IMAGE CLASSIFICATION AND COMPRESSION BASED ON A TWO DIMENSIONAL MULTIRESOLUTION HIDDEN MARKOV MODEL

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Jia Li
June 1999

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Robert M. Gray
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Thomas M. Cover

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Richard A. Olshen

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

G. Leonard Tyler

Approved for the University Committee on Graduate Studies:

To my parents

Abstract

Image classification is a process for dividing an image into its constituent parts. With applications ranging widely from remote sensing to medical image analysis, classification is often a key step for extracting information from images.

For block-based classification, an image is divided into blocks and a feature vector is formed for each block by grouping statistics extracted from the block. Conventional block-based classification algorithms decide the class of a block by examining only the feature vector of this block and ignoring context information. In order to improve classification by context, a new algorithm models images by two dimensional hidden Markov models (HMMs). The HMM considers feature vectors statistically dependent through an underlying state process assumed to be a Markov mesh. The model is estimated by the maximum likelihood (ML) criterion. To classify an image, optimal classes are searched jointly for all the blocks. The 2-D HMM is extended to multiresolution to benefit from more global context and to enable fast progressive classification.

The EM algorithm, a well-known algorithm for ML estimation based on incomplete data, yields converging analytic formulas that improve the estimation of the 2-D HMM iteratively. The computational complexity of using these formulas directly is exponential in the number of image blocks. Successive approximations are made to reduce this computational complexity to quadratic. Applications to documentary and aerial images show that the 2-D HMM algorithm outperforms k-means, the learning vector quantization, and a decision tree algorithm.

Joint compression and classification systems are important in multimedia communication for extracting explicit information about content from compressed image

formats. To design vector quantizers for simultaneous good compression and classification, a new distortion measure is defined as a weighted sum of compression distortion and the penalty of misclassification, the latter being evaluated according to an estimated 2-D HMM. To minimize this distortion, a Lloyd-like algorithm is used to optimize iteratively the three components of a vector quantizer: the encoder, the decoder, and the classifier. Vector quantizers designed by the algorithm for both synthetic and aerial images outperform the Bayes vector quantizers.

Acknowledgments

I would like to thank my principal advisor, Professor Robert M. Gray, for his great guidance, support, and encouragement. He has provided me not only the technical knowledge but also a rigorous attitude towards research, for which I am very grateful. I also thank him for paying special attention to my English and giving me wonderful advice when I started my Ph.D. I feel fortunate to have taken his advice. I would like to thank my associate advisor, Professor Thomas M. Cover, for his guidance and support. His marvelous lectures on information theory made electrical engineering far more interesting. I would also express my gratitude to my associate advisor, Professor Richard A. Olshen, for having many discussions with me and referring me to many papers. I appreciate his tremendous caring about students. I feel fortunate to have had him as my advisor. My gratitude also goes to Professor G. Leonard Tyler for serving on both my orals committee and my reading committee.

I would like to thank the wonderful members of our research group for their help and friendship. Group members: Sharon Perlmutter, Keren Perlmutter, Cheryl Nash, Brad Betts, Anu Aiyer, Sanjeev Mehrotra, Amir Najmi, Xin Tong, Ken Lin, John Young, Remco Teunen, Christine Pepin, Philippe Raffy, and Thomas Wiegand have made a supportive research team. Special thanks go to Sharon Perlmutter, Keren Perlmutter, and Cheryl Nash for helping me during the initial part of my Ph.D; Amir Najmi for many useful discussions; Anu Aiyer for giving me guidance on writing English; and Brad Betts for teaching me about computer software. Without their help, my Ph.D work could not be done. I would also like to thank Charlotte Coe for assisting so many things joyfully and efficiently.

I am fortunate to have a very loving and supporting family. I am grateful to

my grandparents for their love, support, and understanding. I thank my sister and brother-in-law, Cui-wei Li and Chang-feng Tai, for their support and encouragement. I would like to thank my husband, James Ze Wang, whose love and talents make my life much more enjoyable. Besides providing me a warm, caring, and supportive home, he has shared many interests such as traveling and photography with me, for which I am very grateful. I also thank him for interesting technical discussions and helping me in using computers. At last, I would like to thank my parents, Shao-han Zhou and Ping-sen Li, for many things—I thank them for their love, encouragement, and understanding. They made tremendous effort to create good education conditions for me in an uneasy environment. As my mentors, they have set life models by being strong and steady, honest, and optimistic. They have also taught me respect for nature. I will always be grateful for that. I dedicate this thesis to my parents.

This work was supported by research grants from the National Science Foundation and a gift fund from Hewlett-Packard, Inc.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

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, etc. 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. First, images are digitized so that computers can read them. Then, image processing algorithms are applied to instruct computers to handle the images automatically. Among the large variety of image processing techniques, in this thesis, I focus on classification and compression. Classification is a critical step in image understanding. To extract information from images, classification is often performed first to separate 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 certain measurement can be done automatically. The second technique: data compression, is developed 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.

In Chapter 2, background for statistical classification is provided. After reviewing important classification techniques, I 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, I 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, I introduce 1-D HMMs, which are used widely in speech recognition. I 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 and backward algorithm, an efficient recursive algorithm for estimating 1-D HMMs. However, since the forward and 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 documentary images.

In order to obtain classification based upon a combination of global and local information, the 2-D HMM is extended to multiresolution in Chapter 5. The extension allows an image to be represented by features in several resolutions, corresponding to global to local context information. Furthermore, the multiresolution model provides a hierarchical structure for progressive classification, which speeds up 2-D HMM classification significantly. Comparisons are made between the multiresolution model and the single resolution model based upon experiments on aerial images.

A natural question to ask is the accuracy of 2-D HMMs. Obviously, the potential of algorithms with 2-D HMMs depends on the validity of those models. Although good results obtained in Chapter 4 and 5 intuitively justify the models, I examine the validity of the HMMs for images formally 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 systems. 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. My 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 thesis and provides directions to future research.

Chapter 2

Statistical Classification

Classification is the grouping of similar objects. The word clustering is almost synonymous with classification. It is usually used as a term for formal, planned, or purposeful classification. These two terms are used interchangeably here, since our interest is restricted to scientific classification. Clustering techniques were first developed in biological taxonomy. Formal classifications of animals and plants date back to Aristotle. A tree structure, which is now used as a standard classification structure, was applied to name objects consistently. For example, man belongs to the primates, the primates belong to the mammals, and the mammals belong to the animals. Man has an ancestor in common with a tiger, a dolphin, a swallow, and a beetle. Modern classification techniques owe their development largely to computers. Classification is now widely used in problems such as medical diagnosis, 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, we form a class prediction rule 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. I will focus on supervised classification.

2.1 Bayes Rule

To formulate the classification problem, 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 the classification rule $\kappa(x)$ 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 rule, 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 rule is

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

which is determined by majority vote.

2.2 Algorithms

Statistical classification is a rich research field with many algorithms developed and applied to various problems [57, 68, 46, 17, 66]. I review briefly some techniques in this section.

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

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

Good estimation of $P(Y = j \mid X = x)$ 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^{TM} [17], developed by Breiman, Friedman, Olshen and Stone. The basic idea is to partition a feature space by 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. Purity 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 found. 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 [27] 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 [15, 54] and the mixture model method [81, 40] 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 [57, 58, 14, 99] and learning vector quantization (LVQ) [64, 65, 66] are prototype methods. Other classification algorithms include partition-based classifiers: maximum likelihood (ML), linear discriminant, ect.

For the k-means algorithm, suppose observations are $\{x_i : i = 1, ..., L\}$. 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 $\sum_{i=1}^{L} \min_{1 \le j \le k}(x_i - \hat{x}_j)^2$ is minimized. This algorithm is closely related to vector quantization, as I shall discuss in the next chapter. Kohonen *et al.* modified the k-means algorithm and proposed a variety of LVQ algorithms. To appreciate the basic ideas of LVQ, I introduce LVQ1. 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

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

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 [67].

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, we form a window centered around it and compute statistics of pixel intensities in the neighborhood. 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 [87, 93, 96] for examples.

In the above approach, feature vectors are considered independent samples of a distribution, from which performance often suffers because significant information as to context is ignored. More sophisticated methods incorporate the statistical dependence among feature vectors into 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 [63, 50], which are described in the next section.

2.3 Markov Random Fields

Let us first discuss the definition of Markov random fields for images. An image is regarded as a random matrix X. Let $\mathbb{N}_m = \{(i,j) : 0 \le i,j \le m-1\}$ 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

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

For images, common neighborhood systems are homogeneous with the form

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

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

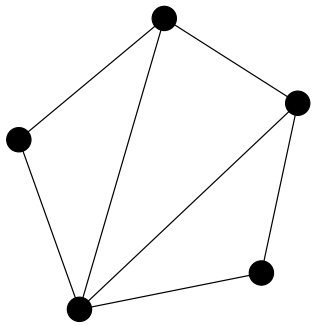


Figure 2.1: An example graph

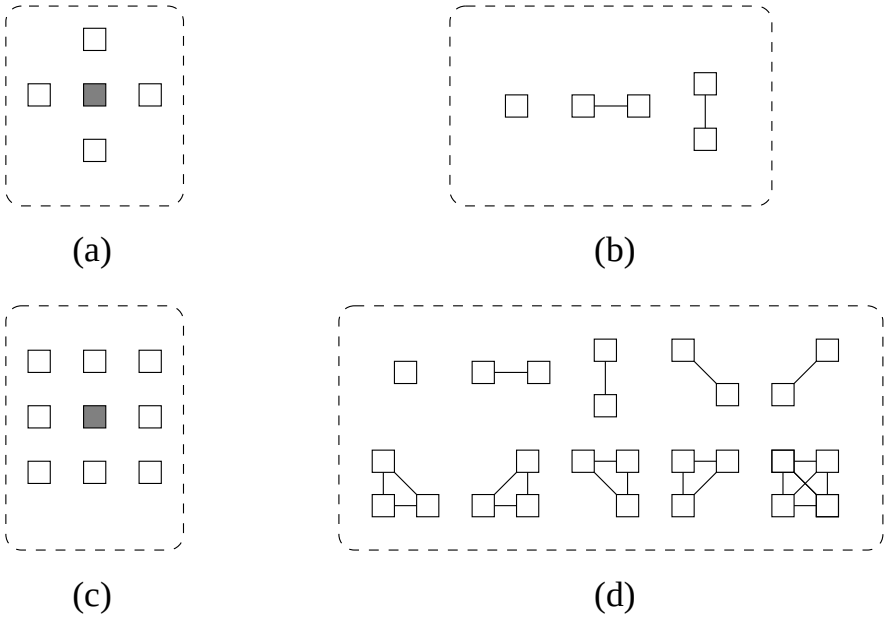


Figure 2.2: Example neighborhood systems of MRFs for images: (a) Neighborhood system with $c = 1$, (b) Cliques for neighborhood system with $c = 1$, (c) Neighborhood system with $c = 2$, (d) Cliques for neighborhood system with $c = 2$

concept of MRFs is essentially due to Dobrushin [38]. Many others also worked in the direction of generalizing the Markovian property. An early contribution was that of Abend, Harley, and Kanal [1] on pattern recognition. They proposed the *Markov mesh* model, for which the Markovian dependence is *causal* (see [12, 98] for more on causal MRFs). The assumption of a Markov mesh 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 mesh, for which

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

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 mesh are special Markov random fields with neighborhood systems shown in Fig. 2.3.

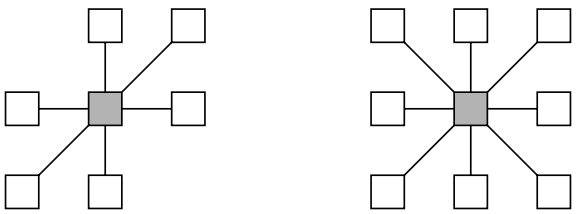


Figure 2.3: Neighborhood systems of the 2nd and 3rd order Markov mesh: (a) The 2nd order Markov mesh, (b) The 3rd order Markov mesh

A considerable amount of work [50, 56, 127, 29] has been done on applying MRFs

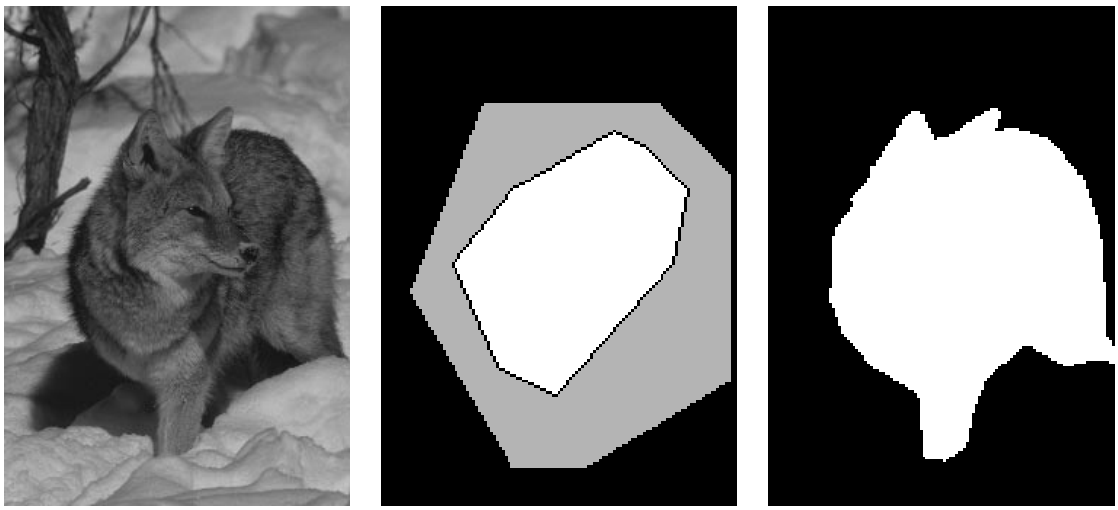
to image segmentation. In practice, to retain feasible computation, further constraints are put on the conditional distribution given all the neighboring pixels. One example is the Gaussian Markov random field for which the conditional distribution is Gaussian with parameters depending on neighboring pixels. Most 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 Multiresolution Image Classification

In recent years, substantial interest and activity in the image and signal processing community have been devoted to multiresolution processing algorithms. One reason for this interest in the particular application of image segmentation is that multiresolution processing mimics the decision procedure of the human vision system (HVS) [83]. For example, the HVS segments a picture shown in Fig. 2.4 into a foreground region (a fox) and a background region. By a glimpse of the picture, equivalent to viewing a low resolution image, the HVS obtains a global impression and is able to locate roughly where the foreground is. As is shown by Fig. 2.4(b), the crude decision leaves only a small unsure area around the boundary. A further careful examination of details at the boundary results in the final decision. Two observations: the HVS makes decisions with both global and local information; and the HVS distributes effort unevenly by looking at more ambiguous regions in higher resolutions.

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. With the original image being the highest resolution, lower resolutions are simply the low frequency bands of wavelet transforms. See [113, 77] for examples.

Another approach to use multiresolution information is to form multiresolution models. One example is the multiscale autoregressive model proposed by Basseville *et al.* [6]. Suppose images are represented by R resolutions, with $r = 1$ being the



(a) (b) (c)

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

crudest resolution. Nodes, or pixel indices, at resolution r are

$$\mathbb{N}_m^{(r)} = \{(i,j) : 0 \leq i,j \leq m/2^{R-r} - 1\} \ .$$

As is shown by Fig. 2.5, a quadtree expansion at every refined resolution is assumed. Pixel intensities at resolution r are denoted by $X^{(r)} = \{X^{(r)}(i,j) : (i,j) \in \mathbb{N}_m^{(r)}\}$. 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 entails 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 was mentioned, the HVS is not only accurate but also fast, whereas algorithms using multiresolution features or models are not necessarily fast because they simply

collect information in several resolutions to make a decision on a local region. Suggested by the observation that the HVS achieves fast segmentation by viewing higher resolutions selectively for ambiguous regions, I propose a multiresolution classification structure shown in Fig. 2.6.

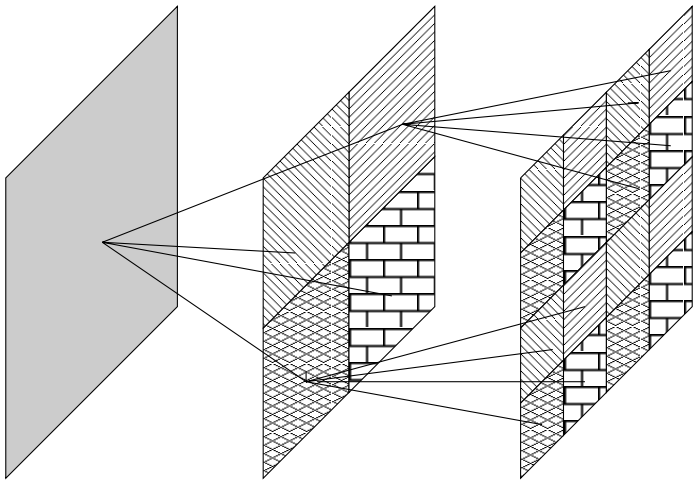


Figure 2.5: The quadtree expansion to higher resolutions

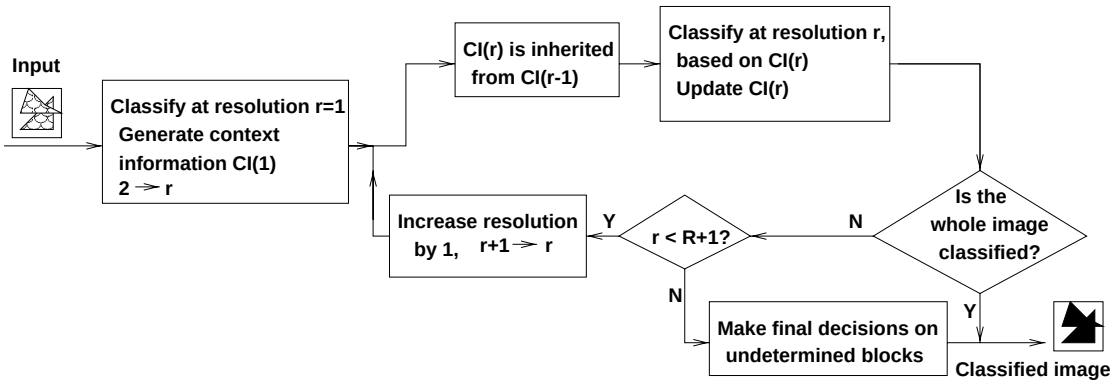


Figure 2.6: A multiresolution classification structure

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 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 used in a few applications [72, 74].

Chapter 3

Vector Quantization

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 remarkable increasing 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, each 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, in order to view real time video sent by a web server, at least $9.12 \times 10^6 / 5.6 \times 10^4 \approx 163$ times compression is needed.

According to whether a receiver can recover compressed data without error, data

compression systems are categorized into lossless compression, or entropy coding, and lossy compression. Entropy coding is required for text compression and more so for computer programs and financial information [11, 107, 28]. It is also often cascaded after a lossy compression system to further reduce bit rates. Speech and image compression [4, 18, 52] 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 bit rate, whereas a picture compressed to 0.5 bits per-pixel by state of the art algorithms [109, 104, 126] 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 [108] 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 [22, 59, 18] 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*, where $\hat{x}_i$ are *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 $\mathcal{A}$. Denote each region of the partition by $\mathcal{A}_i$, $\bigcup_{i=1}^{N} \mathcal{A}_i = \mathcal{A}$, then

$$\alpha(x) = i \quad \text{if } x \in \mathcal{A}_i\ .$$

The decoder β is a mapping from $\{1, ..., N\}$ to $\mathcal{C} = \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_N\}$, that is,

$$\beta(i) = \hat{x}_i\ .$$

In particular for image compression, a common 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.

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 with codebook size N 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

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

Necessary conditions for the optimality of a quantizer are specified by the Lloyd conditions [76].

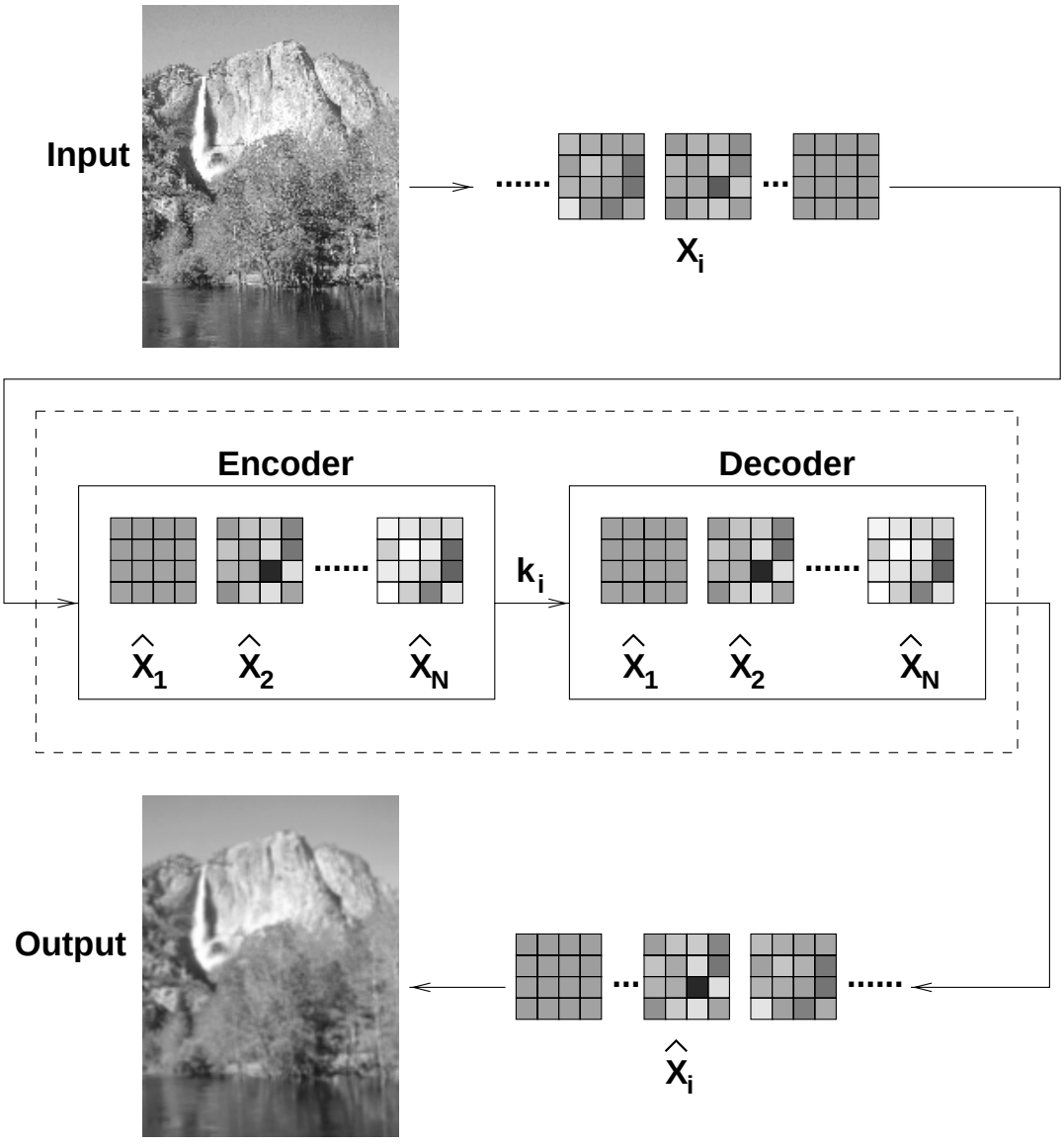


Figure 3.1: Vector quantization for image compression

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

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

  which is the 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 increases 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 many 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 [75, 78, 25, 103] are obtained by recursively splitting already designed quantization cells. The tree structured partition leads to linear encoding complexity in rk. HVQ [53, 23, 115, 21, 21] implements encoding as pure table lookup. In the next section, transform VQ is discussed in detail.

3.2 Transform VQ

Transform vector quantization [101, 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.

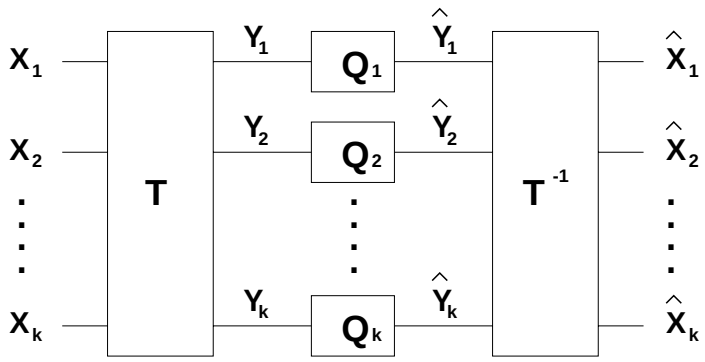


Figure 3.2: A transform vector quantizer

Although a transform vector quantizer may apply vector quantization to several coefficients grouped as one vector, scalar quantization is used more often. The main reason is that a transform can be chosen to decorrelate the components of a random vector so that statistical redundancy among components is greatly reduced. Another advantage of transform VQ is 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. Replacing these coefficients by their mean values results in an inverse transform close to the original vector. Since most of the information in the original vector is contained in a subset of the coefficients, scalar quantization applied to the transformed vector components achieves significantly better performance than scalar quantization applied to the original vector components.

Many international standard image and video compression algorithms use transform VQ. 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 [30, 114, 83] are another important class of transforms used by many state of the art algorithms [109, 104, 126]. 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 random 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 [43, 42, 118, 41] searches for better representative points that give a faster convergence rate. The measure of "loss" used by the number-theoretic method is *discrepancy* [41]. 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 [13] 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 [41]

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

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 [57, 58, 14, 99], 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, I 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. [87, 88, 94, 95, 92, 96, 97, 84, 19]. 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

classification error and a small portion of compression distortion may result in better classification than a pure classifier, and vice versa [97, 92].

The joint compression and classification algorithm developed by Oehler, Gray, Perlmutter, Olshen, et al. [87, 88, 94, 95, 92, 96, 97], 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

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

   which is the Bayes optimal classifier given the encoded input.

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

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

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. [86, 87, 88], is to replace Bayes risk by the compression distortion in the encoding process, which is referred to as the Bayes VQ with MSE encoding [92], since the mean squared error is normally taken as the compression distortion. The more sophisticated approach taken by Perlmutter [92, 97] 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 [92, 97] 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

For most block-based image classification algorithms, such as BVQ [97], images are divided into blocks, and decisions are made independently for the class of each block. This approach leads to an issue of choosing block sizes. 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. A well known method in signal processing to attack this type of problem is to use context information. Trellis coding [52] in image compression is such an example. How to introduce "context" into classifiers is what is of interest to us. Previous work [48, 72] has looked into ways of taking advantage of context information to improve classification performance. Both block sizes and classification rules can vary 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.

I first review the basics of one dimensional HMMs that are used widely in speech recognition. I then discuss the extension to two dimensions. The main difficulty with applying the 2-D model to image 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 [7, 8, 9, 10]. HMMs have earned their popularity in large part from successful application to speech recognition [5, 91, 100, 60, 26]. Underlying an HMM is a basic Markov chain [80]. In fact, an HMM is simply a "Markov Source" as defined by Shannon [106] and Gallager [49]: a conditionally independent process on a Markov chain or, equivalently, a Markov chain viewed through a memoryless 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, I

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 [10] (later shown to be a special case of the EM algorithm [31]), 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 [10]. 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

$$\begin{aligned} L_i(t) &= P(s_t = i \mid \mathbf{u}) = \frac{P(\mathbf{u}, s_t = i)}{P(\mathbf{u})} \\ &= \frac{1}{P(\mathbf{u})} \alpha_i(t) \beta_i(t) \end{aligned}$$

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

For details, see any of the references on speech recognition [91, 100, 60, 125].

4.2 Viterbi Training

An approximation to the maximum likelihood training provided by the Baum-Welch algorithm is what is often termed Viterbi training [125], in which each observation is assumed (with weight of 1) to have resulted 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 [116] is applied to maximize $P(\mathbf{s}, \mathbf{u})$ since $\max_{\mathbf{s}} P(\mathbf{s}, \mathbf{u})$ can be computed by the recursive formulae

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

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 the pseudo 2-D HMM [69, 124]. 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 [69], 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 [32, 33, 34]. 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 [34, 35, 36]. In [32], it was noted that the complexity of estimating the models or using them to perform maximum a posteriori (MAP) 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 [32, 33, 34] were developed by making additional assumptions regarding models or using locally optimal solutions. Worth noting is the deterministic relaxation algorithm [32] 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 4.6, I derive analytic formulas for model estimation and show that computation is exponential in w by using a forward-backward-like algorithm. A suboptimal algorithm is described in Section 4.8 to achieve polynomial-time complexity.

Other work based on 2-D HMMs includes an algorithm for character recognition developed by Levin and Pieraccini [71], and an image decoding system over noisy channels constructed by Park and Miller [90]. In [90], 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 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. I 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 made 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 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.

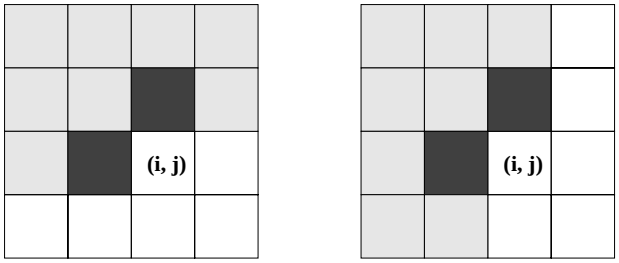


Figure 4.1: The Markovian property of transitions among states

4.5 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 $\mathbb{N} = \{(i,j) : 0 \leq i < w, 0 \leq j < z\}$ refers to all the blocks in an image. To expand this probability efficiently by the conditional probability formula, I 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 $j' < j$. I 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

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

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

$$\tilde{\Psi} \cup \Psi = \{(i',j') : (i',j') < (i,j) \text{ or } (i',j')\tilde{<}(i,j)\} \ ,$$

$$\tilde{\Psi} \cap \Psi = \{(i',j') : (i',j') < (i,j) \text{ and } \ (i',j')\tilde{<}(i,j)\} \ ,$$

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

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

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

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.

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

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

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, I have shown the equivalence of the two assumptions:

$$
P(s_{i,j} \mid s_{i',j'}, u_{i',j'} : (i',j') \in \Psi) \;\; = \;\; P(s_{i,j} \mid s_{i-1,j}, s_{i,j-1}) \text{ and}
$$

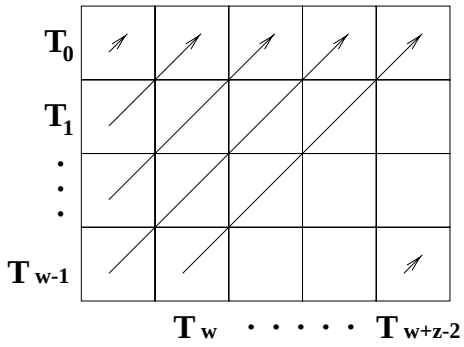


Figure 4.2: Blocks on the diagonals of an image

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

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 $P\{s_{i,j} : (i,j) \in \mathbb{N}\}$:

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

where T_i denotes the sequence of states for blocks on diagonal i, $\{s_{i,0}, s_{i-1,1}, \cdots, s_{0,i}\}$, and w and z are the number of rows and columns respectively, as shown in Fig. 4.2.

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

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

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

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

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 4.8.

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.

4.6 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 [31, 123, 10]. First, the EM algorithm as described in Dempster, Laird and Rubin [31] 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, I introduce a function [31]

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

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 [31] 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

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

We then have

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

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

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

where α is a normalization constant, and $I(\cdot)$ is the obvious indicator function. From this point, I 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

$$\begin{aligned} E(\log f(\mathbf{x} \mid \phi') \mid \mathbf{y}, \phi^{(p)}) = \frac{1}{\alpha} \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \cdot \sum_{(i,j)\in\mathbb{N}} \log a'_{s_{i-1,j}, s_{i,j-1}, s_{i,j}} + \\ \frac{1}{\alpha} \sum_{\mathbf{s}} P(\mathbf{s} \mid \mathbf{y}, \phi^{(p)}) \sum_{(i,j)\in\mathbb{N}} \log P(u_{i,j} \mid \mu'_{s_{i,j}}, \mathbf{\Sigma}'_{s_{i,j}}) \, . \quad (4.10) \end{aligned}$$

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

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

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 attempt 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 equation (4.10).

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

To simplify the above expression, let

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

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

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

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 $\mathbf{\Sigma}'_m$ is the sample covariance matrix of the data [13]. Since in our case, the data are weighted by $L_m^{(p)}(i,j)$, the ML estimate of μ'_m and $\mathbf{\Sigma}'_m$ are

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

In summary, the estimation 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

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

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

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

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

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

$$H^{(p)}_{m,n,l}(i,j) = \sum_{\mathbf{s}} I(m = s_{i-1,j}, n = s_{i,j-1}, l = s_{i,j}) \cdot \frac{1}{\alpha} I(C(\mathbf{s}) = \mathbf{c}) \cdot \prod_{(i,j)\in\mathbb{N}} a^{(p)}_{s_{i-1,j}, s_{i,j-1}, s_{i,j}} \cdot \prod_{(i,j)\in\mathbb{N}} P(u_{i,j} \mid \mu^{(p)}_{s_{i,j}}, \mathbf{\Sigma}^{(p)}_{s_{i,j}}) \,. \tag{4.15}$$

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 $H_{m,l}(k)$ [125] 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, I discuss why this is so and how to reduce the computational complexity.

4.7 Computational Complexity

As is 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 large 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 $(w \times z - 1)^{M_0}$. When applying the HMM algorithm, although one image is often divided into many sub-images such that w, or z, is the number of blocks in one column, or one row, in a sub-image, 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 there are 4 states for each class, the number of the combinations of states is $(w \times z - 1)^{M_0} = 63^4$, which is prohibitive for a straightforward calculation of $L^{(p)}_m(i,j)$. Similar difficulty occurs for estimating a one dimensional HMM. The problem is solved by a recursive calculation of forward and backward probabilities [125].

The idea of using forward and backward probabilities can be extended to the two dimensional HMM to simplify the computation. Recall Equation (4.8) in Section 4.4,

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

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 4.6, we need the following definitions:

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 $\mathbf{M}$ is defined as

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

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

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

that is, $\beta_T(d)$ 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 $\beta_T(d)$, which are listed below.

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

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

We can then compute $L_m(i,j)$ given model $\mathbf{M}$ by

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

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.

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

The subscript 'd' in T_d denotes the diagonal d of block (i,j). In the following calculation of $H_{m,n,l}(i,j)$, the summations are always over state sequences with the same classes as those in the training data.

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

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 $T_{d-1}(i-1,j) = m$, $T_{d-1}(i,j-1) = n$.

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

Although using the forward and backward probabilities significantly reduces the computation for $L_m(i,j)$ and $H_{m,n,l}(i,j)$, computational complexity is still rather 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. With the increase of blocks on a diagonal, the number of state sequences increases exponentially. The same problem occurs with 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 4.6

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

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

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 model $\phi^{(p)}$ can be solved by the Viterbi algorithm, which is described in the next section.

4.8 Variable-state Viterbi Algorithm

Our purpose of using the Viterbi algorithm is to maximize $P(\mathbf{s} \mid \mathbf{y})$, which 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 aim at finding 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 corresponding to class $c_{i,j}$. Expand $P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$ as follows

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

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

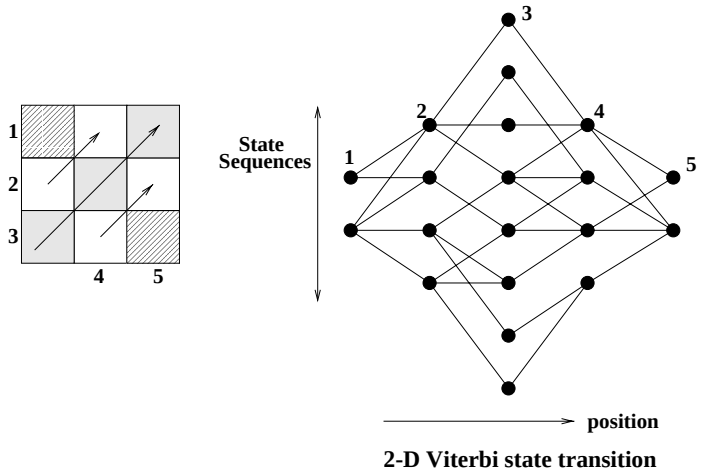


Figure 4.3: The variable-state Viterbi algorithm

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 combination of states maximizing the likelihood $P\{s_{i,j}, u_{i,j} : (i,j) \in \mathbb{N}\}$. The 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. 4.3 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. 4.4. The paths are constrained to pass one of these N nodes. To choose the N sequences of states, the algorithm separates the blocks in the diagonal from the other blocks by ignoring their statistical dependency. Consequently, the posterior probability of a sequence of states on the diagonal is evaluated as a product of the posterior probability of every block. Then, the N sequences with the largest posterior probabilities

are chosen 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 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 [61] since the N nodes are pre-selected to avoid calculating the posterior probabilities of all the M^ν state sequences.

As mentioned in Section 4.5, 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 chosen for the expansion because intuition suggests that the pre-selection of N nodes by ignoring dependence among states on a diagonal 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 choosing such N sequences of states. It is not necessary to calculate the posterior probabilities of all the M^ν sequences in order to choose 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 we ignore context information in the pre-selection of nodes. 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 for finding the N sequences with the largest log likelihood.

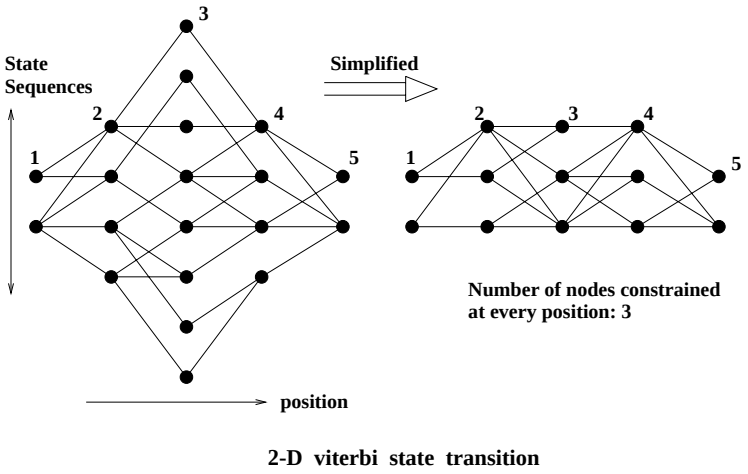


Figure 4.4: The path-constrained Viterbi algorithm

To ensure that the path-constrained variable-state Viterbi algorithm yields results sufficiently close to the variable-state Viterbi algorithm, the parameter N should be larger when there are more blocks in the 2-D Markov chain. As a result, an image is usually divided into sub-images to avoid too many blocks in one chain. Every sub-image is assumed to be a 2-D Markov chain, but the dependence between sub-images is ignored. On the other hand, to incorporate any preassigned amount of context information for classification, the sub-images must contain sufficiently many blocks. The selection of the parameters will be discussed in the section on experiments.

4.9 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 according to the pixel intensities in a block. They aim at describing the statistical properties of the block. 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 [125]. 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 some 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, I 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.

4.10 Aerial Image Segmentation

4.10.1 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.) [86]. 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 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. 4.5. The definitions of the 6 features are:

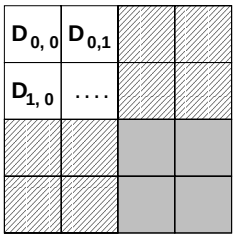


Figure 4.5: DCT coefficients of a 4×4 image block

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. $f_6 = \sum_{i=2}^{3} \sum_{j=2}^{3} |D_{i,j}|/4$.

In addition, 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.

4.10.2 Results

Six-fold cross-validation [112] was used to evaluate algorithms. 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 may not be so sophisticated; and more complicated models require a larger training set. With a fixed training set, the accuracy of estimation becomes less 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 sub-images each containing 16 blocks.

The sub-images 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. I 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 LINUX operating system. The average user CPU time to classify an aerial image is 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 5.4.

The 2-D HMM result was compared with those obtained by CART^{TM} [17] and LVQ1 [66]. As described in Section 2.2, CART is developed for general purposes of decision tree design. We can thus apply it in the scenario of 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, I used programs provided by the LVQ_PAK software package [67]. 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 decreases the error rate to 18.80%. The classification results for Image 6, the image shown in Fig. 4.6(f), are given in Fig. 4.7. 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 [86] and Perlmutter [92]. In both cases, the Bayes vector quantizer (BVQ) [86, 92, 87, 88] 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% [92], comparable to CART 1.

4.11 Document Image Segmentation

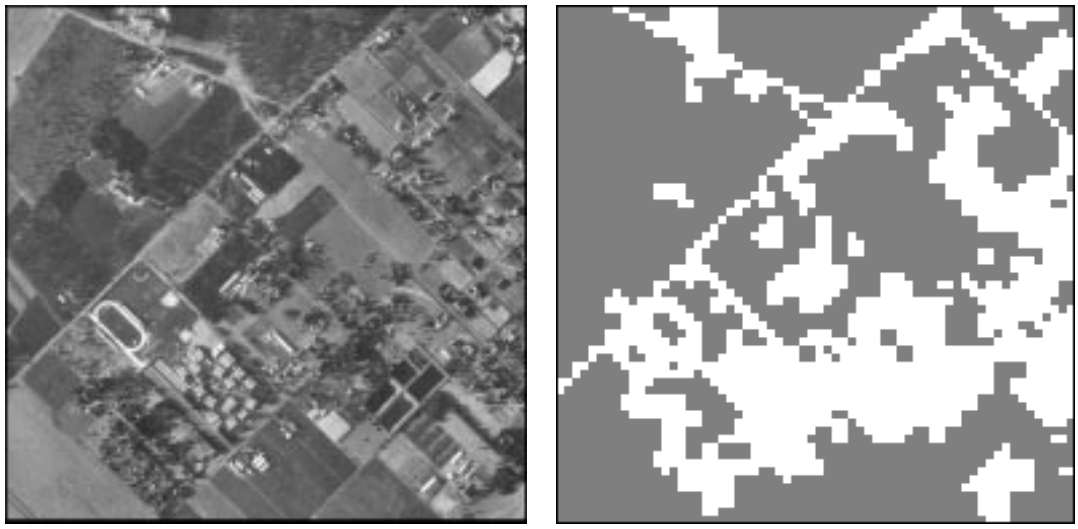
4.11.1 Related Work

The second application of the 2-D HMM algorithm is to segmentation of document images into text and photograph. Photograph refers to continuous-tone images such as scanned pictures; and text refers to normal text, tables, and artificial graphs generated by computer software [73]. I 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. 4.10. 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 [20],

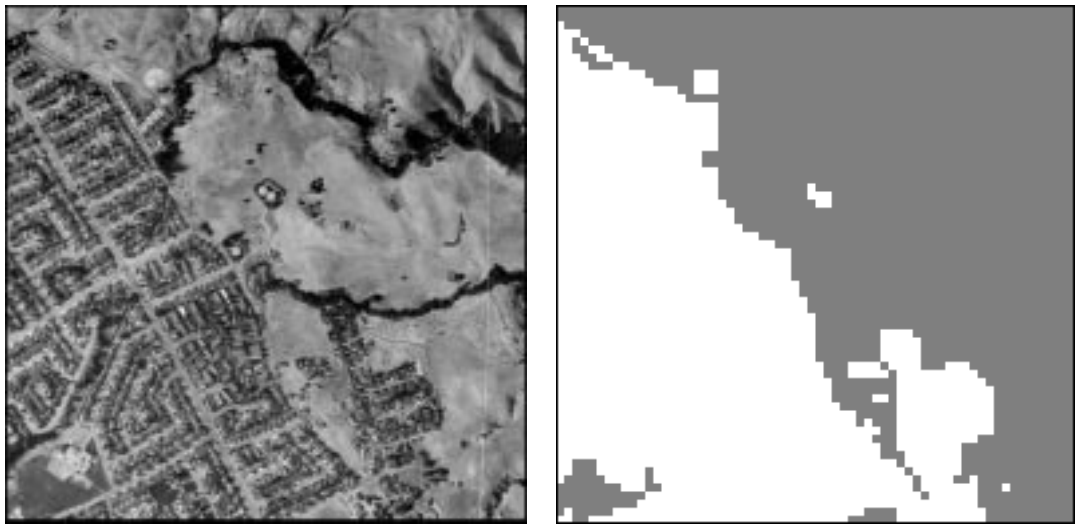
Algorithm	Iteration	sensitivity	specificity	PVP	P_e
2-D HMM	1	0.6250	0.9171	0.8146	0.1904
	2	0.8717	0.6141	0.9074	0.1765
	3	0.9188	0.6974	0.7114	0.2034
	4	0.5543	0.9201	0.8446	0.2405
	5	0.8152	0.9196	0.9986	0.1834
	6	0.8919	0.8533	0.7518	0.1339
	Ave	0.7795	0.8203	0.8381	0.1880
CART 1	1	0.7870	0.7660	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

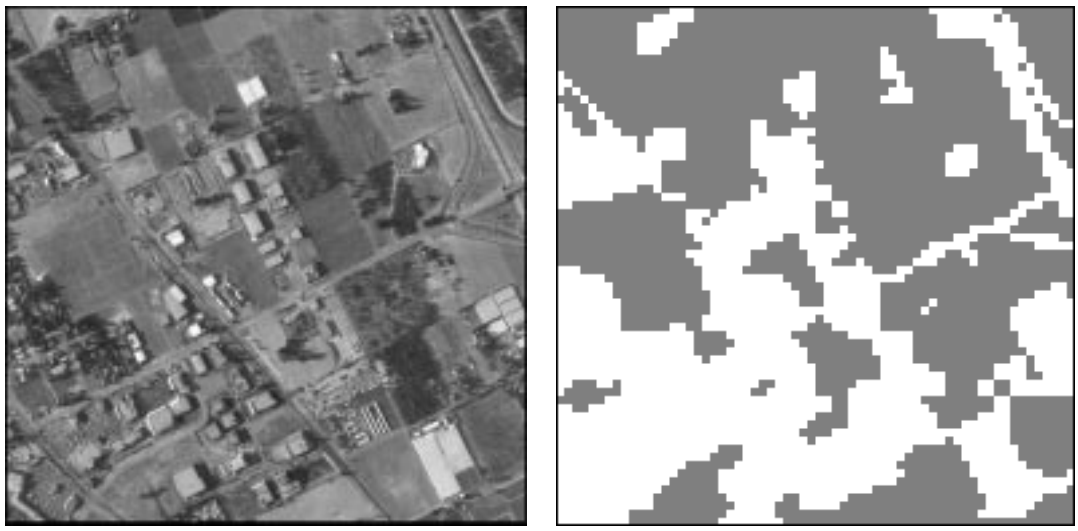


(a)

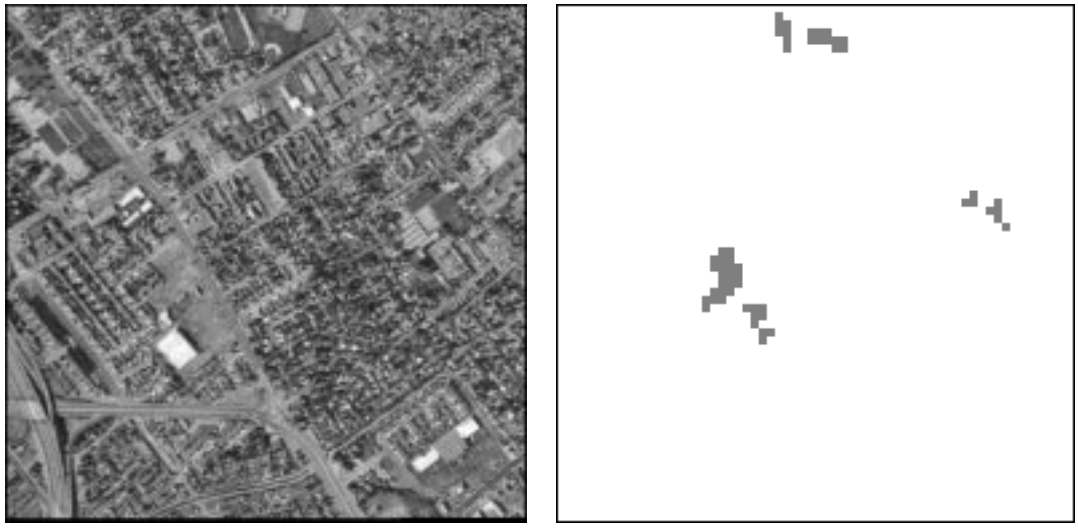
(b)



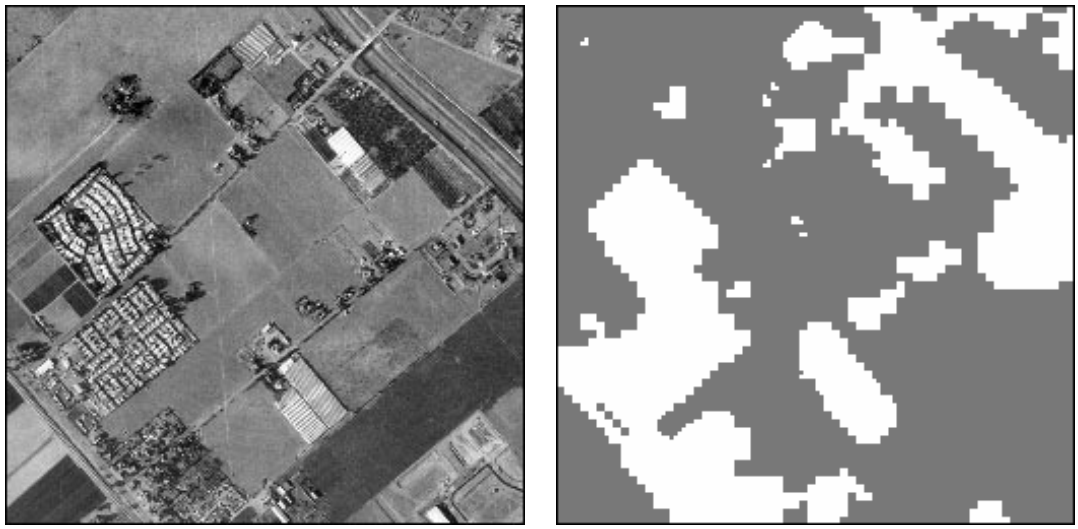
(c)



(d)

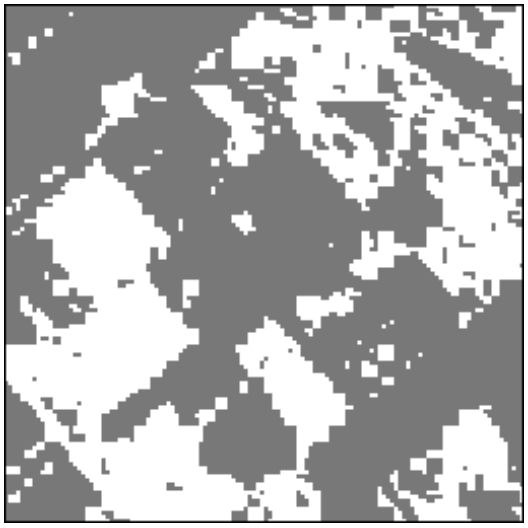


(e)

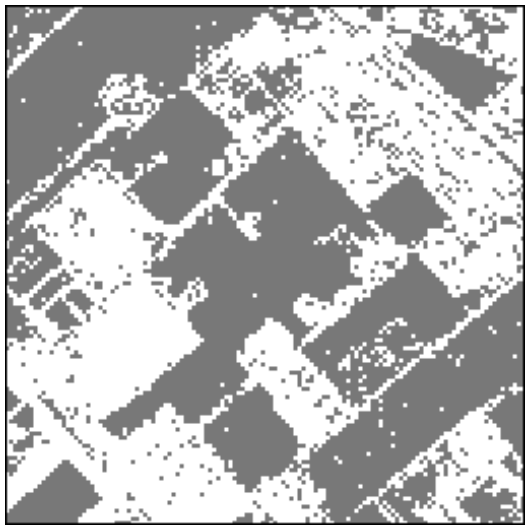


(f)

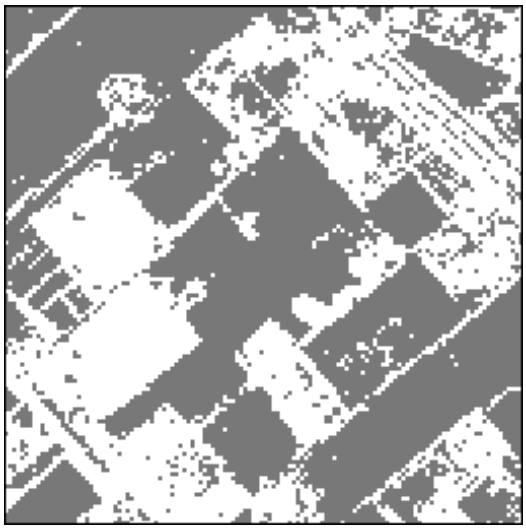
Figure 4.6: Aerial images: (a)∼(f) Image 1∼6. Left: Original 8 bpp images, Right: Hand-labeled classified images. White: man-made, Gray: natural



(a)



(b)



(c)

Figure 4.7: 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

Williams [121], Perlmutter [93, 92], and Ohuchi [89]. Thresholding is used to distinguish image types in [20]. In [121], a modified quadratic neural network [85] is used for classifying features. In [93, 92], the Bayes VQ algorithm is applied.

Another type of document image classification studied more often is the segmentation of half-tone (binary) document images [110, 45, 47, 117, 119, 105, 39]. For binary images, run-length statistics, e.g., the average horizontal run-length of black dots, are important for distinguishing text and photograph [110, 122, 39]. An algorithm proposed by Shih [110] combines techniques in Wong [122] and Fisher [45]. 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.

4.11.2 Feature Extraction

Wavelet transforms [102, 79] were used to form features. Wavelet transforms have played an important role in the classification of texture [113, 77] and abnormalities in medical images [37, 120]. In classification algorithms, the principal wavelet-based approach is to form features based on the statistical behavior of wavelet coefficients. Moments of wavelet coefficients are the most commonly used [113, 77, 37, 120]. 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 in High Frequency Bands

It has been observed that for photographs, wavelet coefficients in high frequency bands, i.e., LH, HL, and HH bands [114], tend to follow a Laplacian distribution. 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. As an example, the histograms of Haar wavelet

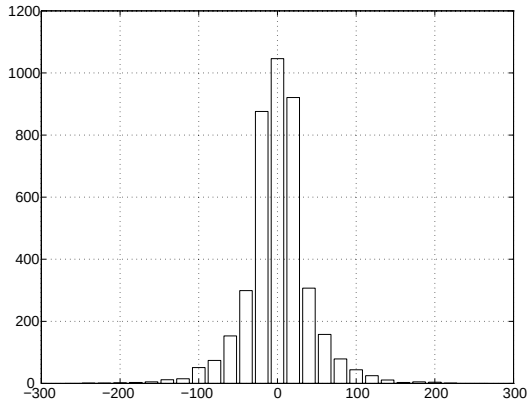
coefficients in the LH band for one photograph image, one graph image, and one text image are plotted in Fig. 4.8. Another important difference to note is the continuity of the observed distribution. The histograms of the graph image and the text image suggest that the values are highly concentrated on a few discrete values, but the histogram for the photograph image shows much better continuity of distribution. In the special case shown in Fig. 4.8, the histogram of the text image contains five bins that are quite distinct because the text image has bi-level intensities. For the graph image, it happens that a very high percentage of data are near 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 dominates in the same way. Instead, the histogram has a peak at zero and attenuates roughly exponentially. It is worth pointing out that in practice the histograms of the three types are usually not as extreme as those of 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.

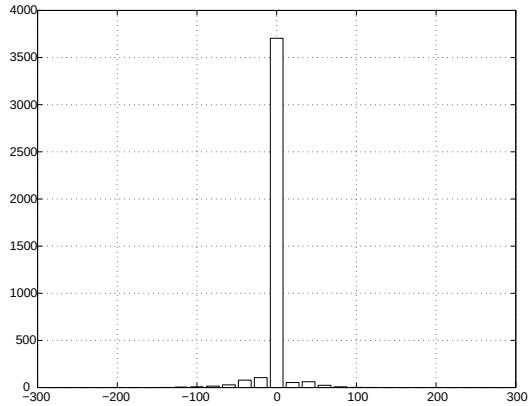
Goodness of Fit to the Laplacian Distribution

In order to measure the goodness of match between the observed distribution and the Laplacian distribution, I use the χ^2 test [111] normalized by the sample size N, denoted by $\bar{\chi}^2$. The χ^2 test is a widely applicable measure of how well a set of observed data matches a specified theoretical distribution.

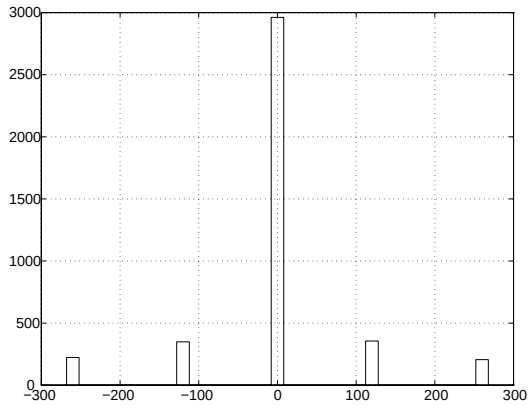
Suppose we have a set of data, denoted by $\{x_1, x_2, \ldots, x_N\}$. The theoretical probability density function to which we compare the observed data is $p(t), t \in \Re$. We divide the set of data into ν categories and denote the number of x_j's in category i by m_i. Usually, the categories are consecutive intervals in the domain of the data.



(a)



(b)



(c)

Figure 4.8: Histograms of wavelet coefficients in the LH band: (a) Photograph image, (b) Graph image, (c) Text image

Specifically, if we denote category i by C_i, then

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

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

$$M_i = N \cdot F_i \ .$$

The χ^2 test criterion is defined as

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

Pearson [111] showed that if the observed data are drawn randomly 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 [111] 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 [111]. 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 [111] of the observed data, the number of degrees of freedom is then $\nu - 2$.

To measure the goodness of match between the observed distribution and the Laplacian distribution, we need to determine the probability density function of the Laplacian distribution. Recall that the pdf of the Laplacian distribution is

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

According to the fact $VAR[X] = 2/\lambda^2$, we estimate the parameter λ 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 evaluated 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, the χ^2 value 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, Fig. 4.8(c) shows that data only lie in the five widely separated categories, indicating a high L. The efficiency of L estimating the likelihood of the wavelet coefficients having 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.

As the fact that data tend to concentrate at a few discrete values also applies to the absolute values of the wavelet coefficients, histograms of the absolute values are

used to calculate L in order to speed up computation. Before defining the function L, let us look at an example to gain some intuition. Suppose we have a histogram as shown in Fig. 4.9. It seems reasonable to assume that V_0 and V_1 are two concentration values in the histogram. They are both peak values (maximum values) in data ranges $[0, Z_1]$ and $[Z_1, Z_2]$. Besides, most of the data in 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 $[Z_1, Z_2]$, we do not take V_2 as a concentration value since it looks more like a fluctuation around V_1; we do not take V_3 and V_4 as concentration values, since neither of them has a narrow neighborhood containing most data in the range $[Z_2, Z_3]$. Abstracting from the example, we would expect a concentration value to have the following properties:

1. It is the maximum value in a certain data range, say $[Z_1, Z_2]$. We define such a data range 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 find concentration values, it is essential to divide the data into zones according to the histogram. Intuitively, we imagine a zone to be a range, isolated from the remainder of the data axis, in which data form clumps. The histogram in a zone should have a peak value and vanish towards the ends of the zone. I now provide a more rigorous definition that captures these ideas.

Suppose the complete data range is $[0, Z]$. The histogram on $[0, Z]$, normalized by the total number of samples, is represented by a nonnegative 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 $t^* \in (t_i, t_{i+1}), \text{s.t.} h(t^*)$ is the maximum value for $t \in [t_i, t_{i+1}]$.

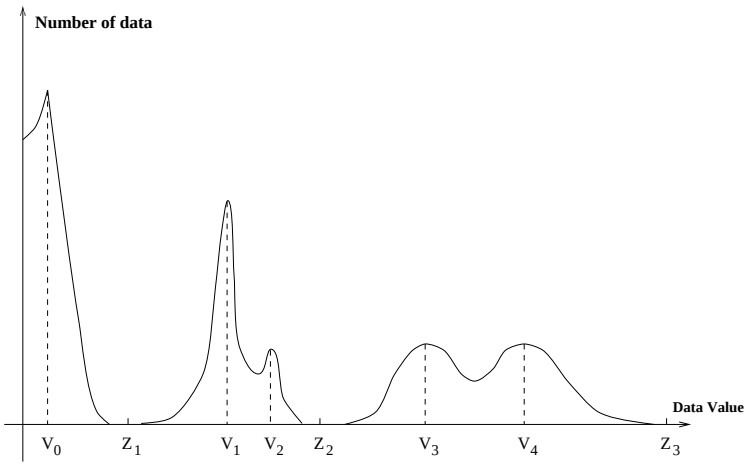


Figure 4.9: The concentration values of a histogram

(b) $h(t_i) \leq h(t)$, for all $t \in (t_i, t^*)$
$h(t_{i+1}) < h(t)$, for all $t \in (t^*, t_{i+1})$

(c) $h(t_i)/h(t^*) < \delta$, and $h(t_{i+1})/h(t^*) < \delta$, where δ is a threshold.

(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 a few relaxed conditions listed below:

   (a) We allow the maximum point t^* 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 all $t \in (t_i, t^*)$.

3. If $t_{i+1} = Z$, we only have one condition for $[t_i, t_{i+1}]$ to be a zone:

   (a) There does not exist $[\tau, \tau'] \subset [t_i, t_{i+1}]$ and $[\tau, \tau'] \neq [t_i, t_{i+1}]$, so that $[\tau, \tau']$ meets the previous definition for a zone.

In the list of conditions above, the first condition applies to a normal zone, while the second and the third conditions allow for special cases at the ends of the range of the data $[0, Z]$. For a normal zone, the first requirement is that the zone contains a maximum inside it. This maximum point is the candidate concentrated value in

the zone. The second requirement 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 requirement guarantees that the empirical distribution densities at the end points of the zone are small enough compared with the maximum in the zone. This requirement guards against local fluctuations of the distribution. In addition, the combination of the second and the third requirements forces the histogram to vanish towards the ends of the zone. The fourth requirement ensures the partition of $[0, Z]$ to be the finest possible. The second and the third conditions specify the requirements for a zone located at one end of the data range. Some requirements in the first condition are dropped so that a partition satisfying the above definition 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 found to be one zone, and the concentration level of the zone is nearly zero.

The algorithm to find the partition of $[0, Z]$ is described in detail in Appendix A. In order to define L, I suppose the partition is obtained, and start with defining a concentration level for every zone. Then, L will be 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 as sufficiently close to t^*, is set to w. Subject to the constrained 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 the zone $[t_i, t_{i+1}]$ 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 and p_i', which is

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

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 β_i's are obtained, L is evaluated by

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

Note that $0 \leq L \leq 1$. The parameter w decreases with the decrease of the sample size of the histogram. The reason is that, for a block of pixels, when the block is small, the pixels are more likely to have similar intensities. In order to be sure that the block has highly concentrated values, the width for closeness should be smaller.

4.11.3 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 is 8×8. The HMM has 5 states for each class. Experiments show that models with 2 to 5 states for each class yield similar results.

The result of HMM is 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. [93, 92]. They are RGB color images with size around 1600×1300. 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 one sample image shown in Fig. 4.10 because the photograph region in this image is very smooth in many places, so it resembles text. The classification results of both CART and the 2-D HMM algorithm are shown in Fig. 4.10. We see that the result using the HMM is much cleaner than the result using CART, especially in the photograph regions. This is expected since the classification based on the HMM takes context into consideration. As a result, some smooth blocks in the photograph regions, which locally resemble text blocks, can be identified correctly as photograph. This trend is also demonstrated in Fig. 4.11 by another example test image.

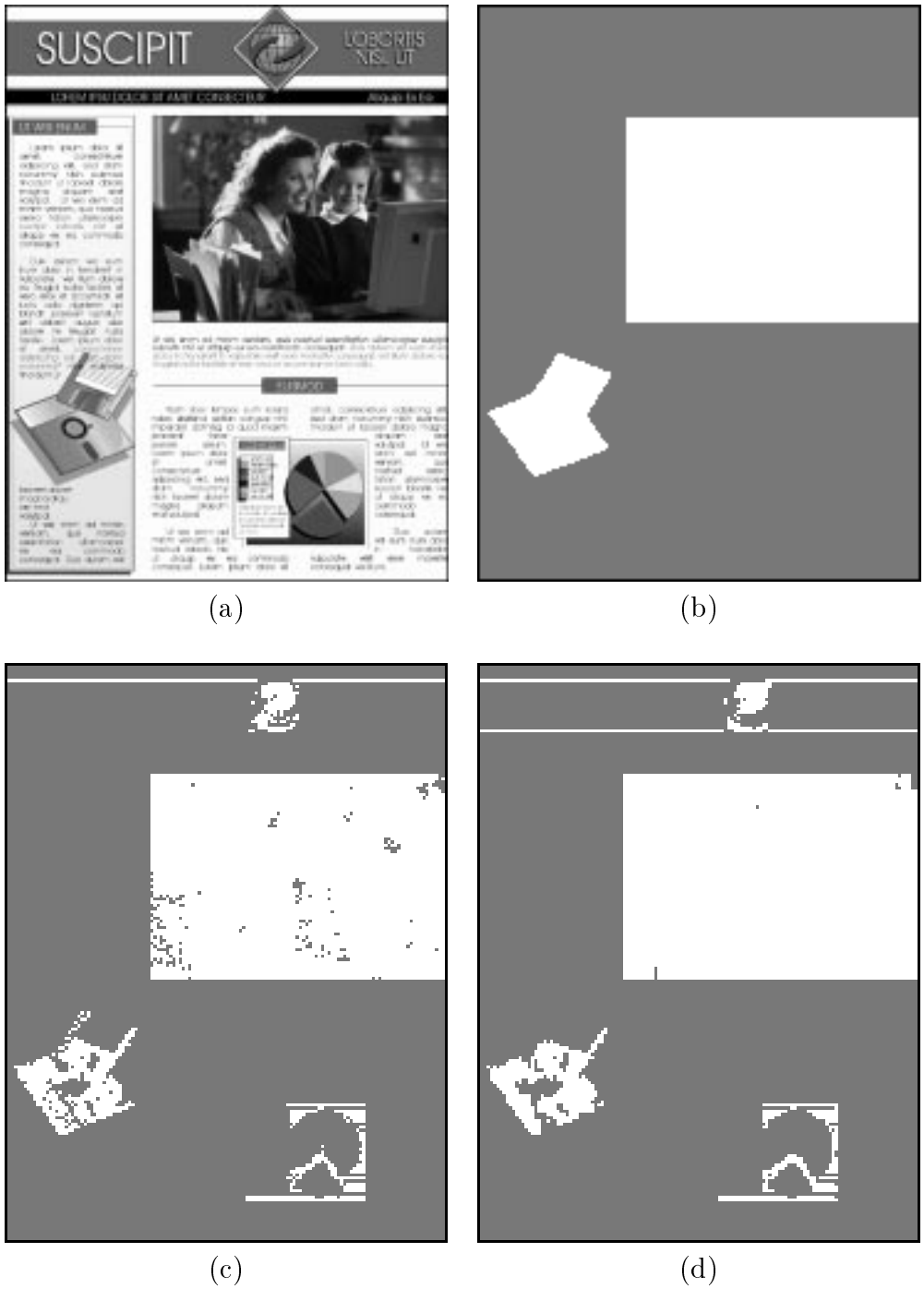


Figure 4.10: Test document image 1: (a) Original image, (b) Hand-labeled classified image, (c) CART classification result, (d) 2-D HMM classification result. White: photograph, Gray: text

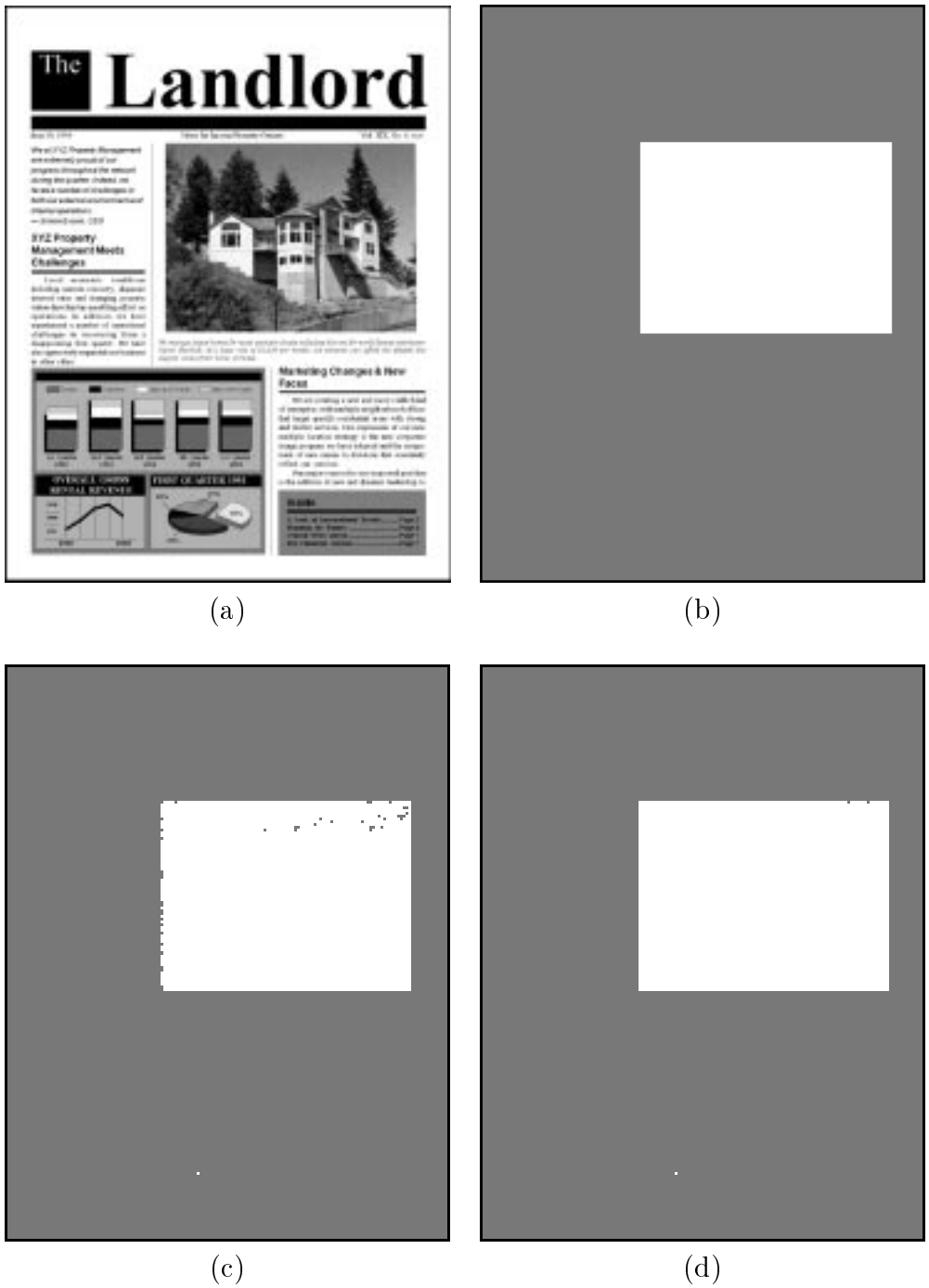


Figure 4.11: Test document image 2: (a) Original image, (b) Hand-labeled classified image, (c) CART classification result, (d) 2-D HMM classification result. White: photograph, Gray: text

Chapter 5

2-D Multiresolution HMM

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, we enable context information to be incorporated in classification. To optimally classify an image based on the 2-D HMM, we need to make a joint decision on classes for the entire image. In real life, however, because of computational complexity, we have to break up 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. In order to overcome the computational complexity so that context information can be used more efficiently, the 2-D HMM is extended to a multiresolution model which represents images hierarchically. The extended model also improves classification by using features extracted from multiple resolutions. Readers are referred to Section 2.4 for background on multiresolution image segmentation.

The 2-D multiresolution hidden Markov model (MHMM) views an image as a collection of feature vectors in several resolutions. Feature vectors in a particular resolution are determined only by the image at that resolution. The 2-D MHMM assumes that 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 in any resolution. Given the state of a block in a particular resolution, the

feature vector in this resolution is assumed to follow a Gaussian distribution. The parameters of the Gaussian distribution depend upon both the state and the resolution. At any fixed resolution, as with the 2-D HMM, the probability of the source entering a particular state obeys the 2-D Markovian assumption. The transition probabilities, however, depend on states in previous resolution as well.

5.1 Model Assumptions

To classify an image, we first obtain several resolutions of the image. The original image corresponds to the highest resolution. Lower resolutions are generated by filtering out high frequency information. Wavelet transforms [30] naturally provide low resolution images by 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 both directions. As shown by Fig. 5.1, the number of blocks in both rows and columns decreases to a half at each lower resolution. Obviously, a block in a lower resolution covers a spatially more global region in the image. As is indicated by Fig. 5.2, the block in the lower resolution is referred to as a parent block, and the four blocks at the same spatial location in the higher resolution are referred to as child blocks. Without loss of generality, in our discussion, we always assume such a quad-tree split in a higher resolution. To simplify later description, the following definitions are introduced.

1. Post-generation block: A block in a lower resolution.

2. Pre-generation block: A block in a higher resolution.

3. Descending block: Block A is called a descending (offspring) block of Block B if A is in the hierarchy of blocks starting from B.

4. Ascending block: Block A is called an ascending block of Block B if B is A's offspring.

5. Uncle block: A non-parent block in the parent resolution.

6. Sibling block: A block in the same resolution with the same parent block.

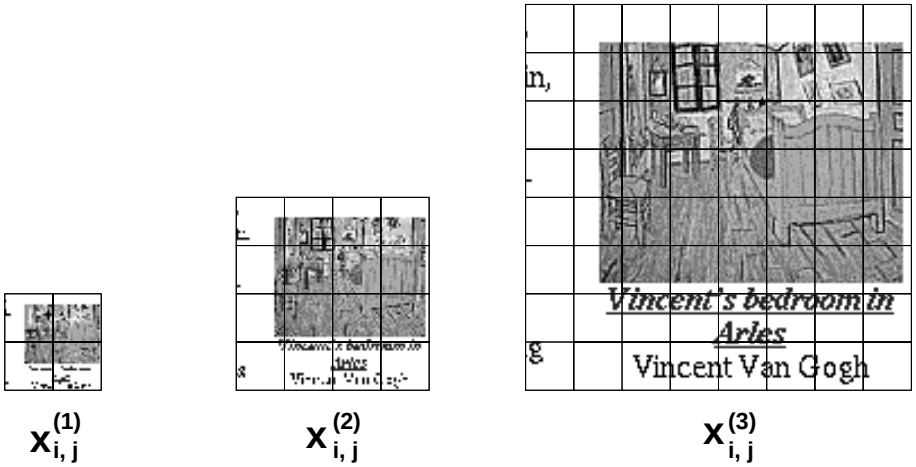


Figure 5.1: Multiple resolutions of an image

Suppose there are R resolutions, where $r = 1$ is the crudest resolution. Feature vectors, states, and classes at resolution r are $u_{i,j}^{(r)}$, $s_{i,j}^{(r)}$, and $c_{i,j}^{(r)}$ respectively, where (i, j) denotes block (i, j). Let the collection of block indices at resolution$r$ be

$$\mathbb{N}^{(r)} = \{(i,j) : 0 \le i < w/2^{R-r}, 0 \le j < z/2^{R-r}\}, \quad r \in \mathcal{R}$$

where $\mathcal{R} = \{1, ..., R\}$. At each resolution r, the set of states is $\{1^{(r)}, 2^{(r)}, ..., M_r^{(r)}\}$. Note that as we differentiate states across resolutions, different resolutions do not share states.

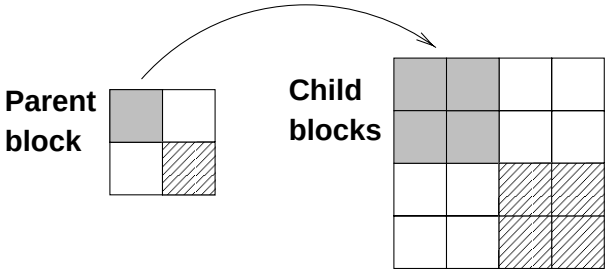


Figure 5.2: The image hierarchy across resolutions

As with the single resolution model, each state in every resolution is uniquely mapped to one class. Since a block in a lower resolution contains several blocks in a higher resolution, it may not be of a pure class. Therefore, except for the highest resolution, there is an extra "mixed" class besides the original classes. Because of the unique mapping between states and classes, the state of a parent block may constrain the possible states for its child blocks. That is, 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.

The first assumption of the 2-D MHMM is a Markovian property across resolutions. The "resolution" here plays a time-like role. Given the states and the features of the parent resolution, the states and the features of the current resolution are conditionally independent of the other previous resolutions, i.e.,

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

At the crudest resolution, $r = 1$, it is assumed that feature vectors are simply generated by a single resolution 2-D HMM. That is, state transition probabilities are $a_{m,n,l}^{(1)}$, and the feature vectors of a particular state follow a single Gaussian distribution with parameters depending on the state.

At a higher resolution, I keep the assumption that given the state of a block, the feature vector, which is independent of all the other statistics, follows a Gaussian distribution. The parameters of the Gaussian distribution depend upon the state in the particular resolution. In addition, it is assumed that given the state of a parent block, the states of its four child blocks are statistically independent of the states of their uncle blocks. It is also assumed that given the states of parent blocks, the child blocks descended from different parent blocks are statistically independent. 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

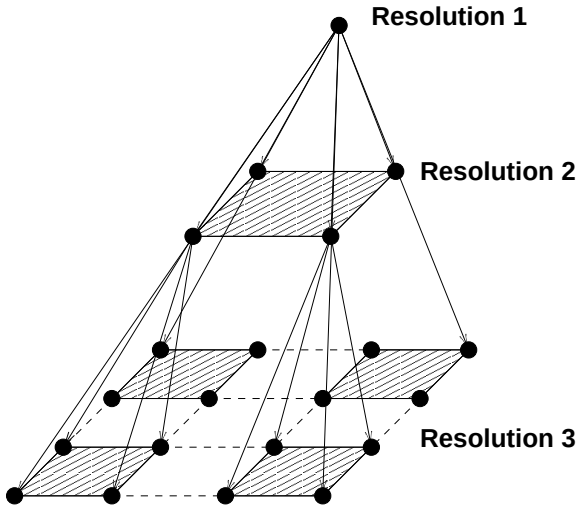


Figure 5.3: The hierarchical statistical dependence across resolutions

the child blocks in resolution r of block (k,l) in resolution $r-1$ by

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

It is assumed that

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

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 $s_{k,l}^{(r-1)}$, $a_{m,n,l}(s_{k,l}^{(r-1)})$. 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.

5.2 Model Estimation and Application

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 in all the resolutions with the maximum a posteriori probability (MAP) is searched by the Viterbi algorithm. These states are then assumed to be real states to update the estimation of parameters. Because of the existence of multiple resolutions, a certain part of the training algorithm used for the single resolution HMM is changed by a recursive procedure.

To apply a trained model to classify images, the MAP criterion is used to search for the optimal set of states in all the resolutions. The states are then mapped into classes. Since the MAP rule is used in both training and testing, the key part of model estimation and application are almost the same. The only difference is that, in training, a state chosen for a block must be mapped into the true class of the block, since true classes are given by the training data. Without loss of generality, the testing scenario is assumed in the following discussion.

For simplicity, we go through a case of two resolutions here. By induction, the algorithm can be extended to models with more resolutions. According to the MAP rule, the optimal set of states maximizes the joint log likelihood of all the feature vectors and states, that is,

$$\begin{aligned} & \log P\{s_{k,l}^{(r)}, u_{k,l}^{(r)} : r \in \{1,2\}, (k,l) \in \mathbb{N}^{(r)}\} \\ = \; & \log P\{s_{k,l}^{(1)}, u_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} + \log P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathbb{N}^{(2)} \,|\, s_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} \\ = \; & \log P\{s_{k,l}^{(1)}, u_{k,l}^{(1)} : (k,l) \in \mathbb{N}^{(1)}\} + \sum_{(k,l) \in \mathbb{N}^{(1)}} \log P\{s_{i,j}^{(2)}, u_{i,j}^{(2)} : (i,j) \in \mathbb{D}(k,l) |\, s_{k,l}^{(1)}\} \,. \end{aligned} \tag{5.1}$$

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 $s^{(1)}_{k,l}$, the child blocks in Resolution 2 are governed by a 2-D HMM with transition probabilities $a_{m,n,l}(s^{(1)}_{k,l})$, the techniques described in Chapter 4 can be applied directly. In order to clarify that $\bar{s}^{(2)}_{i,j}$ depends on $s^{(1)}_{k,l}$, I write $\bar{s}^{(2)}_{i,j}(s^{(1)}_{k,l})$ later on. The next step is to maximize

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

$$= \sum_{\tau} \Big[\log P(T^{(1)}_{\tau} | T^{(1)}_{\tau-1}) + \sum_{(k,l):\Delta(k,l)=\tau} \Big(\log P(u^{(1)}_{k,l} | s^{(1)}_{k,l}) + \log P\{\bar{s}^{(2)}_{i,j}(s^{(1)}_{k,l}), u^{(2)}_{i,j} : (i,j) \in \mathbb{D}(k,l)|\, s^{(1)}_{k,l}\} \Big) \Big] . \tag{5.2}$$

Equation (5.2) follows from Equation (4.20). As in (4.20), $T^{(1)}_{\tau}$ denotes the sequence of states for blocks on diagonal τ in Resolution 1. We can apply the Viterbi algorithm again to search for the optimal $s^{(1)}_{i,j}$ since $T^{(1)}_{\tau}$ still serves as an "isolating" element in the expansion. The only difference with the maximization of (4.20) is the extra term

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

which is already computed and stored by the first step.

5.3 Fast Algorithms

As states across resolutions are statistically dependent, to determine the optimal states according to the MAP rule, the joint consideration of all the resolutions is necessary. The hierarchical structure of the multiresolution model, however, 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 the 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 in a lower resolution covers a larger region in the original image, making decisions in 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. As a result, the degradation of classification due to the low resolution is avoided. Two fast algorithms are proposed.

5.3.1 Fast Algorithm 1

Use the two resolution case in the previous section as an example again. To maximize approximately

$$\log P\{s_{k,l}^{(r)}, u_{k,l}^{(r)} : r \in \{1,2\}, (k,l) \in \mathbb{N}^{(r)}\} ,$$

the first step of the fast algorithm 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

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

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.

5.3.2 Fast Algorithm 2

The second fast algorithm trains a sequence of single resolution HMMs, each of which is 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, i.e., to 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 differently from Fast Algorithm 1. Suppose $\bar{s}_{k,l}^{(1)}$ is mapped into class "mixed," to

decide $s_{i,j}^{(2)}$, $(i,j) \in \mathbb{D}(k,l)$, we form a neighborhood of (i,j), $\mathbb{B}(i,j)$, which contains $\mathbb{D}(k,l)$ as a subset. We then search 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.

5.4 Comparison of Complexity with 2-D HMM

To show that the multiresolution HMM saves computation for the single resolution HMM, I quantitatively analyze 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 Equation (4.20), that is,

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

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 $w = z$. Every node in the Viterbi transition diagram (Fig. 4.3) 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. Denote the number of blocks on diagonal τ by $n(\tau)$,

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

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 assume

$$\gamma(\tau) = c_1 n(\tau) + c_2 \ .$$

The computation at step τ is thus

$$M^{n(\tau)} M^{n(\tau-1)} \gamma(\tau) \ .$$

The total computation for the Viterbi algorithm is

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

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

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

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

For the multiresolution model, considering the two resolution case, the first step applies the Viterbi algorithm to subimages $\mathbb{D}(k,l)$ to search for $\{\bar{s}^{(2)}_{i,j} : (i,j) \in \mathbb{D}(k,l)\}$ maximizing $\log P\{s^{(2)}_{i,j}, u^{(2)}_{i,j} : (i,j) \in \mathbb{D}(k,l) | \, s^{(1)}_{k,l}\}$. 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 in Resolution 2. The total computation for the first step is then of order $M_2^3 \cdot (w/2)^2 \cdot M_1$. where M_1 is the number of states in Resolution 1. Since the second step applies the

Viterbi algorithm to an image of size $(w/2) \times (w/2)$, the computation for the second step is of order $(w/2)M_1^{w-1}$. Suppose 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 $O(wM^{w-1})$. We see that the multiresolution model reduces the amount of computation by an order of M^w.

Since the computation order increases exponentially with w, the side size of an image, we usually break up the image into subimages with side size w_0 and classify the subimages independently. The computation order for the single resolution HMM is 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 computation 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 computation order is $O(w^2 M^{\max\{w_0-1,4\}})$.

In practice, we use the path-constrained Viterbi algorithm; that is, a certain number of nodes are preselected at each step and paths are constrained to pass through one of those nodes. Suppose N nodes are allowed at each step, the computation is then of order $O(w^2 N^2)$ in both cases. However, the result is closer to the optimal for the multiresolution model.

5.5 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. A 2-D MHMM with three resolutions is assumed. Images in the two low resolutions were obtained by the Daubechies 4 [30] wavelet transform. This transform was chosen because it provides better low resolution images compared with the Haar wavelet and also guarantees

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

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

good localization by sufficiently short wavelet filters. The images at Resolution 1 and 2 are the LL bands of the 2-level and 1-level wavelet transforms respectively. 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 4.10.1. The number of states assigned for each class at each resolution is listed in Table 5.1.

The algorithm was tested by six-fold cross-validation. For each iteration, one image was used as test data and the other five as training data. The average probability of classification error achieved 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. 5.4 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.

The two fast algorithms described in Section 5.3 were also applied based on models with the same number of states listed in Table 5.1. The performance is shown 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.

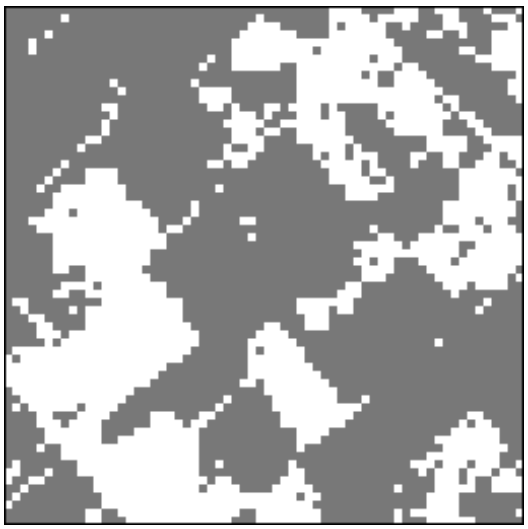


Figure 5.4: The classification result of 2-D MHMM for an aerial image: classification error rate 11.57%. White: man-made, Gray: natural

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.

Experiments were performed on a Pentium Pro 230MHz PC with LINUX operating system. The average user CPU time to classify a 512×512 aerial image is 0.2 second for Fast Algorithm 1, and 7.3 seconds for Fast Algorithm 2. Compared with 200 seconds, the average user CPU time of the single resolution HMM, the speed up is significant. The computation time of Fast Algorithm 1 is very close to that of the CART algorithm, which is 0.16 second on average. In all cases, the classification time provided here does not include the feature computation time, which is a few seconds.

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

Chapter 6

Model Test

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, I examine the validity of the HMMs for images more formally by hypothesis testing in this chapter. Since the main reason for proposing the HMMs is a good balance of model accuracy and complexity, the absolute correctness of the models is less of an issue. The purpose of hypothesis testing is thus more for gaining insight into how improvements can be made rather than arguing for 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 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 I introduce the widely used formalization of hypothesis testing largely due to Neyman and Pearson. Readers are referred to [13] 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 a feature vector is Gaussian distributed given its state. 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 (maximum a posteriori) rule using an estimated model. Feature vectors in each state are then collected as a sample set to verify the normality assumption. 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 4.10.2, which has 5 states for the natural class and 9 states for the man-made class.

The test of normality is based on a 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 follows the normal distribution. For a general Gaussian distribution, a translation followed by a linear transform can generate a random vector obeying 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, non-singular 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 [13]) 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 totally. It is cumbersome to show all the plots. Therefore, details are shown for one state which is representative of the others.

Fig. 6.1(a) provides the normal probability plots for each of the 8 components. Counted row-wise, the 7th and 8th plots in part (a) show typical "bad" fit to the normal distribution for projections onto individual components, whereas the 1st plot is a typical "good" fit. 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. This trend, which owes to a "central limit effect," is 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.

6.3 Test of the Markovian Assumption

For 2-D HMMs, we assume 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 4.4 for details. In this section, I 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

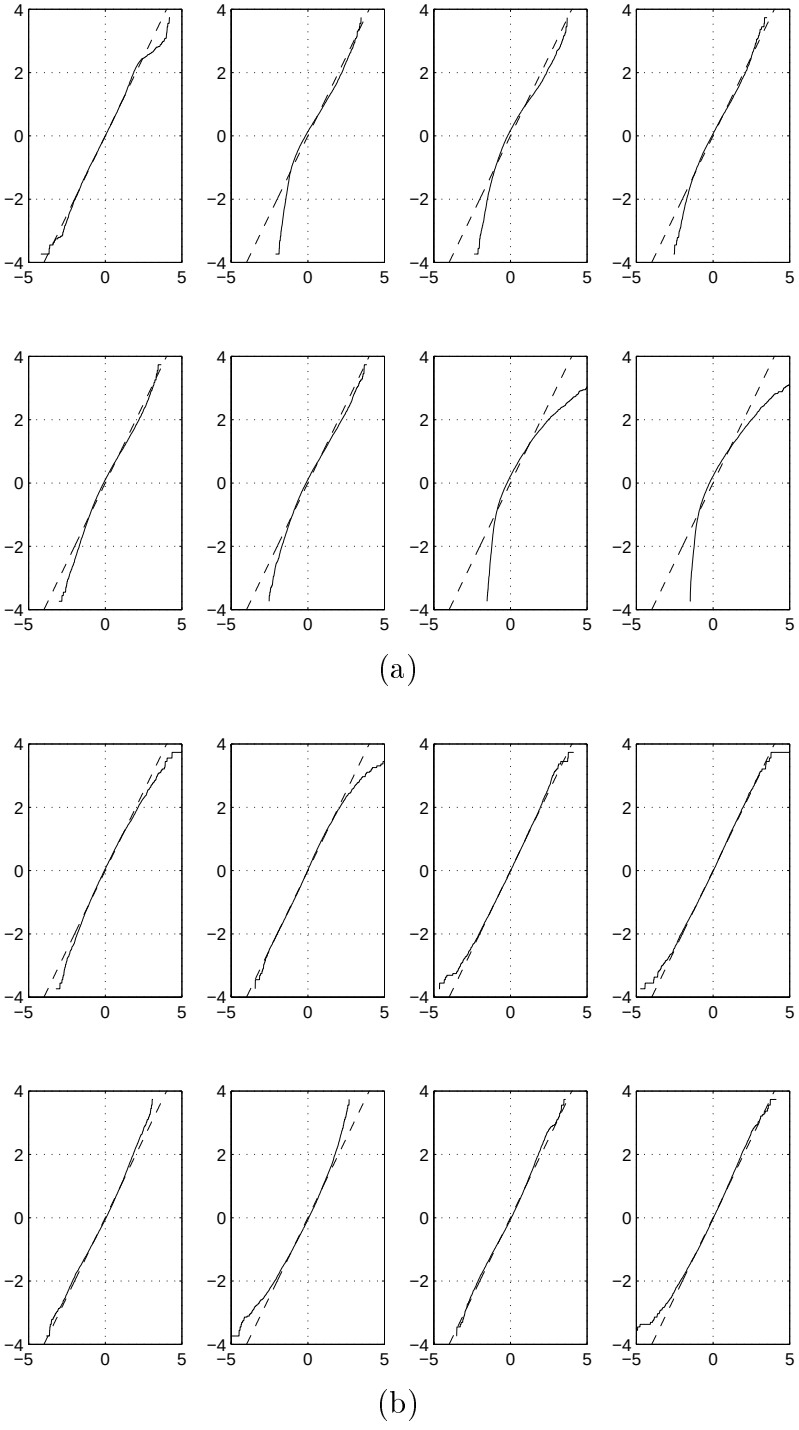


Figure 6.1: Normal probability plots for one state: (a) Each component, (b) The average of all the components and projections onto random directions

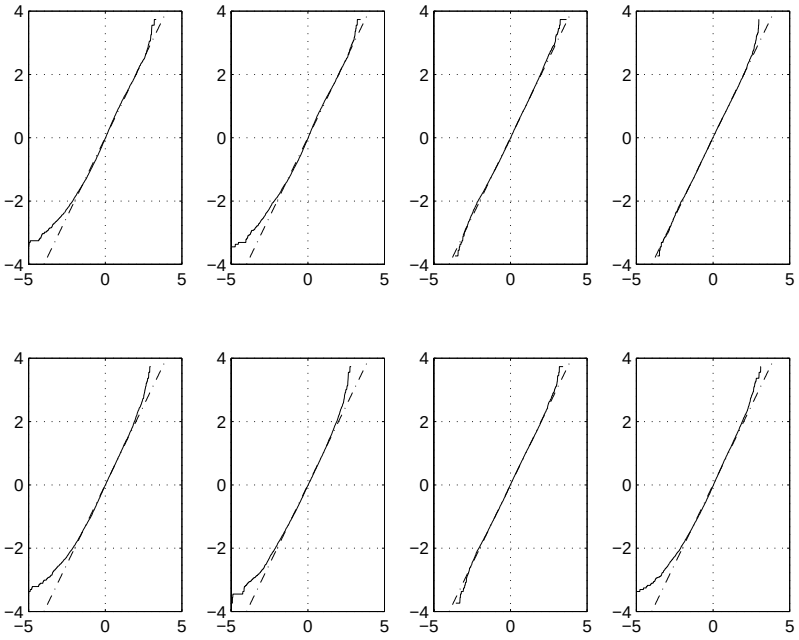


Figure 6.2: Normal probability plots for one state: differences between pairs of components

notational simplicity, we refer to the three cases as Case 1, 2, and 3. Fig. 6.3 shows the three tests. As with the test of normality in the previous section, the test of independence was performed on the aerial image data set. First, states were searched by the MAP rule based on the same model used for the normality test. Then, the test was performed on those states.

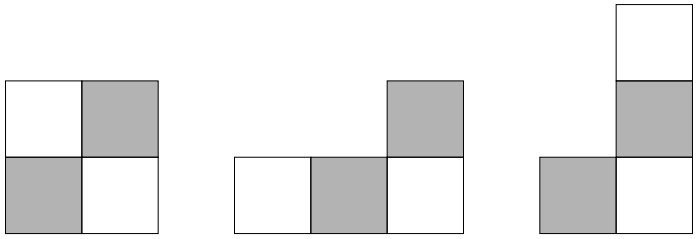


Figure 6.3: Tests of conditional independence. The states of gray blocks are conditional states; the states of white blocks are states upon which tests for independence are performed.

To test the conditional independence, for each fixed pair of conditional states, a

permutation χ^2 test [70, 16] was applied. The idea of a permutation test dates back to Fisher's exact test [44] for independence in a 2×2 contingency table [13]. 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 [82] have taken a network approach to achieve computational feasibility. Boyett [16] proposed 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). 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 marginal counts

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

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 the ones generated by random permutations, we compute Pearson's χ^2 statistic [13],

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

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 model used 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. 6.4. For Case 1, 2, and 3, the median of the p-values is 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.

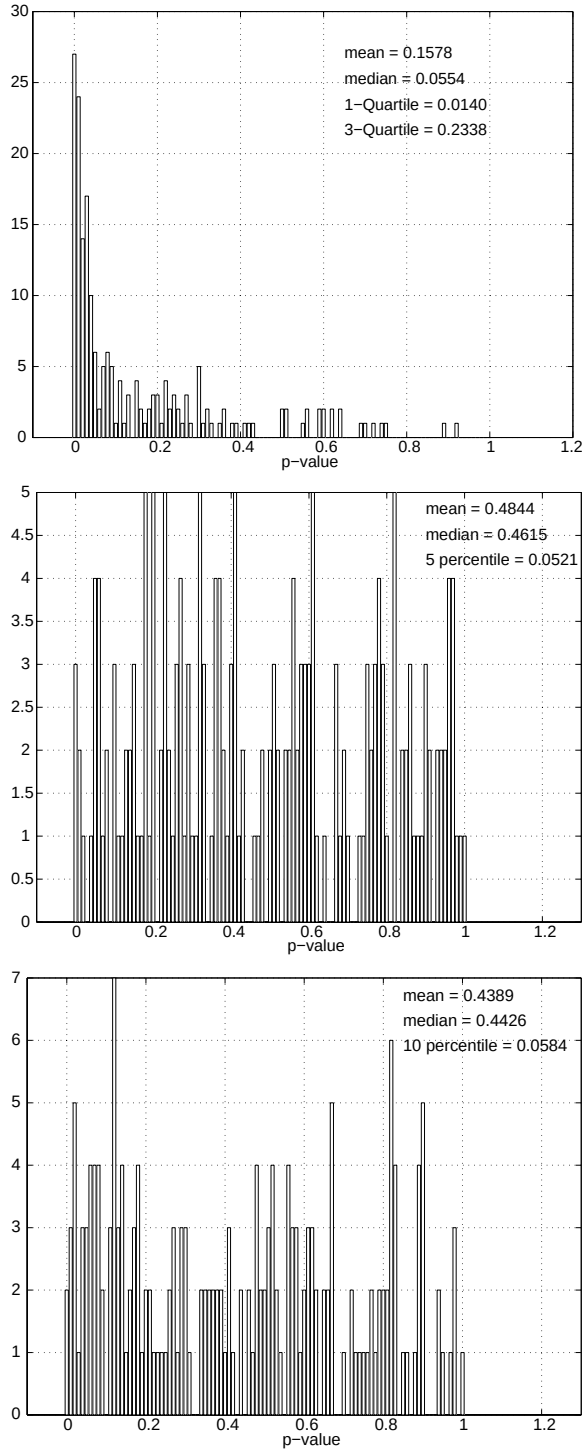


Figure 6.4: Histograms of p-values. Top to bottom: Cases 1 to 3

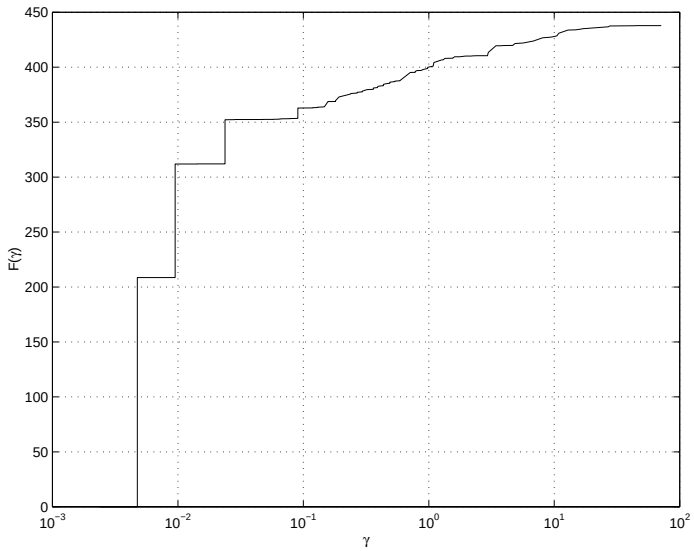


Figure 6.5: $F(\gamma) = \sum_{(i,j):\frac{r_i c_j}{n}<\gamma} \frac{(a_{i,j}-r_i c_j/n)^2}{r_i c_j}$ for a contingency table in Case 1

For 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. 6.5 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, I 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. 6.6. 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 $r_i c_j < n$.

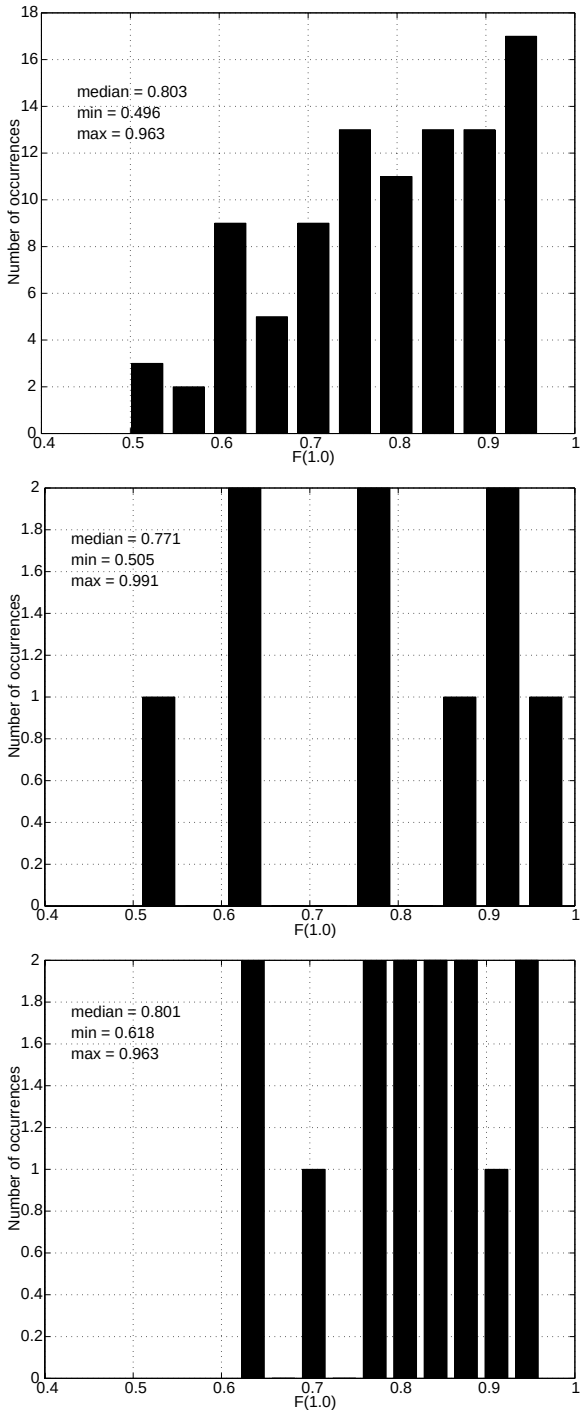


Figure 6.6: Histograms of statistics $F(1.0)$ for contingency tables with p-values below 0.05 in Case 1, 2 and 3. Top to bottom: Cases 1 to 3

Chapter 7

Joint Compression and Classification

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 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 $\{(U_{k,l}, S_{k,l}) : k,l = 1,2,...\}$ is generated by a 2-D HMM.

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

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

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

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

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 posterior probability necessarily yield the classes with the maximum posterior 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 [24, 97]. 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 [55, 94] 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 [55]:

$$\begin{aligned}
J_\lambda(\tilde{\alpha}, \tilde{\beta}, \kappa) &= E(\rho_\lambda) = D(\tilde{\alpha}, \tilde{\beta}) + \lambda B(\tilde{\alpha}, \kappa) ; \\
D(\tilde{\alpha}, \tilde{\beta}) &= E(d(X, \tilde{\beta}(\tilde{\alpha}(X)))) ; \\
B(\tilde{\alpha}, \kappa) &= -\frac{1}{wz} E\left[\max_{s_{k,l}:C(s_{k,l})=\kappa(i_{k,l})} \log(P\{s_{k,l}, u_{k,l} : (k,l) \in \mathbb{N}\})\right] .
\end{aligned}$$

7.2 Optimality Properties and the Algorithm

As with BVQ [55], 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 [55] for the particular case of classification error rate being the penalty for misclassification:

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

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

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

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

  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

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

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 first applied to the training data. Based on the classification, a codebook is designed for each class using the Lloyd algorithm applied to the training data. 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. I 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 [51], 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 m's. 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

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

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

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

Consider

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

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

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

and $\min_{i_{k,l}:\kappa(i_{k,l})=C(s_{k,l})} d(x_{k,l}, \tilde{\beta}(i_{k,l}))$ 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

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

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. 4.2, and $j = 0, 1, ..., w+z-2$. It follows from (4.20) that

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

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

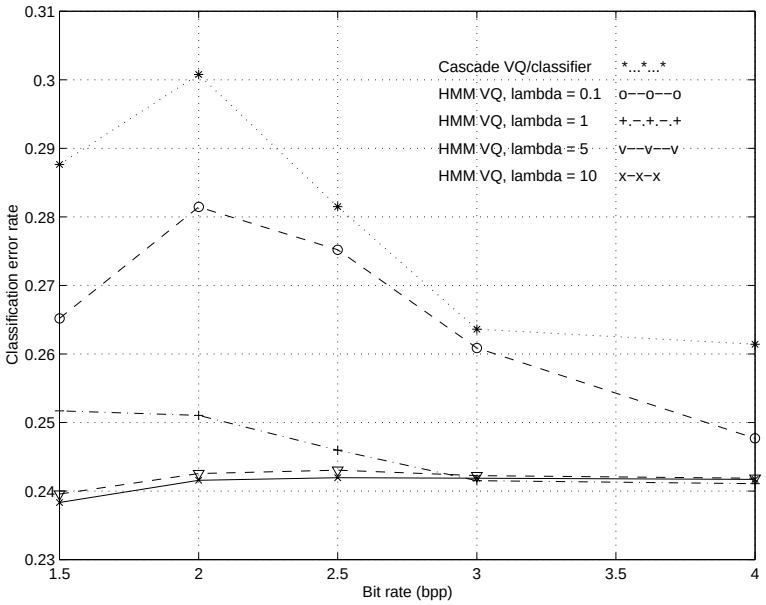
7.5.1 Synthetic Data

The algorithm was simulated on an extended form of the Kohonen Gaussian mixture source [66]. As with an ordinary Kohonen Gaussian mixture, 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, $\sigma_0^2 = 1$,$\sigma_1^2 = 4$. An ordinary Kohonen Gaussian mixture assumes that classes of blocks are iid with equal probabilities for class 0 and 1. I 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

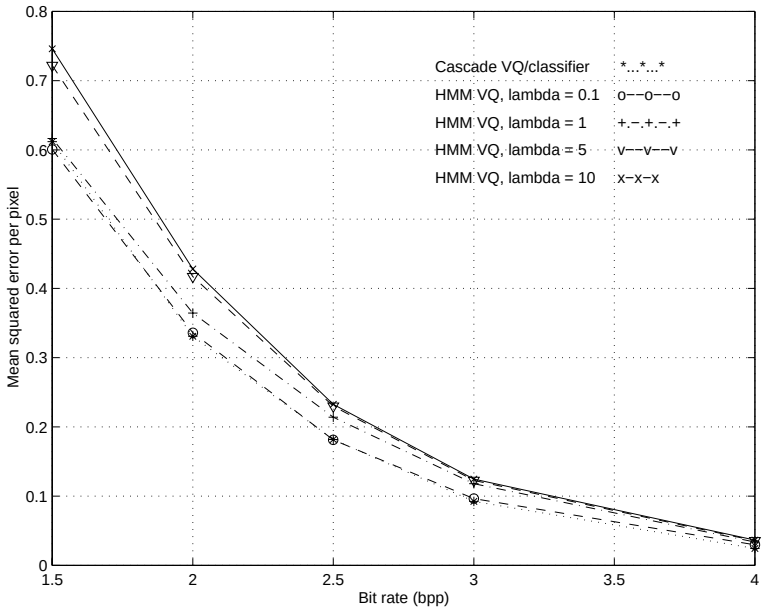
$$\begin{aligned} a_{q,n,r} &= 0.5, \text{ if } q \neq n \\ a_{q,n,r} &= 0.8, \text{ if } q = n = r \\ a_{q,n,r} &= 0.2, \text{ if } q = n \neq r \ , \end{aligned}$$

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 consistent, the probability of remaining in the same state is higher than that of moving to the different state.

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 an ordinary Kohonen Gaussian mixture. The Bayes rule 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 dependency among vectors. The lower bound provided by the Bayes rule is valid only for the performance of classifiers making decisions independently on each block.



(a)



(b)

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

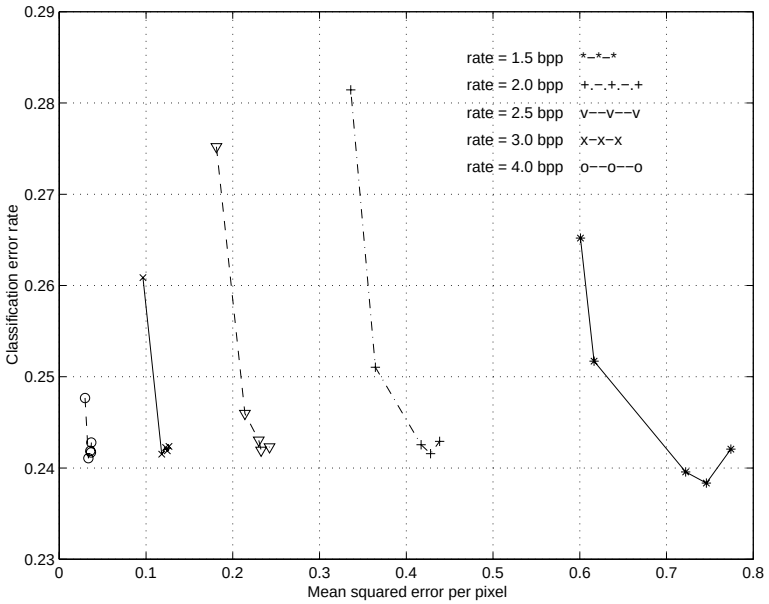


Figure 7.2: Tradeoff between compression and classification with HMM VQ for a Kohonen Gaussian mixture at rates: 1.5, 2.0, 2.5, 3.0, and 4.0 bpp

The compression and classification performance versus bit rates are plotted in Fig. 7.1. Each line in the plots corresponds to a fixed λ, which ranges from 0.1 to 10. Comparison is made against a full search Lloyd vector quantizer followed by a Bayes classifier, which is 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 the 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.

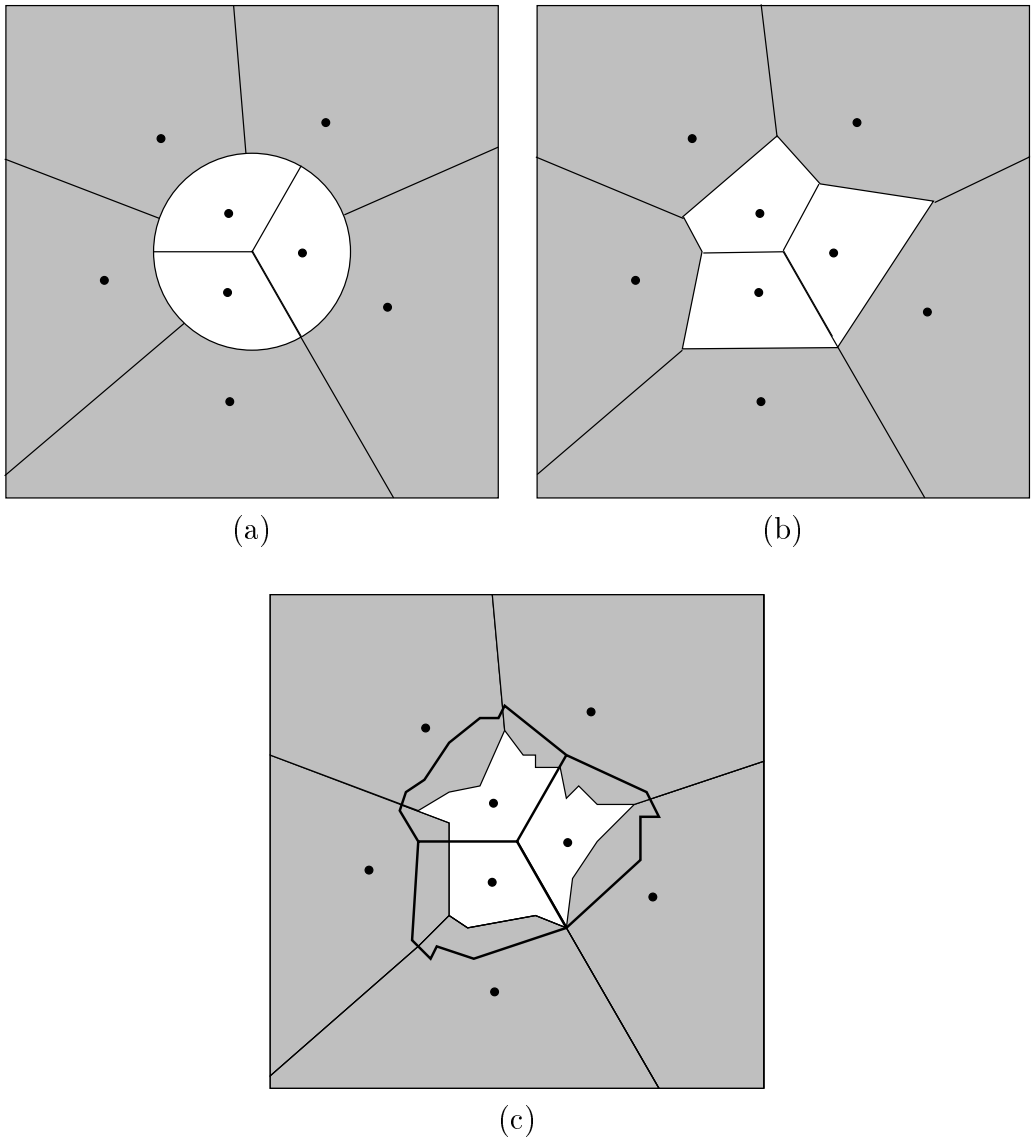


Figure 7.3: Optimal encoders for a Kohonen Gaussian mixture at 1.5 bpp: (a) Bayes classifier followed by Lloyd VQ assuming independent input vectors, (b) Lloyd VQ, (c) HMM VQ, the heavy lines mark the boundaries of the three inner quantization cells, whereas the lighter lines mark the boundaries of the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

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 performance is consistent with comments in [17] 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 the Kohonen 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 decisions, the encoder of HMM VQ has overlapped quantization cells. The boundaries are drawn according to simulation results. 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 [55]. Some of them are listed in Table 7.2. By taking advantage of the inter-block dependencies, 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

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

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

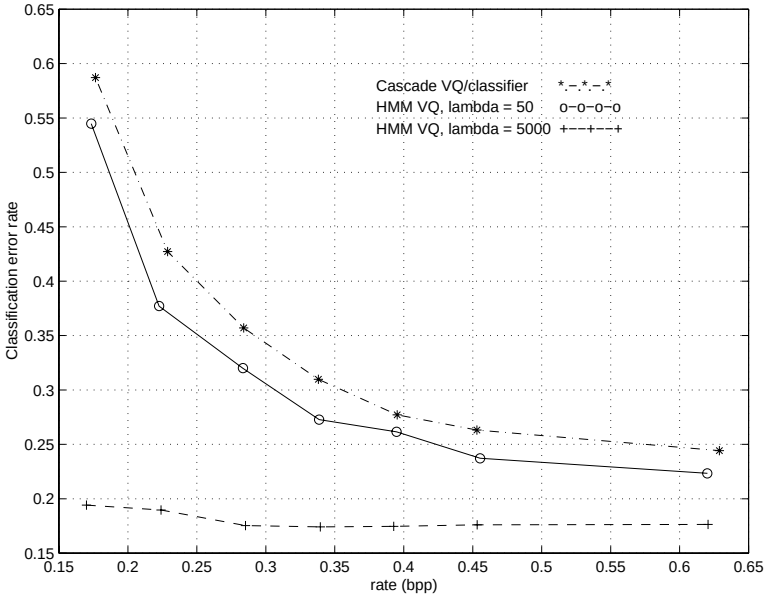
Table 7.3: Classification performance of HMM VQ at 0.338 bpp and $\lambda = 5.0 \times 10^3$

7.5.2 Image Data

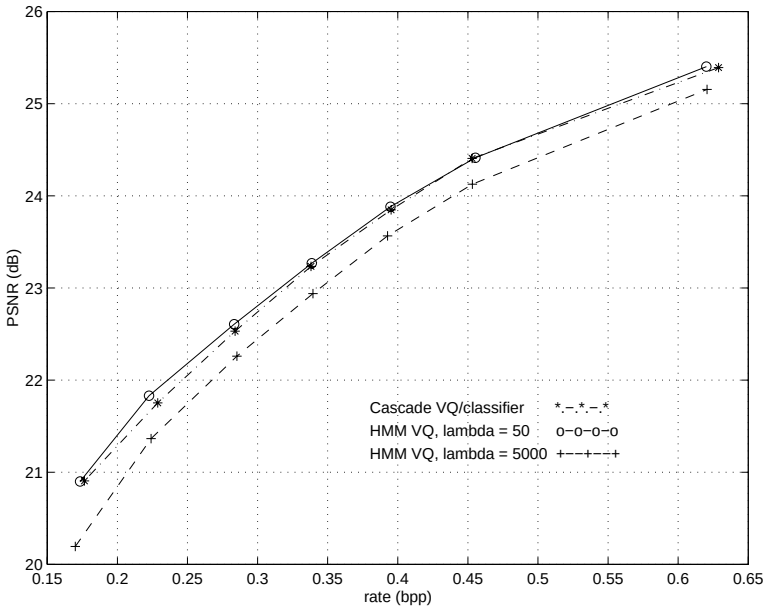
The HMM VQ algorithm was also applied to the data set of aerial images described in Chapter 4. All the results provided were obtained by six-fold cross-validation. Since arithmetic coding was applied to losslessly compress the index output of the quantizer, the rates reported are the rates after arithmetic coding. Experiments show that when a codebook size is small, the entropy coding reduces bit rates by about 10%. Less benefit of entropy coding occurs 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 4.10.1. 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 4.10.2 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.

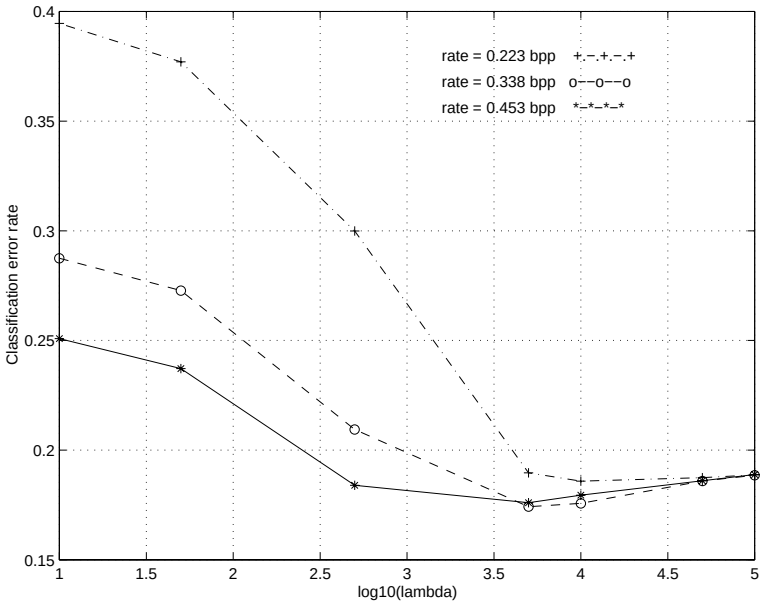


(a)

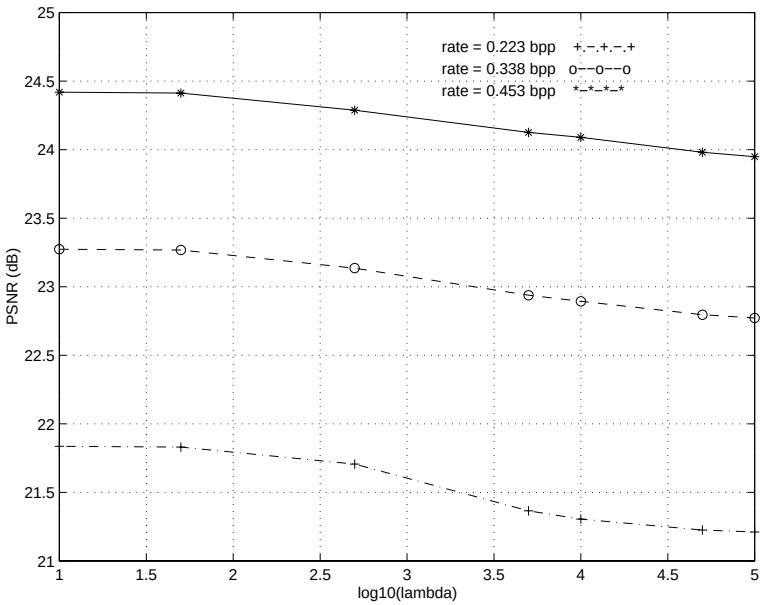


(b)

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)

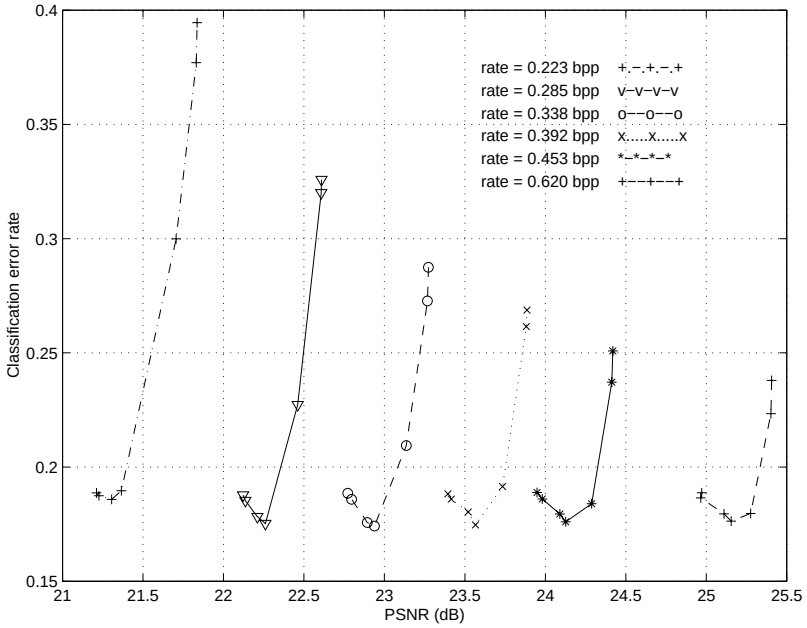


Figure 7.6: Tradeoff between compression and classification with HMM VQ. The λ's are in an increasing order with the increase of PSNR; and $\lambda = 10.0, 50.0, 5.0 \times 10^2, 5.0 \times 10^3, 1.0 \times 10^4, 5.0 \times 10^4, 1.0 \times 10^5$.

To see the effect of λ on compression and classification, I 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 $\lambda = 0$, 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 can 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, I 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

In Chapter 5, the 2-D hidden Markov model is extended to multiresolution. To classify an image based on a multiresolution 2-D HMM, 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 is 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.

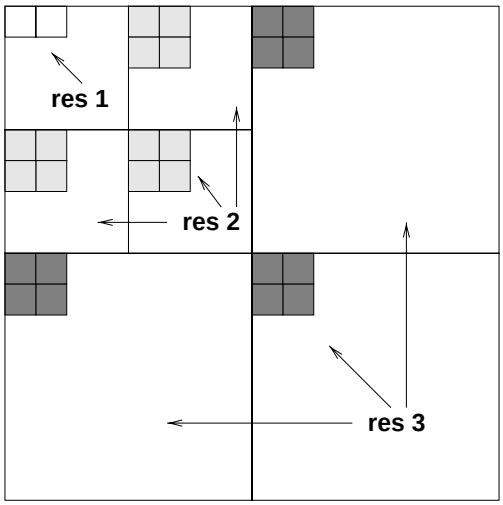


Figure 7.7: Vectors used at the 3 resolutions

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 and Future Work

In this thesis, I 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, I proposed to represent images by 2-D hidden Markov models (HMMs) 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 and backward algorithm for estimating 1-D HMMs efficiently, I 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 (MAP) 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 independently, such as the learning vector quantization and a decision tree algorithm.

For future work on the 2-D HMM classification algorithm, it is of interest to develop better estimation methods. To reduce computational complexity, I started

from the analytic formulas and proceeded through making approximations step by step. Other approaches may 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 may cascade my algorithm and the deterministic relaxation algorithm to obtain better states.

Following hints from the human vision system, I explored ways to segment images in a multiresolution manner. The human vision system can classify images accurately with high speed. Questioning why, I made two observations: the human vision system makes decision with both global and local information; and the vision system distributes effort unevenly by looking at more ambiguous regions in higher resolutions. Since the multiresolution approach enables us to incorporate global and local information in classification, the 2-D HMM is extended to multiresolution.

A multiresolution 2-D HMM represents images by feature vectors at 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. The feature vectors are also statistically dependent across resolutions according to a hierarchical structure. The application to aerial images showed results superior to those of the algorithm based on single resolution HMMs. As the algorithm based on the multiresolution model searches for optimal states across all the resolutions jointly, it is slower than the single resolution algorithm. However, the hierarchical structure of the multiresolution model is naturally suited to progressive classification if we relax the MAP rule. 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.

It is worth applying the algorithms based on multiresolution or single resolution 2-D HMMs to other classification problems. Examples include texture segmentation and character recognition. Since image modeling is for describing images efficiently, it is used for many different purposes. Although multiresolution 2-D HMMs have been developed for image segmentation, in future work I may investigate the application

to other image processing tasks, such as restoration and compression.

As the performance of the classification algorithms depends obviously on the validity of 2-D HMMs, the model assumptions were tested by hypothesis testing techniques. First, I 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, I 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 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. Therefore, to improve a 2-D HMM, future work should assume the state process to be a 3rd order Markov mesh. The consequence of assuming the higher order Markov mesh is a great increase in the number of transition probabilities. With the 3rd order Markov mesh, transition probabilities are indexed by four states, among which three are conditional states. Good estimation of the model requires the reduction of model parameters. One way to solve the problem is to add constraints on the parameters. For example, the symmetry assumption on 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.

In the last part of the thesis, 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. As a result, 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.

I 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 based on 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.

Since the concept of joint compression and classification is potentially important for some applications that require explicit information about content from compressed formats of images, it is worth developing joint algorithms that align well with standard or state of the art compression algorithms. 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 developing more practical schemes.

Appendix A

Histogram Partition

I 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 4.11. From the second condition of the definition for a zone, we see that the two end points for a zone have to yield local minima except for the two zones $[0, t_1]$ and $[t_{r-1}, Z]$, i.e., t_1, ..., 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 minimum point can be an end point of a zone due to the third constraint. The minimum point has to be small enough 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, ..., 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 constraints for a partition, we need to choose a subsequence that forms a partition. This is accomplished

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 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

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 are found, the left end point t_i and the maximum point t_i^* are fixed. The right end point t_{i+1} can 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 distribution density value if there ever exists a replacement. This strategy enables the division between two zones to take place at points with as low distribution density as possible.

To aid 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 false and the algorithm is seeking a maximum point. According to the two flags seek_minimum and pre_is_maximum, the algorithm can be in four states. In table A.1, I list the points that might be set initially or changed at the current state, and the next possible state to enter. The algorithm traverses between states according to Table A.1. Except for determining the first zone $[0, t_1]$, which starts with both seek_minimum and 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.

Bibliography

[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] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

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

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

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

[42] 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.

[43] 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.

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

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

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

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

[53] 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.

[54] 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.

[55] 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.

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

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

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

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

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

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

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

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

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

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

[66] 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.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

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

[73] 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.

[74] 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.

[75] 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.

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

[77] 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.

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

[79] 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.

[80] 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.

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

[82] 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.

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

[84] 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.

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

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

[87] 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.

[88] 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.

[89] 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.

[90] 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.

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

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

[93] 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.

[94] 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.

[95] 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.

[96] 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.

[97] 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.

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

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

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

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

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

[103] 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.

[104] 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.

[105] 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.

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

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

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

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

[110] 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.

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

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

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

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

[115] 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.

[116] 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.

[117] 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.

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

[119] 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.

[120] 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.

[121] 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.

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

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

[124] 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.

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

[126] 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.

[127] 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.