M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 351
Also appears in the IEEE Trans. on Image Processing, Vol. 6, No. 2, February 1997, pp. 268–284

Cluster-based probability model and its application to image and texture processing

Kris Popat and Rosalind W. Picard*
Rm E15-383, The MIT Media Laboratory, 20 Ames St., Cambridge, MA 02139
popat@media.mit.edu, picard@media.mit.edu Tel: (617) 253-6271 Fax: (617) 253-8874
EDICS Number: IP 1.9 (with applications to 1.1, 1.4, and 1.5)

Abstract

We develop, analyze, and apply a specific form of mixture modeling for density estimation, within the context of image and texture processing. The technique captures much of the higher-order, nonlinear statistical relationships present among vector elements by combining aspects of kernel estimation and cluster analysis. Experimental results are presented in the following applications: image restoration, image and texture compression, and texture classification.

1 Introduction

In many signal processing tasks, uncertainty plays a fundamental role. Examples of such tasks are compression, detection, estimation, classification, and restoration — in all of these, the future inputs are not known perfectly at the time of system design, but instead must be characterized only in terms of their "typical," or "likely" behavior, by means of some *probabilistic model*. Every such system has a probabilistic model, be it explicit or implicit. Often, the level of performance achieved by such a system depends strongly on the accuracy of the probabilistic model it employs.

This paper presents a method for finding an explicit probability distribution estimate, and demonstrates its application to a variety of image processing problems. In particular, the focus is on obtaining accurate estimates of conditional distributions, where the number of conditioning variables is relatively large (on the order of ten). If conditional distributions are estimated directly, then care must be taken to ensure consistency [1]. In this work, we begin by estimating the joint distribution — in this way, we avoid consistency problems. Once the joint distribution has been estimated, the conditional can be computed by a simple normalization.

As mentioned, the goal here is obtaining a high-dimensional joint probability distribution, i.e., on the order of $d = 10$ joint variables. Traditional attempts usually stop at $d = 3$ variables or less. Major obstacles exist when estimating high-d distributions [2, 3]. Foremost is the exponential growth of the amount of data required to obtain an estimate of prescribed quality as d is increased. Large regions in the d-dimensional space are likely to be devoid of observations. In the discrete case, the size of the vector alphabet is usually astronomical — for example, a 3×3 neighborhood of 8-bit pixels can assume 2^{72} distinct values. Consequently, processing in the vector alphabet must be bypassed altogether. The situation is no better when the conditional distribution formulation is used. Although the variable is then one-dimensional, the number of conditioning states replaces the vector alphabet size as the astronomical quantity, and the same situation follows. These obstacles are consequences of what is commonly referred to as "the curse of dimensionality."[1] A term that is more specific to the density estimation problem, coined by Scott and Thompson [3], is *the empty space phenomenon*.

The estimation technique described in this paper combines two weapons in combating the empty space phenomenon: kernel estimation and cluster analysis. Kernel estimation (reviewed in Section 1.2) provides a means of interpolating probability to fill in empty regions. It is a means of *generalizing* the observed data. However, kernel estimation is poor at modeling rare events (tail regions), and in high dimensions, almost all events are rare. On the other hand, cluster analysis identifies critical regions of space that need to be covered by kernels, and is a means of *summarizing* the observed data. The combination of these two techniques results in an economized, heterogeneous kernel estimate that works well in both mode and tail regions.

The cluster-based probability model is a type of mixture model, and mixture models are not new. Their estimation and use dates back at least to the 1894 work of Karl Pearson; see [5] or [6] for a survey. Mixture models have customarily been used in situations calling for unsupervised learning. Specifically, a mixture model naturally arises when an observation $\mathbf{x}$ is believed to obey probability law $p_c(\mathbf{x} \,|\, \omega_c)$ with probability $P(\omega_c)$, where ω_c is one of several "states of nature," or classes [7]. Alternatively, a mixture model may be viewed as a means of estimating an arbitrary probability law, even in situations where there is no reason to believe that the true probability law is a mixture [5, p. 118ff]. The cluster-based probability model is viewed in this way.

Mixture models have received considerable attention from the speech processing community over the past two decades [8]. They are also a topic of current interest among researchers in the field of artificial neural networks (ANN's) [9], where the emphasis has been on estimating the system output values themselves, rather than on estimating predictive probability distributions for those values (see Section 6.2). However, the use of mixture models, and more generally, the application of high-dimensional probabilistic modeling, are subjects which are rarely dealt with in the image processing literature. The current paper develops, analyzes, and applies a particular type of mixture for high-dimensional probabilistic modeling, within the context of image and texture processing.

1.1 Terms and notation

Let $\mathbf{x} = [x_1, \ldots, x_d]$ be a random vector, and let $\mathbf{X}$ and X_i denote particular values of $\mathbf{x}$ and x_i respectively. It is assumed that the d elements of $\mathbf{x}$ share the same range of values $\mathcal{X} \subset \mathcal{R}$. In the continuous case, $\mathcal{X}$ is assumed to be a bounded interval of the real line. In the discrete case, $\mathcal{X}$ is assumed to be a set of K real numbers on a bounded interval. The set of possible values of $\mathbf{x}$ is denoted $\mathcal{X}^d \subset \mathcal{R}^d$; it is the d-fold cartesian

*This work was supported in part by HP Labs and NEC Corp.

[1] D.W. Scott [4] has attributed the first use of this term to R.E. Bellman in describing the exponential growth with dimension of the complexity of combinatorial optimization.

product of $\mathcal{X}$ with itself. Note that, in the discrete case, the number of possible values for $\mathcal{X}^d$ is K^d.

It is assumed that successive realizations of $\mathbf{x}$ are independent and that they obey one and the same probability law.[2] In the continuous case, $\mathbf{x}$ is governed by a probability density function (PDF) $f(\mathbf{x})$, which satisfies

$$\int_V f(\mathbf{X})d\mathbf{X} = \mathrm{Prob}\{\mathbf{x} \in V\} \tag{1.1}$$

for all measurable $V \subset \mathcal{X}^d$. In addition to the usual requirements of nonnegativity and integrating to one, it is assumed throughout that $f(\mathbf{x})$ is continuous and bounded. In the discrete case, $\mathbf{x}$ is governed by a probability mass function (PMF) $p(\mathbf{x})$ defined as

$$p(\mathbf{X}) = \mathrm{Prob}\{\mathbf{x} = \mathbf{X}\}, \quad \mathbf{X} \in \mathcal{X}^d. \tag{1.2}$$

The notation $f(x_1, \ldots, x_d)$ will be used interchangeably with $f(\mathbf{x})$; likewise for $p(x_1, \ldots, x_d)$ and $p(\mathbf{x})$. The main use for f and p in the applications will be in providing the conditional, one-dimensional probability laws

$$f(x_d|X_1, \ldots, X_{d-1}) = \frac{f(X_1, X_2 \ldots, x_d)}{f(X_1, \ldots, X_{d-1})}$$

and (1.3)

$$p(x_d|X_1, \ldots, X_{d-1}) = \frac{p(X_1, X_2 \ldots, x_d)}{p(X_1, \ldots, X_{d-1})}.$$

The probability law is to be estimated from a *learning sample* $\mathcal{L}$ of size N (also called the *training data*), where

$$\mathcal{L} = \{\mathbf{X}_n\}_{n=1}^N.$$

The i^{th} element of the n^{th} sample vector is denoted $X_{n,i}$, and an estimate of f or p based on $\mathcal{L}$ is denoted $\tilde{f}$ or $\tilde{p}$.

The quality of an estimate can be measured in a variety of ways. The most commonly used criteria are the L_1 and L_2 norms [2, 10]. A criterion which is relevant in compression and classification applications is the *relative entropy*, defined as

$$D(f||\tilde{f}) = \int_{\mathcal{X}^d} f(\mathbf{X}) \ln \frac{f(\mathbf{X})}{\tilde{f}(\mathbf{X})} d\mathbf{X}$$

and (1.4)

$$D(p||\tilde{p}) = \sum_{\mathbf{X} \in \mathcal{X}^d} p(\mathbf{X}) \log_2 \frac{p(\mathbf{X})}{\tilde{p}(\mathbf{X})}$$

in the continuous and discrete case, respectively. The relative entropy is directly related to efficiency in compression and to error rate in classification [11].

A *partition* $\mathcal{U}$ of $\mathcal{X}^d$ is a collection of M *cells* $\mathcal{U} = \{U_m \subset \mathcal{X}^d\}_{m=1}^M$ such that

$$\bigcup_{m=1}^M U_m = \mathcal{X}^d \quad \text{and} \quad U_m \cap U_{m'} = \emptyset \text{ for } m \neq m'.$$

Given a set of vectors $\mathbf{Y}_1, \ldots, \mathbf{Y}_M$ in $\mathcal{X}^d$, a *nearest-neighbor* partition $\mathcal{U}(\mathbf{Y}_1, \ldots, \mathbf{Y}_M)$ is obtained by including in cell U_m those $\mathbf{X}$ in $\mathcal{X}^d$ which are closer to $\mathbf{Y}_m$ than to every other cell. The Euclidean norm is assumed.

[2]In practice, vectors of image features are generally not independent, so the assumption of independence is usually violated to some degree. However, if the vectors are formed in such a way that the intravector dependence is much stronger than the intervector dependence, then a system that fails to exploit the latter may still perform well.

A *pixel neighborhood* $\mathcal{N}$ is a collection of (row, column) index pairs which specify the locations of conditioning pixels relative to a given *current* pixel location. A neighborhood is *causal* if it includes only pixel locations that precede the current location in raster scan order. In this paper, the vector $\mathbf{x}$ is formed by taking the conditioning pixels as the first $d-1$ elements, and appending the current pixel as the d^{th} element. The neighborhoods used in this paper are shown in Figure 1.

1.2 Histograms and kernel estimates

This section lays the groundwork for the cluster-based estimation technique by briefly reviewing the two most commonly used nonparametric PDF/PMF estimators: histograms and kernel estimates.

For discrete $\mathbf{x}$, the normalized *histogram* $\tilde{p}_H(\mathbf{x})$ of $\mathcal{L}$ is defined as

$$\tilde{p}_H(\mathbf{X}) = N(\mathbf{X})/N, \tag{1.5}$$

where $N(\mathbf{X})$, termed a *bin*, is the number of times that $\mathbf{X}$ appears in $\mathcal{L}$. The histogram is the maximum-likelihood estimator of p, which implies that it is asymptotically unbiased (as $N \to \infty$) and consistent [12]. However, in practice usually $N \ll K^d$, so that asymptotic behavior is not reached. In fact, typically $\tilde{p}_H(\mathbf{X}) = 0$ for all but a small fraction of $\mathcal{X}^d$, even in regions of relatively high probability. Thus, the empty space phenomenon becomes an empty bin problem.

The learning sample is not only well represented by the histogram, but it is too well represented. The histogram *overfits* the learning sample. Yet in overfitting the learning sample, the histogram serves as a relatively compact summary of it. What the histogram in the discrete case lacks is not the ability to *summarize* the learning data, but to *generalize it.* Both properties are indispensable when working in higher dimensions.

Generalizing means inferring probabilities of previously unseen vectors from those in the learning sample. This requires that an assumption be made about how the vectors relate. A smoothness assumption about the probability law is often reasonable: small changes in a vector imply small changes in its probability. (Smoothness of the probability law should not be confused with smoothness of image neighborhoods; the latter is not required for the former.) The smoothness assumption is implicitly used in continuous-valued histograms when adjacent values are grouped into the same bin. As the bin size is increased, the histogram both summarizes and generalizes the data better, but at the cost of decreased resolution.

An alternative means of generalizing the learning sample is *kernel estimation.* A kernel estimate $\tilde{f}_K$ is typically of the form

$$\tilde{f}_K(\mathbf{X}) = \frac{1}{N} \sum_{n=1}^N k(\mathbf{X} - \mathbf{X}_n), \tag{1.6}$$

where the *kernel function* $k(\mathbf{X})$ is itself a PDF that is usually chosen to be spherically symmetric and local to the origin [4]. A Gaussian kernel is often used. The effect of kernel estimation is to "radiate" probability from each vector in the learning sample to the space immediately around it, which is justified by the smoothness assumption. In this way, the learning sample is generalized.

Kernel estimation is a powerful technique in nonparametric statistics with many practical successes reported and a rich supporting theory [13]. However, it is not without its shortcomings. Foremost is its inability to *summarize* the learning sample. In kernel estimation, a kernel is placed at each sample, requiring each training vector to be retained and used whenever the estimate is evaluated. In high dimensional spaces, where large learning samples are necessary, this makes the

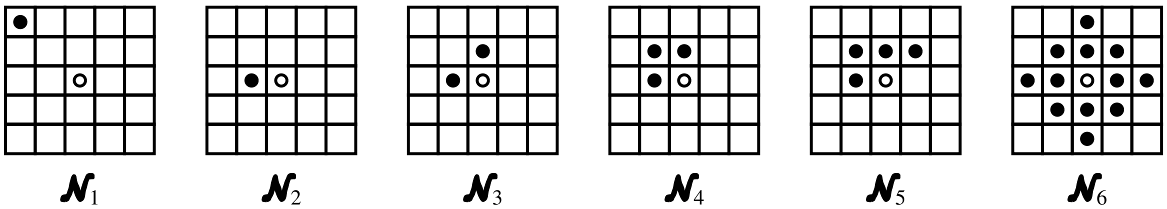


Figure 1: Conditioning neighborhoods used in this paper. In each case, solid circles correspond to $x_1, \ldots, x_{d-1}$, and the open circle to x_d.

kernel method prohibitively expensive in terms of both computation and storage. Attempting to economize by subsampling the training data is tantamount to using a smaller sample, which leads to inaccuracy, most notably in the tails. In higher dimensions, this type of economized kernel estimation becomes problematic, as the tails usually contain most of the total probability (for a nice illustration of this point, see [2], pp. 91–93). Adaptive kernel estimates have been proposed to mitigate this problem [2, pp. 100–110], but they too rely on distribution sampling for the kernel locations, and therefore are prone to poor performance in tail regions.

In the next section, a modification of the kernel method is proposed wherein important regions of $\mathcal{X}^d$ are identified via cluster analysis, then region-specific kernels are fit to these locations. The result is a model that represents both mode and tail regions well, while combining the summarizing strength of histograms with the generalizing strength of kernel estimates.

2 Cluster-based probability estimation

Scott and Thompson [3] have observed, "... the problem of density estimation in higher dimensions involves first of all finding where the action is." Cluster-based kernel probability estimation begins by identifying the locations of important regions of $\mathcal{X}^d$, by means of *cluster analysis*. The details of the cluster analysis are taken up in Section 2.1. Here, we assume that a partition $\mathcal{U}$ consisting of M cells has already been obtained.

To each cell $U_m \subset \mathcal{U}$, fit a single product kernel $\prod_{i=1}^d k_{m,i}(X_i)$, where each $k_{m,i}$ is a one-dimensional PDF. Define $w_m = N(U_m)/N$, where $N(U_m)$ denotes the number of training points that lie in U_m. For continuous $\mathbf{x}$, the cluster-based estimate of the PDF is defined as

$$\tilde{f}_{\mathrm{C}}(\mathbf{X}) = \sum_{m=1}^{M} w_m \prod_{i=1}^{d} k_{m,i}(X_i). \tag{2.1}$$

A natural choice for the kernels is the Gaussian form,

$$k_{m,i}(X_i) = \frac{1}{\gamma\sqrt{2\pi\hat{\sigma}^2_{m,i}}} e^{-(X_i-\hat{\mu}_{m,i})^2/(2\gamma^2\hat{\sigma}^2_{m,i})} \tag{2.2}$$

The parameters $\hat{\mu}_{m,i}$ and $\hat{\sigma}^2_{m,i}$ are taken to be the one-dimensional sample mean $\overline{X}_{m,i}$ and sample variance $S^2_{m,i}$ of the training points that lie in U_m.

In the discrete case, the cluster-based probability estimate is of identical form:

$$\tilde{p}_{\mathrm{C}}(\mathbf{X}) = \sum_{m=1}^{M} w_m \prod_{i=1}^{d} k_{m,i}(X_i), \tag{2.3}$$

where the one-dimensional product kernels $k_{m,i}$ are obtained by sampling the corresponding continuous kernels at the discrete values in $\mathcal{X}$ (e.g., integer grayscale values between 0 and 255), then normalizing to obtain a PMF. All of the experimental results presented in this paper were obtained using Gaussian kernels discretized in this way.

The necessity of the parameter γ in (2.2) is discussed in Appendix B. An appropriate value is determined empirically. In this study, the best values for γ were found to be between 1.0 and 1.5.

2.1 Obtaining the partition: clustering

Though we use the term "clustering," our goal is different from that of traditional cluster analysis. Essentially, what we are after from cluster analysis is much the same as what vector quantization is after: representational efficiency, as opposed to efficiency of discrimination.

A clustering technique that is widely used in vector quantization is the k-means procedure [7], whose origin is often attributed to Forgy [14, 15]. Stagewise application of the k-means procedure, in which the initial guesses for the cluster centroids at each stage are obtained by splitting the centroids resulting from a previous stage, is known as the LBG algorithm [14, 16]. We have adopted the LBG algorithm for all of the experiments here.

The clustering algorithm is now described. Define $\overline{\mathbf{X}}_m = [\overline{X}_{m,1}, \ldots, \overline{X}_{m,d}]$, for $m = 1, \ldots, M$. The following basic procedure is repeated for successively larger values of M, the number of kernels. The parameters θ_1, θ_2, and θ_3 are discussed after the algorithm.

1. Initialization: set $M = 1$, set $\overline{\mathbf{X}}_1$ to be the centroid of all training points, and set D to be the total squared distortion incurred by substituting $\overline{\mathbf{X}}_1$ for each training point. Initialize a partition of $\mathcal{L}$ to consist of a single cell that includes all training points.
2. If the desired M has been reached, or if all cells have population less than θ_3, then stop. Otherwise, increase M by splitting every $\overline{\mathbf{X}}_m$ into two points, one staying in place and the other shifting by a random offset in each dimension, where the offset is uniformly distributed between zero and θ_2 times the magnitude of $\overline{\mathbf{X}}_m$. However, never split $\overline{\mathbf{X}}_m$'s whose cells have populations of less than θ_3.
3. Using $\{\overline{\mathbf{X}}_m\}$, compute a nearest neighbor partition of $\mathcal{L}$.
4. Recompute $\{\overline{\mathbf{X}}_m\}$ using this partition.
5. Recompute D, the total squared distortion incurred by substituting the nearest $\overline{\mathbf{X}}$ for each training vector. If the $\{\overline{\mathbf{X}}_m\}$ resulting from Step 4 reduces D by a factor of more than θ_1 relative to its previous value, return to Step 3. Otherwise, return to Step 2.

Suitable values in all applications described here were determined empirically to be $\theta_1 = 0.01$, $\theta_2 = 0.01$, and $\theta_3 = 10$. In the case of discrete $\mathbf{x}$, a random dither uniformly distributed on $(-\Delta/2, \Delta/2)$ is added to every element of every training point prior to clustering, where Δ is the average spacing between values in the discrete alphabet $\mathcal{X}$ (e.g., $\Delta = 1$ for integer-valued pixel data). Adding this dither allows cell boundaries to migrate more smoothly by preventing multiple points from lying on top of one another.

The number of kernels M can be chosen in one of the following ways. First, if it is determined empirically that choosing M larger than a certain value results in no improvement in performance, then this value can be taken as M. If no such limiting value is found, then M can be chosen on the basis of the available computation resources, or alternatively, it can be chosen to minimize the overall description length of the model and data, i.e., application of the minimum-description length principle[17]. Finally, M can be determined indirectly by the size of the learning sample — eventually, the LBG algorithm will stop creating new clusters on account of low cell populations.

2.2 Optimizing the model parameters via the EM algorithm

An alternative to the method of estimating the weights and kernel parameters described above is to use the expectation-maximization (EM) algorithm [18, 6]. This algorithm results in a local maximum of the model likelihood, which for large training samples approximates a local minimum of the relative entropy $D(p||\tilde{p})$.

The EM algorithm is closely related to the k-means algorithm; in fact it is a "soft" version of it, as will be clear from its description below. However, it is much more computationally expensive than k-means, and is highly sensitive to the initial guess for the parameters being estimated. For these reasons, it is suggested that the EM algorithm be used only to refine the parameter values obtained by the method described previously, rather than to obtain them from scratch.

Preliminary experiments have shown that the performance advantage of optimizing via the EM algorithm can be substantial (about 0.3 bits per pixel improvement in lossless compression of several natural images, using $M = 64$ and $d = 3$). Moreover, use of the EM algorithm obviates the parameter γ in (2.2).

The EM algorithm consists of two steps, the "expectation" step (E-step) and the "maximization" step (M-step). These are iterated until the rate of improvement of the likelihood falls below a specified convergence threshold.

The E-step involves a soft-assignment of the training points to clusters, where the strength of assignment of training point $\mathbf{X}_n$ to cluster m is given by

$$\rho_{n,m} = \frac{w_m \prod_{i=1}^{d} k_{m,i}(X_{n,i})}{\sum_{m'=1}^{M} w_{m'} \prod_{i=1}^{d} k_{m',i}(X_{n,i})}$$

The M-step then updates the values of $\{w_m\}$, $\{\hat{\sigma}^2_{m,i}\}$, and $\{\hat{\mu}_{m,i}\}$, using update rules which can be regarded as weighted versions of the usual maximum-likelihood estimators:

$$\begin{aligned}
\hat{\mu}_{m,i} &= \frac{\sum_{n=1}^{N} \rho_{n,m} X_{n,i}}{\sum_{n=1}^{N} \rho_{n,m}} \\
\hat{\sigma}^2_{m,i} &= \frac{\sum_{n=1}^{N} \rho_{n,m} (X_{n,i} - \hat{\mu}_{m,i})^2}{\sum_{n=1}^{N} \rho_{n,m}} \\
w_m &= \frac{\sum_{n=1}^{N} \rho_{n,m}}{\sum_{m'=1}^{M} \sum_{n=1}^{N} \rho_{n,m'}}
\end{aligned}$$

Note that if $\rho_{n,m}$ is replaced by a hard, nearest-neighbor membership assignment, i.e., if we set $\rho_{n,m} = 1$ if m is the closest cluster to $\mathbf{X}_n$, and $\rho_{n,m} = 0$ otherwise, then the EM algorithm becomes the k-means algorithm.

The main difficulty we have encountered in applying the EM algorithm is its time-complexity for large models. The high complexity comes about because in EM, each training point effectively "belongs" to every cluster, whereas in k-means, each training point belongs to only one cluster. The time-complexity of EM, assuming a direct implementation of the steps given above, is $\mathcal{O}(dMN)$, while that of k-means is only $\mathcal{O}(dN)$. Though some computational savings is possible by eliminating terms that are multiplied by negligible $\rho_{n,m}$'s, the cost is still much higher than that of k-means. It may not be feasible to apply it to the large (e.g., $M = 1024$) cluster-based models considered in this paper. More investigation is required to assess and reduce the computational complexity of EM optimization for very large cluster-based models.

2.3 Componentwise separability

The restriction to separable kernels might seem to be unnecessary and even harmful to accuracy, but the situation is not as clear as it appears. There are at least three good reasons for imposing the restriction.

First, what appears to be greater approximation efficiency in the nonseparable case comes at the price of increased model complexity, since the entire covariance matrix ($d(d+1)/2$ degrees of freedom) must be stored for each cluster, instead of just d variances.

Second, the estimation problem is more difficult when nonseparable kernels are used. The entire covariance matrix, instead of just the dimension variances, must be estimated from the within-cell training data. The quality of the estimates is likely to be more sensitive to low cell populations (which occur frequently in practice), since a greater number of parameters must be computed from the same amount of data. The sensitivity to low populations may be alleviated somewhat by using the EM algorithm to optimize the parameters obtained by k-means, but the time-complexity of EM, which in this case is $\mathcal{O}(d^2MN)$, makes this impractical for large models (e.g., $M = 1024$).

Finally, the use of separable kernels greatly simplifies computation when the estimate is evaluated, as described in the following section.

2.4 Computation

In a huge variety of image processing applications, including all of those considered in this paper, what is needed is the one-dimensional PDF or PMF of a pixel, conditioned on a set of neighborhood pixels. This can be obtained directly from the estimated vector probability law using (1.3). The componentwise separability of the cluster-based estimate simplifies the computation, by allowing the conditional PDF or PMF to be written as a weighted sum of the one-dimensional kernels $k_{m,d}$. In particular, the conditional PDF is

$$\tilde{f}_C(x_d|X_1, \ldots, X_{d-1}) = \sum_{m=1}^{M} r_m k_{m,d}(x_d), \tag{2.4}$$

where the factors $\{r_m\}$ are given by

$$r_m \propto w_m k_{m,1}(X_1) k_{m,2}(X_2) \cdots k_{m,d-1}(X_{d-1}), \tag{2.5}$$

normalized such that $\sum_m r_m = 1$. The discrete case is the same. The normalization of r_m can be distributed over the $d-1$ conditioning dimensions by growing (2.5) as a sequence of partial products and renormalizing after each dimension is multiplied into it. Alternatively, the product can be formed by summing in the logarithm domain, then normalized in two steps: first by shifting the accumulated logarithm to a range that avoids underflow, then exponentiating and renormalizing to sum to one. Both strategies rely on the product structure of r_m, which derives from the componentwise separability of the kernels.

For a given conditioning set $X_1, \ldots, X_{d-1}$, not all of the kernels will contribute significantly to the conditional distribution — i.e., some of the r_m's will be negligible. This makes possible savings in computation by omitting the insignificant kernels. The difficulty lies in knowing which kernels to omit without actually computing them. One method is to weed out insignificant kernels as the product (2.5) is grown, by deleting those r_m for which $k_{m,i}(X_i)$ is smaller than some suitable threshold (determined empirically).

In a previous paper [19] it was suggested that the discretized kernels $k_{m,i}$ be precomputed to further speed execution time. This is appropriate when the need to save execution time far outweighs the need to save memory. However, if the computation of r_m is carried out in the logarithm domain, then the execution-time savings achieved by precomputing $\{\ln k_{m,i}\}$ is negligible, so that the memory advantage of computing-as-needed may take precedence. This is particularly true in applications where multiple cluster-based probability models are to be used.

2.5 Example

We illustrate the cluster-based estimation technique with a simple two-dimensional example. Real applications are considered in Sections 3–5.

A learning sample consisting of $N = 19,700$ vectors was extracted by sliding neighborhood $\mathcal{N}_1$ (see Figure 1) over a 150×150 patch of the aluminum wire texture D1 (see Figure 10). The logarithm of the resulting histogram is shown as a density plot in Figure 2 (a). Notice the speckling throughout, even in the relatively dark (high-probability) areas. This is indicative of the high pointwise variance that arises as a result of the empty-space phenomenon.

Figure 2 (b) and (c) illustrate the cluster-based modeling approach, using $M = 8$. In (b), centroids resulting from the clustering procedure of Section 2.1 are shown with the induced nearest-neighbor partition. Equiprobability ellipses for the corresponding kernels are shown in (c), each marked with its weight w_m.

The structure of the underlying PMF apparent in the histogram has several noteworthy characteristics. The marginal distribution (common to x_1 and x_2) has a strong mode, and there is a great deal of correlation. But the correlation is substantially nonlinear, so that much of it would be left unexploited by linear dimensionality reduction techniques. Also, parametric techniques are unlikely to succeed because of the highly irregular shape. It is a broad-tailed distribution for which we expect kernel estimation to perform poorly when the number of kernels is restricted, as it must always be in practice. This point is made in Figure 2 (d), where a kernel estimate was economized by random subsampling of the learning sample to obtain 128 kernel centers. The tail regions are poorly represented, as expected.

Figures 2 (e)–(h) show the cluster-based estimates for several values of M. The tails are well-represented, and the estimate does not suffer from the speckling that plagues the histogram. In other words, the cluster-based probability model both summarizes and generalizes the training data.

2.6 Asymptotic properties

It is clear from Figure 2 that as M increases, the potential for the cluster-based model to approximate an arbitrary continuous probability law improves. This improvement comes at the cost of increased model complexity. Nevertheless, it would be satisfying to know that the approximation can be made as close as desired by choosing a sufficiently large M. In this section we establish that as M gets indefinitely large, the cluster-based model converges in probability ($N \to \infty$) to the true probability law, in both the discrete and continuous cases.

Let the diameter of the smallest d-sphere that covers cell U be denoted $\mathrm{Diam}(U)$. For a given partition $\mathcal{U}$, define $\mathrm{Diam}(\mathcal{U})$ as

$$\mathrm{Diam}(\mathcal{U}) = \sup_{U \in \mathcal{U}} \mathrm{Diam}(U). \tag{2.6}$$

Let $\{\mathcal{U}_j\}_{j=1}^{\infty}$ be a sequence of partitions such that, for every $\epsilon > 0$, there exists a j' such that $\mathrm{Diam}(\mathcal{U}_j) < \epsilon$ whenever $j > j'$.

Associated with each partition $\mathcal{U}_j$ is a cluster-based probability model $\tilde{f}_{C,j}$.

Let $V \subset \mathcal{X}^d$ be a measurable set, and define the *ϵ-skin* S_ϵ as

$$S_\epsilon = \{\mathbf{X} : ||\mathbf{X} - \mathbf{X}_B|| < \epsilon \quad \forall \mathbf{X}_B \in B\}, \tag{2.7}$$

where B is the boundary of V. It is assumed that B is smooth in the sense that

$$\mathrm{Volume}\,\{S_\epsilon\} \to 0 \qquad \text{as} \qquad \epsilon \to 0. \tag{2.8}$$

Let $U_{j,B}$ be the union of those cells in $\mathcal{U}_j$ that intersect B. Also, let $U_{j,I}$ be the union of the cells in $\mathcal{U}_j$ that lie completely within V. Thus, $U_{j,B} \cap U_{j,I} = \emptyset,\ \forall j$.

Proposition 1. For every $\delta > 0$, there exists a j' for which

$$\lim_{N \to \infty} \mathrm{Prob}\left\{ \left| \int_V \left[f(\mathbf{X}) - \tilde{f}_{C,j}(\mathbf{X}) \right] d\mathbf{X} \right| > \delta \right\} = 0$$

whenever $j > j'$.

A proof is given in Appendix A.

The smoothness restriction (2.8) on the boundary of V is not burdensome, since the sets that are of interest in most applications have boundaries that are decided by some sort of distance-based criterion, and such boundaries are smooth.

A remaining complication is that the LBG algorithm does not, in general, lead to sequences of partitions with the needed property of decreasing maximum cell diameter. The problem is that regions of zero probability are never populated in the learning sample; hence these regions can never be split. Nevertheless, it is clear that any region of *nonzero* probability will always be populated in a sufficiently large learning sample, and will therefore eventually be split. Hence, when the LBG algorithm is used to obtain the sequence of partitions, the asymptotic analysis holds on all regions of $\mathcal{X}^d$ that have nonzero probability.

We now consider asymptotic behavior in the discrete case. We argue that as $\mathrm{Diam}(\mathcal{U}_j) \to 0$, the discrete cluster-based probability model degenerates into the histogram, and therefore inherits all of its asymptotic properties. First note that as $N \to \infty$, every element of $\mathcal{X}^d$ that has nonzero probability will be represented. Now suppose that, for a given j, one or more of the cells in $\mathcal{U}_j$ contains more than a single point in $\mathcal{X}^d$. Then at least one of these must be split in a finer partition, so that in the limiting case, every cell contains exactly one point $\mathbf{X} \in \mathcal{X}^d$. This implies that each cell has zero variance, so that, informally, the kernels are "impulses" of height $N(\mathbf{X})/N$, which is precisely the definition of the histogram. As mentioned in Section 1.2, the histogram is the maximum-likelihood estimate, so that it is asymptotically unbiased and consistent. Hence, the discrete cluster-based probability model converges to the true probability law as required.

It is interesting to contrast the manner in which the histogram, the kernel estimate, and the cluster-based probability estimate construct the probability surface in high-probability regions. The kernel estimate requires that a large number of learning examples be situated in a small region, so that the kernel function can effectively blur them into a single mode. This is wasteful of storage in the sense that many different microscopic configurations lead to the same blurred macroscopic mode, yet the specific configuration must be stored exactly. In

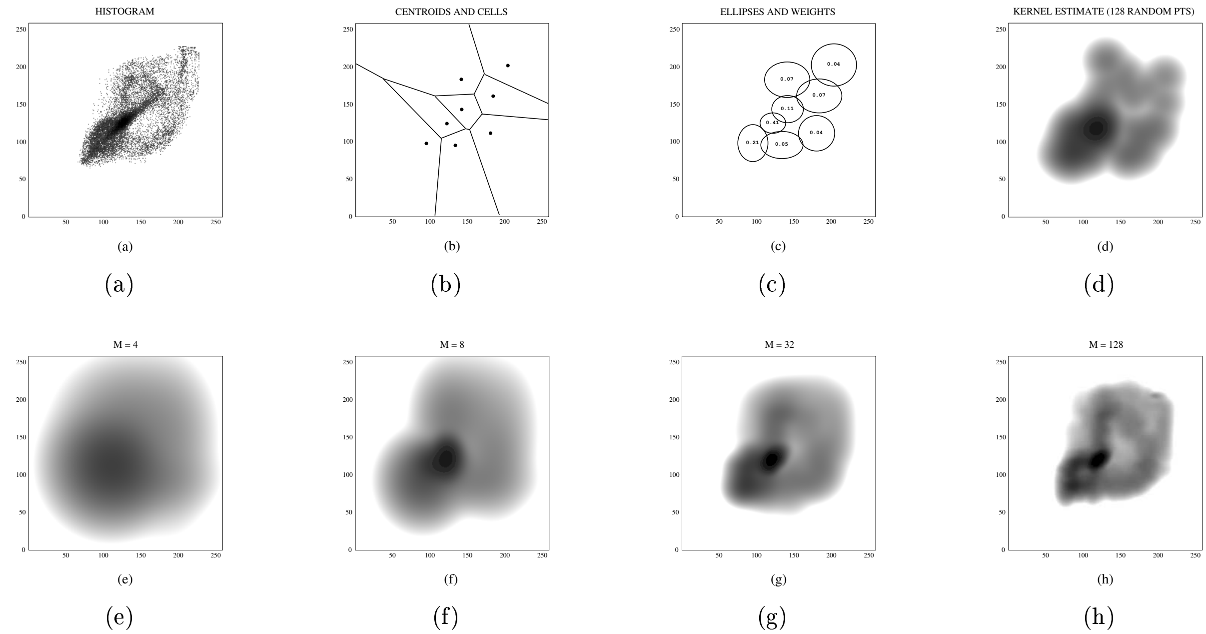


Figure 2: (a) The logarithm of the histogram is shown as a density plot. (b) Centroids and partition, and (c) kernel ellipses, for the cluster-based probability model with $M = 8$. (d) Log density plot of a Gaussian kernel estimate ($\sigma^2 = 30$) using 128 points randomly selected from the training set as the kernel locations. Note that the kernel approach does not represent the tail regions well. Log density plots (e)–(h) correspond to cluster-based probability models with increasing M ($\gamma = 2.0$).

its favor, the kernel estimate has the advantage of data-driven adaptation to local variations in density of the learning sample. At the other extreme is the histogram, which represents the probability surface as a single number for every cell in a partition. The precise configuration of the learning examples within each cell is forgotten, giving the technique the potential for storage efficiency. The disadvantage is that constant probability is assigned over entire cells, even when the training vectors happen to lie in some limited (possibly remote) part of a cell. The cluster-based probability model combines desirable properties of both approaches: it uses a single kernel which may be centered anywhere in the cell, and uses cell population instead of the precise configuration of training points to establish the height of the probability surface.

The remainder of this paper considers some applications of the cluster-based probability estimate. Since all of these applications involve working with discrete pixels intensities, the PMF formulation is used rather than the PDF.

3 Image restoration

Suppose an image has been degraded in an unknown way, or else in a way that is so difficult to describe mathematically that direct inversion of the degradation process is infeasible. We wish to recover an estimate of the original from this degraded version — how can this be accomplished?

Regardless of the difficulty in describing the degradation, the technique illustrated in Figure 3 can be used. First, a learning sample is formed by extracting a vector for every pixel location, using a neighborhood which need not be causal. Only one training pair is shown in the figure, but it should be understood that a large number of such pairs make up the training set, and that the test image to be restored is excluded from that set. After the learning sample has been obtained, a cluster-based PMF is trained on it. The choice of M can be made in one of the ways described at the end of Section 2.1; in our experiments, we obtained acceptable results using values of M ranging from 128 to 2048.

After the cluster-based estimate has been obtained, it can be used to restore a previously unseen degraded image, in the following way. For a given pixel location, let $X_1, \ldots, X_{d-1}$ denote the values of the degraded neighborhood pixels, and let x_d be the unknown original pixel value. The value of x_d is then chosen which best explains the occurrence of $X_1, \ldots, X_{d-1}$ with respect to some appropriate criterion. Common criteria are maximum-likelihood (ML), maximum *a posteriori* probability (MAP), and least expected squared error (LSE):

$$\begin{aligned} \text{ML:} \quad X_d &= \arg\max_{x_d} \tilde{p}_C(X_1, \ldots, X_{d-1} | x_d) \\ \text{MAP:} \quad X_d &= \arg\max_{x_d} \tilde{p}_C(x_d | X_1, \ldots, X_{d-1}) \\ \text{LSE:} \quad X_d &= E_{\tilde{p}_C}(x_d | X_1, \ldots, X_{d-1}) \end{aligned}$$

where the right-hand side in the LSE formula is the conditional expectation with respect to the model. All of these criteria can be used with the cluster-based probability model.

3.1 Restoration experiments

Figure 4 shows the result of cluster-based MAP restoration in an example where a 128×128 8-bit image (a) is degraded by additive white Gaussian noise with a variance of 100 (b). The result of nonadaptive, separable eleven-tap Wiener filtering is shown in (c).[3] For the cluster-based probability model, the square 3×3 neighborhood ($d = 9 + 1 = 10$) of Figure 3 was used, with $M = 1024$. The training set for this example, shown in Figure 9, consisted of twenty-five natural images together with their degraded versions. The restored image (d) exhibits significant noise reduction while maintaining reasonable sharpness. The improvement in signal-to-noise ratio is 3.01 dB.

To get an idea of the effect of the cluster-based MAP technique on objective image quality, the experiment was repeated

[3]The filter length was chosen to give the best result perceptually.

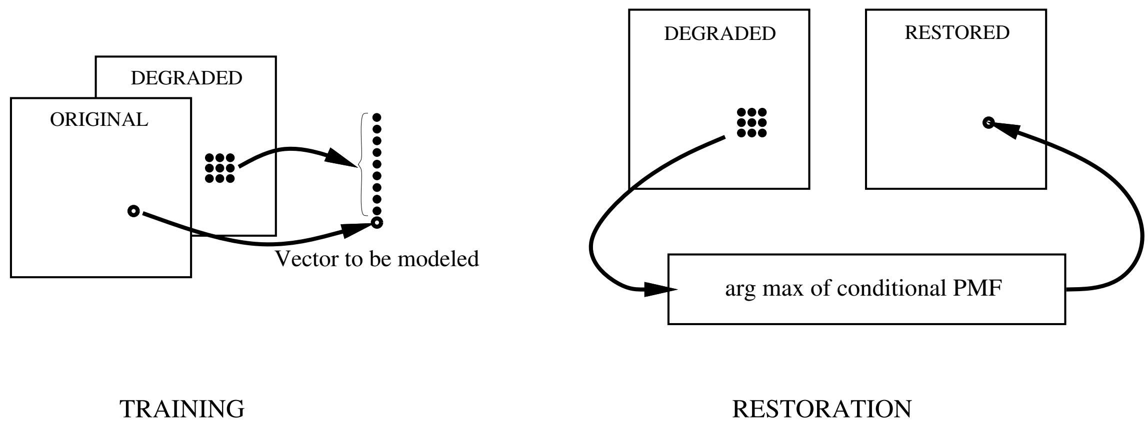
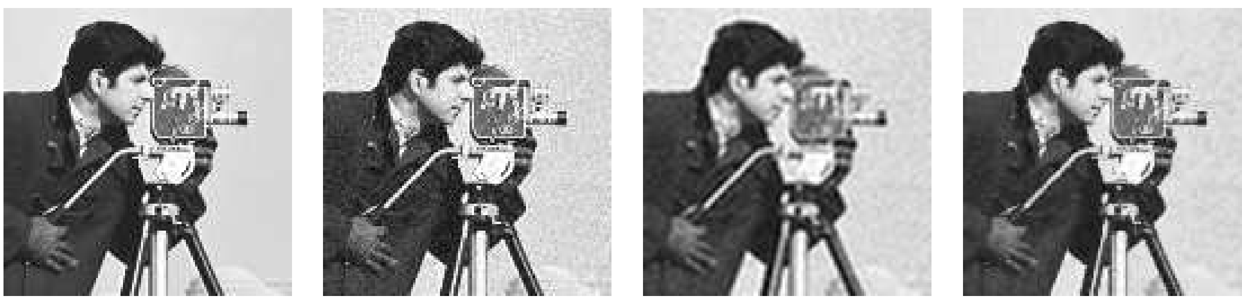


Figure 3: In the training phase (left), training data vectors $\mathbf{x}$ are formed by appending an original pixel to the corresponding degraded neighborhood pixels. In the restoration phase (right), the final component of the vector is unknown; its value is estimated by maximizing its PMF conditioned on the degraded pixels.



(a) (b) (c) (d)

Figure 4: Image restoration example (see text). (a) original image; (b) degraded; (c) Wiener filtered (d) restored using cluster-based model.

by alternately using each of the training images as the test image. For every test image, the cluster-based estimate was retrained using a training set that excluded that test image. This prevented overtraining, which would have lead to unrealistically optimistic performance estimates. In each case, there was an improvement in root-mean-square noise level ranging between 3 and 5 dB. Similar improvement resulted when the conditional expectation (LSE) was used instead of the MAP estimate. The ML estimate is usually used when no prior distribution is available for the quantity being estimated, i.e., x_d is treated as a deterministic but unknown parameter. Since a prior is available for x_d in the cluster-based probability model, the ML estimate was not implemented.

Because cluster-based restoration requires only weak assumptions about the statistics of the degradation (stationarity and locality of spatial dependence), it is flexible and can accommodate high-order, nonlinear statistical interactions that might be present in the degradations. For this reason, it is expected to perform well in restoration problems where the degradation process is spatially local but highly nonlinear and/or difficult to express mathematically. Examples of restoration problems that fit this description are dehalftoning, film grain reduction, and compensating the effects of quantization in lossy compression schemes.

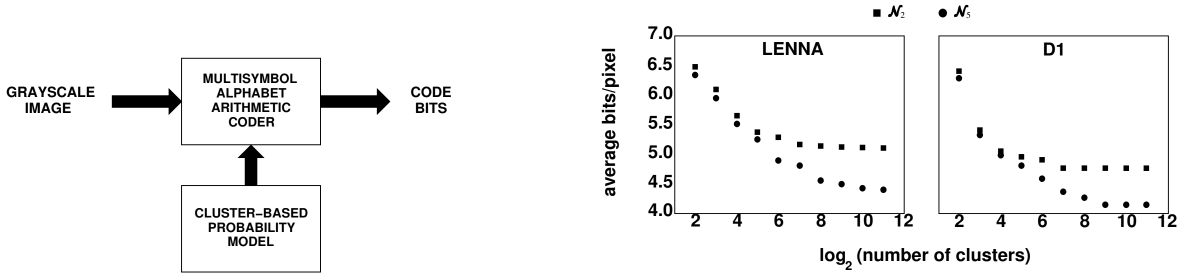
3.2 Relationship to nonlinear interpolation

Cluster-based restoration is similar in spirit to a VQ-based nonlinear interpolation technique proposed by Gersho [20]. Both approaches have the potential to learn nonlinear statistical relationships from training data and to use those relationships to fill in missing values. However, the techniques differ in one important respect. In the cluster-based technique, several kernels interact to determine the restored value, instead of the value being determined by a single codebook entry. Thus, restored values are not limited to only those appearing explicitly in a codebook; instead, they are *synthesized* from the model and the available conditioning information.

4 Lossless compression

Most of the published research in image compression deals with lossy compression: the image that is reconstructed from the compressed representation approximates the original; it is not required to be identical to it. Such techniques routinely achieve compression ratios of 10:1 or more, with little or no noticeable distortion.

On the other hand, lossless (strictly reversible) compression techniques typically yield compression ratios no better than 2:1 on natural images [21, 22]. This comparatively poor performance is a consequence of the requirement that the reconstructed image be bit-for-bit identical to the original. An upper bound on the compression ratio that can be achieved comes from noise that is inevitably introduced in the acquisition process. However, in practice it is not just noise, but also any *noiselike* phenomena that contributes inordinately to bit rate. Often, the regions which appear noiselike are textured regions. In this section we consider losslessly compressing both natural scenes (which include textured regions) and



(a) (b)

Figure 5: (a) Performance of lossless compression system depends on probabilistic model. (b) Dependence of bit rate on M for lossless compression of two images: *lenna* and *D1.*

pure textures.

Despite their low compression ratios, lossless techniques are required in certain applications. For example, in image and video processing research, the original reference images and video sequences must be intact if the research conclusions are to be valid. Some image data, such as satellite imagery, may have been obtained at great expense, and the risk of losing expensive information which might later be required for some unforeseen purpose precludes any archiving scheme based on lossy compression. Another often-cited application for lossless coding is the archiving of digital radiology and other medical images, where ethical and legal considerations make the use of lossy techniques questionable.

But the main reason for considering lossless compression here is that lossless compression is ultimately a modeling problem. Shannon established that an optimal lossless compression system uses $-\log_2 p(X)$ bits to encode the message element X, where $p(X)$ is the probability of X. (For economy of notation, the conditioning is not explicitly mentioned, but it should be understood that p is conditioned on whatever is both relevant and available at both the encoder and decoder). The optimal average bit rate is thus

$$H(x) = -\sum_{X \in \mathcal{X}} p(X) \log_2 p(X),$$

which is the entropy of x. If instead of the true probability, an estimate $\tilde{p}$ is used to construct a code, then the average bit rate $R_{\tilde{p}}$ is

$$\begin{aligned} R_{\tilde{p}} &= -\sum_{X \in \mathcal{X}^d} p(X) \log_2 \tilde{p}(X) \\ &= H(x) + D(p||\tilde{p}), \end{aligned} \tag{4.1}$$

where $D(p||\tilde{p})$ is the relative entropy defined by expression (1.4). It is easy to verify that $D(p||\tilde{p}) \geq 0$, with equality if and only if $\tilde{p}(X) = p(X)$ for all X. Thus, the goal of designing a lossless compression is exactly the same as that of estimating p.

The above assumes an optimal code. It is natural to question whether codes that are achievable in practice also perform best when $\tilde{p} = p$. As will be discussed in the following section, arithmetic coding is a practical technique for lossless compression that very nearly achieves Shannon's optimal code length assignment [23], so that the above analysis pertains to practice as well as theory.

4.1 Lossless compression with arithmetic coding

Arithmetic coding is a form of entropy coding that offers significant advantages over other methods in many applications [23, 24, 25, 26]. It has near-optimal efficiency (relative to the assumed probability law) for a broad class of sources and over a wide range of coding rates. It is also inherently adaptive, and simplifies the encoding of large-alphabet low-entropy sources. Most importantly, it allows the probabilistic model to be specified explicitly and separately from the actual encoder.

TABLE I: Estimated Bits per Pixel ($M = 2048$)

Image	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$	$\mathcal{N}_5$
cman	5.50	4.81	4.77	4.79
lenna	5.11	4.53	4.38	4.41
D1	4.83	4.40	4.24	4.14
D77	6.08	6.04	5.46	5.30

The first use of arithmetic coding as an image compression technique was by Langdon and Rissanen [27]. In their system, which was for binary images, each pixel was encoded using a PMF conditioned on a nearby set of previously encoded pixels, i.e., on a causal neighborhood. Since the input was binary, the number of conditioning states remained manageable even for large neighborhoods, so that a histogram PMF estimate was feasible. For example, ten conditioning pixels means only 1,024 conditioning states.

Extending the Langdon-Rissanen scheme to handle grayscale images is greatly complicated by the empty space phenomenon. For example, in the case of eight-bit pixels, a ten-pixel conditioning neighborhood implies 2^{80} conditioning states, most of which will never be observed in a reasonably sized training sample. Moreover, in the grayscale case, we have good reason to believe that the underlying probability law is smooth, but the histogram PMF estimate makes no use of this prior knowledge.

We substitute the cluster-based probability estimate for the histogram estimate in the Langdon-Rissanen scheme, thereby extending it to handle grayscale images. The system is shown in Figure 5 (a). Pixels are arithmetically encoded in raster order. As in the Langdon-Rissanen scheme, the PMF used for each pixel is conditioned on a set of previously-encoded pixels, so that the decoder has access to the same conditioning information that the encoder had.[4]

4.2 Compression experiments

Using neighborhoods $\mathcal{N}_2$–$\mathcal{N}_5$ shown in Figure 1, cluster-based probability models were trained on the set of 25 natural images shown in Figure 9. These PMF estimates were then used

[4]At the top and left boundaries, unavailable conditioning pixels are arbitrarily set to 128; the resulting local inefficiency has little effect on the overall bit rate.

in an arithmetic coding system to compress two natural images not in the training set: *cman* and *lenna*. The resulting estimated bit rates for $M = 2048$ are shown in the first two rows of Table 4. The rates are based on the assumption that 16-bit arithmetic is used in the encoder and decoder. Next, the experiment was repeated using two natural textures: *D1* (aluminum wire mesh) and *D77* (cotton canvas), but using a 352×352 portion of each texture for training and a disjoint 128×128 portion for testing. The textures are shown in Figure 10.

The performance listed in the first two rows of the table compares favorably with that reported in the literature for natural scenes [21, 22]. The compression performance for the two textures is more difficult to interpret, since no previous results seem to have been published. One could argue that the textures, having fewer blank regions, are more difficult to compress than natural scenes. But this difficulty is offset by their relative homogeneity, which should allow a single model to work well over the entire texture.

The dependence of bit rate on M (the number of kernels) is shown graphically for the *lenna* and *D1* test images in Figure 5 (b), for each of the neighborhoods $\mathcal{N}_2$ and $\mathcal{N}_5$. For both test images, the $\mathcal{N}_2$ curve reaches a limit at about $M = 256$, while $\mathcal{N}_5$ curve continues to improve as M increases to 2048.

4.3 Comparing PMF estimates

How good is an estimate $\tilde{p}$ in terms of relative entropy? It is difficult to estimate $D(p||\tilde{p})$ in (4.1) directly from a sample of limited size, since the true p remains at all times unknown. However, if $\tilde{p}_1$ and $\tilde{p}_2$ are two competing estimates of p, then their *difference* in relative entropy

$$D(p||\tilde{p}_1) - D(p||\tilde{p}_2) = R_{\tilde{p}_1} - R_{\tilde{p}_2}$$

is easily and reliably estimated from a moderate-size sample by subtracting the sample average of $-\log \tilde{p}_2(x)$ from that of $-\log \tilde{p}_1(x)$. These sample averages are just the average bit rates produced by arithmetic coders based on $\tilde{p}_1(x)$ and $\tilde{p}_2(x)$. The comparison can be carried out for more than two estimates in a similar way, since the unknown entropy term is common to all of them. Thus, practical lossless compression serves as a fair test of the relative accuracy or predictive power of a model with respect to the relative entropy measure; it is a level playing field upon which competing models can battle. Whichever model produces the fewest bits, wins. The test is decisive even when the sample sizes are small. As will be seen in the next section, this type of competition can be used as the basis for classification.

5 Texture classification

Classification is an activity humans carry out constantly, to make sense of the world we perceive around us. We recognize similarities and differences among sensory stimuli, and group objects accordingly. The specific similarity metrics we employ seem extraordinarily complex, but one thing about them is certain: they are usually high-dimensional, and involve the integration of diverse elements of knowledge, some high-level and some primitive, some conscious and some at the level of intuition.

Much simpler is the situation in which a machine is to classify objects on the basis of objectively measured features and with respect to some clearly stated criterion (possibly Bayesian). We will see an example shortly involving textures where the result of such classification matches closely what seems subjectively reasonable.

For simplicity and concreteness, we consider the case in which the prior probabilities of the classes are equal, and where the goal is to minimize the overall probability of classification error. In this case, the Bayes decision rule reduces to the familiar ML rule, which is to choose the class which makes the observed data most likely [7].

ML classification can be applied in many different ways. For instance, if it were known beforehand that a particular patch in an image consists of a single texture class, then the model with the greatest likelihood could be chosen.

More typical is the situation where we are not assured that the entire patch came from a single class; indeed, the task is frequently to determine the boundaries between classes in an image that is assumed to be heterogeneous (e.g., image segmentation). In this case, the classification decision should be more or less independent for each pixel. This can be accomplished by centering a neighborhood (not necessarily causal) at each pixel location, and choosing for that pixel the class whose modeled conditional probability is greatest. The resulting classification will of course appear "noisy," since each decision is based on only one neighborhood observation. An alternative is to assume *limited* spatial homogeneity, and to choose a class that maximizes "local" likelihood, appropriately defined. This idea will be made more precise shortly.

Another way around the single-observed-neighborhood problem is to use several different models for each class, each conditioned on a different neighborhood around the current pixel. It is natural to choose these neighborhoods to be at different scales, resulting in a multiresolution system. The following heuristic can then be invoked: *To be assigned to a certain class, a pixel must have high conditional likelihood simultaneously with respect to several different neighborhoods.* This idea has recently been generalized as an independent method of attack on the curse of dimensionality in density estimation [28].

Consider now the problem of classifying regions in heterogeneous images. Suppose that there are J classes, and that vectors drawn from class j follow PMF $p_j(\mathbf{x})$. Associated with each class is a cluster-based probability model $\tilde{p}_j(\mathbf{x})$, trained previously. Let $\mathcal{S} = \{\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N\}$ be a sample drawn from an unknown class which we wish to identify. If we assume for now that the vectors in $\mathcal{S}$ are independent, the likelihood function for class j is simply

$$p_j(\mathbf{X}_1)p_j(\mathbf{X}_2)\cdots p_j(\mathbf{X}_N). \tag{5.1}$$

After taking the logarithm, we obtain the decision rule:

$$\text{Choose } j \text{ for which } \sum_{n=1}^{N} \log p_j(\mathbf{X}_n) \text{ is maximized.} \tag{5.2}$$

To evaluate each $\log p_j(\mathbf{X}_n)$, the technique described in Section 2.4 can be used.

Typically, the vectors are formed at each pixel location using a neighborhood like $\mathcal{N}_6$. Formed in this way, vectors corresponding to adjacent pixels are not independent, so that (5.1) is not strictly justified. The resulting decision rule does not take advantage of the statistical dependence among vectors. However, one can argue that if the dependence among vectors is similar for all classes, ignoring it for all classes results in a useful decision rule.

It is worthwhile noting that the summation in (5.2), after dividing by N, equals the average bit rate of an ideal entropy coder fed by model $\tilde{p}_j$. Consequently, the decision rule can in principle be implemented using the structure shown in Figure 7. Of course in practice, real entropy coders need not be used, since only the rates and not the actual code bits are needed.

The possibility of realizing the decision rule in this way illustrates a connection between data compression and classification. This connection seems especially important with the growth of large libraries of image data, where one will want to search and make decisions on compressed data [29]. Recent

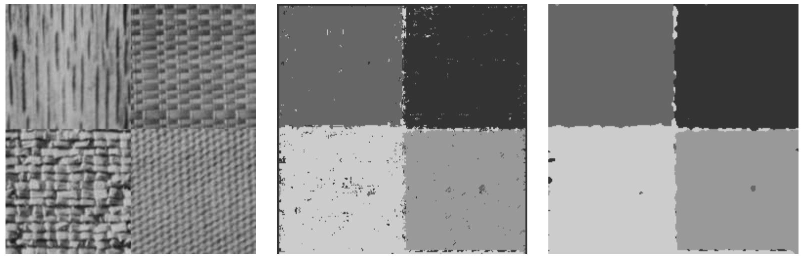


Figure 6: Four-class example of texture classification using the cluster-based model. From left to right, the images are: the original composite test image, classification using resolution averaging only, and classification using both resolution and spatial averaging.

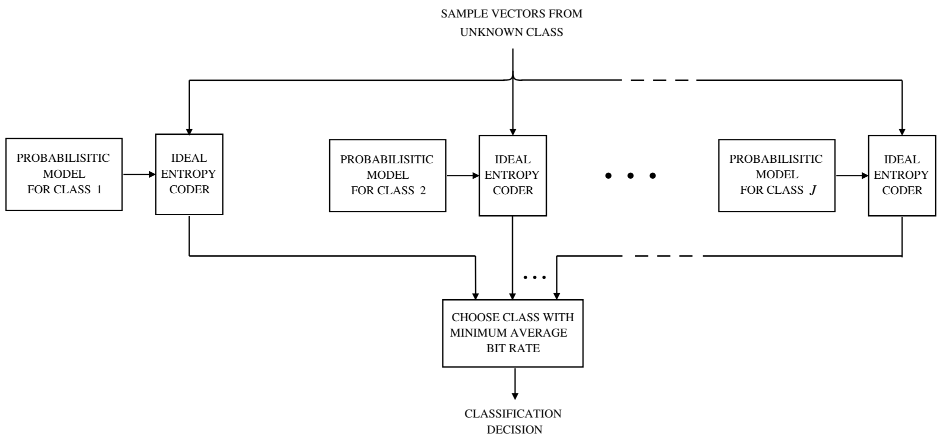


Figure 7: A connection between data compression and classification: the minimum-error-probability rule can be realized using a bank of ideal entropy coders, each tuned to a different source.

work by Perlmutter *et al.* indicates that combining these two tasks can result in improved performance for both [30].

The above assumes that nearby pixels come from the same class. In practice, when working with heterogeneous images, it is desirable to impose this assumption in a soft manner. How can this be accomplished? Thinking of the summation in (5.2) as a local averaging operation is suggestive of spatial lowpass filtering. In particular, for each class, the logarithm of the likelihood is obtained for each pixel location, and the resulting "image" of log-likelihoods can be spatially lowpass filtered. The decision rule then assigns to each pixel location the class with the maximum smoothed log-likelihood function at that location.

As mentioned earlier, it is also possible to average *across several models,* each working at a different spatial resolution. While spatial averaging reflects the assumption that nearby pixels are likely to have come from the same class, resolution averaging imposes the requirement that a candidate match the sample texture at several scales simultaneously. This is because averaging logarithms corresponds to taking a product, which will be small if any of the factors is small. Spatial and multiresolution averaging are not mutually exclusive; in fact, our best results employ both.

5.1 Classification example

The cluster-based classification scheme was applied to the composite test image shown on the left in Figure 6. The image consists of four Brodatz textures, they are (clockwise from top left) D68, D55, D77, and D84. For each class, three cluster-based probability models were trained on data that did not include the test data. Models were obtained at three different resolution scales: one model was trained using neighborhood $\mathcal{N}_6$, another using the same neighborhood but with the pixel location offsets scaled by a factor of 2 around the center, and a third with the offsets scaled by a factor of 4. Since $\mathcal{N}_6$ contains 12 conditioning pixels, the total dimensionality for each of these models is 13.

To perform the classification, three vectors were formed for every pixel in the test image, one for each of the three resolutions. For each class and each resolution, the logarithm of the conditional probability of the center pixel was computed using the corresponding model. These values were then averaged across resolutions. The classification results are shown in the middle image of Figure 6. The total classification error rate is less than five percent. The test was repeated using spatial averaging of the logarithms in addition to the resolution averaging; the results are shown on the right in Figure 6. A 7-tap separable lowpass filter was used to perform the spatial averaging. The overall classification error rate in this case is below one percent.

6 Discussion

The preceding sections described the cluster-based probability modeling technique, and considered some applications. This section follows up on some of the issues raised in previous sections, and suggests topics for further study.

6.1 Preprocessing vs. modeling

In a complex signal processing system, some preprocessing is usually carried out on the input to make the subsequent probabilistic modeling and processing tasks easier. Image compression systems, for example, often employ an invertible, energycompacting transformation as a first step, resulting in a signal that is easier than the original to quantize and encode efficiently.

The problem of improving preprocessing operations has received much attention among researchers in various disciplines over the past several decades. The philosophical direction of such research effort has been towards *more sophisticated preprocessing, enabling less sophisticated probabilistic models to be used.*

Of interest here is the complementary research direction: toward *more sophisticated models, enabling less sophisticated preprocessing to be used.* Examples of techniques in this direction are vector quantization (VQ) in the areas of compression and interpolation [14], and artificial neural networks in the areas of regression and classification [31]. These techniques reduce the effect of preprocessing on system performance, by exploiting nonlinear, higher order statistical relationships that exist among the signal elements. A striking aspect of these approaches is that they can function largely as "black boxes" — one need not understand the information source in order to process it. This is either good or bad, depending on one's objectives. The approach advocated here shares the spirit of these approaches by capturing whatever statistical relationships exist. It differs from them in that these relationships are made available as an explicit probability law, which may then be used for a variety of purposes. Moreover, having the probability law allows us to build a conceptual bridge between these "black-box" approaches and classical approaches.

The strategy taken in the experimental parts of this paper has been to work directly in the untransformed observation space of pixel neighborhoods. This choice draws attention to the fact that more powerful modeling makes the initial transformation or feature selection step less critical. Were maximum performance the main goal, then substantial effort would be justified in devising suitable transformations for use in tandem with the modeling technique described here.

6.2 Relationship to artificial neural networks and radial basis functions

It was commented in Section 1 that the proposed technique differs in philosophy from traditional mixture modeling in that the goal is to approximate a probability law, not to decompose it into physically significant components. Thus, we can view the modeling problem as one of function approximation, where the function to be approximated is the PDF or PMF. In particular, if the $k_{m,i}$'s are chosen appropriately, then (2.3) amounts to a radial basis function (RBF) approximation to $p(\mathbf{x})$. One might hope, therefore, the RBF literature would provide insight into such issues as training, means of implementation, and bounds on approximation accuracy [32, 33, 34]. This is true to some extent, but two differences are apparent. The first is that values of the function being approximated (a probability law) are never actually observed; instead our observations consist only of samples that we believe to be governed by the function. The second difference is in the relevant approximation criterion, which in the applications of interest here is the relative entropy (1.4). This criterion is not a distance metric, since it is asymmetric and does not satisfy the triangle inequality. Consequently, much of the RBF approximation theory does not apply directly.

The relationship to RBF's suggests a connection to artificial neural networks (ANN's). As mentioned in Section 6.1, a system using the cluster-based probability model does have certain elements in common with an ANN. Both are capable of learning complex, nonlinear relationships from training data, and exploiting them to perform various information processing tasks. We expect that the applications we are considering could be handled by an appropriate type of ANN with a comparable level of performance. However, there is nothing inherently "neural" about the cluster-based probability model, and it is not connected to any specific class of hardware topology.

If we compare the cluster-based model with a classical ANN like a multilayer perceptron, another distinction emerges. The set of weights in the perceptron, which completely determines its function, has no obvious interpretation outside the network. For instance, the set of weights cannot be used by some other perceptron to perform a different task. In contrast, the method of this paper provides an explicit probabilistic model for the source, which can be used equally well in a variety of applications like compression, restoration, and classification. In principle, this distinction vanishes when the goal of the ANN is specifically to estimate the PMF[35], rather than to carry out the ultimate information processing task. In this case, the approaches may be accomplishing the same thing in different ways.

Viewing the cluster-based model as an artificial neural network might prove to be useful when it is desired to adapt the kernels as data are being processed, as opposed to the current technique in which the kernels are fixed after an initial training phase. Also, techniques used in pruning insignificant nodes in ANN's might prove useful in eliminating insignificant kernels during computation. To understand the connection to neural networks more fully requires study beyond the scope of this paper.

6.3 Hierarchies of cluster-based models

When cluster-based models are defined directly on pixel neighborhoods, a large number of the clusters are inevitably allocated along the main diagonal of the probability space. This is a consequence of the high positive correlation of spatially adjacent pixels in natural images and textures. This is one reason for working with some other features besides pixels, where the high correlation is absent. However, this type of commonality in the kernel distributions among models is suggestive of a plan for organizing a large collection of models in such a way that they can easily share common attributes when appropriate. In particular, a tree structure can be used. Kernels that are common to all of the models can be stored at the root of the tree. The leaves of the tree would correspond to the individual models, and the path from the root to each leaf would specify which kernels would have to be added to result in the corresponding model. Such hierarchies of models can be expected to play a substantial role when the cluster-based technique is applied to large, real-world classification problems, as occur in digital libraries.

An interesting alternative means of sharing common attributes among several models has been suggested recently by Pudil *et al.* [36], in a classification setting. The approach is to posit a common "background" density for all of the classes, and to express each class-conditional density as a mixture of products of this background density with a class-specific modulating function. Expressing the density in this way simplifies the sharing of common attributes, and provides a basis for feature selection: choose the features that provide maximum deviation from the background. The classification results reported in [36] are for small mixtures (2, 3, and 4 components). In applications when the goal is to characterize the precise shape of the density, not just that portion which provides discriminatory power, much larger mixtures may be necessary.

7 Conclusions

A multidimensional probability model, based on cluster analysis, has been presented, analyzed, and applied to certain problems in image and texture processing. The model combines the summarizing ability of a histogram with the generalizing ability of kernel estimation, while avoiding some of the drawbacks of each. In particular, it avoids the empty-bin problem associated with high-dimensional histograms, and it performs better in tail regions than a traditional kernel estimate of the same complexity.

It was shown that under reasonable conditions, the estimate converges asymptotically to the true probability law as its complexity is allowed to increase arbitrarily. In practice, the model was shown to be successful in applications requiring the estimation of a joint PMF with up to 13 variables.

Applied to image restoration, the model was used to learn complex degradations that cannot be expressed easily in mathematical form. Lossless compression provided a fair test of the accuracy of the model; for natural images the results were competitive with methods designed specifically for lossless compression. In texture classification, several cluster-based models were used effectively in a standard Bayesian framework. Performance was also shown to improve when within-class averaging of log-likelihoods was carried out both spatially and across scales.

By capturing the high-order, nonlinear relationships that exist among features, and by providing an explicit estimate of the governing probability law, the model is able to extend the performance of otherwise traditional systems that rely on probabilistic models. The practical value of the cluster-based probability model was demonstrated by its effectiveness in the following applications: image restoration, lossless image and texture compression, and texture classification.

Acknowledgments

Thanks to the anonymous reviewers for their helpful suggestions, and to Michael I. Jordan for bringing to our attention the relationship between the k-means and EM algorithms.

A Proof of Proposition 1

We consider the case where the kernels have finite support. The infinite-support case is more involved and is not considered here.

Since $U_{j,I} \subseteq V \subseteq (U_{j,I} \cup U_{j,B})$,

$$\begin{aligned}\int_{U_{j,I}} f(\mathbf{X})d\mathbf{X} &\leq \int_V f(\mathbf{X})d\mathbf{X} \\ &\leq \int_{U_{j,I}} f(\mathbf{X})d\mathbf{X} + \int_{U_{j,B}} f(\mathbf{X})d\mathbf{X}\end{aligned} \tag{A.1}$$

and

$$\begin{aligned}\int_{U_{j,I}} \tilde{f}_{C,j}(\mathbf{X})d\mathbf{X} &\leq \int_V \tilde{f}_{C,j}(\mathbf{X})d\mathbf{X} \\ &\leq \int_{U_{j,I}} \tilde{f}_{C,j}(\mathbf{X})d\mathbf{X} + \int_{U_{j,B}} \tilde{f}_{C,j}(\mathbf{X})d\mathbf{X}\end{aligned} \tag{A.2}$$

Expressions (A.1) and (A.2) can be combined to yield

$$\nu - \zeta \leq \int_V [f(\mathbf{x}) - \tilde{f}_{C,j}(\mathbf{X})]d\mathbf{X} \leq \nu + \xi \tag{A.3}$$

where

$$\begin{aligned}\nu &= \int_{U_{j,I}} [f(\mathbf{X}) - \tilde{f}_{C,j}(\mathbf{X})]d\mathbf{X}, \\ \zeta &= \int_{U_{j,B}} \tilde{f}_{C,j}(\mathbf{X})d\mathbf{X},\end{aligned}$$

and

$$\xi = \int_{U_{j,B}} f(\mathbf{X})d\mathbf{X}.$$

Let $N(U_{j,I})$ and $N(U_{j,B})$ denote the number of training samples that lie in $U_{j,I}$ and $U_{j,B}$ respectively. Choose γ sufficiently small so that the all kernels are strictly contained in their respective cells. This is possible because of the finite-support hypothesis. Under this condition, the integral of the cluster-based probability model over the m^{th} cell is simply the weight w_m of that cell, so that

$$\begin{aligned}\int_{U_{j,I}} \tilde{f}_{C,j}(\mathbf{X})d\mathbf{X} &= N(U_{j,I})/N \\ &\xrightarrow{\text{P}} \text{Prob}\{\mathbf{X} \in U_{j,I}\} = \int_{U_{j,I}} f(\mathbf{X})d\mathbf{X}\end{aligned} \tag{A.4}$$

and

$$\begin{aligned}\int_{U_{j,B}} \tilde{f}_{C,j}(\mathbf{X})d\mathbf{X} &= N(U_{j,B})/N \\ &\xrightarrow{\text{P}} \text{Prob}\{\mathbf{X} \in U_{j,B}\} = \int_{U_{j,B}} f(\mathbf{X})d\mathbf{X},\end{aligned} \tag{A.5}$$

where the arrows with the P's above them signify convergence in probability (as $N \to \infty$) by the weak law of large numbers. We can rewrite (A.4) and (A.5) as

$$\nu \xrightarrow{\text{P}} 0 \qquad \text{and} \qquad \zeta \xrightarrow{\text{P}} \xi. \tag{A.6}$$

Combining (A.3) and (A.6) gives

$$\lim_{N\to\infty} \text{Prob}\left\{\left|\int_V \left[f(\mathbf{X}) - \tilde{f}_{C,j}(\mathbf{X})\right] d\mathbf{X}\right| > \xi\right\} = 0,$$

where we have used the definition of convergence in probability, and, in simplifying the left-hand side of (A.3), the fact that if $a \xrightarrow{\text{P}} a'$ and $b \xrightarrow{\text{P}} b'$, then $(a+b) \xrightarrow{\text{P}} (a'+b')$.

All that remains is to show that $\xi < \delta$. To this end, choose ϵ sufficiently small so that

$$\int_{S_\epsilon} f(\mathbf{X})d\mathbf{X} < \delta. \tag{A.7}$$

This is possible because of condition (2.8) and the boundedness of f. Let j' be such that $\text{Diam}(\mathcal{U}_j) < \epsilon$ whenever $j > j'$. It follows that $U_{j,B} \subset S_\epsilon$, so that

$$\xi = \int_{U_{j,B}} f(\mathbf{X})d\mathbf{X} \leq \int_{S_\epsilon} f(\mathbf{X})d\mathbf{X} < \delta. \quad \blacksquare \tag{A.8}$$

Notice the crucial role played by $\{w_m\}$, the normalized cell populations. If these weights were absent, then convergence of the estimate would be contingent on whether the distribution of the kernel centers is the same as the distribution being modeled. Defining the weights in this way makes the estimate asymptotically insensitive to the precise choice of partition.

B The parameter γ

The spread parameter γ in (2.2) is necessary because the sample variance tends to underestimate the value required to allow the Gaussian kernels to add up to a uniform density in regions where the density is in fact uniform. Without $\gamma > 1$, the resulting density estimate would be too high at the kernel centers and too low at the cell boundaries.

This phenomenon is most easily illustrated in one dimension. Suppose that the true density f is uniform on $[-a, a] \subset$

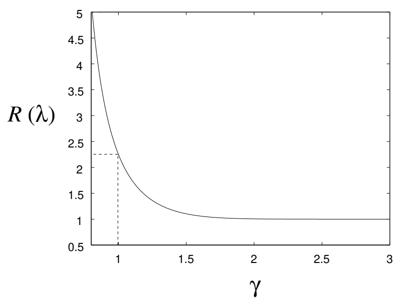


Figure 8: Ratio of estimated cell-center to cell-edge density for a uniform distribution in one dimension, as a function of the global spread parameter γ, for large M.

$\mathcal{R}$. Assume that a partition is obtained by dividing $[-a, a]$ into M equal-sized cells. The within-cell variance is then $a^2/(3M^2)$. Suppose that kernels of the form (2.2) are centered at every cell. For the innermost cell, we can examine the ratio of $\tilde{f}$ evaluated at the center of this cell ($X = 0$) to its value at the right boundary ($X = a/M$). Call this ratio $R(\gamma)$. Since the true distribution is uniform, we desire that $R(\gamma)$ be unity, at least when M is large. Assuming M is odd, the ratio is given by

$$\begin{aligned} R(\gamma) &= \tilde{f}_C(0)/\tilde{f}_C(a/M) \\ &= \frac{1 + 2 \sum_{m=1}^{(M-1)/2} e^{-6m^2/\gamma^2}}{e^{-3M^2/\gamma^2} + 2 \sum_{m=1}^{(M-1)/2} e^{-3(2m-1)^2/2\gamma^2}}, \end{aligned} \tag{B.1}$$

which is plotted in the limit as $M \to \infty$ in Figure 8. The figure shows that if $\gamma = 1$ (corresponding to leaving γ out altogether), then the center-to-edge probability ratio is 2.25, implying that $\tilde{f}$ is strongly nonuniform. As γ increases, the ratio approaches the desired value of unity as a limit. But using an excessively large value for γ reduces the ability of the estimate to adapt to local variations in regions where the true density is nonuniform. The best choice for γ depends on both the true distribution and on the particular partition $\mathcal{U}$. In practice, a good choice can be determined empirically by trying several values, then selecting the one that results in the best performance in the given application. Often this corresponds to choosing the value of γ which maximizes the likelihood of the given data. In obtaining the experimental results presented in Sections 3–5, the best values found for γ ranged between 1.1 and 1.5. It should be noted that the use of a single, global variance multiplier γ is suboptimal; it would be better (though computationally more expensive) to optimize all of the variances (and all of the means, for that matter) by some optimization technique such as the expectation-maximization (EM) algorithm, as discussed in Section 2.2. However, for large M, (e.g., $M = 1,024$), the computational cost of EM appears to be prohibitive, so that the device of introducing the spread parameter γ is a reasonable recourse.

References

[1] J. Besag, "Spatial interaction and the statistical analysis of lattice systems (with discussion)," *J. Roy. Stat. Soc., Ser. B*, vol. 36, pp. 192–236, 1974.

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

[3] D. Scott and J. Thompson, "Probability density estimation in higher dimensions," in *Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface* (J. Gentle, ed.), (Amsterdam), pp. 173–179, North-Holland, 1983.

[4] D. W. Scott, *Multivariate density estimation.* New York: John Wiley & Sons, 1992.

[5] B. Everitt and D. Hand, *Finite mixture distributions.* London: Chapman and Hall, 1981.

[6] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood, and the EM algorithm," *SIAM Review*, vol. 26, pp. 195–239, April 1984.

[7] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis.* New York: Wiley, 1973.

[8] L. R. and B.-H. Juang, *Fundamentals of speech recognition.* PTR Prentice Hall, 1993.

[9] S. Haykin, *Neural networks: a comprehensive foundation.* Macmillan, 1994.

[10] L. Devroye, *A course in density estimation.* Boston: Birkhauser, 1987.

[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* John Wiley and Sons, 1991.

[12] J. A. Rice, *Mathematical Statistics and Data Analysis.* Wadsworth and Brooks/Cole, 1988.

[13] D. Hand, *Kernel discriminant analysis.* Research Studies Press, 1982.

[14] A. Gersho and R. M. Gray, *Vector quantization and signal compression.* Kluwer academic publishers, 1991.

[15] A. K. Jain and R. C. Dubes, *Algorithms for clustering data.* Prentice Hall, 1988.

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

[17] J. Rissanen, "Modeling by shortest desciption length," *Automatica*, vol. 14, pp. 465–471, 1978.

[18] A. P. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Royal Stat. Soc.*, vol. 39, pp. 1–38, 1977.

[19] K. Popat and R. W. Picard, "A novel cluster-based probability model for texture synthesis, classification, and compression," in *Proc. SPIE Visual Communications '93*, (Cambridge, Mass.), 1993.

[20] A. Gersho, "Optimal nonlinear interpolative vector quantization," *IEEE Trans. Comm.*, vol. 38, pp. 1285–1287, Sept. 1990.

[21] G. Langdon, A. Gulati, and E. Seiler, "On the JPEG model for lossless image compression," in *Proc. IEEE Data Comp. Conf.*, (Utah), 1992.

[22] M. Rabbani and P. W. Jones, *Digital image compression techniques.* Bellingham, Washington: SPIE Optical Engineering Press, 1991.

[23] J. J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM J. Res. Develop.*, vol. 23, pp. 149–162, March 1979.

[24] J. Rissanen and G. G. Langdon, "Universal modeling and coding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 12–23, 1981.

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

Figure 9: The training set (first twenty-five images) and test set (last two images, *cman* and *lenna*) for experiments involving natural scenes. All are 8-bit monochrome, 512 × 512, except for *cman*, which is 256 × 256.

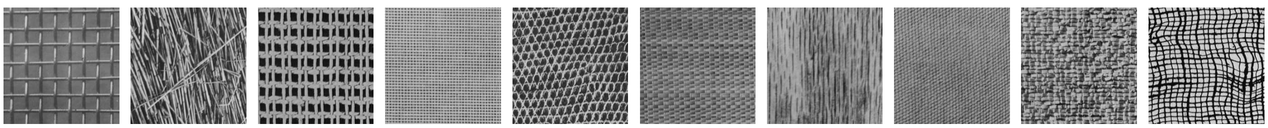


Figure 10: Textures from the Brodatz album [37]: *D1* (aluminum wire mesh), *D15* (straw), *D20* (magnified French canvas), *D21* (French canvas), *D22* (reptile skin), *D55* (straw matting), *D68* (wood grain), *D77* (cotton canvas), *D84* (raffia looped to a high pile), *D103* (loose burlap). These textures are 256 × 256, cut out of a 512 × 512 original. All images are 8-bit monochrome.

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

[27] G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Comm.*, vol. COM-29, pp. 858–867, June 1981.

[28] K. Popat and R. W. Picard, "Exaggerated consensus in lossless image compression," in *ICIP-94: 1994 International Conference on Image Processing*, (Austin, TX), IEEE, Nov. 1994.

[29] R. W. Picard, "Content access for image/video coding: 'The Fourth Criterion'," Tech. Rep. 295, MIT Media Lab, Perceptual Computing, Cambridge, MA, 1994. MPEG Doc. 127, Lausanne, 1995.

[30] K. Perlmutter, S. Perlmutter, R. Gray, R. Olshen, and K. Oehler, "Bayes risk weighted vector quantization with posterior estimation for image compression and classification," *IEEE Transactions on Image Processing*, vol. 5, pp. 347–360, February 1996.

[31] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the theory of neural computation*. Addison-Wesley, 1991.

[32] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, pp. 1481–1497, Sept. 1990.

[33] T. Poggio and F. Girosi, "Extension of a theory of networks for approximation and learning: Dimensionality reduction and clustering," A.I. Memo #1167, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1990.

[34] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[35] M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate Bayesian *a posteriori* probabilities," *Neural Computation*, vol. 3, pp. 461–483, 1991.

[36] 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*, vol. 12, pp. 193–198, April 1994.

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