of $X_{k,l}$ is A_X, $A_X \subset \Re^{\Theta}$, and the domain of $Y_{k,l}$ is A_Y. Assume that there exists a random process $\{(u_{k,l}, s_{k,l}) : (k,l) \in \mathbb{N}\}$ associated with the training image. The random vector $U_{k,l}$ is the feature vector for block (k,l), which is a function of $X_{k,l}$. The random variable $S_{k,l}$ is the underlying state of the block. It is assumed that $\{(U_{k,l}, S_{k,l}) : k,l \in \mathbb{N}\}$ is generated by a 2-D HMM.

Assume that a 2-D HMM has already been estimated from training data. To classify an image with feature vectors $\{u_{k,l} : (k,l) \in \mathbb{N}\}$, we search for the states $\mathbf{s} = \{s_{k,l} : (k,l) \in \mathbb{N}\}$ that yield the maximum a posteriori probability given the feature vectors, that is,

$$\max_{\mathbf{s}}{}^{-1} P\{s_{k,l} : (k,l) \in \mathbb{N} \,|\, u_{k,l} : (k,l) \in \mathbb{N}\} ,$$

which is equivalent to maximizing the joint likelihood with the feature vectors

$$\max_{\mathbf{s}}{}^{-1} P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} .$$

The classes are then mapped from the states. We denote the mapping from states to classes by $C(s_{k,l})$. Although we cannot claim in general that the states with the maximum a posteriori probability necessarily yield the classes with the maximum a posteriori probability given the feature vectors, we use the joint likelihood of the states and the feature vectors, $P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\}$, as an indication of classification risk because the evaluation of $P\{y_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\}$ is computationally too intensive. It is reasonable to make such a replacement because the optimal decision on the states should yield a good decision on the classes. As a result, for the encoder $\tilde{\alpha}$, given $\hat{y}_{k,l}$, the estimation of $y_{k,l}$, the penalty for misclassification is

$$- \max_{s_{k,l}:C(s_{k,l})=\hat{y}_{k,l}} \log(P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\}) ,$$

that is, the negative of the maximum log likelihood of the feature vectors and states that can be mapped into classes $\hat{y}_{k,l}$. For the classifier at the

receiving end, to decide the optimal class of an index, the goal is to minimize the classification error rather than the error rate of the states.

The distortion measure is defined as a Lagrangian function formed in the manner of [29, 111]. Suppose $x_{k,l}$ is encoded as $i_{k,l}$. In our study, the compression distortion $d(x_{k,l}, \tilde{\beta}(i_{k,l}))$ is assumed to be the mean squared error between $x_{k,l}$ and the codeword to which it is decoded. The Lagrangian distortion between an input image $\{x_{k,l} : (k,l) \in \mathbb{N}\}$ and encoder output indices $\{i_{k,l} : (k,l) \in \mathbb{N}\}$ is defined as

$$\rho_\lambda = \frac{1}{wz}\left[\sum_{(k,l)\in\mathbb{N}} d(x_{k,l}, \tilde{\beta}(i_{k,l})) - \lambda \max_{s_{k,l}:C(s_{k,l})=\kappa(i_{k,l})} \log(P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\})\right],$$

where wz is the number of blocks in the image.

The interaction between compression and classification is not as obvious as with BVQ [63, 108] because the encoder and the classifier affect each other indirectly through the choice of states. For each $x_{k,l}$ encoded as $i_{k,l}$, its class $\kappa(i_{k,l})$ determined by the classifier restricts the possible states for block (k,l) to be those satisfying $C(s_{k,l}) = \kappa(i_{k,l})$. The Lagrangian distortion is defined for the entire image because the likelihood cannot be separated into independent items for each block. In practice, due to computational complexity, we do not encode an entire image jointly; instead, we divide the image into subimages containing a set of image blocks and encode the subimages separately. The distortion ρ_λ is the distortion for one subimage. The expected value of ρ_λ quantifies the performance of the vector quantizer. Define the expected distortion for the vector quantizer as in [63]:

$$J_\lambda(\tilde{\alpha}, \tilde{\beta}, \kappa) = E(\rho_\lambda) = D(\tilde{\alpha}, \tilde{\beta}) + \lambda B(\tilde{\alpha}, \kappa) ;$$

$$D(\tilde{\alpha}, \tilde{\beta}) = E(d(X, \tilde{\beta}(\tilde{\alpha}(X)))) ;$$

$$B(\tilde{\alpha}, \kappa) = -\frac{1}{wz} E\left[\max_{s_{k,l}:C(s_{k,l})=\kappa(i_{k,l})} \log(P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\})\right] .$$

7.2 OPTIMALITY PROPERTIES AND THE ALGORITHM

As with BVQ [63], there are necessary conditions for overall optimality of a vector quantizer based on a 2-D HMM. They lead to an iterative

algorithm for designing the quantizer. The conditions for the optimal decoder and the optimal classifier are the same as those stated in [63] for the particular case of classification error rate being the penalty for misclassification:

- Given $\tilde{\alpha}$ and κ, the optimal decoder is

$$\tilde{\beta}(i) = \min_{\hat{x} \in \hat{A}_X}{}^{-1} E[d(x, \hat{x}) \mid \tilde{\alpha}(X) = i] \, .$$

- Given $\tilde{\alpha}$ and $\tilde{\beta}$, the optimal classifier is

$$\kappa(i) = \max_{m \in A_Y}{}^{-1} \hat{P}(Y = m \mid \tilde{\alpha}(X) = i) \, ,$$

  that is, a majority vote on the classes of all the vectors encoded to the index. The $\hat{P}(\cdot)$ denotes the empirical frequency for a class based on all the vectors encoded to the index.

- Given κ and $\tilde{\beta}$, then the optimal encoder is

$$\tilde{\alpha}(x_{k,l} : (k,l) \in \mathbb{N}) = \min_{i_{k,l}:(k,l)\in\mathbb{N}}{}^{-1} \left\{ \sum_{(k,l)\in\mathbb{N}} d(x_{k,l}, \tilde{\beta}(i_{k,l})) - \right.$$

$$\left. \lambda \max_{s_{k,l}:C(s_{k,l})=\kappa(i_{k,l})} \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} \right\} .$$

The algorithm iterates the three steps in succession. Since we cannot claim the penalty for misclassification used in the encoder is an increasing function of the classification error rate, the algorithm is not guaranteed to be descending. However, simulations performed on both synthetic data and real image data with a wide range of λ have always provided descending Lagrangian distortions.

7.3 INITIAL CODEBOOK

To design the initial quantizer, pure classification with the 2-D HMM is applied to the training data. Based on the classification, a codebook is designed for each class using the Lloyd algorithm. The combination of all the codebooks forms the initial codebook. The corresponding classes of codewords form the initial classifier.

To decide the initial number of codewords for each class, techniques of bit allocation are applied. Since the bit allocation for initial codebook design is adjusted automatically by the HMM VQ algorithm through

the updating of classifier κ, the initial bit allocation is not critical for the final performance. We thus make several approximations to come up with a simple bit allocation scheme. Suppose the total number of codewords is N and the number of codewords for class m, $m = 1, ..., M$, is N_m. We want to choose N_m under the constraint $\sum_{m=1}^{M} N_m = N$ such that the average distortion, MSE, is minimized. By Gersho's asymptotic approximation of MSE [59], the MSE of class m is

$$D_m \approx A(\Theta) N_m^{\frac{-2}{\Theta}} \parallel p_m(x) \parallel_{\Theta/\Theta+2}$$

$$\parallel p_m(x) \parallel_{\Theta/\Theta+2} = \left[\int [p_m(x)]^{\frac{\Theta}{\Theta+2}} dx \right]^{\frac{\Theta+2}{\Theta}} ,$$

where $p_m(x)$ is the pdf of X in class m. Suppose given class $Y = m$, the variance of the nth component of X is σ_{mn}^2, and $\tilde{p}_m(x)$ is its *normalized pdf*

$$\tilde{p}_m(x) = \prod_{n=1}^{\Theta} \sigma_{mn} p_m(Sx) ,$$

where S is a diagonal matrix $diag(\sigma_{m1}, ..., \sigma_{m\Theta})$. It is assumed that $\parallel \tilde{p}_m(x) \parallel_{\Theta/\Theta+2}$ is constant for all values of m. A special case satisfying this assumption is that the components of X are independent and their normalized pdfs are the same, e.g., Gaussian distributions. The MSE for class m is then $A' N_m^{\frac{-2}{\Theta}} \prod_{n=1}^{\Theta} \sigma_{mn}^{2/\Theta}$, where A' is a constant. Assuming that the prior probability of class m is q_m, the average MSE is

$$D = \sum_{m=1}^{M} A' q_m N_m^{\frac{-2}{\Theta}} \prod_{n=1}^{\Theta} \sigma_{mn}^{2/\Theta} .$$

The distortion D is minimized by choosing N_m as

$$\begin{aligned} N_m &= \frac{\gamma_m N}{\sum_{m'=1}^{M} \gamma_{m'}} , \quad m = 1, ..., M \\ \gamma_m &= q_m^{\frac{\Theta}{\Theta+2}} \prod_{n=1}^{\Theta} \sigma_{mn}^{\frac{2}{\Theta+2}} . \end{aligned}$$

The number of codewords allocated to class m is the rounding off of N_m to the nearest integer.

7.4 OPTIMAL ENCODING

According to the iterative algorithm presented in the previous section, it is simple to update the decoder and the classifier. Recall that the optimal encoder is

$$\tilde{\alpha}(x_{k,l} : (k,l) \in \mathbb{N}) = \min_{i_{k,l}:(k,l)\in\mathbb{N}}{}^{-1} \Bigg\{ \sum_{(k,l)\in\mathbb{N}} d(x_{k,l}, \tilde{\beta}(i_{k,l})) - \lambda \max_{s_{k,l}:C(s_{k,l})=\kappa(i_{k,l})} \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} \Bigg\} .$$

Consider

$$\begin{aligned} & \min_{i_{k,l}:(k,l)\in\mathbb{N}} \Bigg\{ \sum_{(k,l)\in\mathbb{N}} d(x_{k,l}, \tilde{\beta}(i_{k,l})) - \lambda \max_{s_{k,l}:C(s_{k,l})=\kappa(i_{k,l})} \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} \Bigg\} \\ = \; & -\max_{s_{k,l}} \max_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} \Bigg\{ \lambda \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} - \sum_{(k,l)\in\mathbb{N}} d(x_{k,l}, \tilde{\beta}(i_{k,l})) \Bigg\} . \end{aligned}$$

Since for fixed $s_{k,l}$, $(k,l) \in \mathbb{N}$,

$$\begin{aligned} & \max_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} \Bigg\{ \lambda \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} - \sum_{(k,l)\in\mathbb{N}} d(x_{k,l}, \tilde{\beta}(i_{k,l})) \Bigg\} \\ = \; & \lambda \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} - \sum_{(k,l)\in\mathbb{N}} \min_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} d(x_{k,l}, \tilde{\beta}(i_{k,l})) , \end{aligned}$$

and the summand in the second term, $\min_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} d(x_{k,l}, \tilde{\beta}(i_{k,l}))$, is simply the minimum mean squared error with a codeword classified as $C(s_{k,l})$, the critical step is thus to find $\{s_{k,l} : (k,l) \in \mathbb{N}\}$ that maximizes

$$\lambda \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} - \sum_{(k,l)\in\mathbb{N}} \min_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} d(x_{k,l}, \tilde{\beta}(i_{k,l})) .$$

The first term is what an image classifier based on a 2-D HMM normally maximizes. Let T_j denote the sequence of states on diagonal j, as shown in Fig. 4.3, and $j = 0, 1, ..., w + z - 2$. It follows from (4.20) that

$$\begin{aligned} & \lambda \log P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\} - \\ & \sum_{(k,l)\in\mathbb{N}} \min_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} d(x_{k,l}, \tilde{\beta}(i_{k,l})) \\ = & \sum_{j=0}^{w+z-2} \left[\lambda \log(P(T_j \mid T_{j-1}) P(u_{k,l} : k + l = j \mid T_j)) - \right. \\ & \left. \sum_{(k,l):k+l=j} \min_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} d(x_{k,l}, \tilde{\beta}(i_{k,l})) \right]. \end{aligned} \tag{7.1}$$

Equation (7.1) demonstrates the one-step memory of the Lagrangian distortion in terms of T_j. We can thus apply the Viterbi algorithm to find the optimal $s_{k,l}$. This is the same method used to search for the optimal states by a pure classifier based on a 2-D HMM. The only difference caused by the compression distortion is an extra cost at each step of the Viterbi transition diagram.

7.5 EXAMPLES

7.5.1 SYNTHETIC DATA

The algorithm was simulated on an extension of Kohonen's example of Gaussian mixture source [77]. As with Kohonen's example, there are two classes. Given the class m, $m = 0, 1$, the two dimensional random vector X consists of independent and identically distributed components with Gaussian distribution $\mathcal{N}(0, \sigma_m^2)$. In particular, $\sigma_0^2 = 1, \sigma_1^2 = 4$. Kohonen assumes that classes of blocks are iid with equal probabilities for class 0 and 1. We assume, however, that the classes are produced by a 2-D HMM. In this case, the classes are the same as the underlying states because only one state is assigned to each class. It is also assumed that the feature vectors are the same as the vectors to be encoded, i.e., x. The specific transition probabilities are as follows

$$\begin{aligned} a_{q,n,r} &= 0.5, \text{ if } q \neq n \\ a_{q,n,r} &= 0.8, \text{ if } q = n = r \\ a_{q,n,r} &= 0.2, \text{ if } q = n \neq r\,, \end{aligned}$$

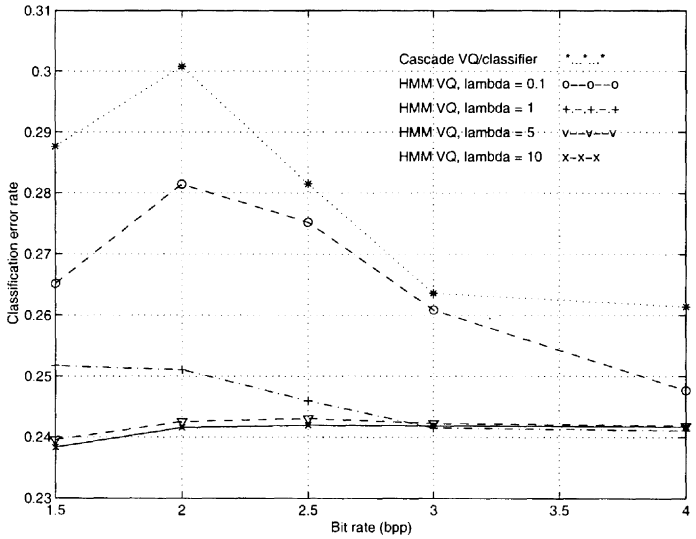
that is, if the classes of two adjacent blocks are different, then the transition probabilities are the same for both classes. If the classes of two

adjacent blocks are the same, the probability of remaining in the state is higher than that of moving to the different one.

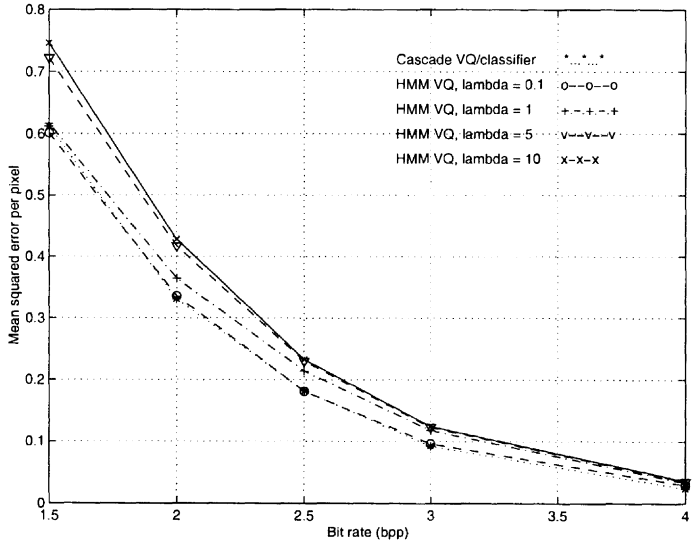
According to this model, a training image of size 256×256 was generated. The training data were used to estimate a 2-D HMM. Based on the estimated HMM, vector quantizers with different distortion weights λ were designed. A test image of size 128×128 was generated to evaluate the quantizers. If the dependence among blocks is ignored, the source is Kohonen's example. The Bayes optimal classifier yields an error probability of 0.264. The classifier with the 2-D HMM obtains an error rate of 0.243 for the test data. The error rate is decreased by exploiting the dependence among vectors. The lower bound provided by the Bayes classifier is valid only for the performance of classifiers making decisions independently on each block.

The compression and classification performance versus bit rates are plotted in Fig. 7.1. Each line in the plots corresponds to a fixed λ, ranging from 0.1 to 10. Comparison is made against a full search Lloyd vector quantizer followed by a Bayes classifier, referred to as the cascade algorithm. Compared with the cascade algorithm, the HMM VQ with $\lambda = 0.1$ yields considerably close compression results and much lower classification error rates. To see directly how compression and classification performance tradeoff through the weighting factor λ, we show the classification error rate versus the mean square error in Fig. 7.2. Each line represents the performance at a fixed bit rate. For all the bit rates, the λ's are $0.1, 1, 5, 10, 100$; and the mean squared error increases with the increase of λ. When λ is sufficiently large, HMM VQ is close to a pure classifier based on a 2-D HMM followed by a vector quantizer. This explains the almost constant classification error rate with $\lambda = 100$ regardless of the bit rate. For each bit rate, although the basic trend is that larger λ improves classification at the cost of increased compression distortion, when λ is too high, the increased compression distortion may not be paid by better classification. For the five bit rates shown in Fig. 7.2, $\lambda = 100$ yields worse results than $\lambda = 10$ for both compression and classification. This fact that tradeoff with compression can actually improve classification is consistent with comments in [20] to the effect that to be greedy for reduction in Bayes risk alone leads to poor classifiers.

At 1.5 bpp (codebook size 8), for Kohonen's Gaussian mixture, the optimal encoders for the Bayes classifier followed by Lloyd VQ assuming independent input vectors, Lloyd VQ, and HMM VQ are shown in Fig. 7.3. As was mentioned before, because of context dependent deci-



(a)



(b)

Figure 7.1. Compression and classification performance of HMM VQ for a Kohonen Gaussian mixture: (a) Compression (MSE), (b) Classification error rate

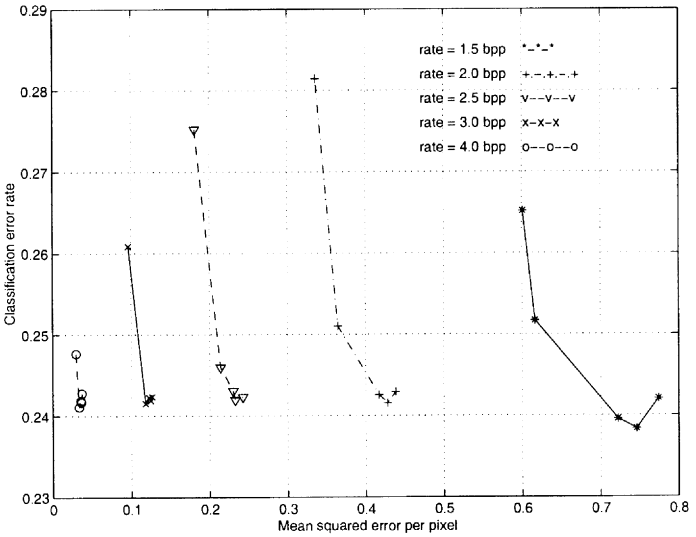


Figure 7.2. Tradeoff between compression and classification with HMM VQ for a Kohonen Gaussian mixture at rates: 1.5, 2.0, 2.5, 3.0, and 4.0 bpp

sions, the encoder of HMM VQ has overlapped quantization cells. The boundaries are drawn according to simulations. The compression and classification performance versus λ at 1.5 bpp are listed in Table 7.1. At $\lambda = 5$ and 10, the vector quantizer achieves better classification than a pure classifier based on HMM. Results at the same bit rate for BVQ with different density estimators are provided in [63]. Some of them are listed in Table 7.2. By taking advantage of the inter-block dependence, the vector quantizer with the 2-D HMM improves both compression and classification.

λ	MSE	P_e
0.1	0.601	0.265
1	0.617	0.251
5	0.722	0.240
10	0.746	0.238
100	0.774	0.242

Table 7.1. MSE and P_e for Kohonen's example achieved by vector quantizers with 2-D HMM

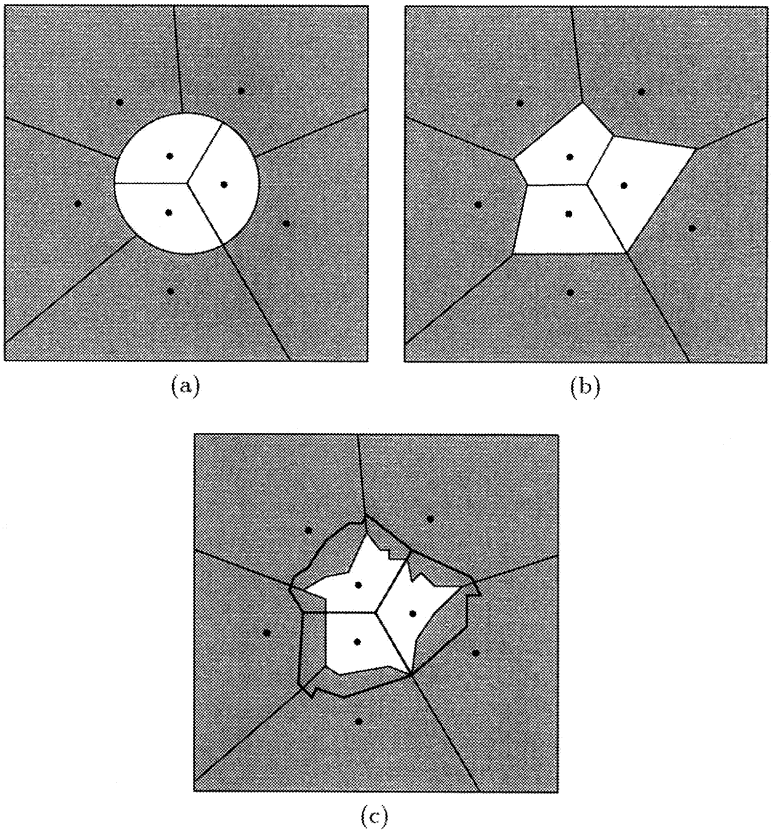


Figure 7.3. Optimal encoders for a Kohonen Gaussian mixture at 1.5 bpp: (a) Bayes classifier followed by Lloyd VQ assuming independent input vectors, (b) Lloyd VQ, (c) HMM VQ, the heavy lines mark the boundaries of the three inner quantization cells, whereas the lighter lines mark the boundaries of the five outer quantization cells

Algorithms	MSE	P_e
BVQ: Inverse halftone estimator	0.655	0.269
BVQ: CART-based estimator	0.653	0.274
Cascade	0.598	0.295
BVQ: TSVQ pmf estimator	0.630	0.270

Table 7.2. MSE and P_e for Kohonen's example achieved by several algorithms

7.5.2 IMAGE DATA

The HMM VQ algorithm was also applied to the data set of aerial images described in Chapter 4. All the results provided were obtained

Iteration	sensitivity	specificity	PVP	P_e
1	0.6732	0.9004	0.7975	0.1832
2	0.9199	0.5877	0.9064	0.1423
3	0.9413	0.7004	0.7183	0.1917
4	0.6241	0.8920	0.8191	0.2256
5	0.8466	0.8973	0.9983	0.1527
6	0.9157	0.8180	0.7148	0.1495
Ave	0.8201	0.7993	0.8257	0.1742

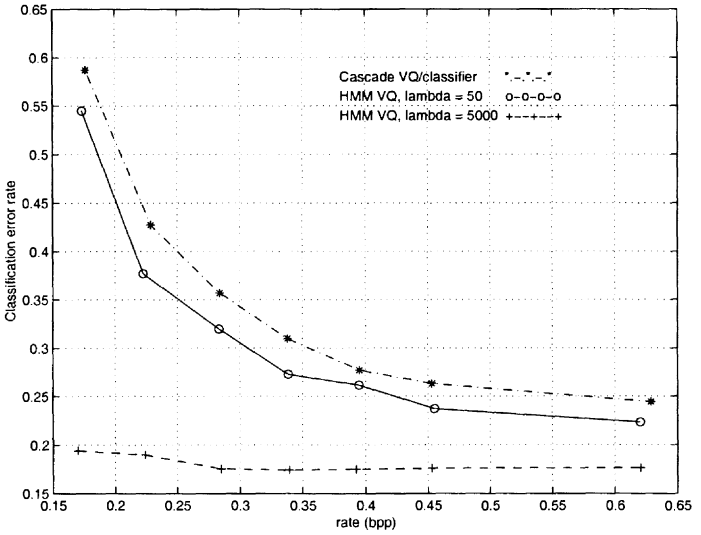
Table 7.3. Classification performance of HMM VQ at 0.338 bpp and $\lambda = 5.0 \times 10^3$

by six-fold cross-validation. Since arithmetic coding was applied to losslessly compress the index output of the quantizer, the rates reported are the ones after arithmetic coding. Experiments show that when a codebook size is small, the entropy coding reduces bit rates by about 10%. Less benefit of entropy coding is obtainable when the codebook size is large.

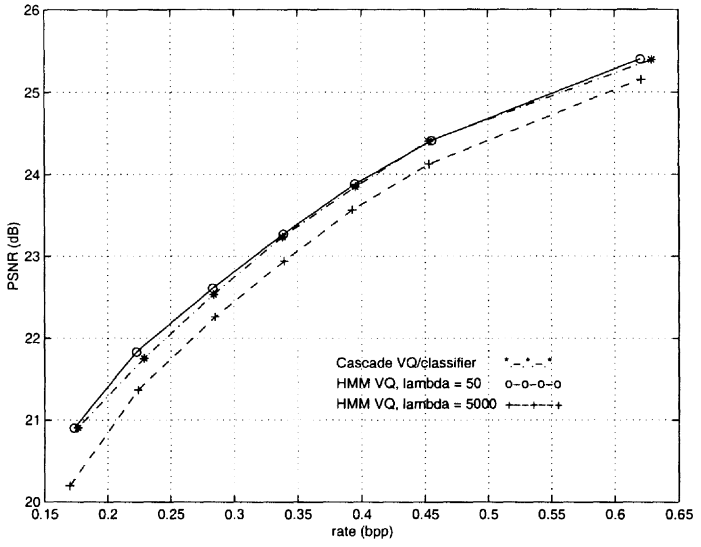
The images were divided into 4×4 blocks and DCT coefficients or averages of some of them were used as features. For details about the features, see Section 4.11.1. The vectors $x_{k,l}$ are 16 dimensional, with each component being the intensity of a pixel. The 2-D hidden Markov model used to describe the probability law of feature vectors has 5 states for the natural class and 9 states for the man-made class. Classification results reported in Section 4.11.2 are based on this model.

For $\lambda = 50.0, 5.0 \times 10^3$, compression and classification as they vary with bit rate are shown in Fig. 7.4. Performance is compared with a cascade system with a standard Lloyd vector quantizer followed by a Bayes classifier. At $\lambda = 50.0$, compared with the cascade system, HMM VQ achieves lower classification error rates consistently. Its compression performance also outperforms that of the Lloyd vector quantizer. At $\lambda = 5.0 \times 10^3$, by trading off PSNR by around 0.35 dB, HMM VQ keeps the classification error rate below 20% for all the rates, whereas the classification error rate of the cascade system ranges from 59% at 0.18 bpp to 24% at 0.63 bpp.

To see the effect of λ on compression and classification, we plot the performance versus $\log(\lambda)$ with base 10 in Fig. 7.5. From the definition of the distortion measure, we see that the larger λ is the higher weight is put on classification, which explains the trend of lower classification error rate and lower PSNR with larger λ in Fig. 7.5. In the extreme case, when

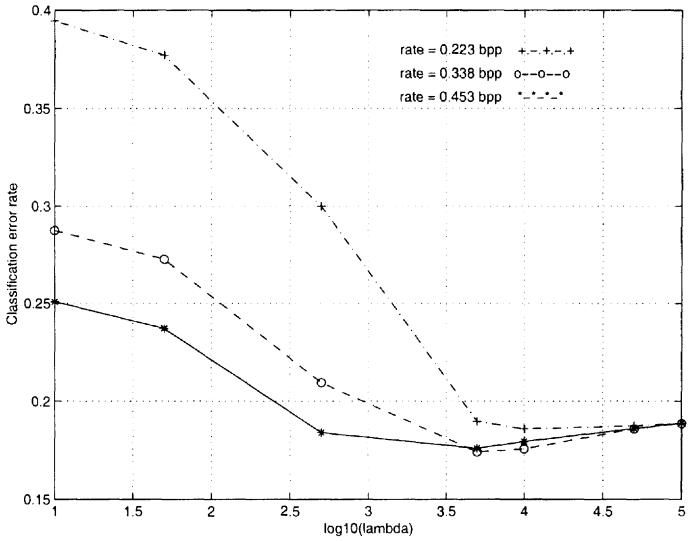


(a)

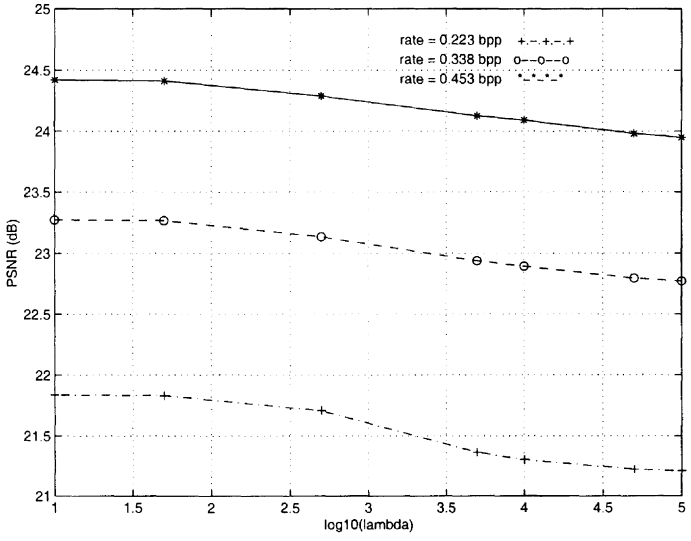


(b)

Figure 7.4. Compression and classification performance of HMM VQ for aerial images at $\lambda = 50.0, 5.0 \times 10^3$: (a) Classification error rate, (b) Compression performance (PSNR)



(a)



(b)

Figure 7.5. Effect of λ on compression and classification at rates 0.223, 0.338, and 0.453 bpp: (a) Classification error rate, (b) Compression Performance (PSNR)

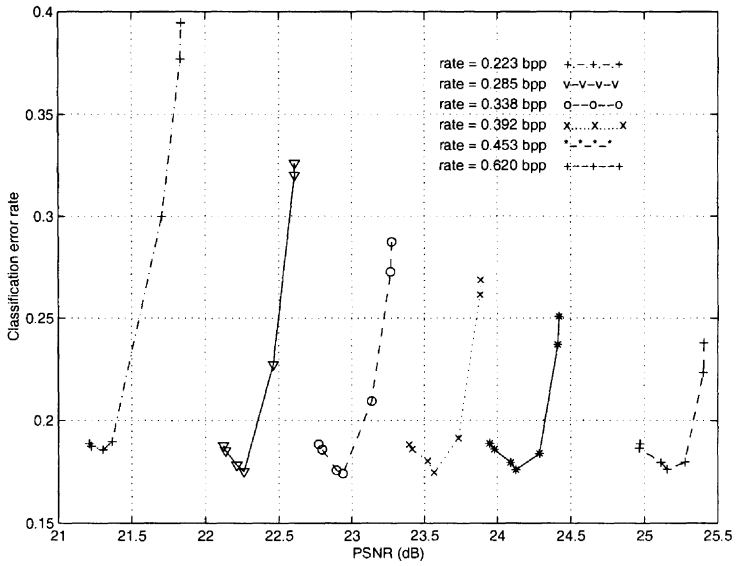


Figure 7.6. Tradeoff between compression and classification with HMM VQ. The λ's are in an increasing order with the increase of PSNR; and $\lambda = 10.0, 50.0, 5.0 \times 10^2, 5.0 \times 10^3, 1.0 \times 10^4, 5.0 \times 10^4, 1.0 \times 10^5$.

$\lambda = 0$, the algorithm is equivalent to a Lloyd vector quantizer followed by a Bayes classifier, and when $\lambda = \infty$, the algorithm is equivalent to a pure HMM classifier followed by a Lloyd vector quantizer. Note that the classification error rate, however, is not monotonic with λ. For 0.338 bpp and 0.453 bpp, the minimum classification error rates occur at $\lambda = 5.0 \times 10^3$. Assigning more weight on classification by larger λ, e.g., 1.0×10^4, leads to worse classification. This fact demonstrates again that tradeoff with compression may actually improve classification. By comparing with Table 4.1, we note that the classification error rate at bit rate 0.338 bpp and $\lambda = 5.0 \times 10^3$ is 17.42%, which is lower than the error rate 18.80% of a pure HMM classifier with the same model. In order to compare with Table 4.1, detailed classification results at 0.338 bpp and $\lambda = 5.0 \times 10^3$ are provided by Table 7.3.

To see the tradeoff between compression and classification more directly, we plot classification error rate versus compression performance in Fig. 7.6. The tradeoff is controlled by λ. At all the bit rates, changing λ causes PSNR to vary within a range about 0.5 dB. On the other hand, the classification error rate decreases significantly with the increase of λ especially when the bit rate is low.

7.6 PROGRESSIVE COMPRESSION AND CLASSIFICATION

To classify an image based on a multiresolution 2-D HMM, as done in Chapter 5, several resolutions of the image are obtained using wavelet transforms or other filtering techniques. Feature vectors at a particular resolution are then evaluated based only on the image at that resolution. Classification can be performed progressively by examining the lowest resolution image first, then successively adding back in high resolution information if the class of a block is marked as "mixed" in lower resolutions. Since the manner of the progressive classification is similar to that of progressive compression, we can perform progressive compression and classification jointly.

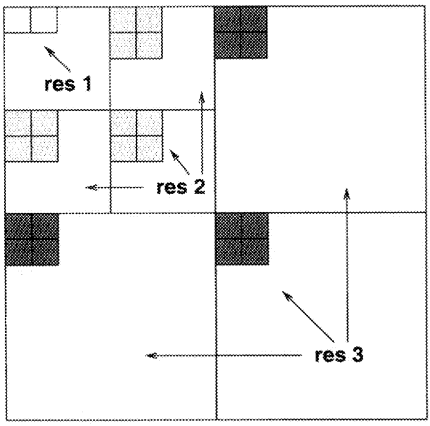


Figure 7.7. Vectors used at the 3 resolutions

A pilot study was done using the aerial images. First, a two level wavelet transform is applied to an image to decompose it into 3 resolutions. For compression, the vector dimension at resolution 1 to 3 is 2, 12, and 12 respectively. At resolution 1, although the block size for classification is 4×4, coefficients in an entire block are not grouped as one vector because the training data set is not sufficiently large even for moderate bit rates. The set of coefficients grouped into one vector at each resolution is shown in Fig. 7.7. As with the joint compression and classification based on single resolution 2-D HMMs, the distortion measure is defined as a weighted sum of compression distortion and the negative of the joint log likelihood of states and feature vectors across

all the resolutions. Each codeword at each resolution is mapped into one class by the classifier κ. An algorithm similar to HMM VQ is applied to iteratively improve the encoder $\tilde{\alpha}$, the decoder $\tilde{\beta}$ and the classifier κ. Results show that the progressive algorithm improves performance considerably when the bit rate is low, which is consistent with the fact that progressive subband coding usually outperforms the full search Lloyd VQ when the bit rate is low.

Chapter 8

CONCLUSIONS

Imagination is more important than knowledge.

—Albert Einstein

8.1 SUMMARY

We have presented several image classification algorithms and an algorithm for designing vector quantizers aimed at joint compression and classification.

In order to improve classification by context information, we represented images by 2-D hidden Markov models with the underlying state processes being 2nd order Markov meshes. Formulas were derived for estimating the models based on the EM algorithm developed for maximum likelihood estimation with incomplete data. Following the idea of the forward-backward algorithm for estimating 1-D HMMs efficiently, we developed a similar algorithm to reduce computation significantly. To further decrease computation, the maximum likelihood estimation was replaced by Viterbi training. The key issue for both training and testing became searching for states with maximum a posteriori probability, which can be done by the Viterbi algorithm. An approximation was made for the Viterbi algorithm to eliminate exponential computational complexity. The algorithm was applied to both aerial images and document images. The aerial images consisted of two classes: man-made and natural regions. The document images were classified into photograph and text. Results showed that this algorithm achieves better performance compared with algorithms classifying image blocks inde-

J. Li et al., *Image Segmentation and Compression Using Hidden Markov Models*

pendently, such as the learning vector quantization and a decision tree algorithm.

Following hints from the human visual system, we explored ways to segment images in a multiresolution manner. The human visual system can classify images accurately with high speed. Questioning why, we made two observations: the human visual system makes decision with both global and local information; and the visual system distributes effort unevenly by looking at more ambiguous regions at higher resolutions. The 2-D HMM is extended to multiresolution so that context information can be used more efficiently.

A multiresolution 2-D HMM represents images by feature vectors extracted from a number of resolutions. At any particular resolution, the feature vectors are statistically dependent through the underlying Markov mesh state process, similar to the assumptions of a 2-D HMM. Additionally, the feature vectors are statistically dependent across resolutions according to a hierarchical structure. The application to aerial images showed results superior to those obtained by the single resolution HMM. As the algorithm based on the multiresolution model searches for optimal states over all the resolutions jointly, it is computationally expensive. However, the hierarchical structure of the multiresolution model is naturally suited to progressive classification if the MAP rule is relaxed. Suboptimal fast algorithms were developed by searching for states in a layered fashion instead of the joint optimization. At much higher speed, the fast algorithms achieve results competitive with those of the single resolution algorithm.

Since the performance of the classification algorithms depends obviously on the validity of the 2-D HMM and MHMM, the model assumptions were tested by hypothesis testing techniques. First, we tested the assumption that feature vectors are Gaussian distributed given states. Normal probability plots were drawn to show how accurate the Gaussian assumption is. Second, we tested the Markovian property of states. A permutation χ^2 test was used to test the conditional independence of states. The results showed that the bias of the Markovian property of the 2-D HMM is mainly due to assuming the conditional independence of a state and its neighboring state at the left upper corner given the left and above neighboring states.

In the last part of the book, the issue of designing vector quantizers for joint compression and classification was considered. Although vector quantization (VQ) was developed originally for data compression, its underlying mathematical meaning is much broader than compression.

VQ addresses a fundamental problem, that is, how to select representative points of a set so that properties of the set are well retained. The goodness of the representative points is measured by a "loss" function. Different applications raise different definitions of "loss." For most data compression systems, in particular, the "loss" is the average mean squared error with respect to a certain probability measure on the set. By assigning a class to each quantization cell, VQ is naturally suited to classification. Actually, the well-known k-means clustering algorithm has essentially the same goal as a Lloyd vector quantizer. VQ for combined compression and classification thus can be regarded as a pure classification algorithm, being intermediate among different data clustering approaches. The combined algorithm has the potential to outperform existing classification algorithms by taking advantage of several approaches. This thought is supported by the study of the Bayes vector quantization (BVQ) algorithm. BVQ demonstrates that compression and classification do not always compete for performance; instead they assist each other in certain circumstances.

We presented a new algorithm for designing joint compression and classification vector quantizers based on 2-D HMMs. The new algorithm defines the penalty for misclassification as the negative of the maximum joint log likelihood of states and feature vectors under a 2-D HMM. The algorithm for pure classification with 2-D HMMs was extended easily to designing such quantizers. The algorithm was applied to aerial images. As with BVQ, results showed that assigning a small weight to compression may improve classification, and vice versa.

8.2 FUTURE WORK

One approach toward improving 2-D HMM classification algorithms is to develop better estimation methods. To reduce computational complexity, we started from the analytic formulas and proceeded through making approximations step by step. Other approaches might be investigated. For example, deterministic relaxation, a locally optimal algorithm described in Section 4.3, sets initial states for an image and optimizes the states by making local changes iteratively. Its result depends critically on the initial states. Future work might cascade our algorithm and the deterministic relaxation algorithm for improvement.

In terms of constructing better models, results obtained by testing models suggest extending the underlying state process from the 2nd order Markov mesh to 3rd order. The consequence of assuming the higher order Markov mesh is a great increase in the number of transition prob-

abilities. With the 3rd order Markov mesh, transition probabilities are indexed by four states, of which three are conditional states. Good estimation of the model demands the reduction of parameters. One compromise solution is to add restrictions on the parameters. For example, a symmetry assumption on the horizontal and vertical directions decreases the number of transition probabilities by half. It would be interesting to try the new model assumptions and make comparisons with results obtained.

It is worth applying the algorithm based on the multiresolution or single resolution 2-D HMM to other classification problems, such as texture segmentation and character recognition. Since image modeling is a means of describing the statistical behavior of images efficiently, it is applicable to many different problems. Although the 2-D HMMs were developed initially for image segmentation, it may prove equally useful for other image processing tasks, such as restoration and compression.

To extend the study of joint compression and classification, it is of interest to develop joint algorithms that align well with standard or state of the art compression methods. Such algorithms are potentially useful for applications requiring explicit information about content from compressed formats of images. Although VQ is not used directly by most current image compression systems, principles learned from designing a VQ for the joint purpose may be helpful for creating more practical schemes.

References

[1] K. Abend, T. J. Harley, and L. N. Kanal, "Classification of binary random patterns," *IEEE Trans. Inform. Theory*, vol. IT-11, no. 4, pp. 538-544, Oct. 1965.

[2] S. Adlersberg and V. Cuperman, "Transform domain vector quantization for speech signals," *Proc. Int. Conf. Acoust., Speech and Signal Processing*, vol. 4, pp. 1938-1941, Dallas, TX, April 1987.

[3] K. Aizawa, H. Harashimia, and H. Miyakawa, "Adaptive vector quantization of picture signals in discrete cosine transform domain," *Electronics and Communications in Japan, Part I*, vol. 70, no. 5, pp. 105-114, 1987.

[4] B. S. Atal, V. Cuperman, and A. Gersho, editors, *Advances in Speech Coding*, Kluwer Academic Publishers, July 1991.

[5] J. K. Baker, "The dragon system—an overview," *Proc. Int. Conf. Acoust., Speech and Signal Processing*, vol. ASSP-23, no. 1, pp. 24-29, Feb. 1975.

[6] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2743-60, Oct. 1998.

[7] M. Basseville, A. Benveniste, K. C. Chou, S. A. Golden, R. Nikoukhah, and A. S. Willsky, "Modeling and estimation of multiresolution stochastic processes," *IEEE Trans. Inform. Theory*, vol. 38, no. 2, pp. 766-784, March 1992.

[8] L. E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of finite state Markov chains," *Inequalities III*, pp. 1-8, Academic Press, New York, 1972.

[9] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," *Bulletin of American Mathematical Statistics*, vol. 37, pp. 360-363, 1967.

[10] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1554-1563, 1966.

[11] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164-171, 1970.

[12] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[13] J. Besag, "Spatial interaction and the statistical analysis of lattice systems (with discussion)," *Journal Royal Statistics Society*, series B, vol. 34, pp. 75-83, 1972.

[14] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics*, Prentice Hall, Englewood Cliffs, NJ, 1977.

[15] T. Binford, T. Levitt, and W. Mann, "Bayesian inference in model-based machine vision," *Uncertainty in Artificial Intelligence*, J. F. Lemmer and L. M. Kanal, editors, Elsevier Science, 1988.

[16] R. K. Blashfield and M. S. Aldenderfer, "The literature on cluster analysis," *Multivariate Behavioral Research*, vol. 13, pp. 271-295, 1978.

[17] C. A. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. Image Processing*, vol. 3, no. 2, pp. 162-177, March 1994.

[18] A. W. Bowman, "A note on consistency of the kernel method for the analysis of categorical data," *Biometrika*, vol. 67, no. 3, pp. 682-684, 1980.

[19] J. M. Boyett, "Random RxC tables with given row and column totals," *Applied Statistics*, vol. 28, pp. 329-332, 1979.

[20] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees,* Chapman & Hall, 1984.

[21] A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoust., Speech and Signal Processing*, vol. ASSP-28, pp. 562-574, Oct. 1980.

[22] N. Chaddha, K. Perlmutter, and R. M. Gray, "Joint image classification and compression using hierarchical table-lookup vector quantization," *Proc. Data Compression Conference (DCC)*, pp. 23-32, Snowbird, UT, March 1996.

[23] N. Chaddha, R. Sharma, A. Agrawal, and A. Gupta, "Text segmentation in mixed-mode images," *Proc. Asilomar Conf. Signals, Systems and Computers*, vol. 2, pp. 1356-1361, Nov. 1994.

[24] N. Chaddha, M. Vishwanath, and P. A. Chou, "Hierarchical vector quantization of perceptually weighted block transforms," *Proc. Data Compression Conference*, pp. 3-12, Snowbird, UT, March 1995.

[25] D. L. Chaffee and J. K. Omura, "A very low rate voice compression system," *Proc. IEEE Int. Symp. Inform. Theory*, Oct. 1974.

[26] P.-C. Chang, R. M. Gray, and J. May, "Fourier transform quantization for speech coding," *IEEE Trans. Commun.*, vol. COM-35, no. 10, pp. 1059-1068, Oct. 1987.

[27] P.-C. Chang, J. May, and R. M. Gray, "Hierarchical vector quantization with table-lookup encoders," *Proc. Int. Conf. Commun.*, pp. 1452-1455, Chicago, IL, June 1985.

[28] H. Choi and R. G. Baraniuk, "Image segmentation using wavelet-domain classification," *Proc. SPIE Technical Conference on Mathematical Modeling, Bayesian Estimation, and Inverse Problems*, pp. 306-320, July 1999.

[29] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust., Speech and Signal Processing*, vol. 37, pp. 31-42, Jan. 1989.

[30] P. A. Chou, T. Lookabaugh, and R. M. Gray, "Optimal pruning with applications to tree-structured source coding and modeling," *IEEE Trans. Inform. Theory*, vol. 35, no. 2, pp. 299-315, March 1989.

[31] R. Cole, L. Hirschman, L. Atlas, M. Beckman, et al., "The challenge of spoken language systems: research directions for the nineties," *IEEE Trans. Speech and Audio Processing*, vol. 3, pp. 1-21, 1063-6676, Jan. 1995.

[32] G. F. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, vol. 42, pp. 393-405, 1990.

[33] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 21-27, Jan. 1967.

[34] T. M. Cover and R. C. King, "A convergent gambling estimate of the entropy of English," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 413-421, July 1978.

[35] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, pp. 25-39, Jan. 1983.

[36] M. S. Crouse, R. D. Nowak, and R. G. Baraniuk, "Wavelet-based statistical signal processing using hidden Markov models," *IEEE Trans. Signal Processing*, vol. 46, no. 4, pp. 886-902, April 1998.

[37] I. Daubechies, *Ten Lectures on Wavelets*, Capital City Press, 1992.

[38] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal Royal Statistics Society*, vol. 39, no. 1, pp. 1-21, 1977.

[39] P. A. Devijver, "Probabilistic labeling in a hidden second order Markov mesh," *Pattern Recognition in Practice II*, pp. 113-123, Amsterdam, Holland, 1985.

[40] P. A. Devijver, "Segmentation of binary images using third order Markov mesh image models," *Proc. 8th Int. Conf. Pattern Recognition*, pp. 259-261, Paris, Oct. 1986.

[41] P. A. Devijver, "Modeling of digital images using hidden Markov mesh random fields," *Signal Processing IV: Theories and Applications (Proc. EUSIPCO-88)*, pp. 23-28, 1988.

[42] P. A. Devijver, "Real-time modeling of image sequences based on hidden Markov mesh random field models," *Proc. 10th Int. Conf. Pattern Recognition*, vol. 2, pp. 194-199, Los Alamitos, California, 1990.

[43] P. A. Devijver and M. M. Dekesel, "Experiments with an adaptive hidden Markov mesh image model," *Philips Journal of Research*, vol. 43, no. 3/4, pp. 375-392, 1988.

[44] A. P. Dhawan, Y. Chitre, C. Kaiser-Bonasso, and M. Moskowitz, "Analysis of mammographic microcalcifications using gray-level image structure features," *IEEE Trans. Medical Imaging*, vol. 15, no. 3, pp. 246-259, June 1996.

[45] R. L. Dobrushin, "The description of a random field by means of conditional probabilities and conditions of its regularity," *Theory Prob. Appl.*, vol. 13, pp. 197-224, 1968.

[46] A. M. El Sherbini, "A new feature vector for classification of binary images," *Proc. 11th IASTED Int. Conf. Applied Informatics*, pp. 330-333, 1995.

[47] B. S. Everitt, "A finite mixture model for the clustering of mixed-mode data," *Statist. Probab. Lett.*, vol. 6, no. 5, pp. 305-309, 1988.

[48] K.T. Fang and Y. Wang, *Number-theoretic Methods in Statistics*, Chapman & Hall, 1994.

[49] K.T. Fang, Y. Wang, and P. M. Bentler, "Applications of number-theoretic method in multivariate statistics," *Int. Symp. Multivariate Analysis and Its Applications*, Hong Kong, 1992.

[50] K.T. Fang and C.Y. Wu, "The extreme value problem of some probability function," *Acta Math. Appl. Sinica*, vol. 2, pp. 132-148, 1979.

[51] R. A. Fisher, *The Design of Experiments*, Edinburgh, Oliver and Boyd, 1953.

[52] J. L. Fisher, S. C. Hinds, and D. P. D'Amato, "A rule-based system for document image segmentation," *Proc. IEEE 10th Int. Conf. Pattern Recognition*, pp. 567-572, Atlantic City, NJ, June 1990.

[53] E. Fix and J. L. Hodges, Jr., "Discriminatory analysis—nonparametric discrimination: consistency properties," USAF School of Aviation Medicine, Randolf Field, TX, Project 21-49-004, Rep. 4, pp. 261-279, 1951.

[54] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 910-918, Nov. 1988.

[55] C. H. Fosgate, H. Krim, W. W. Irving, W. C. Karl, and A. S. Willsky, "Multiscale segmentation and anomaly enhancement of SAR imagery," *IEEE Trans. Image Processing*, vol. 6, no. 1, pp. 7-20, Jan. 1997.

[56] W. T. Freeman and E. C. Pasztor, "Learning low-level vision," *Proc. 7th Int. Conf. Computer Vision*, Corfu, Greece, Sept. 1999.

[57] R. G. Gallager, *Information Theory and Reliable Communication*, John Wiley & Sons, Inc., 1968.

[58] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-6, no. 6, pp. 721-741, Nov. 1984.

[59] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inform. Theory*, vol. IT-25, no. 4, pp. 373-380, July 1979.

[60] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[61] A. Gersho and Y. Shoham, "Hierarchical vector quantization of speech with dynamic codebook allocation," *Proc. Int. Conf. on Acoust., Speech and Signal Processing*, March 1984.

[62] C. Goutis, "Nonparametric estimation of a mixing density via the kernel method," *Journal of the American Statistical Association*, vol. 92, no. 440, pp. 1445-1450, 1997.

[63] R. M. Gray, K. O. Perlmutter, and R. A. Olshen, "Quantization, classification, and density estimation for Kohonen's Gaussian mixture," *Proc. Data Compression Conference*, pp. 63-72, Snowbird, UT, March 1998.

[64] A. Habibi and P. A. Wintz, "Image coding by linear transformation and block quantization," *IEEE Trans. Communication Technology*, vol. COM-19, no. 1, pp. 50-62, Feb. 1971.

[65] F. R. Hansen and H. Elliott, "Image segmentation using simple Markov field models," *Computer Graphics and Image Processing*, vol. 20, pp. 101-132, 1982.

[66] J. A. Hartigan, *Clustering Algorithms*, John Wiley & Sons, New York, 1975.

[67] J. A. Hartigan and M. A. Wong, "Algorithm AS136: a k-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100-108, 1979.

[68] D. Heckerman, "A tutorial on learning with Bayesian Networks," *Microsoft Research Technical Report MSR-TR-95-06*, Nov. 1996.

[69] E. E. Hilbert, "Cluster compression algorithm: a joint clustering/data compression concept," Jet Propulsion Lab., Pasadena, CA, Publication 77-43, Dec. 1977.

[70] X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov Models for Speech Recognition*, Edinburgh University Press, 1990.

[71] F. Jelinek and J. B. Anderson, "Instrumentable tree encoding of information sources," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 118-119, Jan. 1971.

[72] L. N. Kanal, "Markov mesh models," *Image Modeling*, pp. 239-243, New York: Academic, 1980.

[73] R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications*, American Mathematical Society, 1980.

[74] D. Knill and W. Richards, editors, *Perception as Bayesian Inference*, Cambridge University Press, 1996.

[75] T. Kohonen, "An introduction to neural computing," *Neural Networks*, no. 1, pp. 3-16, 1988.

[76] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1989.

[77] T. Kohonen, G. Barna, and R. Chrisley, "Statistical pattern recognition with Neural Networks: benchmarking studies," *IEEE Int. Conf. Neural Networks*, pp. I-61-68, July 1988.

[78] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola, "LVQ_PAK: The learning vector quantization program package (version 3.1)," Technical Report, Helsinki University of Technology, Laboratory of Computer and Information Science, Finland, April, 1995. Available via anonymous ftp to cochlea.hut.fi.

[79] S. R. Kulkarni, G. Lugosi, and S. S. Venkatesh, "Learning pattern classification—a survey," *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2178-2206, Oct. 1998.

[80] S. S. Kuo and O. E. Agazzi,"Machine vision for keyword spotting using pseudo 2D hidden Markov models," *Proc. Int. Conf. Acoust., Speech and Signal Processing*, vol. 5, pp. 81-84, 1993.

[81] M. S. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, MIT Press, Cambridge, MA, 1991.

[82] L. C. Lazzeroni and K. Lange, "Markov chains for Monte Carlo tests of genetic equilibrium in multidimensional contingency tables," *Annals of Statistics*, vol. 25, pp. 138-168, 1997.

[83] E. Levin and R. Pieraccini, "Dynamic planar warping for optical character recognition," *Int. Conf. Acoust., Speech and Signal Processing*, vol. 3, pp. 149-152, San Francisco, CA, March 1992.

[84] J. Li and R. M. Gray, "Context based multiscale classification of images," *Proc. Int. Conf. Image Processing*, Chicago, Oct. 1998.

[85] J. Li and R. M. Gray, "Text and picture segmentation by the distribution analysis of wavelet coefficients," *Proc. Int. Conf. Image Processing*, Chicago, Oct. 1998.

[86] J. Li, J. Z. Wang, R. M. Gray, and G. Wiederhold, "Segmentation for images with low depth of field: a multiresolution context dependent approach," *Int. Conf. Image Analysis and Processing*, Venice, Italy, Sep. 1999.

[87] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, Jan. 1980.

[88] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 127-135, March 1982.

[89] G. Loum, P. Provent, J. Lemoine, and E. Petit, "A new method for texture classification based on wavelet transform," *Proc. 3rd Int. Symp. Time-Frequency and Time-Scale Analysis,* pp. 29-32, June 1996.

[90] J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. 73, pp. 1551-1587, Nov. 1985.

[91] S. G. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, no. 7, pp. 674-693, July 1989.

[92] A. A. Markov, "An example of statistical investigation in the text of 'Eugene Onyegin' illustrating coupling of 'tests' in chains," *Proc. Acad. Sci. St.*, Petersburg, VI Series 7, pp. 153, 1913.

[93] G. J. McLachlan and K. E. Basford, *Mixture Models: Inference and Applications to Clustering*, Marcel Dekker Inc., New York, 1988.

[94] C. R. Mehta and N. R. Patel, "A network algorithm for performing Fisher's exact test in $r \times c$ contingency tables," *Journal of the American Statistical Association*, June 1983.

[95] Y. Meyer, *Wavelets Algorithms and Applications*, SIAM, Philadelphia, 1993.

[96] C. L. Nash, K. O. Perlmutter, and R. M. Gray, "Evaluation of Bayes risk weighted vector quantization with posterior estimation in the detection of lesions in digitized mammograms," *Proc. 28th Asilomar Conf. Circuits, Systems and Computers*, vol. 1, pp. 716-720, Pacific Grove, CA, Oct. 1994.

[97] N. J. Nilsson, *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, McGraw-Hill, NY, 1965.

[98] R. D. Nowak, "Multiscale hidden Markov models for Bayesian image analysis," *Bayesian Inference in Wavelet Based Models*, pp. 243-266, Springer-Verlag, 1999.

[99] K. L. Oehler, "Image compression and classification using vector quantization," *Ph.D thesis*, Stanford University, 1993.

[100] K. L. Oehler and R. M. Gray, "Combining image classification and image compression using vector quantization," *Proc. Data Compression Conference*, pp. 2-11, Snowbird, UT, March 1993.

[101] K. L. Oehler and R. M. Gray, "Combining image compression and classification using vector quantization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 461-473, May 1995.

[102] S. Ohuchi, K. Imao, and W. Yamada, "Segmentation method for documents containing text/picture (screened halftone, continuous tone)," *Transactions of the Institute of Electronics, Information and Communication Engineers D-II*, vol. J75D-II, no. 1, pp. 39-47, Jan. 1992.

[103] M. Park and D. J. Miller, "Image decoding over noisy channels using minimum mean-squared estimation and a Markov mesh," *Proc. Int. Conf. Image Processing*, vol. 3, pp. 594-597, Santa Barbara, CA, Oct. 1997.

[104] D. B. Paul, "Speech recognition using hidden Markov models," *The Lincoln Laboratory Journal*, vol. 3, no. 1, pp. 41-62, 1990.

[105] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, 1988.

[106] K. O. Perlmutter, "Compression and classification of images using vector quantization and decision trees," *Ph.D thesis*, Stanford University, 1995.

[107] K. O. Perlmutter, N. Chaddha, J. B. Buckheit, R. M. Gray, and R. A. Olshen, "Text segmentation in mixed-mode images using classification trees and transform tree-structured vector quantization," *Proc. Int. Conf. Acoust., Speech and Signal Processing*, vol. 4, pp. 2231-2234, Atlanta, GA, May 1996.

[108] K. O. Perlmutter, R. M. Gray, K. L. Oehler, and R. A. Olshen, "Bayes risk weighted tree-structured vector quantization with posterior estimation," *Proc. Data Compression Conference*, pp. 274-283, Snowbird, UT, March 1994.

[109] K. O. Perlmutter, R. M. Gray, R. A. Olshen, and S. M. Perlmutter, "Bayes risk weighted vector quantization with CART estimated

posteriors," *Proc. Int. Conf. Acoust., Speech and Signal Processing*, vol. 4, pp. 2435-2438, May 1995.

[110] K. O. Perlmutter, C. L. Nash, and R. M. Gray, "A comparison of Bayes risk weighted vector quantization with posterior estimation with other VQ-based classifiers," *Proc. IEEE Int. Conf. Image Processing*, vol. 2, pp. 217-221, Austin, TX, Nov. 1994.

[111] K. O. Perlmutter, S. M. Perlmutter, R. M. Gray, R. A. Olshen, and K. L. Oehler, "Bayes risk weighted vector quantization with posterior estimation for image compression and classification," *IEEE Trans. Image Processing*, vol. 5, no. 2, pp. 347-360, Feb. 1996.

[112] D. K. Pickard, "A curious binary lattice process," *J. Appl. Prob.*, vol. 14, pp. 717-731, 1977.

[113] D. Pollard, "Strong consistency of k-means clustering," *Annals of Statistics*, vol. 9, pp. 135-140, 1981.

[114] L. Rabiner and B. H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, NJ, 1993.

[115] B. Ramamurthi and A. Gersho, "Classified vector quantization of images," *IEEE Trans. Commun.*, vol. COM-34, pp. 1105-1115, Nov. 1986.

[116] O. Rioul and M. Vetterli, "Wavelets and signal processing," *IEEE Signal Processing Magazine,* vol. 8, no. 4, pp. 14-38, Oct. 1991.

[117] E. A. Riskin and R. M. Gray, "A greedy tree growing algorithm for the design of variable rate vector quantizers," *IEEE Trans. Signal Process.*, Nov. 1991.

[118] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243-250, June 1996.

[119] J. Sauvola and M. Pietikainen, "Page segmentation and classification using fast feature extraction and connectivity analysis," *Proc. 3rd Int. Conf. Document Analysis and Recognition*, Montreal, Que., Canada, Aug. 1995.

[120] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423, July 1948.

[121] C. E. Shannon, "Prediction and entropy of printed English," *Bell System Technical Journal*, pp. 50-64, Jan. 1951.

[122] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE Nat. Conv. Rec.*, pp. IV-142-163, March 1959.

[123] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3445-3462, Dec. 1993.

[124] F. Y. Shih and S. Chen, "Adaptive document block segmentation and classification," *IEEE Trans. Systems, Man and Cybernetics*, vol. 26, no. 5, pp. 797-802, Oct. 1996.

[125] G. W. Snedecor and W. G. Cochran, *Statistical Methods,* Iowa State University Press, Ames, Iowa, 1989.

[126] M. Stone, "Cross-validation: a review," *Math. Operationforsch. Statist. Ser. Statist.*, no. 9, pp. 127-139, 1978.

[127] M. Unser, "Texture classification and segmentation using wavelet frames," *IEEE Trans. Image Processing,* vol. 4, no. 11, pp. 1549-1560, Nov. 1995.

[128] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*, Chapter 7, Prentice-Hall Inc., 1995.

[129] M. Vishwanath and P. A. Chou, "An efficient algorithm for hierarchical compression of video," *Proc. Int. Conf. Image Processing*, vol. 3, pp. 275-279, Austin, TX, Nov. 1994.

[130] A. J. Viterbi and J. K. Omura, "Trellis encoding of memoryless discrete-time sources with a fidelity criterion," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 325-332, May 1974.

[131] F. M. Wahl, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Vision, Graphics, Image Processing*, vol. 20, pp. 375-390, 1982.

[132] Y. Wang and K.T. Fang, "A note on uniform distribution and experimental design," *Kexue Tongba*, vol. 26, pp. 485-489, 1981.

[133] D. Wang and S. N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics, Image Processing*, vol. 47, pp. 327-352, 1989.

[134] J. B. Weaver, D. M. Healy, H. Nagy, S. P. Poplack, J. Lu, T. Sauerland, and D. Langdon, "Classification of masses in digitized mammograms with features in the wavelet transform domain," *Proc. SPIE - Wavelet Applications,* vol. 2242, pp. 704-710, April 1994.

[135] P. S. Williams and M. D. Alder, "Generic texture analysis applied to newspaper segmentation," *Proc. Int. Conf. Neural Networks*, vol. 3, pp. 1664-1669, Washington, DC, June 1996.

[136] K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document analysis system," *IBM J. Res. Develp.*, vol. 6, pp. 642-656, Nov. 1982.

[137] C. F. J. Wu, "On the convergence properties of the EM algorithm," *Annals of Statistics*, vol. 11, no. 1, pp. 95-103, 1983.

[138] C. C. Yen and S. S. Kuo, "Degraded documents recognition using pseudo 2d hidden Markov models in gray-scale images," *Proc. SPIE*, vol. 2277, pp. 180-191, 1994.

[139] S. Young, J. Jansen, J. Odell, D. Ollason, and P. Woodland, *HTK - Hidden Markov Model Toolkit*, Cambridge University, 1995.

[140] A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek, "CREW: compression with reversible embedded wavelets," *Proc. Data Compression Conference*, Snowbird, UT, March 1995.

[141] J. Zhang, J. W. Modestino, and D. A. Langan, "Maximum-likelihood parameter estimation for unsupervised stochastic model-based image segmentation," *IEEE Trans. Image Processing*, vol. 3, no. 4, pp. 404-420, 1994.

Index