MIT Media Laboratory Perceptual Computing Group Technical Report #234

Appears in *Proc. SPIE Visual Communications and Image Processing '93,* Boston, Nov. 8–11, 1993

Novel cluster-based probability model for texture synthesis, classification, and compression

Kris Popat and Rosalind W. Picard

Massachusetts Institute of Technology, Media Laboratory
Cambridge, Massachusetts 02139-4307
Telephone: 617/253-0335 *Fax:* 617/253-8874 *email:* popat@media.mit.edu, picard@media.mit.edu

ABSTRACT

We present a new probabilistic modeling technique for high-dimensional vector sources, and consider its application to the problems of texture synthesis, classification, and compression. Our model combines kernel estimation with clustering, to obtain a semiparametric probability mass function estimate which summarizes — rather than contains — the training data. Because the model is cluster based, it is inferable from a limited set of training data, despite the model's high dimensionality. Moreover, its functional form allows recursive implementation that avoids exponential growth in required memory as the number of dimensions increases. Experimental results are presented for each of the three applications considered.

1. INTRODUCTION

In many information processing tasks individual data samples exhibit a great deal of statistical interdependence, and should be treated jointly (e.g., in vectors) rather than separately. For some tasks this requires modeling vectors probabilistically.

Estimating and using probabilistic models when dealing with a vector source poses difficulties not present in the scalar case. To illustrate, consider a simple histogram estimate of the probability density function (PDF) of a 10-dimensional vector. Suppose uniform hypercubic bins are used in the histogram, with 256 to a side. Then the total number of bins is astronomically large: $256^{10} \approx 1.2 \times 10^{24}$. This causes two serious problems: one cannot obtain enough training data to fill this histogram to the point where it would provide a reliable PDF estimate, nor can one obtain enough computer memory to store the result.

Consider the training problem. The overwhelming majority of the histogram cells will be empty, even after training on as large a data set as is practical. It is essential that all cells, including the empty ones, be assigned reliable probability estimates. (In fact, when the model is used on data outside the training set, *most* of the encountered vectors will correspond to empty histogram cells.) For certain sources it is reasonable to infer the probability of an empty cell from the populations of "surrounding" non-empty cells. To do this is to make an assumption about the *smoothness* of the PDF — that a small change in the vector causes a small change in the probability. This assumption is reasonable in situations where the vector elements correspond to measured physical quantities, such as light, temperature, or sound.

A well-known method of PDF estimation that uses this assumption is Parzen estimation. In that approach, a kernel PDF is reproduced at each training point, and the results summed. The kernel serves to "spread out" probability to the space that surrounds each training vector. In the Parzen approach, the model *contains* the training data, and while it does not require as much memory as a histogram, it nevertheless becomes unwieldy and computationally expensive as the amount of training data gets large. It is desirable that the model summarize the data rather than contain it. The Parzen technique can be modified to this end by replacing the training data with a much smaller set of representative points, and by adapting the sizes and shapes of the kernels to match the statistics of the regions they represent. The model presented here uses clustering to obtain this smaller set of points.

A remaining problem with high-dimensional PDF modeling is revealed by assuming that a good estimate of the PDF has already been obtained, and that we now wish to apply the model to a particular information processing task. Returning to our example of the 10-dimensional source, suppose we wish to quantize the vectors into their histogram bins and arithmetic code the result. Such a problem arises, for instance, in entropy-constrained lattice quantization.[1] To perform the entropy coding directly would require working with an alphabet of about 10^{24} members, each of which has probability on the order of 10^{-24} — clearly an infeasible task. An obvious alternative is to break down the joint PDF into a product of conditional PDFs for the vector elements, but to do so, the model PDF must be of a form that is amenable to such factorization. For example, a separable PDF model is of such a form, but does not capture the interdependence of the vector elements. The proposed model, which does capture much of the interdependence while allowing simple recursive computation of the terms in the factorization, is described in the next section.

2. CLUSTER-BASED PROBABILITY MASS FUNCTION

Let $\mathbf{x} = [x_1, \ldots, x_N]$ be a discrete* random vector that follows a probability mass function (PMF) $p(\mathbf{x})$. We assume that the vector elements $\{x_i\}$ share a common alphabet of K letters, so that the alphabet for $\mathbf{x}$ has K^N members. We wish to estimate $p(\mathbf{x})$ from a large but limited set of training vectors $\{\mathbf{X}^1, \ldots, \mathbf{X}^L\}$, using a model that makes feasible element-by-element processing via the chain rule

$$\begin{aligned} p(\mathbf{x}) &= p(x_1, \ldots, x_N) \\ &= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\cdots p(x_N|x_1, \ldots, x_{N-1}). \end{aligned} \tag{1}$$

2.1 Formulation

Our approach is to combine kernel estimation with clustering. First, a cluster analysis is performed on the training data using a standard clustering procedure; for our work we selected the LBG algorithm[2]. The desired PMF is then modeled as a weighted sum of M component PMFs, each centered on a different cluster. The number of components M is a parameter, and determines both the accuracy and computational complexity of the model. Let $q(\mathbf{x})$ denote the estimate; let $q_m(\mathbf{x})$ denote the m^{th} component PMF of this estimate. Then

$$q(\mathbf{x}) = \sum_{m=1}^{M} w_m q_m(\mathbf{x}), \tag{2}$$

where $w_m > 0$ for all m and $\sum_{m=1}^{M} w_m = 1$.

To facilitate factoring $q(\mathbf{x})$ as in (1), the q_m's are restricted to be separable. That is, each q_m is required to be expressible in the form

$$q_m(\mathbf{x}) = \prod_{n=1}^{N} f_{m,n}(x_n), \tag{3}$$

where the $f_{m,n}$'s are 1-dimensional PMFs. It is worth emphasizing that, although the component PMFs are separable, the resulting overall PMF $q(\mathbf{x})$ is generally far from separable.

A number of choices are possible for the $f_{m,n}$'s; indeed there may be no unique best choice in a given situation. A reasonable choice can be obtained by assuming that, in general, the density of training points peaks at the cluster centers, and smoothly and monotonically diminishes away from those centers. This suggests that a discretized Gaussian is a reasonable choice, i.e.,

$$f_{m,n}(x) = K_{m,n} e^{-(x_n - \mu_{m,n})^2/(2\sigma_{m,n}^2)}, \tag{4}$$

where $K_{m,n}$ is such that

$$\sum_{\text{all } x} f_{m,n}(x) = 1.$$

* In practice, continuous-valued quantities must be discretized for computer processing. We therefore limit our consideration to discrete distributions for the remainder of this paper.

This choice of $f_{m,n}$ is used throughout this paper. The parameters $\mu_{m,n}$ and $\sigma_{m,n}$ are obtained directly from the statistics of each cluster. In particular, $\mu_{m,n}$ and $\sigma_{m,n}$ are taken as the sample mean and sample standard deviation, respectively, along dimension n, of the training points falling within the Voronoi region of cluster m. The weighting factor w_m is taken to be the population of cluster m in the training data, divided by the total number of training points.

Combining the above, we have the following expression for $q(\mathbf{x})$:

$$q(\mathbf{x}) = \sum_{m=1}^{M} w_m \prod_{n=1}^{N} K_{m,n} e^{-(x_n - \mu_{m,n})^2/(2\sigma^2_{m,n})}. \tag{5}$$

Although use of the proposed technique mitigates some of the problems associated with high dimensionality, it does not eliminate them. As a practical matter, the value of N must be restricted. If N is unreasonably large, then the problem of insufficient training data once again emerges; in addition, an excessive number of clusters is required to approximate the true PMF. We have found that restricting N to be between 5 and 15 generally results in the best system performance.

2.2 Recursive computation

The principal motivation for restricting the form of the PMF is to allow separate processing of the vector elements via the chain rule (1) without sacrificing accuracy. This is an important property; it makes explicit evaluation of the cluster-based PMF feasible. In this section we describe how such computation can be accomplished.

We first note that if no restriction is placed on the form of the PMF, then the chain rule would really offer no simplification. Computing the final factor on the right-hand side of (1), for instance, involves evaluating $p(\mathbf{x})$, which is precisely what we are trying to avoid.

Let X_n denote the particular value assumed by x_n in a sample realization $\mathbf{X}$ of $\mathbf{x}$. It is assumed that $X_1, \ldots, X_{n-1}$ are available for use in the computation of $q(x_n \,|\, X_1, \ldots, X_{n-1})$. This assumption is justified in a number of important applications, including those considered in this paper. Specifically, in the cases of decoding and synthesis, X_n is the direct result of processing $q(x_n \,|\, X_1, \ldots, X_{n-1})$. In typical applications of encoding and classification, all of the X_n's are available *a priori*. Let $\mathcal{I} = \{i_1, \ldots, i_{N'}\}$ be a subset of the indices $\{1, \ldots, N\}$, where $N' \leq N$. By summing over the indices not in $\mathcal{I}$, it is easy to verify that

$$q(x_{i_1}, \ldots, x_{i_{N'}}) = \sum_{m=1}^{M} w_m \prod_{n \in \mathcal{I}} f_{m,n}(x_n), \tag{6}$$

i.e., the estimate of the marginal PMF can be obtained by simply ignoring the unwanted dimensions. Using this property and the definition of conditional probability, we obtain

$$\begin{aligned} q(x_1) &= \frac{\sum_{m=1}^{M} w_m f_{m,1}(x_1)}{C_1}; \\ q(x_2 \,|\, X_1) &= \frac{\sum_{m=1}^{M} [w_m f_{m,1}(X_1)] f_{m,2}(x_2)}{C_2}; \\ q(x_3 \,|\, X_1, X_2) &= \frac{\sum_{m=1}^{M} [w_m f_{m,1}(X_1) f_{m,2}(X_2)] f_{m,3}(x_3)}{C_3}; \\ &\vdots \\ &\text{etc.,} \end{aligned} \tag{7}$$

where each C_n is the sum of the corresponding numerator over all values of x_n. A recursive procedure for computing the conditional PMFs is obtained by introducing a variable $r_{m,n}$ and identifying it with the bracketed quantities above. Specifically, if we define

$$r_{m,n} = \begin{cases} w_m & \text{if } n = 0; \\ C'_n r_{m,n-1} f_{m,n}(X_n) & \text{if } 1 \leq n \leq N, \end{cases} \tag{8}$$

where C'_n (different from C_n above) is chosen such that

$$\sum_{m=1}^{M} r_{m,n} = 1,$$

then the conditional PMF estimate for the n^{th} element can be written

$$q(x_n|X_1,\ldots,X_{n-1}) = \sum_{m=1}^{M} r_{m,n-1} f_{m,n}(x_n). \tag{9}$$

Expressions (8) and (9) provide the desired means of processing the vector elements separately.

2.3 Relationship to mixture models, radial basis functions, and neural networks

From expression (5) it is clear that the proposed model can be thought of as a special case of a mixture density, in particular, one having separable components. However, the interpretation and use of the quantities that comprise the proposed model, as well as the overall goal of the model, are very different from those of a traditional mixture density, as we now briefly discuss.

A mixture density approach is traditionally used in situations calling for unsupervised learning. Specifically, a mixture model arises when an observation $\mathbf{x}$ is believed to obey probability law $p_m(\mathbf{x}\,|\,\omega_m)$ with probability $P(\omega_m)$, where $\omega_1,\ldots,\omega_M$ are "states of nature," or classes.[3] Formally $P(\omega_m)$ corresponds to our w_m and $p_m(\mathbf{x}\,|\,\omega_m)$ to our $q_m(\mathbf{x})$, but the interpretations are completely different. In the mixture case each of the component densities corresponds to a different, physically meaningful class, and the objective is to estimate the class-conditional component densities $\{p_m(\mathbf{x}\,|\,\omega_m)\}$ and the priors $\{P(\omega_m)\}$ from observations. In contrast, the component PMFs in the proposed model are merely "building blocks" used to achieve the desired overall PMF shape; their physical significance, if any, has yet to be explored. Our objective is not to decompose the PMF of the observation, but rather to summarize its shape in a computationally convenient form, so that good performance can be achieved in applications where the model is employed.

If the $f_{m,n}$'s are chosen appropriately, then expression (5) amounts to a radial basis function (RBF) approximation to $p(\mathbf{x})$. Consequently, one might expect that the RBF literature can provide insight into such issues as training, means of implementation, and bounds on approximation accuracy.[4,5] Much of the RBF literature is concerned with interpolation[6] and approximation with respect to L_2 norm[7,8] or L_1 norm.[9] In our case, the function being approximated is a PMF. As it turns out, the approximation criterion that is most natural in the applications we consider is not a distance metric in the strict sense of the term. Specifically, the appropriate criterion is the information divergence, or Kullback-Leibler number,[10] which is asymmetric and does not satisfy the triangle inequality. Consequently, much of the RBF approximation theory does not apply. In any event, regarding expression (5) as an RBF expansion is useful in that it provides an alternative interpretation of the model — as a means of *interpolating probability.*

The relationship to RBFs suggests a connection to neural networks. Our approach does have certain elements in common with neural network approaches.* Both are capable of learning complex, nonlinear relationships from training data, and exploiting them in performing various information processing tasks. We expect that the applications we are considering could be handled by an appropriate type of neural network with a comparable level of performance. However, a distinction can be drawn between the two approaches: there is nothing inherently "neural" about our technique, and it is not connected to any specific class of hardware topology.

If we compare our technique with a classical neural network 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, our method provides an explicit probabilistic model for the source, which can be used equally well in a variety of applications like compression, restoration, synthesis, and classification. In principle, this distinction vanishes when the goal of the neural network is specifically to estimate the PMF,[11] rather than to carry out the ultimate information processing task. In this case, the approaches may be accomplishing the same thing in different ways. To understand the similarities and differences more fully requires study beyond the scope of this paper.

* If a sufficiently broad interpretation is used, then an implementation of our approach using a parallel structure might be regarded as a particular neural network with a specific learning algorithm.

3. APPLICATION TO TEXTURE SYNTHESIS

The cluster-based probability model summarizes the training data – but how well? One way to evaluate its success is to consider the question, "what patterns is this model most likely to synthesize?" Texture synthesis not only helps us *see* what information the model has captured, but also has direct applications in areas such as model-based image coding, computer graphics, and multimedia design.

There has been a lot of previous work on texture synthesis, with methods loosely categorized as either structural or statistical.[12] Structural methods tend to work best on regular patterns like bricks or lizard skin; statistical tend to work best on stochastic patterns like sand or cork.

The greatest success with probabilistic models in the literature has been on "stochastic microtextures", patterns which not only lack regular structure, but which also exhibit only local variations. (Informally, this usually translates into variations that span fewer than ten pixels.) Probabilistic models used for synthesis of such patterns include Markov random fields,[13] noncausal autoregressive models,[14] and Long-Correlation random fields.[15] Deterministic methods have also been successful for synthesizing textures with a given second- or third-order PMF.[16,17] However, none of these have been shown to be successful at synthesizing general structured textures.

The key problem with probabilistic texture synthesis models has been their inability to capture long-range or higher-order information in the textures. To do so with most models would lead to the types of difficulties described in Section 1. Consequently, there do not appear to be any attempts in the literature to incorporate higher than third-order statistics.

The new cluster-based probability model is not only able to characterize interactions beyond third-order, but, as we will now see, it is also able to synthesize structure that other probabilistic models have not yet been shown to capture.

3.1 Nonhierarchical texture synthesis

Using the recursive implementation, pixels are synthesized sequentially. The simplest approach is to synthesize the pixels in raster order. Each pixel is assigned a pseudorandom value that is generated according to its conditional PMF, where the conditioning values are a subset of the previously generated pixels. The locations of the conditioning pixels relative to the current one define a *causal conditioning neighborhood.* Two such neighborhoods are shown in Figure 1.

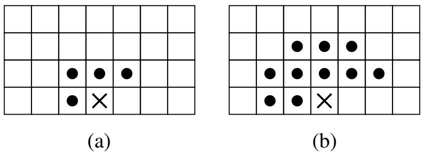

Figure 1. Causal conditioning neighborhoods used with the proposed model in nonhierarchical texture synthesis and compression.

Before the above procedure can be applied, the model must be trained. The first step is to obtain a sequence of sample vectors from a training image. For each pixel in the training image, the neighborhood pixels and current pixel are concatenated into a vector, with the current pixel appearing as the last element. A cluster-based model is then trained on these vectors using the procedure described in Section 2.1. The parameter M is selected according to the available computational resources and the desired fidelity of the model; in our work we have used values ranging between 32 and 2048. To speed the training process, the set of training vectors can be decimated prior to clustering. We used at least $20M$ training vectors for the examples in this paper.

The final step is to apply the model to generate each pixel in raster order. Expressions (8) and (9) are used to obtain the conditional PMF for each pixel, which is then used in generating the pseudorandom pixel value. In cases where the neighborhood specifies conditioning locations outside the image, a constant value of 128 is assumed; we have found that border effects due to this choice usually die away after a few rows or columns.

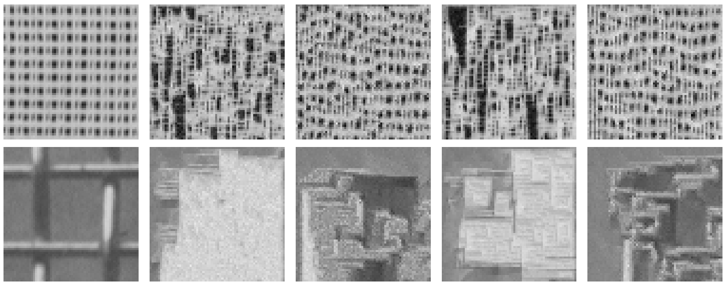

Figure 2. Results of applying nonhierarchical texture synthesis procedure to two Brodatz textures: D21 (French canvas) and D1 (aluminum wire mesh). A 64×64 patch of the original is shown on the left, followed by four synthesized textures that differ in their condition masks and values of M. From left to right, mask (a) and $M = 128$; mask (b) and $M = 128$; mask (a) and $M = 512$; mask (b) and $M = 512$.

Figure 2 shows the results of using this technique to synthesize two natural textures selected from the Brodatz[18] collection. Notice that in the best case, the technique performs reasonably on the microtexture (top), but fails completely on the bottom texture which has macroscopic structure. This is to be expected, since the pixels in the macroscopic texture have statistical interdependences that extend spatially well beyond the size of the neighborhood. For the reasons mentioned at the end of section 2.1, the problem cannot be fixed by simply using larger neighborhoods.

3.2 Hierarchical texture synthesis

The above technique can be modified so that the texture is synthesized in several stages, beginning with a coarse resolution version of the texture (establishing macroscopic structure), and proceeding to progressively finer resolutions (filling in microscopic detail).

A number of variations on this theme are possible. Here we consider a simple hierarchical approach, wherein each finer resolution is obtained by first upsampling the current resolution, then filling in the missing pixel values using the proposed cluster-based model. Unlike filtering, which is the conventional method of interpolating the missing pixels, the proposed model is able to exploit the complex nonlinear statistical relationships that exist among pixels, both within and across resolution levels.

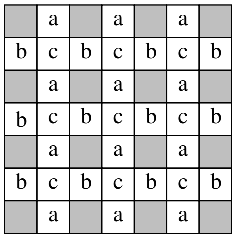

Figure 3. Three classes of pixel — indicated as a, b, and c — are determined by positional relationship to the upsampled (shaded) pixels.

The details of this technique are now summarized. First, a coarse image is obtained using the nonhierarchical technique described earlier. This image is upsampled by a factor of two in each dimension. The pixels that need to be filled in can be divided into three classes, according to their position relative to the upsampling lattice (see Figure 3). Each of these classes has a different statistical relationship to the coarser resolution pixels, and must therefore have its own probabilistic model. The models also differ across pairs of resolution levels; that is, we do not impose statistical self-similarity across scale. As in the nonhierarchical case, the neighborhood includes only those pixels which have already been given values: those carried over from the coarser level, and those previously synthesized at the current level. A possible set of neighborhoods is shown in Figure 4.

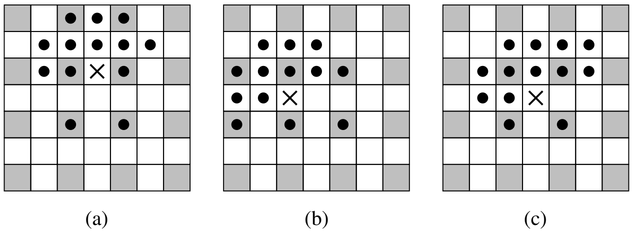

Figure 4. Conditioning neighborhoods used with the proposed model in hierarchical texture synthesis. Shaded pixels are those carried over from the previous resolution level.

The models are trained in a straightforward way. Training data is obtained by first generating a set of subsampled versions of the training image, then collecting vectors defined by the neighborhoods for each of the three pixel classes, and for each pair of resolution levels. To obtain sufficient training data at coarse resolutions, the process is repeated using different subsampling offsets. Parameter estimation is carried out using the technique described in section 2.1.

We applied this technique in two different ways. In the first, pseudorandom values are assigned to each pixel according to its conditional PMF, just as in the nonhierarchical technique. The second way uses the conditional PMF to assign pixel values by maximum-likelihood (ML) interpolation. In the first method the synthesized texture falls within a typical set[10] with respect to the model, while in the second, the synthesized texture has individually high probability. We will refer to the first method as *random*; the second, as *deterministic.*

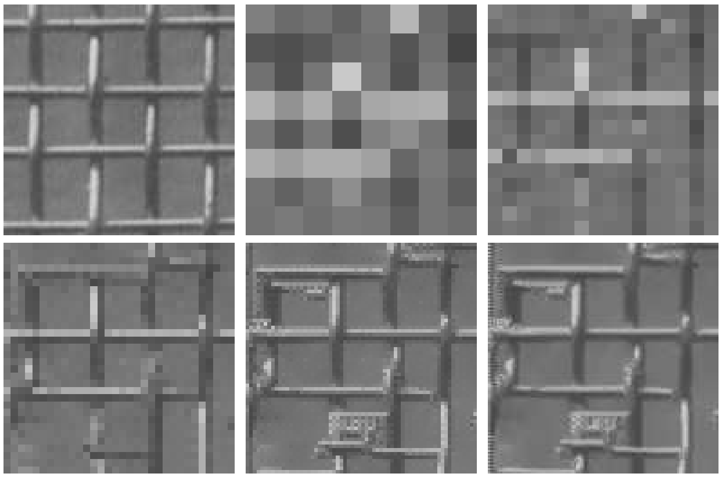

Figure 5. Results of deterministic hierarchical synthesis of Brodatz texture D1, with $M = 2048$ and $N = 14$. The original training image is shown at the top left; the remaining images are the synthesis results at five successively finer resolution levels. The conditioning neighborhoods used are those shown in Figure 4. Not shown is the coarsest resolution level; in it the pixels appear to be essentially random.

Figure 5 shows the evolution of detail in the deterministic hierarchical synthesis of Brodatz texture D1. Notice that much of the character of the original is present in the full-resolution result, including highlighting, bending and occlusion of the wire strands.

Full-resolution results for eight Brodatz natural textures are shown in Figure 6, using both the random and deterministic methods. For all textures, both methods capture much of the statistical and structural character of the original. The deterministic outperforms the random method on textures having greater macroscopic structure.

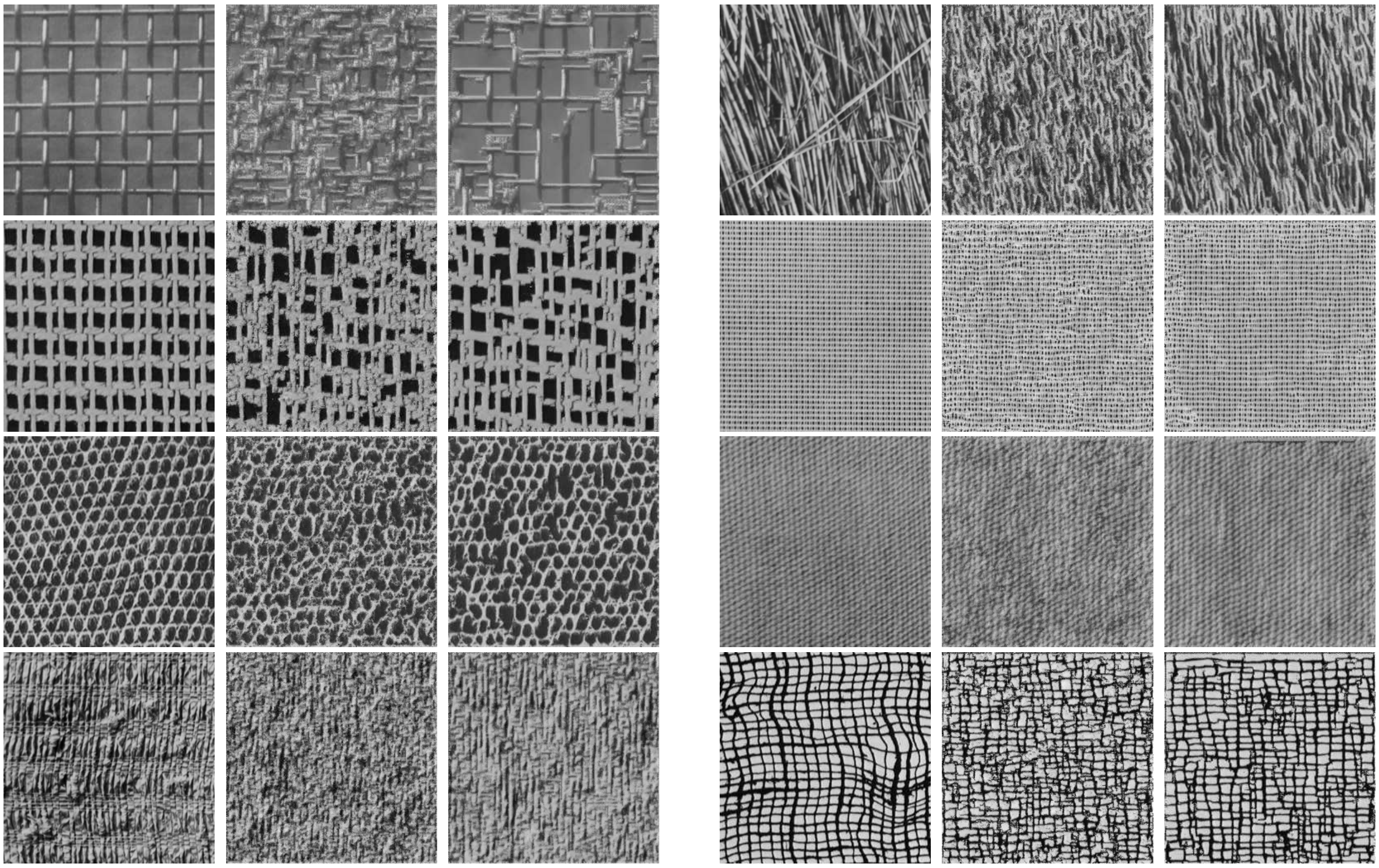

Figure 6. Results of hierarchical synthesis of eight Brodatz textures. Left to right and top to bottom, the textures are D1 (aluminum wire mesh), D15 (straw), D20 (magnified French canvas), D21 (French canvas), D22 (reptile skin), D77 (cotton canvas), D80 (straw cloth), and D103 (loose burlap). In each case the three pictures are the original (left), random synthesized (middle), deterministic synthesized (right). The pictures are 256×256. In all cases, $M = 2048$ and the neighborhoods shown in Figure 4 are used.

4. APPLICATION TO TEXTURE COMPRESSION

Data compression systems can be divided into two classes: lossless systems, which are strictly reversible, and lossy systems, which introduce some distortion but generally achieve much greater compression.

In a sense we have already described a form of lossy texture compression in the previous section — texture synthesis from a model. One can easily imagine an image compression system based on this approach. Segmentation and model information would be encoded and sent to the receiver. There, the information would be used to fill in the segments so that the reconstructed image resembles the original. The large number of bits needed to specify the model would be amortized over the size of the region, so that significant compression would be achieved when the regions are large. However, the fidelity of such a system would be quite low in regions for which the model is weak. Consequently, such an approach would be suitable only in applications where strict fidelity is not required.

Lossless compression systems are based on entropy coding, a process in which a sequence of source letters is reversibly mapped to a compact, variable-rate code bit stream, according to an assumed probabilistic model of the source. Langdon and Rissanen[19] have devised an efficient lossless compression technique for binary images, and their technique has been extended to grayscale images by breaking the image up into bit planes.[20] This section considers an alternative approach, wherein the proposed PMF model is used to directly arithmetic-code the pixel values. Our approach differs from the two approaches cited mainly in that the proposed model is able to estimate the probability of previously unseen states, whereas the models used in the binary and bit-plane methods are restricted to occurrence-count estimation.

Although we restrict consideration to lossless compression, much of our discussion is relevant to certain lossy methods as well, since lossy compression systems usually employ a lossless compression element.

4.1 Using the model with arithmetic coding

Arithmetic coding is a form of entropy coding that in certain applications offers significant advantages over more traditional methods. 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. Arithmetic coding is also inherently adaptive, allows encoding of large-alphabet low-entropy sources without alphabet extension, and allows the probabilistic model to be specified explicitly and separately from the actual coder.[21,22,23] The last property, which is of both philosophical and practical importance, is illustrated schematically in Figure 7. There are several varieties of arithmetic coder; here we assume that the K-ary version of the Langdon-Rissanen technique described in previous work[23] is employed. The use of the proposed model with such an arithmetic coder is now discussed.

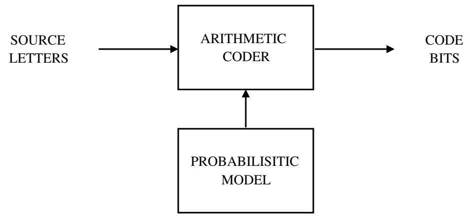

Figure 7. An arithmetic coder allows convenient and natural separation of the probabilistic model from the bit-producing mechanism.

The probabilistic model supplied to the arithmetic coder is in the form of a sequence of 1-dimensional PMFs, one for each source letter. The PMFs supplied to the coder must satisfy certain constraints to ensure unique decodability of the code bit sequence. Assume that the source alphabet consists of K letters. Let $\xi(k)$ denote the true probability that the current source letter assumes value k, let $\tilde{\xi}(k)$ denote the approximation to $\xi(k)$ used by the arithmetic coder, and let ρ represent the number of bits of precision with which quantities can be manipulated and stored by the processor. Unique decodability is ensured when $\tilde{\xi}$ satisfies

$$\tilde{\xi}(k) = \gamma_k 2^{-\rho} \quad \text{where } \gamma_k \text{ is an integer} \geq 2. \tag{10}$$

Greatest coding efficiency would be achieved when $\tilde{\xi} = \xi$, but constraint (10) generally prevents this. The effect can be minimized by choosing the particular $\tilde{\xi}$ from the feasible set that minimizes average bit rate, which is well approximated by

$$\begin{aligned} R &= E[-\log_2 \tilde{\xi}(k)] \\ &= -\sum_{k=1}^{K} \xi(k) \log_2 \tilde{\xi}(k). \end{aligned} \tag{11}$$

Comparing this with the information divergence[10] between ξ and $\tilde{\xi}$,

$$D(\xi \,||\, \tilde{\xi}) = \sum_{k=1}^{K} \xi(k) \log_2 \frac{\xi(k)}{\tilde{\xi}(k)},$$

we see that minimizing the average bit rate with respect to the assumed PMF is equivalent to minimizing the information divergence between the true and assumed PMFs.

Subject to (10), R can be minimized over $\tilde{\xi}$ via a particular constrained optimization algorithm.[23] When the cluster-based model is used to supply the probabilities for arithmetic coding, $q(x_n \,|\, X_1, \ldots, X_{n-1})$ plays the role of $\tilde{\xi}$. The

Table 1

Estimated* arithmetic coding rates (bits/pixel)

texture	empirical marginal entropy	$M = 128$		$M = 512$	
		mask (a)	mask (b)	mask (a)	mask (b)
D1	6.65	4.10	4.26	4.04	4.09
D20	7.02	4.54	4.88	4.45	4.59
D22	7.27	5.50	5.76	5.48	5.56
D77	6.82	5.10	5.24	5.06	5.03
D103	7.50	5.41	5.74	5.13	5.38

* Assuming at least 24 bits of precision.

computational cost of carrying out the optimization on $q(x_n \,|\, X_1, \ldots, X_{n-1})$ for every n would be prohibitive, so we seek an alternative.

Notice that the $f_{m,n}$'s need not be recomputed each time they are used; they can instead be stored in an $N \times M \times K$ table. Suppose that the $f_{m,n}$'s are adjusted before they are stored so that they individually satisfy (10). Then it can be shown that every $q(x_n \,|\, X_1, \ldots, X_{n-1})$ will satisfy (10), provided that the computation in expression (9) is carried out using an intermediate precision of 2ρ bits. Although the $q(x_n \,|\, X_1, \ldots, X_{n-1})$'s that result do not necessarily minimize R even if the individual $f_{m,n}$'s do, in practice, we have found that the inefficiency resulting from failing to optimize the $q(x_n \,|\, X_1, \ldots, X_{n-1})$'s is usually quite small.

4.2 Compression of textures

The model can be used with arithmetic coding to compress textures (or natural scenes, for that matter) in a number of ways. The simplest approach is a nonhierarchical technique similar to that described for texture synthesis in Section 3.1. The pixels are coded sequentially in raster order, using for each pixel a conditional PMF based on a causal neighborhood. Conditioning pixels which fall outside the image can be set to some constant value, say 128.

Compression results are summarized for five Brodatz textures in Table 1, using the masks shown in Figure 1. The numbers are estimates based on PMFs quantized for use with the arithmetic coder, but the actual bit stream was not produced. Hence these results must be regarded as preliminary. For comparison, the empirical entropy based on the marginal histogram is also given. Training was done on a 256×256 portion of the original, while testing was performed outside the training set on a separate 128×128 patch. Notice that the smaller mask usually gives slightly better compression than the larger mask, with the difference decreasing as M increases. This is probably due to the value of M not being large enough for the PMF to be accurately modeled in higher dimensional space.

Performance appears to be in the right ballpark, based on a comparison with numbers reported for lossless compression of natural scenes (typically, between 4.5 and 5.5 bits/pixel).[20] The compression ratio is expected to improve substantially by using a hierarchical approach and making the model adaptive. This is left as an area for future work.

5. APPLICATION TO TEXTURE CLASSIFICATION

Because the cluster-based model provides an explicit PMF for the source, it is straightforward to use it in a Bayesian classifier. Moreover, the classification can be carried out directly in the pixel domain, instead of employing more complex features. This is possible because the model captures much of the high-order statistical characteristics of the source, without having to rely on a lower-dimensional feature space.

Suppose there are J equally likely classes, and that vectors drawn from class j follow the PMF $p_j(\mathbf{x})$. Let $\mathcal{S} = \{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^L\}$ be a sample drawn from some unknown class, the identity of which we wish to determine with minimum probability of error. Since the priors are equal, the minimum probability of error decision rule is simply the maximum likelihood rule.[24] Assuming for now that the vectors in $\mathcal{S}$ are independent, the likelihood function for class j is simply

$$p_j(\mathbf{X}^1)p_j(\mathbf{X}^2)\cdots p_j(\mathbf{X}^L). \tag{12}$$

By taking the logarithm we obtain the decision rule:

$$\text{Choose } j \text{ for which } \sum_{l=1}^{L} \log p_j(\mathbf{X}^l) \text{ is maximized.} \tag{13}$$

To evaluate each $\log p_j(\mathbf{X}^l)$, the recursive technique described in Section 2.2 can be used, taking the sum of the logarithms of the conditional probabilities.

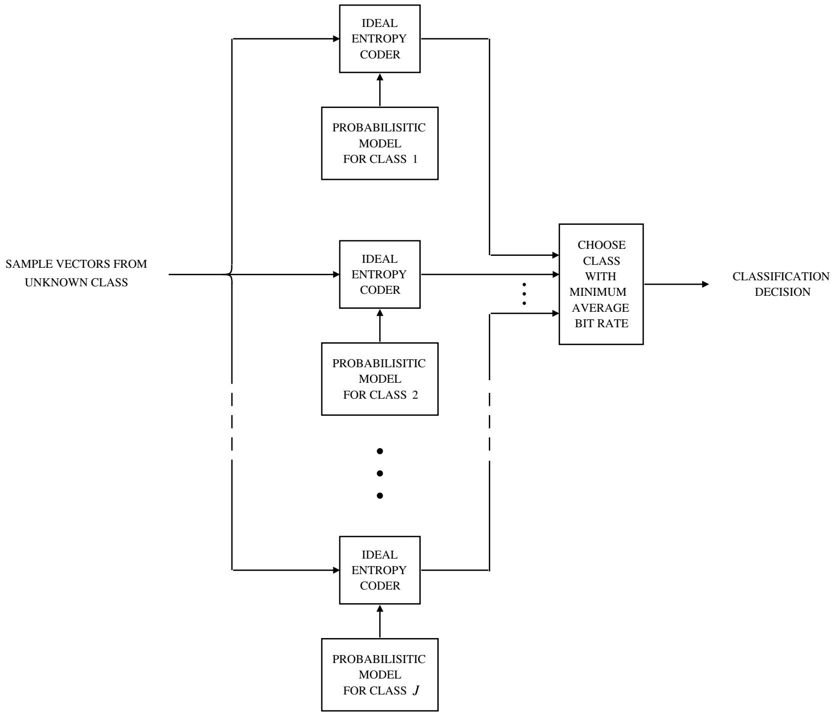

Figure 8. A connection between data compression and classification: the minimum-error-probability rule can be realized using a bank of ideal entropy coders, each tuned to a different source.

Because an ideal entropy coder produces on average $-\log_2 p_j(\mathbf{x})$ bits per vector $\mathbf{x}$, the decision rule can be implemented using the structure shown in Figure 8. Of course in practice, real entropy coders would not be used, since only the rates and not the actual code bits are needed. Nevertheless, the implementation shows a connection between data compression and classification.

5.1 Texture Classification

Since all of the pixels to be classified are available *a priori*, the vectors can be based on noncausal neighborhoods. For instance, the neighborhood shown in Figure 9 can be used.

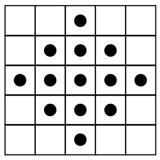

Figure 9. Neighborhood used with proposed model in texture classification.

Formed in this way, vectors corresponding to adjacent pixels are not independent, so that (12) is not strictly justified. The resulting decision rule essentially ignores 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 still results in a useful decision rule.

The summation in the decision rule (13) can be accomplished by averaging, and this averaging can be carried out in at least two different ways. One is by spatial lowpass filtering, which is based on the assumption that pixels close together come from the same class. Another is by averaging *across several models,* each working at a different resolution. Because averaging logarithms corresponds to taking a product, this is a way of requiring that the sample texture match the candidate texture simultaneously at several scales. These two types of averaging are not exclusive; a classification system can employ both.

5.2 Four-class example

We applied the technique to classifying regions in a composite test image made of four Brodatz textures, shown on the left in Figure 10. The textures are, clockwise from top left, D68, D55, D77, and D84. For each class, three models were trained: one using the neighborhood shown in Figure 9, and the other two using the same neighborhood but with the pixel locations scaled by factors of 2 and 4, respectively, around the center. Thus, models were obtained at three different resolution scales.

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, the logarithm of the probability of each vector was computed using the appropriate model, and the results averaged across resolutions. The classification results are shown in the middle image in Figure 10. The test was repeated using spatial averaging in addition to resolution averaging; the results are shown on the right in Figure 10. A 7-tap separable lowpass filter was used to perform the spatial averaging.

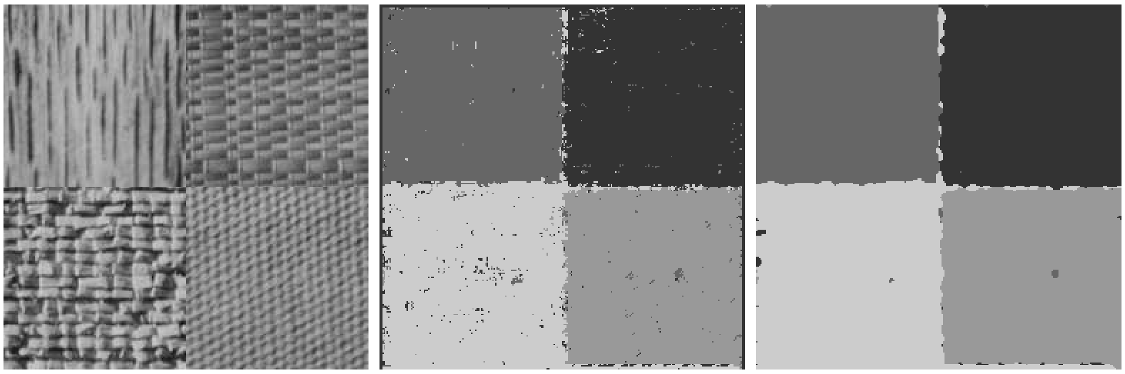

Figure 10. Four-class example of texture classification using the proposed model. From left to right, the images are: the original composite test image, classification using resolution averaging only, and classification using both resolution and spatial averaging.

6. SUMMARY

A novel, cluster-based probabilistic modeling technique for vector sources has been presented and applied to three problems with textured data. The model uses clustering to *summarize* the data, and is of a form that is amenable to recursive chain-rule evaluation without loss of accuracy. It is thus able to represent high-dimensional probability densities without the outrageous demands on training data and memory that are usually required by high-dimensional models.

A one-pass hierarchical method for synthesizing textures using the model was developed. Two variations on this method were considered: one random and the other deterministic. Both were found to perform well in recreating natural microtextures. In the case of textures with macroscopic structure, the deterministic approach was found to perform better.

The texture synthesis method developed here can be used as part of a model-based lossy compression system. Additionally, a lossless compression system was developed using the model and found to compress by an average factor of about 1.6 over 5 textures. It was also shown that the proposed model interfaces naturally with an arithmetic coder, resulting in a simple and quite general lossless coding system.

Finally, we applied the new model to the problem of supervised texture classification. This application differs fundamentally from most model-based classification, in that the classification is carried out directly in the pixel domain, instead of employing more complex features. Estimation of the model parameters does not need to be done for the incoming data. These properties give the classifier the potential to work even on small and irregularly shaped regions. The classifier was shown to give outstanding performance on a four-class problem.

Acknowledgement. The authors would like to thank Alex Sherstinsky for valuable comments on Section 2.3. This research was supported in part by Hewlett Packard Laboratories, Palo Alto, California.

7. REFERENCES

1. K. Popat and R. Picard, "Cluster-based probability model applied to image restoration and compression," Perceptual Computing Group Technical Report #TR 233, M.I.T. Media Laboratory, 1993.
2. 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.
3. R. O. Duda and P. E. Hart, *Pattern classification and scene analysis.* Wiley, 1973.
4. T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, pp. 1481–1497, Sept. 1990.
5. 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.
6. M. Powell, "Radial basis functions for multivariate interpolation: A review," in *Algorithms for Approximation* (J. Mason and M. Cox, eds.), Oxford, U.K.: Clarendon Press, 1987.
7. D. Broomhead and D. Lowe, "Radial basis functions, multi-variable functional approximation and adaptive networks," Memorandum No. 4148, Royal Signals and Radar Establishment, Great Malvern, Worc., U.K., 1988.
8. S. Chen, C. Cowen, and P. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–308, March 1991.
9. G. Watson, "Data fitting by positive sums of exponentials," in *Algorithms for Approximation* (J. Mason and M. Cox, eds.), Oxford, U.K.: Clarendon Press, 1987.
10. T. M. Cover and J. A. Thomas, *Elements of Information Theory.* Wiley, 1991.
11. M. D. Richard and R. P. Lippmann, "Neural network classifiers estimate Bayesian *a posteriori* probabilities," *Neural Computation*, vol. 3, pp. 461–483, 1991.
12. R. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. 67, pp. 786–804, May 1979.
13. G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Trans. Patt. Analy. and Mach. Intell.*, vol. PAMI-5, no. 1, pp. 25–39, 1983.
14. R. Chellappa and R. L. Kashyap, "Texture synthesis using 2-D noncausal autoregressive models," *IEEE Trans. Acoustics, Speech, and Signal Process.*, pp. 194–203, February 1985.
15. R. L. Kashyap and P. M. Lapsa, "Synthesis and estimation of random fields using long-correlation models," *IEEE Trans. Patt. Analy. and Mach. Intell.*, vol. PAMI-6, no. 6, pp. 800–809, 1984.
16. A. Gagalowicz and S. D. Ma, "Sequential synthesis of natural textures," *Comp. Vis., Graph., and Img. Proc.*, vol. 30, pp. 289–315, 1985.
17. A. Gagalowicz and C. Tournier-Lasserve, "Third-order model for non-homogeneous natural textures," in *Proc. Int. Conf. on Pattern Recog.*, 1986.
18. P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* New York: Dover, 1966.
19. G. G. Langdon and J. Rissanen, "Compression of black-white images with arithmetic coding," *IEEE Trans. Comm.*, vol. COM-29, pp. 858–867, June 1981.
20. M. Rabbani and P. W. Jones, *Digital image compression techniques.* Bellingham, Washington: SPIE Optical Engineering Press, 1991.
21. J. Rissanen and G. G. Langdon, "Arithmetic coding," *IBM J. Res. Develop.*, vol. 23, pp. 149–162, March 1979.
22. J. Rissanen and G. G. Langdon, "Universal modeling and coding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 12–23, 1981.
23. K. Popat, "Scalar quantization with arithmetic coding," Master's thesis, Dept. of Elec. Eng. and Comp. Science, M.I.T., Cambridge, Mass., 1990.
24. J. H. Shapiro and A. S. Willsky, "Notes for course 6.432, Stochastic Processes, Detection, and Estimation," 1988.