Lloyd Clustering of Gauss Mixture Models for Image Compression and Classification [1]

Anuradha Aiyer

Analogic Corporation, 8 Centennial Drive, Peabody, MA 01960

Kyungsuk (Peter) Pyun

Hewlett-Packard Company, 11311 Chinden Blvd. ms 276, Boise, ID 83714

Ying-zong Huang

Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA

Deirdre B. Obrien Robert M. Gray *

Information Systems Laboratory, Department of Electrical Engineering, 350 Serra Mall, Stanford, CA 94305

Abstract

Gauss mixtures have gained popularity in statistics and statistical signal processing applications for a variety of reasons, including their ability to well approximate a large class of interesting densities and the availability of algorithms such as the EM algorithm for constructing the models based on observed data. We here consider a different motivation and a framework based on the information theoretic view of Gaussian sources as a "worst case" for robust signal compression. Results in high rate quantization theory suggest distortion measures suitable for Lloyd clustering of Gaussian components based on a training set of data. The approach provides a Gauss mixture model and an associated Gauss mixture vector quantizer which is locally robust. We describe the quantizer mismatch distortion and its relation to other distortion measures including the minimum discrimination information and log-likehood distortions. The resulting Lloyd clustering algorithm is demonstrated by example to image vector quantization, texture classification, and North Atlantic pipeline image classification.

Key words: Clustering, compression, quantization, Gauss mixture, statistical classification, segmentation

1 Introduction

Gauss mixtures have played an important role in modeling random processes for purposes of both theory and design. [49,18,17,33,71,37,45,20,14]. Although newly popular, they have been used in signal processing for many decades. For example, linear predictive coded speech (LPC) can be viewed as fitting Gauss mixture models to speech when the autoregressive (AR) models fit to segments of speech are excited by Gaussian residual processes. In this case the synthesized speech becomes a composite Gaussian process and hence locally a Gauss mixture. The most popular means of fitting a Gauss mixture model to data is the EM algorithm, but Lloyd clustering techniques with suitable distortion measures between observed data and resulting model can be used, as was the Itakura-Saito distortion [38] used for fitting AR models to sampled speech waveforms [27]. The Itakura-Saito distortion is an example of a distortion measure based on model fitting techniques of Kullback using relative entropies (Kullback-Leibler numbers, cross entropies) [41] and our approach can be viewed as a two dimensional extension of the speech techniques. The Lloyd algorithm [47], like the later k-means algorithm [48], originally considered only squared error distortion, but was subsequently generalized (see, e.g., [29] for a history). Potential advantages of Lloyd clustering techniques over the traditional EM algorithm are the use of minimum distortion rules for model selection and the formulas describing centroids with respect to the distortion measures, formulas which when combined with quantization theory provide quantitative relations between minimum discrimination information distortion measures and the performance of optimized robust compression systems. Furthermore, Lloyd clustering does not require the assumption that the observed process is in fact a Gaussian mixture, it instead takes the view that an observed training set produced by an arbitrary probability density is to be fit by a Gauss mixture model by minimizing an average distortion between model and data. One of the key properties of Gaussian models is their role as a "worst case" for designing robust compression/source coding problems, a characterization originally developed in an information theory context by Sakrison [60] and subsequently extended to Gauss mixture models using high rate quantization theory [30]. This extension suggests a natural distortion measure for measuring how well Gauss and Gauss mixture models fit observed data.

* Corresponding author.
Email addresses: pyun@hp.com (Kyungsuk (Peter) Pyun), rmgray@stanford.edu (Robert M. Gray).

[1] This work was supported by the National Science Foundation under NSF Grants MIP-9706284-001 and CCR-0073050 and by a gifts from the Hewlett Packard Corporation and Norsk Electro Optikk. Portions of this work were presented at the IEEE 2001 International Conference on Image Processing, 2001 International Conference on Acoustics, Speech, 2002 Signal Processing, Multimedia and Expo, 2004 IEEE Data Compression Conference.

The distortion measure is a variation on Kullback-Leibler distortion measures such as the Itakura-Saito, minimum discrimination information distortion, the log-likelihood distortion, and local Mahalonobis distortion measures. Its motivation derives from a formula for the performance mismatch resulting when a quantizer designed for one source (such as Gaussian or Gauss mixture) is applied to another (perhaps real data to which a Gaussian or Gauss mixture model has been fit). We here describe Lloyd clustering algorithms based on this distortion measure for the design of Gauss mixture models and develop the properties of the resulting models. We provide examples to demonstrate the algorithms and their potential for both compression and statistical classification of images.

2 Quantization

Image compression or coding is the mapping of an analog or high rate digital image into a relatively low rate representation for efficient storage or transmission. Image compression is also useful for reducing the complexity of other signal processing techniques such as classification and segmentation based on the image. We model a compression system as a Shannon source code subject to a fidelity criterion (or vector quantizer) [62,13,24,29]: The information source consists of a random process of vectors $\{X_1, X_2, \dots\}$ in k-dimensional space, $\Re^k$. The vectors may represent, for example, blocks or lines of pixels in a single image or successive image frames in video. We assume that the source has a stationary distribution described by a probability density function (pdf) f for a generic random vector X. The vector X is encoded into a binary code vector $\alpha(X)$ with length $\ell(\alpha(X))$. Without loss of generality this can be considered as a mapping of X into an index $i = \alpha(X)$ which is losslessly encoded using a uniquely decodable code with resulting codeword length $\ell(i)$. The decoder β converts the index into a source reproduction $\hat{X}$. The cost or lack of quality of the reproduction is measured by a distortion $d(X, \hat{X}) = d(X, \beta(\alpha(X))$ and the cost or rate of transmission or storage is measured by the length $\ell(\alpha(X))$. Requiring a uniquely decodable lossless code for the index means that the lengths of the binary vectors must satisfy the Kraft inequality, i.e., if the binary codeword corresponding to index i has length $\ell(i)$, then if the lengths are measured in nats

$$\sum_i e^{-\ell(i)} \leq 1\,. \tag{1}$$

A *length function* ℓ satisfying (1) is said to be *admissible*.

Performance is measured by average distortion, $E\left[d(X, \beta(\alpha(X)))\right]$ and average rate $E\left[\ell(\alpha(X))\right]$. The optimal performance is found by constraining either rate or distortion and minimizing the other performance measure over all quantizers, or by minimizing a Lagrangian combination of the two. We

emphasize the Lagrangian approach, as it has led to the most general theory and design algorithms based on optimally trading off average distortion and rate.[12,24,29]

A quantizer Q is characterized by the triple $\{\alpha, \beta, \ell\}$. The expected Lagrangian distortion for a code Q is $\rho(\lambda, f, Q) = E_f\left(d(X, \beta(\alpha(X))) + \lambda\ell(\alpha(X))\right)$ and the optimal performance for a fixed λ is given by $\rho(f, \lambda) = \inf_Q \rho(\lambda, f, Q)$, where the infimum is over all quantizers Q with admissible length functions.

There are three necessary conditions for a code with components α, β and ℓ to be optimal in the sense of yielding the best possible rate-distortion tradeoff for a given value of λ [47,46,12,24,29]. These three conditions provide an iterative descent algorithm to construct an optimal quantizer — the generalized Lloyd clustering algorithm. As our algorithm is an application of these ideas to a nonEuclidean distortion measure, we recall these properties in their general form: The optimal α for a given β, ℓ is $\alpha(x) = \arg\min_i \left(d(x, \beta(i)) + \lambda\ell(i)\right)$, the optimal β for a given α, ℓ is $\beta(x) = \arg\min_y E[d(X, y)|\alpha(X) = i]$, and the optimal ℓ for a given α, β is $\ell(i) = -\ln \Pr(\alpha(X) = i)$. Iterative application of these three optimality properties to an initial code constitutes the generalized Lloyd design algorithm. Typically the Lloyd algorithm is run on a set of training data so that the expectations become sample averages.

The distortion measure between observed data and probability density functions or models that will be used here is based on high rate quantization theory, the theory describing the asymptotically optimal performance in the high rate or small λ case [70,23,29,31,30]. High rate theory shows that under suitable techincal assumptions the asymptotic performance using the squared error distortion is characterised by

$$\lim_{\lambda\to 0}\left(\inf_Q\left(\frac{E_f[d(X, \beta(\alpha(X)))]}{\lambda} + E_f\ell(\alpha(X))\right) + \frac{k}{2}\ln\lambda\right) = h(f) + \theta_k \qquad (2)$$

where the infimum is over all quantizers with admissible length functions, $\theta_k = \inf_{\lambda>0}\left(\rho(u_1, \lambda)/\lambda + k/2\ln\lambda\right)$, and u_1 is the uniform pdf on the k-dimensional unit cube. Intuitively this means that given a small λ there exists a quantizer which is approximately optimal for this λ: $\rho_\lambda(f, Q) \approx \lambda\theta_k + \lambda h(f) - (k/2)\lambda\ln\lambda$. Only the squared error is considered in the high rate results, but these formulas will prove useful for quantizing models or probability density functions with nonEuclidean distortion measures since we will measure the distance between models by how similar their quantized outputs are.

Since the high rate performance depends on the source pdf f only through its differential entropy, this implies that if one is given any constraints on the pdf, the *worst case* source in the sense of having the worst high rate compression performance will be the source which has the largest differential entropy. Thus, for example, if all that is known about a pdf is its $\mu = E_f X$ and covariance

$K = E_f[(X-\mu)(X-\mu)^t]$, then the pdf with the largest differential entropy is well known to be the Gaussian pdf

$$f(x) = \mathcal{N}(x, \mu, K) = \frac{1}{(2\pi)^{\frac{k}{2}}|K|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^t K^{-1}(x-\mu)\right)$$

with differential entropy [13] $h(f) = (1/2)\ln(2\pi e)^k|K|$, where $|K|$ is the determinant of K.

Not only is the Gaussian the worst case, it also turns out that a code designed for a Gaussian source with a given μ and K will provide the approximately the same performance on the Gaussian source and on any other source with the same mean and covariance. This behavior was defined as *robustness* of a code in the Shannon rate-distortion sense by Sakrison [60] and was extended to the high rate quantization case in [30].

Suppose that for a fixed small λ an optimal code Q_g^* is designed assuming that the source pdf is g so that it has performance $\rho(\lambda, g, Q_g^*)/\lambda \cong \theta_k + h(g) - (k/2)\ln\lambda$. Suppose that the actual source pdf is f and not g, that is, there is a *mismatch* between the pdf used to design the code and the pdf to which the code is applied. Two natural questions arise in this case: What is the resulting performance of the code Q_g^* on f, $\rho(\lambda, f, Q_g^*)$ and how far is this performance from the optimal performance possible for f, $\rho(\lambda, f)$? It can be shown either from heuristic arguments based on Gersho's conjecture [1] or by rigorous application of high rate theory [30] that the answers to these questions are given by

$$\frac{\rho(\lambda, f, Q_g^*)}{\lambda} - \frac{\rho(\lambda, g, Q_g^*)}{\lambda} \cong h(f) - h(g) + I(f||g)$$
$$\frac{\rho(\lambda, f, Q_g^*)}{\lambda} - \frac{\rho(\lambda, f)}{\lambda} \cong I(f||g),$$

where $I(f||g) = \int dx f(x)\ln f(x)/g(x)$ is the relative entropy or Kullback-Leibler divergence between pdfs f and g.

If f and g are known to have means μ_f and μ_g and covariances K_f and K_g, respectively, and g *alone* is assumed to be Gaussian (f can be arbitrary except for the assumed moments), then $h(f) - h(g) + I(f||g) = -(k/2) + (1/2)\,\mathrm{Trace}\, K_g^{-1}K_f - (\mu_f - \mu_g)^t K_g^{-1}(\mu_f - \mu_g)$. Thus if we choose equal means and covariances, $\mu_g = \mu_f = \mu$ and $K_f = K_g = K$, then $h(f) - h(g) + I(f||g) = 0$ and hence indeed the performance of the code designed for g asymptotically equals that when the code is applied to f and the performance loss of the mismatched code for f to the optimal code for f is $I(f||g) = (1/2)\ln(2\pi)^k|K_f| - h(f)$, the difference between the actual differential entropy of f and its maximum value.

A single Gaussian source provides both a worst case and a robust design for a source whose second order properties are known. Code design based on a single Gaussian, however, can be overly conservative. Speech coding tradition suggests that instead of designing a code based on a single Gaussian model, a collection of Gaussian models can be used by fitting a distinct Gaussian model to each segment of speech (an autoregressive model matching the local covariance behavior) and designing a separate code for each model. For each input vector a decision is first made about which code (or model) is best, and then that code is used on the current input vector. Current codebook excited linear prediction (CELP) techniques can be viewed as having this structure.

Again let f be the "true" pdf on Euclidean space $\Re^k$. Consider a finite partition of this space, $\mathcal{S} = \{S_m;\ m \in \mathcal{J}\}$, where $\mathcal{J} = \{1, \ldots, M\}$. For the moment the partition is assumed to be arbitrary except for the requirement that the cells S_m have nonzero probability, $P_f(S_m) > 0$ for all m. Given this partition, we can consider f to be a mixture source of the form $f(x) = \sum_m p_m f_m(x)$, where the f_m are the conditional pdfs $f_m(x) = f(x)/p_m$ for $x \in S_m$ and 0 otherwise, where $p_m = P_f(S_m)$.

We consider a form of classified vector quantizer [57] or composite code: we design a separate code, say Q_m, for each f_m, and then apply a two-step coding procedure. The first step is to estimate which component is in effect, which here is simply a question of determining which partition cell S_m contains the input vector. The second step is to then apply the corresponding code, Q_m. The idea is that the worst-case/robustness results can be applied to each of the component pdfs f_m, allowing for a *locally worst case and robust* design.

Assume that we have a collection of model pdf's $\{g_m\}$ on $\Re^k$. Each g_m can be thought of as a design model for f_m, but it is important to note that g_m *need not have the same support as* f_m, and we choose it to be Gaussian for robustness of the resulting code. Let $\mathcal{M}$ denote the class of all nonsingular Gaussian pdfs of dimension k and hence $g_m \in \mathcal{M}$ for all m. We assume a fixed (and small) λ and that for each m we have a code $Q_m = (\alpha_m, \beta_m, \ell_m)$ optimized for λ and g_m, that is, $\rho(\lambda, g_m, Q_m) \cong \rho(\lambda, g_m)$.

The overall code Q operates as follows: If $x \in S_m$, then $\alpha_m(x) = i$ is used to provide an index indicating which codeword $\beta_m(i)$ is to be used for the reproduction and that its cost (length) is $\ell_m(i)$. For the decoder to know which code to use, however, we assume another length function $L(m)$ indicating how many nats are required to specify m to the decoder, where L must also satisfy the Kraft inequality. To summarize, the overall code $Q = (\alpha, \beta, \ell)$ is defined by $\alpha(x) = (m, \alpha_m(x))$ if $x \in S_m$, $\beta(m, i) = \beta_m(i)$, $\ell(m, i) = L(m) + \ell_m(i)$. The overall performance of the composite quantizer on the full pdf f is given by

[30]

$$\frac{\rho(f,\lambda,Q)}{\lambda} - h(f) = \sum_m p_m \left(\frac{E_{f_m} d(X, \beta(\alpha(X)))}{\lambda} + E_{f_m} \ell(\alpha(X)) - h(f_m) \right) - H(Z)$$

where Z has distribution $\Pr(Z = m) = \Pr(X \in S_m) = p_m$.

By construction, the length function $\ell(m,i) = L(m) + \ell_m(i)$ and with the optimal choice of $L(m) = -\ln p_m$, the average code length of the composite quantizer is $EL(Z) + \sum_m p_m E_f \ell_m(\alpha_m(X)) = H(Z) + \sum_m p_m E_f \ell_m(\alpha_m(X))$. With this choice we have for the composite quantizer Q that

$$\frac{\rho(f,\lambda,Q)}{\lambda} - h(f) = \sum_m p_m \left(\frac{\rho(f_m,\lambda,Q_m)}{\lambda} - h(f_m) \right) .$$

Since each quantizer Q_m is optimized for a Gaussian model g_m and applied to the pdf f_m, we have from the single Gaussian case that the quantizer mismatch will be approximately

$$\begin{aligned} &\frac{\rho(\lambda, f_m, Q_m)}{\lambda} - \frac{\rho(\lambda, g_m, Q_m)}{\lambda} \\ &\cong h(f_m) - h(g_m) + I(f_m || g_m) \\ &= -\frac{k}{2} + \frac{1}{2} \operatorname{Trace} K_{g_m}^{-1} K_{f_m} + (\mu_{f_m} - \mu_{g_m})^t K_{g_m}^{-1} (\mu_{f_m} - \mu_{g_m}) \end{aligned}$$

which will be zero if the Gaussian moments are chosen to match those of f_m.

Continuing to assume matched moments, application of the mismatch theorem yields the high rate approximation

$$\frac{\rho(f,\lambda,Q)}{\lambda} - \frac{\rho(f,\lambda)}{\lambda} \cong \sum_m p_m I(f_m || g_m) . \tag{3}$$

Unlike the single component Gaussian, we have the possibility of minimizing this mismatch by judicious choice of the partition $\mathcal{S}$, which, as we shall see, can be accomplished using the Lloyd clustering algorithm

3 Distortion Measures for Clustering Gauss Mixtures

The Quantizer Mismatch Distortion

We seek a collection $\mathcal{G} = \{g_m\}$ of pdf's from an allowed collection of Gaussian pdf's and a partition $\mathcal{S} = \{S_m\}$ of $\Re^k$ which minimizes the overall mismatch

defined by $\overline{I}_f = \inf_{\mathcal{S},\mathcal{G}} \overline{I}_f(\mathcal{S},\mathcal{G})$, where $\overline{I}_f(\mathcal{S},\mathcal{G}) = \sum_m P_f(S_m) I(f_m||g_m)$. This minimization can be solved by clustering and, in fact, posed as a quantization problem with an encoder $a : \Re^k \to \mathcal{J}$ described by the partition $\mathcal{S} = \{S_m\}$ by $a(x) = m$ if $x \in S_m$, $m \in \mathcal{J}$, and a decoder $b : \mathcal{J} \to \mathcal{M}$ defined by $b(m) = g_m$.

Given an encoder index m corresponding to encoder cell S_m, the best possible g_m in terms of minimizing the mismatch is the Gaussian solution to $g_m = \arg\min_{g \in \mathcal{M}} I(f_m||g)$, which has been shown to be the Gaussian source with the same mean and covariance as f_m. With this decoder the minimum mismatch problem becomes

$$\overline{I}_f = \inf_{\mathcal{S}} \sum_m P_f(S_m) \min_{g \in \mathcal{M}} I(f_m||g).$$

To describe the encoder and performance requires a distortion measure, and this we choose in a way that minimizing average distortion is equivalent to finding the minimum mismatch. Consider the distortion defined by $d_I(x,m) = \ln(f(x)/g_m(x)) + L(m)$, where L is an admissible length function. d_I is not a distortion in the strict sense since it need not be nonnegative, but its average with respect to f is nonnegative from the divergence inequality and it meets all of the requirements of the Lloyd algorithm. The optimal encoder is a minimum distortion encoder and hence for a given decoder codebook $\mathcal{G}$ $a(x) = \arg\min_m d_I(x,m) = \arg\min_m \left(L(m) - \ln g_m(x)\right)$. The corresponding encoder partition $\mathcal{S}$ will then yield average distortion

$$\int dx f(x) d_I(x, a(x)) = \sum_m p_m \left(L(m) + \int_{S_m} dx f_m(x) \ln \frac{f_m(x) p_m}{g_m(x)} \right)$$

where as before $p_m = P_f(S_m)$ and $f_m(x) = f(x)/p_m$ for $x \in S_m$. Thus

$$\begin{aligned} \int dx f(x) d_I(x, a(x)) &= \sum_m p_m I(f_m||g_m) + \sum_m p_m \ln \frac{p_m}{e^{-L(m)}} \\ &\geq \sum_m p_m I(f_m||g_m) \end{aligned}$$

with equality if and only if we choose the optimal length function $L(m) = -\ln p_m$. Thus if we choose an optimal decoder and length function for a partition, *the average distortion according to d_I is exactly the mismatch which we are trying to minimize*. Thus iterating the Lloyd optimality properties of optimizing encoder, decoder, and length function can only decrease average distortion and hence also the mismatch.

When using individual Gaussian models with optimal codebooks and length functions, the distortion takes on the form

$$d_I(x,m) = \ln f(x) - \ln p_m + \frac{1}{2}\ln\left((2\pi)^k |K_m|\right) + \frac{1}{2}(x-\mu_m)^t K_m^{-1}(x-\mu_m)$$

where μ_m and K_m are the mean and covariance of the Gaussian pdf g_m.

The $\ln f(x)$ term has no effect on the encoder a, that is, on the minimum distortion match of a model from the codebook to an input vector. Likewise the additive constant terms have no effect on the encoder. Distortion measures are *equivalent* for quantization if they yield the same encoder. Thus we define the *quantizer mismatch* or *QM* distortion by

$$\begin{aligned} d_{\mathrm{QM}}(x,m) &= \ln \frac{1}{g_m(x)} - \ln p_m \\ &= \frac{1}{2}\ln|K_m| + \frac{1}{2}(x-\mu_m)^t K_m^{-1}(x-\mu_m) - \ln p_m. \end{aligned} \tag{4}$$

This is no longer a 'distortion measure' in the strict sense of rate-distortion theory because in is not a nonnegative function, but it can be made nonnegative and it will yield the same encoder and decoder when used in a quantization or source coding application. The first two terms of (4) involve only the shape of the model pdf. Distortion measures equivalent to the first term have been used in clustering with names like "maximum likelihood" or "log likelihood" distortion since minimizing this distortion over m for a given x is equivalent to choosing the maximum likelihood estimate for m assuming the vector was actually produced by one of the models g_m [28,1,32]. If the pdf f is retained, this can also be interpreted as a log likelihood ratio distortion measure. Furthermore, with the optimal length function providing the $\ln p_m$ term, minimizing the distortion is equivalent to a maximum a posteriori selection of a Gauss model from a collection of Gauss models g_m with a probability mass function p_m, i.e., the MAP selection of which Gauss component of a Gauss mixture is in effect if the unknown source is in fact a Gauss mixture source.

We argue that this distortion measure provides a meaningful measure of the distortion between an input vector x and a "model" $\{g_m, p_m\}$ consisting of a pdf together with a probability of the pdf being in effect. The intuition is that if one has a collection of models together with a prior (and hence a mixture model) which yields a small average distortion with respect to this distortion measure, then a composite quantizer designed using the mixture components will provide a good code for the true underlying source, where "good" here has the double meaning that the code will be robust and have performance as close as possible to the optimal performance for the unknown pdf f.

We shall see that the Lloyd algorithm for minimizing mismatch produces a collection of Gaussian models g_m together with a probability mass function, p_m. A collection of pdf's together with a pmf can be viewed as a mixture and hence the algorithm can be viewed as a means of fitting mixtures of specified

families of densities to an arbitrary pdf. Thus the preceding development provides a Lloyd clustering algorithm for the design of Gauss mixture models, an algorithm which can be viewed as quantizing the space of Gaussian models.

Autoregressive Models

If further structure is imposed on the Gauss components of the mixture, then the QM distortion can be approximated in a way that can reduce the computational complexity of the encoder. This leads to an example of the closely related minimum discrimination information (MDI) distortion implicit in Kullback [41] which in speech coding applications yields the highly successful Itakura-Saito distortion [27]. Unfortunately, dealing with images or random fields means more complicated analysis than the corresponding results for speech. We constrain our models to be shift-invariant (spatially stationary) recursible autoregressive (AR) models as developed by Lev-Ari et al. [44]. (See in particular Sections II and IV of [44]). This is a class of two-dimensional AR models with properties very similar to their one-dimensional counterparts.

Consider the random field representation of a random vector $X = \{X_n; n \in \mathcal{I} = \mathcal{Z}_J^2\}$, $\mathcal{Z}_J = 0, 1, \ldots, J-1$; that is, the random variables X_n are indexed by a two-dimensional integer-valued vector n. The random vector dimension is $k = J^2$. The mean of X is $\mu = EX$ and the covariance $K = E[(X - \mu)(X - \mu)^t]$, where μ is no longer a one-dimensional vector and K no longer a two-dimensional matrix because of the vector indexing, both can be viewed as (mixed) tensors. Both can be put into vector form by imposing an ordering ϕ on the index set $\mathcal{I}$. In fact, the tensors can be vectorized in many ways. Thus tensor equations can be transformed into vector/matrix equations. We will usually not differentiate between the two notations unless necessary. Note that since all distinct orderings are simply permutations, some operations (like determinants) will not depend on the ordering chosen.

An AR model is specified by a pair $A = \{a_{t,s};\ t \in \mathcal{I}, s \in \mathcal{I}\}$, where $a_{t,t} = 1$ for $t \in \mathcal{I}$, $\Sigma = \mathrm{diag}(\sigma_t^2;\ t \in \mathcal{I})$. The *support set* of the model is $D = \{(t, s) : t \in \mathcal{I}, s \in \mathcal{I}, a_{t,s} \neq 0.\}$. We consider only finite order models, that is, the support set is finite. The corresponding AR model takes the form

$$X_n = W_n - \sum_{s:(n,s)\in D, s\neq n} a_{n,s} X_s;\ n \in \mathcal{I}, \tag{5}$$

where the W_n are independent, identically distributed random variables with mean μ_W and variance σ_W^2. An AR model is shift-invariant or stationary if $a_{t,s} = a_{t-s}$ for all $(t, s) \in D$. In particular following (22) of [44] we assume $D = \{(t, s) : t \in \mathcal{I}, s \in \mathcal{I}, t - s \in \Delta\}$, where Δ is a fixed set of nonnegative integer pairs. (This corresponds to a causal model. Generalizations are considered in

[44].)

An AR model is recursible if there is a permutation of the index set $\mathcal{I}$ such that the resulting matrix A corresponding to the tensor A is lower triangular.

Lev-Ari et al. [44] derive a variety of useful properties of shift-invariant recursible AR models for the case where $\mathcal{I}$ is the set of all integers, in which case the AR process has a covariance function that is shift invariant. We will be interested in the finite case, but the following properties will still be helpful.

(1)
$$K(n,n) = \sigma_n^2 = \sigma^2;\ n \in \mathcal{I} \tag{6}$$

(2) The covariance and regression coefficients A are related by

$$\sum_{n\in\Delta} a_n K(l-n) = \sigma^2 \delta_{l,0},\ l \in \Delta. \tag{7}$$

A key point is that given a shift invariant covariance K, the corresponding AR coefficients can be found by solving the above linear equations, which are just the multidimensional version of the Yule-Walker or normal equations as noted in [44] and can also be considered as the standard linear prediction or maximum entropy formulas. Furthermore, not all of the covariance coefficients are needed, only those values $K(t,s)$ in the "band" $B = \{(t,s) : t - s \in \Delta\}$.

These results yield the structure of the inverse covariance K^{-1}, the operator satisfying

$$\sum_i K(t,i)K^{-1}(i,s) = \delta_{t,s}. \tag{8}$$

In particular, direct substitution into (7) and the fact that $a_n = 0$ for $n \notin \Delta$ verifies that the inverse covariance is given by the Toeplitz operator

$$K^{-1}(t,s) = \begin{cases} \frac{1}{\sigma^2}\sum_{l\in\Delta} a_l a_{l-(t,s)} & (t,s) : t-s \in \Delta \\ 0 & \text{otherwise} \end{cases}. \tag{9}$$

In particular, the inverse covariance K^{-1} has nonzero entries only in this same band; that is, $K^{-1}(t,s) = 0$ unless $t - s \in \Delta$ [44].

Lastly, the mean is easily found to be a constant:

$$\overline{\mu} = \mu_W \sum_{t\in\cdot} a_t. \tag{10}$$

We now make a stronger assumption then done in [44]: we assume that the order of the autoregressive model is a small number M in the sense that maximum integer component of any vector in Δ is $M << k$, the vector or random field dimension. In speech modeling, M is typically 10. In our

more complicated image application we will take $M = 1$ – a first order two-dimensional recursible AR model. In particular, in our simple case we will take $\Delta = \{(0,0),(0,1),(1,0),(1,1)\}$. With an eye towards modeling finite images, we also now consider $\mathcal{I}$ to be $\{0,1,2,\ldots,K\}$. Now we cannot have both a shift-invariant AR model and a shift-invariant covariance because of the edge effects, but $K^{-1}(t,s)$ will depend only on the vector difference $t-s$ except for the "edges" where t or s has components between 0 and $M-1$. In particular, for all $(t,s) \in\in \Delta \in D$ except where at least one of the components of t or s is less than the model order, (9) will hold. In particular, the operator $K^{-1}(t,s)$ is not Toeplitz, but it is approximately Toeplitz (asymptotically Toeplitz as $J \to \infty$).

Similarly, in the finite extent case, the mean is a constant vector except near the edge, that is, when one of the components of the index is smaller than M.

Combining the above ideas yields the the approximation

$$\begin{aligned}(x-\mu)^t K^{-1}(x-\mu) &= \sum_{(i,j)\in D} (x(i)-\overline{\mu})(x(j)-\overline{\mu})K^{-1}(i,j) \\ &\cong \sum_{n\in\Delta} K^{-1}(n) \times \sum_{i,j:i-j=n} (x(i)-\overline{\mu})(x(j)-\overline{\mu})).\end{aligned}$$

The rightmost term can be recognized as a sample average estimate of the covariance of the underlying shift invariant random field X_i about a constant vector $\overline{\mu}$ (possibly not the process mean) for a vector lag n. If we define an estimate

$$\hat{K}_{x,m}(n) = \frac{1}{N(n)} \sum_{i,j:i-j=n} (x_i - m)(x_j - m);\ n \in \mathcal{I}^2 \tag{11}$$

where $N(n) = \#\{i,j : i-j=n\}$, then the approximation becomes

$$\begin{aligned}(x-\mu)^t K^{-1}(x-\mu) &= \sum_n N(n) K^{-1}(n) \hat{K}_{x,\overline{\mu}}(n) \\ &= \sum_{i,j} K^{-1}(i,j) \hat{K}_{x,\overline{\mu}}(i,j),\end{aligned}$$

so that the quantizer mismatch distortion between an input x and a model (μ_m, K_m, p_m) can be approximated as

$$\begin{aligned}d_{\mathrm{QM}}(x,m) &= \frac{1}{2}\ln|K_m| + \frac{1}{2}(x-\mu_m)^t K_m^{-1}(x-\mu_m) - \ln p_m \\ &\cong \frac{1}{2}\ln|K_m| - \ln p_m + \frac{1}{2}\sum_{i,j\in D} K_m^{-1}(i,j)\hat{K}_{x,\overline{\mu}}(i,j).\end{aligned} \tag{12}$$

It is important to note that *only the estimates $\hat{K}_{x,m}(n)$ for $n \in \Delta$ are required.*

The final form of the approximation (eq:mdi) is almost the same as the minimum discrimination information distortion between an observed x and the model (μ_m, K_m, p_m), which is given by [41,27,32]

$$d_{MDI}(x,(\mu_m,K_m)) = \frac{1}{2}[\log\frac{|K_l|}{|\hat{K}_x|} + \sum_{i,j\in\mathcal{I}} K_l^{-1}(i,j)\hat{K}_{x,\mu_l}(i,j) - k],$$

where $\hat{K}_x = \hat{K}_{x,\hat{\mu}}$, the sample covariance about the sample mean $\hat{\mu} = (1/k)\sum_{i\in\mathcal{I}} x_i$, so that

$$\begin{aligned} d_{\mathrm{QM}}(x,(m_l,K_l)) &\cong \ln \mid K_l \mid + \sum_{i,j} K_l^{-1}(i,j)\hat{K}_{x,m_l}(i,j) \\ &= d_{\mathrm{MDI}}(x,(m_l,K_l)) + \ln|\hat{K}_x| + k - \log p_m \end{aligned}$$

The log determinant term depends only on the input x, hence it makes no difference to the encoder. Similarly, the constant term has no effect. The $\log p_m$ term can be viewed as a Lagrangian distortion term with multiplier equal to 1, or it can be eliminated by assigning the models equal prior probabilities, effectively using a fixed-rate model codebook. Thus the MDI distortion and the QM distortion can be viewed as approximately equivalent distortion measures. The MDI has a different structure and hence can yield simpler implementations, as is the case in speech coding. We consider both forms in our simulations. We also observe that other forms of covariance structure can be imposed on the allowed Gaussian components, which may be useful in some problems. For examples of such structured covariances, see,e.g., [44], who treat the maximum entropy formulation which is a special case of the minimum discrimination information formulation.

4 Centroids

To design a codebook using the Lloyd clustering algorithms requires the computation of the centroids with respect to a fidelity criterion. For the QM distortion, these were derived in [32,30]. For the MDI distortion the derivation is a natural extension of the corresponding result for one-dimensional speech [27] and can be found in [32,26]. The results are summarized here for convenience. Here the expectation is with respect to the empirical distribution of a training set of data, i.e., the conditional expectations are conditional sample averages.

The goal is to find m_l and K_l to minimize the conditional average distortion

$$E[d_{\mathrm{QM}}(X, g_l) \mid \alpha(X) = l] = E[\ln |K_l| + (X - m_l)^t K_l^{-1}(X - m_l)|\alpha(X) = l]$$

The optimal mean regardless of the covariance is given by $m_l = E[X \mid \alpha(X) = l]$ since this choice minimizes the quadratic term in the mean as 0 (the centroid with respect to a weighted quadratic measure is the mean).

Define the average $\overline{K}_l = E[(X - m_l)(X - m_l)^t]$. Then

$$\begin{aligned} E[d_{\mathrm{QM}}(X, g_l) \mid \alpha(X) = l] &= [\ln \frac{|K_l|}{|\overline{K}_l|} + \mathrm{Tr}(K_l^{-1}\overline{K}_l) - k] + k + \ln |\overline{K}_l| \\ &\geq k + \ln |\overline{K}_l| \end{aligned}$$

with equality if $K_l = \overline{K}_l = E[(X - m_l)(X - m_l)^t \mid \alpha(X) = l]$, so that centroids are computed by averaging.

For the MDI distortion we must minimize the conditional expectation

$$\begin{aligned} E[d_{\mathrm{MDI}}(X, g_l) \mid \alpha(X) = l] = \frac{1}{2} E[\ln \frac{|K_l|}{|\hat{K}_X|} &+ \mathrm{Tr}(\hat{K}_X K_l^{-1}) \\ &+ (\hat{m}_X - m_l)^t K_l^{-1}(\hat{m}_X - m_l) - k \mid \alpha(X) = l] \end{aligned}$$

where $\hat{m}_X$ and $\hat{K}_X$ are the mean and the covariance estimates for observation X.

The mean centroids are given by $m_l = E[\hat{m}_X \mid \alpha(X) = l]$ regardless of K_l as before. With this choice of m_l we need K_l to minimize

$$\begin{aligned} &E[\ln \frac{|K_l|}{|\hat{K}_X|} + \mathrm{Trace}(\hat{K}_X K_l^{-1}) - k \mid \alpha(X) = l] \\ &= \ln \frac{|K_l|}{|\overline{K}_l|} + \mathrm{Trace}(\overline{K}_l K_l^{-1}) - k + E[\ln \frac{|\overline{K}_l|}{|\hat{K}_X|} \mid \alpha(X) = l] \\ &\geq E[\ln \frac{|\overline{K}_l|}{|\hat{K}_X|} \mid \alpha(X) = l] \end{aligned}$$

with equality if $K_l = \overline{K}_l$ (since the first three terms are just the Kullback-Leibler distortion between two Gaussian distributions with the given covariances and 0 means). Since the end goal is an AR model, this computation is simpler than it seems. All that is needed is the conditional average of the covariance estimates for the multidimensional "lags" $n \in \Delta$, in our case of a first order model this is just four numbers. These numbers are then used in the normal equations to derived the centroid AR coefficients.

In theory the centroid step can result in a singular covariance matrix, which means the resulting Gaussian component will have a singular density. This can be avoided by means of the standard statistical trick of regularization, e.g., forming a final update $\overline{K}_m = \alpha K_m + (1-\alpha)\overline{K}$ for $\alpha \in (0,1)$ and

$$\overline{K} = \frac{1}{N-L} \sum_{l=1}^{L} \sum_{x_j \in S_l} (x_j - \mu_l)(x_j - \mu_l)^t.$$

5 Gauss Mixture Design

There is extensive literature for designing Gauss mixture models, with the EM algorithm based on maximum likelihood and its extensions [15,58,20] being the most popular. Both the EM algorithm and the Lloyd clustering approach proposed here operate on a training or learning set $\{x_1, x_2, \ldots, x_N\}$ and produce a finite Gauss mixture $\{g_m, p_m\}$. The Lloyd clustering design goal is to find a Gauss mixture to minimize

$$\frac{1}{N} \sum_{n=1}^{N} d_{\mathrm{QM}}(x_i, a(x_j)) = \frac{1}{N} \sum_{n=1}^{N} \min_m \left(-\ln p_m - \ln g_m(x_i) \right), \tag{13}$$

where we focus on the QM distortion with the understanding that the MDI distortion is also applicable with the obvious modifications. In contrast, the EM algorithm seeks a minimum of $\prod_{i=1}^{N} \sum_{m=1}^{M} p_m g_m(x_j)$, or, equivalently, a minimum of

$$\sum_{i=1}^{N} \ln \left[\frac{1}{\sum_{m=1}^{M} p_m g_m(x_j)} \right].$$

The EM algorithm assumes that each training vector is in fact a sample from the Gaussian mixture. The Lloyd algorithm assumes that each training vector is a sample from one of a collection source components that is to be modeled by an individual Gaussian component, but no claim is made to the effect that the unknown source in fact is a Gauss mixture.

Lloyd Clustering Algorithm Initialization Begin with an initial codebook of Gaussian models $\{g_m^{(0)};\, m = 1, \ldots, M\}$ described by mean vectors $\{\mu_m^{(0)}\}$ and covariance matrices $\{K_m^{(0)}\}$. This can be accomplished using ordinary MSE Lloyd and then using the sample means and covariances for each index. Set iteration number $n = 1$ and set D_0 to the average distortion resulting from the initial codebook. Pick a convergence threshold ϵ.

Minimum Distortion Encoder Encode each training vector x_i into the index i using the optimal encoder $\alpha^{(n)} = \arg\min_m d_{\mathrm{QM}}(x_i, g_m^{(n-1)})$, the quantizer mismatch distortion of (4).

Centroid Decoder Update the models in the codebook. The mean μ_m of the codeword with index m is the conditional expectation of all training vectors which were mapped into m in the previous step. The covariance is then the corresponding conditional average covariance with respect μ_m. Since the averages are computed with respect to the sample distribution,

$$\mu_m{}^{(n)} = \frac{1}{N_m} \sum_{i:\alpha(i)^{(n)}=m} x_i$$

where N_m is the number of training vectors for which $a^{(n)}(i) = m$. Similarly,

$$K_m^{(n)} = \frac{1}{N_m} \sum_{i:\alpha^{(n)}(i)=m} (x_i - \mu_m^{(n)})(x_i - \mu_m^{(n)})^t \,.$$

Optimal Length Function $\ell(m) = \ln p_m^{(n)}$, where $p_m^{(n)} = N_m/N$ if $N_m > 0$. If $N_m = 0$, remove the cell from the code and reduce M by 1. If a fixed rate code is used, this step is skipped and the p_m are assumed to be equally likely.

Test Compute the average distortion D_n with the new code. If $(D_{n-1} - D_n)/D_{n-1} < \epsilon$, quit. Otherwise go to the minimum distortion step and continue.

In practice the sample average mean, covariance, and counts for each cell are computed on the fly as each training vector is encoded. In addition, the test for the current iteration is actually computed during the minimum distortion step of the next iteration.

To illustrate the operation of the Lloyd clustering algorithm, consider the following toy problem. Example: GMVQ design of GM for dataset. Assume that the input process is a two-dimensional Gaussian mixture with two components: $m_1 = (0,0)^t$, $K_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $p_1 = 0.8$

$m_2 = (-2,2)^t$, $K_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1.5 \end{bmatrix}$, $p_2 = 0.2$. The process is depicted in Figure 5.

The training sequence is randomly generated from the Gauss mixture. Figure 5 depicts the training set and the initial code. Figure 5 depicts the Lloyd generated Gauss mixture models from initial codebook to convergence, and Figure 5 shows the QM distortion as a function of the iteration number.

EM Algorithm Let θ denote the complete parameter set of a Gaussian mixture density, namely $\{p_m, \mu_m, K_m; m = 1, \ldots, M\}$. Given N independent, identically distributed samples, $\{x_1, x_2, \ldots, x_N\}$ let $\mathcal{L}$ denote the log-likelihood

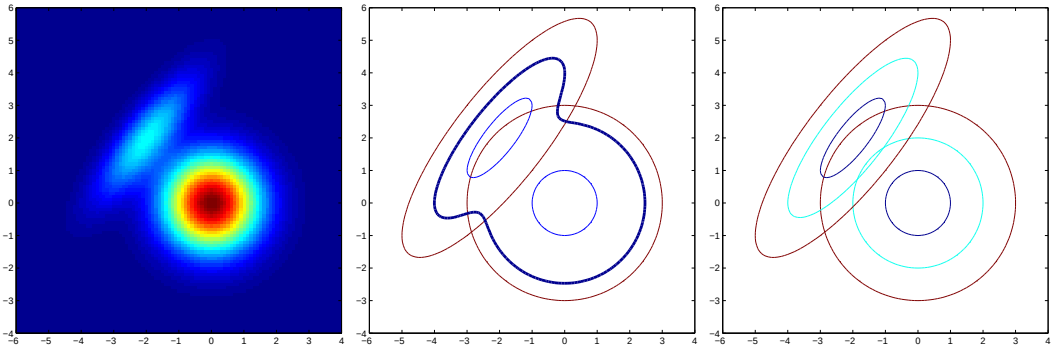


Fig. 1. Densities for components 1 and 2 and the mixture

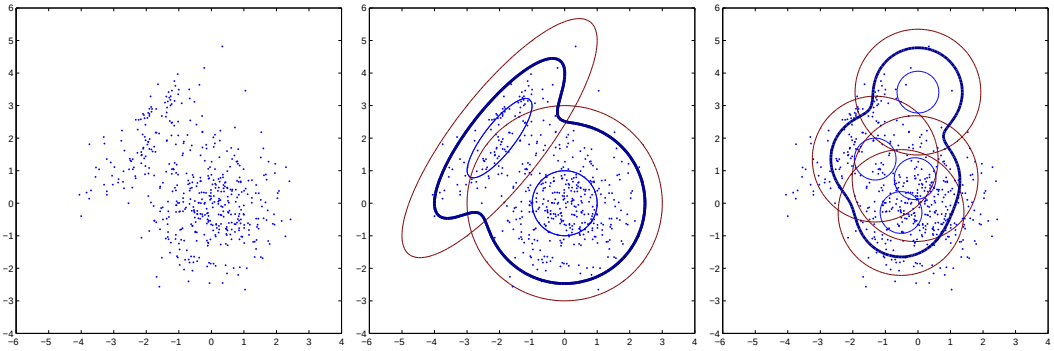


Fig. 2. The training data, the training data with the true mixture density superimposed, and the initial codebook

function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \ln \left(\sum_{m=1}^{M} p_m g_m(x_i) \right).$$

The EM algorithm is an iterative algorithm for improving the log-likelihood:

$$p_i^{(n+1)} = \frac{1}{N} \sum_{j=1}^{N} \nu_i^{(n)}(j) \quad , \quad \mu_i^{(n+1)} = \frac{\sum_{j=1}^{N} \nu_i^{(n)}(j) x_j}{\sum_{j=1}^{N} \nu_i^{(n)}(j)}$$

$$K_i^{(n+1)} = \frac{\sum_{j=1}^{N} \nu_i^{(n)}(j)(x_j - \mu_i^{(n+1)})(x_j - \mu_i^{(n+1)})^t}{\sum_{j=1}^{N} \nu_i^{(n)}(j)}$$

$$\nu_i^{(n)}(j) = \frac{p_i^{(n)} g_m^{(n)}(x_j)}{\sum_{l=1}^{M} p_l^{(n)} g_m^{(n)}(x_j)}$$

In the EM algorithm, a training vector has a probability of belonging to each mixture component, and these probabilities add to unity. In the Lloyd algorithm, a training vector is associated with exactly one mixture component. Thus the EM algorithm makes "soft" decisions about component membership, whereas the Lloyd algorithm makes "hard" decisions. The Lloyd algorithm literally quantizes the input vector space and assigns to each cell a Gaussian

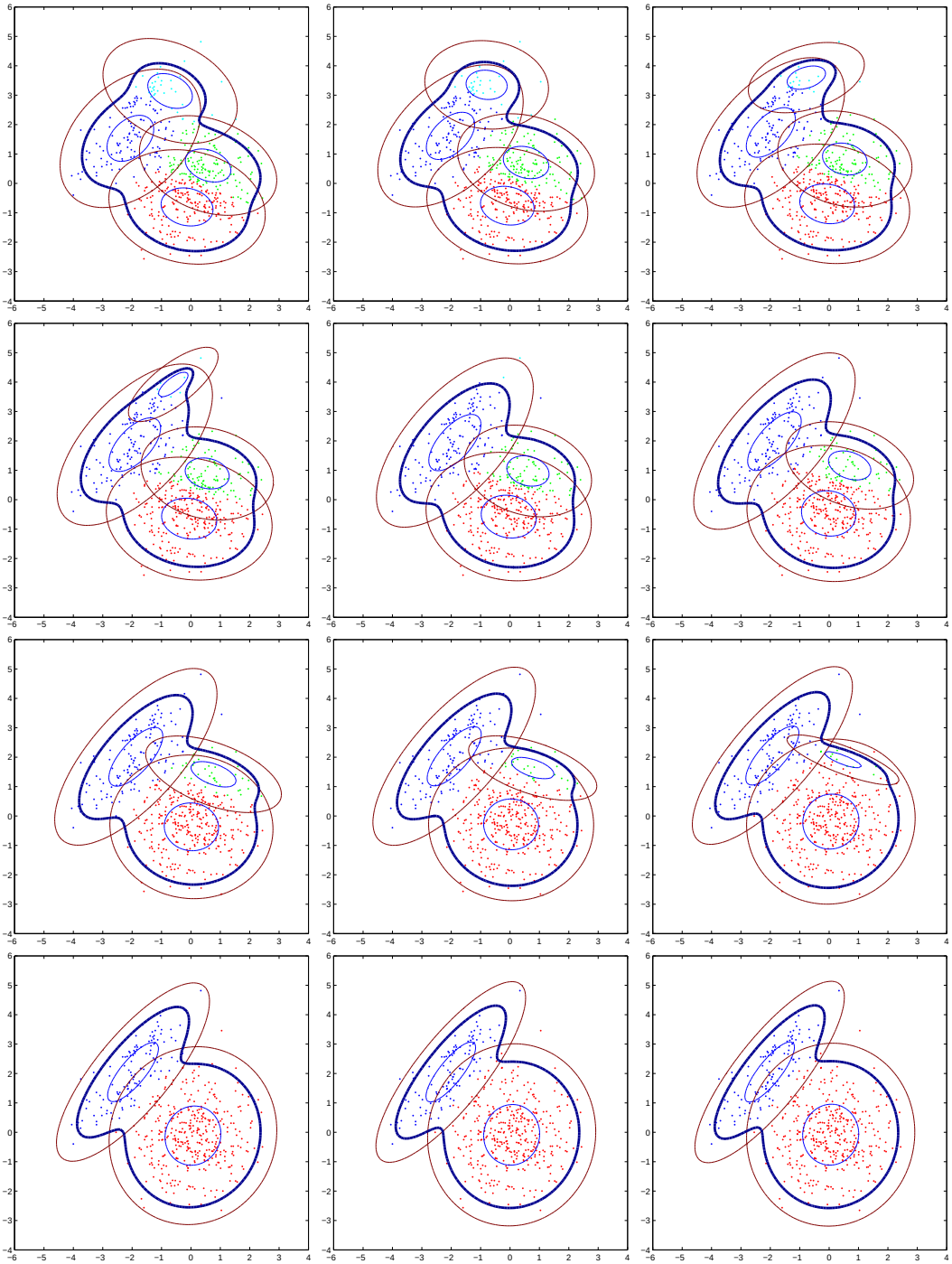


Fig. 3. The Lloyd generated Gauss mixture model from first iteration to convergence

component with matching mean and covariance. The associated probability is the probability of the cell. For those familiar with EM, this makes Lloyd seem counter-intuitive. For example, the set of training vectors assigned to a particular Gaussian component will lie in a Voronoi region that may be *bounded*, but the vectors generated by a Gaussian are unbounded. Thus it is possible for a Gaussian component to generate a training vector that has no chance

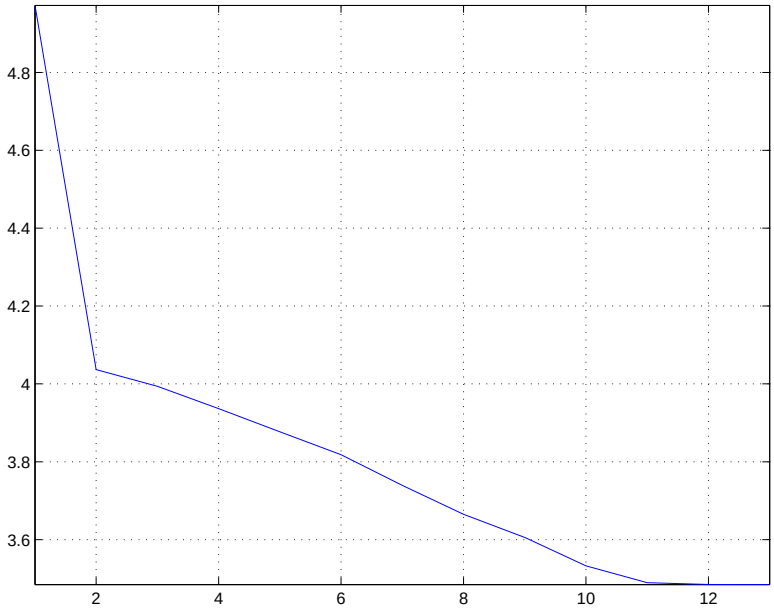


Fig. 4. Lagrangian distortion vs. iteration

of being assigned to it. Using the Lloyd algorithm will not yield a maximum-likelihood model, but that is not the intent. Nonetheless, the model generated from a Lloyd-based algorithm will look similar to the model based on the EM algorithm.

A key issue in Gauss mixture design is the number of mixture components M. The larger the number of components, the greater is the possibility to describe the fine structure of the underlying data distribution. On the other hand, with a high degree of freedom in the estimation, there is an obvious risk of overfit where the estimated model significantly reflects random properties associated with the data. Thus, the number of mixture components must be curtailed. Much work has been done to estimate the number of components in the EM algorithm [3,21,2,61,39,66]. Many of the algorithms incorporate a penalty term along with the log-likelihood function [20]. On the other hand, the Lloyd algorithm is typically begun with a reasonably large number and the algorithm prunes cells that it does not need. While there is clearly no inference of an optimal final number of cells, the structure of the Lloyd algorithm guarantees at least a locally optimum choice, i.e., cells are pruned only when the result in a reduction of the Lagrangian average distortion.

The EM algorithm is designed to provide an approximately maximum likelihood estimate for a Gauss mixture density when the data being observed in fact is produced by a Gauss mixture density. Lloyd clustering takes an alternative approach in that it makes no assumption about the actual data which produced the observation, but it tries to fit a Gauss mixture model to the data by trying to find a collection of Gaussian models and a mapping from input vectors into a model that is "best" in the sense of minimizing a prescribed

average distortion measure. The distortion measure itself is a measure of how great the mismatch is between the model being considered and the data being observed in the sense of how badly a quantizer designed for the model will perform on the data. In a coding application with a collection of available codes, one needs a rule for selecting which code is best for the observed data. The Lloyd approach specifically chooses the Gaussian components (which with the data imply a probability mass function on the components and hence an overall Gauss mixture) precisely so as to optimize the overall composite code. In other words, the resulting reconstructed process will look as much like the original process as possible. Lloyd optimization aims to construct a Gauss mixture process that will produce waveforms that closely resemble the original process (which is in general not a Gauss mixture). EM design aims to provide an approximately maximum likelihood estimate of the Gauss mixture model assumed to be producing the observed data.

6 Classification using GM Models

Statistical classification techniques provide an important application of density estimation in general and of Gauss mixture modeling in particular. [20,14,18,17,49,33,71,45] The components of the mixture may be imagined to represent classes, or classes may each consist of a collection components or a separate mixture. In this section we compare the EM and Lloyd approaches to the design of Gauss mixture models by using each to provide the models used in a classification algorithm and compare the resulting performance.

Plug-in Classification

One approach to classification is to estimate the underlying densities and then plug them into a Bayes classifier. The typical setup assumes a random process $(X_i, Y_i), i = 0, 1, \ldots$, where the X_i are k-dimensional real-valued vectors and take values in a space $\mathcal{X}$, and the Y_i designate membership in a class and take values in a set $\mathcal{Y} = \{1, \ldots, L\}$ where L is the number of classes. A classifier $\kappa(x)$ predicts the class identities of input vectors. The performance of the classifier is measured by average Bayes risk

$$B(\kappa) = \sum_{k=0}^{L-1} \sum_{j=0}^{L-1} C_{jk} P(\kappa(x) = k \text{ and } Y = j) = E \sum_{j=0}^{L-1} P(Y = j|X) C_{j,\kappa(X)}\,, \tag{14}$$

where $C_{j,\kappa(x)} \geq 0$ represents the cost of classifying x as class k when the true class Y is j. The Bayes risk is minimized by minimizing the sum for each value of x by choosing the class selected by $\kappa(x)$ to be the value of k that minimizes

the sum over j, i.e., by using the Bayes classifier for the observable x. The optimal classifier is thus given by $\kappa(x) = \arg\min_k \sum_{j=0}^{L-1} C_{jk} P(Y = j|X = x)$. In the case of equal costs for incorrect decisions, i.e., $C_{jk} = 1$ for $j \neq k$ and 0 for $j = k$, the Bayes risk reduces to the probability of error $P(\kappa(X) \neq Y)$ and the minimum Bayes risk classifier is a maximum a posteriori (MAP) or minimum probability of error (MPE) classifier.

In practice, the probabilities are estimated based on the labeled training set, $\mathcal{L} = (x_i, y_i)$ for $i = 1, 2, \ldots, N$, where $y_i \in \mathcal{Y}$ is the class label of the observation x_i and the estimated probabilities are plugged into the Bayes optimal estimator. If the estimator is good, then the estimated distribution should converge to the true distribution in some sense as the training sequence grows, and the approximate Bayes classifier should provide performance near that of the ideal Bayes classifier. One method of estimating these conditional densities is to fit a Gauss mixture model. Each class can be modeled independently as a GM. All the Gaussian mixtures can then be combined with the estimated class probabilities to provide the densities needed to Bayes classification. When this plug-in approach is taken, one can directly compare the results of Lloyd clustering with the EM algorithm for determining the GM models.

Minimum Distortion Classification

A distinct approach which is natural for the quantization viewpoint adopted here is to simply use the QM or MDI distortion in a minimum distortion or "nearest neighbor" classifier. Design the Gauss mixture models for each class as described above using the Lloyd algorithm. Instead of plugging into a Bayes estimator, however, use the same distortion measure minimized in the model design as the classification rule. In particular, use the QM or MDI distortion to select the Gauss component from all of the mixtures, that is, the best Gaussian in the composite code combining all of the class Gauss mixtures. The mixture containing the component corresponds to a class, and that class is the output of the classifier. If a single component were a member of multiple mixtures, then a more Bayesian rule could be used incorporating the appropriate probabilities, but experiments suggest this rarely happens.

A variation on this theme classifies by finding the best codebook rather than the best codeword. Suppose for example we have designed a collection of Gauss mixture models, one for each class, based on input vectors of dimension k. Now instead of classifying a single input vector of dimesion k, we look at a sequence of input vectors, say N of them for a "supervector" of dimension kN. For example, the codebooks might be designed for 8×8 pixel blocks of an image, but we now want to classify an entire image. For example, in a two class problem we might have two codebooks, one for each class. Encode the

observed image using the first codebook and record the overall distortion and do the same for the second codebook. The codebook resulting in the lowest average distortion is declared the winner and its class label is selected. In fact this idea is very old and was used in isolated utterance speech recognition (using an MDI distortion) by Shore and Burton [63].

The minimum distortion classifiers, both the ordinary and the codebook version, do not perform explicit density estimation — instead they classify by trying to optimize the same cost function that was used to design the models.

7 Image Coding

We first consider using the Lloyd clustering algorithm to design a classified or composite vector quantizer. This is the application which motivates the QM distortion measure, which is specifically aimed at providing a classified VQ that is optimal in the sense of yielding the overall minimum average mean squared error when used as a waveform coder. The idea is to design a collection of Gaussian codebooks, each corresponding to a Gaussian component of the overall mixture. The classified quantizer chooses the codebook by minimizing a QM distortion, and then picks the best word in the codebook using ordinary mean squared error. This structure is referred to as a Gauss mixture vector quantizer or GMVQ. It is similar in structure to Hedelin and Skoglund's vector quantizer based on an EM construction of the models [37], the key differences being that we use a Lloyd clustering algorithm with QM to design the codebooks and we consider variable rate coding. These results are not presented as a serious candidate for this application, but simply to reinforce the assumptions used to derive the QM distortion measure. In particular, we wish to demonstrate that the theoretical mismatch provides a good prediction of the mismatch occurred in practice and that the codes designed in this way perform well in comparison to entropy-constrained vector quantization (ECVQ) [11] using a Lagrangian distortion based on MSE. We here provide only a brief survey of these results to point out their behavior. More detailed results may be found in [1].

The setup for the GMVQ design and test proceeds in two phases. In the training phase, image data is used to estimate the parameters of the GM model using the Lloyd algorithm. The Lloyd algorithm is used to estimate the parameters of the model. Upon completion of density estimation, the resulting Gauss mixture model is used to generate a large number of synthetic training data. These training data are used to design the quantizer. In the testing phase, real image data is used to evaluate the performance of the encoder and the decoder. We evaluate the results both on the images used to generate the model (to test the robustness of the Gaussian model on the original emperical

distribution) and on new image data (to test the robustness of the overall approach as a compression system).

Three experiments were run based on GMVQ:

Experiment $\mathrm{GM}_{\mathrm{synth}}$: construct a GM model using the Lloyd clustering algorithm and then design a composite code for this model using synthetic Gaussian data. Test the code on separate synthetic Gaussian data.

Experiment $\mathrm{GM}_{\mathrm{train}}$: construct a GMVQ as in the first simulation. Test the performance on the real images in the training set used to design the model. The intention is to test robustness when the covariance structure is known. Recall that if second-order moments of the new source match the model that the quantizer mismatch approximately zero. In the simulations, we can verify that the distortion and rate from this experiment should match closely the distortion and rate from $\mathrm{GM}_{\mathrm{synth}}$ experiment. By using the same images as the ones used to generate the model, we are ensuring a way of getting very similar second-order moments. We verify the theory at high-rate and see how robust the quantizer is at low rates.

Experiment $\mathrm{GM}_{\mathrm{test}}$: Construct a GMVQ as above, but test on images *outside* the training set. This is a true test of robustness of the approach in a practical sense.

We compare the GMVQ with ECVQ. The ECVQ experiments also proceed in two phases. In the training phase, we directly use image data to design the vector quantizer. In the testing phase, test images are provided as input data. Each image is quantized and its performance is noted. There are two experiments that are run based on ECVQ.

Experiment $\mathrm{ECVQ}_{\mathrm{train}}$ Design a VQ directly on the training data using the entropy-constrained Lloyd algorithm without any model clustering or composite codes. The code is tested on the training set.

Experiment $\mathrm{ECVQ}_{\mathrm{test}}$: Modify $\mathrm{GM}_{\mathrm{test}}$ Design a VQ directly on the training data using the entropy-constrained Lloyd algorithm without any model clustering or composite codes. The code is tested on images *outside* the training set.

Ten training images were used to estimate the GM model. All the images are gray-scale, 8 bits per pixel and 512×512. These images are divided into 8×8 blocks and processed independently. The images are of various types including people, indoor, outdoor. The Gauss mixture design is initiated with 10 clusters. The Lloyd approach may prune this to fewer components. The final design performance is denoted $\rho_{\mathrm{de}} = \overline{d}_{\mathrm{de}} + \lambda \overline{r}_{\mathrm{de}}$. where $\overline{d}_{\mathrm{de}}$ and $\overline{r}_{\mathrm{de}}$ are the average model design distortion and rate. The block size for encoding is 8×8.

We vary λ from 10 to 1000 to cover both high and low rates, respectively. We start the Lloyd algorithm with a large number of codewords ($500,000$ in the experiments) and let the algorithm reach an optimal number of codewords using ECVQ. The rate and distortion achieved by this quantizer on the synthetic training data are denoted by $r_{\mathrm{GM}_{\mathrm{synth}}}$ and $d_{\mathrm{GM}_{\mathrm{synth}}}$, respectively. If the training images are used to test the quantizer, the performance obtained is $d_{\mathrm{GM}_{\mathrm{train}}}$ and $r_{\mathrm{GM}_{\mathrm{train}}}$. On the other hand, if images outside the training data are used to test the quantizer, call the performance obtained as $d_{\mathrm{GM}_{\mathrm{test}}}$ and $r_{\mathrm{GM}_{\mathrm{test}}}$. Figure 5 shows thes robustness results in graphical form. The experiments indicate that the overall quantizer is indeed robust in the desired sense.

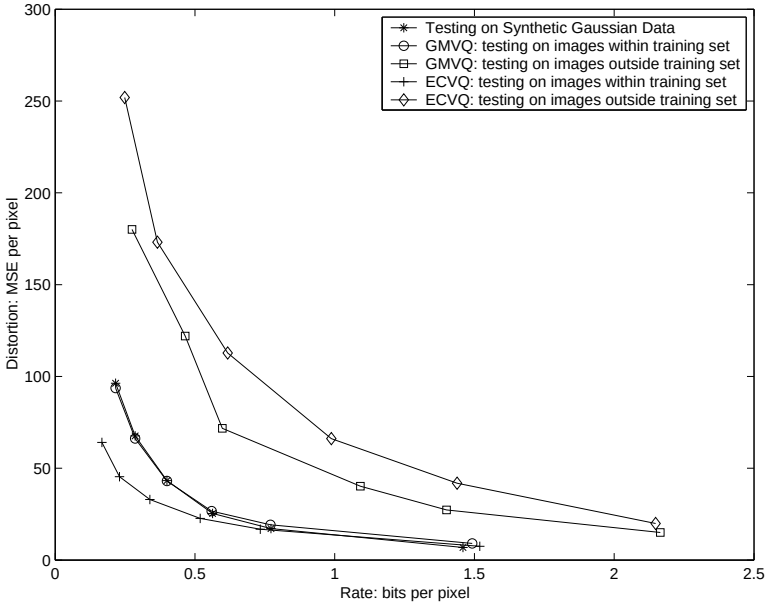


Fig. 5. Distortion-rate performance

The salient points in the figure are the following: $\mathrm{ECVQ}_{\mathrm{train}}$ should be the lowest curve since the quantizer is tested on the same images on which it is trained. ECVQ should provide the best overall Lagrangian performance if it is designed using the true distribution. $\mathrm{ECVQ}_{\mathrm{test}}$ curve should be the highest and should give the worst performance since the testing images and training images are different and the code is not designed to be robust. According to the theory $\mathrm{GM}_{\mathrm{train}}$ should closely match $\mathrm{GM}_{\mathrm{synth}}$. This experiment is designed to test the robustness result of little difference between the performance of the Gaussian code on the Gaussian source and on the original source with matching second order moments. The two curves being close to each other implies that there is very little or no quantizer mismatch. Both the $\mathrm{GM}_{\mathrm{train}}$ and $\mathrm{GM}_{\mathrm{test}}$ should lie between $\mathrm{ECVQ}_{\mathrm{train}}$ and $\mathrm{ECVQ}_{\mathrm{test}}$. Using the GM model followed by vector quantizer, we hope to get lower performance on the

training images by virtue of the locally robust composite code. The GMVQ in this case is built for Gaussian data and not on the training images and hence we expect that GMVQ will yield worse performance on the training data, but hope that its robustness will result in better performance than ECVQ outside the training set. Although we lose on performance of training image data, we hope to be more robust to images outside the training image data. Indeed the figure demonstrates the expected behavior and the simulations closely track the theoretical results; that is, $d_{\mathrm{GM}_{\mathrm{synth}}} \approx d_{\mathrm{GM}_{\mathrm{train}}}$ and $r_{\mathrm{GM}_{\mathrm{synth}}} \approx r_{\mathrm{GM}_{\mathrm{train}}}$. Even at low rates (high λ), the accuracy is very good. In the graph, we can see that the $\mathrm{GM}_{\mathrm{train}}$ *d-r* curve closely tracks the $\mathrm{GM}_{\mathrm{synth}}$ *d-r* curve. Outside the training sequence, GMVQ does better on than ECVQ. In the graph, the $\mathrm{GM}_{\mathrm{test}}$ is lower than the $\mathrm{ECVQ}_{\mathrm{test}}$. Another way to state this result is that the gap between $\mathrm{GM}_{\mathrm{train}}$ and $\mathrm{GM}_{\mathrm{test}}$ is much smaller than the gap between $\mathrm{ECVQ}_{\mathrm{test}}$ and $\mathrm{ECVQ}_{\mathrm{train}}$.

We compare the $\{d, r\}$ performance of the quantizer with the predicted $\{d, r\}$ from the density estimation. That is, $\frac{1}{\lambda}(d_{\mathrm{synth}} + \lambda r_{\mathrm{synth}}) \approx \theta_k - \frac{k}{2} \ln \lambda_{\mathrm{de}} + (d_{\mathrm{de}} + \lambda_{\mathrm{de}} r_{\mathrm{de}})$. Figure 6 shows the prediction result in graphical form. At high rates, the actual value of distortion and rate closely track the predicted value. However, as λ increases, there is a deviation between the two curves.

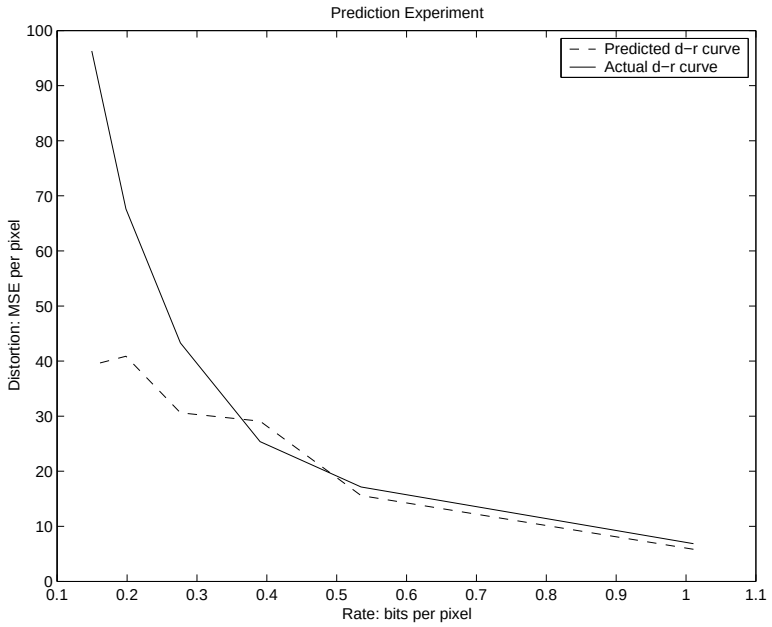
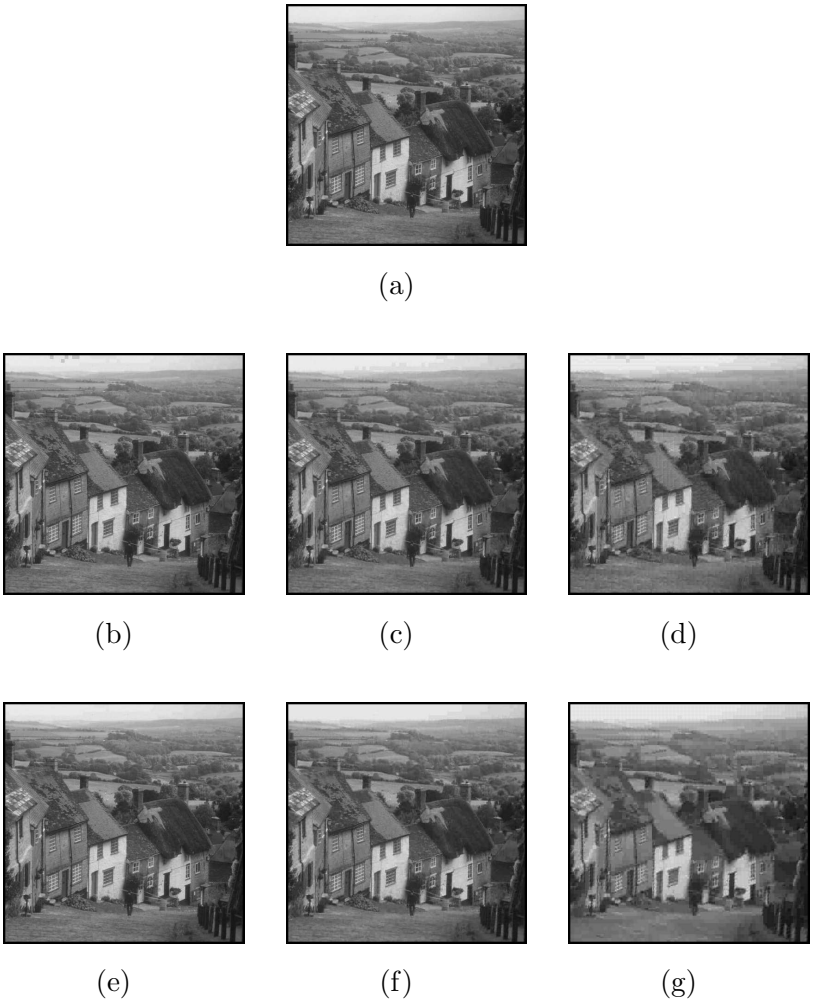


Fig. 6. Distortion-rate curve showing prediction. Graph shows both the actual distortion and rate performance of the quantizer as well as the predicted distortion and rate performance based on density estimation.

As a visual aid, we show in Figure 7 examples of an original image compressed at three different λ values for both the fixed rate and variable rate GM design.

Observe that the image quality of the GMVQ images is comparable to image quality of the ECVQ images. The corresponding images at a given λ have similar distortion and rate. From Figure 5, we know that ECVQ may have better performance on training images. However the important point here is not compression, necessarily, but robustness. The advantage of GMVQ is that it provides robustness, gives a predicted performance, and still provides decent image compression.



(a) (b) (c) (d) (e) (f) (g)

Fig. 7. Experiments and $\mathrm{GM}_{\mathrm{train}}$ and $\mathrm{ECVQ}_{\mathrm{train}}$: GMVQ on training images. (a) original, (b) $\lambda = 10$, PSNR = 39.58 dB, Rate = 1.790 bpp, (c) $\lambda = 100$, PSNR = 33.17 dB, Rate = 0.627 bpp, (d) $\lambda = 1000$, PSNR = 28.15 dB, Rate = 0.215 bpp, ECVQ VQ on training images. (e) $\lambda = 10$, PSNR = 38.61 dB, Rate = 1.702 bpp, (f) $\lambda = 100$, PSNR = 32.01 dB, Rate = 0.526 bpp, (g) $\lambda = 1000$, PSNR = 27.36 dB, Rate = 0.144 bpp

The experiment was repeated using AR models of order 1 (nonzero $a_{0,0} = 1, a_{0,1}, a_{1,0}$ and $a_{1,1}$) and the MDI distortion measure. The rate-distortion results are depicted in Figure 7, which shows similar relative behavior to the

corresponding figures for the QM distortion. Figure 9 shows the difference between the QM, which uses unconstrained Gaussian models, and MDI, which uses the highly constrained first order two-dimensional autoregressive models. It merits noting that within the training set the AR models provide worse performance, which is to be expected since the models are more constrained, but that outside of the training set the AR models provide better performance. This possibly surprising behavior suggests that the general models can overfit the training data and that the more structured AR models are more robust on test data. The visual appearance using both techniques are too close to distinguish [1].

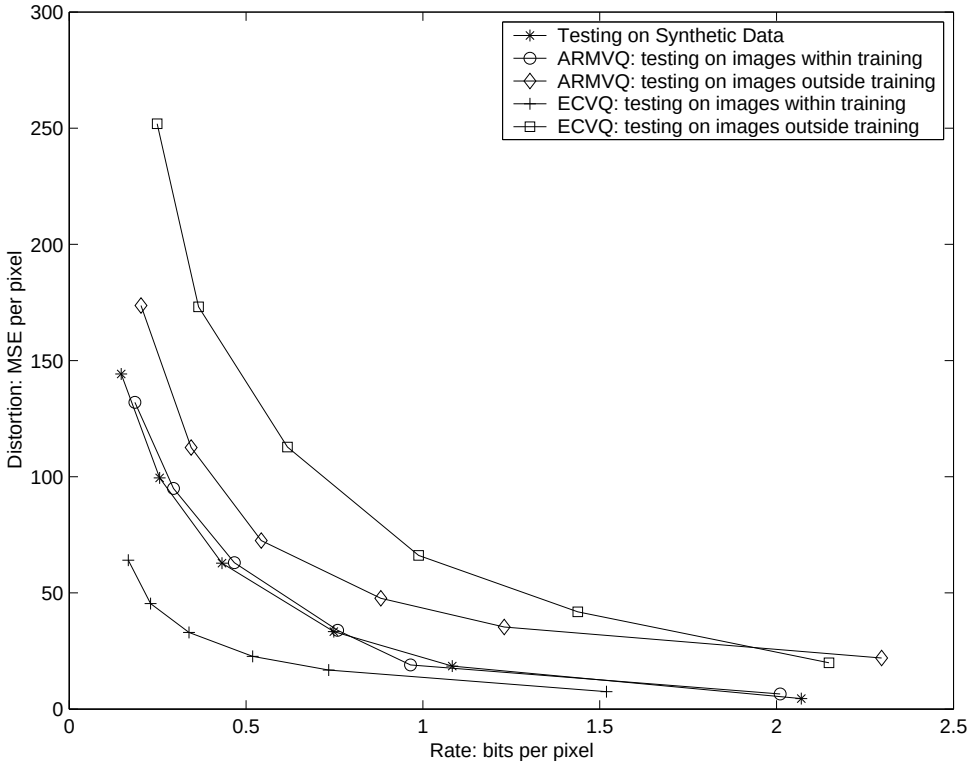


Fig. 8. Distortion-rate performance. Note that $\mathrm{ARM}_{\mathrm{synth}}$ is very close to $\mathrm{ARM}_{\mathrm{train}}$. Also, $\mathrm{ARM}_{\mathrm{test}}$ performs better than $\mathrm{ECVQ}_{\mathrm{test}}$. The relative positions of these curves is very similar to the expected results shown in Figure 6.

8 Image Block Classification

As a simple example of Lloyd classifier, we designed a classifier into manmade and natural pixel blocks for a collection of aerial images of the San Francisco Bay area provided by TRW (formerly ESL, Inc.) [53]. Segmentation of the same dataset of aerial images was also studied in [54]. The data set contains 6 images, whose hand-labeled segmented images are used as the truth set of

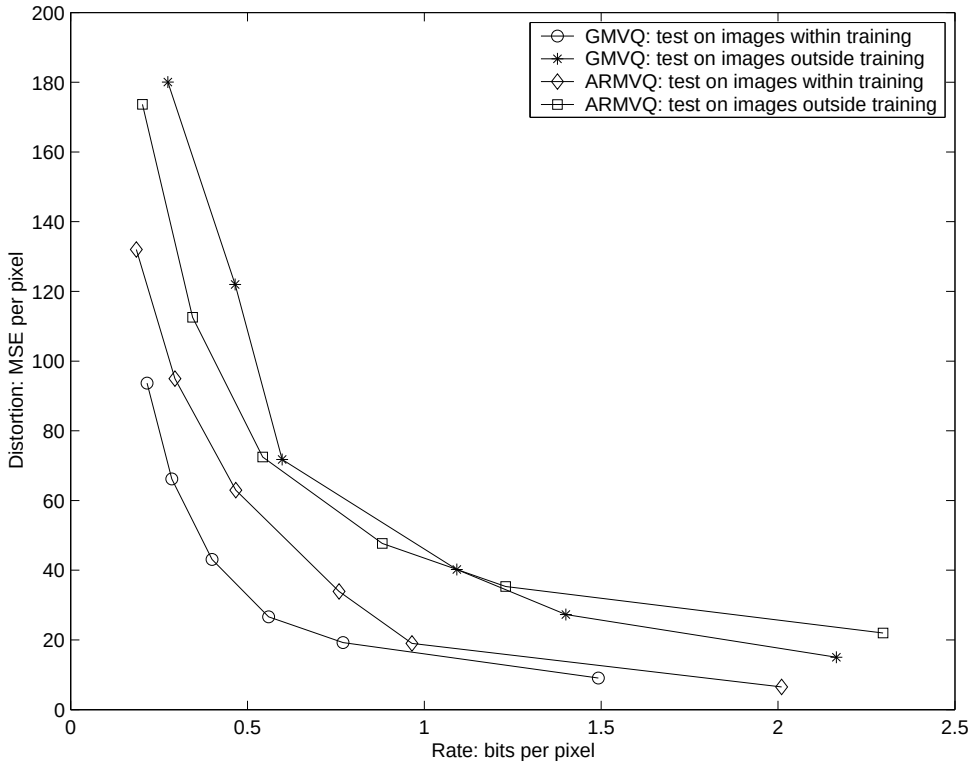


Fig. 9. Comparison of quantizer performance based on AR mixture model and GM model. Note that $\mathrm{ARM}_{\mathrm{train}}$ performs worse than $\mathrm{GM}_{\mathrm{train}}$ and that $\mathrm{ARM}_{\mathrm{test}}$ performs better than $\mathrm{GM}_{\mathrm{test}}$. This shows that AR mixture model is more robust to out-of-training images; however, it is worse on training images.

classes. The images are 512×512 gray-scale images with 8 bits-per-pixel. The Lloyd clustering algorithm was used to design both general Gauss mixtures using the QM distortion and autoregressive Gauss mixtures using the MDI distortion. Handlabeled data was used to construct separate training sets for each class and a Gauss mixture was designed for each class separately. Six-fold cross validation [66] was used, that is, six experiments were run where each used a group of five images for training and the sixth for test. Results report the average over the six experiments.

Ten clusters are used as a starting point for each class. Each image is split into 8×8 blocks and processed independently. We compare our results against several classification algorithms including EM, CART, LVQ, Bayes vector quantization and a classifier based on two-dimensional hidden Markov model (HMM). For more information regarding these algorithms, refer to [15,5,40,54,43].

The classification performance for image 1 and image 6 used as test (not training) images are shown in Figure 10. We compare the classification performance using the Lloyd GM model and the EM algorithm. The classification performance of an image is measured by the probability of error. The probability of error is the number of pixels in the image that are classified incorrectly

(according to the hand-labeled image) divided by the total number of pixels in the image. We can see that the GM model gives good results.

The algorithms were tested by a six-fold cross-validation [66]. For each iteration, one image is used as test data and the remaining 5 are used as training images. The average probability of error achieved using the Lloyd GM model is .14. (???As I recall, this was overly optimistic. Need to see if better value.)

The average performance of all the algorithms is summarized in Table 1. Using the GM model for classification yields the lowest probability of error. Also, note that using this Lloyd GM based classifier also lends itself nicely to a composite model to do both compression and classification. The algorithms were compared under the same conditions; the same set of images were used for training and the EM algorithm had the same number of Gaussian components as the GM algorithm.

Experiments were performed on Pentium II 450 MHz PC with Linux operating system. The average time to classify a 512×512 image was less than 0.1 seconds. Compared with single resolution HMM, which takes about 200 seconds, this is a large benefit. It is slightly faster than CART, which takes around 0.16 seconds on average.

Algorithm	ARM	GM	EM	HMM	CART	LVQ	BVQ
P_e	0.178	0.144	0.233	0.188	0.216	0.218	0.215

Table 1
Average classification performance on test data using 6-fold cross validation.

9 Texture Classification

Texture classification [65,42,67] plays an important role in many image processing applications such as content-addressable image retrieval [4,59,64,69]. We here use the Lloyd clustering algorithm to design Gauss mixture models for a collection of textures from the Brodatz texture database [6]. There have been numerous papers devoted to the classification of Brodatz textures, e.g., [50,9,8,34]. We will describe these in comparison with our own results.

Six samples out of the total 112 Brodatz textures are shown in Figure 11. The left two textures are examples of micro textures, where the granularity is small and the regularity is preserved. The middle two textures are examples of macro textures, where the scale is larger and more irregular than the micro textures. The right two textures are examples of irregular textures, where there is no specific pattern in the texture.

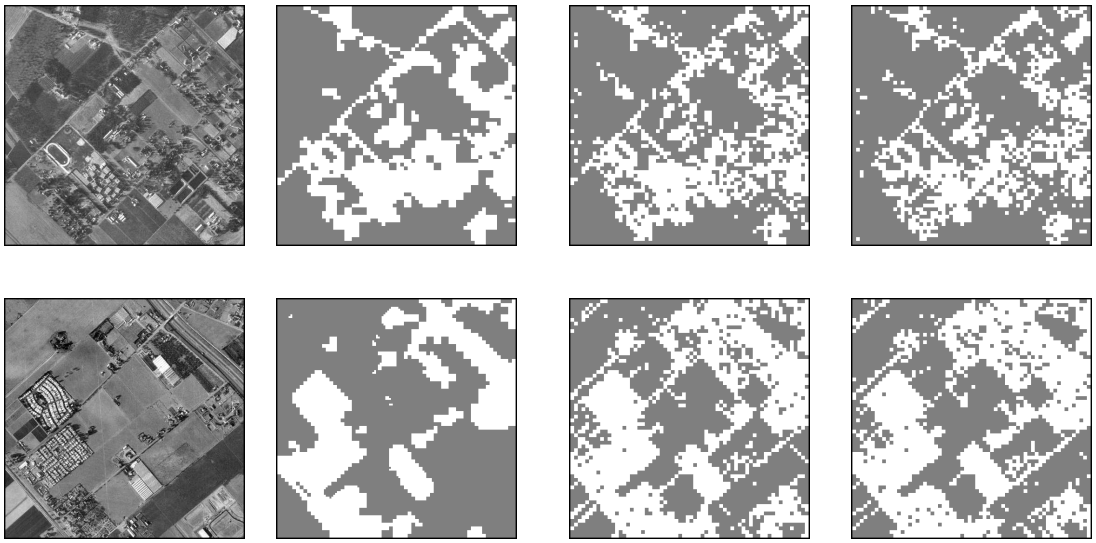


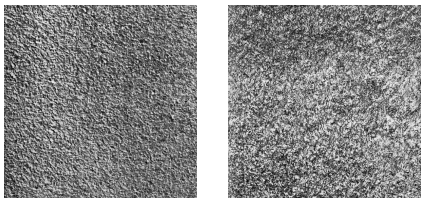
Fig. 10. Comparison of the classification performance of Lloyd GM and EM for an aerial image outside the training set: Top row from left to right: original image, hand-labeled classified image, Lloyd GM with probability of error .21, EM with probability of error .24. Bottom row:original, hand-labled classified image, Lloyd GM with probability of error .17, EM with probability of error .17. White: man-made, gray: natural.

Fig. 11. Examples of micro, macro, and irregular Brodatz texture.

The three pairs from Brodatz textures in Figure 12, 13, and 14 are examples of very similar looking textures. The pair in Figure 13 have a similar look of lizard skin, the one in Figure 14 has a similar mixture of intensity with vertical strips, and the one in Figure 15 resembles human eyes. These examples show that the classification problem for the Brodatz texture database is a challenging problem.

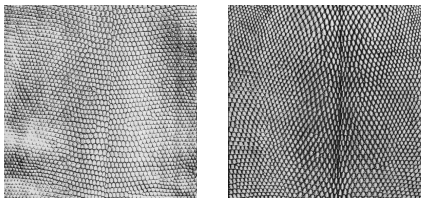
We used the codebook classification approach with an initial block size of $8 \times 8 = 64$ used in the Lloyd algorithm using first order autoregressive models and the MDI distortion.

We applied the GMVQ classifier to 24 arbitrarily chosen textures (20 from the Brodatz database and 4 from the USC database [68]) and we used the full



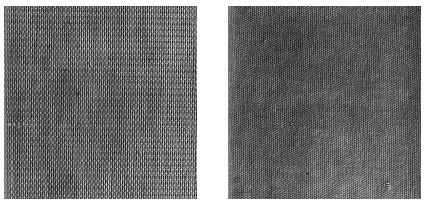
(a) (b)

Fig. 12. Sample 'look alike' Brodatz textures (a) D4, (b) D9.



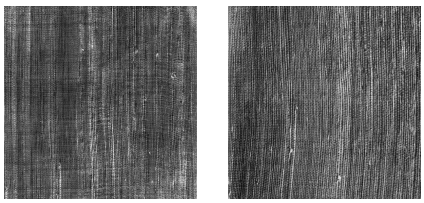
(a) (b)

Fig. 13. Sample 'look alike' Brodatz textures (a) D3, (b) D36.



(a) (b)

Fig. 14. Sample 'look alike' Brodatz textures (a) D53, (b) D77.



(a) (b)

Fig. 15. Sample 'look alike' Brodatz textures (a) D78, (b) D79.

Brodatz texture set for comparison to several well-known algorithms. The full Brodatz texture set can be found in [6]. All 512×512 textures were gray level images with 8 bits per pixel. Our goal was to classify the input at different scales according to one of the known textures. The 20 Brodatz textures were D3, D4, D6, D9, D12, D15, D16, D19, D24, D29, D38, D68, D77, D78, D83, D84, D92, D94, D102, and D112, and the 4 USC textures [68] were 1.5.02,

1.5.03, 1.5.05, and 1.5.07. A problem with texture classification for supervised learning is that it is very hard to obtain the training data. We increased the training set size by diviging textures into smaller blocks which were used as training data since the divided textures are more regular and look similar to other types of images.

We divided each 512×512 texture into sixteen non-overlapping blocks of size 128×128. The performance was evaluated through a sixteen-fold cross validation, where fifteen blocks were used for training and the remaining one for testing. There were 3840 $(= 15 \times 128 \times 128/(8 \times 8))$ vectors for training. In the experiment, we chose the number of mixture components as six (*i.e.*, $L = 6$) during training and used it for testing.

After training, we have multiple Gauss mixture codebooks. During testing, the left-out block of size 128×128 was divided into sub-blocks of various sizes $S = \Big\{(128 \times 128), (64 \times 64), (32 \times 32), (16 \times 16)\Big\}$.

Each large block was classified based on the sample average distortion resulting for codebooks applied sequentially to all subblocks. Classification took less than 1.3 seconds for 128×128 blocks and 0.4 seconds for 64×64 blocks on a Pentium II 450MHz computer with a Linux operating system, providing real-time operation for browsing applications.

Comparisons

We compare the Lloyd Gauss mixture codebook classifier with a variety of alternatives.

Gauss Markov Random Fields Approach

Gauss mixture random field (GMRF) approach of Chellappa et al. [9] attempts to capture the global information of the texture by expanding the neighborhood of a Markov random field. The classification results of the GMVQ classifier were compared with the best results of GMRF methods for 64×64 and 32×32 block sizes. Generally the performance of the classifier degrades where the block size becomes smaller as shown later. Seven textures (Wood (D68), Grass (D9), Bark (D12), Pigskin (D92), Leather (D24), Raffia (D84), and Wool (D19)) from the Brodatz album were used for the experiment with two different input scales, 32×32 and 64×64.

The GMRF classifier assumes that the texture within the specified window was generated from a Gaussian density and that the dependency between the neighboring pixels is Markovian. Two features, least-square estimates of

GMRF model parameters and sample correlations over the symmetric window, were extracted and used independently for the classification in the GMRF model. One of the most important characteristics of the texture is the scale. We captured a large scale by expanding the number of neighbors in GMRF. A fourth-order GMRF model was assumed in [9]. By using this context information, we tried to capture the global statistics of the image.

In our model, the context information is hierarchical. The low-level context information is modeled by the covariance of the neighboring pixels within the AR window whereas the high-level information is modeled by the superblock formula. Instead of using these approaches at the same time during training, we follow the divide and conquer approach. The low-level information is captured in training whereas the high-level information is captured in testing. The superblock formula has arithmetic complexity, which makes it extremely fast. The calculation can be done recursively.

While MDI distortion is used for the GMVQ classifier, least-square distortion is used for the GMRF classifier. We modeled the smooth density of various textures using Gauss mixture models, but used a single Gaussian density for the GMRF classifier. The GMVQ classifier does not assume Markov properties and does not follow Markov models. ????Is this true? Our models are our codewords, which are AR and hence Markov. Smoothing using the superblock formula during testing implicitly assumes that the neighboring blocks can not abruptly change in the texture being modeled, which is a similar but slightly different assumption since the dependency is on the pixel level in Markov random field models.

GMVQ outperformed GMRF for both scales. The performance degradation for reduced block size was not as severe as it was in [9], because Gauss mixtures provide more robust models of multi-modal properties of the smaller block density and the superblock formula captured the context information. Whereas GMRF failed to capture the different scales of the textures, GMVQ successfully captures them by using the superblock formula. The Gauss mixture captures the variability of the density of the neighboring pixels during the training phase, but the superblock formula improved the recognition performance of the irregular textures during the testing phase, resulting in GMVQ performance better than GMRF. Our results were based on 1792 test samples for 32×32 blocks and 448 test samples for 64×64 blocks. They were more reliable than the results based on 112 test samples used in [9].

Method	GMVQ†	GMRF†	GMVQ‡	GMRF‡
CCR	99.8	93.75	100	99.1

Table 2
Comparison between the classification performance between GMVQ and GMRF for † 32×32 and ‡ 64×64 block size. CCR denotes the correct classification rate.

Comparison with multi-resolution TSWT

A problem with GMRF and the second-order statistics methods [10,19,35] is that they fail to capture the different scales of textures because the features depend only on the coupling between the neighboring pixels on a single scale. To overcome this difficulty, multi-resolution approaches such as tree-structured wavelet transforms (TSWT) [8] have been proposed to capture spatial and frequency information at the same time. Conventional pyramid-structured wavelet transforms expand the frequency resolution into the low frequency region. For textures, however, the important information is normally in the middle frequency region [8]. Therefore, TSWT expands the tree adaptively according to the energy of the leaves of the tree, assuming that the energy has the information for discriminating textures. Chang et al. [8] compared three algorithms having a fixed number of features and progressively determined features with popular methods such as DCT, pyramid structured wavelet transform, and Gabor transform using four different distance measures: Bayes, Mahalanobis, simplified Mahalanobis, and Euclidean.

Table 3 presents the results comparing GMVQ with TSWT [8]. An average over the thirty Brodatz textures in table 6 from [8] were used for comparison. The distance measures used for the classification were the following: The

Distortion →	MDI†	d1‡	d2‡	d3‡	d4‡
Average	99.6%	93.5%	99.6%	99.4%	98.9%

Table 3
Comparison of the classification performance of MDI using GMVQ and other distance measures d1, d2, d3, and d4 using TSWT; d1 is the Euclidean distance, d2 is the Bayes distance, d3 is the Mahalanobis distance, and d4 is the simplified Mahalanobis distance.

Euclidean distance is shown in in 15, the Bayes distance in 14, the Mahalanobis distance in 17, and the simplified Mahalanobis distance in 18. The Mahalanobis distance is useful when the statistical properties of textures are known. When the covariance matrix is diagonal, the Mahalanobis distance reduces to the simplified Mahalanobis distance. This makes the calculation recursive and allows fast evaluation of the distance. Assuming that the features are Gaussian with common covariance and different means, a Bayes decision rule is equivalent to a minimum distortion rule with the Mahalanobis distance plus the logarithm of the covariance matrix, *i.e.*, the Bayes distance. Generally the MDI distance provided the best result, with its correct classification rate (CCR) equal to 99.6% of that of the Bayes distance, outperforming slightly the 99.4% CCR of the Mahalanobis distance. The Euclidean distance provided worse performance (CCR 93.5%) than other distance measures, implying that the simple mean square error is not appropriate for classification purposes.

$$D_{1,i} = \sum_{j=1}^{J} \Big(x_j - m_{i,j}\Big)^2 \tag{15}$$

$$D_{2,i} = \Big(x - m_i\Big)^T {C_i}^{-1} \Big(x - m_i\Big) + \ln |C_i| \tag{16}$$

$$D_{3,i} = \Big(x - m_i\Big)^T {C_i}^{-1} \Big(x - m_i\Big) \tag{17}$$

$$D_{4,i} = \sum_{j=1}^{J} \frac{\Big(x_j - m_{i,j}\Big)^2}{c_{i,j}} \tag{18}$$

The block size was 128×128 for our result whereas it was 256×256 for the TSWT results. As pointed out in [50,9,8] and verified experimentally in section 9.0.1, the performance degraded as the block size became smaller. In this sense, our result is more conservative and better than TSWT results based on the non-MDI distortion measures due to GMVQ's comparable performance with smaller block sizes. If overlapping blocks of size 256×256 were used for GMVQ, the performance would likely be higher than the results reported in table 3. All textures except D74 were perfectly classified by the GMVQ classifier based on the MDI in table 3. This is remarkable since even people often find the textures difficult to match (for example, D4 vs. D9 in Figure 12, D3 vs. D36 in Figure 13, D53 vs. D77 in Figure 14, D78 vs. D79 in Figure 15. However, the GMVQ classifier perfectly distinguished these similar looking textures and did so quickly.

Chang et al. [8] made the assumption that the higher energy of the wavelet channel corresponded to more discrimination power and used this as a criterion for progressive expansion of the tree. Our features were parameterized prototypes of the smooth density of neighboring pixels and we used MDI distortion between densities as a criterion for discrimination.

A Gabor wavelet classifier

As shown in the former section, wavelet transforms such as TSWT are popular methods for capturing the different scale of textures. Another popular method is the Gabor transform method since it exhibits properties that are similar to visual sensory systems. The Gabor transform provides the joint localization in the time and frequency domains and has the nice property of invariance to rotation when it is expressed in a polar form. The classification performance of the GMVQ classifier incomparison to the performance of the Gabor wavelet classifier [34] on the entire Brodatz texture database may be found in [55]. Summarizing these results, out of 1776 sample images, 1578 (88.9%) were classified correctly by GMVQ. Haley et al. [34] who used the Gabor wavelet classifier, reported a correct classification rate of 80.4% for 872 sample images, *i.e.*, 701 images were correctly classified (see table 2 in [34]). Both of these results were based on 128×128 blocks, but our results were more reliable than

their results since they were based on twice as many test samples.

Many of the textures in the Brodatz album are not homogeneous. The GMVQ classifier showed a remarkable improvement in CCR for some inhomogeneous textures including D2, D7, and D73. From table **??**, CCR improved from the 12.5% attained with the Gabor wavelet classifier to 93.75% attained with the GMVQ classifer for D2, from 12.5% to 81.25% for D7, and from 12.5% to 43.75% for D73. But we observed a sharp decline in performance, from 81.25% to 40.63% for D7, and from 43.75% to 28.13% for D73, when the superblock size was reduced from 128×128 to 64×64. This shows that the superblock formula combined with GMM successfully captures the multi-modal density of irregular textures. For homogeneous textures such as D4, D9, D35, and D37, 100% CCR was recorded for the GMVQ classifier, which is the same performance as that of the Gabor wavelet classifier. Our performance improvement is due mainly to the improvement we achieved on irregular textures and is based on the combination of GMM and the superblock approach.

In [34], the micro features of textures contained local amplitude, frequency, phase, direction, and directionality information, which were extracted by formulas from the Gabor wavelet coefficients. A Gabor function is a bandpass filter, the product of a Gaussian and a complex sinusoid. [34] extracted six micro features and used them to formulate a single multivariate Gaussian distribution. They selected the features to be rotation-invariant based on the polar form of the two-dimensional Gabor function. The macro features of the textures were based on micro features to capture global amplitude, frequency, and directionality information. Nine macro features were used to estimate the mean and covariance of a single multivariate Gaussian distribution. The classification was determined by the macro features.

Classifying Pipeline Images

Our data were provided by Norsk Electro Optikk (NEO), a company that maps the interior walls of gas pipelines with an optical scanner. NEO intends to catalogue features of interest (e.g. surface characteristics) in the pipeline segments. Accurate classification of this pipeline data allows for early detection of pipeline damage, which is of significant commercial interest. The images are grayscale with size 96×128 pixels. In addition to the raw data, there is a derived dataset consisting of features (22 for each image) hand-picked for their ability to distinguish classes [51,52].

There are, in total, 12 classes in the pipeline dataset, as described in [51], corresponding to various surface characteristics of the pipeline segments. We choose to build classifiers to distinguish three macroclasses: Plain Steel (here-

after Class S), Longitudinal Weld (Class V), and Field Joint (Class W).

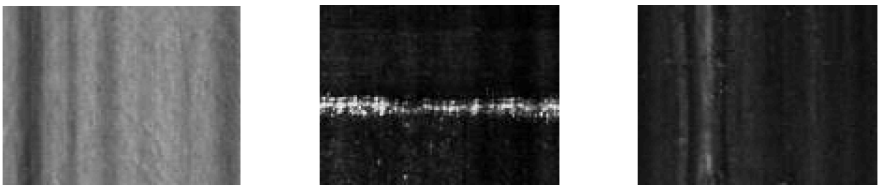


Fig. 16. Normal pipeline image, field joint, longitudinal weld

Macroclass	**Component Classes**	**Sample Count**
S	Normal, Osmosis Blisters, Black Lines, Small Black Corrosion Dots, Grinder Marks, MFL Marks, Corrosion Blisters, Single Dots	153
V	Longitudinal Welds	20
W	Weld Cavity, Field Joint	39

We choose these three macroclasses because they present a realistic classification problem to test our methods upon. The macroclasses, by their very nature, are mixtures, so GM models are well suited here.

The hand-picked (derived) dataset and the image-based dataset have very different characteristics. In the former, vector dimension is low (22) and the information is dense in the dimensions due to human effort. In the latter, vector dimension is high for the whole image ($128 \times 96 = 122880$), much of which is devoid of classifiable content. We apply the appropriate algorithm to each dataset:

- For the hand-picked features, we choose to build classifiers by modeling the source as a random variable in $\mathbb{R}^{22}$. We fit a Gauss mixture model to the training data from each macroclass separately. Final classification is by by minimizing plugin Bayes risk. This is done for both EM and GMVQ using the QM distortion. As a third alternative for comparison we also used ECVQ with $\lambda = 1$ and MSE distortion and then for each resulting quantization cell used the mean and covariance centroid formulas to form a corresponding Gauss component. We used regularized covariances to avoid singularities with $\alpha = 0.01$.
- For the image-based data, we use the codebook version of the GMVQ. since practically, we cannot take the whole image as a single feature vector. Noting that the images in our dataset have been previously stored using JPEG compression and subsequently decompressed, we do two things to avoid JPEG artifacts. For each image, we divide it into 192 8×8 blocks. Instead of using raw pixel values, each 8×8 block is also Fourier transformed, and

the 15 coefficients in the upper-left triangle, with the DC component at position $(1, 1)$, are taken and reshaped into a vector. (In this experiment, including higher frequency coefficients beyond the 15 does not appear to be an improvement as they contain much JPEG quantization noise.) Unrelated to JPEG compression, we take the magnitude of the Fourier transform only, discarding the phase, since we are not interested in shift variations of features in blocks. The 15 dimensional real vectors, then, are used for training with GMVQ. We train separately for the original component classes and combine the classification results into the three macroclasses as the last step. (Again we fix $\lambda = 1$ and $\alpha = 0.01$.)

For comparison, results are also obtained using other established classification methods (Regularized QDA, 1-NN, MART) [36] on the hand-picked features. MART is a gradient boosted version of a classification tree [22]. [2] LDA fits a Gaussian with the same covariance to each class. QDA calculates the covariance independently for each class. Regularized QDA uses a weighted average of the LDA and QDA covariances for each class. The image is assigned to the class with highest probability. The final algorithm considered is a simple one-nearest-neighbor classifier (1-NN) using Euclidean distance.

All methods above are run on the dataset using leave-one-out cross-validation.

The table below shows classification results from all these methods. The first six algorithms classify hand-picked features whereas the final one classifies images using the codebook method. The last four algorithms are GM based, as contrasted with the first three, which are not.

Recall is defined to be (#assigned correctly to class)/# in class), whereas precision is defined to be (# assigned correctly to class/(# assigned to class). Overall accuracy, defined to be (# correct assignments)/(# assignments) is displayed in the rightmost column.

[2] MART was implemented using code available at http://www-stat.stanford.edu/~jhf/

Method	Recall			Precision			Accuracy
	S	V	W	S	V	W	
MART	0.9608	0.9000	0.8718	0.9545	0.9000	0.8947	**0.9387**
Reg. QDA	0.9869	1.0000	0.9487	0.9869	0.9091	1.0000	**0.9811**
1-NN	0.9281	0.7000	0.8462	0.9221	0.8750	1.0000	**0.8915**
MAP-ECVQ	0.9737	0.9000	0.9437	0.9739	0.9000	0.9487	**0.9623**
MAP-EM	0.9739	0.9000	0.9487	0.9739	0.9000	0.9487	**0.9623**
MAP-GMVQ	0.9935	0.8500	0.9487	0.9682	1.0000	0.9737	**0.9717**
Image-GMVQ	0.9673	0.8000	0.9487	0.9737	0.7619	0.9487	**0.9481**

On the hand-picked feature set, the GM based methods (Plug-in-Bayes-ECVQ, Plug-in-Bayes-EM, Plug-in-Bayes-GMVQ) are competitive with the non-GM based methods, outperforming both 1-NN and MART. Arguably, Plug-in-Bayes-GMVQ does equally well as regularized QDA. In fact, excepting Class V, which suffers from a paucity of training and testing data, Plug-in-Bayes-GMVQ does somewhat better. We emphasize that we do not optimize for the best regularization coefficient α in the GM based methods, as is done in regularized QDA. We expect that in a completely equivalent comparison between MAP-GMVQ and regularized QDA, (i.e. optimizing for α in both), and with enough data, the former would do better than the latter for datasets with significant local features.

Next, we compare the three underlying GM clustering algorithms. We find that GMVQ tends to perform slightly better than EM, here and in other test cases. ECVQ, on the other hand, assumes nothing about the shape of the distribution during the clustering process, and tends to overfit the data and can perform poorly at times. Consistently accurate classification on different datasets empirically shows that GMVQ can be an excellent alternative to the more popular EM method for fitting GM models to data, considering that GMVQ converges more quickly than EM and, supplied with a Lagrangian distortion, needs no specialized pruning procedure as EM does.

Whole image classification also performs surprisingly well compared to the other methods, again outperforming MART and 1-NN. Though it is not as good as the best of the others, we must keep in mind that no class-specific features are pre-selected for this classification, which is a compelling advantage in favor of this method.

Figure 2 depicts the action of the clustering algorithm and Figure 2 shows the convergence of the QM distortion with Lloyd iteration. The convergence of the distortion is plotted in Figure 2

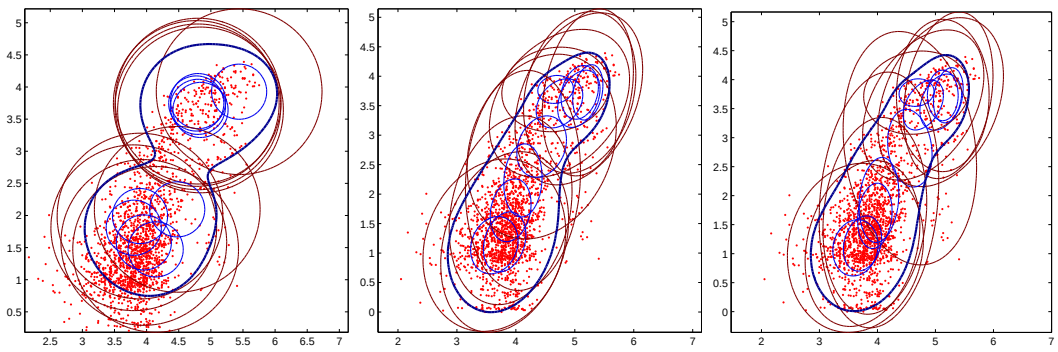


Fig. 17. Projection of feature vectors onto first two dimensions for Field Joint Class: initial code, first iteration, convergence. Ellipses show equal values of Gaussian component densities.

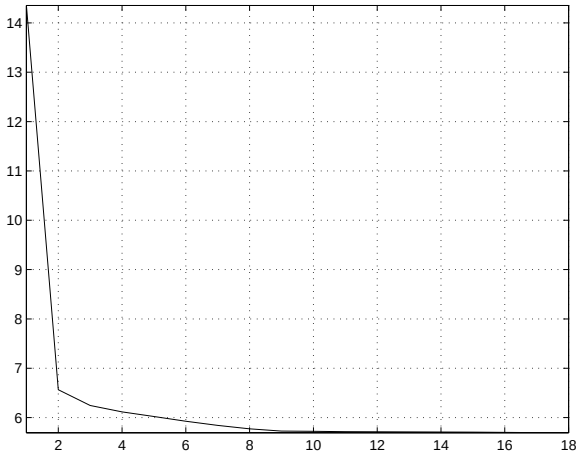


Fig. 18. Convergence of the QM distortion with Lloyd iteration.

The experiments were also run using first order two-dimensional autoregressive models. The results were similar but slightly worse.

10 Conclusions

We have described and developed an algorithm for Gauss mixture design based on Lloyd clustering and demonstrated its application to both image compression and statistical classification of image blocks. We have compared and contrasted the Lloyd algorithm to the dominant approach, the EM algorithm. While EM provides an approximate maximum likelihood estimation of the parameters of an observed Gauss mixture, the Lloyd approach quantizes the space of all nonsingular Gaussian models so that quantizers or source codes designed for the models will together produce a good composite quantizer or source code for the observed data, which is not itself assumed to be from a

Gauss mixture. EM assumes a Gauss mixture model for the observed data and tries to to estimate the density, Lloyd does not assume a specific model for the observed data, but tries to fit a Gauss mixture model to the data in a way that the model produces waveforms that provide a good codebook for the data. Both approaches are "optimum" for specific signal processing problems, but they differ in philosophy and the optimization goal. We have shown by example that the robustness predicted by the theory is consistent with actual designs using Lloyd clustering. We have also argued based on theory and demonstrated by example that the Lloyd clustering design of a Gauss mixture model using a minimum distortion encoder and the quantizer mismatch encoder yields a variation on nearest neighbor classification where the nearest neighbor is found using the same distortion measure as was used to design the mixture. We also point to the success of similar methods in LPC and CELP speech as examples which reinforce the simple image coding and classification experiments described here.

References

[1] A. K. Aiyer. *Robust Image Compression using Gauss Mixture Models*. PhD thesis, Stanford University, 2001.

[2] H. Akaike. Information theory and an extension of the maximum likelihood principle. In *Proceedings IEEE International Symposium Information Theory*, pages 267–281, 1973.

[3] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993.

[4] S. Belongie, C. Carson, H. Greenspan, and J. Malik, "Color-and texture-based image segmentation using em and its application to content-based image retrieval," in *Proceedings of International Conference on Compueter Vision*, Bombay, India, 1998, pp. 675–682.

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

[6] P. Brodatz, "Textures: Photographic album for artists & designers," New York: Dover, New York, 1966.

[7] J. A. Bucklew and G. L. Wise. Multidimensional asymptotic quantization theory with rth power distortion measures. *IEEE Transactions on Information Theory*, 28:239–247, March 1982.

[8] T. Chang and C.-C. J. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE Trans. On Image Processing*, vol. 2, no. 4, pp. 429–441, October 1993.

[9] R. Chellappa and S. Chatterjee, "Classification of textures using gaussian-markov random fields," *IEEE Trans. Acoust., Speech. Signal Process.*, vol. 33, pp. 959–969, August 1985.

[10] P. C. Chen and T. Pavlidis, "Segmentation by texture using correlation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 5, pp. 695–707, January 1983.

[11] P. A. Chou, T. Lookabaugh, and R. M. Gray. Entropy-constrained vector quantization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37:31–42, jan 1989.

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

[13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, 1991.

[14] N. M. B. de Pinho Cruz de Vasconcelos. *Bayesian Models for Visual Information Retrieval*. PhD thesis, Massachusetts Institute of Technology, 2000.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Jounral of the Royal Statistics Society*, 39(1):1–21, 1977.

[16] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York, 1973.

[17] B. S. Everitt. A finite mixture model for the clustering of mixed-mode data. *Statistical Probability Letter*, 6(5):305–309, 1988.

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

[19] O. D. Faugeras and W. K. Pratt, "Decorrelation methods of texture feature extraction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 2, pp. 323–332, July 1980.

[20] M. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[21] C. Fraley. Algorithms for model-based gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 20(1):270–281, 1998.

[22] J. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, vol. 39, no. 5, 2001.

[23] A. Gersho. Asymptotically optimal block quantization. *IEEE Transactions on Information Theory*, 25:373–380, July 1979.

[24] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic publishers, Boston, 1992.

[25] S. Graf and H. Luschgy. *Foundations of Quantization for Probability Distributions*. Springer, Berlin, 2000.

[26] R. M. Gray, "Gauss mixture vector quantization," *Proceedings 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Vol. 3, pp. 1769–1772, Salt Lake City, May 2001.

[27] R. M. Gray, Jr. A. H. Gray, G. Rebolledo, and J. E. Shore. Rate distortion speech coding with a minimum discrimination information distortion measure. *IEEE Transactions on Information Theory*, 27(6):708–721, November 1981.

[28] R. M. Gray, A. Buzo, A. H. Gray, Jr., and Y. Matsuyama. Distortion measures for speech processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-28:367–376, aug 1980.

[29] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, October 1998.

[30] R.M. Gray and T. Linder. Mismatch in high rate entropy constrained vector quantization. *IEEE Trans. Inform. Theory*, 49:1204–1217, May 2003.

[31] R.M. Gray, T. Linder, and J. Li. A Lagrangian formulation of zador's entropy-constrained quantization theorem. *IEEE Trans. Inform. Theory*, 48(3):695–707, Mar. 2002.

[32] R.M. Gray, J.C. Young, and A. K. Aiyer. Minimum discrimination information clustering: modeling and quantization with gauss mixtures. In *Proceedings 2001 IEEE International Conference on Image Processing*, volume 2, pages 14–17, Thessaloniki, Greece,, Oct. 2001.

[33] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. *Journal of Royal Statistical Society*, B 58:155–176, 1996.

[34] G. M. Haley and B. S. Manjunath, "Rotation-invariant texture classification using a complete space-grequency model," *IEEE Trans. On Image Processing*, vol. 8, no. 2, pp. 255–269, 1999.

[35] R. M. Haralick, "Statistical and structural approaches to texture," *Proc. IEEE*, vol. 67, pp. 786–804, May 1979.

[36] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, Springer-Verlag, New York, 2001.

[37] P. Hedelin and J. Skoglund. Vector quantization based on Gaussian mixture models. *IEEE Transactions on Speech and Audio Processing*, 8(4):385–401, July 2000.

[38] F. Itakura and S. Saito, "A statistical method for estimation of speech spectral density and formant frequencies," *Electron. Commun. Japan*, vol. 53-A, pp. 36–43, 1970.

[39] A. K. Jain and J. V. Moreau. Bootstrap technique in cluster analysis. *Pattern Recognition*, 20(5):547–568, 1987.

[40] T. Kohonen. An introduction to neural computing. *Neural Networks*, 1:3–16, 1988.

[41] S. Kullback. *Information Theory and Statistics*. Dover, New York, 1968. Reprint of 1959 edition published by Wiley.

[42] A. Jain and A. Vailaya, "Image retrieval using color and shape," *Pattern Recognition Journal*, vol. 29, pp. 1233–1244, August 1996.

[43] J. Li, A. Najmi, and R.M. Gray, "Image classification by a two dimensional hidden Markov model," *IEEE Transactions on Signal Processing*, Vol. 48, pp. 517–533, February 2000.

[44] H. Lev-Ari, S. R. Parker, and T. Kailath, "Multidimensional maximum-entropy covariance extension," *IEEE Transactions on Information Theory*, Vol. 35, pp. 497–508, May 1989.

[45] X. Q. Li and I. King. Gaussian mixture distance for information retrieval. In *Proceedings International Conference on Neural Networks*, volume 4, pages 2544–2549, Washington, DC, July 1999.

[46] Y. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

[47] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 1957. Bell Laboratories Technical Note. Reprinted in *IEEE Transactions on Information Theory*, 28:127–135, March 1982.

[48] J. MacQueen, "Some methods for classification and analysis of multivariate observations." In *Proc. of the Fifth Berkeley Symposium on Mat. Stat. and Prob.*, volume 1, pages 281–296, 1967.

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

[50] G. F. McLean, "Vector quantization for texture classification," *IEEE Trans. Systems, Man, Cybernetics*, vol. 23, no. 3, May/June 1993.

[51] D.B. O'Brien, M. Gupta, R.M. Gray, J.K. Hagene, "Automatic Classification of Images from Internal Optical Insepection of Gas Pipelines," *ICPIIT VIII Conference 2003*, Houston.

[52] D.B. O'Brien, M. Gupta, R. M. Gray, J. K. Hagene, "Analysis and classification of internal pipeline images," *Proceedings of ICIP 2003*, Barcelona, Spain.

[53] K.L. Oehler and R.M. Gray, "Combining image compression and classification using vector quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 17, pp. 461–473, May 1995.

[54] K. O. Perlmutter, C. L. Nash, and R. M. Gray. A comparison of Bayes risk weighted vector quantization with posterior estimation with other VQ-based classifiers. In *Proceedings of International Conference on Image Processing*, volume 2, pages 217–221, Austin, TX, 1994.

[55] Kyungsuk Pyun, *Classification and Segmentation of Images using Hidden Markov Gauss Mixture Models*, PhD Dissertation, Department of Electrical Engineering, Stanford University, June 2003.

[56] K. Pyun, C.S. Won, J. Lim, R.M. Gray, "Texture classification based on multiple Gauss mixture vector quantizers," *Multimedia and Expo, 2002*, pp 501-4, 2002.

[57] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Transactions on Communication Technology*, COM-34(11):1105–1115, November 1986.

[58] R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the em algorithm. *SIAM Review*, 26(2):195–239, 1984.

[59] Y. Rui, T. Huang, and S.-F. Chang, "Image retrieval: Current techniques, promising directions and open issues," *Journal of visual communication and image representation*, vol. 10, pp. 39–62, March 1999.

[60] D. J. Sakrison. Worst sources and robust codes for difference distortion measures. *IEEE Transactions Information Theory*, 21:301–309, May 1975.

[61] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

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

[63] J. E. Shore and D. K. Burton, "Discrete utterance speech recognition without time alignment," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, may 1982, p. 907.

[64] J. Smith and S. Chang, "Visualseek: a fully automated content-based image query system," *ACM Multimedia*, pp. 87–98, 1996.

[65] J. R. Smith and S. F. Chang, "Tools and techniques for color image retrieval," in *IST/SPIE - Storage Retrieval for Image and Video Databases*, vol. 2670, San Jose, CA, February 1996, pp. 426–437.

[66] M. Stone. Cross-validation. *Mathematics Operationsforsch. Statistical Ser. Statistics*, 9(1):127–139, 1978.

[67] N. Vasconcelos, "Bayesian models for visual information retrieval," Ph.D. dissertation, Massachusetts Institute of Technology, Programs in Media Arts and Science, 2000.

[68] A. Weber, "Image database," USCIPI Rep. 1070, Image Processing Institute, March 1983.

[69] J. C. Young, "Clustered gauss mixture models for image retrieval," Ph.D. dissertation, Stanford University, Electrical Engineering Department, April 2003.

[70] P. L. Zador. Topics in the asymptotic quantization of continuous random variables, 1966. Unpublished Bell Laboratories Memorandum.

[71] X. Zhuang, Y. Huang, K. Palaniappan, and Y. Zhao. Gaussian mixture density modeling, decomposition and applications. *IEEE Transactions on Image Processing*, 5(9):1293–1301, September 1996.