CONJOINT PROBABILISTIC SUBBAND MODELING

by

Ashok Chhabedia Popat

S.B. Electrical Engineering, Massachusetts Institute of Technology, 1986

S.M. Electrical Engineering, Massachusetts Institute of Technology, 1990

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1997

© 1997 Massachusetts Institute of Technology
All rights reserved

Signature of Author: ______________________________

Program in Media Arts and Sciences
August 8, 1997

Certified by: ______________________________

Rosalind W. Picard
NEC Career Development Professor of Computers and Communications
Thesis Supervisor

Accepted by: ______________________________

Stephen A. Benton
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Conjoint Probabilistic Subband Modeling

by

Ashok Chhabedia Popat

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on August 8, 1997, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

A new approach to high-order-conditional probability density estimation is developed, based on a partitioning of conditioning space via decision trees. The technique is applied to image compression, image restoration, and texture synthesis, and the results compared with those obtained by standard mixture density and linear regression models. By applying the technique to subband-domain processing, some evidence is provided to support the following statement: the appropriate tradeoff between spatial and spectral localization in linear preprocessing shifts towards greater spatial localization when subbands are processed in a way that exploits interdependence.

Thesis Supervisor: Rosalind W. Picard
Title: NEC Career Development Professor of Computers and Communications

Doctoral Dissertation Committee

Thesis Supervisor

Rosalind W. Picard
NEC Career Development Professor of Computers and Communications
Program in Media Arts and Sciences

Thesis Reader

Robert M. Gray
Professor and Vice Chair
Department of Electrical Engineering
Stanford University

Thesis Reader

Josef Kittler
Professor of Machine Intelligence
Director of the Center for Vision, Speech, and Signal Processing
University of Surrey

Thesis Reader

Alex Pentland
Academic Head and Toshiba Professor of Media Arts and Sciences
Program in Media Arts and Sciences

ACKNOWLEDGMENTS

I would like to thank Professor Rosalind Picard for her advice, encouragement, support, and patience over the years. She has made an effort to make working in her group enjoyable, and has taught me a great deal along the way.

I would also like to thank the other members of my committee: Professors Robert Gray, Josef Kittler, and Alex Pentland. Professor Gray provided needed guidance and encouragement at several stages of this work. Some of his early comments in a discussion we had in Adelaide in early 1994 picqued my interest in the use of decision trees for density estimation. Both Professor Gray and Professor Kittler traveled great distances to attend my defense; it was a great pleasure and a privilege for me to have had each of them serve on my committee. Professor Pentland's early work and comments about predicting subbands were inspirational, and every conversation I've had with him has been a pleasure and a learning experience.

Many other people helped me greatly along the way, including most of the Vision and Modeling Group and much of the Garden. It is not possible to list all of them here individually; a partial list follows: Professor Ted Adelson, Dr. Rich Baker, Jeff Bernstein, Professor Aaron Bobick, Sumit Basu, Professor Michael Bove, Bill Butera, Lee Campbell, Dr. Ron Christensen, Thunyachate (Bob) Ekvetchavit, Dr. Bill Freeman, Professor Bernd Girod, Anh Viet Ho, Tony Jebara, Professor Michael Jordan, Professor Murat Kunt, Dr. Wei Li, Professor Jae Lim, Fang Liu, Tom Minka, Professor Bruno Olshausen, Professor Al Oppenheim, Ken Russell, Professor William Schreiber, Professor Eero Simoncelli, Dr. Andy Singer, Professor David Staelin, Thad Starner, Dr. Gary Sullivan, Martin Szummer, Dr. John Y. A. Wang, Yair Weiss, Chris Wren, and Professor Ken Zeger.

Most of all, I'd like to thank my wife Rita for being the best part of my life, and for making me a part of hers.

This research was supported in part by Hewlett-Packard Company and NEC Corporation. Some of the expressions used in the software were obtained using MACSYMA, courtesy of the U.S. Department of Energy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1:

Introduction

Generally, there is a theoretical performance advantage in processing data jointly instead of separately. In data compression, for example, the advantage of joint processing results from the ability to exploit dependence among the data, the ability to adapt to the distribution of the data, and the ability to fill space efficiently with representative points [45, 71].

It is generally much easier and often more natural to process data sequentially than jointly. It is of interest, therefore, to find ways of making the performance of sequential processing approach that of joint processing to the extent possible. There are some situations in which sequential processing does not incur a large performance penalty, for instance when compressing independent data. This case is of limited interest however, as real information sources are seldom usefully modeled as independent. A more interesting example is compression of data that is not required to be independent. In lossless compression, a message is to be represented in compact form with no loss of information. It is well known that the best lossless encoding will produce on average a number of bits about equal to the minus log of the joint probability that has been assigned to the message [124]. However, it is difficult to work with the set of all messages, even if we restrict consideration to those of a fixed moderate length. By simply writing the joint probability as a product of conditional probabilities of the individual letters, we can achieve the same compaction sequentially by using the optimum number of bits for each letter. The order in which we encode the letters can be arbitrary, though often a natural ordering is suggested by the application.

The key to effective sequential lossless data compression lies in finding the right conditional probabilities to use. This generally requires that some structure or model be assumed, and that the probabilities be estimated from the data with respect to that model. Thus, the problem of lossless compression is really one of modeling the sequence of conditional probability distributions.

It is natural to seek to generalize this strategy to applications besides lossless data compression, hopefully in a way that likewise approximates the performance of joint processing. This thesis proposes that the generalization be accomplished through the use of high-order conditional probabilistic modeling.

1.1 Conjoint Processing

We use the term *conjoint processing* to refer to the sequential processing of data according to a corresponding sequence of conditional probability laws chosen such that their product approximates the joint probability law. This approach to processing information has been advocated by Rissanen [108, 111]. Our choice of term is somewhat arbitrary, but it is convenient to have a short way of describing this style of processing. The dictionary definition of "conjoint" is not much different from that of "joint;" both can be taken in our context to mean roughly "taking the other values into account." Since "joint" is already claimed, we instead appropriate the term "conjoint" for our purpose. A related term coined by Dawid [30] is *prequential* (for *pre*dictive se*quential*), but it has come to be used in a relatively narrow statistical sense, rather than to describe a general style of processing information.

There are many areas in which conjoint processing is currently being employed, though not always to the full potential afforded by its framework. We have already considered lossless compression. What would be required to extend the technique effectively to lossy compression, where the message need not be reconstructed perfectly? In most applications of interest, we would need to be able to model and estimate conditional probability laws accurately when the data are continuous or well approximated as continuous. If the lossless case is any indication, the accuracy of such modeling will have great impact on the performance achieved by a conjoint lossy compressor. To begin to take full advantage of the conjoint processing framework then in a general setting, we must be able to model conditional densities accurately.

This thesis proposes a technique for accurately modeling conditional densities (Chapter 3). The technique combines aspects of decision trees and mixture densities to achieve accurate modeling by appropriately partitioning conditioning space, while exploiting the known smoothness of the underlying probability law. An important feature is that it adjusts its complexity automatically to an appropriate level for the data, avoiding overfitting while still allowing sufficient flexibility to model the data effectively.

Conjoint processing can be used successfully in applications other than data compression. This is shown in Chapter 4, where it is applied to texture synthesis and image restoration in addition to compression. In the texture synthesis application, it is found that conjoint processing can result in synthesized textures that have characteristics more realistic than those obtained using standard

Figure 1.2.1: Separable 3×3 uniform-band subband decomposition of the luminance component of the standard 720×576 test image *Barbara*. A jointly localizing, quasi-perfect-reconstruction, critically sampled analysis filter bank designed using the method described in [92] was used. The non-DC bands have been shifted and scaled in amplitude for display. Strong spatial and spectral interdependence of subband pixels is evident, yet the empirical correlation is nearly zero among pairs of non-DC pixels for a variety of displacements, both within and across subbands. It may be concluded that there is a great deal of nonlinear dependence.

techniques. It is also shown to perform effectively in image restoration because of its ability to model nonlinear relationships over relatively broad spatial areas.

1.2 Independence in Image Subbands

In *subband coding*, an image is broken up into several frequency bands called *subbands*, after which each is subsampled spatially. The subsampled subbands are then lossy-compressed as if they were independent — that is, by using techniques which are known to work well under the assumption that the data are independent. After decoding, the reconstructed subbands are pieced back together to form an approximation to the original image. Subband coding will be described more fully in Section 2.1, and an explanation will be given there for why the subband pixels are usually modeled as approximately independent, or more precisely, uncorrelated.

In practice, pixels in subbands formed from an image of a natural scene are far from independent. An example is shown in Figure 1.2.1. Recently, there has been growing interest in extending the traditional subband coding approach to take advantage of this dependence. One approach is to group subband pixels into disjoint sets and to quantize the pixels in each set jointly, ignoring any interdependence between sets. This approach to subband coding is an active area of research; see the recent paper by Cosman et al. [25] for an extensive survey. Also described there are techniques that we might call conjoint, except that, as far as I am aware, the conditional distribution is either modeled only implicitly as in [126], or explicitly but using fairly restrictive models as in [73]. It would be interesting to see whether an explicit, powerful conditional density model such as that described in Chapter 3 might be used advantageously in such systems. This question is considered in Chapter 5.

The purpose of the subband decomposition, which is clear when independent compression of the subbands pixels is employed (see Section 2.1), is less clear when joint or conjoint compression is allowed. The question of what filter characteristics are best to use in this case must be answered on the basis of practical rather than theoretical considerations. This is because most subband decompositions that one would normally consider for use today are reversible or nearly so, implying that all are equally good (or in this case, equally without effect) when unconstrained joint compression is allowed. When restrictions are placed on the complexity and dimensionality of the processing, as they must be in practice, the problem becomes one of joint optimization of the subband decomposition and the characteristics of the subsequent processing. Such an optimization can be feasible if the system is sufficiently constrained; for example, Brahmanandam et al. [11] have considered jointly optimizing an affine block transform and a vector quantizer. A more general optimization procedure for a fixed encoding scheme might be to select a structure for the subband decomposition, then search over filters to use in that structure that achieve varying levels of tradeoff between spatial and spectral localization. Such a search is also considered in Chapter 5. Evidence is presented there to support the conjecture that in some applications, preprocessing filters that achieve greater spatial localization are required when joint or conjoint processing is used than are required when independent processing is used.

CHAPTER 2:

Background

This chapter provides background information relevant to subband-domain processing and density estimation.

2.1 Subband-Domain Processing

Subband-domain processing is a generic term given to a family of procedures that operate by dividing the frequency spectrum into approximately disjoint regions called *subbands*, processing each, and combining the results to get a desired output. The division into subbands is called *analysis* and the recombination is called *synthesis*. A good general reference on the subject is the textbook by Vetterli and Kovacevic [138]. We shall briefly summarize the essentials, explain why subband pixels tend to be approximately uncorrelated, and summarize how subband decompositions have traditionally been used in image compression.

Critically Sampled Filter Banks

The basic building-block in analysis and synthesis is the *critically sampled filter bank*, a special case of which is shown in Figure 2.1.1. It is a special case because in general the subsampling factors need not be equal; however, the sum of their reciprocals must equal unity if they are to be critically sampled. Critical sampling refers to retaining the minimum number of samples necessary to reconstruct the filter outputs under the assumption that the filters are perfectly frequency-selective. If the output is equal to the input in the event of no intermediate corruption of the subbands, then the filter bank is said to have the *perfect-reconstruction* property. The filters need not be perfectly frequency-selective in order for the perfect-reconstruction property to hold. Henceforth, all of the filter banks considered will be assumed to have the perfect-reconstruction property.

The subsampling operations indicated by the blocks labeled $\downarrow K$ involve discarding all samples except every K^{th} one. To describe this operation precisely, it is necessary to specify a subsampling phase, which indicates the offset of the retained sample indices relative to the multiples of K. The upsampling operations denoted by $\uparrow K$ replace the retained samples into their original sequence positions, filling in zero for the other samples. It is important to realize that the subsampling

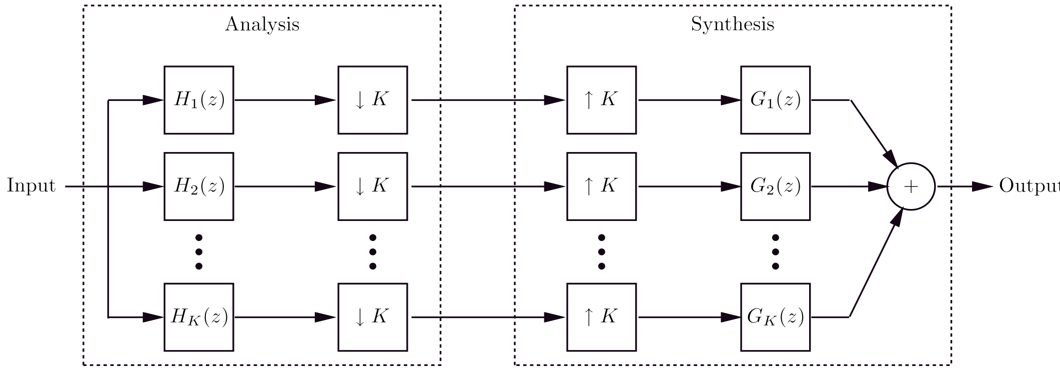

Figure 2.1.1: A critically sampled filter bank, consisting of an analysis and a synthesis stage. The system shown is for processing a one-dimensional signal; it may be applied to images by applying it in the horizontal and vertical directions separately.

phases can be and often are different for every band. A trivial perfect-reconstruction critically sampled filter bank can be obtained by taking every filter to be an identity operation and the subsampling phases to be a permutation of $(0, \ldots, K-1)$. Henceforth, *subband* will refer to the filtered signal after subsampling.

The foregoing description assumes one-dimensional signals. The structure can be extended to two or more dimensions in a general way. A simpler alternative, which is standard practice in image subband coding, is to perform two-dimensional subband analysis and synthesis by performing the one-dimensional operations separately in the horizontal and vertical directions. Two-dimensional critically sampled filter banks that can be implemented in this way are termed *separable*. Consideration will be restricted to separable critically sampled filter banks throughout this thesis.

Often, an unequal division of the frequency spectrum is desired. One can be obtained by using a more general filter bank with an appropriate choice of filters and subsampling factors. However, here we restrict consideration to uniform-bandwidth critically sampled filter banks. Using these, an unequal spectral division can be obtained by recursively performing subband analysis on selected subbands, resulting in a tree-structured decomposition. Often, only the lowest-frequency subband is recursively subdivided. In this case, when the analysis is carried out separably in the vertical and horizontal dimensions, the decomposition is termed a *subband pyramid*, or just a *pyramid* (see Figure 2.1.2). Such a structure can also be used to implement certain discrete wavelet transforms [138]. We will restrict consideration to pyramids obtained by using the same two-band critically sampled filter bank to carry out all of the analyses. Moreover, the filter banks considered will be orthogonal in the sense defined by Simoncelli and Adelson [131].

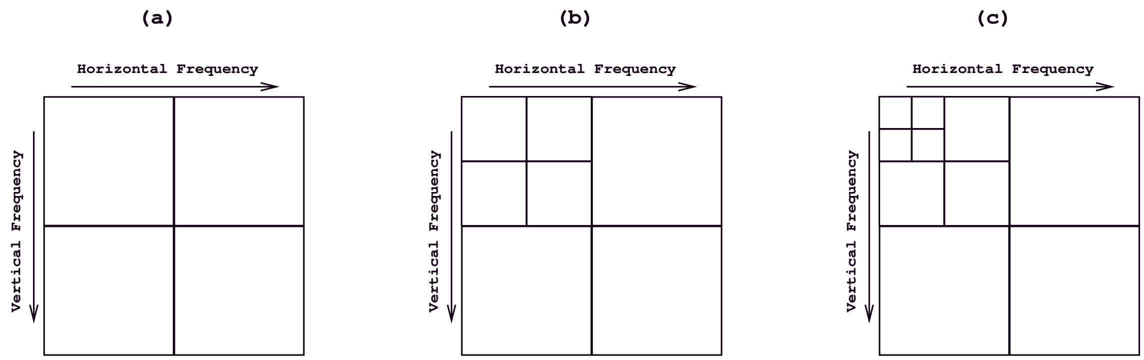

Figure 2.1.2: Construction of a three-level subband pyramid. An image is analyzed into four subbands as shown in (a) by applying a two-band critically sampled filter bank separately in the horizontal and vertical directions. These four subbands constitute the first level of the pyramid. The analysis is then repeated on the low-horizontal, low-vertical frequency subband, resulting in the two-level pyramidal decomposition shown in (b). Repeating the analysis once more on the new low-horizontal, low-vertical frequency subband results in the three-level pyramidal decomposition shown in (c). Because all of the subbands are critically sampled, the total number of pixels in the pyramidal representation is equal to that in the original image.

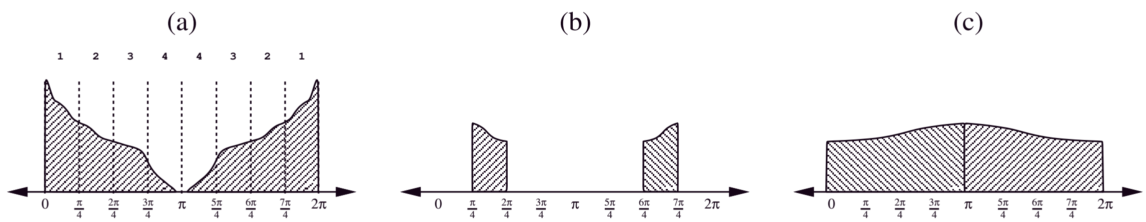

Figure 2.1.3: A spectral-domain illustration of within-subband uncorrelatedness. A hypothetical signal having the lowpass magnitude spectrum shown in (a) is input to an ideal four-band critically sampled filter bank. The spectrum of the output of $H_2(z)$ before subsampling is shown in (b), while that after subsampling (scaled in amplitude by a factor of four) is shown in (c). Note that after subsampling, the spectrum is relatively flat, implying that pixels within a given subband are approximately uncorrelated. Assuming a smooth input spectrum, flatness increases as the filters get narrower, which in a uniform filter bank corresponds to increasing the number of bands. For natural images, the assumption of smooth spectrum is grossly violated in the lowest-frequency (DC) band, so that the DC band in general cannot usefully be modeled as uncorrelated.

Uncorrelatedness of Subband Pixels

An important property of subband representations is that subband pixels for natural images tend to be approximately uncorrelated, both within and across subbands. Uncorrelatedness across subbands can be understood by viewing the subband decomposition as a generalization of an energy-compacting orthogonal block transform, where the basis vectors are allowed to extend beyond the block boundaries. For an orthogonal block transform, diagonalization of the covariance matrix occurs as a by-product of maximizing energy compaction subject to the constraint of constant total energy [62]. Therefore, to the extent that an orthogonal subband decomposition achieves energy compaction, pixels are uncorrelated *across* subbands.

Approximate uncorrelatedness *within* subbands can be easily understood in the frequency domain. The spectra of natural images are often modeled as smooth, except at near zero-frequency (DC) where there tends to be a strong peak. Assume that the analysis filters have nearly ideal narrow brickwall frequency responses. Then for any subband except the one containing DC, the spectrum of the output of the analysis filter will be approximately flat within the passband and zero outside the passband, as shown in Figure 2.1.3 (b). To show that the correlation of pixels in the subband is nearly zero, it is sufficient to show that after subsampling, the magnitude spectrum is nearly flat across the whole spectrum [84]. Let $u[n]$ denote the output of the analysis filter and let $v[m]$ denote the result of subsampling $u[n]$ by a factor of K, so that

$$v[m] = u[Km].$$

For simplicity, a subsampling phase of zero has been assumed; a nonzero subsampling phase would result in an identical magnitude spectrum. By using the fact [114] that

$$\sum_{k=0}^{K-1} e^{j\frac{2\pi}{K}kn} = \begin{cases} K & \text{if } n \equiv 0 (\text{mod} K); \\ 0 & \text{otherwise,} \end{cases}$$

we can express the z-transform of $v[m]$ in terms of that of $u[n]$ as

$$V(z) = \frac{1}{K} \sum_{k=0}^{K-1} U(e^{-j\frac{2\pi}{K}k} z^{1/K}),$$

which corresponds to a discrete-time Fourier transform of

$$V(\omega) = \frac{1}{K} \sum_{k=0}^{K-1} U([\omega - 2\pi k]/K).$$

In this way, the filter passband is mapped onto the entire spectrum $[-\pi, \pi]$, which under the stated assumptions results in an approximately flat spectrum as indicated in Figure 2.1.3 (c). Thus, pixels within the subbands are approximately uncorrelated.

The approximate uncorrelatedness within and across subbands does not imply approximate independence of the subband pixels, as previously noted in Chapter 1.

Subband coding of images

Subband and subband-pyramid decomposition are often used in lossy compression. Subband-based lossy compression is termed *subband coding.* Subband coding was first applied to speech [62] in the seventies and extended to images in the mid-eighties [143]. Much of the underlying motivation and theory carries over directly from an older technique called *transform coding* [56]. As mentioned above, a block transform is a special case of subband decomposition. The basic idea behind image subband coding is now reviewed briefly.

The analysis of an image into subbands by a critically sampled filter bank results in an exact representation of that image with no reduction in the total number of pixels. Although no compression occurs in the decomposition, it is an empirical fact that for natural images the subbands tend to be easier to compress than the original image. In particular, a subband decomposition generally results in a predictably uneven distribution of energy among the subband pixels, so that the available code bits can be allocated among the subband pixels in accordance with how much they are needed. In addition, the subband pixels are nearly uncorrelated as discussed above, so that they can be quantized independently using scalar quantization with reasonable efficiency.

Typically, subband pixels are modeled as having a variance that changes slowly over space and spatial frequency. Suppose that the subband pixels have been grouped into subsets, each characterized by a local variance σ_i^2. The pixels in each subset are typically taken to be nearby one another in both space and spatial frequency. Assuming independent scalar quantization, high rate, small quantization bins, and smooth densities, the mean-square per-pixel quantization error for each subset can be approximated as $\epsilon\sigma_i^2 2^{-2R_i}$, where R_i is the average rate allocated to subset i and ϵ is a performance factor which depends on both the type of quantizer and on the marginal distribution of the pixels [62]. If the variances are large enough so that no negative rate allocations would result, then the method of Lagrange multipliers can be used to derive the asymptotically (high rate) optimal rate allocation

$$R_i = R_0 + \frac{1}{2}\log_2 \sigma_i^2, \tag{2.1.4}$$

where R_0 is a constant selected to meet the constraint on the total available rate. In this case, the improvement in mean-square error (MSE) over quantizing the original pixels can be approximated by the ratio of the arithmetic to the geometric mean of the variances [62]. Neither (2.1.4) nor this formula for the improvement is valid in the important case where one of more of the variances are nearly zero, as this would result in negative rate allocations which would violate the assumption behind the use of Lagrange multipliers. In such situations, an optimal iterative rate allocation procedure can be used instead [58, 127], but a simple formula for the improvement in MSE no longer exists in general.

Alternatively, rate can be allocated implicitly by entropy-coded uniform-threshold scalar quantization (see Section 2.3). In this case the rate allocated to each subband is determined by the activity of the subband pixels and by the stepsize of the quantizer. Using the *same* stepsize for all subband pixels can be shown to result in a nearly optimal implicit rate allocation with respect to mean-square error [42, 93]. A frequency-weighted mean-square error criterion can be easily accommodated by varying the stepsizes across subbands, as is implicitly done in JPEG [101]. Perceptually optimized rate allocation has been considered by Jayant et al. [61] and Wu and Gersho [144].

As mentioned previously, subband pixels exhibit a great deal of statistical dependence despite their near-uncorrelatedness. This dependence may be exploited for coding gain in a number of ways [25]. In Chapter 5, we will consider exploiting the dependence by conjoint processing; specifically, by sequential scalar quantization with conditional entropy coding.

2.2 Density Estimation

This section introduces ideas and notation related to probabilistic modeling in general and to density estimation in particular.

Heuristic Introduction

In general terms, the problem of density estimation is to infer the probability law hypothesized to govern a random object on the basis of a finite sample of outcomes of that random object as well as on any available prior information. The term "random object" will be defined later in this section. Density estimation is an example of a problem in inductive inference, and inherits all of the well-known foundational difficulties attendant to that class of problem [23, 105, 136, 139]. In particular, *induction is not deduction:* it is not possible on the basis of observing a finite sample to show analytically that a particular density estimate is correct or even reasonable, unless strong assumptions are made that reach beyond the information present in the available data. The following thought experiment, presented with a sampling of the more popular ways of approaching it, will help to clarify this point.

Imagine being presented with a large opaque box, perhaps the size of a refrigerator, and told that it contains a large number of marbles of various colors. You reach your hand in and pull out ten marbles, one at a time (but without replacement). Suppose that the first nine are red and the tenth is blue. Based on this sequence of observations, you wish to infer the proportion of every color represented among the remaining marbles in the box. This is essentially the density estimation problem in the discrete case. (A continuous version, discussed later in this section, might involve the weights of the marbles instead of the colors.)

A starting point would be to assume that the order in which the marbles are observed is irrelevant. This assumption of *independence* (or *exchangeability* in Bayesian parlance [43]) is reasonable if the box had been shaken before you reached in, and if the number of marbles in it is in fact very large. Relaxing this assumption will be considered later.

At this point there are several ways in which you might proceed, and none can be shown analytically to be superior to another without making additional assumptions about the contents of the box. Fisher's celebrated maximum likelihood (ML) principle gives the proportions as 9/10 for red, 1/10 for blue, and zero for any other color. To be useful in practice, this principle must usually be modified; often this is accomplished by adding a penalty term [129, 130]. Laplace's rule of succession [85] requires that you know the possible colors beforehand; if there are (say) five of them, then his rule results in the estimated proportions 2/3 for red, 2/15 for blue, and 1/15 for each of the three remaining unseen colors. The *minimum-description length* (MDL) principle chooses the proportions that result in the shortest possible exact coded representation of the observed data, using an agreed upon prefix-free code for the empirical distribution itself [3, 109]. See [137], p. 106 for an interesting criticism of this approach. In this particular example, again assuming five possible colors, Laplace's rule and MDL happen to agree on the estimated proportions when the prior probability of each empirical distribution is taken to be proportional to the number of possible sequences of marbles that support that distribution;[1] that is,

$$\hat{P}(\hat{N}_{\text{red}}, \hat{N}_{\text{blue}}, \ldots) = \frac{10!}{5^{10}(\hat{N}_{\text{red}})!(\hat{N}_{\text{blue}})!\ldots},$$

where the $\hat{N}$'s are constrained to be integers between 0 and 10 inclusively. (It is easiest to demonstrate that Laplace's rule and MDL give the same result in this case by computer simulation; the optimized description length turns out to be about 20.6 bits.) Both in this case and typically, the MDL principle functions by initially "lying" about the empirical proportions, then subsequently correcting the lie when the observed sequence of ten marbles is specified exactly using an ideal (with respect to the lie) Shannon code. This particular version of MDL assumes a simple two-part model/message lossless encoding of the observed data, and corresponds to Rissanen's earliest work on MDL [106]. His subsequent work generalizes the MDL principle considerably, removing the strict model/data separation, and allowing variable precision representation of the model param-

[1] In other words, proportional to the cardinality of the type class [26].

eters [110]. It should be emphasized that the immediate goal of the MDL principle is inference, not data compression: the hope is that the lie turns out to be a more accurate description of the remaining contents of the box than the truth.

Another point of view is the Bayesian; it is quite different in its motivation from those mentioned above, and involves the following sort of reasoning. You believe that the box was carried up from the basement, where you know you had stored two boxes: one containing an equal number of red and blue marbles, and the other containing only red. Since a blue marble was among those observed, you reason that the proportions must be 1/2 red and 1/2 blue. In a more fully Bayesian setting, the two boxes would have possibly unequal chances of being picked, but this is a minor point. The main Bayesian leap (from Fisher's point of view, not Rissanen's) is to view the proportions themselves (or more generally, *anything* that isn't known with certainty) as being described by a probability distribution. Rissanen uses a prefix-free code, which amounts to the same thing via the Kraft inequality [41].

It is even possible to argue coherently for estimating a lower proportion of red than blue. A rule that does so is termed *counterinductive* [136]. One way to justify such an estimate would be on Bayesian grounds, using a suitable prior. Another justification, valid in the case of a small total marble population, would be on the grounds that more red marbles than blue have been used up by the observation sequence. Misapplying this reasoning to the case where the sampling is done with replacement, or where the population is infinite, leads to the well known gamblers' fallacy of betting on an infrequently observed event because "it is due to occur."

The main reason for presenting the foregoing brief survey of the differing major views is to emphasize that there really is no generally agreed upon best way to perform density estimation, and also to underscore the crucial role played by assumptions and prior information.

Assumptions become even more important in the continuous case. Suppose now that the distribution of the *weights* of the marbles, rather than their colors, is in question, and that all of the ten marbles drawn differ in weight slightly. The added difficulty now is that there is no natural *a priori* grouping of events into like occurrences; instead, one must be assumed. Usually, this is done implicitly in the form of a statement like, "if one-ounce marbles are frequent, then marbles weighing close to an ounce must also be frequent." A statement of this type amounts to a *smoothness assumption* on the density, with respect to an adopted distance measure in the observation space.

So far in this example, we have considered only a single characteristic of the marbles at a time; that is, we have considered only *scalar* density estimation. An example of multivariate or *joint* density estimation would be when not only the weight of the marbles, but also their size and perhaps other characteristics, are brought into question. These characteristics may of course interact, so that each combination of characteristics must be treated as a different occurrence (unless and until it is found out otherwise). The added complication here is quantitative rather than qualitative: as the number of characteristics or *dimensions* grows, the number of possible combinations grows exponentially, as does the requisite number of observations required to obtain a reliable density estimate. In the continuous case, it is the volume of space that grows exponentially rather than the number of possible combinations, but the effect is the same. The general term used to refer to difficulties ensuing from this exponential growth is *the curse of dimensionality* (see [120], pp. 196–206 for a good discussion).

A similar problem plagues *conditional density estimation*, which refers to the estimation of proportions within a subpopulation satisfying a specified condition. An example would be estimating the proportion of marbles that are about an inch in diameter among the shiny ones that weigh about an ounce. Since the number of possible conditioning classes grows exponentially in the number of characteristics conditioned on, exponentially more observations are likely to be required before a reliable conditional estimate can be obtained.

A final complication arises when the order in which the marbles are observed is considered to be significant; that is, when successive observations are regarded as dependent. In such cases, it is natural to view the sequence of observations as constituting a *sample path* of a *random process* [48, 84]. In this framework, the marbles are assumed always to come out of a given box in the same fixed sequence, but the box itself is assumed to have been selected at random from a possibly infinite collection or *ensemble* of boxes, according to some probability distribution. This is not the same as the Bayesian point of view described above, since in this case there is only one level of randomness. Since only a single box is observed, it is necessary that certain assumptions be made so that the observations over the observed sequence can be related to statistics across boxes. The chain of inference must proceed from the specific observed initial subsequence, to the ensemble statistics across boxes for the subsequence, to the ensemble statistics for the whole sequence, and finally to the statistics of the remainder of the sequence in the particular box at hand. To enable such a chain of inference, strong assumptions must be made about the random process. At the very least, it must be assumed that the relevant statistics across boxes can be learned by looking at the contents of only a single box; roughly speaking, this is to assume that the process has the *ergodic property* [45]. In addition, in practice some sort of assumption regarding stationarity (invariance of statistics to shifts in time) and the allowed type of interdependence among observations must be conceded. Many specialized processes, each defined by its own set of assumptions, have been proposed over the years; of these, the most useful have been ones in which dependence is limited and localized. The simplest is the stationary *Markov* process of order m, in which the conditional density of each observation given the infinite past is stipulated to equal that given the past m observations, and, moreover, this conditional density is assumed unchanging over the entire sequence. Such a model is readily generalized to two-dimensional sequences by the notion of *Markov random field*, which has been used extensively in texture and image processing [60]. Relaxation of strict stationarity can be introduced in a controlled and extremely useful manner via the concept of the *hidden Markov* source [102] and its various generalizations, which have found application in speech processing, gesture recognition, and other important areas. Interestingly, hidden Markov models were discussed in the context of source coding by Gallager [41, Chapter 3] more than a decade before they became popular in speech processing.

For the remainder of this chapter, successive observations will be assumed to be independent. This is not as restrictive as it may sound, because the observations will be allowed to be multivariate and high-dimensional. In particular, the estimation of high-order conditional densities will be considered extensively. High-order conditional densities are those in which many conditioning variables are used. In the applications to be considered in subsequent chapters, the conditioning orders range between zero and fourteen. Conditional densities can be used to characterize a stationary m^{th}-order Markov process by simply sliding a window of length $m+1$ along the observed sequence, and, at each position, taking the observations that fall within the window as the observed vector. Subsequent chapters will employ this device often, extending it as necessary to multidimensional sequences (e.g., image subbands).

Basic Terms and Notation

Since the topic at hand is probability density estimation, it seems appropriate to begin the technical discussion by briefly reviewing relevant terms and concepts from elementary probability theory. Besides making the presentation more self-contained, such a review offers the opportunity to introduce notation in a natural way, and will also allow us to develop a slightly simpler, albeit more restrictive, notion of density than is usually presented in textbooks.

A rigorous, measure-theoretic approach will *not* be taken here, for three reasons. First, the density estimation techniques to be discussed involve complex, data-driven procedures such as tree-growing and -pruning, for which a theoretical analysis would be difficult, though considerable progress has been made in this general area — see the recent work by Lugosi and Nobel [74], Nobel [81], Nobel and Olshen [82] and Devroye et al. [34, Chapters 20–23]. Second, the techniques and analyses to be presented were arrived at through practically motivated, somewhat heuristic reasoning, and it is desirable from a pedagogical standpoint not to obscure this path through *post hoc* theorizing. Finally, the added precision afforded by a formal approach would have bearing mostly on asymptotic analysis, whereas in practice near-asymptotic performance is seldom reached, especially in the high dimensional setting. Although some theoretical analysis is in fact possible in the case of finite samples, the resulting conclusions would necessarily be given in terms of a probability measure which, in practice, remains at all times unknown.[2] In other words, a precise and rigorous theoretical analysis would ultimately require that the true distribution be known, either explicitly, or else implicitly as contained in an infinitely large sample.

Proceeding informally then, we begin by defining a *random object* to be an experiment which produces, upon each performance or *trial*, a result called an *outcome*. The random object is hypothesized to be governed by a fixed *probability law* which specifies, for any set A of interest, the probability $\mathrm{Prob}[A]$ that the outcome will lie in A on any trial. It is assumed that $\mathrm{Prob}[\,]$ satisfies the usual requirements of a probability measure: nonnegativity, normalization, and countable additivity [2]. The quantity $\mathrm{Prob}[A]$ can be interpreted as (but need not be defined in terms of) the limiting relative frequency of values that lie in A in a hypothetical, infinitely long sequence of trials.[3]

A *random variable* is a random object having scalar-valued outcomes, and a *random vector* is a random object having vector-valued outcomes. It will be assumed throughout that all vector

[2] Such assumptions can be self-serving, as Rissanen points out in Chapter 2 of [110]. For instance, if a lossless source code produces 5000 bits in encoding a particular sequence of 1000 binary outcomes, then it should come as little consolation when the designer of the code claims (even if correctly) that the expected code length is only 200 bits.

[3] Although there is now nearly universal agreement about what the axioms in a mathematical theory of probability ought to be (though there remains some dissent — see [38]), there is comparatively little agreement about what the *interpretation* of probability ought to be. The frequentist and subjectivist interpretations, which underlie traditional and Bayesian statistics respectively, are the two most familiar opposing views; there exist others as well. For an overview of viewpoints and issues, the interested reader is referred to the engaging book by Weatherford [140]. The variety in viewpoints regarding the interpretation of probability closely reflects those regarding statistical inference, some of which are touched upon elsewhere in this section.

outcomes lie in Euclidean space of the appropriate dimension. Occasionally, the word "point" will be used in place of "vector." The coordinates of a random vector are *jointly distributed* random variables, so termed because the probability law of each coordinate generally depends on the outcomes of the other coordinates.

A random variable will generally be denoted by an uppercase math italic letter, and a particular outcome by the corresponding lowercase letter. Random vectors and their outcomes will be distinguished in the same way, but in boldface. In general, the d^{th} coordinate of a random vector $\boldsymbol{Z}$ or outcome $\boldsymbol{z}$ will be denoted by Z_d or z_d, respectively. Occasionally, a D-dimensional random vector $\boldsymbol{Z}$ will be written explicitly in terms of its coordinates as

$$\boldsymbol{Z} = (Z_1, \ldots, Z_D),$$

and analogous notation will be used to express a particular outcome in terms of its coordinates. Vectors will be treated at times as rows and at other times as columns; the appropriate interpretation should always be clear from the context. An object in a sequence will be identified by a parenthesized superscript; for example, the n^{th} observation in a sequence of observations of $\boldsymbol{Z}$ will be denoted $\boldsymbol{z}^{(n)}$. The sequence itself will be denoted by $(\boldsymbol{z}^{(n)})_{n=1}^{N}$, or, if N is understood or unspecified, simply by $(\boldsymbol{z}^{(n)})$. In general, sequences will be represented by calligraphic letters. Analogous notation will be used for sequences of scalars, matrices, and functions as needed. When no potential for conflict exists, a subscript will sometimes be used to index a sequence instead of a parenthesized superscript. The d^{th} coordinate of $\boldsymbol{z}^{(n)}$ will be written $z_d^{(n)}$, while the element in the i^{th} row and j^{th} column of the n^{th} matrix $A^{(n)}$ in a sequence will be written $a_{ij}^{(n)}$.

The probability law governing a D-dimensional random vector $\boldsymbol{Z}$ can be summarized by the *cumulative distribution function* $F_{\boldsymbol{Z}}$, defined as

$$F_{\boldsymbol{Z}}(\boldsymbol{z}) = \mathrm{Prob}[\{\boldsymbol{z}' \in \mathbb{R}^D : z_1' \le z_1 \text{ and } z_2' \le z_2 \text{ and } \cdots \text{ and } z_D' \le z_D\}].$$

If the probability law is such that the limit implicit in the right-hand side of

$$f_{\boldsymbol{Z}}(\boldsymbol{z}) = \frac{\partial^D F_{\boldsymbol{Z}}(\boldsymbol{z})}{\partial z_1 \partial z_2 \cdots \partial z_D} \tag{2.2.1}$$

exists everywhere except possibly on a set that is assigned zero probability, then $\boldsymbol{Z}$ is said to be *continuous*. For vectors that are suspected to be continuous on physical grounds, an assumption of continuity is always tenable in the practical density estimation setting in that it is operationally non-falsifiable: it cannot be refuted on the basis of a finite sample of outcomes. Under the assumption of continuity, $f_{\boldsymbol{Z}}(\boldsymbol{z})$ as defined in (2.2.1) is called the *probability density function* or *PDF* or simply the *density* of $\boldsymbol{Z}$, and serves as an alternative means of specifying the governing probability law. In particular, for any set $A \subseteq \mathbb{R}^D$ of interest,

$$\text{Prob}[A] = \int_{\boldsymbol{z} \in A} f_{\boldsymbol{Z}}(\boldsymbol{z}) d\boldsymbol{z}. \tag{2.2.2}$$

We arbitrarily take $f_{\boldsymbol{Z}}(\boldsymbol{z})$ to be zero whenever $\boldsymbol{z}$ lies in the zero-probability set for which the limit in the right-hand side of (2.2.1) fails to exist. When it is desired that the dependence on $f_{\boldsymbol{Z}}$ be indicated explicitly, the left-hand side of (2.2.2) will be written as $\text{Prob}\{A \,|f_{\boldsymbol{Z}}\}$. Also, the notation $\boldsymbol{Z} \sim f_{\boldsymbol{Z}}$ will be used to indicate that $f_{\boldsymbol{Z}}$ is the PDF for $\boldsymbol{Z}$. Analogous assumptions and definitions are made in the scalar case. When no confusion is likely to result, the subscript identifying the random object referred to by F or f will be omitted, and the argument will be specified as an outcome or a random object, depending on whether it is to be regarded as fixed or indeterminate, respectively. For example, $f(\boldsymbol{X})$ refers to the PDF of $\boldsymbol{X}$, while $f(\boldsymbol{x})$ refers to the value of that PDF evaluated at the point $\boldsymbol{X} = \boldsymbol{x}$. The notations $f(\boldsymbol{z})$ and $f(z_1, \ldots, z_D)$ will be used interchangeably, the latter being convenient when it is necessary to indicate that some coordinates are fixed others indeterminate.

If $\{d_1, \ldots, d_i\} \subset \{1, \ldots, D\}$, then the *marginal* PDF $f(Z_{d_1}, \ldots, Z_{d_i})$ is obtained by integrating $f(\boldsymbol{z})$ over those coordinates indices not in $\{d_1, \ldots, d_i\}$. For any two disjoint subsets $\{d_1, \ldots, d_i\}$ and $\{d'_1, \ldots, d'_{i'}\}$ of coordinate indices, and for any *conditioning vector* $(z_{d'_1}, \ldots, z_{d'_{i'}})$ such that $f(z_{d'_1}, \ldots, z_{d'_{i'}}) \neq 0$, the conditional density $f(Z_{d_1}, \ldots, Z_{d_i} | z_{d'_1}, \ldots, z_{d'_{i'}})$ is defined as

$$f(Z_{d_1}, \ldots, Z_{d_i} | z_{d'_1}, \ldots, z_{d'_{i'}}) = \frac{f(Z_{d_1}, \ldots, Z_{d_i}, z_{d'_1}, \ldots, z_{d'_{i'}})}{f(z_{d'_1}, \ldots, z_{d'_{i'}})}.$$

To facilitate moving back and forth between discussion of joint and conditional density estimation, it will prove convenient to regard $\boldsymbol{Z}$ as being formed by concatenating two jointly distributed component vectors,

$$\boldsymbol{Z} = (\boldsymbol{X}, \boldsymbol{Y})$$

where $\boldsymbol{X}$ has dimension D_x and $\boldsymbol{Y}$ has dimension $D_y = D - D_x$. That is, we posit the correspondence

$$Z_d = \begin{cases} X_d & \text{if } 1 \leq d \leq D_x; \\ Y_{d-D_x} & \text{if } D_x < d \leq D. \end{cases}$$

The component vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ will be termed *independent* and *dependent*, respectively, while the vector $\boldsymbol{Z}$ itself will be termed *joint*. This terminology is suggested by the roles that will generally be played by these variables. In particular, $\boldsymbol{x}$ will usually be observed, and a value of $\boldsymbol{Y}$ will be estimated, generated, or encoded, as appropriate to the task at hand, according to the conditional density $f(\boldsymbol{Y}|\boldsymbol{x})$. In such cases it will always be assumed that $f(\boldsymbol{x}) \neq 0$. When $D_y = 1$, we will write $\boldsymbol{Z} = (\boldsymbol{X}, Y)$.

The assumption that underlies (2.2.1) cannot hold when the outcomes are restricted to assume values in a discrete alphabet. In such cases, the random variable or vector is termed *discrete*, and the governing probability law can be specified by the *probability mass function* or *PMF*, defined in the vector case as

$$P_{\boldsymbol{Z}}(\boldsymbol{z}) = \mathrm{Prob}[\{\boldsymbol{z}\}],$$

and analogously in the scalar case. As with the PDF, the subscript will be omitted when no confusion is likely to result.

The *expectation* of a function g of a random vector $\boldsymbol{Z}$, denoted $E[g(\boldsymbol{Z})]$, is defined as

$$E[g(\boldsymbol{Z})] = \int_{\boldsymbol{z}} g(\boldsymbol{z}) f(\boldsymbol{z}) d\boldsymbol{z} \qquad \text{and} \qquad E[g(\boldsymbol{Z})] = \sum_{\boldsymbol{z}} g(\boldsymbol{z}) P(\boldsymbol{z})$$

in the continuous and discrete cases, respectively. When the probability law with respect to which the expectation is to be computed is not clear from context, it will be indicated by a suitable subscript to the expectation operator E.

For a random vector, the mean $\boldsymbol{\mu}_{\boldsymbol{Z}}$ is defined as

$$\boldsymbol{\mu}_{\boldsymbol{Z}} = E\boldsymbol{Z},$$

and the autocovariance matrix $K_{\boldsymbol{Z}}$ as

$$K_{\boldsymbol{Z}} = E\Big[(\boldsymbol{Z} - \boldsymbol{\mu}_{\boldsymbol{Z}})(\boldsymbol{Z} - \boldsymbol{\mu}_{\boldsymbol{Z}})^{\mathrm{T}}\Big],$$

where $(\boldsymbol{Z} - \boldsymbol{\mu}_{\boldsymbol{Z}})$ is a column vector and the notation $\boldsymbol{x}^{\mathrm{T}}$ indicates the transpose of $\boldsymbol{x}$. In the scalar case, the mean and variance are defined respectively as

$$\mu_Z = EZ \qquad \text{and} \qquad \sigma_Z^2 = E\Big[(Z - \mu_Z)^2\Big].$$

An estimate of the density function f will be denoted by $\hat{f}$. Similarly, if the density is determined by a parameter $\boldsymbol{\theta}$, then $\hat{\boldsymbol{\theta}}$ will refer to an estimate of $\boldsymbol{\theta}$. When necessary to avoid ambiguity, a subscript will be introduced to indicate the particular method of estimation (this may be in addition to an existing subscript identifying the random object).

The term *density estimation* will be used to refer to the construction of an estimate $\hat{f}$ of f from a finite sequence of observations that are regarded as independent outcomes of a random object having the density f. Specific estimation criteria will be considered in Section 2.3. The term *model* will be used to refer either to a specific estimate or to a class of estimates. Generally, the observed sequence $(\boldsymbol{z}^{(n)})$ of outcomes of $\boldsymbol{Z}$, termed the *training sequence*, will be denoted by $\mathcal{L}$ and its length by $|\mathcal{L}|$. When there is occasion to consider several training sequences simultaneously as in Chapter 3, a specific one will be distinguished by writing $\mathcal{L}$ with an appropriate subscript.

Parametric, Semiparametric, and Nonparametric Density Models

In practice, a necessary first step in density estimation in both the continuous and infinite-alphabet discrete cases is *model specification*, wherein the estimate is constrained to be of a specified functional or procedural[4] form having limited degrees of freedom. The inevitability of this step is perhaps best appreciated by contemplating its omission, which would require that f be estimated and specified independently at each of the infinitely many (uncountable, in the continuous case) possible values of its argument.

Models having a fixed small number of degrees of freedom are generally termed *parametric*. The best-known and most important example is the Gaussian density, which is given in the scalar and vector cases respectively by

$$\hat{f}_{\mathrm{G}}(z) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} e^{-(z-\hat{\mu}_Z)^2/(2\hat{\sigma}^2)} \tag{2.2.3}$$

and

$$\hat{f}_{\mathrm{G}}(\boldsymbol{z}) = (2\pi)^{-D_z/2} |\hat{K}_{\boldsymbol{Z}}|^{-1/2} \exp[-\frac{1}{2}(\boldsymbol{z}-\hat{\boldsymbol{\mu}}_{\boldsymbol{Z}})^{\mathrm{T}} \hat{K}_{\boldsymbol{Z}}^{-1}(\boldsymbol{z}-\hat{\boldsymbol{\mu}}_{\boldsymbol{Z}})]. \tag{2.2.4}$$

Models having arbitrarily many degrees of freedom (but still a fixed number of them) are termed *semiparametric*. A simple but important univariate example is the uniform-bin histogram estimate, which partitions a segment of the real line that covers all observations into M cells or *bins* of equal width, and assigns to each a uniform density proportional to the number of occurrences in $\mathcal{L}$ that fall into that bin. The histogram functions by explicitly grouping observations into like occurrences, essentially transforming the continuous problem into a discrete one. A good reference for the asymptotic properties of histogram and other estimates is the recent book by Devroye et al. [34]. Another important semiparametric density model is the *finite mixture*, defined in the vector case

[4] Not all models are specified analytically; some are defined by computer programs; e.g., regression trees. Hence the term "procedural."

as

$$\hat{f}(\boldsymbol{x}) = \sum_{m=1}^{M} \hat{P}(m)\hat{f}_m(\boldsymbol{x}),$$

where $(\hat{f}_m)_{m=1}^M$ is a sequence of *component* densities (usually parametric), and $\hat{P}$ is a PMF over the components. Mixture models will be discussed in greater detail later, particularly in Sections 2.4–2.6.

Finally, a model having a number of degrees of freedom that varies with the amount of training data is generally termed *nonparametric.*[5] An important example of such a model is the *kernel density estimate* [49, 120], which in the scalar case is given by

$$\hat{f}(z) = \frac{1}{h|\mathcal{L}|} \sum_{n=1}^{|\mathcal{L}|} K\Big(\frac{z - z^{(n)}}{h}\Big), \tag{2.2.5}$$

where $h > 0$ is called the *bandwidth parameter* (usually chosen to depend on $|\mathcal{L}|$), and typically the kernel K is itself selected to be a valid PDF. Extension to the multivariate case is straightforward; see Section 2.4 or, for a more general approach, [120] pp. 150–155. Histogram, kernel, and mixture estimates are compared heuristically in Section 2.4.

Some general observations can be made regarding the relative complexity, estimability, and expressive power of models in each of these three categories. Generally speaking, parametric models are the least complex and easiest to estimate, but are comparatively lacking in expressive power. They are most suitable in situations in which the assumed parametric model has a strong physical justification, or when simplicity and mathematical tractability are considered paramount. At the other extreme are nonparametric models, which are extremely flexible in the sense that they can typically approximate any density that lies within a broad class (e.g., all densities that meet a specified smoothness criterion). However, nonparametric models require large amounts of training data for good performance, and their complexity (as measured by representation cost in a given descriptive framework, for example) is correspondingly large. Semiparametric models lie somewhere in the middle. More flexible than parametric models, they are also more complex and correspondingly more difficult to estimate. However, unlike nonparametric models, their complexity does not necessarily grow with increasing $|\mathcal{L}|$. In exchange, the complexity of such models must be carefully selected as part of the estimation procedure (see Section 2.3).

[5] This taxonomy is not universally observed. In particular, the distinction between semiparametric and nonparametric models is not always made; instead, the term "nonparametric" is often used for both. Moreover, the dividing line between "parametric" and "nonparametric" is itself not always clear. See [120], p. 44 for a discussion of this latter point.

The role of probabilistic modeling

Semiparametric and nonparametric density estimation are very much products of the computer age, owing to their representational and computational complexity, but their recent emergence also reveals an evolution in thought about the role of probabilistic modeling. In engineering, statistics, and science in general, one of the chief goals of modeling is *prediction,* and some models that have been extremely and consistently predictively accurate have come to be termed *laws.* Maxwell's equations, for instance, are considered "laws" of electromagnetism, though their justification is wholly empirical [19]. The most general form of prediction involves specifying the degree of confidence that ought to be placed in the predictions, and this can be accomplished in a natural way via a sequence of predictive probability distributions.[6] Another commonly sought goal, more important perhaps in science than in engineering, is *explanation.* This goal is somewhat subjective and therefore difficult to state precisely. Roughly speaking, "explanation" refers to the model's serving to further human understanding of the phenomenon being modeled. Explanation is often tied to the notion of *simplicity*, which is itself subjective or at least language-dependent (with "language" intended here in its broadest sense, referring to any framework for description). In many instances, simplicity is thought to be consistent with the goal of predictive accuracy, but there is no reason to suppose that the putative consistency is necessary [118]. Indeed, some of the most successful techniques for regression and classification currently in use involve extremely complex semiparametric models and procedures that do not seem to lend themselves well to human interpretation [12]. There is no reason to suppose that the situation should be any different for density estimation.

Our goal in modeling subbands is clear: to improve the performance of compression and other systems that operate in the subband domain. This generally translates directly into a goal of predictive accuracy (as will be made more precise in Section 2.3). Explanation is really not a goal; if it were, then we would be better off working in the original domain, since the subbands are deterministic but rather complicated functions of the original signal. Simplicity is therefore of secondary importance, and the model to be preferred is simply that which performs best predictively. Such models tend to be semiparametric or nonparametric, and relatively complex.

[6] Besides providing a descriptive framework, sequential probabilistic modeling can also serve as the basis for a powerful inferential framework [30, 108].

2.3 Evaluation Criteria for Density Estimation

As discussed Section 2.2, in a practical setting there is no *analytic* justification of preferring one density estimate or estimation procedure over another. Therefore, any such justification must be empirical. The ultimate test of a density estimate is how well it will perform when used by a system to carry out a specific task, but unfortunately this criterion is of little use at design time because it depends on future (i.e., unavailable) data. A reasonable recourse is to view the available training sequence $\mathcal{L}$ as being representative of future data, and to adopt accordingly an evaluation criterion that is a function of $\mathcal{L}$. Specific function forms for such an evaluation criterion are considered later in this section; for now, the requirements on the criterion will be discussed in general terms.

Regularization and Crossvalidation

An important use of an evaluation criterion is as a target function to be optimized in removing the degrees of freedom from the model; that is, in *training* the model. Models having many degrees of freedom, such as semiparametric and nonparametric ones, can usually be trained to fit the training sequence quite well in terms of any criterion that is solely a function of $\mathcal{L}$. However, as the number of degrees of freedom in the model increases, it becomes less likely that a training sequence of a given length is truly representative of future data in those aspects that are important to the model. The empirical criterion then becomes less reliable as an estimate of performance on future data, generally resulting both in an optimistic performance estimate, and in a trained model that *overfits* the training sequence at the expense of future performance. Sometimes this phenomenon is explained in terms of *bias* and *variance*. Roughly speaking, bias refers to the inflexibility of the model, while variance refers to the unreliability of the training. Inflexible models can generally be trained reliably (high bias, low variance), while flexible models tend to be more difficult to train reliably (low bias, high variance).

Several approaches are commonly used to combat overfitting in the training process. One is simply to obtain a longer training sequence, but in many situations this may not be practical. Also, lengthening the training sequence doesn't address another potential cause of overfitting that can occur even when the model has few degrees of freedom: one or more singularities in the model likelihood being coincident with individual training points. To illustrate, consider a one-dimensional mixture with two components, one having a broad variance, and the other a vanishingly small one. If the latter component happens to lie on a training point, the likelihood will be large, and such a model is grossly overfit. This type of overfitting can be prevented by regularization or detected by crossvalidation, but merely lengthening the training sequence provides neither a preventative nor a diagnostic effect. In any event, it would remain desirable to minimize when possible even the lesser degree of overfitting resulting from the use of a longer training sequence. A widely used strategy is to incorporate a penalty or *regularization* term into the criterion to discourage overfitting; such a penalty term may be regarded as an embodiment of prior knowledge about the density being estimated. Regularization is the basis for every approach mentioned in Section 2.2, as well as one not mentioned there: Akaike's information criterion [116], which will not be discussed except to say that it is similar to MDL [110]. Parameter estimates obtained by optimizing regularized criteria can, in certain cases, be shown to have desirable asymptotic theoretical properties, such as consistency [4, 110]. However, unpenalized ML is also fine asymptotically, but notoriously unreliable on small samples, so it is not clear exactly what consistency *per se* really buys us in a practical setting.

An important and somewhat different approach, applicable to semiparametric models, is *crossvalidation*. The idea is to identify a low-dimensional (usually scalar) *structural metaparameter* M that controls the flexibility of the model (and hence its propensity to overfit), then to adjust its value empirically. This metaparameter is usually selected to be directly reflective of the complexity of the model. For instance, in a mixture model, a natural choice for M would be the number of components. The model is trained for each of several candidate values of M, and the performance in each case is estimated using an evaluation criterion which is also based on $\mathcal{L}$, but which is chosen to be largely independent of the training of the model. One way of achieving this independence is to divide $\mathcal{L}$ into two segments, one slated for training and the other for selection of M. In this approach, known as *holdout crossvalidation,* the term *training sequence* and the symbol $\mathcal{L}$ are redefined to refer to one of the two segments of the original training sequence, while term *test sequence* and symbol $\mathcal{T}$ are used for the other.

A variation on this technique, known as $|\mathcal{L}|$-fold crossvalidation, involves training the model for a given value of M a total of $|\mathcal{L}|$ times, each time using a $(|\mathcal{L}| - 1)$-long training sequence that consists of the original training sequence with a different observation deleted. Although this is also known as the "leave-one-out method;" we adopt the terminology used by Breiman et al. [13] because we use their notation elsewhere extensively. Each such trained model for that M is then evaluated for performance, taking as $\mathcal{T}$ the single observation that was deleted in training. The results of these $|\mathcal{L}|$ evaluations are then averaged to obtain the overall evaluation criterion for M. This approach doesn't quite meet the desired goal of having the evaluation of

M be independent of training, since the selective deletion process generally results in a unduly pessimistic estimate (but overfitting is avoided *a fortiori*). Also, unlike holdout crossvalidation, its use is limited to sequences of independent observations (assumed here anyway). Nevertheless, $|\mathcal{L}|$-fold crossvalidation has proven very useful for the purpose of selecting model complexity, and is extremely parsimonious with the training sequence, allowing useful inference of M even when $|\mathcal{L}|$ is quite small (e.g., $|\mathcal{L}| \approx 30$). For this reason, $|\mathcal{L}|$-fold crossvalidation will be a valuable tool in the fitting of one-dimensional mixture densities to leaves in the tree-based methodology to be described in Chapter 3. The requirement that M be low-dimensional is to prevent overfitting of the model complexity with respect to the independent evaluation criterion. Conceivably, a second level of crossvalidation could be performed for M, but since the structural metaparameter usually has only one degree of freedom, overfitting of M seldom occurs as a result of selecting it in either of the ways described above.

While the foregoing discussion has focused specifically on the use of crossvalidation in selecting model complexity, it should be pointed out that the general strategy of testing on different data than was used in training is useful whenever an honest estimate of future performance is sought. Any such testing will generically be referred to as crossvalidation.

Specific Criteria: Likelihood and Relative Entropy

We now discuss the choice of functional form of the evaluation criterion for a density estimate $\hat{f}$ of f based on a particular training sequence, considering joint density estimation first. In theoretical analyses of density estimation [33, 120], where the training sequence and therefore $\hat{f}$ are considered random, the usual L_p norms for functions are often used, where

$$L_p(\hat{f}, f) = \Big[\int \Big|\hat{f}(\boldsymbol{z}) - f(\boldsymbol{z})\Big|^p d\boldsymbol{z}\Big]^{1/p}. \tag{2.3.1}$$

The most frequently used values of p are 1 and 2, with the square of L_2 usually referred to in the density estimation literature as *integrated square error* (ISE). A related criterion that is often easier to manipulate [120] is the average of the ISE, called *mean integrated square error* (MISE):

$$\text{MISE} = E_{\mathcal{L}}\Big\{\int \Big[\hat{f}(\boldsymbol{z}) - f(\boldsymbol{z})\Big]^2 d\boldsymbol{z}\Big\}. \tag{2.3.2}$$

Although MISE can be viewed as a special case of L_2 with an appropriate choice of integration measure, we follow the convention used in Scott's book [120] of taking L_p to be with respect to Lebesgue measure, so that MISE is distinct. The above criteria are useful when f is known or can reasonably be assumed, or when a convergence statement is sought relative to f (as when demonstrating consistency), but in the practical density estimation setting, none of these conditions is satisfied. The MISE is perhaps a bit closer than L_p to being useful in the practical setting, since the expectation may be brought inside the integral and approximated by a sample average, but the interaction of $\hat{f}$ and f in the cross term of the integrand limits its use as well. Another disadvantage of the above criteria is that they are not particularly well tailored to densities; for instance, no special penalty or weighting is given by (2.3.1) or (2.3.2) to extremely small (relative to f) or even slightly negative values of $\hat{f}$; all that is measured is the absolute deviation from f. Of course, in the asymptotic analyses where these criteria are typically applied, some sort of convergence to f is usually demonstrated, and that convergence can often be used to establish convergence with respect to other metrics. However, this requires that the density estimate become essentially perfect in an appropriately defined limiting sense, which of course does not happen in the practical setting where $|\mathcal{L}|$ is finite.

What is really needed is an empirical criterion that remains meaningful when f is not known, and that predicts the performance of a real system that uses the density estimate. For our purposes, it is not required that the criterion be a metric between f and $\hat{f}$. It is emphasized that estimating f is a practical problem of inductive inference rather than a theoretical exercise. One approach would be to implement or simulate the system that will employ the density estimate, then use its empirical (crossvalidation) performance as the evaluation criterion for density estimation. While desirable on purist grounds and workable in some situations, this approach generally has two disadvantages. The first is computational complexity: having a simulated system in the loop may make training unacceptably slow. The second is a potentially rough optimization error surface, owing to the real-system effects of quantization, integer code length constraints, and the like, hampering the training process.

We now consider as a possible criterion the likelihood of the model given the training data. Note that for fixed training data, the likelihood is independent of f, as desired. It will be argued that the log likelihood is a reasonable predictor of system performance in certain important applications. This is particularly easy to show in the case of lossless compression of a sequence of independent observations of a discrete-valued random vector $\boldsymbol{Z}$ having PMF $P(\boldsymbol{Z})$. The number of bits produced by an arithmetic coder [68, 112] is likely to be close to the ideal Shannon code length of

$$-\sum_{n=1}^{|\mathcal{L}|} \log_2 \hat{P}(\boldsymbol{z}^{(n)}) \tag{2.3.3}$$

bits, which is simply the negative log likelihood. For $|\mathcal{L}|$ large, this in turn is likely (with respect to P) to be close after dividing by $|\mathcal{L}|$ to the expectation $-\sum_{\boldsymbol{z}} P(\boldsymbol{z}) \log_2 \hat{P}(\boldsymbol{z})$, which can be thought of as representing average future performance. Thus, the problem of designing a lossless compressor is actually equivalent to that of estimating the governing probability law. This was long known to be true in principle, but became true in practice primarily through the advent of arithmetic coding in the late 1970s [113, 86]. The excess of the expectation over the entropy, which is the achievable lower bound for compressing independent, identically distributed (i.i.d.) sources, turns out to be the *relative entropy* between P and $\hat{P}$, which is denoted and defined in the discrete case as

$$D(P||\hat{P}) = -\sum_{\boldsymbol{z}} P(\boldsymbol{z}) \log_2 \frac{P(\boldsymbol{z})}{\hat{P}(\boldsymbol{z})}. \tag{2.3.4}$$

The relative entropy is not a true metric between densities because it is not symmetric and does not satisfy the triangle inequality [26]. On the other hand, it behaves in many ways like a metric and has a similar geometric interpretation in many problems [27]. Note that we cannot estimate $D(P||\hat{P})$ directly because P is never known, but we can estimate the difference in relative entropies for two competing density estimates, since P appears in

$$D(P||\hat{P}) \approx -\frac{1}{|\mathcal{L}|} \sum_{n=1}^{|\mathcal{L}|} \log_2 \frac{P(\boldsymbol{z}^{(n)})}{\hat{P}(\boldsymbol{z}^{(n)})}$$

only as an additive constant, approximately the negative of the entropy, which is common to the relative entropies for both estimates. In other words, we can compare relative entropies by comparing average log likelihoods. Contrast this with the discrete analog of the MISE criterion given in (2.3.2), in which the interaction between P and $\hat{P}$ would prevent such a comparison without knowledge of P.

To motivate the adoption of the likelihood criterion in the general lossy compression setting is more difficult because the degree of compression achieved for a given source is not a function of just the assumed probability model, but also of the choice of distortion measure and the allowed level of distortion with respect to that measure. The approachable bound on performance in the lossy setting is given by the *rate-distortion function* (RDF) [125], which is known analytically for comparatively few sources and distortion measures [41, 6], and in general must be computed numerically [10]. Therefore, it would be difficult to make a strong case for the crossvalidated likelihood criterion (or any other criterion) by attempting to relate it to actual or even best-case system performance in any sort of general way. Instead, we shall be content to show that maximizing likelihood is the right objective for a specific method of encoding and distortion measure, at least in the high-rate portion of that method's operational rate-distortion curve. In particular, we consider entropy-constrained scalar quantization (ECSQ), using the mean-square error (MSE) distortion measure. ECSQ is a simple but important and widely used method of quantization that performs fairly well for i.i.d. sources. For example, forms of it are used in both JPEG and MPEG. Later in this section, and also in Chapters 4 and 5, this technique will be extended to apply to non-independent sources by employing conditional instead of marginal density estimates, i.e., Markov sources.

Mathematically, a scalar quantizer is a function q that maps a continuous-valued input z into a discrete and often finite sequence of *representation levels* (r_i), according to the relationship

$$q(z) = r_i : d_i \leq z < d_{i+1},$$

where the cells in the sequence $([d_i, d_{i+1}))$ form a partition of the real line (in the case of a finite partition, the first cell is taken to be open on the left at $-\infty$). The d_i's are termed *decision levels* or *thresholds*. It will be convenient to define the input-to-cell indexing function

$$i(z) = i : d_i \leq z < d_{i+1},$$

so that $q(z) = r_{i(z)}$. Note that the entropy of $i(Z)$, which is the entropy of the quantizer output q(Z), is generally finite even when the number of cells is not.

It is well-known that under an entropy constraint, uniformly spaced thresholds are asymptotically optimal for small levels of distortion with respect to several measures (including MSE), for a broad class of i.i.d. sources [46, 7]. Uniform spacing has also been found experimentally to be nearly optimal at all rates for several important sources and distortion measures [94]. Accordingly, we assume a uniform cell width of Δ. For tractability, we further assume that there are infinitely many cells, and that the representation levels are the midpoints of the cells. The latter assumption incurs some loss in MSE-performance, generally significant only at low rates. If Δ is small and if f is continuous and smooth (see [18, 80] for rigorous statements of these conditions), the quantization error at high rates will be nearly uniform on the interval $[-\Delta/2, \Delta/2]$, so that the MSE will be very close to $\Delta^2/12$, largely independent of the actual rate. Approximations of this type were first developed by Bennett [5]. The average rate R (in nats) per sample produced by ideal entropy coding of the quantizer output for the training sequence, assuming that $\hat{f}(q(z^{(n)})) > 0\ \forall n$, will be

$$
\begin{aligned}
R &= -\frac{1}{|\mathcal{L}|}\sum_{n=1}^{|\mathcal{L}|} \ln \int_{d_{i(z^{(n)})}}^{d_{i(z^{(n)})+1}} \hat{f}(z)dz \\
&\approx -\frac{1}{|\mathcal{L}|}\sum_{n=1}^{|\mathcal{L}|} \ln \Delta \hat{f}(q(z^{(n)})) \\
&= -\ln \Delta - \frac{1}{|\mathcal{L}|}\sum_{n=1}^{|\mathcal{L}|} \ln \hat{f}(q(z^{(n)})),
\end{aligned}
\tag{2.3.5}
$$

which is the negative of the average log likelihood plus a constant. Thus, the likelihood criterion is directly relevant to uniform ECSQ, and hence to the method of lossy compression to be considered in subsequent chapters. It is instructive to continue the chain (2.3.5) with the approximation

$$
\begin{aligned}
R &\approx -\ln \Delta - \int_{-\infty}^{\infty} f(z)\ln \hat{f}(z)dz \\
&= -\ln \Delta + h(Z) + D(f||\hat{f}),
\end{aligned}
$$

where $h(Z) = \int_{-\infty}^{\infty} f(z)\ln f(z)dz$ is the *differential entropy* of Z, and $D(f||\hat{f})$ is the continuous version of relative entropy [26], defined as

$$
D(f||\hat{f}) = \begin{cases} \int_{z:\hat{f}(z)\neq 0} f(z)\ln \frac{f(z)}{\hat{f}(z)}dz & \text{if Prob}\{z : \hat{f}(z) = 0\ |f\} = 0 \\ +\infty & \text{otherwise.} \end{cases}
\tag{2.3.6}
$$

As in the discrete case, the difference in average log likelihoods for two competing density estimates will approximate (with a sign reversal) the difference in relative entropies. Again, $D(f||\hat{f})$ can be thought of as measuring the discrepancy between the two densities f and $\hat{f}$.

While the foregoing justification of the likelihood criterion for lossy compression does not extend in direct way to vector sources, we can assume (with some further loss in performance) that scalar quantization is performed sequentially on the vector coordinates, leading again to the likelihood criterion for vectors via the chain rule for densities.

In both the lossless and lossy compression systems described above, the i.i.d. assumption may be relaxed and a stationary m^{th}-order Markov model assumed. The average rates for both the lossy and lossless systems are then obtained by substituting $\hat{P}(y^{(n)} \mid \boldsymbol{x}^{(n)})$ and $\hat{f}(y^{(n)} \mid \boldsymbol{x}^{(n)})$ for $\hat{P}(\boldsymbol{z})$ and $\hat{f}(z)$ in (2.3.3) and (2.3.5) respectively, where $(y^{(n)})$ is the sequence of observations and $\boldsymbol{x}^{(n)}$ is defined as the m-vector whose i^{th} coordinate is $y^{(n-m-1+i)}$ (the dimension of $\boldsymbol{x}^{(n)}$ is reduced as necessary near the boundary $n = 0$ by omitting coordinates of $\boldsymbol{x}^{(n)}$ that would correspond to negative values of n). The rates are minimized by maximizing the corresponding total log conditional likelihoods, which are $\sum_{n=1}^{|\mathcal{L}|} \log_2 \hat{P}(y^{(n)} \mid \boldsymbol{x}^{(n)})$ and $\sum_{n=1}^{|\mathcal{L}|} \ln \hat{f}(y^{(n)} \mid \boldsymbol{x}^{(n)})$ for the lossless and lossy cases, respectively. Thus, adoption of the average log conditional likelihood criterion is well motivated for sequential compression of a Markov source. As in the unconditional case, differences in likelihood for two competing models can be related to differences in relative entropy, but the appropriate version of relative entropy is now in terms of conditional density. We consider only the continuous case. Define $D(f_{Y|\boldsymbol{x}}||\hat{f}_{Y|\boldsymbol{x}})$ to be the relative entropy between $f(Y|\boldsymbol{X}=\boldsymbol{x})$ and $\hat{f}(Y|\boldsymbol{X}=\boldsymbol{x})$. Then the *average conditional relative entropy* between $f_{Y|\boldsymbol{X}}$ and $\hat{f}_{Y|\boldsymbol{X}}$ is defined as

$$D(f_{Y|\boldsymbol{X}}||\hat{f}_{Y|\boldsymbol{X}}) = \begin{cases} E_{\boldsymbol{X}} D(f_{Y|\boldsymbol{x}}||\hat{f}_{Y|\boldsymbol{x}}) & \text{if } \mathrm{Prob}\{\boldsymbol{y} : \hat{f}(\boldsymbol{y}|\boldsymbol{x}) = 0 \;|f_X\} = 0 \\ +\infty & \text{otherwise.} \end{cases}$$

So far, only compression has been considered. Although classification will not be explicitly considered in detail, the likelihood criterion is easily justified by its role in the (empirical) Bayes risk. Likelihood is also an appropriate criterion when seeking to synthesize textures by generating pseudorandom numbers according to an appropriate conditional distribution (i.e., synthesis by typical sequence generation), as will be considered in Chapter 4. However, there are two applications to be considered in subsequent chapters in which likelihood is not as well motivated: image restoration and deterministic texture synthesis. In these, a sequence of estimates of pixel values is needed, rather than a sequence of estimates of the distribution of values. While knowledge of the latter is generally sufficient for computing the former, it is seldom necessary for doing so, and (to paraphrase Vapnik [137]) a performance penalty may result from attempting to solve a harder problem (density estimation) as an intermediate step towards solving an easier one (function approximation). The same comment can be seen to apply to classification. While likelihood is indeed well motivated by considering Bayes risk, estimating the entire density is usually not. What matters are the classification region boundaries, and this usually depends on the behavior of class-conditional densities only in the vicinity of those boundaries. The behavior elsewhere is largely irrelevant to determining the shape of the boundaries, and hence to the classification task. Allocating precious model resources to characterize the entire densities may be wasteful in this application. On the other hand, knowing where the decision boundaries lie requires some knowledge of the class-conditional distributions. These considerations suggest that for efficient density estimation in the classification setting, the estimation criteria should target discrimination rather than representation. A good example of this is the work by Pudil et al. [100], wherein the dimensions providing the greatest discriminatory power are modeled separately for each class, while those dimensions with less discriminatory power are modeled by a common "background" distribution.

In concluding this section, it is worthwhile to compare the ontological status of f with that of $\hat{f}$. For a particular training sequence, $\hat{f}$ is a fixed function in the usual sense, and may, for example, be realized at least approximately in a computer program.[7] On the other hand, f is merely hypothesized to exist, and in any case remains at all times unknown. Fortunately, f enters the picture only tangentially. In the foregoing discussion, crossvalidated likelihood gained its significance in two ways: operationally from its relation to the average number of bits produced by actual encoders, and conceptually from the inductive belief that the training sequence is representative of future data. The density f was necessary only in defining relative entropy, which, though intuitively appealing, is dispensible from the point of view of arguing for adoption of the likelihood criterion. It is slightly ironic that a notion of a true governing probability law is unnecessary for the term "density estimation" to be meaningful, but perhaps no more so than in those other situations in inference and coding that can be approached without positing a true distribution or even a probability space [28, 30, 77, 108].

2.4 Histogram, Kernel, and Mixture Density Models

A probabilistic model should both generalize and summarize the training sequence. *Generalization* refers to the property that performance on future data will be comparable to that observed on the training sequence; that is, it refers to the absence of overfitting.[8] *Summary* refers to the process of extracting and representing in the model only the information in the training sequence that is actually relevant to the information source (the sufficient statistics, where applicable). These objectives are not intrinsically at odds with each other; indeed, a model that uses only (and all of) the relevant information in the training sequence necessarily generalizes. On the other hand, the terms are not synonymous, since a model can generalize without being parsimonious. To illustrate how these two objectives are typically met to varying degrees in practice, we compare three well-known models — histogram, kernel, and mixture density estimates — in a simple two-dimensional example involving discrete-amplitude pixel data taken from a natural texture.

[7] Contrast this with the interpretation of $\hat{f}$ as a random object, necessary in theoretical analyses. Such an interpretation derives from the training sequence being regarded as random instead of fixed.

[8] Of course, this definition implies that generalization is as much a property of the future data as it is of the model. This dichotomy parallels one that underlies the inductive hypothesis, which asserts that *the future data will resemble the past in all the ways that count*. Here, the "ways that count" are determined by the flexibility of the model, which is under our control, while the assertion "will resemble the past" is up to nature and can only be hypothesized [136].

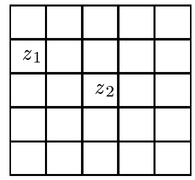

Figure 2.4.1: A training sequence was obtained by successively centering the pictured neighborhood at every pixel location (in raster order) on a 150×150 subimage of the 512×512 8-bit Brodatz texture *D1*. The $150^2 - 300 - 148 = 22,052$ vectors for which both coordinates fell within the subimage boundaries were retained. Three different density estimates for the vectors are shown in Figure 2.4.2.

As described in the caption to Figure 2.4.1, a training sequence of two-dimensional vectors was obtained from the Brodatz texture *D1*. Although the values of these observed vectors are restricted to a discrete alphabet of cardinality 256^2, they may be viewed as having originated from the quantization to 8 bits per coordinate of a hypothetical original sequence of continuous-valued vectors . Thus, the number of occurrences in the training sequence of a particular vector value can be interpreted as the occupancy of a corresponding bin in a histogram density estimate performed on the hypothetical continuous data. In particular, the number of occurrences of (z_1, z_2), assuming $z_1, z_2 \notin \{0, 255\}$, can be viewed as representing the occupancy of the square 1×1 bin in $\mathbb{R}^2$ centered at (z_1, z_2). Because the histogram is two-dimensional, it may be conveniently displayed as an image, as in done in Figure 2.4.2(a). The logarithm of the density has been taken to render the tail regions more clearly. It should be noted that most of the bins are empty; this is to be expected because there are $65,536$ bins and only $20,052$ observed points. Moreover, many of the empty or nearly empty bins occur near populous ones; this is evidence of the histogram failing to generalize adequately the observed training points to surrounding regions of space. The speckled overall appearance indicates that this particular histogram estimate is unreliable, again owing to low average bin populations. The problem is worse in higher dimensions; for example, if triples of pixels were used instead of pairs, then at most about 1% of the bins could be nonempty. This manifestation of the curse of dimensionality has been dubbed the *empty space phenomenon* by Scott and Thompson [121]. An alternative would be to use larger bins so that more observations fall into each, or equivalently, coarser quantization. This would result in better generalization, but at the expense of resolution, and the tradeoff achievable in this way by the histogram generally becomes unacceptable in higher dimensions. Another approach would be regularization, such as Laplace's rule as discussed in Section 2.2. In this example, applying that regularizer would correspond to adding a constant background value to the histogram image shown in the figure, eliminating empty cells, but doing little to reduce the apparent noisiness (speckling). The main problem with the histogram is that it imposes a hard categorization, attempting to transform a metric-space observation into a categorical one. In doing so, it fails to make good use of prior knowledge of the smoothness of the underlying distribution, and this failure is exacerbated in high dimensions. While various modifications have been proposed to mitigate this problem [120], none seems entirely satisfactory, and consequently the appropriateness of the histogram and its close variants seems restricted to the scalar case.

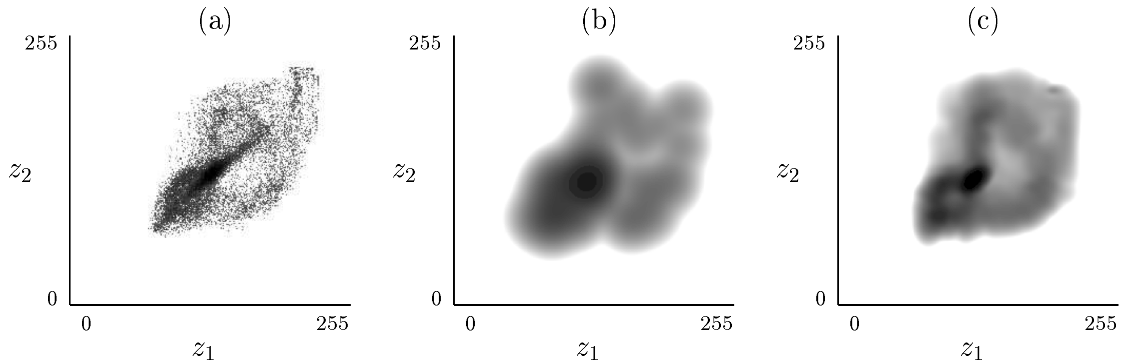

Figure 2.4.2: Three different density estimates for the two-dimensional data described in Figure 2.4.1 are shown after taking the logarithm. Estimate (a) is a histogram estimate, where the bins have been defined implicitly by the 8-bit amplitude resolution of the source image. Estimate (b) is an economized kernel estimate based on 128 randomly selected points using an isotropic Gaussian kernel function having a variance of $\sigma^2 = 30$ in each dimension. Estimate (c) is a mixture density estimate also having 128 components, trained using the generalized Lloyd algorithm as described in Section 2.5.

Kernel density estimation is an important technique with a well-developed supporting theory [34, 49, 120], much of it focusing on the asymptotic properties as the training sequence becomes arbitrarily large [33]. The one-dimensional kernel density estimate (2.2.5) mentioned in Section 2.2 can be extended in a straightforward way to multiple dimensions, by employing isotropic kernels which are themselves two-dimensional densities:

$$\hat{f}_{\mathrm{K}}(\boldsymbol{z}) = \frac{1}{|\mathcal{L}|} \sum_{n=1}^{|\mathcal{L}|} f_{\mathrm{K}}(\boldsymbol{z} - \boldsymbol{z}^{(n)}) \tag{2.4.3}$$

In this notation, the bandwidth parameter in (2.2.5) has been absorbed into the kernel function f_{K}. The kernel density estimate exploits the assumed smoothness of the underlying distribution by radiating probability to the space immediately surrounding the observations, resulting in direct generalization without hard categorization. However, it does not summarize: unlike the histogram, the kernel estimate retains the precise locations of the observations instead of pooling them into bins. Additionally, the summation in (2.4.3) is over the entire training sequence, which in this example consists of 20,052 points. Thus, both the representational and computational complexities of the kernel estimate are quite high, and this level of complexity is not justified by the length of the training sequence.[9] The complexity of the histogram is also unreasonably high in this example, but it is fixed independently of the training sequence. A modification to the kernel estimate would be to use fewer training points by randomly subsampling the training sequence. In this way, the complexity M of the kernel estimate can be controlled independently of $|\mathcal{L}|$ provided that $M \leq |\mathcal{L}|$, making it a semiparametric rather than nonparametric technique. For the data at hand in this example, using an isotropic Gaussian kernel function with a variance of $\sigma^2 = 30$ in each dimension and selecting the number of kernel locations to be $M = 128$, the density estimate depicted in Figure 2.4.2(b) was obtained. The result is free of the speckling that plagues the histogram estimate, but its resolution is apparently lower. The tradeoff can be controlled by adjusting the width of the kernel, just it can be controlled in the histogram by adjusting the bin size. It should be noted however that the kernel estimate, after subsampling, does a poor job in characterizing the tail regions. This is because the kernel estimate relies on sampling from the distribution that it is trying to model, and improbable regions are represented by correspondingly few samples in the subsampled training sequence. Increasing the length of the subsequence would help in the representation of the tails, but would also increase the representational and computational complexity of the model.

A natural further modification would be to select specifically a set of locations for the kernels that are deemed representative, rather than to rely on random subsampling. In this way, the kernel estimate can be made to summarize the training sequence. Two useful additional modifications would be to allow the shapes of the kernels to adapt to the local characteristics of the density, and to allow unequal weighting of the kernels in the summation. These changes would transform the kernel estimate into the *mixture density estimate*, which was discussed briefly in section Section 2.2:

$$\hat{f}_{\mathrm{mix}}(\boldsymbol{z}) = \sum_{m=1}^{M} \hat{P}_{\mathrm{mix}}(m)\hat{f}_{\mathrm{mix}}(\boldsymbol{z}|m) \tag{2.4.4}$$

The kernel functions $\{\hat{f}_{\mathrm{mix}}(\boldsymbol{z}|m)\}$ are known in this context as the *component densities* or simply *components*, and the probability masses $\{\hat{P}_{\mathrm{mix}}(m)\}$ which act as weighting factors are called *mixing parameters* or *mixing probabilities*. A mixture density estimate using $M = 128$ components for this

[9] For several semiparametric model classes for discrete data (e.g., the multinomial PMF), the natural complexity of the model grows approximately with the logarithm of the length of the training sequence. This observation provides a basis for two-part universal coding [29] as well as an intuitive justification of Rissanen's stochastic complexity measure of $\frac{M}{2} \log |\mathcal{L}|$ [110]. A similar dependence might hold for continuous densities.

example is shown in Figure 2.4.2(c), where the components and mixing parameters have been selected in the manner to be described in Section 2.6. It can be seen that the mixture model does a better job of characterizing the source than either the histogram or the kernel estimates, by both summarizing and generalizing the training sequence well.

Historically, the main use of mixture models has been in classification and clustering, wherein the structure of the model is regarded as reflective of the physical process by which the data were generated or observed [76]. An alternative view of mixtures is as general-purpose semiparametric density estimators, not necessarily reflective of the structure of the data source [37, p. 118ff]. This interpretation, which is the one adopted here, has been steadily growing in popularity in recent years, particularly among researchers interested in artificial neural networks [51, 9]. However, the important distinction between density estimation and function approximation is often blurred in that literature; see [95] for a brief discussion.

For general-purpose density estimation, a reasonable and popular choice for the form of the component densities is the Gaussian PDF (2.2.4), which can be written in mixture-specific notation as

$$\hat{f}_{\mathrm{mix}}(\boldsymbol{z}|m) = (2\pi)^{-D_z/2}|K_m|^{-1/2}\exp[-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu}_m)^{\mathrm{T}}K_m^{-1}(\boldsymbol{z}-\boldsymbol{\mu}_m)],$$

As will be discussed in Section 2.6, there are often advantages to restricting the covariance matrices to be diagonal. The resulting components are termed *separable*, and may be written

$$\hat{f}_{\mathrm{mix}}(\boldsymbol{z}|m) = \prod_{d=1}^{D_z}\hat{f}_{\mathrm{mix}}(z_d|m),$$

where $\hat{f}_{\mathrm{mix}}(z_d|m)$ is a one-dimensional Gaussian as given by (2.2.3), with mean $\hat{\mu}_{m,d}$ and variance $\hat{\sigma}^2_{m,d}$. Henceforth, mixtures components will be assumed to be separable Gaussian densities unless otherwise stated.

Estimating the mixture components and mixing probabilities is in general a complex task. Several approaches have been proposed over the years [35, 40, 76, 104, 135], and the topic continues to be an active area of research today. Much of the work has focused on low-complexity mixtures. When mixtures of high complexity are required or allowed, as they are here, robustness and computational complexity of the training process become important considerations. In the next section, a heuristically justified method of mixture estimation based on clustering is presented. Following that, an iterative ML-seeking approach will be discussed.

2.5 Mixture Estimation via the Generalized Lloyd Algorithm

As mentioned in the previous section, a mixture density estimate can be viewed as an economized kernel estimate in which the kernel locations are chosen to be representative of the training data. Selecting a set of representative points for a given set of data is precisely the problem of codebook design in vector quantization. It is natural therefore to consider the adaptation of algorithms normally used for codebook design to the task of mixture density estimation.

The *generalized Lloyd algorithm* (GLA) [45], which is the best known[10] and perhaps most widely used of these algorithms, involves iterating the following two steps after initializing the codebook:

(1) Assign each training point to the nearest entry in the current codebook, and

(2) update the codebook by replacing each entry with the centroid of the points just assigned to it.

The word "nearest" is used in the Euclidean sense. It can be shown that the sum of the squared distances between training points and their nearest codebook entries does not increase as the algorithm progresses, and in fact converges (to some value) in a finite number of iterations [45, pp. 363–368]. Once the codebook has been designed, it can be used for mixture density estimation in a straightforward way. In particular, a component density can be fit to the portion of the training data that falls into each cell in the nearest-neighbor partition induced by the codebook, and the mixing parameters can be taken to be the normalized cell populations. When the mixture component densities are separable Gaussians, fitting can be accomplished simply by computing the sample means and variances in each dimension separately for the m^{th} cell in the induced nearest-neighbor partition, and taking these as $\hat{\mu}_{m,d}$ and $\hat{\sigma}^2_{m,d}$, respectively. It has been found that slightly better performance (likelihood) often results by setting $\hat{\sigma}^2_{m,d}$ to be slightly larger than the sample variance; see [95, Appendix B] for a possible explanation.

The GLA was the real "work horse" algorithm for most of our earlier work in applying mixture models to texture and image processing applications [96, 97, 95]. Later, an improvement was generally obtained in all applications by following the GLA with a few iterations of the computationally more expensive expectation-maximization (EM) algorithm (to be described in Section 2.6), but

[10] In the clustering literature, this algorithm has come to be known as *k-means*. However, the term "generalized Lloyd algorithm" may be more appropriate, for the reasons cited in [45, p. 362].

the degree of improvement was usually slight, e.g., not more than about two tenths of a bit per pixel when applied to the lossless compression of natural images.

The main difficulty we encountered in running the GLA was the well-known problem of the assignment step occasionally resulting in one or more empty cells. Two methods described in [45] for handling this situation were tried. Both involved replacing the empty cell's codebook entry with a slightly perturbed version of another entry whose corresponding cell was deemed to be a good candidate for splitting into two. In one method, the most populous cell was split; in another, the one with the greatest sum of squared distances from member points to its centroid was split. Neither method worked perfectly in all cases, but the latter was found to be the more reliable of the two in getting the algorithm to use all of the allocated components.

A *partial distance search* [45, pp. 479–480] technique was employed to expedite the assignment step, resulting in significant computational savings (typically, a factor of 2) for large M. The philosophy behind this technique was also adapted to the task of speeding up the EM algorithm, as described in Section 2.6 and more fully in Appendix A.

Initialization of the codebook (cluster locations) for the GLA appears to be somewhat less critical than for EM. The main goal is to have the initial cluster locations well spread out over the region of space occupied by training points. Having them spread out in this way makes it relatively unlikely that a Voronoi cell will be empty during the early iterations, and may speed convergence by possibly reducing the length of the migration path to be followed by each cluster as the algorithm converges. Two commonly used initialization strategies are random assignment to training points and LBG splitting [45]. We have found that random assignment often leads to empty cells, possibly because the initial clusters are occasionally insufficiently spread out. Traditional LBG splitting is appropriate when the number of clusters is a power of two, because each split doubles the number of clusters. Here, it is desired that any number of clusters be allowed. The following initialization was found to yield consistently reasonable initializations for any number of clusters less than the number of training points. Begin by setting the collection of cluster locations to contain only one element: a vector selected at random from the training set. Then, take as the location of the next cluster a training point that maximizes distance to the nearest cluster among the subset already determined. Repeat this until the desired number of clusters has been obtained. It should be clear that this technique results in a set of locations that approximately minimizes the maximum distance of each training point to its nearest cluster. To be able to find a set of locations that is truly locally optimal in this minimax sense, it is sufficient to be able to find the smallest hypersphere containing a given set of points. For if this can be accomplished, then the desired local optimum could be found by substituting the center of the minimum sphere containing a cluster's member points for the centroid in each iteration of the GLA. Unfortunately, an efficient algorithm does not seem to exist for finding such a sphere.

While use of the GLA is suggested heuristically by the desire to center the mixture components on representative points, generally it does not result in the adopted criterion of likelihood being maximized. Direct solution of the mixture likelihood equations appears to be hopelessly intractable, even for relatively small mixtures in low dimensions [104]. A possible numerical approach is gradient search [72], but the gradients involved are complex enough to make this technique computationally unattractive for large M.

2.6 Mixture Refinement via Expectation-Maximization

The EM algorithm [9, 31, 104] provides an alternative gradient-free means of locally maximizing the likelihood over the mixture parameters, and has come to be a standard method of parameter estimation in mixtures and related models. When applied specifically to mixture estimation, the EM algorithm amounts to a softened version of the generalized Lloyd algorithm. In particular, the hard assignment of each training point to a single cluster that is carried out during each iteration of the GLA is replaced in EM by a weighted assignment to all clusters, with the weights corresponding to the current estimate of the posterior PMF over clusters for that training point. This soft assignment constitutes what is called the *expectation* step. The other step, *maximization*, involves re-estimating all of the parameters. Let $\{\hat{P}_{\text{mix}}^{(i)}\}$, $\{\hat{\mu}_{m,d}^{(i)}\}$, and $\{\hat{\sigma}_{m,d}^{2\,(i)}\}$ denote the mixture parameters at the beginning of the i^{th} iteration. The EM update rule [104, 9] can be written as

$$\rho_m^{(n)} = \frac{\hat{P}_{\text{mix}}^{(i)}(m) \prod_{d=1}^{D_z} \hat{f}_{\text{mix}}(z_d^{(n)} \mid \hat{\mu}_{m,d}^{(i)}, \hat{\sigma}_{m,d}^{2\,(i)})}{\sum\limits_{m'=1}^{M} \hat{P}_{\text{mix}}^{(i)}(m') \prod_{d=1}^{D_z} \hat{f}_{\text{mix}}(z_d^{(n)} \mid \hat{\mu}_{m',d}^{(i)}, \hat{\sigma}_{m',d}^{2\,(i)})}, \quad 1 \le n \le |\mathcal{L}|,\ 1 \le m \le M; \tag{2.6.1a}$$

$$\hat{\mu}_{m,d}^{(i+1)} = \sum_{n=1}^{|\mathcal{L}|} \frac{\rho_m^{(n)} z^{(n)}}{\sum\limits_{n} \rho_m^{(n')}}, \qquad 1 \le d \le D_z,\ 1 \le m \le M; \tag{2.6.1b}$$

$$\hat{\sigma}_{m,d}^{2\,(i+1)} = \sum_{n=1}^{|\mathcal{L}|} \frac{\rho_m^{(n)} [z^{(n)} - \hat{\mu}_{m,d}^{(i+1)}]^2}{\sum\limits_{n} \rho_m^{(n')}}, \qquad 1 \le d \le D_z,\ 1 \le m \le M; \tag{2.6.1c}$$

$$\hat{P}_{\text{mix}}^{(i+1)}(m) = \frac{\sum_{n=1}^{|\mathcal{L}|} \rho_m^{(n)}}{\sum\limits_{m'=1}^{M} \sum_{n=1}^{|\mathcal{L}|} \rho_{m'}^{(n)}}, \qquad 1 \le m \le M; \tag{2.6.1d}$$

where $\rho_m^{(n)}$ is an intermediate quantity that represents the strength of membership of the n^{th} training point to the m^{th} mixture component. Expression (2.6.1a) constitutes the expectation step, while the remaining expressions in (2.6.1a) constitute the maximization step. It is readily seen that if a nearest-$\boldsymbol{\mu}$ indicator function is substituted for $\rho_m^{(n)}$, then the EM algorithm becomes the generalized Lloyd algorithm; hence the interpretation of EM as a soft version of the GLA. A consequence of the softening is that in EM, every training point effectively belongs to every cluster, making the computational cost of EM about M times that of the GLA.

In practice, particularly when M is large, it is almost always the case that only a small fraction of the cluster-membership weights for a given training point turn out to be significant. Thus, the majority of terms in the likelihood calculation and the maximization-step sums are usually negligible, and may be safely omitted without appreciably affecting either the final likelihood or the final mixture parameter estimates. Properly exploited, this can result in substantial savings in the maximization step. The rub is that discovering which terms may be safely omitted is itself a potentially expensive procedure.

A method for taking advantage of this potential savings is now proposed. Computationally inexpensive discovery of which terms are insignificant may be accomplished in a manner analogous to the partial search mentioned briefly in Section 2.5. In that technique [45], to speed up the search for the codebook entry or cluster nearest to a given training point, the partial Euclidean distance is tested after each coordinate's contribution is accumulated into the sum, and a candidate-nearest cluster is eliminated from further consideration as soon as it is discovered that the partial sum exceeds the minimum of the previously computed distances. An analogous procedure for early discovery and elimination of negligible clusters in EM would require that the posterior probability of cluster membership for a given training point be suitably upper-bounded in terms of an appropriate incrementally computable quantity. In the case of separable mixture components, a suitable upper bound may be given in terms of the partial likelihood, and this upper bound may be tested sequentially after each coordinate's contribution is accumulated. If, after a particular coordinate is processed, it happens that the bound falls below a specified threshold, then the candidate cluster can be deemed insignificant without any need to process the remaining coordinates, resulting in computational savings (the additional cost of the comparison operations is usually more than offset by the resulting savings in likelihood computation). A difference between this procedure and the partial distance method used in the assignment step of the generalized Lloyd algorithm is that the latter does its job with no approximation; it finds the nearest cluster. However, it does not estimate the posterior PMF. In contrast, the proposed technique does estimate the posterior PMF, but does so only approximately.

A slight complication arises from the need to adjust the significance threshold for each cluster according to the partial training-point likelihood from previously processed clusters, so that the amount by which the resulting approximate likelihood underestimates the exact likelihood may be strictly bounded. A new, economized EM algorithm based on this strategy, suitable for estimating mixtures having separable components, is described more fully in Appendix A.

Two major difficulties were encountered in applying the EM algorithm to mixtures having many components. The first has already been mentioned: high computational complexity. The economized version of the EM algorithm described in Appendix A results in modest savings, but the execution time per iteration is still several times that of the GLA. Therefore, the use of EM was mostly restricted to a few iterations (seldom more than 10) to refine a mixture estimate obtained via the GLA.

The second difficulty was EM's propensity to overfit the data by shrinking the variances down to zero on components that coincide with training points, causing the likelihood to increase without bound. Crossvalidation can detect this, but detection alone does not provide a solution, since this sort of overfitting is not necessarily a symptom of excessive model complexity. For a given level of complexity, there might well be (and usually is) a better, less-overfit estimate that EM missed along the way to finding the local degenerate optimum. One workable solution is to place a lower limit on the allowable variance in each dimension, which is a form of regularization. However, when this is done, it is no longer clear precisely what criterion EM is attempting to optimize. A completely satisfactory treatment of this problem does not appear in the literature, and in light of the comments in Section 3 of [104] regarding the conceptual and practical difficulties associated with ML mixture estimation, perhaps one should not be expected. The approach settled upon in this thesis was pragmatic. A lower limit of 1/1000 of the maximum coordinatewise sample variance of the training data was enforced on the variances in all dimensions to prevent severe overfitting. It is recognized that the resulting estimate is unlikely to be optimal in any meaningful sense when the resulting estimate lies on this variance constraint boundary. Nevertheless, the technique resulted in mixture estimates that performed well in their intended applications, as will be described in the subsequent chapters.

Although the use of unconstrained (i.e., possibly nonseparable) Gaussian mixture components might allow an estimate of a given complexity to fit the data better, doing so could introduce difficulties with overfitting. This is because there are many more ways in which the determinant of the covariance matrix can vanish, and many more degrees of freedom in the model. The number of degrees of freedom in each covariance matrix increases from D_z to $D_z(D_z+1)/2$, resulting in correspondingly less reliable estimates of the covariance matrices given a fixed amount of training data. It is clear that in situations where there is linear dependence in the data, allowing nonseparable components can be advantageous. However, frequently such linear dependence is global, and as such can be removed simply by rotating the coordinate system by appropriate linear preprocessing. For these reasons, it is not clear whether allowing arbitrary covariances results in any real advantage in general. Investigation of this issue in depth is left as a topic for future research. Only separable-component mixtures were employed in the experiments to be described in this thesis.

2.7 Conditional Density Estimation using Finite Mixtures

In many situations, what is needed is an estimate of the conditional density $f(y \mid \boldsymbol{x})$ rather than of the joint density $f(\boldsymbol{x}, y)$. These two entities are of course related by the formula

$$f(y \mid \boldsymbol{x}) = \frac{f(\boldsymbol{x}, y)}{\int_{-\infty}^{\infty} f(\boldsymbol{x}, y')dy'}, \tag{2.7.1}$$

which is valid whenever $f(\boldsymbol{x}) \neq 0$.

When estimates rather than the true densities are substituted into (2.7.1), the formula is no longer an identity, but rather a recipe for obtaining a conditional density estimate from a joint one. For example, when the mixture (2.4.4) is used for the form of $\hat{f}(y \mid \boldsymbol{x})$, the conditional density estimate suggested by 2.7.1 becomes

$$\begin{aligned}\hat{f}_{\text{mix}}(y \mid \boldsymbol{x}) &= \frac{\sum_{m=1}^{M} \hat{P}_{\text{mix}}(m)\hat{f}_{\text{mix}}(\boldsymbol{x}, y \mid m)}{\int \sum_{i=1}^{M} \hat{P}_{\text{mix}}(i)\hat{f}_{\text{mix}}(\boldsymbol{x}, y' \mid i)dy'} \\ &= \sum_{m=1}^{M} \frac{\hat{P}_{\text{mix}}(m)\hat{f}_{\text{mix}}(\boldsymbol{x} \mid m)}{\hat{f}_{\text{mix}}(\boldsymbol{x})}\hat{f}_{\text{mix}}(y \mid \boldsymbol{x}, m) \\ &= \sum_{m=1}^{M} \hat{P}_{\text{mix}}(m \mid \boldsymbol{x})\hat{f}_{\text{mix}}(y \mid \boldsymbol{x}, m),\end{aligned} \tag{2.7.2}$$

which simplifies to

$$\hat{f}_{\text{mix}}(y \mid \boldsymbol{x}) = \sum_{m=1}^{M} \hat{P}_{\text{mix}}(m \mid \boldsymbol{x})\hat{f}_{\text{mix}}(y \mid m) \tag{2.7.3}$$

when the mixture components are restricted such that y is conditionally independent of $\boldsymbol{x}$ given m, as is true for example when the components are separable. Note that the conditional density estimate is itself a mixture, where the component densities are one-dimensional and the mixing parameters are the posterior probabilities of cluster membership after observation of $\boldsymbol{x}$.

At issue is the quality of such an estimate of the conditional density when the GLA or EM is used to train the joint density estimate. A simple example will illustrate that for a fixed number of mixture components M, approximately maximizing the joint likelihood via the GLA or EM does not make the best possible use of the available complexity from the point of view of conditional density estimation.

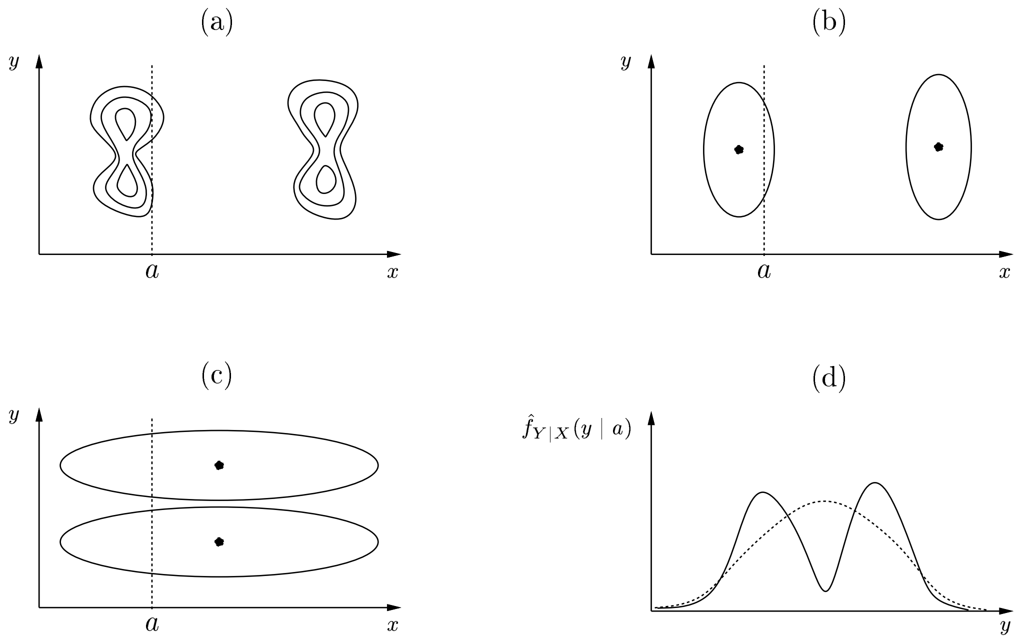

Figure 2.7.4: Failure of EM to capture conditional structure. A density having the equiprobability contours shown in (a) is fit by a two-component Gaussian mixture using the EM algorithm as indicated in (b). The corresponding conditional density estimate along the line $x = a$, shown as the dashed line in (d), misses the bimodal conditional structure. Using the mixture components indicated in (c) would result in a better conditional density estimate (the solid line in (d)), although in this case the components are centered on low probability regions, which are shunned by EM.

Consider a true distribution having the equiprobability contours shown in Figure 2.7.4. For a Gaussian mixture of allowed complexity $M = 2$, the GLA or EM would result in the components indicated in Figure 2.7.4(b). The corresponding conditional estimate $\hat{f}_{\text{mix}}(y \mid \boldsymbol{x})$ evaluated along the indicated line $x = a$ is unimodal, as indicated by the dashed line in Figure 2.7.4(d). A clearly better choice for the components would have been as indicated in Figure 2.7.4(c), resulting in the desired bimodal distribution indicated by the solid line in Figure 2.7.4(d). However, these are centered on regions having almost zero probability, so that neither the GLA nor EM would ever find such a solution, or even consider such a solution to be desirable.

The question is how such a solution can be found in an automatic way, and for problems of arbitrary dimensionality. Qualitatively, one difficulty is that maximizing joint likelihood tends to focus the model's resources on the dependent and independent variables evenly, even when some of the independent variables are irrelevant (or conditionally irrelevant, given others) to the distribution of the dependent variable.

It is important to note that the mixture components indicated in Figure 2.7.4(c) could be shifted to the left or right, and their widths increased or decreased, without appreciably damaging the quality of the estimate $\hat{f}_{\mathrm{mix}}(y \mid \boldsymbol{x})$. This implies that the conditional log-likelihood surface does not have a distinct localized peak in the space of mixture parameters; instead, the optimal region is relatively broad, and in general includes parameter values that may not be desirable on practical grounds. For example, infinite widths for the ellipses in Figure 2.7.4(c) would be acceptable from the point of view of conditional likelihood, but not from the point of view of implementation. Thus, any direct search would have to be constrained or regularized in some way. An approach based on a three-layer neural network implementation of a mixture is considered in [9]. It requires careful parameterization of the mixture parameters, and its apparent complexity indicates that it may not scale well to large problems.

Since the EM algorithm is well-suited to mixture estimation for joint densities, it is reasonable to ask whether it might be applied fruitfully to conditional density estimation. EM is traditionally used when some of the data can be regarded as missing. When estimating joint densities using mixtures, the mixture component labels are taken as the missing data. However, when estimating conditional densities, taking the "true" component labels to be the missing data can result in an inefficient use of mixture resources, as shown in the example of Figure 2.7.4. It is therefore not clear what expectation should be computed in the E-step. A number of *ad hoc* modifications to the E-step have been tried in joint work with Tony Jebara with limited success; see Section 3.9. Based on these experiences and on discussions with Michael Jordan [65], it seems to me unlikely that a completely satisfactory EM algorithm for conditional-density mixture estimation will be found, but more research is required.

A plausible approach to the conditional-density mixture estimation problem is suggested if we take a step back and reconsider the problem once again in abstract terms. The possible values of $\boldsymbol{x}$ are the conditioning states within each of which it is sought to characterize the distribution of y. For continuous $\boldsymbol{x}$, it is not meaningful to consider these values in isolation; instead, it is necessary both in concept and in practice to impose some sort of grouping of conditioning values into categories. The grouping can be accomplished in either a hard or a soft manner. The structure of the conditional mixture density estimate (2.7.3), at least potentially, strikes a compromise between these two. A finite grouping into M categories is defined by the densities $\{\hat{f}_{\text{mix}}(y \mid m)\}$, and the final estimate $\hat{f}_{\text{mix}}(y \mid \boldsymbol{x})$ is obtained by linearly combining these using weights that depend on $\boldsymbol{x}$. This interpretation suggests that the first order of business is to determine a good choice for the M categories. That the weighting factors appear as posterior probabilities of category membership suggests that the categories themselves be subsets of the conditioning space for which the posteriors are both meaningful and practically estimable. One possibility is to take as the categories cells in a partition of the conditioning space. The immediate problem then becomes one of finding a partition for which the average conditional log likelihood is maximized. Recalling the close relationship between likelihood and entropy discussed in Section 2.3, an approximately equivalent formulation is: find a partition for which the conditional entropy of y is minimized.

This formulation is suggestive of a partioning technique proposed by Breiman, Friedman, Olshen, and Stone [13] as part of a tree-structured approach to classification and regression. In their methodology, here generically termed $CART^{®}$, a fine partition is initially grown by recursively splitting cells to achieve the greatest stepwise reduction in a specified empirical measure of *impurity*. Though not strictly required in principle, in most implementations the splits are required to be binary, and such that the splitting boundary is always perpendicular to one of the axes. The cells of the resulting partition correspond to the leaves of a binary tree, where the root node represents the original conditioning space, and each interior node represents an intermediate cell that is subsequently refined. Each splitting operation corresponds to a comparison of one of the conditioning coordinates with a threshold; the coordinate and threshold are jointly chosen to provide the greatest immediate reduction in impurity for the given cell being split. After the initial tree is grown, it is pruned in a bottom-up manner by selectively undoing some of the splits, to obtain a tree that is of an appropriate complexity for the data. Classification of a new observation is then accomplished by first finding out to which cell in the partition it belongs, then taking as the class label that which occurred most often (for example) in that cell in the training data. A similar procedure is used for regression, but the representative value is taken to be the sample average of the values assumed by the dependent variable among the training points falling into that cell.

One way to apply this technique to density estimation would be to treat the dependent variable as categorical rather than ordinal, and use a standard CART classification tree to estimate posterior class-membership probabilities using leaf histograms. The use of such trees to estimate posteriors for categorical data is actually discussed in some detail by Breiman et al. [13] (see Section 3.9 of this thesis), and a nearly identical approach is presented by Rissanen [107] in the context of conditional PMF estimation for universal data compression. The difficulty with such a strategy is that by quantizing the dependent variable and treating it as categorical, advantage is not taken of the *meaning* of its value. In particular, smoothness and metric space assumptions are not exploited, and this is too much to throw away.

In the early stages of development of the CART classification methodology, the empirical entropy was used as a substitute splitting criterion for the empirical misclassification rate, because the latter was found to be ill-behaved in a number of respects [13, pp. 94–98]. Later, other substitute criteria (notably, the Gini index) were used and even preferred. In at least one respect, then, matters are simpler for tree-structured density estimation than for classification: The natural criterion, empirical entropy, is known to be well-behaved and therefore a substitute need not be sought. However, in order to adapt the CART methodology to conditional density estimation, or more specifically to the problem of estimating mixture parameters for conditional density estimation, several important issues must still be resolved. These are discussed briefly below, and in more detail in Chapter 3.

First, a simple example is considered to illustrate how a tree-structured approach might work, and to set the stage for identifying the major issues. Consider the hypothetical mixture of six isotropic Gaussians, with the locations and variances as indicated by the equiprobability spheres shown in Figure 2.7.5(a). Data obeying this distribution are observed, and it is desired to estimate the conditional density of Y given any observation (x_1, x_2) using a four-component Gaussian mixture. Using EM results in the estimate (d), which misses the bimodal conditional structure in the foreground region. In contrast, the estimate (e) captures the conditional structure everywhere. To obtain it, two levels of axis-perpendicular splitting are performed. The first divides conditioning space into two half-planes by comparing x_2 with a threshold, separating the four foreground blobs from the two background ones. This encapsulates the conditional irrelevance of x_1 in the foreground. However, in the background cell, x_1 is still relevant; this is reflected by the choice of the next split, which is shown in (c). Finally, one component each of the four-component model is allocated and fit to the two rear cells, while the two remaining components are fit to the foreground cell. In this example, the mixture was allowed to have only four components. If six or more were allowed, then both methods could be made to fit the data perfectly.

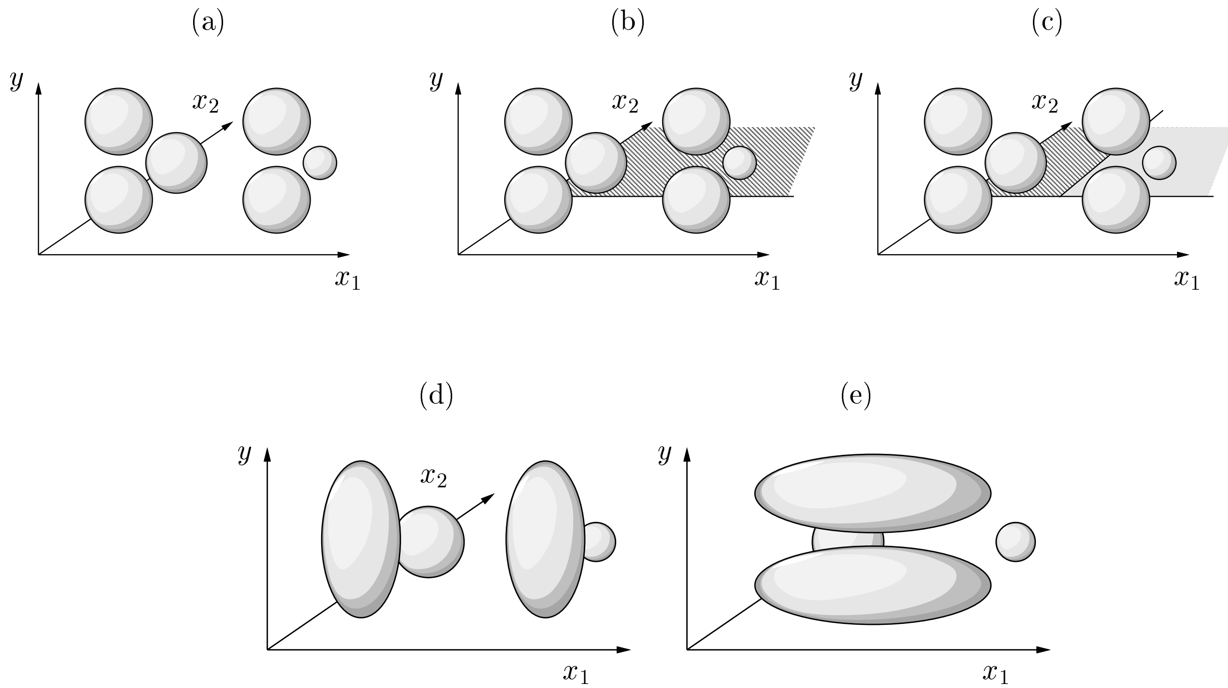

Figure 2.7.5: Fitting a four-component mixture model to a six-component distribution. The true distribution is shown in (a), and a corresponding four-component mixture density estimate obtained via EM is shown in (d). Two levels of recursive partitioning are shown in (b) and (c). A corresponding four-component mixture estimate is shown in (e). Joint likelihood is maximized by the EM result (d), while conditional likelihood by (e). Numerical results based on simulation for this example are presented in Table 3.8.1.

A number of issues must be resolved to make the method outlined in this example workable. First, unlike in the case of classification trees, the dependent variable is continuous rather than categorical, so that it is not so clear how empirical entropy or likelihood should be estimated when considering candidate splits or pruning. One answer might be that they should be computed with respect to the final estimated model that would have resulted from selecting each split, but that would involve training two mixtures (one for each subcell after splitting) for every candidate threshold considered for every coordinate, which would not be feasible computationally. The second issue is the determination of when to stop splitting. Here, the CART strategy carries over intact: split until an extremely large tree is obtained, then prune back to an appropriate size using weakest-link cutting with respect to a cost-complexity tradeoff, where cost in this case is measured in terms of crossvalidated likelihood. A final issue is the determination of appropriate leaf-conditional density estimates. Since the leaf cells are constrained to be rectangles by the splitting process, little would be gained by allowing anything other than unimodal separable densities for the conditioning-space dimensions, or by allowing the conditioning-space characteristics for multiple components allocated to a single cell to differ. Consequently, it is reasonable to stipulate that the leaf-conditional densities be expressible as the product of a separable unimodal density in the conditioning coordinates with a mixture in the dependent variable. The question then remains of how to best determine the appropriate complexity of the dependent-variable mixture. In this example, two components were selected for the foreground cell and one for each of the two background cells, but in general an automatic procedure for such complexity allocation is needed. Each of these issues will be addressed in the next chapter.

2.8 Chapter Summary

Background information about subband-domain processing and probabilistic modeling was provided. Joint and conditional density estimation were considered in both abstract and practical terms, and various approaches were discussed and compared. Special attention was focused on mixture density models because of their potential to model arbitrary distributions. For such models, it was shown that good joint density estimates cannot always be used to obtain good conditional ones. Fitting of the mixture for the purpose of conditional density estimation requires specialized training; in particular, the standard EM algorithm is not well suited to the task. A suitable training method will be proposed in the next chapter.

CHAPTER 3:

Partitioned Conditioning via Decision Trees

A tree-structured methodology for conditional density estimation, dubbed *partitioned conditioning via decision trees* (PCDT), is described in detail in this chapter. First, some useful notation and terminology specific to trees will be introduced.

3.1 Trees: Basic Concepts, Notation, and Terminology

Notation will closely follow the conventions established in [13], with some minor additions and modifications. Towards making the presentation self-contained, discussion begins with some basic definitions.

A *directed graph* is an ordered pair (Γ, g), where $\Gamma = \{t_i\}$ is a collection of primitive elements called *nodes*, and $g : \Gamma \times \Gamma \to \{0, 1\}$ is a boolean function defined on pairs of nodes.[11] Let t and t' be any two nodes in Γ. If $g(t, t') = 1$, then we say that a *directed arc* connects the two nodes, originating in t and terminating in t'. Conversely, if $g(t, t') = 0$, then we say that no such arc exists. The notion of "arc" is particularly useful when depicting and manipulating directed graphs pictorially. A *path* is a connected sequence of one or more directed arcs, and the *length* of a path is defined as the number of directed arcs that constitute it. A directed graph is termed *acyclic* if, for every node $t \in \Gamma$, there does not exist a path both originating and terminating in t. The *fan-out* of a node $t \in \Gamma$ is defined as the number of arcs that originate in t; similarly, the *fan-in* is defined as the number that terminate in t.

A *tree* is a directed acyclic graph in which each node has a fan-in of unity, except for exactly one node, called the *root*, which has a fan-in of zero. A node that has a fan-out of zero is termed a *leaf*. A tree will generally be denoted by the symbol T, its root node by $\lambda(T)$ or simply λ if T is understood, and the set of its leaf nodes by the notation $\tilde{T}$. Nodes that are not leaves are termed *interior* nodes. A K-*ary tree* is one in which each node has a fan-out of at most K, and a *complete* K-*ary tree* is one in which each interior node has a fan-out of exactly K. A directed graph or tree that has a finite number of nodes is termed *finite*.

Let $T = (\Gamma_T, g_T)$ be a tree. The notation $t \in T$ will be used as a shorthand for $t \in \Gamma_T$. For any two nodes $t, t' \in T$, if there exists a path originating in t and terminating in t', then that path

[11] If Γ is finite, then g can be specified by an incidence matrix.

is unique (because the fan-in at each node is at most unity), and t is said to be an *ancestor* of t'. This relationship is indicated symbolically as $t < t'$. Note that the root node is an ancestor of every other node, and that the ancestor relation $<$ is a strict partial order on Γ_T (see [79], p. 69ff). The *depth* of a node is defined to be the length of the path from the root to that node. A finite complete K-ary tree in which all of the leaves have the same depth is termed *balanced*. The notation $t \leq t'$ indicates that either $t < t'$ or else $t = t'$. If t is an ancestor of t', then t' is said to be a *descendant* of t, indicated as $t' > t$. The notation $t \geq t'$ indicates that either $t > t'$ or else $t = t'$. In the special case where the path length between t and t' is unity, t is termed the *parent* of t', indicated as $t = t'_P$. In this case, t' is termed a *child* of t. In a complete binary tree, each interior node t has two children, a *left child* and a *right child*, denoted t_L and t_R, respectively. Thus, in a complete binary tree, $(t_L)_P = (t_R)_P = t$ for every interior node $t \in T$.

For any node $t \in T$, let Γ_{T_t} denote the set that consists of t and all of its descendants in T. Let g_{T_t} be the restriction of g_T to $\Gamma_{T_t} \times \Gamma_{T_t}$. Then the *branch* T_t, defined as $T_t = (\Gamma_{T_t}, g_{T_t})$, is the tree obtained by "snipping off" the arc coming into t, then taking t as the root. Similarly, the tree obtained by deleting or *pruning* T_t from T, denoted by $T - T_t$, can be defined precisely as the ordered pair $(\Gamma_{T-T_t}, g_{T-T_t})$, where Γ_{T-T_t} is the relative complement of Γ_{T_t} in Γ_T, and g_{T-T_t} is the restriction of g_T to $\Gamma_{T-T_t} \times \Gamma_{T-T_t}$. If a tree T' can be obtained from T by performing zero or more such prunings, then T' is called a *subtree* of T, and we indicate this relationship symbolically by $T' \preceq T$ or $T \succeq T'$. If both $T' \preceq T$ and $T' \neq T$, then T' is termed a *proper subtree* of T, and we indicate this by either $T' \prec T$ or $T \succ T'$.

3.2 Tree-Induced Partitions and the Hard PCDT Model

Henceforth, we restrict consideration to finite complete binary trees. For a given such tree T, we associate with each interior node t two numbers $d_t^* \in \{1, \ldots, D_x\}$ and $\tau_t^* \in \mathbb{R}$, termed the *splitting dimension* and *splitting threshold* respectively. These determine a recursive partitioning of conditioning space according to

$$
\begin{aligned}
&U_\lambda = \mathbb{R}^{D_x}; \\
&U_{t_R} = \{\boldsymbol{x} \in U_t : x_{d_t^*} \leq \tau_t^*\}; \quad U_{t_L} = U_t - U_{t_R}, \quad t \in T - (\{\lambda\} \cup \tilde{T}).
\end{aligned}
$$

For every node $t \in T$, we define $\mathcal{L}_t = \{(\boldsymbol{x}_t^{(i)}, y_t^{(i)})\}$ to be the subsequence of the training sequence $\mathcal{L} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}$ obtained by deleting each observation $(\boldsymbol{x}^{(n)}, y^{(n)})$ for which $\boldsymbol{x}^{(n)} \notin U_t$. Thus, for every subtree $T' \preceq T$, $\{U_t : t \in \tilde{T}'\}$ is a partition of $\mathbb{R}^{D_x}$ and $\{\mathcal{L}_t : t \in \tilde{T}'\}$ is a "partition"[12] of $\mathcal{L}$. We define the conditioning-to-leaf indexing function $\tilde{t} : \mathbb{R}^{D_z} \to \tilde{T}$ as

$$\tilde{t}(\boldsymbol{x}) = t \in \tilde{T} : \boldsymbol{x} \in U_t.$$

Finally, we associate with each leaf node $t \in \tilde{T}$ a one-dimensional density function $\hat{f}(y \mid t)$, and define the *hard PCDT* (PCDT-H) conditional density model as

$$\hat{f}(y \mid \boldsymbol{x}) = \hat{f}(y \mid \tilde{t}(\boldsymbol{x})). \tag{3.2.1}$$

This model is parameterized by the structure of the tree, the splitting dimensions and thresholds, and the leaf densities. We now describe the estimation of all of these from the training sequence. Later, it will be described how the model can be softened by mixing leaf densities in the fashion prescribed by (2.7.3).

3.3 Growing the Initial Tree

In the CART methodology, the splitting dimensions and thresholds are determined in a greedy fashion as the tree is initially grown, and the structure of the tree is finalized when this initial tree is pruned back to an appropriate level of complexity. Adapting this process to the conditional density estimation setting requires that the issues raised at the end of Section 2.7 be addressed.

We first consider growing the initial tree. The process begins by starting with a tree that consists of just the root, which is therefore also a leaf. For a given current tree T, a new tree is obtained by splitting each leaf node that satisfies a splitting eligibility criterion (which will be described later). The process is repeated until none of the leaves is deemed eligible to be split. The resulting maximum-complexity tree is denoted $T_{\max}$.

In this process, a leaf node t is split in the following way. For each dimension $d \in \{1, \ldots, D_x\}$ and candidate threshold value $\tau \in \Big\{x : x = x_{t,d}^{(n)} \text{ for some } n \in \{1, \ldots, |\mathcal{L}_t|\}\Big\}$, an estimate $\hat{\phi}(t, d, \tau)$ is computed of the negative partial log likelihood

$$\phi(t, d, \tau) = \left[- \sum_{n=1}^{|\mathcal{L}_{t_L}|} \ln \hat{f}(y_{t_L}^{(n)} \mid t_L) - \sum_{n=1}^{|\mathcal{L}_{t_R}|} \ln \hat{f}(y_{t_R}^{(n)} \mid t_R) \right] \Bigg|_{d^*=d;\, \tau^*=\tau} \tag{3.3.1}$$

where the dependence on d and τ is implicit in the definitions of $\mathcal{L}_{t_L}$ and $\mathcal{L}_{t_R}$ in terms of $\mathcal{L}_t$, d^*, and τ^*. Dividing ϕ by $|\mathcal{L}_t|$ provides an estimate of the empirical entropy as alluded to earlier, but

[12] The term is in quotes because $\mathcal{L}$ is a sequence, not a simple set. It is tempting to treat $\mathcal{L}$ as a set to simplify notation, but since some of its elements can repeat, it must be treated as a sequence.

referring to it in terms of likelihood emphasizes that it is as much a functional of the leaf density models as it is a function of the training sequence.

An estimate of ϕ must be used instead of the actual value because $\hat{f}(Y \mid t_L)$ and $\hat{f}(Y \mid t_R)$ have not yet been determined. Values of d and τ that jointly result in the least value of $\hat{\phi}$ are selected as d_t^* and τ_t^* respectively. Since this process involves searching over joint values of d and τ, the latter of which assumes values in a set of cardinality typically in the thousands, it is important from a practical standpoint that $\hat{\phi}$ be simple to compute. Of course, it is also important that it be a good estimate of ϕ.

Of several candidates considered, the choice for $\hat{\phi}$ finally settled on was the one obtained by substituting $\hat{f}_{\mathrm{H}}$ for $\hat{f}$ in (3.3.1), where $\hat{f}_{\mathrm{H}}$ is a leaf-conditional histogram density estimate for y, defined in several steps as follows. First, let $y_{\min}$ be defined as

$$y_{\min} = \min\Big\{ y : y = y_t^{(n)} \text{ for some } n \in \{1, \ldots, |\mathcal{L}|\} \Big\},$$

and let $y_{\max}$ be defined analogously. Let M_t^{H} be an integer histogram complexity parameter associated with leaf t. A suitable value of M_t^{H} must be determined for each leaf, as will be discussed shortly. The interval $[y_{\min}, y_{\max}]$ is partitioned into bins of uniform width

$$\Delta_t = \frac{y_{\max} - y_{\min}}{M_t^{\mathrm{H}}},$$

indexed via a binning function $q_t : [y_{\min}, y_{\max}] \to \{1, \ldots, M_t^{\mathrm{H}}\}$, defined as

$$q_t(y) = \begin{cases} \Delta_t^{-1} \lfloor y - y_{\min} \rfloor + 1 & \text{if } y_{\min} \le y < y_{\max}; \\ M_t^{\mathrm{H}} & \text{if } y = y_{\max}. \end{cases}$$

Letting $N_t(m)$ denote the number of observations $(\boldsymbol{x}_t^{(n)}, y_t^{(n)})$ in $\mathcal{L}_t$ for which $q_t(y_t^{(n)}) = m$, we estimate the leaf-conditional probability mass function over the bins by

$$\hat{P}_t(m) = \frac{N_t(m) + \xi}{|\mathcal{L}_t| + M_t^{\mathrm{H}} \xi}, \qquad m \in \{1, \ldots, M_t^{\mathrm{H}}\},$$

where $\xi \ge 0$ is a regularizer.[13] When $\xi > 0$, $\hat{P}_t$ is not an unbiased estimate of P_t, but when ξ and M_t^{H} are finite, it is asymptotically unbiased as $|\mathcal{L}_t| \to \infty$ and consistent because it converges to

[13] Note that omitting ξ yields the ML estimate, which is prone to overfitting. The choice $\xi = 1$ corresponds to a straightforward generalization of Laplace's rule of succession [85] to the multinomial setting, while the choice $\xi = 1/2$ corresponds to what has come to be known as the Krichevski-Trofimov estimator in the universal coding literature [142]. In a Bayesian setting, choosing ξ can be shown equivalent to selecting parameters for a Dirichlet prior on $\hat{P}_t$ [43], though such an interpretation is not essential. More will be said about ξ later.

the ML estimator. Finally, the desired leaf-conditional histogram density for Y is defined as

$$\hat{f}_{\mathrm{H}}(y \mid t) = \begin{cases} \Delta_t^{-1} \hat{P}_t(q_t(y)) & \text{if } y_{\min} \leq y \leq y_{\max}; \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the final form of $\hat{\phi}$ is

$$\hat{\phi}(t, d, \tau) = \left[- \sum_{n=1}^{|\mathcal{L}_{t_L}|} \ln \hat{f}_{\mathrm{H}}(y_{t_L}^{(n)} \mid t_L) - \sum_{n=1}^{|\mathcal{L}_{t_R}|} \ln \hat{f}_{\mathrm{H}}(y_{t_R}^{(n)} \mid t_R) \right] \Bigg|_{d^*=d;\, \tau^*=\tau} \tag{3.3.2}$$

where $\hat{f}_{\mathrm{H}}$ is as defined above. The main advantage of using the histogram density estimate instead of a mixture or kernel estimate for computing $\hat{\phi}$ lies in its property that for a given d, it is possible to arrange the search over τ so that recomputing $\hat{\phi}$ for each successive candidate τ involves merely shuffling one term from the second summation in (3.3.2) to the first. To arrange this, all that is essentially required is to sort $\mathcal{L}_t$ in ascending order of x_d once for each dimension d before searching over τ, which speeds up but does not affect the outcome of the search. It is interesting to note that if M_t^{H} were constant, then the function $\hat{\phi}(t, d, \tau)$ could be used to estimate information-divergence cost, which is listed by Chou [20, Appendix A] as satisfying conditions necessary for a particular method of fast search more general than the search for thresholds considered here.

The complexity parameter M_t^{H} must be optimized if $\hat{\phi}$ is to be a good estimate of ϕ, and this is where the regularizer ξ comes into play. Were ξ taken to be zero, then the likelihood would be optimized trivially by taking M_t^{H} to be as large as possible. On the other hand, it is easy (albeit tedious) to show that the best M_t^{H} for $\xi > 0$ is always finite. Crossvalidation (in either form mentioned in Section 2.3) isn't a viable solution, as it ends up requiring that no bins be empty in the leaf's training sequence that are not also empty in its test sequence, leading to an overly conservative and grossly suboptimal choice for M_t^{H}.[14] Moreover, the application of simple rules to determine M_t^{H} directly from $|\mathcal{L}_t|$ independently of the data, such as Sturges' rule [120]

$$M_t^{\mathrm{H}} = 1 + \log_2 |\mathcal{L}_t|,$$

work when $|\mathcal{L}_t|$ is sufficiently large, but this is precisely when they are not needed. For large $|\mathcal{L}_t|$ the exact value of M_t^{H} becomes unimportant, and may safely be set to a maximum resolution level that reflects the precision of the data.

It has been found experimentally using both synthetic and real image data that setting ξ to be anywhere in the range of 1.0 to 3.0 tends to result in reasonable values of M_t^{H} (as judged by

[14] Although it would be possible to combine regularization with holdout crossvalidation, regularization alone has been found to be adequate for the purpose of determining a suitable value of M_t^{H}.

the overall likelihood performance of the resulting PCDT model). The choice $\xi = 1$ (Laplace's rule) was arbitrarily selected for use in all subsequent experiments. For efficiency, the search over values of M_t^{H} was accomplished using a multiresolution grid search, which works because $\hat{\phi}$ tends to be approximately unimodal in M_t^{H}. To further improve computational efficiency in the initial stages of growth where the leaf populations tend to be large, the search was bypassed in favor of the heuristic rule

$$M_t^{\mathrm{H}} = \min\{\eta|\mathcal{L}_t|, M_{\max}^{\mathrm{H}}\}$$

whenever $|\mathcal{L}_t|$ exceeded a fixed population threshold $N_{\max}^{\mathrm{H}}$. Usually, the values $\eta = 0.1$, $M_{\max}^{\mathrm{H}} = 512$, and $N_{\max}^{\mathrm{H}} = 2000$ were used, but occasionally more liberal values were employed to speed program execution.

To complete the description of the growing of the $T_{\max}$, the criterion used to decide whether or not a particular leaf should be split is now specified. A leaf t is declared ineligible for splitting when either of the following conditions is satisfied:

(1) The values assumed by y in $\mathcal{L}_t$ lie in an interval of width less than ν;

(2) $|\mathcal{L}_t| < N_{\min}$.

The following values were usually found to be suitable: $\nu = 10^{-3}$ and $N_{\min} = 80$. Automatic determination of appropriate values from the data for these and other algorithmic parameters is left as a topic for future research.

3.4 Determination of Appropriate Tree Complexity (Pruning)

The overall complexity of the PCDT model is distributed among the complexity of the tree itself and the complexities of the individual mixtures that are fit to the leaves. There is a tradeoff to be made here. At one extreme, the tree would consist of a single node, and all of the complexity would be in the mixture assigned to that node. With the restriction (2.7.3) imposed by the PCDT approach, this would correspond to using a finite mixture to estimate the marginal density of Y, completely ignoring $\boldsymbol{X}$. At the other extreme, the tree might be taken to be so large that the resulting leaf populations would not support mixtures having more than a single component each. This would correspond to a regression tree in the usual sense (albeit based on an unusual splitting criterion), with the error explicitly modeled as Gaussian for each cell. For interesting distributions, the optimum is somewhere in the middle.

The question is how to determine the appropriate tradeoff between tree complexity and leaf-mixture complexity. The solution appears to be very difficult if we take the point of view that a given allotment of complexity is to be divided between the tree and the leaf mixtures. Fortunately, there is another approach. We can appeal directly to the likelihood criterion to get the tree to determine its own optimal complexity. Once this complexity has been established, the leaf complexities can then be optimized independently.

To determine the best tree complexity, we prune $T_{\max}$ back to a subtree using *minimum cost-complexity pruning*, as described in [13], taking crossvalidated likelihood as the cost criterion. Since the leaf mixtures have not yet been determined, we must approximate the likelihood with an estimate, just as when splitting. By assuming that the leaf mixtures would fit the data in the best possible way without overfitting, the estimate can try for the same goal, thereby achieving the desired approximation. We can therefore use the same criterion in pruning as was used in splitting, but with crossvalidation. Crossvalidation will also be used later in determining the best complexities for each leaf.

Note that $\hat{\phi}$ can be made to correspond to the mutual information functional listed by Chou et al. [22, Table I] by making a suitable choice of probability measures (namely, the regularized histogram estimates). Consequently, it satisfies the conditions necessary for the more general pruning algorithm described there to be used, wherein tree-complexity measures other than the number of leaf nodes can be employed. Investigation of more general pruning methods for tree-based conditional density estimation is a topic for possible future work.

To accomplish the crossvalidated likelihood estimation using the histogram estimate, the estimate needs to be regularized. Let $\mathcal{L}$ be the first 2/3 of the available data, and let $\mathcal{T}$ be the remaining 1/3. Growing $T_{\max}$ is done on the basis of $\mathcal{L}$; this results in a histogram (of varying complexity) for each node. We can use these histograms, with regularization, to obtain a crossvalidated likelihood estimate by evaluating the histograms on the leaf populations corresponding to $\mathcal{T}$. We then apply minimum cost-complexity pruning in the usual way [13, pp. 66–71], using the negative crossvalidated likelihood for cost. The tree with the best crossvalidated likelihood is selected.

The process is illustrated best by an example. Consider the six-component mixture of isotropic Gaussians having the parameters given in Table 3.4.2. The equiprobability spheres are shown in Figure 3.4.1(a). The true conditional differential entropy $h(Y \mid X_1, X_2)$ in nats is shown by

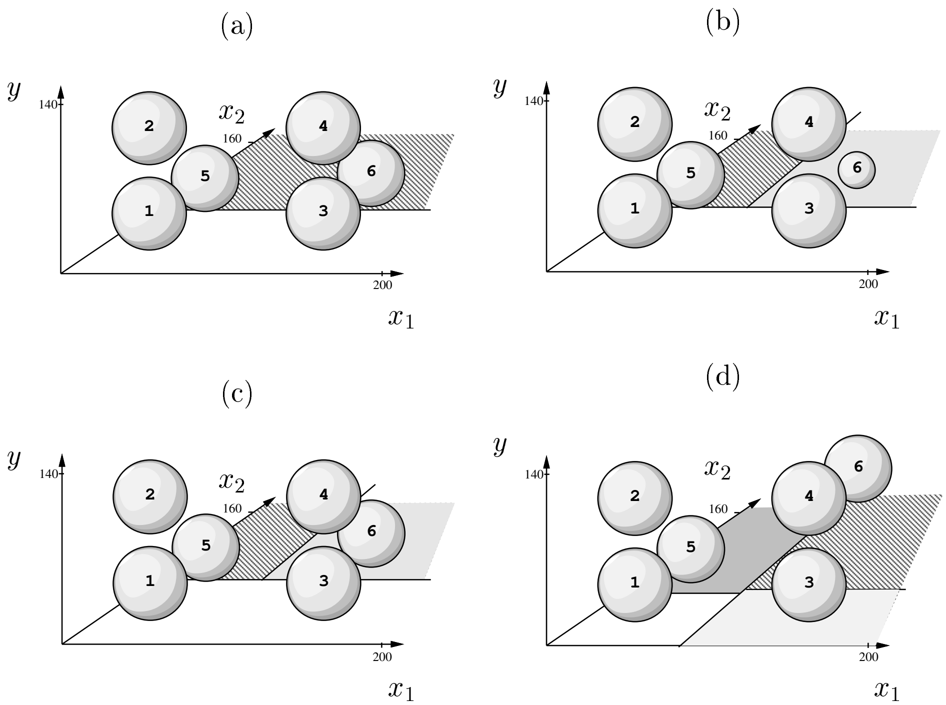

Figure 3.4.1: Four six-component Gaussian mixture densities, each differing in the parameters of the right-rear component, were used to generate 10,000 pseudorandom vectors each. The parameters for each are given in Tables 3.4.2 and 3.4.3. The resulting sequences were split into training and test sequences, using a 2/3-1/3 training/test ratio, and a PCDT model was trained on each. The tree-induced partition in each case is indicated qualitatively in the x_1-x_2 plane.

Table 3.4.2: Parameters for Mixture Density Shown in Figure 3.4.1(a)

m	$P(m)$	$\mu_1^{(m)}$	$\mu_2^{(m)}$	$\mu_Y^{(m)}$	$\sigma^{2\ (m)}$
1	1/8	64	64	64	100
2	1/8	64	64	128	100
3	1/8	192	64	64	100
4	1/8	192	64	128	100
5	1/4	64	128	64	100
6	1/4	192	128	64	100

Table 3.4.3: Parameter Differences from Table 3.4.2 for Figures 3.4.1(b-d)

Figure Part	$\mu_Y^{(6)}$	$\sigma^{2\ (6)}$
(b)	64	30
(c)	74	100
(d)	128	100

the finely dotted horizontal line in Figure 3.4.4(a). The average resubstitution (splitting) cost is indicated by the lower curve in the same graph, while the crossvalidated (pruning) cost is indicated by the upper curve. If a decision were to be made based on the lower curve, the maximum

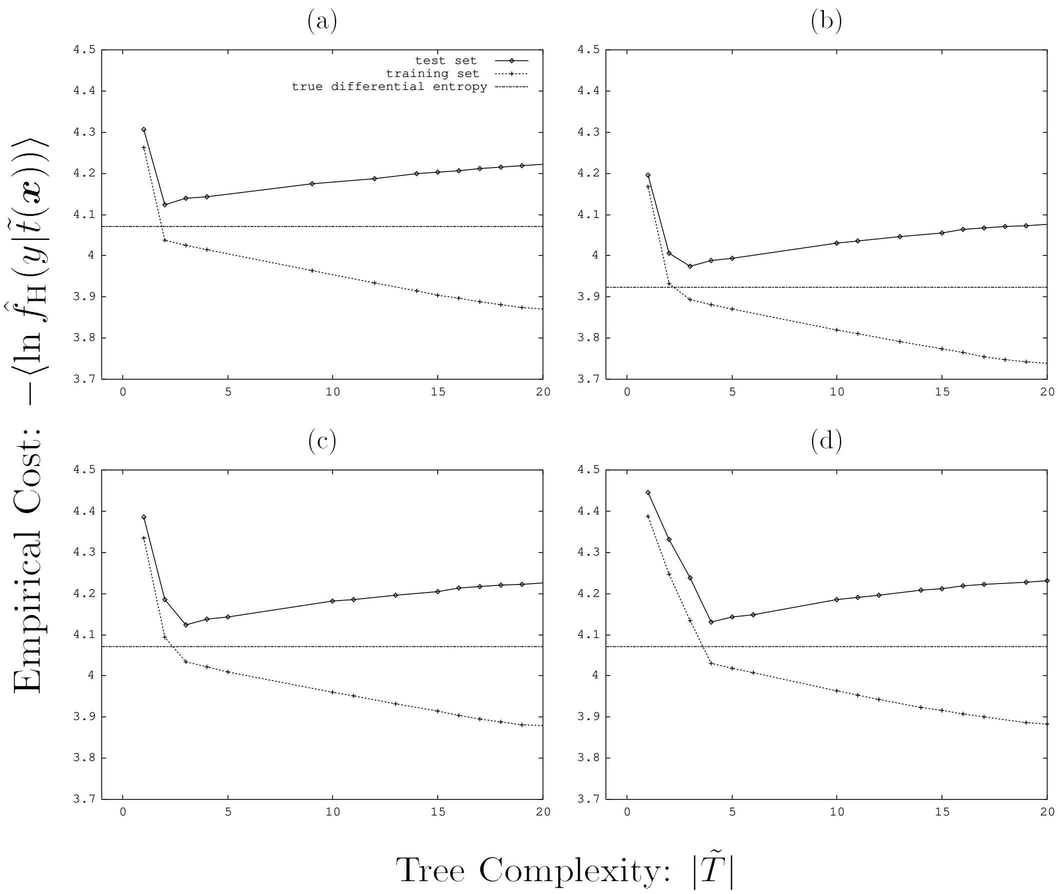

Figure 3.4.4: Empirical resubstitution (splitting) and crossvalidated (pruning) costs, measured as minus average log regularized leaf-histogram likelihood, versus tree complexity for training sequences sampled from each of the densities shown in Figure 3.4.1. The uneven spacing of points along the complexity axis reflects the fact that entire branches can be pruned off at once, and these vary in size.

complexity would be selected. The crossvalidated curve, however, achieves its minimum when the tree has two leaves; the corresponding cells are indicated by the foreground and background half-planes of the x_1-x_2 plane in Figure 3.4.1(a). As can be seen in both the resubstitution and crossvalidation curves, there is great advantage to making this split. All other splits suggested by the training data, however, were found not to hold up to crossvalidation's scrutiny.

To see how the tree adapts to changes made in this distribution, we consider adjusting the parameters of the right-rear (6^{th}) component Gaussian. First, we reduce its variance from 100 to 30, as indicated in row (b) of Table 3.4.3. This results in the same example that was considered when introducing mixture-based conditional density estimation in Section 2.7, and three leaves are obtained as expected, shown in Figure 3.4.1(b). The incremental numerical advantage of the second split can be seen in the crossvalidation curve of Figure 3.4.4(b). Note that a traditional regression tree would not have introduced this second split, because although the conditional distribution of Y is different in the left and right parts of the rear half-plane, the conditional mean is constant throughout that half-plane. This shows clearly by example that a simpler splitting cost function such as sample variance (mean-square regression error) could not have been used.

We next consider changing $\mu_Y^{(6)}$ instead of $\sigma^{2\ (6)}$. Specifically, we keep the variance at 100 but shift the component up by 10, as shown in Figure 3.4.1(c). This also creates an advantage to the second split, and a qualitatively identical partition results. The numerical advantage to this second split is actually slightly greater than for when the mean was the same but the variance differed, as is evident by comparing the graphs in Figures 3.4.4(b) and (c). Finally, as the component continues to migrate upward, there comes a point at which the structure of the tree catastrophically changes as shown in Figure 3.4.1(d). This example illustrates that the greedy approach to splitting sometimes misses good splits by not looking ahead. In this case, Y and X_2 are actually independent when a value for X_1 has not been specified, so that a top-level split on X_2 would result in no improvement in conditional likelihood. Therefore, the top-level split is on X_1, resulting in a left-right division. The next level of splits then divides each of these into a front and back part, resulting in a total of four leaves. It can be seen that the two front-most cells could be combined without hurting performance, but pruning the greedily split tree cannot do this. It would have been better had the first split been on X_2, as that would have resulted in an equally well performing tree using only three leaves (front, left-back, and right-back), but the greedy technique is not able to find such a tree. However, the likelihood cost of the final PCDT tree is comparable to that of the optimal tree, and the complexity is only slightly greater than what it should ideally be (four leaves instead of three). In real applications, the ideal tree complexity is unknown, so it is not as clear what the tree-complexity cost of greedy splitting is in actual practice. The size of the trees encountered in the real applications to be described in the subsequent chapters ranged from approximately 20 leaves to 400 leaves, with the depths after pruning ranging between approximately 4 and 10.

A final useful observation can be made from this example. As expected, the resubstitution estimates are consistently much better than the true differential conditional entropy, even for complexities near the optimum small values. The curves do not shoot off to negative infinity however, as they would if the regularization parameter ξ were set to zero. Equally interesting is the size of the gap between the lowest point on the crossvalidation curve and the differential entropy line. Had computational considerations not required the use of an easily (re)computable estimate of likelihood, then these could be made to shrink. For example, the PCDT performance data to be shown in Table 3.8.1 indicates that the gap in Figure 3.4.4(b) is made close to zero when a mixture is used instead of the histogram.

3.5 Fitting the Leaves

The differences discussed so far between the PCDT methodology and that of traditional CART can be traced almost entirely to the specialized requirements that PCDT places on the splitting and pruning criteria. Structurally and philosophically, the CART paradigm has thus far carried over largely intact, though it has been motivated from a different angle.

Fitting densities to the leaves of the pruned tree is where PCDT becomes substantially more complicated, and issues arise that have no parallel in the established CART methodology. In both classification and regression, the hard part for CART is finding the tree; once that is done, choosing the representative value (either the class label or the estimate of the dependent variable) is comparatively straightforward. In the conditional density estimation setting however, matters are different. All of the usual considerations that make density estimation a difficult problem apply to fitting the leaves, plus there is now the additional apparent complication that the complexities of the models must be appropriately balanced among the leaves.

As discussed previously, the leaf densities $\{\hat{f}(y \mid \tilde{t}(\boldsymbol{x}))\}$ in (2.7.3) are chosen to be finite mixtures of Gaussians in the PCDT approach, because of the expressive power and generality of such models. We have already visited the problem of fitting mixtures in Sections 2.5 and 2.6, and have noted that the task is generally nontrivial. We are now faced with the problem of fitting dozens and sometimes hundreds of mixtures in a single PCDT model. The one saving grace is that the densities are now one-dimensional, so that training is much easier than in the scenarios previously considered.

The approach finally settled on is now described; a flow chart is given in Figure 3.5.1. In this description, "cost" refers to average negative log likelihood. For each leaf t, mixtures ranging in complexity $M_t = 1, \ldots, M_{\max}$, where usually $M_{\max} = 25$, are each trained on a training set via the Lloyd algorithm,[15] with the number of iterations limited to i_{L}, typically 10. If one or more empty cells appear on an iteration, then that iteration is not counted towards the limit i_{L}, and an attempt to replace the cell(s) is made by splitting the cell(s) with the largest contribution to distortion. If i_{r} such attempts fail, then the Lloyd algorithm is aborted, and the complexity

[15] Not "generalized" here because it is applied to one-dimensional data, which was considered by Lloyd [70].

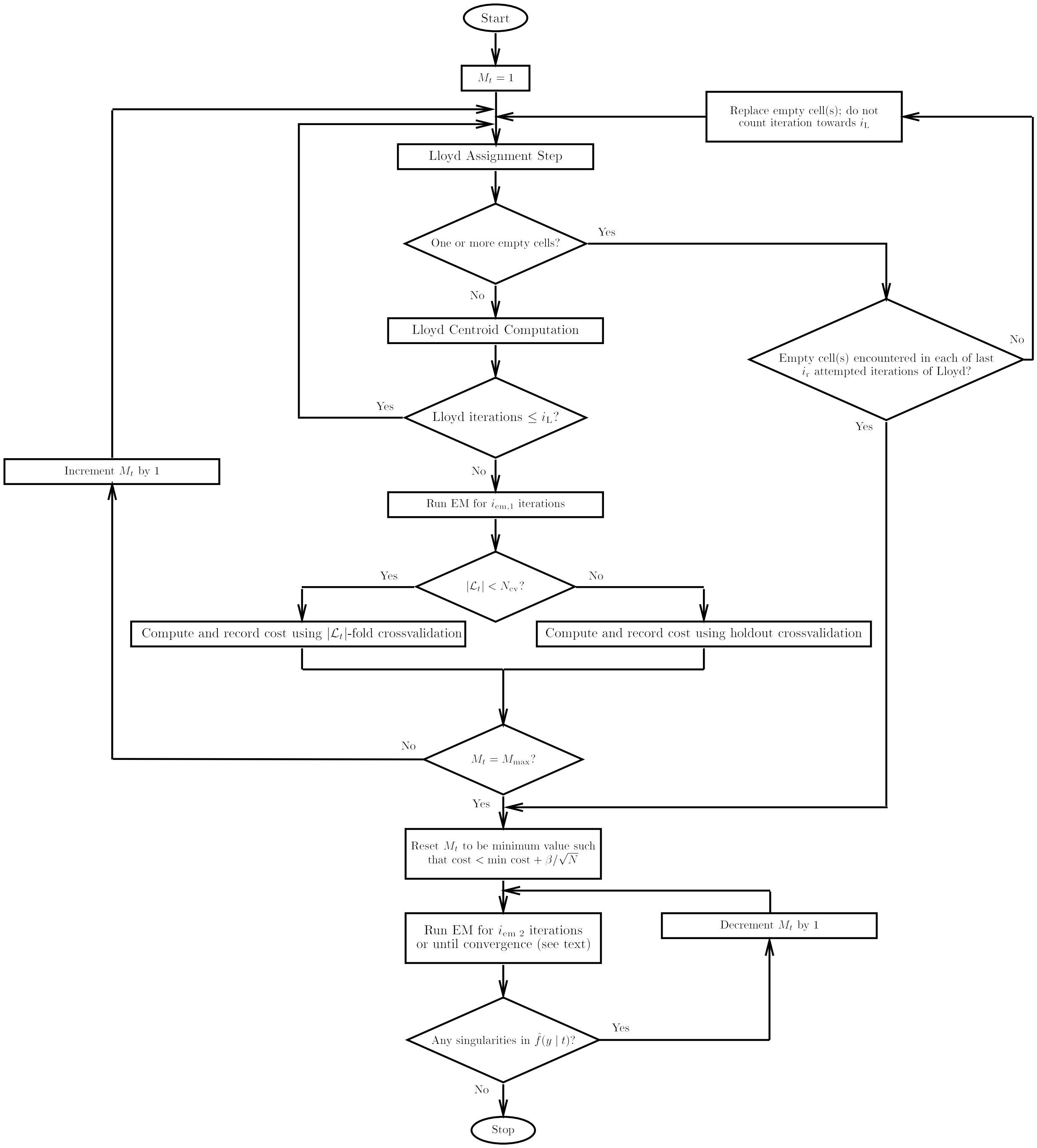

Figure 3.5.1: Flow chart of leaf-mixture complexity determination and fitting. See Section 3.5 for a detailed description and for the parameters values used in most of the experiments reported in this thesis.

search is also terminated. Usually, i_r is taken to be 10. If there are no empty cells or if the replacement succeeds, then the resulting mixture is handed off to the EM algorithm, which is allowed to run for $i_{em,1}$ iterations, where $i_{em,1}$ is usually set at 5. Next, the crossvalidated cost using the resulting model is calculated, using $|\mathcal{L}_t|$-fold crossvalidation when $|\mathcal{L}_t| < N_{\text{cv}}$ (normally 60), and (for computational efficiency) holdout crossvalidation otherwise. This value is recorded, and the next complexity tried, until the crossvalidated costs for all of the candidate complexities have been obtained. The minimum of these costs is then determined, and an acceptance threshold set at $\beta/\sqrt{N}$ above this minimum, where β is usually set at 0.3, and $N = |\mathcal{T}_t|$ when holdout crossvalidation is used and $N = |\mathcal{L}_t|$ when $|\mathcal{L}_t|$-fold crossvalidation is used. The smallest-complexity mixture whose crossvalidated cost is less than this acceptance threshold is then tentatively selected as M_t. The corresponding mixture is then refined by running the EM algorithm until one of the following conditions is satisfied: (1) the number of iterations exceeds $i_{\text{em},2}$ (usually 20); (2) the algorithm is judged to have converged because the per-observation log likelihood has increased by less than 10^{-5} nats; or (3) the mixture develops a singularity by shrinking one of the variances below a threshold σ^2_{min} (usually set at 10^{-3}). Termination because of condition (2) has been never been observed. If termination occurs because of condition (3), then successively lower complexities are tried until the EM algorithm no longer results in a singularity. The resulting mixture is taken as $\hat{f}(y \mid t)$. This completes the description of the leaf-density fitting procedure that was used in all of the experiments to be discussed henceforth.

Ideally (from the traditional statistical point of view), the acceptance threshold should be set according to the standard error of the empirical cost, but this would require assuming a particular distribution for y. Even if such a distribution could reasonably be assumed, deriving the standard error would be intractable, because it would require averaging over independent trainings of mixture models (treating the mixtures as random objects) as well as over test samples. However, because the observations are assumed to be independent, the square of the standard error can be assumed to be inversely proportional to the number of terms used in computing the crossvalidated cost; hence the formula for the acceptance threshold given in the previous paragraph. An alternative would be to estimate the standard error empirically, over many trainings. This was not attempted, because there seems to be little potential advantage in doing so, and the technique described above seems to work adequately.

The above procedure can be illustrated with a simple example. We consider fitting a mixture in the above manner to the leaf corresponding to the foreground cell in Figures 3.4.1(a)–(c). Conditioned on being in this cell, the true density of y is a mixture of two equally weighted and equal-variance Gaussians, one centered at 64 and the other at 128. Thus, the best complexity for the mixture assigned to this node t, given an infinitely long training sequence, would be $M_t = 2$.

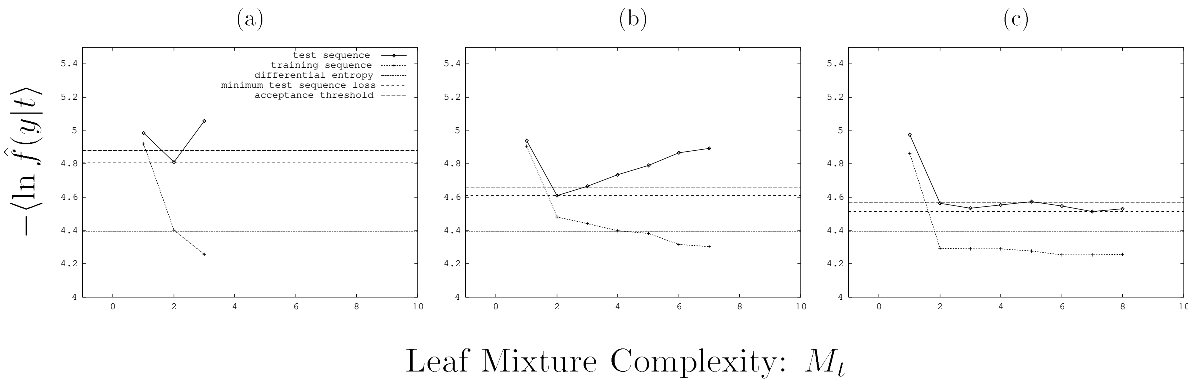

Figure 3.5.2: Example of how leaf mixture complexities are determined in PCDT. Samples of size 20 (a), 40 (b), and 80 (c) are drawn from the Gaussian mixture density corresponding to the first four components listed in Table 3.4.2, with the mixing probabilities appropriately renormalized. The empirical costs (resubstitution and crossvalidated) costs are shown, along with the theoretical optimum cost (the differential entropy), and the complexity acceptance band. The algorithm shifts from $|\mathcal{L}_t|$-fold crossvalidation to 2/3-1/3 holdout crossvalidation for computational efficiency reasons whenever $|\mathcal{L}_t|$ exceeds a threshold; in this case, the transition has occurred between (b) and (c), causing the acceptance band to widen to reflect a test sample size of $80/3 \approx 27$.

The graphs in Figure 3.5.2 show how the above procedure selects an appropriate level of complexity for a leaf mixture while simultaneously training the mixture. In the leftmost graph, $|\mathcal{L}_t| = 20$. Therefore, $|\mathcal{L}_t|$-fold crossvalidation is performed to make maximal use of this small sample in estimating cost. The minimum cost occurs when $M_t = 2$, and this is also the smallest complexity for which the corresponding cost lies below the acceptance threshold (in fact, it is the only one). In the middle graph, $|\mathcal{L}_t|$ has been increased to 40. Again, the complexity that minimizes cost is also the only complexity whose cost is below the acceptance threshold. Had the cost corresponding to $M_t = 3$ squeaked in under the threshold, $M_t = 2$ would still have been correctly chosen, since it is the smaller of the two. The rightmost graph is a bit more interesting. Here, the leaf population has been increased to $|\mathcal{L}_t| = 80$. Since this is above the cutoff for $|\mathcal{L}_t|$-fold crossvalidation of 60, 2/3-1/3 holdout crossvalidation is employed. As a result, the acceptance band has widened back almost to its width in the leftmost graph, reflecting the test sample size of $|\mathcal{T}_t| = 27$. This widening allows the value of $M_t = 2$ to once again be correctly selected. Note that the range of complexities which have nearly the minimum cost has broadened and hence more candidate complexities lie below the acceptance threshold. This broadening is to be expected; it simply reflects the fact that a more complex model does no worse than a less complex one unless it is overfit, and the availability of more data in (c) lessens the chance of overfitting. The actual minimizing value of $M_t = 7$ is rejected in favor of the correct choice $M_t = 2$, whose performance just makes it in under the threshold.

Table 3.5.3: Number of Possible Leaf Complexity Allocations

M	$\psi(M)$	$\ln \frac{\psi(M)}{\psi(M-1)}$
1	1	0
2	2	0.693147
3	5	0.916291
4	15	1.09861
5	51	1.22378
6	188	1.30462
7	731	1.35797
8	2950	1.39515
9	12235	1.4225
10	51822	1.44351

In some situations, it may be desirable to find the best PCDT estimate when the equivalent mixture complexity $\sum_{t \in \tilde{T}} M_t$ has been constrained in advance. In the procedure described above, each leaf node determines its own best mixture complexity M_t independently, according to the degree of complexity demanded by $\mathcal{L}_t$ and justified by crossvalidation. While it is not difficult to imagine modifications to the above procedure to accommodate such a prior constraint, it is interesting to consider the feasibility of a more substantial change that involves combining the pruning and leaf-complexity allocation procedures to meet the constraint directly. Assume that the tree has not yet been pruned, and that the total leaf complexity is constrained *a priori* to satisfy $\sum_{t \in \tilde{T}} M_t = M$ for some fixed $M > 0$. This constraint restricts the allocation to subtrees of $T_{\max}$ having at most M leaves, implicitly pruning away a potentially large portion of $T_{\max}$. The total number of possible allocations among all remaining feasible subtrees is given by the recursively defined function

$$\psi(M) = \begin{cases} 1 & \text{if } M = 0 \text{ or } M = 1; \\ 1 + \sum_{i=1}^{M-1} \psi(i)\psi(M-i) & \text{otherwise,} \end{cases}$$

which can be obtained by noting that there is one way to allocate the complexity when $|\tilde{T}| = 1$ (namely, put $M_\lambda = M$), and when $|\tilde{T}| > 1$, there are $\psi(i)\psi(M-i)$ ways to allocate M among T_{λ_L} and T_{λ_R} for each of the $M-1$ possible allocations i to T_{λ_L}. Table 3.5.3 lists $\psi(M)$ in the range $1 \le M \le 10$. For values of M less than six or seven, $\psi(M)$ is small enough so that brute-force search of the best pruning/allocation combination is feasible (using holdout crossvalidation, for instance). For larger values of M, $\psi(M)$ can be seen to grow faster than exponentially (note that the entries in the third column of Table 3.5.3 exceed unity for $M > 3$), making the brute-force strategy quickly impractical. In particular, this approach would not be appropriate for attempting to optimize M itself, because in practice the optimizing value can turn out to be quite large (typically between 200 and 1,200 in the applications to be discussed in subsequent chapters). However, it may be possible to find some more clever way to combine the pruning and leaf-complexity determination stages that is not susceptible to such explosive growth in computation; investigation of this is left as a topic for future work.

3.6 Variations on PCDT: Softening the Splits and Hybrid Modeling

Two simple but generally very useful modifications to the hard PCDT technique described are now presented. The first involves essentially softening the PCDT partition by mixing the leaf densities together according to the posterior probabilities of leaf membership, as suggested by (2.7.3). This is accomplished simply by taking $\hat{f}(\boldsymbol{x} \mid t)$ to be a separable Gaussian, taking the cell-conditional sample means and variances as the parameters, and by using the normalized leaf populations for the mixing parameters. Thus, the softened version of PCDT corresponds directly to a joint density that is a mixture of separable Gaussians that has been trained in a manner specifically tailored to the conditional density estimation problem. Henceforth, the abbreviation PCDT will be used to refer to this softened version, as will PCDT-S. The hard version will always be identified as PCDT-H.

The second modification is to compute first a linear least-squares regression estimate of Y based on $\boldsymbol{x}$, then to use PCDT to conditionally model the residual, where the conditioning is on the original $\boldsymbol{x}$. This procedure provides a decomposition of the modeling into a linear and nonlinear part. Since the linear part has only $D_x + 1 = D_z$ degrees of freedom, this added complexity is a small price to pay for the insurance it provides against possibly having to spend much more complexity in the PCDT model to capture any global linear dependence (for which PCDT is not well suited). However, there are situations (mostly artificial) in which the removal of the global linear dependence actually requires the complexity of the PCDT model to increase; in fact every distribution in Figure 3.4.1 except (d) is an example where this happens. Fortunately, the complexity is only slightly increased in such cases and drastically reduced in most real applications, so the insurance premium is probably worth paying in practice. Note that this approach doesn't actually require that the regression be linear or that the conditional model be PCDT. However, these were found to work exceptionally well together. The resulting hybrid technique is termed LIN/PCDT.

3.7 Implementation Considerations

Overall, the implementation complexities of PCDT and those of standard EM are comparable. We have found that most of the training time in PCDT is taken up by crossvalidation in fitting the leaf mixtures. PCDT-H and PCDT-S take roughly the same time to train; in fact, the only additional operation required for training the PCDT-S model is computing the leaf-specific means and variances for $\boldsymbol{X}$, which is a negligible incremental burden. Using the parameter values listed in this chapter, the overall training time for PCDT was roughly equal to that of EM for the same dimensionality, model complexity, and length of training sequence.

Note that the PCDT-S model can conveniently be represented and implemented as a mixture of separable Gaussians. Both the computational complexity and storage complexities are thus comparable for PCDT-S and traditional separable mixtures. PCDT-H however can be applied considerably faster, because no influences from adjacent leaves are required in computing the conditional density, so that the conditioning vector can simply be sent down the tree to its leaf and the corresponding simple one-dimensional leaf mixture evaluated. On the other hand, PCDT-H has generally not performed quite as well as PCDT-S, though the difference has always been observed to be small. Note that the storage cost of PCDT-S is actually less than that of an arbitrary mixture of separable Gaussians having the same number of components, because in PCDT-S multiple components in a leaf are constrained to be coincident in $\boldsymbol{X}$ and they share the same diagonal covariance matrix in the $\boldsymbol{X}$-plane.

The training, storage, and application complexities of the LIN/PCDT hybrid model are typically not much more than those of non-hybrid PCDT. The number of linear regression coefficients is equal to the dimensionality of the joint space (counting the offset term), which is small relative to the number of parameters in the tree and leaf mixtures. Also, least squares regression fitting can be carried out quickly using standard linear algebra packages, so that there is little implementation reason not to use the hybrid scheme in general.

3.8 More Examples

We conclude our discussion of tree-structured conditional density estimation with two more examples. The first was introduced in Section 2.7; it involves samples drawn from the distribution specified by row (b) of Table 3.4.3. The empirical relative-entropy performance of PCDT-S is compared for this example with that of EM in Table 3.8.1 when the complexity of the equivalent mixture is limited to $M = 4$, for both the joint and conditional cases. The length of the training sequence was 10,000. As expected, EM does much better for joint density estimation, and PCDT does much better for conditional. Although this example is admittedly contrived, it is interesting to note that in this case PCDT was able to model the conditional density essentially perfectly using four components. Had six components been allowed instead of four, then both would have worked perfectly in the conditional case, although PCDT would have elected to use only four of the allowed six. It is not known whether PCDT always performs at least as well standard EM for conditional-density mixture estimation in terms of conditional likelihood, but in all of the experiments done in this thesis, involving both real and synthetic data, PCDT was always found to perform better than EM in terms of conditional likelihood when the complexities of the mixtures were comparable. However, in the texture synthesis application to be discussed in Chapter 4, EM was generally found to result in more pleasing textures than PCDT; see the discussion at the end of Section 4.4.

Table 3.8.1: Example Performance of PCDT vs EM

Training Method	$\left\langle \ln \frac{f(\boldsymbol{x},y)}{\hat{f}(\boldsymbol{x},y)} \right\rangle$	$\left\langle \ln \frac{f(y\vert\boldsymbol{x})}{\hat{f}(y\vert\boldsymbol{x})} \right\rangle$
EM	0.267 nats	0.265
PCDT-S	0.605	0.001

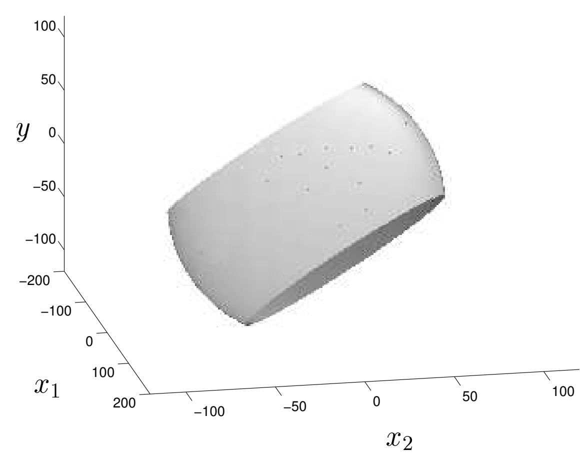

Figure 3.8.2: An example Gaussian density in three dimensions, represented by its one-σ ellipsoid, clipped off at $x_1 = \pm 200$ to show the orientation more clearly. Note that x_1 is irrelevant to the prediction of y, and that the dependence of y on x_2 is linear but noisy. The covariance matrix is given in the text.

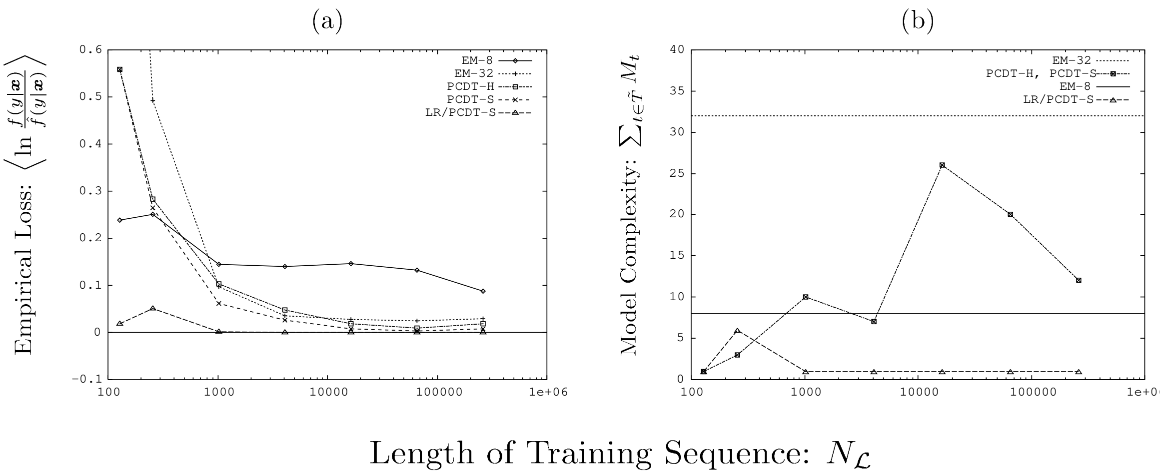

Figure 3.8.3: Empirical conditional relative entropy costs for three versions of PCDT and two EM-trained mixtures of separable Gaussians, as a function of the amount of training data.

The second example involves samples drawn from the inclined three-dimensional Gaussian distribution shown in Figure 3.8.2, whose covariance matrix is $K = ADA^{\mathrm{T}}$, where

$$D = \begin{bmatrix} 400 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 10 \end{bmatrix} \qquad \text{and} \qquad A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sqrt{2}/2 & -\sqrt{2}/2 \\ 0 & \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}.$$

The performance and complexity of each of several conditional density estimation techniques is shown in Figure 3.8.3. Mixtures of separable Gaussians tend to require many components (high complexity) and also large amounts of training data to perform well on distributions such as the one in this example, because a large number of separable mixture components are needed to characterize the correlation of X_2 and y (corresponding to the inclination of the ellipsoid in Figure 3.8.2), and because some of the components are wasted on the irrelevant conditioning variable X_1. This is an example where of course allowing nonseparable Gaussians would result in a substantial improvement in likelihood while actually reducing complexity, but the general use of nonseparable Gaussians in other situations may risk a loss in training robustness, for the reasons mentioned in Section 2.6. Note that LIN/PCDT-S does nearly perfectly, with very low complexity, for all but the smallest training samples. Both PCDT-H and PCDT-S can be seen to outperform both EM-trained mixtures, at comparable or lower complexity and with less training data. Further, note that PCDT-S performs better than PCDT-H; that is, interpolating smoothly between leaf model densities results in a performance advantage. Finally, note that the complexity of PCDT adapts in an interesting way to the size of the training sample. The complexity is small for small learning samples, in seeming observance of Rissanen's dictum: "make as much use as possible of the information in the data, and as little use as possible of information not in the data" (personal communication). As the amount of data increases, the model complexity at first increases, as expected, but then decreases. It is not clear whether the decrease constitutes a trend or an anomaly in this example; more investigation would be required. However, it does seem clear that in general, the complexity growth of PCDT is source-dependent, and that the large-sample asymptotic complexity of a PCDT model will be finite for sources that really are finite mixtures of separable Gaussians. No attempt will be made to prove the latter conjecture however.

3.9 Discussion of New Work Relative to Prior Work

Semiparametric and nonparametric statistical testing traces its roots to work done by Wilcoxon, Mann, and Whitney in the 1940's, as recounted in [103]; however, application of the ideas to multivariate density estimation became conceivable only with the advent of powerful digital computers in the 1960's. The most popular existing techniques have already been discussed in the early sections of this chapter.

The most obvious relevant prior work is the use of CART trees to estimate the posterior class-membership probabilities, which is discussed in the original work by Breiman et al. [13, pp. 121–126]. There, the objective adopted was mean-squared error between the true and estimated PMFs, i.e., a discrete version of the MISE criterion. Restricting the estimated probabilities to be normalized counts, the appropriate splitting criterion for that choice of objective was shown to be the Gini index of diversity, which is similar in form to the Hellinger- and Bhattacharyya distances. In that approach, there is no need to determine the complexity of the leaf probability models; it is always fixed and equals the number of classes. Because of this, and because the discrete MISE criterion doesn't treat incorrect probability estimates of zero specially, there is no need for either regularization or crossvalidation in estimating the leaf models. This is one important difference between class-probability trees and PCDT. But more substantially, PCDT pays attention to and exploits the *meaning* of the dependent variable. In building and using a class-probability tree, the meaning of the class label, if any, is ignored. This makes that technique in a sense more general than PCDT, since it doesn't rely on metric space or smoothness assumptions for the dependent variable. On the other hand, this generality comes at the expense of performance in those applications in which failure to exploit such assumptions severely limits the model's ability to generalize; one such application is conditional density estimation for continuous-valued observations. Recently, class probability trees have been adapted for use in a system that simultaneously performs compression and classification by Perlmutter et al. [90], where the criterion adopted was a weighted sum of Bayes risk and MSE on the observations.

The *Context* algorithm for universal lossless coding proposed by Rissanen in [107] is similar to the CART class-probability tree, but is adaptive (i.e., "on-line"). However, as with class-probability trees, it is also inherently discrete. Adapting it to continuous sources in a way that exploits prior knowledge of smoothness (as is done naturally by mixtures in PCDT) has apparently not yet been done.[16]

Nong Shang, a former student of Breiman, has proposed applying the CART paradigm to the general problem of multivariate density estimation [123]. In that approach, constant probability is assigned to the cells of a CART tree; that is, the density itself plays the role of the dependent variable. Each leaf model is therefore represented by a constant, and these constants are ultimately determined by the relative populations of the cells. Therefore, the approach can be viewed as a multivariate histogram in which the cells are tailored to the training data via CART splitting and pruning. The criterion employed for this was a local "roughness" measure, which in turn was derived from the MISE objective. From this point of view, the technique seems to solve a difficult problem (adaptive design of histogram bins in higher dimensions) in an original and potentially locally-optimal way (at least with respect to MISE). However, there are a number of questions regarding this technique which must be answered before its relevance to conditional density estimation can be properly gauged. The obvious one regards the choice of MISE as the objective, the appropriateness of which has been commented on previously. Another question is technical in nature but potentially important, and relates to the justification of the derived roughness criterion after MISE has already been adopted. Specifically, the derivation of the roughness criterion involves several steps which seem to make strong and restrictive but unstated assumptions about the true underlying probability law. This issue aside, the main point to be made here is that the technique proposed by Shang [123] focuses on the joint rather than conditional density estimation problem, and seems to do so by what amounts to an adaptive-bin histogram.

In 1968 Henrichon and Fu [54] proposed a recursive partitioning technique for mode separation which is similar in several respects to both CART and projection pursuit (see the final paragraph in this section), and therefore deserves mention. Their technique is summarized by

[16] An attempt is made in [141], but the approach makes use of the application-specific assumption that image pixel differences are Laplacian-distributed. While tenable for an i.i.d. model of pixel differences, this assumption is arguably too restrictive in a system that is allowed to adapt. In any case, this approach adapts the problem to the Context algorithm, not the other way around.

Devijver and Kittler [32, chapter 11]. Briefly, it involves sequencing through the eigenvectors of the covariance matrix for the data, proceeding from largest to smallest, partitioning space with eigenvector-perpendicular splits between modes whenever they are detected. To detect the modes, the data points are projected onto the current eigenvector being considered, then a histogram density estimate is formed and its bins are searched for valleys. Once a partition has been obtained in this way, the entire procedure is repeated for each cell in the partition. The process is continued until all cells are unimodal.

Determining model complexity is an extremely important issue in practice. PCDT determines its complexity automatically, but the result is specific to that method of training, and the cross-validation procedures it uses are computationally intensive. An important problem is determining the appropriate complexity of mixture models more efficiently and in a more general setting. Sardo and Kittler have recently investigated efficient methods for determining the appropriate complexities of separable-Gaussian mixture density estimates, particularly for small samples. In a recent paper [118], they invoke Dawid's notion [30] of calibration of the predictive performance of a model specifically to detect underfitting. In addition to avoiding the computational burden of crossvalidation, their approach does not seem to suffer from the propensity towards oversimplistic models that is sometimes associated with MDL and related information criteria. In another recent paper [117], the same authors focus on the situation in which the data are correlated, and develop a maximum penalized likelihood criterion with a free parameter termed a *relaxation factor.* Based on experimental data, they postulate that the appropriate choice of the relaxation factor (hence, model complexity) depends more on the intrinsic dimensionality of the data than on the degree of correlation. These findings may have important implications on PCDT and other techniques when it is desired to bypass the computationally intensive crossvalidation procedure used in fitting the leaf mixtures while still having the model complexity automatically determined according to the data.

As mentioned in Section 2.7, there has been some prior work in direct optimization of mixtures for conditional density estimation using gradient-based techniques. The technique described in [9] implements the mixture as a three-layer neural network, and involves a carefully chosen parameterization of the weights. The resulting gradient expressions appear to be computationally complex, perhaps limiting the applicability of that technique to mixtures having relatively few components and to low-order conditional densities. The technique has the advantage though of attempting to optimize the desired criterion directly without recourse to PCDT's artifice of explicitly partitioning conditioning space. Its application to large mixtures should therefore be investigated, and the results compared with those obtained with PCDT.

Recently, Tony Jebara has proposed a fixed-point maximization technique [63, 64] which seems to show promise for conditional density estimation, again primarily in the case of low-complexity mixtures. The development mirrors that of ordinary EM [31], but the conditional density is substituted for the joint density when lower-bounding the per-iteration likelihood improvement via Jensen's inequality. The resulting lower bound requires what essentially amounts to a mini-EM loop for its maximization with respect to the mixture parameters, making the computational complexity of the technique greater than that of ordinary EM (hence the hypothesized restriction of its suitability to small mixtures). The method has been applied in the context of computer vision to learn a mapping between facial feature locations and image data, yielding results superior to those obtained using EM-estimated joint densities [63]. However, convergence even to a local optimum is not reliable; typically several initializations must be tried before convergence is obtained. It is not known whether convergence can always be obtained by repeatedly retrying with a different initialization. One possible explanation for the unreliable convergence is that an inequality conjectured to hold in demonstrating convergence might occasionally fail, removing the guarantee of convergence in some cases. This issue requires further investigation. Nevertheless, when the technique does converge, it usually results in a better conditional density estimate than would be obtained by straight EM. An appropriate use of Jebara's method might be to refine an initial guess obtained by the seemingly more robust PCDT method, in the same way that EM can be used to refine a joint density obtained via the GLA.

Stephen Luttrell has proposed two different semiparametric methods for multivariate density estimation that should be mentioned, though they relate more to the general topic of high-dimensional density estimation than to PCDT specifically. The first, dubbed *partitioned mixture distributions*, pertains potentially to any situation wherein a large array of separate mixture models are required to be represented simultaneously. Specifically, in [75], a scheme is presented whereby the individual mixtures are made to share structure (interpreting the mixtures as three-layer neural networks), thereby saving representation cost and presumably improving training reliability (at some expense of modeling flexibility). The second technique, called the *adaptive cluster expansion*, involves combining several conditional density estimates for a given dependent variable that are based on different conditioning variables. A maximum-entropy approach is used, and it is suggested that the technique be carried out hierarchically. This would be another means for obtaining a high-order conditional density estimate that would be interesting to compare performance-wise to PCDT. A philosophically similar technique, without the hierarchical component, was proposed independently in [98]. Recently the same idea has been developed and analyzed extensively in the discrete case by Christensen et al. [24]; its extension to the continuous case, where its performance with PCDT could be compared, is a topic for possible future research.

An important nonparametric technique in multivariate density estimation that has not yet been mentioned is *projection pursuit* [57, 39]. In that technique, a multivariate (i.e., joint) density is expressed as a background density (which must be assumed) multiplied by the product of M univariate functions of linear combinations of the coordinates. The estimation technique is an iterative procedure to determine suitable univariate functions and linear combinations of the coordinates to maximize likelihood. The linear combinations of the coordinates are usually constrained to be projections; hence the name "projection pursuit." Just as the multivariate mixture model was adapted to conditional density estimation by a particular training method (PCDT), projection pursuit might likewise be adapted so that it can be used as an alternative model in conjoint processing. This idea was not investigated in this thesis, but would be a worthwhile area of future research.

3.10 Chapter Summary

In response to the observation made at the end of Chapter 2 that good joint density estimates cannot always be used to get good conditional ones, a technique specifically tailored to the general problem of conditional density estimation was developed. The two main advantages of the technique are its suitability specifically for high-order conditional density estimation, which is required in conjoint processing, and its ability to automatically determine its appropriate level of complexity.

The technique combines aspects of decision trees, histograms, and mixtures. Specifically, it attempts to make best use of the conditioning information by first finding a suitable tree-induced partition of the conditioning space, and then fitting one-dimensional mixture densities to each leaf. In order to do this, the greedy tree-growing and minimum cost-complexity pruning of the CART methodology were adapted to the task by developing an appropriate splitting and pruning criterion that employs a histogram-based approximation to the model likelihood. An additional problem that had to be solved, not addressed by the traditional CART paradigm, was that of fitting the leaf-conditional mixture densities. The solution presented here integrates the task of training the leaf mixtures with that of automatically determining the best level of complexity based on the data.

Two modifications, which may be used in tandem, were described, one in which the partition is softened, and the other in which the technique is used in conjunction with linear regression. The functioning and performance of PCDT was illustrated through several examples involving simulated data, and it was found to outperform mixture models trained via EM in those examples.

CHAPTER 4:

Conjoint Image and Texture Processing

In this chapter, the versatility afforded by having high-order conditional density estimates is demonstrated by applying some of the models developed in Chapters 2 and Chapters 3 to image compression, image restoration, and texture synthesis. The methods to be considered operate directly on the pixels, without preprocessing. In Chapter 5, preprocessing will be considered. Some of the ideas to be explored here are independent of the precise choice of conditional density model, provided that it is able to capture high-order, nonlinear statistical interaction. Except in the case of texture synthesis, results for both EM-trained mixtures and PCDT models will be presented.

A comment is in order regarding the performance figures to be presented. Images tend to consist of separate regions differing in their characteristics. Therefore, in order to perform well, an image processing system has to employ an adaptive model. A fixed conditional density estimate, high-order or not, is adaptive only inasmuch as the supplied conditioning information allows it to be. If appropriate adaptation information is not available in $\boldsymbol{x}$, then it will not be exploited. Therefore, the performance figures reported in this chapter may not reflect the full potential of conjoint processing when used with more fully adaptive models.

4.1 Lossless Compression of Greyscale Images

There are many situations in which lossless (i.e., noiseless; reversible) compression is necessary or desirable. For natural-scene images digitized to usual spatial and amplitude resolutions, the lossless compression ratios typically achieved are quite modest: a two-to-one ratio generally would be regarded as quite good. Better ratios might reasonably be expected when the source has been digitized at a high spatial resolution, or when the images are especially "clean" as in computer-generated graphics. Our main motivation for considering lossless compression is that it serves as a particularly pure test of probabilistic modeling, as was discussed in Section 2.3.

When a model is explicitly available, arithmetic coding [112] provides a means of using that model directly to perform lossless compression. The performance achieved is nearly perfect with respect to the model; if the model happens to match the source, then the actual bit rate will be very close to the entropy rate. Thus, the availability of arithmetic coding transforms the lossless compression problem into a probabilistic modeling problem.

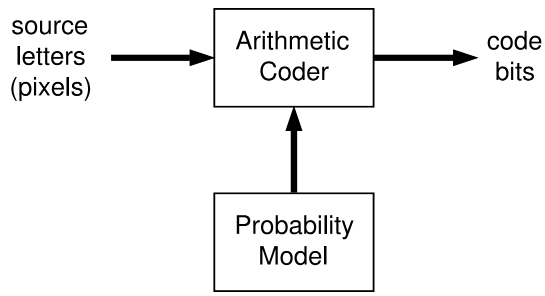

Figure 4.1.1: An arithmetic coder is a lossless compressor that allows the probability model to be explicitly separated from the encoder and the PMF to be specified independently for each pixel encoded. The PMF can be conditioned on anything that the decoder will have access to prior to the time of decoding.

One of the first published applications of arithmetic coding was a scheme for lossless compression of binary images [69], which is depicted in Figure 4.1.1. Pixels (which can be either black or white) are scanned in raster order and fed to an arithmetic coder, along with a corresponding estimate of the conditional PMF. The arithmetic coder then produces a compact sequence of bits from which the original image may be recovered exactly. The critical step is estimating the conditional PMF. For each pixel to be encoded, a neighborhood like one of those shown in Figure 4.1.2 is centered on the pixel. The pixel to be encoded corresponds to y, while those that are to serve as conditioning values correspond to the coordinates of $\boldsymbol{x}$. For some pixels to be encoded near the top and left boundaries of the image, one or more of the specified conditioning pixels would lie outside the image. In such cases, those pixels are assumed arbitrarily to be white. The neighborhoods are required to be such that the conditioning pixels precede y in raster order. This ensures that $\boldsymbol{x}$ is available to the decoder prior to its recovery of y, so that both the encoder and decoder can use the same sequence of PMFs, which is a necessary condition for correct decoding. Because the pixels are binary valued in the images considered by Langdon and Rissanen, the size of the alphabet of $\boldsymbol{x}$ for any of the neighborhoods shown in Figure 4.1.2 is small enough that it is feasible to estimate the conditional PMF for each $\boldsymbol{x}$ by counting $\boldsymbol{x}$-specific occurrences of the values of y. Several variations of such a count-based estimation procedure are described in [69].

Count-based PMF estimation is not appropriate for greyscale images, because the number of conditioning states becomes prohibitively large for even moderate-size neighborhoods. Also, that method of estimation when applied to scalar observations does not exploit the relationship between nearby amplitude levels. As an alternative, any of the techniques for high-order conditional density estimation presented in Chapter 2 and 3 provides a means of directly extending the Langdon-Rissanen framework to work on greyscale images. Specifically, the observed pixels may be regarded

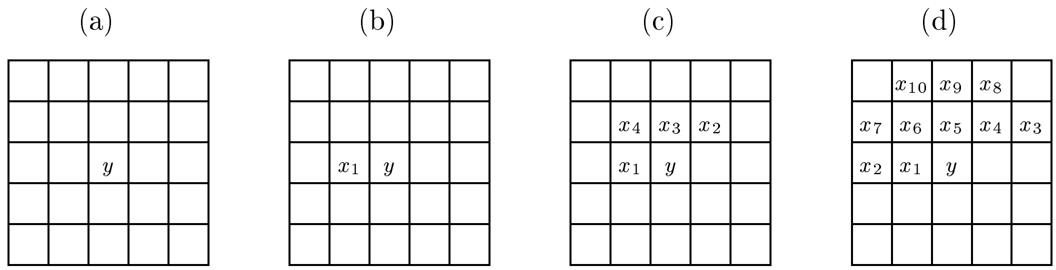

Figure 4.1.2: Pixel-extraction neighborhoods that can be used when processing an image sequentially in raster order. The x_i's serve as the coordinates of a conditioning vector $\boldsymbol{x}$, while the center pixel y is taken to be the dependent variable.

Figure 4.1.3: The 256×256, 8 bpp monochrome *cman* test image (left) and the 512×512 8 bpp monochrome *Lenna* test image (right).

as uniformly quantized versions of hypothetical continuous-valued pixels, each quantized value corresponding to the center of its quantization region. Assume that the pixel values are represented by integers as usual, so that the quantizer stepsize is unity. Then the estimated conditional density can be used to obtain an estimate of the conditional probability mass as follows:

$$\hat{P}(y) = \int_{y-1/2}^{y+1/2} \hat{f}_{Y|\boldsymbol{X}}(y' \mid \boldsymbol{x})\,dy'.$$

Once the PMF has been estimated, it can be supplied to an arithmetic coder that has been designed for a pixel alphabet of the appropriate size. As mentioned, it is a property of arithmetic coding that the resulting compression will be good if the model is accurate in terms of relative entropy.

This approach was used to losslessly compress the standard 256×256, 8-bit *cman* image using each of the neighborhoods shown in Figure 4.1.2. Training was carried out on 131,072 $(\boldsymbol{x}, y)$ pairs extracted at pseudorandom locations from a quilt of 26 natural images that did not include the *cman* test image. Simulated compression results in average number of bits per pixel (bpp) are shown in Table 4.1.4 for each of three methods of conditional density estimation described in Section 2.6 and Chapter 3. Conditioning values corresponding to locations outside the image were taken arbitrarily to be 128. For comparison, we note that the entropy of a discrete i.i.d. source corresponding to the histogram of *cman* is about 6.9 bpp. The marginal rates achieved here are greater than this, which is expected because the training was carried out on a set of images having a different combined histogram.[17] The estimated bit rate for lossless compression of the 512×512 monochrome (Y component) version of *Lenna* using this scheme was 4.26 bpp, which is only about 0.05 bpp worse than that recently reported by Weinberger et al. [141] for a modified version of Rissanen's *Context* algorithm [107].

Table 4.1.4: Lossless Compression Rates (bpp) for the *cman* Image

Neighborhood	EM-128	PCDT-S	LIN/PCDT
(a)	7.96	7.94	7.94
(b)	5.54	5.50	5.51
(c)	5.24	5.22	5.21
(d)	5.17	5.09	5.11

4.2 Conjoint Scalar Quantization

We next consider how the modeling techniques described in Chapter 3 might be applied to lossy compression. The most general and powerful lossy compression method, at least in principle, is to quantize the pixels jointly rather than separately; i.e., to use vector quantization (VQ) on the whole image. A more feasible alternative would be to divide the image into non-overlapping equal-size blocks and apply VQ to each one. Unlike the scheme to be presented, VQ can always approach the rate-distortion bound in the limit of large vector dimension [6], which is the optimum achievable curve (distortion-measure dependent) in the rate vs. distortion plane. However, VQ can be complex to design and, at the encoding end, to implement.

The ability to model high-order conditional densities effectively allows a system to quantize and encode pixels sequentially while still retaining some (though not all) of the advantages of processing the pixels jointly. Specifically, the spatial interdepence among pixels can be exploited, as can the shape of their joint distribution [71]. Space is not filled with representative points as efficiently as it would be were the pixels processed jointly, but the penalty for this can be small in many applications.

[17] Note that we could have encoded the image-specific histogram as a header to the code bits, allowing us to encode at a rate close to 6.9 bpp. Such a technique and variants of it are termed *universal*. Recently, several strategies for universal coding of images were proposed and investigated in [36]; these techniques appear to hold considerable promise.

The proposed method of lossy encoding is uniform scalar quantization with conditional entropy coding, wherein both the quantizer and the entropy coder are adapted on the basis of previously decoded pixels. The system is shown in Figure 4.2.1. It is a form of predictive coding, but the function of the prediction is *not* the traditional one of minimizing the residual's variance and correlation. To understand this, note that in the proposed system the unconditional variance of the residual does not directly affect either the average distortion or the average rate, and that correlation in the residual can be exploited when the quantizer outputs are entropy coded using a conditional model. Instead, the real purpose of prediction in this system is just to shift the mode of the conditional density (when it has just one mode) to lie in the center of the innermost quantization cell, making the conditional entropy of the quantizer output small (arbitrarily close to zero for sufficiently large stepsizes). This has two related and desirable effects, both at low rates. First, it reduces average distortion by making it more likely that the nearest representation level will be close by. Second, it minimizes the chance that a narrow mode will be split by a quantization threshold, which would significantly increase the conditional entropy and hence the rate. At high rates, the prediction $\hat{y}[n]$ could be set to zero and the system would still perform well, which demonstrates that the role played by the prediction is not the usual one. When the conditional density has two or more modes of comparable probability, shifting the distribution in this way does not necessarily reduce the conditional entropy of the output of the quantizer, but there is no reason to believe that it increases it either. The predictor really comes into play at low rates.

In many (but not all) versions of predictive coding, the prediction is a linear combination of previously decoded values [45, 62]. Such prediction is termed *linear.* From this point of view, the prediction made by the proposed system is potentially highly nonlinear, since it can be an arbitrary functional of the estimated conditional distribution. In the experiments presented here and in the following chapter, the conditional mean was used as the predictor.

Note that the distortion incurred by this scheme comes entirely from the lossy encoding of $r[n]$. This system has the advantage of strictly bounding instantaneous error to the range $[-\Delta/2, \Delta/2]$, provided that there are enough levels to avoid overload.

To train the model, samples that have been quantized by the system are needed to make up the $\boldsymbol{x}$ values. Unfortunately, the quantization depends on the model through the predictor, and hence quantized values are not available at the time of training. A way around this problem is to simulate the effect of quantization without actually performing it. When the quantizer has

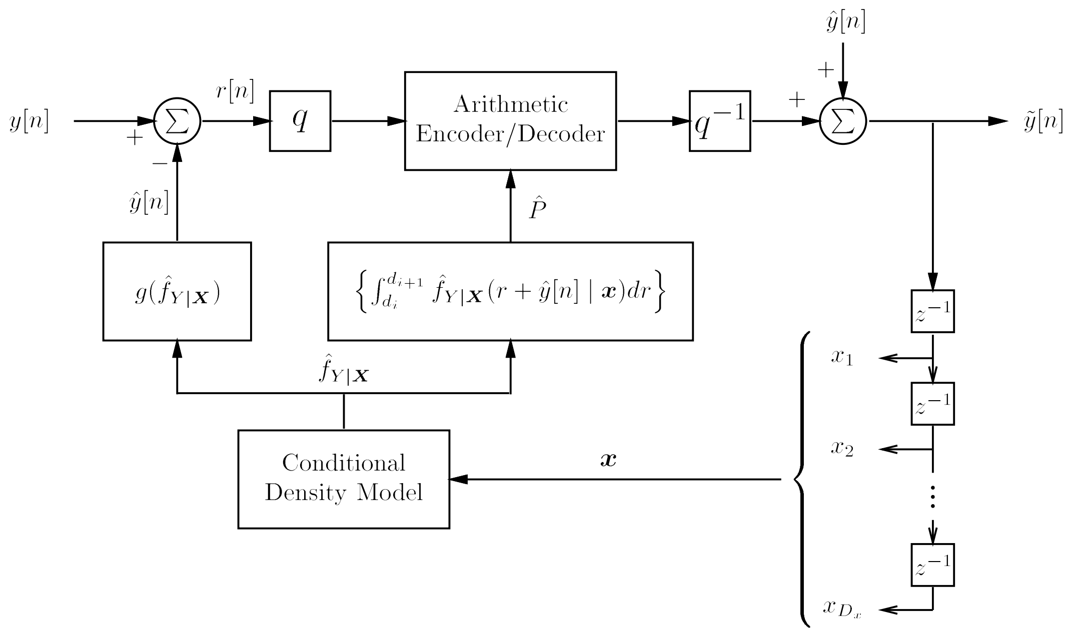

Figure 4.2.1: Nonlinear predictive coding system that employs a high-order conditional probability model. The predictor is allowed to be any functional of the predictive (i.e., conditional) density for $y[n]$; in the experiments presented here, the conditional mean was used. Scalar quantization with entropy coding is employed, which is reasonably efficient when the residual $r[n]$ is memoryless or, as in this case, the memory has been modeled so that the entropy coding is conditional. The probability model $\hat{P}$ for the arithmetic coder makes use of the numerical meaning of the quantizer output, which is an important feature not usually achieved by models used with arithmetic coding (e.g., that employed by the Q-coder [87]). The true function of the predictor is a bit subtle here. See the main text for an explanation.

uniformly spaced representation levels, as is henceforth assumed, the simulation of quantization can be accomplished by adding i.i.d. noise uniformly distributed on $[-\Delta/2, \Delta/2]$ to $y[n]$. It is important to note that the input to the quantizer is not continuous because the predictor is a function of quantized data, implying that this model of the effect of quantization is not strictly justified, even at high rates. Nevertheless, the instantaneous error is confined to the interval $[-\Delta/2, \Delta/2]$ even if it is not uniform on that interval; moreover, it has been observed in all applications tested that the quantization error variance is close to the value of $\Delta^2/12$ predicted by the uniform-error model.

To see how the system performs, we first test it on a very useful signal model in image, video, and speech processing: the autoregressive (AR) source. Perhaps the simplest source that exhibits memory (statistical dependence among successive observations), the AR source is both analytically tractable and easy to simulate, thus facilitating both conceptual and experimental testing of a new system. For this source, we know both the theoretical performance limit and the theoretically optimum conditioning order.

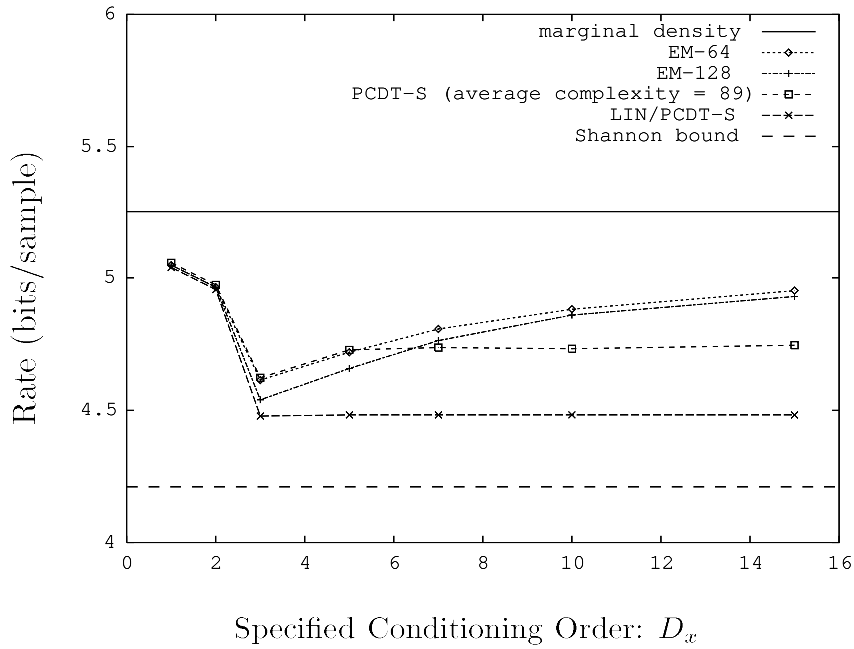

Figure 4.2.2: The predictive coding scheme described in Section 4.2 was used to encode a sample sequence of a third-order Gaussian autoregressive process having coefficients $a_1 = 0.9$; $a_2 = -0.8$; and $a_3 = 0.7$. The quantizer stepsize was set at 0.185, resulting in a nearly constant signal-to-noise ratio of 30.05 ± 0.05 dB. Rate versus specified conditioning order (memory) are shown for: no prediction, EM-trained mixture models at two levels of complexity, the PCDT model, and the linear/PCDT hybrid model. To obtain the curves, holdout crossvalidation was employed on a 131,072-long original synthesized sequence, with a 2/3-1/3 training-test split. For reference, also shown is the rate-distortion bound at 30.05 dB. Note that the performance of both the PCDT and linear/PCDT hybrid models is approximately maintained as the specified order is increased beyond the optimum value of 3, while that of each of the EM-trained mixtures diminishes significantly. Note also that the hybrid is only about $4.48 - 4.21 = 0.27$ bits/sample worse than the rate-distortion bound. Assuming an ideal predictor, the residual for this process is i.i.d. Gaussian. Therefore, using ideal entropy-coded scalar quantization with ideal prediction would result in the gap width being $\frac{1}{2}\log_2(\pi e/6) \approx 0.255$ bits at high rates, which is not much better than that achieved here by LIN/PCDT.

A discrete-time Gaussian autoregressive process $\{Y[n], n = 0, 1, \ldots\}$ of order m is described by the recursion

$$Y[n] = \sum_{i=1}^{m} a_i Y[n-i] + V[n] \tag{4.2.3}$$

where $V[n]$ is a sequence of i.i.d. Gaussian random variables, such that future V's are always independent of current and previous Y's [6]. For simplicity, it is assumed that the V's have zero mean and unit variance. Figure 4.2.2 presents an example of applying the proposed system to a Gaussian AR source described by the coefficient values $a_1 = 0.9$; $a_2 = -0.8$; and $a_3 = 0.7$, using several different conditional density estimation techniques. The caption provides the details. Note that LIN/PCDT performs best, which in view of the linear autoregressive nature of the source is not surprising. Simple linear prediction with a Gaussian residual would have done as well in this case, but it is significant that the much more flexible LIN/PCDT scheme knew enough to constrain itself, so that no penalty had to be paid for the added flexibility.

It is useful to compare performance with Shannon's distortion-rate function. For a Gaussian autoregressive source of order m and assuming mean-squared distortion, this limiting R-$\mathcal{D}$ curve is given parametrically by the following expressions [47, 6]:

$$\mathcal{D}_\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi} \min\Big[\theta,\ \frac{1}{g(\omega)}\Big]\, d\omega,$$

and

$$R(\mathcal{D}_\theta) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \max\Big[0,\ \frac{1}{2}\log_2 \frac{1}{\theta g(\omega)}\Big]\, d\omega,$$

where

$$g(\omega) = \Big|1 - \sum_{i=1}^{m} a_i e^{-ji\omega}\Big|^2.$$

Varying θ from zero (corresponding to zero distortion) to infinity (zero rate) traces out the rate-distortion bound. The integrands in these expressions were found to be sufficiently well-behaved to allow good trapezoidal-rule approximation using no more than a few hundred terms. The calculated bound is indicated in Figure 4.2.2 as the dashed horizontal line at the bottom of the graph. It is at about the expected distance of 0.25 bits below the LIN/PCDT curve, serving as a check that the system is performing properly.

We now apply the system to the *cman* image. Training is carried out on the same data as for the lossless case, except now the conditioning values are taken from the output image instead of from the original. Since they are not available for training (they depend on the model through $\hat{y}[n]$), the effect of quantization is simulated by adding the appropriate level of uniform noise to each coordinate of $\boldsymbol{x}$. The resulting rates are shown in Table 4.2.4 for three quantization stepsizes and for the various methods of conditional density estimation. It is interesting to note that the performance of EM flattens out and even worsens as the neighborhood size increases, in part because it expends model complexity on coordinates of $\boldsymbol{x}$ that are conditionally irrelevant to y given certain values for the other conditioning coordinates. In contrast, the performance of PCDT and its hybrid does not degrade with more conditioning, perhaps because it allocates model complexity on the basis of relevance to predicting y.

Note that if the scalar quantizer q is chosen such that its set of representation levels contains the union of the source alphabet and the set of all pairwise differences of elements of the source alphabet, then the system depicted in Figure 4.2.1 can be made into a lossless compression system

Table 4.2.4: Lossy Compression Rates (bpp) for the *cman* Image

STEPSIZE = 10 (PSNR ≈ 39dB)			
Nbd.	EM-128	PCDT-S	LIN/PCDT
(a)	4.65	4.70	4.70
(b)	2.50	2.48	2.40
(c)	1.99	1.90	1.94
(d)	2.00	1.81	1.86

STEPSIZE = 20 (PSNR ≈ 33dB)			
Nbd.	EM-128	PCDT-S	LIN/PCDT
(a)	3.63	3.64	3.64
(b)	1.78	1.71	1.77
(c)	1.07	1.02	1.03
(d)	1.16	1.03	1.02

STEPSIZE = 40 (PSNR ≈ 27dB)			
Nbd.	EM-128	PCDT-S	LIN/PCDT
(a)	2.61	2.65	2.65
(b)	1.26	1.03	1.05
(c)	0.61	0.62	0.60
(d)	0.66	0.50	0.51

by selecting the predictor $g(\hat{f}_{Y|\boldsymbol{x}})$ to be the composition of any real-valued functional of $\hat{f}_{Y|\boldsymbol{x}}$ with q. The system is lossless because the residual is among the representation levels, so that nearest neighbor quantization preserves its value. The system described in Section 4.1 is then actually a special case, resulting from the choice $g(\hat{f}_{Y|\boldsymbol{x}}) \equiv 0$ (which is legal because of the stated condition on q).

It is interesting to compare the proposed conjoint scalar quantization system with traditional VQ. In the proposed system, model complexity is determined by the discovered complexity of the source, to the extent allowed by the amount of available training data. In contrast, the complexity of unconstrained fixed-rate VQ is determined by the target rate and the desired dimensionality. For example, to achieve a rate of 2.0 bits per coordinate for 10-dimensional vectors, a VQ codebook of size 2^{20} would be required. For this reason, operating at medium or high rates is difficult to achieve with VQ when the dimensionality is high. In the proposed system, arbitrarily high rates can be achieved simply by reducing the quantization stepsize sufficiently.

Another point of difference is the bounding of instantaneous error, which the proposed system accomplishes while achieving respectable MSE-versus-rate performance. Thus, the proposed system may perform well with respect to L_∞ in addition to L_2. Traditional VQ targets only MSE-versus-rate performance, without bounding the instantaneous error. Lattice vector quantization can bound instantaneous error, provided that the lattice is not truncated in a way that allows quantizer-overload distortion. In fact, the proposed system can be interpreted as a sliding-block entropy-coded Z-lattice VQ in which the lattice is shifted to center the input distribution on the center of the innermost quantization cell.

Finally, note that the proposed system provides a unified approach to prediction and conditional entropy coding. Traditionally, entropy coding in lossy compression systems is based on probabilities estimated from occurrence counts of quantizer output values. Doing so does not take full advantage of prior knowledge that the quantizer output values are related to the input and to

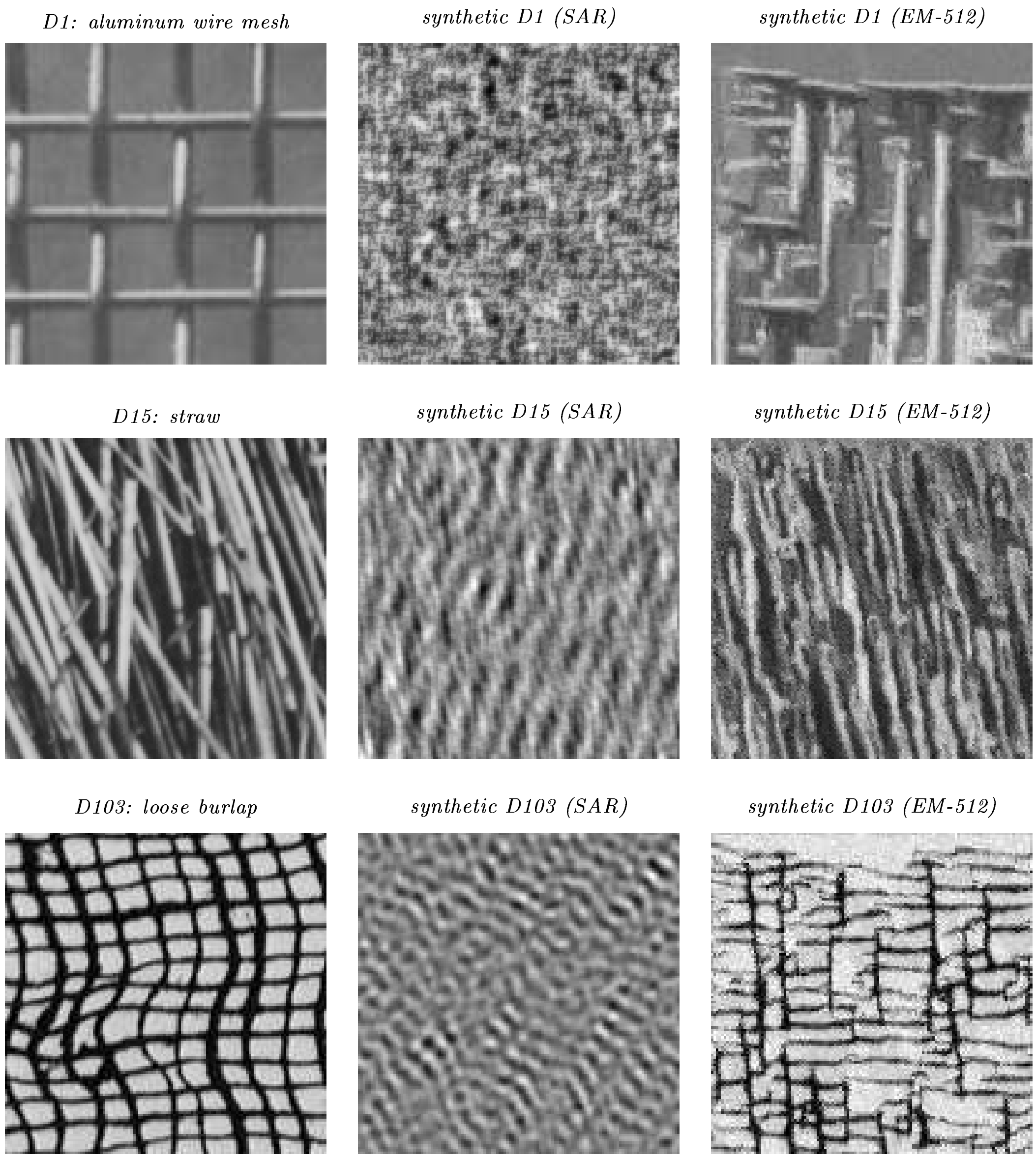

Figure 4.2.5: For each of three textures selected from the Brodatz collection [14], a 256×256 patch (left column) was extracted from the original 512×512 8-bit image. A simultaneous autoregressive (SAR) model [67] was trained on each of these using a 21×21 estimation window. The resulting SAR model was then used to synthesize the texture shown in the middle column. The order of the SAR model was 12, corresponding to a 5×5 neighborhood. The right column shows the result of synthesis by sequentially sampling from a 10^{th}-order conditional distribution estimated by a 512-component, EM-trained mixture of separable Gaussians. The neighborhood shown in Figure 4.1.2(d) was used to extract the training data from the corresponding original full-size image.

each other, in that they lie in a metric space. In contrast, the conditional density model used by the entropy coder in the proposed system effectively makes use of such prior knowledge by focusing on the quantizer input rather than output.

4.3 Sequential Texture Synthesis

Imagine taking the decoder portion of the lossless compression system of Section 4.1, and feeding it random coin flips instead of a code-bit sequence for an actual image, until an image comes out. If this were repeated many times, it would be noticed that it almost always takes about the same number of coin flips to get an image out. This is one of the properties of *typical sets* — they all have about the same probability or ideal code length. The "decoded" image will very likely be a member of this typical set. Thus, this is a way to peek into the mind (or more precisely, model) of a compression system, to see what its "view of the world" is. If the compression system is efficient for the source it is designed for, then decoding random bits will result in images that resemble the original in all of the ways that the model thinks are important. By training the compression model on a natural texture, random bits will be decoded into an image that resembles a texture.

Decoding random coin flips is not the only way to generate typical outputs, although it is efficient in usage of coin flips. We could just as easily write a random number generator routine that samples from the appropriate conditional distributions. However, the decoding interpretation allows us to draw an interesting parallel with a well known method of texture synthesis: passing white noise through a linear shift invariant system. Here, we are also starting with a kind of noise, albeit binary, and passing it through a system to generate a texture. However, the system is no longer linear. Results for this technique are shown in Figure 4.2.5 when the conditional density estimate is a GLA-trained mixture model. The neighborhood shown in Figure 4.1.2(d) was used, so the conditioning order of the model was 10. For comparison, also shown is the result of a 12^{th} order simultaneous autoregressive model[18] (SAR) synthesis [67] which corresponds to a filtering of white noise. The proposed nonlinear technique based on high-order conditional density estimation is able to capture much more of the characteristics of the textures that are visually important than is the SAR model. However, like the SAR model, it misses some of the large-scale structure of the textures. We also note that the computational cost of training and synthesis in the proposed conjoint approach is greater than for the SAR model; estimation and synthesis of each of the SAR textures shown took about 15 minutes combined on an otherwise unloaded HP 725 workstation, while the conjoint training and synthesis took about 110 minutes combined on the same unloaded machine.

[18] Fang Liu kindly provided the code for generating these.

4.4 Multiresolution Texture Synthesis

Since the dimensionality of the conditioning vectors grows as the square of the diameter of the pixel neighborhoods, the size of neighborhood is effectively constrained in the above approach on practical grounds. However, to better capture interdependence that might exist on a larger spatial scale, it would be advantageous to include in the neighborhood pixels at large displacements from y. One way to approximate the effect of this is to synthesize the pixels in a coarse-to-fine order, resulting in a progressive synthesis. The idea is to first synthesize a subset of the pixels spaced widely apart in the image, then fill in more pixels in between them, continuing in this way until all of the pixels have been assigned values. For this to work, we cannot get by with just a single conditioning neighborhood, since the pixels used for conditioning in this process are no longer always in the same locations relative to the pixel being synthesized. The fact that multiple neighborhoods are needed implies that multiple conditional models are also needed. The flip side is that this provides a means of combining multiple models together in a divide-and-conquer fashion.

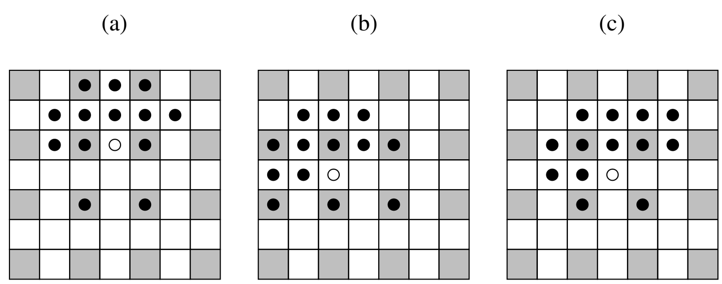

Figure 4.4.1: Conditioning neighborhoods used in multiresolution texture synthesis for each of three classes of synthesized pixel defined by their position relative to the subsampling lattice.

There is another way to look at this process when the synthesis order described above is sufficiently regular. In such cases, the synthesis can be thought to occur at different resolution levels in a recursive subsampling of the finest resolution image. The synthesis proceeds in a coarse-to-fine manner. Initially, a small, coarse texture is synthesized using the method described in Section 4.3. This texture is then upsampled by a factor of two in each dimension, with blank pixels filling in the gaps. Next, these blank pixels are filled in by sampling from an appropriate conditional distribution. The conditioning can be on any of the pixels that have already been synthesized. This includes all of the pixels carried over from the coarser level, plus any that have already been synthesized at the current level. Since some of the pixels carried over follow the current pixel in raster order, we have achieved a relaxation in the strict spatial causality requirement. In the case of 2×2 subsampling, there are three neighborhoods that must be defined (and three corresponding

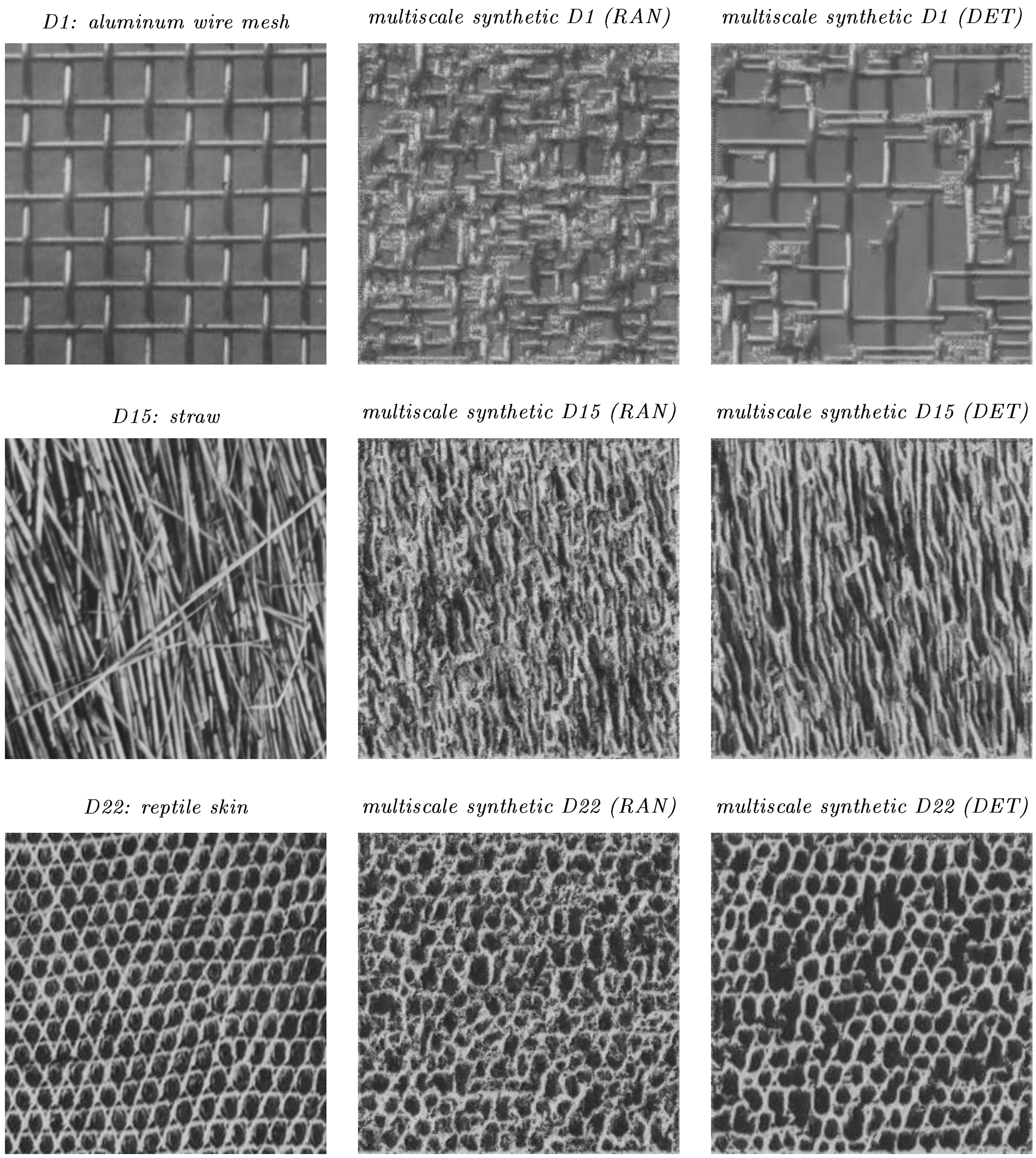

Figure 4.4.2: Two different multiresolution approaches to texture synthesis based on the high-order conditional density modeling using mixture models trained with the EM algorithm as described in Chapter 2. Originals are shown on the left. The textures in the middle column were synthesized by generating each pixel as a random number that obeys $\hat{f}_{Y|X}$, while pixels in the textures in the right column were sequentially set to $\arg\max_y \hat{f}_{Y|X}$. See the main text for a description of the models, neighborhoods, and training process.

models trained), since there are three pixels that must be filled in for every one carried over, and each is in a different position relative to the subsampling lattice. Examples of such multiresolution texture synthesis are shown in the middle column of Figure 4.4.2. The three neighborhoods shown in Figure 4.4.1 were used, so that the conditioning order was 13. EM-trained 512-component Gaussian mixtures were used to estimate the conditional densities. In order to obtain enough

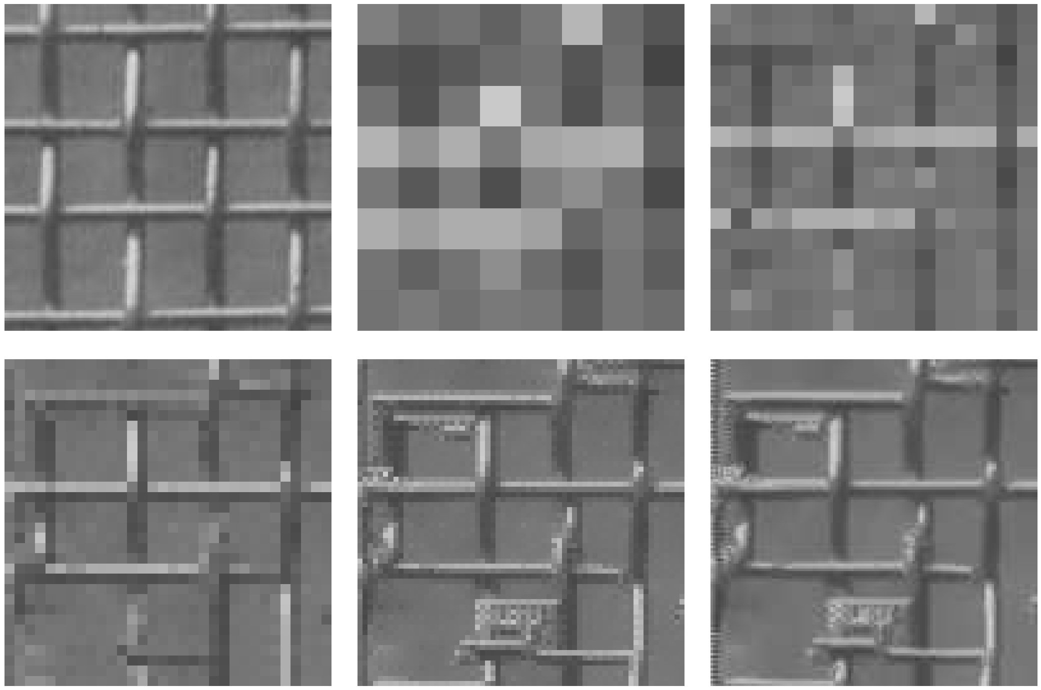

Figure 4.4.3: Evolution of resolution in deterministic version of texture synthesis. In raster order, the images are: a patch from the original *D1* aluminum wire texture, followed by the results of deterministic multiresolution synthesis at successively finer resolution levels. The details of training and synthesis are the same as for the rightmost column in Figure 4.4.2. Note the ability of the method to capture complex phenomena such as occlusion and bending in the wire — this is normally not seen in statistical methods.

training data at the coarsest resolutions, all of the possible subsampling phases were used. The right column shows the result of the same texture synthesis procedure, but where the arg max of $\hat{f}_{y|\boldsymbol{x}}$ is used instead of a pseudorandom number obeying that distribution. Note the high-level structure that is evidently captured by this technique as the resolution increases, as shown in Figure 4.4.3 for *D1*. Note also that there are small regions in which the synthesis appears to be corrupted, for instance at the bottom center of the highest-resolutions (bottom right) image. It is believed that this phenomenon is triggered at a coarser level when the pixels generated at that level in a local area are atypical with respect to the joint model with the next highest resolution level. Because of their low likelihood, the conditional models are not optimized for such conditioning pixels, so that the new pixels generated in that local area at the next level are likewise atypical. The effect is that this is perpetuated through all the subsequent levels of synthesis, resulting in the anomalies in the figure.

We first presented the above methods of texture synthesis in 1993 using a discretized mixture of Gaussians trained by the GLA [96]. The results shown here use a continuous model trained by EM, allowing less complex models for comparable visual quality in the multiresolution case, and resulting in noticeably better visual quality in the single-resolution case. The multiresolution texture synthesis technique will be generalized in the next chapter to use filtering with the subsampling, instead of subsampling alone.

PCDT has been tried in the texture synthesis application, but thus far the results obtained using PCDT have not been as visually similar to the originals as those obtained using EM. An initial effort was made to try to characterize the differences in the visual characteristics of the synthesized textures resulting from the two approaches, but this preliminary investigation was inconclusive; more research is required.

4.5 Image Restoration

As mentioned in Section 2.3, if all that is required is an estimate of the dependent variable for a given condition value, then estimating its entire distribution might be wasteful. But does estimating the density as an intermediate step necessarily incur a performance cost as well? Consider the PCDT-H technique. Had the squared-error criterion been used in splitting and pruning, a regression tree would have resulted essentially for free. If the leaf-conditional distributions happen to be unimodal Gaussians, as is often assumed in regression, then PCDT-H would find that out (unless the data were anomalous), and absolutely no harm would be done. Thus, the tree is no worse than it would have been without the added step of leaf-mixture fitting. The risk really is that PCDT may mistakenly model the residual as more complex than it really is. There are safeguards built in against this (extensive crossvalidation), but slight overfitting is possible. On the other hand, if the leaf-specific density is much different from Gaussian, then PCDT will react by assigning a more complex mixture to that leaf. The added complexity would be wasted, since in the end, the global y-mean of $\mathcal{L}_t$ is selected by both methods (assuming least-squares regression). But PCDT has not done any harm from a performance point of view.

A good example problem to illustrate the effective use of complexity-controlled conditional density estimation in the function approximation setting is image restoration, which we now consider. Given a patch in a degraded image, what is the best estimate of the original, undegraded pixel that was at the center of that patch? High-order conditional density estimation can be applied to this problem by trying to learn the relationship between a patch of degraded pixels and the corresponding original pixel, using for example one of the neighborhood pairs shown in Figure 4.5.1. Note that the dependent variable is taken from the original image for training and its

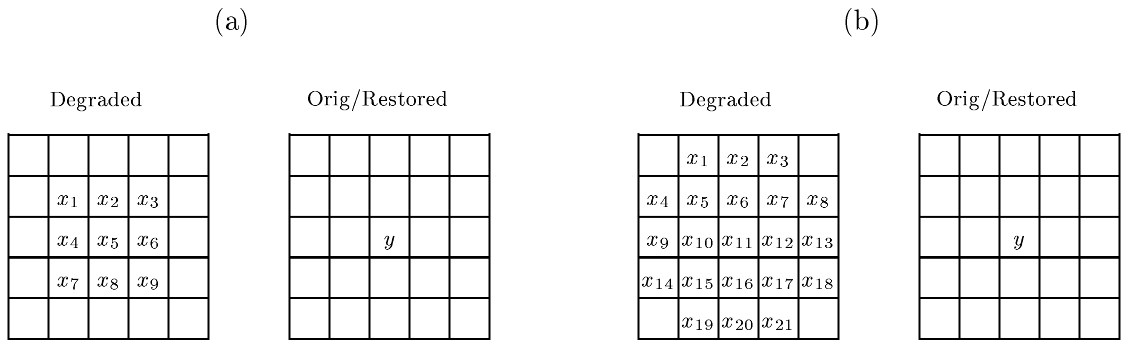

Figure 4.5.1: Pixel-extraction neighborhood pairs used for image restoration based on the use of a high-order conditional model. Note that spatial causality is not required, as the entire degraded image is assumed available at once.

Figure 4.5.2: Example of image restoration using high-order conditional modeling. See the main text for an overall description of the experimental set-up. In raster order, the images are: original, degraded, optimum linear restoration (Wiener filtering) using a (nonseparable) 7×7 FIR filter, restoration using an EM-trained 256-component mixture model and the neighborhood shown in Figure 4.5.1(a), restoration using LIN/PCDT with the same neighborhood, and finally restoration using LIN/PCDT with the neighborhood shown in Figure 4.5.1(b). In raster order, the PSNR values are infinity, 28.09, 29.32, 29.25, 30.98, and 30.69. Note that the bottom three results are conjoint — that is, they are based on sequential processing using high-order conditional density estimation.

estimated value is placed in the output image during restoration, at the location corresponding to the center of the degraded-pixel neighborhood.

We now consider a specific example. Twenty-six (original, degraded) pairs of natural images were prepared such that the degradation consisted of additive zero-mean white Gaussian noise having a variance of 150 (the pixels themselves range from 0 to 255). The resulting set of vectors was pseudorandomly subsampled to reduce the number of training vectors to the more manageable number of 131,072, and conditional density models were trained on the resulting set. The *cman* image, which was not included among the training images, was likewise degraded, and used as the test image. Figure 4.5.2 shows the results of restoration using several models. The top right attempt corresponds to standard linear regression using a square 7×7 neighborhood for the degraded conditioning pixels, which is equivalent to linear shift invariant filtering. In fact, it is equivalent to Wiener filtering, though viewed in the spatial domain. The bottom three examples result from conjoint processing. Note that they all appear to be sharper than the result of Wiener filtering. Note however that Wiener filtering does slightly better than the EM-trained mixture in terms of PSNR, though the EM-trained mixture result using the neighborhood pair Figure 4.5.1(a) appears to be less noisy, though somewhat blotchy. The formula used for PSNR was

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\text{error variance}}.$$

The LIN/PCDT hybrid also using neighborhood pair (a) does about 1.6 dB better than either of these in terms of PSNR; it is sharper than the linear technique and slightly less blotchy than the EM result. On the other hand, the EM result appears to have slightly less noise. Using a larger degraded-pixel neighborhood with LIN/PCDT (bottom right) actually results in a slight reduction in PSNR, but the subjective quality is slightly better on the diagonal edges; for instance the camera man's jaw-line. None of the methods is adaptive, so the results may not be great in absolute terms, but the comparison is fair. The tentative conclusion is that there is a performance advantage to the sort of nonlinear processing that high-order conditional density estimation affords. Also, it appears that the added flexibility provided by density estimation over function approximation need not result in a performance penalty. On the other hand, we have informally estimated the computational cost of the conjoint methods to be about two orders of magnitude greater than that of Wiener filtering for training, and about one order of magnitude greater for application. The latter cost might be greatly reduced if schemes such as lookup tables are used in implementing the conjoint methods.

4.6 Relationship to other methods

The applications we have considered are intended only as examples to illustrate the potential applicability of sequential processing using conditional density estimation to real problems; we were not trying to solve those problems. Therefore, we consider only the prior work in these areas that seem to relate to the approaches considered here.

The lossless compression scheme of Weinberger et al. [141] is based on explicit probability estimation and therefore deserves mention. Their technique is inherently adaptive owing to its use of Rissanen's *Context* algorithm [107]. That algorithm is count-based and hence requires categorical data. In an attempt to make use of prior knowledge that the underlying probability law is smooth for natural greyscale images, pixel differences are formed and used for the conditioning contexts. These differences are modeled as Laplacian, which may be appropriate in nonadaptive systems, but such an assumption needs to be justified in an adaptive system. In contrast, both the EM- and PCDT-based conjoint methods of lossless compression proposed here make direct use of prior knowledge of smoothness when estimating the densities.

There is a vast amount of literature on lossy compression. Connections can be drawn between the proposed conjoint scalar quantization system and several variations on VQ, including predictive, multistage, and nonlinear interpolative VQ [45]. Perhaps the most relevant prior work is the conditional entropy-constrained VQ technique proposed by Chou and Lookabaugh [21]. In that technique, entropy coding of VQ codewords is carried out conditionally using a first-order Markov model estimated from codeword histograms (occurrence counts). The authors report a significant improvement using this technique over unconditional entropy coded VQ in compressing linear predictive coefficients for speech. The main points of difference between their approach and the one presented here are (1) we use high-order density estimates that directly exploit smoothness (which occurrence counts don't do), and (2) they use VQ in their approach, which is more general but more complex to implement than the scalar quantization proposed here.

Markov random mesh models based on parametric conditional densities have been studied extensively for use in texture and image modeling. In 1965 Abend et al. [1] applied such models to binary images. Besag [8] studied consistency requirements to allow the use of conditional rather than joint probability models to specify a two-dimensional stochastic process, and considered several exponential-family parametric conditional models. In 1980, Kanal [66] summarized and reinterpreted the earlier approach of Abend et al., and traced the early development of Markov random field (MRF) models for image processing. The explicit use of MRFs for textures, again using parametric conditional models, was proposed in 1980 by Hassner and Sklansky [50]. Much of the research in MRFs has focused on the problem of estimating model parameters; see for example the 1987 paper by Pickard [91]. The texture synthesis methods described here fit into the Markov random mesh framework in the sense that textures are generated according to locally conditioned probability models. However, the conditional models are not required to be of the simple parametric form assumed throughout the MRF literature.

A texture synthesis method using pyramid structures has recently been proposed by Heeger and Bergen [53, 52]. In their approach, a texture is synthesized by iteratively transforming the marginal distributions of subbands in a steerable-pyramid decomposition [133] of a pseudorandom noise image to match those of a given texture sample. The marginal distributions are estimated by histograms. Although the histogram matching is done on the individual subsampled pixels, the synthesis filtering causes that matching to affect spatial dependence in the reconstructed image, resulting in a synthetic texture that strongly resembles the original. A subtle additional difference between their method and our multiresolution one derives from the iterative nature of their procedure. It would not be quite enough to match the histogram in the subbands prior to synthesis; an additional requirement imposed in their method is that re-analysis into subbands maintain that marginal distribution. As suggested to me by Olshausen [83], this may have the effect of forcing the subband phases to line up to match the original phase after synthesis as well. Zhu et al. [145] have proposed a technique similar to that of Bergen and Heeger, invoking the principle of maximum entropy to relate the joint and marginal distributions. In contrast to both of these methods, the synthesis techniques proposed here operate directly on the original pixels and capture spatial interdependence explicitly via high-order conditional density estimates. As will be discussed in the next chapter, it is not clear at this time whether combining high-order density estimation with extensive filtering can result in a strong improvement over either one alone.

4.7 Chapter Summary

Mainly to show the power and flexibility afforded by the conditional density estimation techniques developed in the previous chapter, this chapter considered the application of such techniques to selected image and texture processing problems by working directly in the original pixel domain. The use of high-order conditional densities of the type considered here represents a break from tradition, as the resulting systems are neither linear nor based on simple parametric assumptions.

In particular, it was shown in this chapter that high-order conditional density estimation can be used for predictive sequential compression in an effective and straightforward way. In both cases, the derived probability mass functions used by the entropy coder made use of both conditioning (memory) and prior knowledge of smoothness of the underlying distribution, the latter of which is not typically used by entropy coding systems in image compression. Texture synthesis was considered next. One method involved generating pixels as pseudorandom numbers with the appropriate conditional distribution in raster order. While performing much better than a standard linear technique of comparable conditioning order, it was evident that more spatial context would have to be employed to capture the macroscopic properties of the textures. To this end, a modification of the ordering was considered, and interpreted in terms of coarse-to-fine processing. Also, a variation in which the pixels were assigned values in a deterministic rather than random manner was presented. The technique was found to capture complex structure not usually associated with statistical approaches. Finally, we considered the problem of image restoration. The conjoint approach, which is nonlinear, was found to yield a sharper image than optimum linear filtering in the example considered, but at greater computation cost for both training and application. Also in that example, an objective improvement as measured by PSNR was achieved by using the linear/PCDT hybrid instead of either EM-trained mixtures or Wiener filtering. More research is required to characterize the performance relationships and computational efficiencies achievable by the various approaches more generally.

CHAPTER 5:

Preprocessing

All of the conjoint processing described in the previous chapter was carried out directly on the pixels themselves; no preprocessing was done. We now consider conjoint processing when preprocessing is allowed. Ultimately, we are interested in gaining insight into the question of what filters should be used in a conjoint subband-pyramid coding system. Since there are many parameters in such a system, including the ordering through the subband-pyramid and the choice of conditioning structures used by the models, working in that application is not chosen as the best way to build insight. Instead, we consider first a much simpler application system which can be interpreted (albeit loosely) as a one-band subband coder. Surprisingly, it can achieve objective coding gain in its intended domain of applicability.

5.1 One-band Subband Coding

The one-band subband coding system shown in Figure 5.1.1 was first proposed as a means for robust quantization in [99]. There, it was demonstrated that (1) minimum-MSE fixed-rate scalar quantization (here referred to as *Lloyd-Max quantization*) can be made robust, i.e., insensitive to modeling errors, by appropriate choice of the pre- and post-filter; and that (2) for several important sources, the MSE-versus-rate performance of such a system is actually better than that of direct Lloyd-Max quantization of the original source. Here we are concerned with (2).

The way that the system achieves coding gain for certain sources is now described. Two i.i.d. model sources commonly used in image and speech coding are the Laplacian (two-sided exponential) and gamma densities. Fixed-rate scalar quantization is known to perform relatively poorly on these sources with respect to MSE when compared with either VQ or variable-rate scalar quantization. In fact, the MSE-performance of Lloyd-Max quantization is worse for these sources than for the Gaussian source, actually resulting in a reversal of the performance ordering indicated by the rate-distortion bounds for these three sources. By temporarily transforming the original signal into one that has an approximately Gaussian amplitude distribution, the system of Figure 5.1.1 achieves an overall SNR comparable to that of Lloyd-Max quantization of a Gaussian source at the same rate, which is an improvement for the two sources mentioned. The amplitude distribution is made

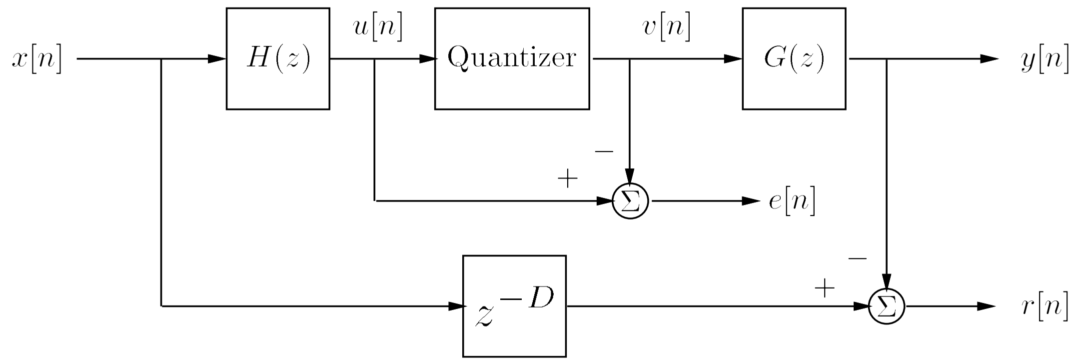

Figure 5.1.1: Robust quantization system originally proposed in [99]. The dispersive all-pass filter $H(z)$ temporarily transforms the amplitude distribution of an arbitrary i.i.d. input to be approximately Gaussian via Liapunov's central limit theorem, allowing a Lloyd-Max quantizer for a Gaussian source to be employed. As an added bonus for certain sources, the overall MSE-performance after inverse-filtering is actually better than that of direct, correct-model Lloyd-Max quantization, at the cost of some delay due to filtering.

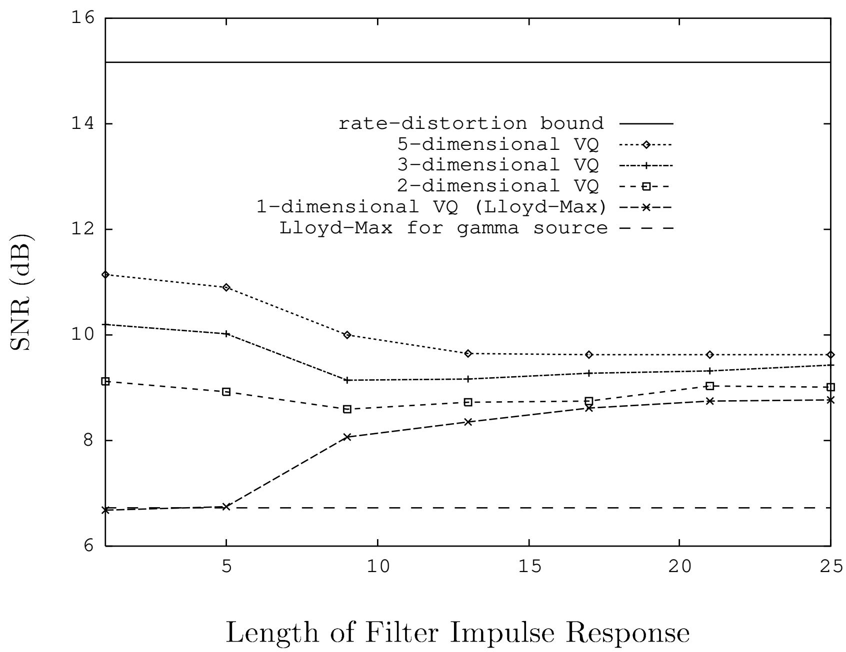

Figure 5.1.2: Empirical MSE-performance as a function of impulse-response length for $H(z)$ of the robust quantization system of Figure 5.1.1, applied to an i.i.d. gamma source at a rate of 2.0 bits/sample, using vector quantization. The filtered signal $u[n]$ is blocked into vectors of varying dimension, then full-search vector-quantized at fixed rate using a GLA-designed codebook. A filter length of unity corresponds to no preprocessing, while a dimension of unity corresponds to Lloyd-Max quantization. For reference, the rate-distortion bound (computed via the Blahut algorithm [10]) is also shown. Note that preprocessing helps performance when the simplest VQ is used, but is actually harmful when higher dimensionality is permitted. This qualitative result has been observed at all rates tried (of which 2 bpp was the maximum, because of the exponential growth of the VQ codebook as a function of rate).

approximately Gaussian because the output of $H(z)$ is the sum of a large number of independent random variables whose variances are not too different.

It is ironic that there should be an improvement, because as mentioned the rate-distortion bound is worse for a Gaussian than for a Laplacian or a gamma source. Had we been willing to use joint or conjoint quantization instead of fixed-rate scalar quantization, we would have been better off not filtering, as shown for 2 bit/sample fixed-rate VQ in Figure 5.1.2. Note that increasing VQ dimensionality improves performance for all filter lengths. However, both the improvement and absolute performance of high-dimensional VQ are greater when filtering is not used, i.e., at the extreme left of the graph. In this example, fixed-rate VQ was used instead of conjoint SQ, because Lloyd-Max quantization is conveniently a special case of fixed-rate VQ but not of conjoint SQ, which is entropy-coded. The formula used for SNR was

$$\text{SNR} = 10 \log_{10} \frac{\text{variance of } x}{\text{variance of } r}.$$

5.2 Filter Banks for Conjoint Subband Coding

Traditionally, subband pixels have been treated as if they were independent. In recent years, there has been growing interest in taking advantage of the nonlinear statistical dependence among subband pixels [17, 25, 55, 89, 88, 122, 126]. Recall from Section 2.1 that a simple but effective traditional approach to subband image coding is to employ entropy-coded uniform-threshold scalar quantization on the subbands, maintaining the same stepsize when quantizing all of the pixels in all of the subbands. In this section, we assume that the subband decomposition is a pyramid structure of the type shown in Figure 2.1.2. Subband regions of high activity or variance will cause the quantizer to have greater output entropy, resulting in greater rate being implicitly allocated to those regions and subbands. As mentioned in Section 2.1, it can be shown that such a simple strategy results in a nearly MSE-optimal rate allocation [42, 93], without the need for a computationally intensive rate-allocation procedure. Of course this assumes that the probabilistic model upon which the entropy coding is performed is accurate.

Traditionally, non-DC subband pixels have been modeled as independent random variables that are identically distributed with a distribution (typically Gaussian or Laplacian) that is known except for an activity factor or variance that changes relatively slowly over space and spatial frequency. The DC subband, which is in many ways just a smaller version of the original image, is often quantized and encoded at at fixed rate of 8 bits per pixel. Because in a pyramid the DC subband accounts for only a small fraction of the total number of samples, treating it in this way does not appreciably increase the total bit rate. If conjoint processing is employed, then more complex statistical relationships among the non-DC subband pixels can be exploited by training and applying high-order conditional density models. Specifically, we define a total ordering on the pixels in the pyramid. Corresponding to any specific such ordering is a collection of conditioning neighborhoods that satisfy causality constraints with respect to that ordering. Using the conjoint scalar quantization system of Section 4.2, we can compress an image by fixing a quantizer stepsize and quantizing subband-pyramid pixels sequentially in the adopted order, entropy-coding each quantized value according to a probability law that is conditioned on previously encoded pixels.

In designing such a system, one has considerable freedom in selecting the ordering, the neighborhoods, and of course the training and structural parameters of the conditional models. Of present interest is the change (if any) of the role played by the filter bank in determining the performance of such a system. Note that decorrelation and energy-compaction are no longer the manifest goals of the filter bank. Indeed, correlation or even statistical dependence among the subband pixels can be exploited by the conditional entropy coding. Moreover, in such a system, high rate is no longer implicitly allocated simply to high-activity regions, but rather to either moderate- or high-activity regions *that are relatively unpredictable by the model* (recall the material on rate allocation from Section 2.1). Conversely, a low rate can be allocated even to a high-activity region, provided that the conditional model is able to predict adequately the subband pixel values in that region. Thus, both the predictive accuracy of the model and the choice of conditioning neighborhoods can be expected to exert substantial influence over the performance of such a system.

As was explained in the last paragraph of Section 1.2, the question of what filter characteristics are best to use in a given subband coding system must be answered on practical rather than theoretical grounds. One way to begin to understand how the requirements on the filter bank might change when joint or conjoint quantization is allowed is to perform an experiment wherein filters achieving different degrees of tradeoff between spatial and spectral localization, but which are structurally identical, are each tried in an otherwise fixed conjoint subband coding system. The characteristics of the low-frequency analysis band of a set of such two-band critically sampled filter banks are shown in Figures 5.2.1 and 5.2.2, and their impulse responses tabulated in Appendix B, where the procedure used to design the set is also given. That procedure attempts to minimize a weighted sum of spatial and spectral localization, where the relative importance of spatial localization is specified as a parameter α. As is typical for two-band critically sampled filter banks, the analysis filter for the high-frequency subband is obtained simply by reversing the sign of the odd-indexed samples of the impulse response of the low-frequency filter. As a result, the frequency response of the high band mirrors that of the low band, and the spatial energy spread

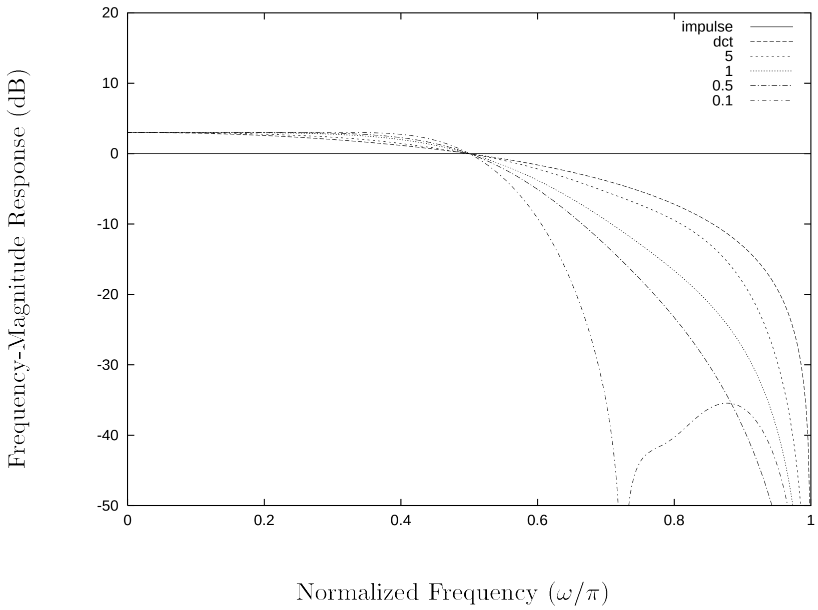

Figure 5.2.1: Frequency-Magnitude responses of the low-band filter in a set of two-band critically sampled filter banks that can be used in an experiment to gauge the appropriate tradeoff point between "no filtering" and "a lot of filtering" preferred by a subband coding system. See Appendix B for details. For reference, the curve for the DCT (which in this two-band case is also the Haar) is also shown. The impulse filter corresponds to $\alpha = \infty$.

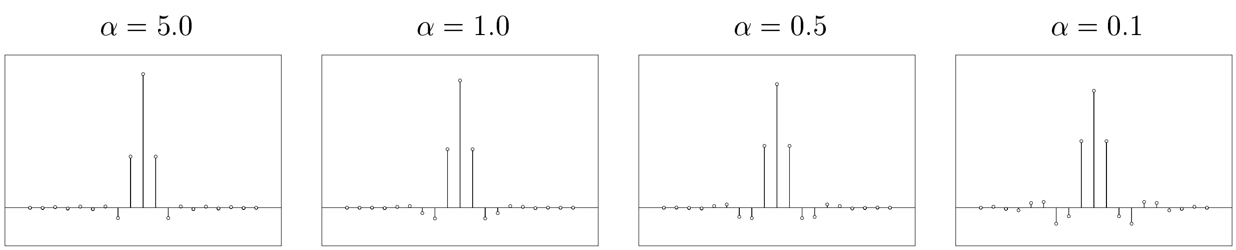

Figure 5.2.2: Impulse responses for the low-band filters whose frequency responses are shown in Figure 5.2.1. Note the increasing spatial dispersiveness, as confirmed by Table 5.2.3. Here, the definition of dispersiveness is the second central spatial moment; a formula is given in the text.

Table 5.2.3: Spatial Dispersiveness of Filters Shown in Figures 5.2.1 and 5.2.2

α	Dispersiveness
∞	0
5.0	0.271678
1.0	0.361359
0.5	0.452197
0.1	0.726879

or *dispersiveness*, as measured by the second central spatial moment

$$\text{Dispersiveness} = \sum_{n=-L}^{L} n^2 h^2(n),$$

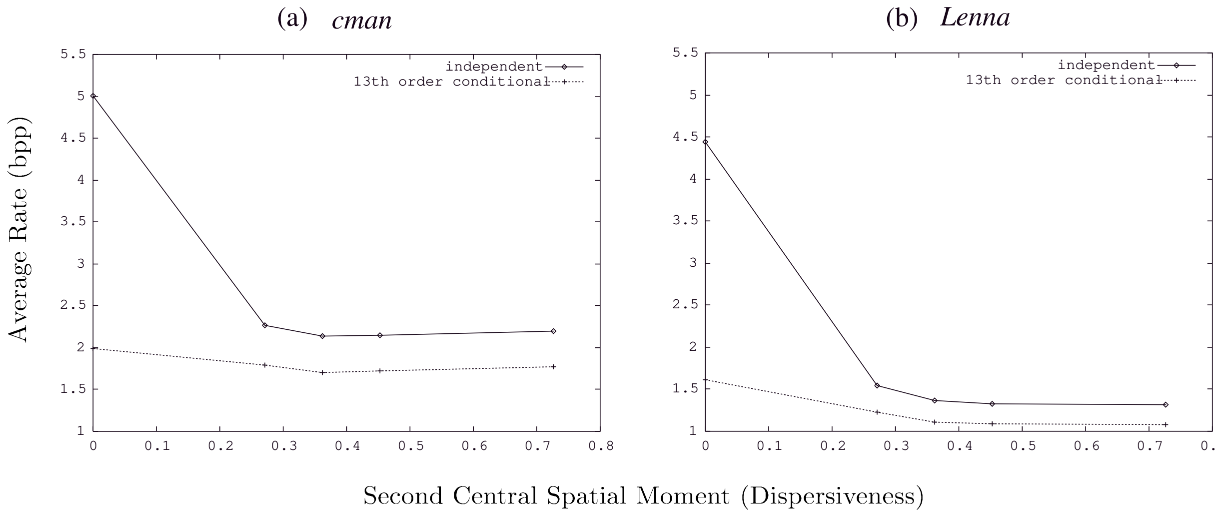

Figure 5.2.4: Average rate versus filter dispersiveness for conjoint and independent subband pyramid coding of the *cman* and *Lenna* test images, using the system described in Section 4.2 with a quantizer stepsize of 10.0, for a near-constant PSNR of about 39 dB. See the text for details and an interpretation.

is the same for the low- and high-frequency analysis filters. The synthesis filters in these filter banks are the same as the analysis filters. The initialization for the optimization procedure used in generating these filters was the 9-tap filter bank proposed by Simoncelli and Adelson [131], and like that filter bank, the subsampling phases differ for the low- and high-frequency bands.

Figure 5.2.4 shows the average rate in bits/pixel versus filter dispersiveness for both independent and conjoint quantization of subband pixels in a four-level pyramid using the filters of Figures 5.2.1 and 5.2.2. Quantization and entropy coding were carried out using the system described in Section 4.2, with independent quantization achieved by taking the conditioning dimension in the model to be zero. The conditional models themselves were EM-trained mixtures of 256 separable Gaussian components. Different models were used at different levels; that is, no assumption of statistical self-similarity across scale was made. However, such an assumption would have been reasonable and might have simplified training. An ordering of pixels in the pyramid was defined as follows. First, all of the DC subband pixels were encoded. Next, for each subsampled spatial location in the coarsest level of the pyramid, the three pixels corresponding to each of the three oriented subbands were encoded in the following order: horizontal, vertical, and diagonal. After looping over all such spatial locations in raster order, the four subbands in the coarsest level of the pyramid were combined to yield the DC band of the next higher resolution level. This then served as conditioning information for encoding the three oriented bands at that level. The process was continued in this manner until all pixels in the pyramid were encoded. In the conjoint

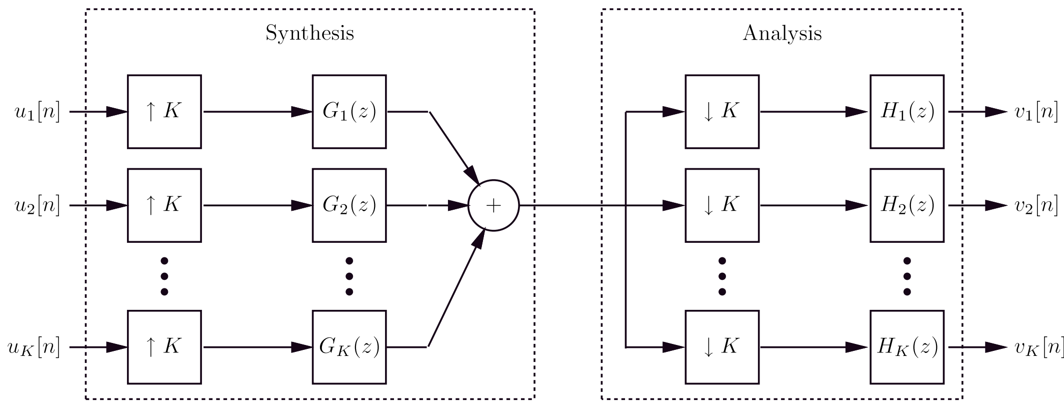

Figure 5.2.5: By treating the subbands as input and performing the synthesis first, a critically sampled perfect-reconstruction filter bank can be used as what amounts to a frequency-domain multiplexer. If the lowest-frequency input band is given but the others are not, then the synthesized image will be lacking in detail. By generating the higher-frequency input subbands in a way that is consistent with a statistical model that chacterizes interdependence both within and across subbands, realistic detail can be "invented" in the synthesized image. This general idea was first suggested by Pentland and explored by Holtzman [55], who used implicit models determined by VQ codebooks to increase the resolution of natural images. Here, we apply the technique to texture synthesis, using explicit conditional density models.

case, the conditioning neighborhoods for each of the four subbands at a given level were taken to consist of the nearest thirteen pixel locations that preceded the current location in the specified ordering, where Euclidean distance in the four-dimensional space of spatial and subbands index offsets was used to determine nearness. In all cases, conditioning values that fell outside of the subband image boundaries were integrated out of the joint density prior to computing the conditional density. The quantizer stepsize was fixed at 10.0, so that the overall MSE was anticipated to be approximately $100/12 \approx 8.33$, corresponding to an anticipated PSNR of 38.9 dB. The actual PSNR values observed ranged between about 38.5 and about 40.2 dB for the twenty points in the graphs. The models were trained on pyramids formed from a set of 32 512×512 8 bpp natural monochrome images that did not include the two test images. While it is not possible to draw general conclusions from such a limited experiment, the curves do seem to suggest (1) that the filter characteristics play a greater role when traditional, independent processing is performed than when conjoint processing is performed, and (2) that conjoint processing in this case performs better by about 0.25 bit/pixel than independent processing, though at greater computational cost. Contrary to our expectation, the curves do not seem to indicate that less spatial dispersiveness is called for in conjoint subband coding, although any conclusion would have to await additional experiments in which the orderings, neighborhoods, conditioning dimensionalities, model complexities, and quantizer stepsizes were varied.

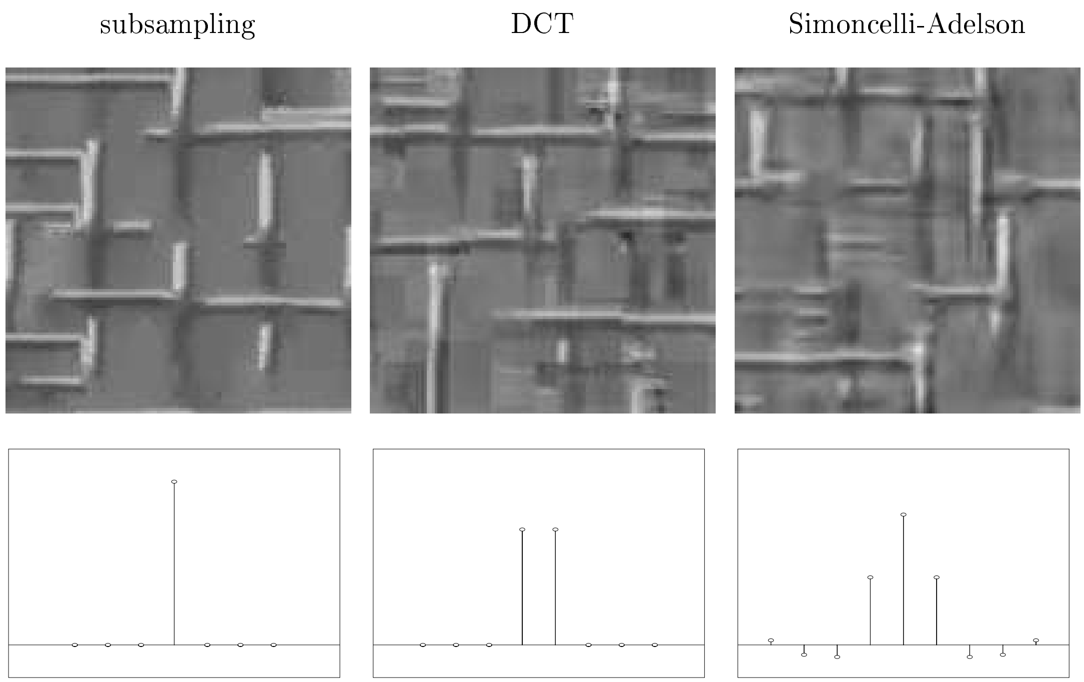

Figure 5.2.6: Effect of filter spatial-spectral localization tradeoff in subband-pyramid texture synthesis of the Brodatz texture *D1*. See the main text for details. The leftmost texture has been synthesized without filtering, in the multiresolution fashion described in the previous chapter. The middle and right textures were synthesized using the two-point DCT and the 9-tap Simoncelli-Adelson filters [131], respectively. The spatial dispersion factors for these filters are: zero (subsampling), 0.25 (DCT), and 0.48 (Simoncelli-Adelson).

5.3 Conjoint Subband Texture Synthesis

We now consider the potential role of preprocessing in multiresolution texture synthesis. It is possible to reinterpret the texture synthesis method presented in Section 4.4 in terms of filling in pixels in a subband pyramid wherein out-of-phase subsampling and identity (i.e., impulse) filters are employed. Pixels are synthesized sequentially in the same order that pixels are encoded in the conjoint subband coding system described in Section 5.2.

At each higher-resolution level of the pyramid, the synthesis can be thought of in terms of adding in detail to the image from the previous level. To accomplish this, the coarser image is taken to be the low band input to a synthesis filter bank viewed as a frequency-domain multiplexer, as shown for one dimension in Figure 5.2.5. Viewing it in this way emphasizes that the bands can be specified independently: no constraint is imposed by the filter bank structure as to what values may be used for the subbands. In two dimensions, there are three non-DC subbands. Pixels in these higher-frequency bands are synthesized sequentially to be statistically consistent with the low band and with the previously synthesized pixels in the higher bands as determined by the set of conditional models. As was described in Section 4.4, the pixels can be synthesized as pseudorandom numbers or by using a fixed functional of the conditional distribution such as the conditional mean.

Coarse-to-fine synthesis proceeds as follows. First, the DC subband is synthesized using the technique described in Section 4.3. Next, pixels in the three remaining subbands at that resolution level are synthesized, in the order described in Section 5.2. Note that this ordering allows the conditioning neighborhoods defined in Figure 4.4.1 to be used. Once all four subbands have been synthesized at that level, they are fed into the synthesis bank, resulting in the low-band image at next higher resolution level. The process is repeated until a texture of full resolution is obtained. Note that although the same neighborhoods can be used at different levels, each level must have its own set of models, for corresponding to each neighborhood.

An example of conjoint subband texture synthesis is shown in Figure 5.2.6. A five-level subband pyramid was used. In synthesizing the low band at the coarsest level, a conditioning neighborhood that consisted of the nearest 13 pixels that preceded the current one in raster order was used. The three neighborhoods shown in Figure 4.4.1 were used in synthesizing the other subbands, with Figure 4.4.1(a) used for horizontal, (b) for vertical, and (c) for diagonal. The conditional models were EM-trained mixtures of 256 separable Gaussians. Although a strong conclusion cannot be drawn from this single example, the results do seem to suggest that a greater degree of filtering (as measured by spatial dispersion factor, for example) results in the synthesized texture itself appearing to have been filtered, i.e., exhibiting what appear to be blurring and ringing artifacts. The results obtained by Heeger and Bergen [52, 53] suggest that using different filters — specifically, oriented ones — may result in an improvement over the filtered synthesis results shown here while using relatively simple probabilistic models for the subbands. However, as indicated in Section 4.6, their approach cannot be simulated simply by generating subband pixels as pseudorandom numbers that obey the appropriate marginal distributions, because the phase of each subband is adjusted implicitly by their iterative synthesis procedure.

Conjoint subband texture synthesis is not limited to visual textures. In joint work with Nicolas Saint-Arnaud [115], a one-dimensional version of the technique described above was used to synthesize sound textures. In that work, it was found that the technique resulted in some realistic renditions of steady-state textured sounds such as the applause of a crowd, but had difficulty in capturing the onset and decay of dynamic sounds such as those made by an automatically fed photocopier. As discussed in Chapter 4, incorporating adaptivity into the models is expected to improve the ability to handle such dynamic behavior.

5.4 Chapter Summary

We have considered three examples in which the role of preprocessing appears to change when conjoint processing is employed. In two of these examples, one-band subband coding and subband texture synthesis, conjoint processing seems to work best when the filtering is minimal. The remaining example, conjoint subband coding of images, was inconclusive in this regard. These results lead us to conjecture that in systems that employ linear filters for preprocessing, the best spatial-spectral tradeoff may shift in favor of spatial localization when conjoint or joint processing is used instead of independent processing.

In the subband image coding example, it was further observed that the filters played a lesser role in determining system performance when conjoint processing was used than when independent processing was used. Finally, in both the one-band subband coding and conjoint subband image coding examples, it was found that conjoint processing offered a definite objective performance advantage over independent processing.

CHAPTER 6:

Recommendations for Further Work

This chapter suggests topics for further research involving PCDT, conjoint processing in general, and conjoint subband coding in particular.

6.1 PCDT

The PCDT methodology could be improved and generalized in many ways. The most obvious would be relaxing the separability constraints on the leaf mixtures. Another interesting modification would be the combining of pruning and leaf-complexity allocation, the possibility of which was discussed at the end of Chapter 3.

Another important improvement would be handling the situation in which one or more conditioning coordinates are either unreliable or missing. The CART methodology uses surrogate splits for this. Whether this technique carries over to PCDT, and whether it is the best way to accomplish the objective, needs to be investigated. In the case of PCDT-S, an equivalent joint distribution is available in analytic form. Thus the EM framework, which is exactly suited to filling in missing data, might be more appropriate. In fact, in the form of PCDT-S considered where the leaf-conditional densities are separable in $\boldsymbol{x}$ as well as $(\boldsymbol{x}, y)$, the result actually amounts to ignoring the missing value, which is simple enough to implement. However, only the leaf-conditional separability of $\boldsymbol{x}$ and y (from each other) seems essential in the formulation of the methodology.

As mentioned in Section 3.3, the regularized histogram splitting/pruning cost function used in PCDT appears to have the necessary properties that Chou [20] and Chou et al. [22] list as allowing efficient and more general tree growing and pruning strategies to be used. In particular, it may be feasible to allow splitting by arbitrary hyperplanes instead of restricting the splits to hyperplanes that are perpendicular to the coordinate axes. Allowing more general splits might allow for better data fitting at a given level of model complexity and should therefore be investigated.

Splitting could be expedited at the higher levels of the tree by using a random subset of the data passed to the node in computing $\hat{\phi}$ instead of working with the entire data, since at these higher levels, there is usually more data than necessary. However, it is important that such subsampling *not* be done on either the crossvalidated pruning or on the fitting of mixtures to leaves, since this is where the complexity of the model is being automatically determined on the basis of the available data.

Another topic that should be investigated is the simultaneous use of crossvalidation and regularization in determining the appropriate splitting/pruning leaf-model complexity instead of using only regularization. The choice of $\hat{\phi}$ itself can probably be improved, and there is certainly room for improvement if one recalls the sizes of the gaps in the curves of Figure 3.4.4 between true relative entropy and crossvalidated cost. Recall that the main restriction on $\hat{\phi}$ is that it be easy to recompute as samples are shuffled from the right child to the left, as this shuffling must be done tens of thousands of times. There may be a way around this by finessing the search in some way. Even if there is not, it may be possible to find some sort of kernel estimate which shares the fast recomputability property of the histogram.

It was mentioned in Chapter 2 that there has been significant progress recently by Lugosi and Nobel [74], Nobel [81], Nobel and Olshen [82] and Devroye et al. [34, Chapters 20–23] in establishing various asymptotic properties of tree-structured and other data-driven statistical techniques. Typically, the proofs involved in demonstrating such properties require that the cells in the partition shrink to zero volume as the sample size gets large. It is not clear whether this condition is met in PCDT, as splits are made only if doing so will increase the average conditional likelihood of Y. More research is required to determine whether such asymptotic properties as consistency in L_2 can be established for PCDT, and if so, under what assumptions on the true underlying probability law.

In developing the PCDT algorithm, a number of choices were made for pragmatic reasons based on the assumption that current general-purpose workstations would be used. Certain operations are inherently parallelizable, such as the fitting of leaf mixtures. Further research may reveal specialized architectures that might make PCDT or related techniques more attractive computationally.

The Gaussian mixing posteriors used in PCDT-S occasionally exhibit the anomalous behavior of weighting leaves far-away in conditioning space more heavily than nearby ones. Specifically, this occurs when the tail of a far-away large-variance component becomes greater than that of a nearby narrow-variance one. The same phenomenon occurs in the version of Jacob and Jordan's mixtures-of-experts architecture [59] proposed by Gershenfeld et al. [44]. Although consistent with the view of conditioning space consisting of multiple classes, each characterized by a Gaussian density, this type of weighting is inconsistent with the locality-of-experts motivation that underlies much of the original mixture-of-experts framework. Characterizing, and if necessary compensating for the effect of, this non-local weighting may lead to an improvement in the performance of PCDT-S.

Perhaps the biggest question surrounding the PCDT methodology is how it might be made adaptive. A possibility is to use HMMs or stochastic context-free grammars to characterize and accommodate dynamics in the spatial domain. Often, it is desired that the adaptation take place on something besides the immediate observations, e.g., the estimated state in a hidden Markov model. One could in principle encode that state as an independent component of $\boldsymbol{x}$ and then have an adaptive model. PCDT-H would allow this, but the Markov mechanism would have to be specified separately.

6.2 Applications

Because the local statistics of natural images tend to vary from region to region, the lossless and lossy compression systems described in Sections 4.1 and Section 4.2 would need to be made adaptive in order to perform better. In the conjoint processing framework, such adaptivity would be most naturally achieved by making the conditional models themselves be adaptive.

Note that the image restoration procedure described in Section 4.5 was not as general as it might have been within the conjoint framework. In that application, incorporating feedback into the conditioning would help in certain cases. For example, in inverse-halftoning, the conditioning vectors formed from the "degraded" image have binary-valued coordinates, and therefore index into a restrictive set. Allowing feedback would introduce more greylevels into the restored image by requiring the restored pixels to be statistically compatible with the previously restored ones.

The multiresolution processing that was so successful in synthesizing textures can probably be applied advantageously in lossless compression and restoration (it has already been applied to lossy compression in Chapter 5). To apply it to lossless compression, use might be made of the fact that the system in Figure 4.2.1 can be made lossless in a straightforward way, as was discussed in Section 4.2. Multiresolution restoration seems particularly promising in light of the recent work by Simoncelli and Adelson on coring [132], wherein small-amplitude subband pixels are set to zero in a manner that the authors interpret in terms of a statistical model of the nonlinear interdependence of subband pixels.

6.3 Subband Coding

As mentioned in Chapter 5, the performance of a conjoint subband coding system depends on many factors, including the filter bank structure and the filter coefficients themselves, the order in which subband pixels are processed, the conditioning neighborhoods used, the complexities and types of conditional models, and the quantization parameters. Further research is required to determine the interaction of these factors, and to draw conclusions about the characteristics of a filter bank best suited to conjoint subband coding.

CHAPTER 7:

Conclusions

Motivated by the possibility of combining the implementation advantages of sequential processing with the performance advantages of joint processing, the potential use of high-order conditional density estimates in conjoint processing was investigated.

To enable effective conjoint processing, a suitable conditional density model had to be developed. The selected performance criterion for the model was predictive accuracy as measured by empirical conditional relative entropy. The CART methodology was used to partition conditioning space in an appropriate way; mixture densities were then fit to each leaf in the tree. A suitable splitting/pruning cost was developed, and suitable means were devised to determine the complexity of the tree and the leaf mixtures automatically from the data. This was accomplished using a histogram approximation to the would-be current mixture likelihood for the splitting/pruning cost and crossvalidation to determine the appropriate complexities of the leaf mixtures as well as that of the tree itself. It was found that a suitable complexity for the histogram approximation to the current likelihood could be selected reliably using Laplace's rule for regularization. This basic technique was then softened by estimating the posterior leaf membership probabilities using another mixture of Gaussians in the conditioning space, fitting exactly one component to each leaf, then multiplying by the leaf populations in the training data and normalizing to get the posteriors. These were then used as weights in mixing together the leaf-specific conditional densities, thereby in effect softening the splits. Finally, the technique was preceded by an initial linear regression step in a hybrid scheme. The technique and its two variants were found to perform well in experiments while having the additional desirable characteristic of automatically controlling their own complexity in an appropriate way. In examples involving simulated data and later in real applications involving images and textures, the proposed approaches were found to consistently outperform mixtures trained by standard algorithms, which we and others in the lab had been using previously. Gratifyingly, little if any performance cost seems to result from the added flexibility, though in general more training data is required for a given level of performance than in models that make more restrictive correct prior assumptions. Of course, if the restrictive prior assumptions are incorrect, then the more flexible scheme we have proposed enjoys a strong advantage.

The conjoint processing experiments on natural images and textures were successful and encouraging. We had no expectation of beating the best performance reported in the literature for the applications considered, because our models were not designed to adapt to changing source statistics. Still, the performance of conjoint lossless and lossy compression was respectable even in absolute terms. In the texture synthesis application, the conjoint approach was found to perform surprisingly well, resulting in synthesized textures which in some cases exhibited much more realistic structure than those obtained using standard approaches. Multiresolution texture synthesis using multiple neighborhoods and models and a coarse-to-fine pixel ordering was found to perform exceptionally well. In Chapter 5, the multiresolution texture synthesis procedure was reinterpreted in terms of filling in subband pixels in a pyramid. Based on the example presented there, it is conjectured that the visually best textures result from using the simplest (i.e., identity) filters. It is believed that simpler filters work better in this application because nonlinear dependence is easier for the conditional density models to discover or understand in the original pixel domain. Another example of apparently the same phenomenon was observed in the case of a filter-aided quantization system applied to broad-tailed i.i.d. sources. There, filtering that was beneficial when independent processing was used became significantly detrimental when joint processing (this time, vector quantization) was allowed. Based on these examples and the apparent mechanism of spatial smearing behind the phenomenon, it is conjectured that the best filters for conjoint or joint subband coding will have less frequency selectivity and more spatial selectivity than the filters considered best for standard subband coding. Although the experimental performance curves for conjoint subband coding presented there did not confirm this, those curves were based on a single choice for the many orderings, neighborhoods, and parameter values possible, and it is therefore difficult to draw any strong conclusions. The curves did seem to suggest, however, that the precise characteristics of the filter bank is less critical for conjoint than for traditional subband coding, and that conjoint subband coding can offer a performance advantage over traditional subband coding.

Finally, we note that the research described in this thesis has begun to find application in research done by other groups within the Media Lab. Application of our multiresolution method of texture synthesis to sound textures in joint work with Nicolas Saint-Arnaud [115] has already been noted in Chapter 5. Recently, Judith Brown [15] adapted our conjoint texture classification technique [96] to the problem of identifying musical instruments from sound samples, with encouraging results [16]. Eric Metois [78] and Bernd Schoner [119] trace the development of their *Cluster-Weighted Modeling* work [44] with Neil Gershenfeld to ideas gleaned from our papers on the application of mixture models to high-order conditional density estimation for texture synthesis [96] and compression [98].

APPENDIX A

Economized EM Algorithm for Mixture Estimation

This appendix presents a computationally economized, approximate version of the expectation-maximization (EM) algorithm for mixture density estimation, based on the standard version presented in Section 2.6. The degree of computational savings achieved is strongly data dependent, and it is even possible that the technique will result in no savings at all on a given training sequence. However, on the image data and image subband data considered in this thesis, the technique has generally resulted in a computational savings of between 10 and 25 percent. The fractional savings appears to increase slightly with the dimensionality of the observation space and the number of mixture components, but this dependence appears to be negligible relative to the effect of variations in the specific training sample.

The economized algorithm requires that the mixture component densities be separable and bounded. For concreteness, it is also assumed here that the components are Gaussian, though this assumption can be obviated by making minor modifications.

The savings is achieved in part by curtailing the computation of each posterior probability in the E-step as soon as it can be determined that that probability will be negligible. Once insignificant clusters have been so identified in the E-step, further computational savings is achieved in the M-step by simply eliminating the corresponding insignificant terms from the parameter-update summations. The latter savings is straightforward; the more interesting part is the E-step. To describe it, it is sufficient to consider the processing of a single training point $\boldsymbol{z}$. As elsewhere, the dimensionality of $\boldsymbol{z}$ is denoted by D_z, and the number of mixture components by M.

There is necessarily a degree of arbitrariness in selecting the threshold value of posterior probability required for a cluster to be deemed significant. It will prove convenient to specify the criterion for significance through a parameter $\xi \in [0, 1)$, as defined below. This parameter will be seen to have two related interpretations: as an upper bound on the fraction of the training-point likelihood that might be discarded by neglecting insignificant clusters (see the final inequality in (A.4), below), and as an absolute threshold such that clusters having posterior probabilities greater than ξ/M are guaranteed to be retained. The converse is not true; clusters having posterior probabilities less than ξ are not guaranteed to be discarded.

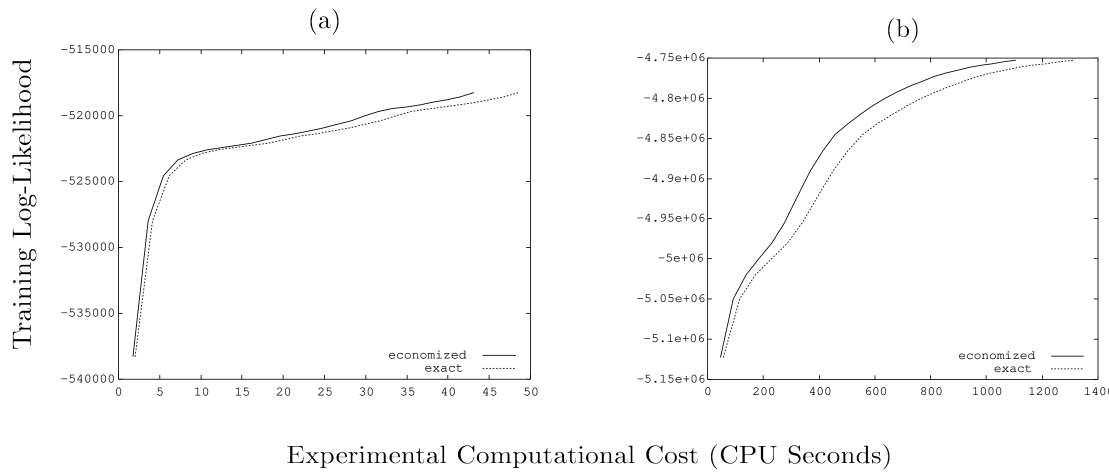

Figure A.1: The economized approximate EM algorithm typically results in a modest but useful computational savings over the exact algorithm, with greater savings occuring in the case of more complex mixtures. These graphs illustrate the difference in computational cost between the exact and approximate EM algorithms, for training data consisting of 5×5 neighborhoods (treated as 25-dimensional vectors) randomly selected from a collection of 26 monochrome natural images. Graph (a) corresponds to a training set of size $|\mathcal{L}| = 5,000$ and a mixture complexity of $M = 16$, while for graph (b) $|\mathcal{L}| = 50,000$ and $M = 64$. The computational savings in these cases, measured at the time of program termination at equal likelihood, are (a) 11.3%, and (b) 15.7%. The algorithms were compiled using Gnu C++ version 2.7.2 with the -O2 optimization flag specified, and executed on an HP 9000/755 workstation. The compuational costs in CPU seconds were calculated from actual counts of CPU clock cycles specifically used by the processes.

With $\xi \in [0,1)$ fixed, each cluster is considered in sequence. For the m^{th} cluster, a set of cluster indices $\mathcal{M}_m$ and a partial likelihood Q_m are defined recursively as

$$\mathcal{M}_m = \left\{ i \le m : P(i) f(\boldsymbol{z}|i) \ge \frac{\xi}{M} Q_{i-1} \right\} \tag{A.2a}$$

$$Q_m = \sum_{i \in \mathcal{M}_m} P(i) f(\boldsymbol{z}|i) \tag{A.2b}$$

with $\mathcal{M}_0$ taken to be the empty set. $\mathcal{M}_m$ is the set of indices of the clusters deemed significant among those considered thus far, while Q_m can be interpreted as the portion of the training-point likelihood $f(\boldsymbol{z})$ accounted for by those clusters. Trivially, the following inequalities hold for every $\xi \in [0,1)$ and every $m \in \{1, \ldots, M\}$:

$$0 \le Q_m \le f(\boldsymbol{z}) \tag{A.3a}$$

$$\emptyset \subseteq \mathcal{M}_m \subseteq \{1, \ldots, M\} \tag{A.3b}$$

Note that in order that the upper bounds hold with equality, it is sufficient (but not necessary) that $\xi = 0$ and $m = M$.

Now let ${\mathcal{M}_m}^{\mathrm{C}}$ denote the complement of $\mathcal{M}_m$ with respect to $\{1,\ldots,M\}$. Then Q_M satisfies

$$\begin{aligned} f(\boldsymbol{z}) &= \sum_{m\in\mathcal{M}_M} P(m)f(\boldsymbol{z}|m) + \sum_{m\in{\mathcal{M}_M}^{\mathrm{C}}} P(m)f(\boldsymbol{z}|m) \\ &\geq Q_M \\ &\geq f(\boldsymbol{z}) - \sum_{m\in{\mathcal{M}_M}^{\mathrm{C}}} \frac{\xi}{M} f(\boldsymbol{z}) \\ &\geq (1-\xi)f(\boldsymbol{z}) \end{aligned} \tag{A.4}$$

This chain of inequalities establishes that, for small ξ, Q_M is a good approximation to the training-point likelihood $f(\boldsymbol{z})$. Thus, equations (A.2a) and (A.2b), along with the initial condition $\mathcal{M}_0 = \emptyset$, define an algorithm for approximating the likelihood with bounded fractional approximation error.

After $\mathcal{M}_M$ and Q_M have been computed in a first pass through the clusters, the posterior probability $P(m|\boldsymbol{z})$ for each cluster may be approximated in a second pass as

$$P(m|\boldsymbol{z}) \approx \begin{cases} P(m)f(\boldsymbol{z}|m)/Q_M & \text{if } m\in\mathcal{M}_M \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $P(m|\boldsymbol{z}) > \xi/M$, then $f(\boldsymbol{z}) > Q_M\xi/M$, so that clusters with significant posteriors are in fact guaranteed to be retained, albeit with approximated posterior cluster-membership probabilities.

If $\boldsymbol{z}$ is such that $|\mathcal{M}_M| \ll M$ (which is usually the case if M is large), then the comparison called for in (A.2a) can be carried out efficiently as follows. Separability, boundedness, and Gaussianity of the mixture components are now assumed. The likelihood of the m^{th} component density is

$$f(\boldsymbol{z}|m) = \prod_{d=1}^{D_z} (2\pi\sigma_{md}^2)^{-\frac{1}{2}} \exp(-\frac{1}{2}(z_d-\mu_{md})^2/\sigma_{md}^2).$$

Define the auxilliary quantity $\underline{\sigma}_{md}^2$ for $d = 0,\ldots,D_z-1$ as

$$\underline{\sigma}_{md}^2 = \min_{j>d}\{\sigma_{mj}^2\}$$

The boundedness of $f(\boldsymbol{z}|m)$ implies $\underline{\sigma}_{md}^2 > 0$. To determine whether $P(m)f(\boldsymbol{z}|m) \geq \frac{\xi}{M}Q_{m-1}$, only the first d_{0m} coordinates of $\boldsymbol{z}$ need be considered, where

$$d_{0m} = \begin{cases} \min\{d : (2\pi\underline{\sigma}_{md}^2)^{\frac{D_z-d}{2}} P(m)\prod_{j=1}^{d} f(z_j) < \frac{\xi}{M}Q_{m-1}\} & \text{if such an } n \text{ exists;} \\ D_z & \text{otherwise.} \end{cases}$$

This is the part of the economized algorithm that effectively generalizes the partial distance method [45] of fast codebook lookup to the probabilistic setting, as alluded to in Section 2.6.

A remaining issue concerns those training points which are initially outliers for every cluster. Even when the mixture parameters are initialized by the generalized Lloyd algorithm using Voronoi-cell-specific statistics, it occasionally happens that $f(\boldsymbol{z})$ is vanishingly small so as to cause numerical difficulty in evaluating the summation called for in (A.2a). In such cases, the numerical difficulty can be removed by replacing Q_m with

$$Q'_m = \max_{i \in \mathcal{M}_m} \{P(i) f(\boldsymbol{z}|i)\},$$

then working in the logarithmic domain. This substitution has been found to result in a useful estimate of the posterior cluster-membership PMF for outliers, although the final inequality in (A.4) is no longer automatically satisfied.

Note that an infinitessimal value for $f(\boldsymbol{z})$ is not inconsistent with a meaningful posterior cluster-membership PMF. In fact, such outliers should be allowed to — and are expected to — exert considerable influence on the evolution of parameter values of the clusters to which they belong.

APPENDIX B

Variable-Localization Filter Banks

To facilitate study of the effect of the space-frequency localization tradeoff on the performance of systems that employ criticially sampled filter banks, it is desirable to have a collection of filter banks which differ in this tradeoff but which are otherwise as identical as possible. The uncertainty principle for continuous signals [128] limits the degree of localization that can be achieved simultaneously in the spectral and spatial (or temporal, in the case of audio signals) domains. It is well known that signals with a Gaussian envelope achieve the limit when the product of the second central moments in the two domains is used as the joint localization criterion. For discrete-space or discrete-time signals, such a product is minimized trivially by any filter having a single nonzero coefficient and a correspondingly flat spectrum; hence the motivation for Gaussian envelopes does not carry over to discrete time. In particular, sampled Gaussian-envelope responses are not guaranteed to be optimally localizing in any usual sense of the term (though qualitatively they may seem reasonable). More importantly for present purposes, given the absence of a corresponding localization criterion, there does not seem to be a natural way of incorporating the objective of near-Gaussianity into a design procedure for filter banks.

An alternative joint-localization criterion directly applicable to the discrete case was proposed by Slepian [134], and one we specifically devised for the design of nonuniform filterbanks for image processing was proposed in [92]. The latter criterion is quite complex in its generality, but simplifies greatly in the case of a uniform two-band orthogonal filter bank, becoming a weighted sum of the spatial and spectral second central moments. A numerical procedure for designing such filter banks is summarized below. At the end of this appendix, a set of resulting filter banks varying in localization tradeoff is tabulated; these were used in the experiments described in Chapter 5.

It is assumed that the filter bank is as shown in Figure 2.1.1, with $K = 2$, and the subsampling performed out-of-phase. That is, the even samples are retained in one band and the odd in the other. Let the region of support be $\{-L, \ldots, L\}$ for all of the filters. Subject to the constraint of perfect overall reconstruction and the additional constraints

$$h_1(n) = g_1(n) = (-1)^n g_2(n) = h_2(n), \qquad n \in \{-L, \ldots, L\}; \tag{B.1}$$

$$h_1(n) = h_1(-n), \qquad n \in \{-L, \ldots, L\}; \tag{B.2}$$

$$\sum_{n=-L}^{L} h_1^2(n) = 1; \tag{B.3}$$

$$\sum_{n=-L}^{L} h_2(n) = 0, \tag{B.4}$$

the object is to choose $\{h_1(n)\}$ to minimize the joint-localization criterion

$$\int_0^{\pi} \omega^2 |H_1(\omega)|^2 d\omega + \alpha \sum_{n=-L}^{L} n^2 h_1^2(n), \tag{B.5}$$

where

$$H_1(\omega) = \sum_{n=-L}^{L} h_1(n) e^{-j\omega n},$$

and where α is a parameter intended to control the tradeoff between spatial and spectral localization. Constraints (B.1) and (B.3) together imply strict orthogonality, while (B.4) ensures that DC doesn't leak into the high band. The symmetry constraint (B.2) is necessary for the filters to have zero phase, which is generally desirable in image processing applications.

Since the gradient of (B.5) is readily computable, the required optimization search is straightforward. The adaptive hill-descending technique described in [92] was used to obtain filter banks having the low-pass analysis impulse responses tabulated in Table B.6. These responses are also presented graphically in Figure 5.2.2, while the corresponding magnitude-frequency responses are shown in Figure 5.2.1. To speed the optimization and to reduce the chances of falling into an undesirable local minimum, the orthogonal nine-tap filters proposed in [131] were used as initial guesses after appropriately padding by zeroes.

Table B.6: Numerically Computed Low-Band Filter Bank Coefficients

n	$\alpha = 5$	$\alpha = 1$	$\alpha = 0.5$	$\alpha = 0.1$
-9	-0.000678	0.000025	0.000022	-0.000387
-8	-0.001779	-0.000272	0.000956	0.006392
-7	0.003983	0.000454	-0.001733	-0.007325
-6	-0.005156	-0.002772	-0.004758	-0.017299
-5	0.007218	0.005809	0.010560	0.030681
-4	-0.009002	0.010479	0.021303	0.037582
-3	0.008213	-0.035557	-0.060045	-0.104877
-2	-0.068062	-0.070450	-0.068467	-0.055986
-1	0.334857	0.382826	0.404758	0.435601
0	0.875205	0.833145	0.809056	0.765990
1	0.334857	0.382826	0.404758	0.435601
2	-0.068062	-0.070450	-0.068467	-0.055986
3	0.008213	-0.035557	-0.060045	-0.104877
4	-0.009002	0.010479	0.021303	0.037582
5	0.007218	0.005809	0.010560	0.030681
6	-0.005156	-0.002772	-0.004758	-0.017299
7	0.003983	0.000454	-0.001733	-0.007325
8	-0.001779	-0.000272	0.000956	0.006392
9	-0.000678	0.000025	0.000022	-0.000387

REFERENCES

[1] K. Abend, T. J. Harley, and L. N. Kanal. Classification of binary patterns. *IEEE Transactions on Information Theory*, IT-11:538–544, 1980.

[2] Robert B. Ash. *Real Analysis and Probability.* Academic Press, New York, 1972.

[3] Andrew R. Barron and Thomas M. Cover. Minimum complexity density estimation. *IEEE Trans. on Information Theory*, 37(4):1034–1054, July 1991.

[4] Andrew R. Barron, László Györfi, and Edward C. van der Meulen. Distribution estimation consistent in total variation and in two types of information divergence. *IEEE Transactions on Information Theory*, 38(5):1437–1454, September 1992.

[5] W.R. Bennett. Spectra of quantized signals. *Bell System Technical Journal*, 27:446–472, July 1948.

[6] Toby Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[7] Toby Berger. Minimum entropy quantizers and permutation codes. *IEEE Transactions on Information Theory*, IT-28(2):149–156, March 1982.

[8] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1974.

[9] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[10] Richard E. Blahut. Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, IT-18(4):460–473, July 1972.

[11] M.B. Brahmanandam, S. Panchanathan, and M. Goldberg. An affine transform based image vector quantizer. In *Proceedings of the SPIE Visual Communications and Image Processing '93 Conference*, volume 2094, pages 1639–1648, Cambridge, MA, November 1993.

[12] Leo Breiman. Simplicity vs. accuracy — Heisenberg's principle in statistics. Abstract from a talk given at the 1996 the AMS-IMS-SIAM Joint Summer Research Conference on Adaptive Selection of Models and Statistical Procedures, June 1996. Available by request from this author.

[13] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Wadsworth& Brooks/Cole, Pacific Grove, CA, 1984.

[14] P. Brodatz. *Textures: A Photographic Album for Artists and Designers.* Dover, New York, 1966.

[15] Judith C. Brown. Cluster-based probability model for musical instrument identification. *Journal of the Acoustical Society of America*, 102(2), May 1997.

[16] Judith C. Brown. private communication. MIT Media Lab, January 1997.

[17] Robert W. Buccigrossi and Eero P. Simoncelli. Progresive wavelet image coding based on a conditional probability model. In *ICASSP-97: 1997 International Conference on Acoustics, Speech, and Signal Processing*, Munich, Germany, April 1997. IEEE.

[18] James A. Bucklew and Gary L. Wise. Multidimensional asymptotic quantization theory with rth power distortion measures. *IEEE Trans. Inform. Theory*, IT-28:239–247, March 1982.

[19] Edwin A. Burtt. *The Metaphysical Foundations of Modern Physical Science.* Humanities Press, Atlantic Highlands, NJ, 1980.

[20] Philip A. Chou. Optimal partitioning for classification and regression trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):340–354, April 1991.

[21] Philip A. Chou and Tom Lookabaugh. Conditional entropy-constrained vector quantization of linear predictive coefficients. In *Proceedings of the 1990 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 197–200, Albuquerque, NM, April 1990.

[22] Philip A. Chou, Tom Lookabaugh, and Robert M. Gray. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Transactions on Information Theory*, 35(2):299–315, March 1989.

[23] Ronald Christensen. *Foundations of Inductive Reasoning.* Entropy Ltd., Lincoln, Mass., 1980.

[24] Ronald A. Christensen, Kris Popat, and Thomas A. Reichert. Expert resolution using maximum entropy. *Management Science*, 1997. in preparation.

[25] Pamela C. Cosman, Robert M. Gray, and Martin Vetterli. Vector quantization of image subbands: a survey. *IEEE Transactions on Image Processing*, 5(2):202–225, February 1996.

[26] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley and Sons, 1991.

[27] Imre Csiszár. Information type measures of difference of probability distributions and indirect observations. *Studia Sci. Math. Hungar.*, 2:229–318, 1967.

[28] Imre Csiszár and János Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems.* Academic Press, 1981.

[29] Lee D. Davisson. Universal noiseless coding. *IEEE Trans. on Information Theory*, IT-19:783–795, November 1973.

[30] A.P. Dawid. Present position and potential developments: some personal views, statistical theory, the prequential approach. *Journal of the Royal Statistical Society, Series A*, 147, Part 2:278–292, 1984.

[31] A. P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc.*, 39:1–38, 1977.

[32] Pierre A. Devijver and Josef Kittler. *Pattern Recognition: a Statistical Approach.* Prentice-Hall International, Engelwood Cliffs, NJ, 1982.

[33] Luc Devroye and László Györfi. *Nonparametric Density Estimation: the L_1 View.* Wiley, New York, 1985.

[34] Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, New York, 1996.

[35] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[36] Michelle Effros, Philip A. Chou, and Robert M. Gray. Weighted universal image compression. Submitted for publication.

[37] B.S. Everitt and D.J. Hand. *Finite Mixture Distributions.* Chapman and Hall, London, 1981.

[38] Terrence L. Fine. *Theories of Probability; an Examination of Foundations.* Academic Press, New York, 1973.

[39] Jerome H. Friedman, Werner Stuetzle, and Ann Schroeder. Projection pursuit density estimation. *Journal of the American Statistical Association*, 79(387):599–608, September 1984.

[40] K. Fukunaga and T.E. Flick. Estimation of the parameters of a gaussian mixture using the method of moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(4):410–416, July 1983.

[41] Robert G. Gallager. *Information Theory and Reliable Communication.* Wiley, 1968.

[42] Robert G. Gallager. Source coding for subband coding. Unpublished manuscript, Codex Corp., August 1985.

[43] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis.* Chapman and Hall, 1995.

[44] Neil Gershenfeld, Bernd Schoner, and Eric Metois. Cluster-weighted modeling for time-series prediction and characterisation. Preprint, 1997.

[45] Allen Gersho and Robert M. Gray. *Vector Quantization and Signal Compression.* Kluwer academic publishers, 1992.

[46] H. Gish and J.N. Pierce. Asymptotically efficient quantization. *IEEE Transactions on Information Theory*, IT-14:676–683, September 1968.

[47] Robert M. Gray. Information rates of autoregressive processes. *IEEE Transactions on Information Theory*, IT-16:412–421, 1970.

[48] Robert M. Gray. *Probability, Random Processes, and Ergodic Properties.* Springer-Verlag, New York, 1988.

[49] D.J. Hand. *Kernel Discriminant Analysis.* Research Studies Press, 1982.

[50] M. Hassner and J. Sklansky. The use of Markov random fields as models of texture. *Computer Graphics and Image Processing*, 12:357–370, 1980.

[51] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, 1994.

[52] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the IEEE International Conference on Image Processing*, volume 3, pages 648–651, Washington, DC, October 1995.

[53] D.J. Heeger and J.R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH '95*, pages 229–238, Los Angeles, CA, August 1995.

[54] E.G. Henrichon and K.S. Fu. On mode estimation in pattern recognition. In *Proceedings of the 7th Symposium on Adaptive Processes*, pages 3–a–1. UCLA, 1968.

[55] Henry Holtzman. Increasing resolution using pyramid coding and pattern matching. Available by request from this author, May 1990.

[56] J.-Y. Huang and P.M. Schultheiss. Block quantization of correlated Gaussian random variables. *IEEE Transactions on Communications*, CS-11:289–296, September 1963.

[57] Peter J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

[58] Toshihide Ibaraki and Naoki Katoh. *Resource Allocation Problems : Algorithmic Approaches.* MIT Press, Cambridge, MA, 1988.

[59] R.A. Jacobs, M.I. Jordan, and S.J. Nowlan. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[60] Anil K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, N.J., 1989.

[61] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, October 1993.

[62] Nuggehally S. Jayant and Peter Noll. *Digital Coding of Waveforms : Principles and Applications to Speech and Video.* Prentice-Hall, Englewood Cliffs, NJ, 1984.

[63] Tony Jebara. Conditionally maximized density estimation for incomplete data via the CEM method. Class project report, MIT Course 9.390: Computational Laboratory in Cognitive Science, 1997.

[64] Tony Jebara and Kris Popat. Methods for conditionally maximized likelihood density estimation. Technical report, M.I.T. Media Laboratory, 1997. In preparation.

[65] Michael I. Jordan. private communication. MIT, April 1997.

[66] Laveen N. Kanal. Markov mesh models. *Computer Graphics and Image Processing*, 12:371–375, 1980.

[67] R. L. Kashyap. Characterization and estimation of two-dimensional ARMA models. *IEEE Trans. Inform. Theory*, IT-30(5):736–745, 1984.

[68] Glen G. Langdon. An introduction to arithmetic coding. *IBM J. Res. Develop.*, Mar. 1984.

[69] Glen G. Langdon and Jorma J. Rissanen. Compression of black-white images with arithmetic coding. *IEEE Trans. Comm.*, COM-29:858–867, June 1981.

[70] S.P. Lloyd. Least squares quantization in PCM. Unpublished Bell Laboratories Technical Note, 1957. Reprinted in *IEEE Transactions on Information Theory*, vol. IT-28, March 1982, pp. 127–135., 1957.

[71] Tom D. Lookabaugh and Robert M. Gray. High-resolution quantization theory and the vector quantizer advantage. *IEEE Trans. Inform. Theory*, IT-35:1020–1033, 1989.

[72] David G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, Massachusetts, 2nd edition, 1989.

[73] Mark R. Luettgen. *Image Processing with Multiscale Stochastic Models*. PhD thesis, Massachusetts Institute of Technology, May 1993.

[74] Gabor Lugosi and Andrew Nobel. Consistency of data-driven histogram methods for density estimation and classification. Technical Report UIUC-BI-93-01, Beckman Institute, University of Illinois at Urbana-Champaign, 1993.

[75] Stephen P. Luttrell. The partitioned mixture distribution: multiple overlapping density models. In *Proceedings of the 14th International Maximum Entropy Workshop*, Malvern, U.K., 1994. Defense Research Agency. Preprint.

[76] Geoffrey J. McLachlan and Kaye E. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

[77] N. Merhav and M. Feder. Universal schemes for sequential decision from individual data sequences. *IEEE Transactions on Information Theory*, 39(4):1280–1292, July 1993.

[78] Eric Metois. *Musical Sound Information: Musical gesture and Embedding synthesis*. PhD thesis, Massachusetts Institute of Technology, 1996.

[79] James R. Munkres. *Topology: A First Course*. Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[80] S. Na and D.L. Neuhoff. Bennett's integral for vector quantizers. *IEEE Transactions on Information Theory*, 41(4):886–900, July 1995.

[81] Andrew Nobel. Histogram regression estimation using data-dependent partitions. Technical Report UIUC-BI-94-01, Beckman Institute, University of Illinois at Urbana-Champaign, 1994.

[82] Andrew Nobel and Richard Olshen. Termination and continuity of greedy growing for tree-structured vector quantizers. *IEEE Transactions on Information Theory*, 42(1):191–205, January 1996.

[83] Bruno Olshausen. private communication. MIT Media Lab, July 1996.

[84] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill,

New York, 1991.

[85] Emanuel Parzen. *Modern Probability Theory and its Applications.* Wiley, 1960.

[86] R. Pasco. *Source coding algorithms for fast data compression.* PhD thesis, Stanford University, 1976.

[87] W.B. Pennebaker and J.L. Mitchell. Probability estimation for the q-coder. *IBM Journal of Research and Development*, 32(6):737–752, November 1988.

[88] Alex Pentland. Fractal-based image compression and interpolation. U.S. Patent No. 5,148,497, 1992.

[89] Alex Pentland and Bradley Horowitz. A practical approach to fractal-based image compression. In *Proc. IEEE Data Comp. Conf.*, Utah, 1991.

[90] 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 Transactions on Image Processing*, 5(2):347–360, February 1996.

[91] David K. Pickard. Inference for discrete Markov fields: the simplest nontrivial case. *Journal of the American Statistical Association*, 82(397):90–96, 1987.

[92] Ashok Popat, Wei Li, and Murat Kunt. Numerical design of parallel multiresolution filter banks for image coding applications. In *Proceedings of the SPIE 1991 International Symposium on Optical Applied Science and Engineering*, volume 1567, pages 341–353, San Diego, California, July 1991. SPIE–the International Society for Optical Engineering, SPIE.

[93] Ashok Popat, André Nicoulin, Andrea Basso, Wei Li, and Murat Kunt. Subband coding of video using energy-adaptive arithmetic coding and statistical feedback-free rate control. In *Proceedings of the Visual Communications and Image Processing '91 Conference*, volume 1605, pages 940–953, Boston, Massachusetts, November 1991. SPIE–the International Society for Optical Engineering, SPIE.

[94] Ashok C. Popat. Scalar quantization with arithmetic coding. Master's thesis, Dept. of Elec. Eng. and Comp. Science, M.I.T., Cambridge, Mass., 1990.

[95] Kris Popat and Rosalind Picard. Cluster-based probability model and its application to image and texture processing. *IEEE Transactions on Image Processing*, 6(2):268–284, February 1997.

[96] Kris Popat and Rosalind W. Picard. A novel cluster-based probability model for texture synthesis, classification, and compression. In *Proc. SPIE Visual Communications '93*, pages 756–768, Cambridge, Mass., 1993.

[97] Kris Popat and Rosalind W. Picard. Cluster-based probability model applied to image restoration and compression. In *ICASSP-94: 1994 International Conference on Acoustics, Speech, and Signal Processing*, pages 381–384, Adelaide, Australia, April 1994. IEEE.

[98] Kris Popat and Rosalind W. Picard. Exaggerated consensus in lossless image compression. In *ICIP-94: 1994 International Conference on Image Processing*, pages 846–850, Austin, TX, Nov. 1994. IEEE.

[99] Kris Popat and Kenneth Zeger. Robust quantization of memoryless sources using dispersive FIR filters. *IEEE Transactions on Communications*, 40(11):1670–1674, November 1992.

[100] P. Pudil, J. Novovicova, and J. Kittler. Simultaneous learning of decision rules and important attributes for classification problems in image analysis. *Image and Vision Computing*, 12(3):193–198, April 1994.

[101] Majid Rabbani and Paul W. Jones. *Digital Image Compression Techniques*. SPIE Optical Engineering Press, Bellingham, Washington, 1991.

[102] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[103] Ronald H. Randles and Douglas A. Wolfe. *Introduction to the Theory of Nonparametric Statistics*. Wiley, New York, 1985.

[104] Richard A. Redner and Homer F. Walker. Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, 26(2):195–239, April 1984.

[105] Nicholas Rescher. *Induction, an Essay on the Justification of Inductive Reasoning*. University of Pittsburgh Press, 1980.

[106] Jorma J. Rissanen. Modeling by shortest desciption length. *Automatica*, 14:465–471, 1978.

[107] Jorma J. Rissanen. A universal data compression system. *IEEE Trans. on Information Theory*, IT-29(5):656–664, September 1983.

[108] Jorma J. Rissanen. A predictive inference principle for estimation. Technical report, IBM Research Lab, San Jose, CA, 1984. This report was no longer on file at IBM at the time of this writing. In case of difficulty in obtaining one, a copy is available by request from this author.

[109] Jorma J. Rissanen. Minimum-description-length principle. In Samuel Kotz, Norman Lloyd Johnson, and Campbell B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 5, pages 523–527. Wiley, 1985.

[110] Jorma J. Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, Singapore, 1989.

[111] Jorma J. Rissanen. Stochastic complexity and its applications. Unpublished report, available by request from this author, 1996.

[112] Jorma J. Rissanen and Glen G. Langdon. Arithmetic coding. *IBM J. Res. Develop.*, 23(2):149–162, March 1979.

[113] Jorma J. Rissanen and Glen G. Langdon. Universal modeling and coding. *IEEE Trans. Inform. Theory*, IT-27:12–23, 1981.

[114] E.B. Saff and A.D. Snider. *Fundamentals of Complex Analysis for Mathematics, Science, and Engineering*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[115] Nicolas Saint-Arnaud and Kris Popat. Analysis and synthesis of sound textures. In *Proceed-*

ings of IJCAI-95 Two-Day Workshop on Computational Auditory Scene Analysis, Montreal, August 1995.

[116] Yosiyuki Sakamoto. *Akaike Information Criterion Statistics.* KTK Scientific Publishers, Hingham, MA, 1986.

[117] Lucia Sardo and Josef Kittler. Complexity analysis of RBF networks for pattern recognition. In *Proceedings of the 1996 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 574–579, San Fransisco, June 1996.

[118] Lucia Sardo and Josef Kittler. Predictive performance for the detection of underfitting in density estimation. In *Proceedings of the IEE International Conference on Artificial Neural Networks*, pages 24–28, 1997.

[119] Bernd Schoner. private communication. MIT Media Lab, August 1997.

[120] David W. Scott. *Multivariate Density Estimation.* John Wiley & Sons, New York, 1992.

[121] D.W. Scott and J.R. Thompson. Probability density estimation in higher dimensions. In J.E. Gentle, editor, *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface*, pages 173–179, Amsterdam, 1983. North-Holland.

[122] Takanori Senoo and Bernd Girod. Vector quantization for entropy coding of image subbands. *IEEE Transactions on Image Processing*, 1:526–533, Oct. 1992.

[123] Nong Shang. Tree-structured density estimation and dimensionality reduction. In J. Sall and A. Lehman, editors, *Proceedings of 26th Symposium on the Interface of Computing Science and Statistics (INTERFACE '94)*, volume 26, pages 172–176, Fairfax Station, VA, June 1994. Interface Found. North America.

[124] Claude E. Shannon. A mathematical theory of communication. *Bell Sys. Tech. Journal*, 27, 1948.

[125] Claude E. Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE National Convention Record, Part 4*, pages 142–163, 1959.

[126] Jerome M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3345–3462, December 1993.

[127] Y. Shoham and A Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics Speech and Signal Processing*, ASSP-36(9):1445–1453, September 1988.

[128] William McC. Siebert. *Circuits, Signals, and Systems.* MIT Press, Cambridge, MA, 1986.

[129] B. W. Silverman. Penalized maximum likelihood estimation. In Samuel Kotz, Norman Lloyd Johnson, and Campbell B. Read, editors, *Encyclopedia of Statistical Sciences*, volume 6, pages 664–667. Wiley, 1982.

[130] B. W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, London, 1986.

[131] Eero P. Simoncelli and Edward H. Adelson. Subband transforms. In *Subband Image Coding*, pages 143–192. Kluwer, Boston, 1991.

[132] Eero P. Simoncelli and Edward H. Adelson. Noise removal via Bayesian wavelet coring. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 379–382, Lausanne, Switzerland, September 1996.

[133] Eero P. Simoncelli, William T. Freeman, Edward H. Adelson, and David J. Heeger. Shiftable multi-scale transforms. *IEEE Transactions on Information Theory*, 38:587–607, 1992.

[134] D. Slepian. Prolate spheroidal wave functions, Fourier analysis and uncertainty – V: the discrete case. *Bell System Technical Journal*, 58:1371–1430, May-June 1978.

[135] S.T. Sum and B.J. Oommen. Mixture decomposition for distributions from the exponential family using a generalized method of moments. *IEEE Transactions on Systems, Man and Cybernetics*, 25(7):1139–1149, 1995.

[136] Richard Swinburne. *The Justification of Induction*. Oxford University Press, 1974.

[137] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[138] Martin Vetterli and Jelena Kovacevic. *Wavelets and Subband Coding*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

[139] Georg Henrik von Wright. *The Logical Problem of Induction*. Macmillan, 1957.

[140] Roy Weatherford. *Philosophical Foundations of Probability Theory*. Routledge and Kegan Paul, London, 1982.

[141] Marcelo J. Weinberger, Jorma J. Rissanen, and Ronald B. Arps. Applications of universal context modeling to lossless compression of gray-scale images. *IEEE Transactions on Image Processing*, 5(4):575–586, April 1996.

[142] Frans M.J. Willems, Yuri M. Shtarkov, and Tjalling J. Tjalkens. The context-tree weighting method: basic properties. *IEEE Trans. on Information Theory*, 41(3):653–664, May 1995.

[143] J.W. Woods and S.D. O'Neil. Subband coding of images. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34:1278–1288, October 1986.

[144] S.W. Wu and A. Gersho. Rate-constrained picture-adaptive quantization for jpeg baseline coders. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 389–392, Minneapolis, MN, April 1993.

[145] Song Chun Zhu, Yingnian Wu, and David Mumford. Filters, random fields, and maximum entropy (FRAME) — towards a unified theory for texture modeling. Technical Report 95-02, Harvard University Robotics Laboratory, Cambridge, MA 02138, 1995.