Adaptive Principal Component Analysis

Cynthia Archer and Todd K. Leen
Department of Computer Science and Engineering
Oregon Health & Science University

Technical Report CSE-02-008

November 27, 2002

Abstract

We develop a new signal modeling method, entropy-constrained adaptive PCA, that has the flexibility to accurately model the cluster structure of non-stationary data. Using a latent data framework, we derive a statistical model for a broad category of real world signals that includes images and measurements from natural processes. Data of this type consists of a collection of low-dimensional patterns embedded in a high-dimensional observation or measurement space. We use this statistical model to develop our adaptive PCA algorithm. Our algorithm adjusts the model parameters to minimize the dimension reduction error between the model and sample data subject to a constraint on the entropy.

We evaluate the quality of models produced by adaptive PCA using image texture data and salinity and temperature measurements from the Columbia river. Compared to entropy-constrained vector quantization, local PCA and full-covariance models, adaptive PCA proved to be a more effective tool for analyzing the salinity and temperature data. In addition, our results show that our model segments texture images as well as entropy-constrained vector quantizers, yet uses substantially fewer model components. Adaptive PCA models conform to the data structure better than full covariance models when training data is sparse.

1 Introduction

Classical methods for signal modeling, e.g. global linear models, are limited in that they accurately model only simple, invariant signals. Complex real-world signals require more innovative modeling approaches, since the statistical characteristics of such data vary within the data space. *Collections* of local linear models, which partition the data space and then model data within each region, offer a promising approach to modeling such signals. However, most "collection of model" approaches have their own limitations: they either require large amounts of training data, limiting their usefulness on small data sets; or must be heavily constrained geometrically, enforcing too much uniformity of model components to accurately model non-stationary data. Our goal in this paper is to develop a new method for creating collections of local linear models that strikes a balance between these two extremes, allowing us to derive models appropriate for real-world data.

The classic example of a collection or mixture of linear models is the Gaussian mixture model (GMM) with full or unconstrained covariance. Such models are often a poor choice for high-dimensional data, as sufficient training examples are rarely available to produce robust models. To reduce training data requirements, one typically constrains the covariance to be spherical or diagonal [1], which limits the ability of the model components to conform to the natural data structure. Adaptive principal component analysis (PCA), which models data as a collection of hyperplanes, has the potential to strike a balance between full covariance and spherical GMMs. Recently several researchers [2, 3, 4] have developed effective dimension reduction methods using adaptive PCA. In addition, Tipping and Bishop [5] and Ghararamani and Hinton [6] have developed statistical models for mixture PCA and the related technique, mixture factor analysis, respectively. Despite their success, these methods under-utilize the potential of local or adaptive PCA models by requiring a single global dimension for all model components.

The intrinsic dimension of real-world signals, such as image or speech data, varies throughout the signal space. In prior work [7], we found that dimension reduction performance of local PCA methods could be substantially improved by allowing the dimension to vary. Meinicke and Ritter [8] have recently proposed a mixture PCA model that incorporates variable dimension and produces higher likelihood models than fixed-dimension methods.

Recently, we developed a statistical model for transform coding [9, 10], a common methods of signal compression. From this model, we derived a new generalized Lloyd algorithm for transform coding, in which coder complexity is controlled by an entropy constraint.

The entropy constraint arises naturally from the associated statistical model. A similar construction for adaptive PCA should provide an effective way to allow the dimension to conform to the data structure, while limiting model complexity.

In order to develop more flexible adaptive PCA models, we first develop a statistical model of the data. Using a latent framework, we derive a model for a broad category of real-world data that consist of collections of several distinct low-dimensional patterns, or classes, embedded in a high-dimensional observation space. This probability model is the same as that developed independently by Meinicke and Ritter [8]. However, we take the development further by recognizing the entropy-constrained form of the cost function and developing a new hard-clustering algorithm for adaptive PCA.

Following our algorithm derivation, we describe several training methods used to fit model parameters to sample data. We conclude with an evaluation of our adaptive PCA algorithm on both low and high dimensional real-world data; salinity and temperature measurements from the Columbia River Estuary and image texture data. As an additional extension of prior adaptive PCA work [5, 8], we compare the ability of our adaptive PCA model to separate data into its distinct classes to that of both spherical and full-covariance models. We find that our adaptive PCA approach indeed allows us to specify models that are neither overly "data-hungry" nor overly constrained geometrically. These models accurately represent this broad category of real-world data even when the sample data is sparse.

2 Adaptive PCA Model

In this section, we present the statistical model from which we derive our entropy constrained adaptive PCA algorithm. This model is developed within a latent data framework, which follows that presented by Tipping and Bishop [5] for probabilistic PCA and Basilevsky [11] and Roweis and Ghahramani [12] for factor analysis. The latent data framework is based on the presumption that observed signals are not as complex as they appear. Instead they have some simple latent structure, which is obscured by linear transformations and noise. Our goal is to recover this underlying structure in order to improve our understanding of the data and to reduce the size of the signal representation.

For adaptive PCA, we envision a d dimensional latent data space S, where data from the latent space is mapped to a d dimensional observation space X. The latent data, s, is

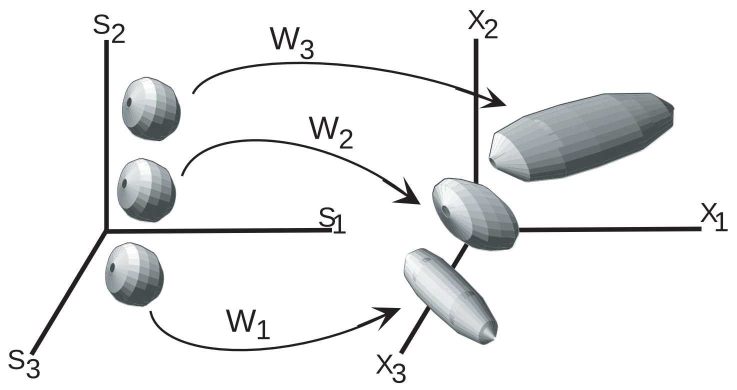


Figure 1: Adaptive PCA Model. Structure of latent variable space, S, and mapping to observed space, X. The data density in the latent space consists of a three Gaussians. This latent data is mapped to the observed data space by orthogonal transform, W, which stretch and rotate the data.

modeled with a simple *mixture* density of the form

$$\mathrm{p}(s) = \sum_{\alpha=1}^{M} \pi_\alpha \, \mathrm{p}(s|\alpha) \tag{1}$$

where π_α are the mixing coefficients and the components are spherical Gaussians $\mathrm{p}(s|\alpha) = \mathcal{N}(\eta_\alpha, \rho^2 \mathrm{I})$ with means η_α and variance ρ^2.

Unique linear maps with translation μ_α and rotation plus scaling transform W_α embed the latent data in the observed space, X. W_α consists of two parts, an orthogonal transform U_α and a diagonal scaling transform Γ_α, so that $W_\alpha = U_\alpha \Gamma_\alpha^{\frac{1}{2}}$. Zero entries in Γ_α suppress latent variables, which causes the model dimension d_α to drop below d. The number of columns in U_α is set by the number of non-zero entries in Γ_α, so that U_α is a $d \times d_\alpha$ matrix. The embedded data is corrupted with additive Gaussian noise, $\epsilon_\alpha \sim \mathcal{N}(0, \sigma_\alpha^2 \mathrm{I})$. Figure 1 illustrates this mapping from latent to observed space.

The observed data generated from a sample s drawn from latent component α is

$$x = W_\alpha(s - \eta_\alpha) + \mu_\alpha + \epsilon_\alpha \tag{2}$$

with conditional densities

$$\mathrm{p}(x|s,\alpha) = \mathcal{N}(\mu_\alpha + W_\alpha(s - \eta_\alpha), \sigma_\alpha^2 \mathrm{I}) \tag{3}$$

The latent data density and mapping induces a mixture of constrained Gaussians density on x of the form

$$\mathrm{p}(x) \quad = \quad \int \sum_\alpha \mathrm{p}(x|s,\alpha)\mathrm{p}(s|\alpha)\pi_\alpha ds$$

$$\mathrm{p}(x) \;=\; \sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x|\alpha) \tag{4}$$

where π_α are the same mixing coefficients given in (1) and $\mathrm{p}(x|\alpha) = \mathcal{N}(\mu_\alpha, \Sigma_\alpha)$. The covariance is constrained such that

$$\Sigma_\alpha = \sigma_\alpha^2 \mathrm{I} + U_\alpha \Gamma_\alpha U_\alpha^T \tag{5}$$

where, without loss of generality we choose the latent variance ρ^2 to be one. We make no assumptions about the latent means, η_α.

The Expectation-Maximization algorithm (EM) [13] fits parametric probability models to data by maximizing the log likelihood of the model for some training data set $\{x_n,\ n = 1 \dots N\}$. For additional information on mixture model fitting see chapter two of [14]. The log likelihood for this model is given by

$$\mathcal{L} = \sum_{n=1}^{N} \log\left(\sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x_n|\alpha) \right) \tag{6}$$

To simplify the log likelihood equation (6), we introduce the density $z(\alpha, x_n)$ over the unknown component assignments.

$$\mathcal{L} = \sum_{n=1}^{N} \log\left(\sum_{\alpha=1}^{M} z(\alpha, x_n) \frac{\pi_\alpha \mathrm{p}(x_n|\alpha)}{z(\alpha, x_n)} \right) \tag{7}$$

where $\sum_\alpha z(\alpha, x) = 1$. Using Jensen's inequality to bring the sum over α outside the logarithm function, we find $\mathcal{L}$ is bounded below by the *expected* log likelihood

$$\mathcal{L} \geq \sum_{\alpha=1}^{M} \sum_{n=1}^{N} \left(z(\alpha, x_n) \log \pi_\alpha \mathrm{p}(x_n|\alpha) - z(\alpha, x_n) \log z(\alpha, x_n) \right) \tag{8}$$

with equality when the $z(\alpha, x_n)$ are the posterior probabilities $\mathrm{p}(\alpha|x_n)$ [Neal and Hinton (1998)]. This choice of z produces soft-clustering models.

Researchers have recently developed two different probability models for PCA [5, 8], which can be derived from this framework. In Tipping and Bishop's model [5] the dimension is the same for all components, $d_\alpha = d_o,\ \forall \alpha$. The target dimension d_o is specified prior to model fitting and the noise variances σ_α^2 are fit to data. In the limit that all noise variances are identical, $\sigma_\alpha^2 = \sigma^2$, and go to zero, the EM algorithm for fitting this model reduces to Kambhatla and Leen's Local PCA algorithm [2] for clustering by reconstruction distance. In this hard-clustering limit, the posterior probabilities become zero or one.

$$\mathrm{p}(\alpha|x) \rightarrow \begin{cases} 1 & \text{if } (x-\mu_\alpha)^T(\mathrm{I} - U_\alpha U_\alpha^T)(x-\mu_\alpha) \leq (x-\mu_\gamma)^T(\mathrm{I} - U_\gamma U_\gamma^T)(x-\mu_\gamma)\ \forall\, \gamma \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In addition, the expected log likelihood (8) reduces to the cost function for local PCA

$$\mathcal{C} = \frac{1}{N}\sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\left((x_n - \mu_\alpha)^T(1 - U_\alpha U_\alpha^T)(x_n - \mu_\alpha)\right) \tag{10}$$

We take a different approach and use the noise variance to control model complexity instead of constraining the dimension to be the same everywhere in the data space. Our approach was inspired by our development of statistical models for transform coding [9, 10]. We found that choosing the noise variance to be the same for all components, $\sigma_\alpha^2 = \sigma^2$, $\forall\alpha$ produces entropy-constrained cost functions for *variable-rate* coding. A similar construction for adaptive PCA should allow the local dimensions to adjust to the data structure. Consequently, like Meinicke and Ritter [8], we choose the noise variances for all components to be the same and fit the local dimensions d_α to the data. With identical component noise variances σ^2, the local covariance matrices (5) become

$$\Sigma_\alpha = \sigma^2\mathrm{I} + U_\alpha\Gamma_\alpha U_\alpha^T \tag{11}$$

In the limit that σ^2 goes to zero, each local covariance matrix (11) reduces to $\Sigma_\alpha = U_\alpha\Gamma_\alpha U_\alpha^T$ with $d_\alpha = d$. That is, this latter model becomes a classic Gaussian mixture model with unconstrained covariance matrices.

To expand the log likelihood (8) for our adaptive PCA model, we first invert Σ_α (11) using the Sherman-Morrison-Woodbury formula [16]

$$\Sigma_\alpha^{-1} = \frac{1}{\sigma^2}(\mathrm{I} - U_\alpha U_\alpha^T) + U_\alpha\Lambda_\alpha^{-1}U_\alpha^T \tag{12}$$

with diagonal $d_\alpha \times d_\alpha$ matrix $\Lambda_\alpha = \Gamma_\alpha + \sigma^2\mathrm{I}$. Using (12) to expand (8) gives the expected data log likelihood

$$\begin{aligned}\mathcal{L} \;=\; & \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\,\ln\,\pi_\alpha + \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\frac{-1}{2\sigma^2}\left((x_n - \mu_\alpha)^T(\mathrm{I} - U_\alpha U_\alpha^T)(x_n - \mu_\alpha)\right) + \\ & \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\frac{-1}{2}\left((x_n - \mu_\alpha)^T U_\alpha\Lambda_\alpha^{-1}U_\alpha^T(x_n - \mu_\alpha) + \ln|\Lambda_\alpha| + (d - d_\alpha)\ln\sigma^2\right) - \\ & \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\ln z(\alpha, x_n)\end{aligned} \tag{13}$$

where z are the posterior probabilities, $p(\alpha|x)$. Our model parameters include the component means, μ_α, the component dimensions, d_α, the component stretching matrices, Γ_α, the component transform matrices, U_α, and the number of components, M. The noise variance σ^2 is considered a control variable, rather than a model parameter.

3 Entropy-Constrained Adaptive PCA

Many signal processing applications, such as compression or on-line classification, benefit from incorporating hard-clustering methods that assign each data item to one and only one model component. For example, compression involves finding a compact representation for data and hard assignments can be coded more efficiently than posterior probabilities. For exploratory data analysis, hard-clustering is easier to visualize and interpret. On-line and embedded classification applications have tight memory and computational time constraints. Hard clustering implementations require less memory and processing time than comparable soft clustering methods making them more suitable for such applications.

3.1 Adaptive PCA Cost Function

The EM algorithm provides a template for deriving hard-clustering algorithms from latent data probability models. To achieve hard-clustering, instead of the soft clustering provided by $\mathrm{p}(\alpha|x)$, we choose $z(\alpha, x_n)$ to be one or zero.

$$z(\alpha, x_n) = \begin{cases} 1 & \mathrm{p}(\alpha|x_n) > \mathrm{p}(\gamma|x_n)\ \forall\gamma \neq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The hard assignments given in (14) partition the data space into regions R_α such that

$$\sum_{x \in R_\alpha} f(x) = \sum_{n=1}^{N} z(\alpha, x_n) f(x) \tag{15}$$

for any function $f(x)$. By choosing hard clustering with z given by (14), the expected log likelihood (13) reduces to the entropy-constrained cost function for adaptive PCA

$$\begin{aligned} \mathcal{C} &= \frac{1}{\sigma^2} \sum_{n=1}^{N} \sum_{\alpha=1}^{M} z(\alpha, x_n) \Big[(x_n - \mu_\alpha)^T (1 - U_\alpha U_\alpha^T)(x_n - \mu_\alpha) - 2\sigma^2 \ln\, \pi_\alpha + \\ & \quad 2\sigma^2 \left(\frac{1}{2}(x_n - \mu_\alpha)^T U_\alpha \Lambda_\alpha^{-1} U_\alpha^T (x_n - \mu_\alpha) + \frac{1}{2} \ln|\Lambda_\alpha/\sigma^2| + \frac{d}{2} \ln \sigma^2 \right) \Big] \end{aligned} \tag{16}$$

The modeling cost consists of two parts, an error term and entropy term linked by $2\sigma^2$. The distortion contribution of data vector $x \in R_\alpha$ is the error due to reducing the dimension of x to d_α

$$D_\alpha(x) = (x - \mu_\alpha)^T (1 - U_\alpha U_\alpha^T)(x - \mu_\alpha). \tag{17}$$

The differential entropy contribution of x is the sum of its discrete entropy contribution

$$H_\alpha(x) = -\ln \pi_\alpha + \frac{1}{2} \ln|\Lambda_\alpha/\sigma^2| + \frac{1}{2}(x - \mu_\alpha)^T U_\alpha \Lambda_\alpha^{-1} U_\alpha^T (x - \mu_\alpha) \tag{18}$$

and the log of a quantizer bin size $\frac{d}{2}\ln\sigma^2$ [17]. The $\ln\sigma^2$ term quantifies measurement uncertainty and in this respect set the resolution of the model.

The discrete entropy $H = \sum_\alpha \sum_n z(\alpha, x_n) H_\alpha(x_n)$ is the sum of the entropy associated with selecting a model component, $-\sum_\alpha \pi_\alpha \ln \pi_\alpha$, the entropy associated with coding the data within a component, $\frac{1}{2}\sum_\alpha \pi_\alpha \ln|\Lambda_\alpha/\sigma^2|$, and half the average dimension $\frac{1}{2}\sum_\alpha \pi_\alpha d_\alpha$. The average dimension comes from the Mahalanobis distance term in (18), since

$$\text{Trace}\left[U_\alpha^T(\sum_n z(\alpha, x_n)(x_n - \mu_\alpha)(x_n - \mu_\alpha)^T)U_\alpha \Lambda_\alpha^{-1}\right] = d_\alpha \tag{19}$$

Selecting the noise variance σ^2 is equivalent to setting a penalty on the entropy. Choosing an entropy penalty controls model resolution and complexity by determining both the number of components and the dimension of each component. When σ^2 is large relative to the data variance, the local dimensions are close to zero and the resulting (nearly spherical) model has only a few components. As σ^2 decreases, both the number of components and component dimensions increase. However, when σ^2 becomes small, the number of components decreases and the local dimensions approach the full dimension. As σ^2 approaches zero, the model becomes a hard-clustering version of a GMM with unconstrained covariance matrices. At most choices of noise variance, different model forms, from spherical to full covariance, can appear in a single adaptive model. This flexibility will allow us to effectively model non-stationary data.

3.2 Adaptive PCA Model Fitting

The EM procedure inspires a generalized Lloyd algorithm for minimizing the constrained cost. This algorithm iteratively optimizes the partition and model parameters to minimize modeling cost (16). To optimize the partition, each data vector is assigned to the region R_α that represents it with the lowest cost. This is equivalent to assigning a data vector to the region with the highest posterior probability (14). The partition consists of regions R_α such that

$$R_\alpha = \{x \mid D_\alpha(x) + 2\sigma^2 H_\alpha(x) < D_\gamma(x) + 2\sigma^2 H_\gamma(x) \ \forall\, \gamma \neq \alpha\} \tag{20}$$

Note that the $\ln\sigma^2$ term is the same for all components, so it does not affect partition optimization and can be ignored. The discrete entropy shifts the partition away from the minimum distortion solution by increasing the cost for components with large entropies. Components with low priors, large variances, or large dimension may have no data vectors assigned to them, in which case, they can be removed from the model. Consequently, the model conforms to the cluster structure by fitting small, low-dimensional components to the data in densely populated areas of the signal space.

We optimize the model parameters, $\pi_\alpha, \mu_\alpha, d_\alpha, U_\alpha,$ and Γ_α, by finding the values that minimize cost(16) for the current partition. The equations for the priors are

$$\pi_\alpha = \frac{1}{N}\sum_n z(\alpha|x_n) = \frac{N_\alpha}{N} \tag{21}$$

where N_α are the number of data items assigned to component α. Minimizing cost with respect to the translation vectors places each μ at the mean of its region

$$\mu_\alpha = \frac{1}{N_\alpha}\sum_{x\in R_\alpha} x \tag{22}$$

The embedding transform is constrained to be orthogonal, that is, $U^T U = \mathrm{I}$. Minimizing cost with respect to $W_\alpha = U_\alpha \Gamma_\alpha^{\frac{1}{2}}$, while meeting this orthogonality constraint, yields the relation

$$U_\alpha^T S_\alpha = \Lambda_\alpha U_\alpha^T \tag{23}$$

where $\Lambda_\alpha = \Gamma_\alpha + \sigma^2\mathrm{I}$ and the data covariance is

$$S_\alpha = \frac{1}{N_\alpha}\sum_{x\in R_\alpha}(x-\mu_\alpha)(x-\mu_\alpha)^T \tag{24}$$

Consequently, U_α and Λ_α contain the d_α leading eigenvectors and eigenvalues of the data covariance S_α, respectively. The stretching factors are $\Gamma_\alpha = \Lambda_\alpha - \sigma^2\mathrm{I}$.

To find the optimal dimensions d_α, we evaluate the change in cost due to increasing each local dimension by one. If we order the eigenvalues in Λ_α from largest to smallest, then increasing the dimension from $q-1$ to q results in a change of cost

$$\Delta\mathcal{C} = \ln\frac{\lambda_q}{\sigma^2} - (\frac{\lambda_q}{\sigma^2} - 1) \tag{25}$$

where λ_q is the q^{th} entry in Λ_α. By Jensen's inequality $\ln\theta \le \theta - 1$, therefore increasing the dimension will decrease the cost ($\Delta\mathcal{C} < 0$) until the next eigenvalue is as small as the noise variance, $\lambda_q = \sigma^2$. In addition, the model dimension must be no larger than the number of stretching values γ greater than zero. Since $\gamma_q = \lambda_q - \sigma^2$, λ_q must be greater than σ^2. These two conditions set the local dimension d_α equal to the number of eigenvalues in Λ_α greater than the noise variance σ^2.

We perform a search for the best model size, M. The next section describes three different training methods that incorporate this search for the optimal number of components. We achieved our best results by initializing the model with a large number of components and iteratively removing components until the best model size was found. After training the initial model to convergence, we record the modeling cost (16) for a separate validation set. The search process iteratively removes the least probable components, retrains, and records the modeling cost. The model with the optimal number of components has the lowest cost of those tested.

To select an appropriate noise variance, we noticed that models that contain several low-dimensional components rather than just a few high-dimensional components conform better to the data structure. Further work is needed to refine this observation into a principaled method for selecting an appropriate noise variance. However, a simple heuristic can be used to select a good value. We choose the σ^2 that gives the largest *average* model size over a set of different model initializations. At the chosen value of σ^2, we report results for the model with the lowest validation set cost. For the data we evaluated, this heuristic method selected models that conformed well to the natural cluster structure.

4 Algorithm Implementation

An important aspect of implementing the adaptive PCA algorithm is the determination of the optimal model size. For any selection of noise variance, there is some optimal number of model components. In this section, we present three model training methods that incorporate searches for this number of components. The first method uses deterministic annealing for constrained cost functions developed by Rose [18]. The second method starts training from a random initialization for a range of model sizes. The third method starts with a random initialization at a large number of components and iteratively removes the least probable component, retraining the model after each deletion. In all three cases, we retain the model that minimizes modeling cost a separate validation data set. In our work, we found that the third method produced the lowest cost and most consistent models.

4.1 Deterministic Annealing

Deterministic annealing is motivated by viewing clustering as a minimization of free energy [19]. For data X and model parameters Y with joint probability $\mathrm{p}(x,y)$, we wish to minimize the average distortion $D(X,Y) = \sum_x \sum_y \mathrm{p}(x,y)\hat{d}(x,y)$ with some distortion measure $\hat{d}$ while keeping the entropy $H(X,Y) = \sum_x \sum_y \mathrm{p}(x,y) \log \mathrm{p}(x,y)$ below some value. That is, we wish to minimize free energy $F = D - TH$, where T is a Lagrange multiplier. Minimizing F with respect to cluster assignments $z(x,y)$, yields a Gibbs distribution [18]

$$z(x,y) = \frac{\exp(-\hat{d}(x,y)/T)}{\sum_y \exp(-\hat{d}(x,y)/T)} \tag{26}$$

For adaptive PCA, our distortion function $\hat{d}$ is

$$d(\hat{x,}y) = D_\alpha(x) + 2\sigma^2(H_\alpha(x) + \frac{d}{2}\ln\sigma^2) \tag{27}$$

where D_α is given by (17) and H_α is given by (18). Substituting (27) into (26) and using (26) to expand the free energy F yields

$$F = -T\sum_x \ln \sum_\alpha \exp\Big(-\ [D_\alpha(x) + 2\sigma^2(H_\alpha(x) + \frac{d}{2}\ln\sigma^2)]/T\Big) \tag{28}$$

The Lagrange multiplier T controls clustering hardness. When $T = 2\sigma^2$, we have soft assignments and F is the log likelihood of the Gaussian mixture model associated with adaptive PCA (6). As T approaches zero, the assignments becomes hard, as in (14), and each x is assigned to a single cluster. In this hard-clustering limit, F reduces to the adaptive PCA cost function (16).

The free energy formulation of adaptive PCA (28) allows implementation of deterministic annealing using the template described by Rose [18]. It does not require the two-stage training process proposed by Meinicke and Ritter [8]. To use deterministic annealing for training an adaptive PCA model, we start with M components placed at the mean of the data plus small random perturbation. Without these perturbations, all components will remain at the global mean during the training process [8, 19]. We initialize T to twice the largest global eigenvalue of the data. Gradually reducing T with $\sigma^2 = T/2$ increases the model complexity, since the local dimension d_α increases as σ^2 decreases. We use an annealing schedule of $T_{\mathrm{new}} = 0.9\ T_{\mathrm{old}}$ and at each value of T the model is trained to convergence. At the desired entropy or noise variance, we freeze σ^2 and turn T to zero to achieve hard-clustering.

Unfortunately, deterministic annealing does not produce consistent models. The final model size is sensitive to the numbers of components used at initialization. Consequently, it was necessary to repeat the deterministic annealing process using different numbers of initial components M to find the optimal model size. In addition, the training process and resulting model are sensitive to the random perturbations introduced at initialization. To insure good models, we must investigate models from a number of different initializations. Deterministic annealing seems susceptible to the same problems as less sophisticated methods, yet has much heavier training time requirements.

4.2 Random Initialization

Random initialization is a classic and simple method for initializing parameters for EM or generalized-Lloyd algorithms. To use it for adaptive PCA training, we initialize M component means to randomly sampled training vectors. We then train the model to convergence using the adaptive PCA algorithm. We repeat this process using different numbers of initial components M and retain the model that minimized modeling cost for a separate validation data set. During the training process, some components may have no data assigned to them, in which case, they can be discarded. Hence, the final model size may be smaller than the initial number of components. This method has the advantage of being simple and fast, but the selected numbers of components varies significantly for different initializations. This model inconsistency increases the training time, as one must investigate models from many initializations to insure a good fit to the data.

4.3 Iterative Pruning

While working with generalized-Lloyd algorithms, we found it critical that the initial model represent all regions of the data space. Otherwise, some data clusters will be poorly modeled by too few components. To ensure a good initialization, we propose a simple heuristic method that starts from a large number of components, which should adequately cover the data space. We then iteratively shrink the model size, searching for the optimal number of components. We examined two methods of shrinking the model: combining the two components with smallest Kullback-Leibler distance and deleting the component with the lowest probability on a separate validation set. The second method, deleting the least probable component, produced models that better conformed to the natural cluster structure.

The search process starts with a large number of components (we found 40 to 80 worked well) with means assigned to randomly selected data vectors. This large model is trained to convergence. During the training process, some components have no data assigned to them and they can be discarded. Consequently, the trained model size may be smaller than the initial number of components. The training process then removes the least probable components, one at a time, retraining the model after each deletion. Once again, we retain the model with the lowest modeling cost. This training method produced the most consistent and accurate models with respect to model size of the three methods. As a result, the time spent searching for a good model fit is kept small.

4.4 Training Method Evaluation

We evaluated these three training methods on several artificial data sets. Here we show results for a 1000 points training set drawn from a mixture of five low dimensional Gaussians embedded in a three dimensional space. Figure 2 contains a scatterplot of the 400 point test data projected to the two leading global eigendirections.

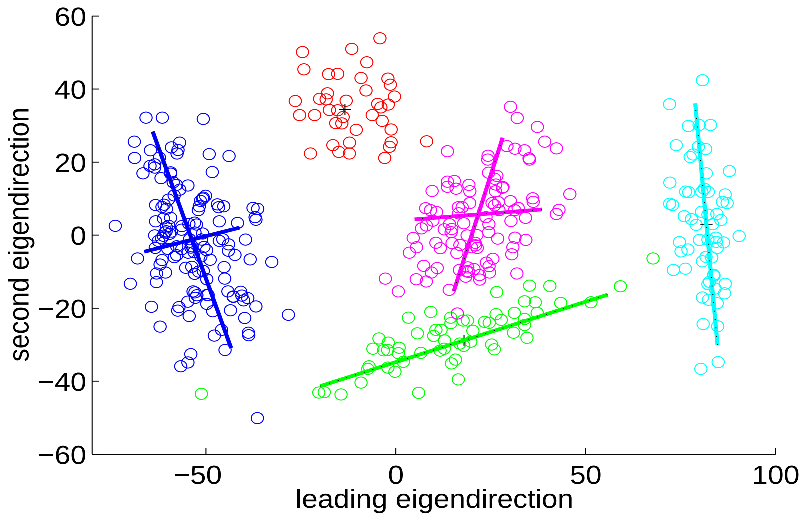


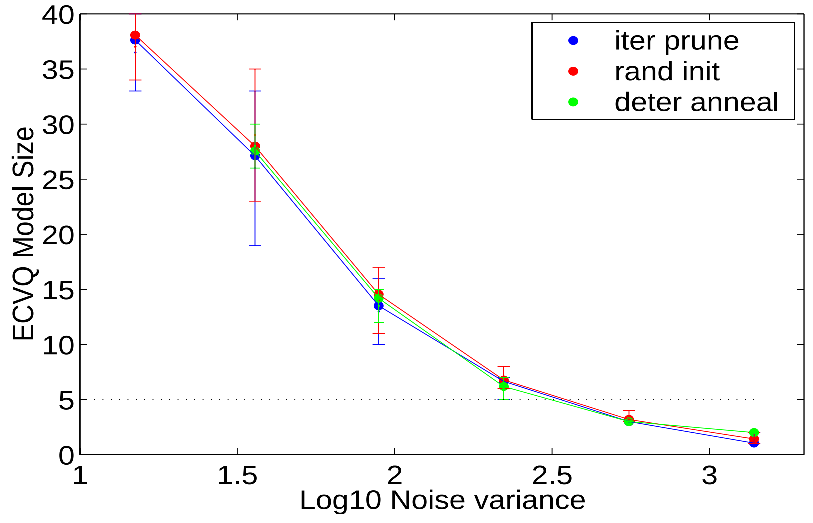
Figure 2: Mixture of Five Gaussians Test Data. Scatterplot of artificial data set used for testing. Data consists of 400 three-dimensional points drawn from a mixture of five Gaussians. Colored lines indicate the principal eigenvectors of each component and the number of lines corresponds to the dimension. Data is projected to the two leading eigendirections.

Our two evaluation criteria for these methods were how closely the model size matched the number of generating components and how much the model size varied between different initializations. We trained both entropy-constrained vector quantizers [20] and adaptive PCA models using all three methods. Figure 3 shows the average, maximum, and minimum model sizes for 25 different initializations for the random and iterative pruning methods and for 10 different initializations for the deterministic annealing method. Fewer initializations were performed for the deterministic annealing method due to the long training times.

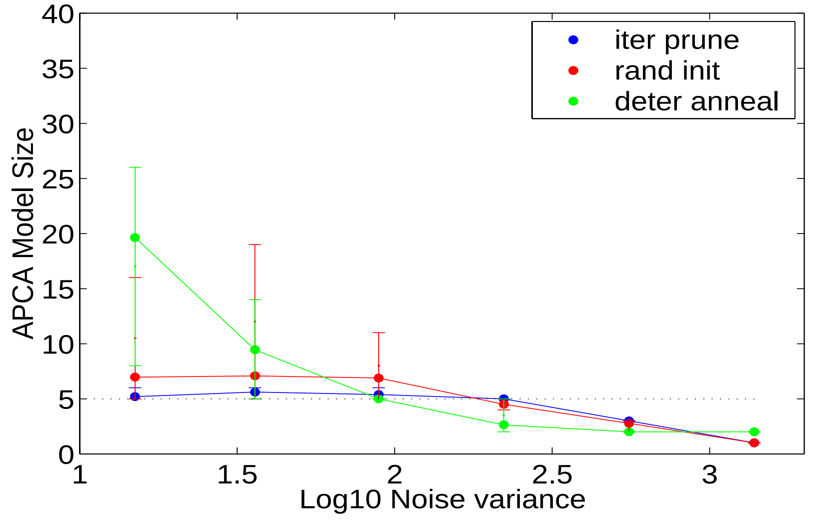
For the vector quantizer, all three methods had similar average model size and the models produced by deterministic annealing have lower variability than those from the other two methods. At low noise variances, however, the vector quantizer model sizes are much larger than the true size of five. For these spherical models, the noise variance sets the component variance or size. Consequently, when the noise variance is small, it takes many components to cover the data space.

For adaptive PCA, all three methods produce models with similar numbers of components at high noise variances. At lower noise variances, the deterministic annealing and random initialization methods produce models with too many components. In addition, the model size varied widely for different initializations. In contrast, the iterative pruning method produces models of size five or six, a good match to the true model size. Figure 3 shows examples of five and six component models. Each model component matches one of the generating clusters in Figure 2 and no components bridge multiple clusters.

We also found that our iterative pruning method improves the quality and consistency of full-covariance models. We trained hard-clustering versions of full covariance GMM on this mixture of five Gaussians data using both random initialization and iterative pruning. Since this is low dimensional artificial data, we can generate enough data to fit accurate full-covariance models. For small model sizes (less than ten components), iterative pruning produced models that better matched the natural cluster structure of the data. The models were similar for larger model sizes. Figure 5 contains scatterplots that show the match between model components and data. The model developed using random initialization contains components that span natural clusters, whereas the model developed using iterative pruning matches the natural cluster structure. For the rest of the experiments presented in this paper, we use our iterative pruning method for model training, since it produces better quality and more consistent models than those developed from random initializations.

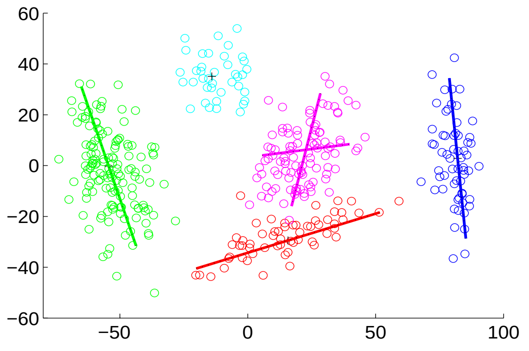


(a) ECVQ

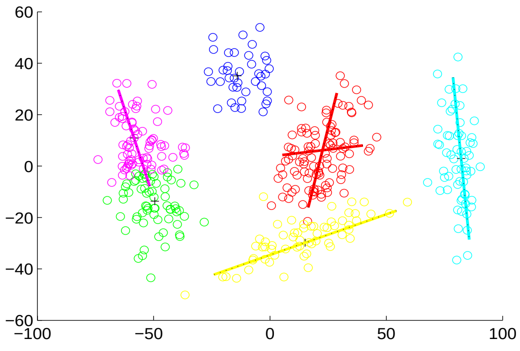


(b) APCA

Figure 3: Selected model size for different training methods. Plot (a) shows model size for entropy-constrained vector quantizer and plot (b) for entropy-constrained adaptive PCA. Models were trained using deterministic annealing (green), random initialization (red) and iterative pruning (blue).

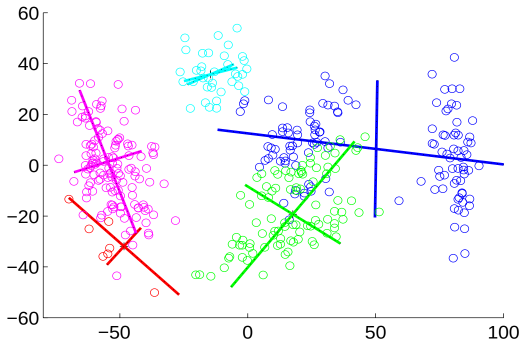


(a) Five Components

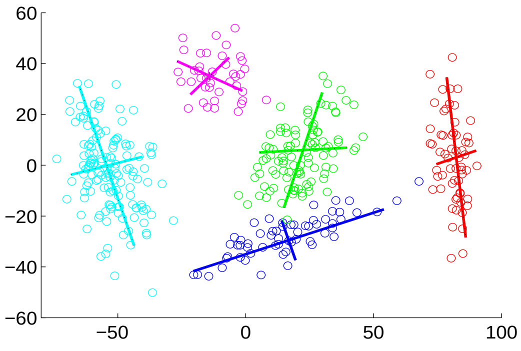


(b) Six Components

Figure 4: Clustering with Adaptive PCA. Two adaptive PCA models, one with five and one with six model components. Colors indicate assignment of data points to model components. Components conform to the natural cluster structure without bridging clusters. In the right-hand scatterplot, one large clusters is represented by two components.



(a) Random Initialization



(b) Iterative Pruning

Figure 5: Hard-Clustering GMM Models with Different Training Methods. Scatterplots show the assignment of data points to model components where each model component is represented with a different color. Scatterplot (a) shows a model initialized with five randomly selected data vectors. Scatterplot (b) shows a model initialized with forty randomly selected data vectors followed by iterative pruning down to five components. The model developed via iterative pruning closely matches the natural clusters.

5 Evaluation

We compare the modeling performance of our entropy-constrained adaptive PCA algorithm (APCA) to an entropy-constrained VQ (ECVQ) [20] and a hard-clustering version of a full covariance GMM (HGMM). When the model noise variance is large, APCA discards all dimensions and reduces to ECVQ. When the model noise variance becomes small, APCA retains all dimensions and fits the full covariance matrices to data like HGMM. Consequently, these two methods provide bounds on the modeling behavior of APCA. When there is sufficient training data, we expect the HGMM algorithm to provide the best match between model and data. However, when training data is sparse, the APCA algorithm should be less susceptible to overfitting. In this latter case, we expect the APCA models to match unseen test data better than HGMM models.

5.1 Evaluation Criteria

In order to evaluate our APCA algorithm, we wish to quantify how well the resulting model represents *true* data structure. For low dimensional data, we can determine how well the model matches the natural cluster structure, by visually evaluating the assignment of data to model components. To model quality quantitatively, we evaluate both the ability of the model to correctly classify the test data and how closely the number of components matches the number of data clusters.

We measure classification ability using the conditional entropy of the *cluster* or generating class given the model component $H_p = H(clus|\alpha)$. Component impurity, H_p, measures the number of information bits required to specify the generating class when the model component is known. It is zero when each model component contains points from just one cluster. If all model components contain equal proportions of each of N clusters, $H_p = \log_2 N$.

Spherical models with many small components have good classification performance, however they provide little insight into the natural cluster structure. Consequently, we also measure model component (over)abundance using the conditional entropy of the model component given the cluster $H_a = H(\alpha|clus)$. Component abundance, H_a, measures the number of information bits required to specify the model component when the generating class is known. H_a is zero when each model component completely contains one or more clusters. If all data clusters are modeled by N equally probable components, $H_a = \log_2 N$.

Normalized mutual information combines these two aspects of model to data structure correspondence into a single metric. Mutual information between the model components, α, and clusters is given by

$$I(clus, \alpha) = H(\alpha) + H(clus) - H(\alpha, clus) \tag{29}$$

where $H(x) = -\sum_x \mathrm{p}(x) \log \mathrm{p}(x)$ is the discrete entropy. Normalizing by $H(\alpha) + H(clus)$, which is the value of $H(\alpha, clus)$ when the model components and clusters are independent, yields the normalized mutual information.

$$NMI(clus, \alpha) = 1 - \frac{H_p + H_a}{H(\alpha) + H(clus)} \tag{30}$$

When the model components and clusters match perfectly, the normalized mutual information is one (H_p and H_a are zero). It decreases to zero as the correspondance between the model and data structure decrease.

5.2 Visual Evaluation of Model Quality

To qualitatively evaluate how well a model matches the data, we visually examine scatterplots of the data that are color coded to indicate the assignment of data vectors to model components. These scatterplots reveal where multiple components are representing a single cluster and where a component covers all or part of several different clusters. Here we present clustering results on a real world data set, salinity and temperature measurements gathered in the Columbia River Estuary.

5.2.1 Columbia River Data

Sensors deployed in the Columbia River Estuary by environmental scientists at Oregon Health & Science University [21] gather information on salinity and temperature. The salinity sensors are susceptible to gradual response degradation known as bio-fouling. Recently, we developed classifiers to successfully detect this degradation during the summer months, when bio-fouling is most prevalent [22]. We are now in the process of extending these bio-fouling detectors to operate year round.

Developing robust bio-fouling detectors is complicated by normal changes in measured salinity due to fluctuating river and ocean conditions. Our current detectors incorporate temperature information to distinguish normal changes in salinity from bio-fouling. However, the relationship between measured temperature and salinity changes throughout the year, although we see similar behavior from year to year. Visual examination of time series of salinity and temperature measurements indicate that there are at least five behavioral regimes or classes.

The Columbia River data contains measurement from two sensor stations located near the mouth of the estuary. It consists of 698 measurements spanning all seasons and acquired over several years (1997 - 2001). Each measurement contains three values: salinity and temperature at the highest diurnal tidal flood and temperature at the deepest diurnal tidal ebb. The temperature measurements are normalized by the estimated difference in ocean and river temperatures. We divided the data set into three equal parts to create training, validation, and test sets. Figure 6 contains a scatterplot of the test data set. Colors indicate different classes identified from visual examination of the time series: blue is summer period, red is winter period, and cyan, orange, and green occur during spring and fall. Green indicates measurements from when the river and ocean temperatures are close together. Yellow indicates measurements taken during periods of abnormally low salinity. Cyan indicates measurements taken during periods of rapid river temperature warming or cooling.

5.2.2 River Data Analysis

We use several modeling methods to cluster the salinity and temperature data, including our entropy-constrained adaptive PCA (APCA), entropy-constrained vector quantization (ECVQ), hard-clustering full covariance GMM (HGMM), and local PCA (LPCA). Local PCA [2] partitions the data space in order to minimize dimension reduction error for some fixed target dimension. All models were trained using the iterative pruning method described previously with model sizes selected to minimize cost on the *validation* set. We developed models from six different initializations. For ECVQ and APCA, we report results for a noise variance of 0.5, which gives an average dimension between 1 and 1.5 for the APCA models. For the LPCA model, we set the target dimension to one. To evaluate the models, we visually compared how well model components matched data clusters. Scatterplots are from the model that had the lowest validation set cost.

The ECVQ models partition the space into many small spherical components. Figure 7a shows clustering by a ECVQ model with thirteen components and noise variance 0.5. This model does not identify regimes of good temperature and salinity correlation, consequently we found ECVQ to be a poor choice for this data analysis.

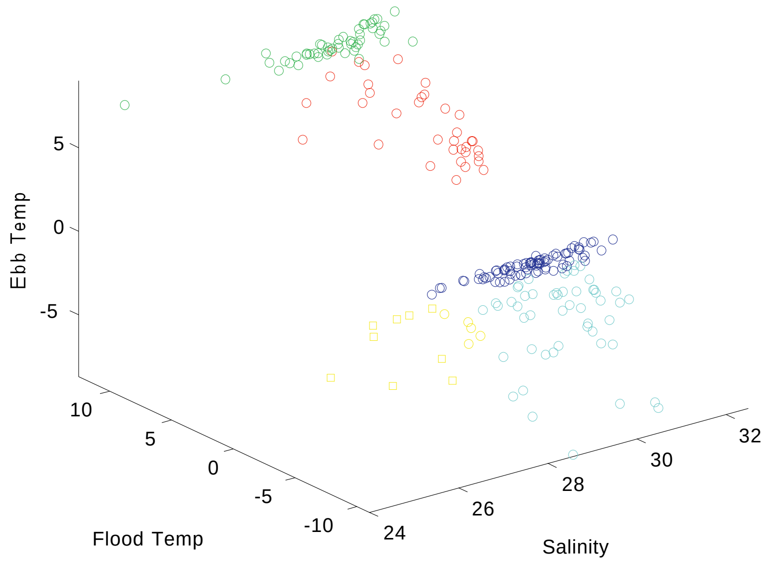
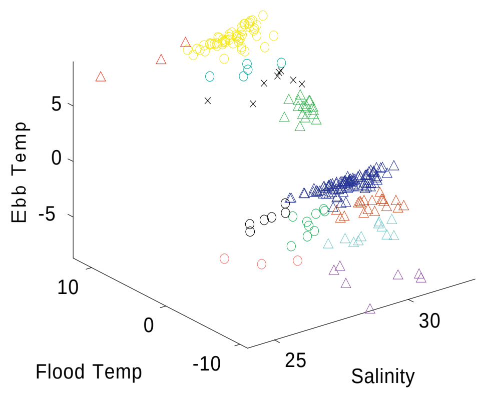


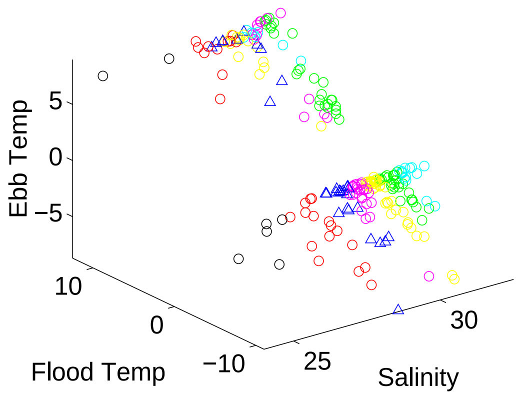
Figure 6: Columbia River Salinity and Temperature Data. Scatterplot of salinity and temperature at largest diurnal tidal flood and temperature at deepest diurnal ebb. Temperatures have been normalized by the estimated difference between the ocean and river temperatures. Colors indicate different classes identified from visual examination of the time series. The red and cyan regions may contain more than one cluster.

The LPCA models partition the space into many one-dimensional subspaces. Figure 7b shows clustering by a seven component LPCA model. Model size selection indicated that the number of components should be at least forty (largest size tested). This large a model is not instructive, so we present results for a model size of seven. The class separation provided by the LPCA model is of extremely poor quality. Model components cut across the natural cluster structure of the data. We found the LPCA modeling method to be a poor choice for clustering or data analysis.

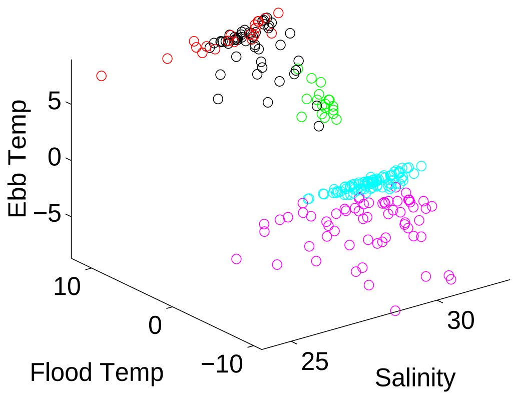
The HGMM models partition the space into five regions, but they do not match the classes show in Figure 6. Figure 7c shows clustering by the HGMM model. The summer (aqua) data is well delineated, however, the river temperature transition and low salinity classes (pink) are grouped into one cluster. The component indicated by the black circles bridges the cold temperature classes. We observed similar clustering behavior from all the HGMM models. While HGMM models selected reasonable numbers of components (4 to 7), the components did not conform well to the natural cluster.



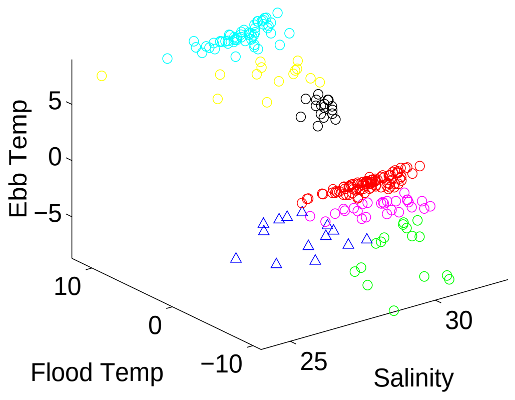
(a) ECVQ



(b) LPCA



(c) HGMM



(d) APCA

Figure 7: Clustering Examples for ECVQ, LPCA, HGMM, and APCA. Scatterplot (a) shows ECVQ model clustering with thirteen model components and noise variance of 0.5. Scatterplot (b) shows LPCA model clustering with seven components and a target dimension of one. Scatterplot (c) shows HGMM model clustering with five model components. Scatterplot (d) shows APCA model clustering with seven components and noise variance of 0.5. Each color and symbol combination represents a different model component.

Unlike the previous three model types, APCA models are well-matched to the natural cluster structure of the data. Figure 7d shows clustering by the APCA model. This model correctly identifies the summer (red), low salinity (blue), and equal river and ocean temperature (cyan) classes. It identifies two clusters within the temperature transition region (pink and green) and two clusters within the winter region (yellow and black). Of the methods tested, the APCA model corresponded most closely to the natural cluster structure and produced the most information about different salinity and temperature correlation regimes.

5.3 Quantitative Evaluation of Model Quality

To qualitatively evaluate how well a model corresponds to the data structure, we measure the normalized mutual information between the model components and data clusters. This metric measures how well the model classifies the data into its generating classes and how closely model size matches the true number of clusters. Here we present results from segmenting image texture data with a known number of textures. Segmenting high-dimensional texture data provides a realistic application for evaluating how the APCA algorithm performs when training data is sparse. As an added advantage, this data can be organized into a map for effective visualization of segmentation or clustering accuracy.

5.3.1 Image Texture Data

Our image texture data consists of 81-dimensional vectors (9×9 blocks) sampled from four different gray-scale textures. The textures are images of dense leaves, cloth, marble, and paper. Figure 8 shows the test map used to evaluate the models. We generated three different training files with 200, 500, and 1000 vectors, one 500 vector validation set and one 2500 vector test set. We develop all models using the iterative pruning training method described earlier. Model size was selected to minimize cost on the validation set. Rather than selecting a single noise variance, we report results for a range of noise variances to demonstrate the full range of adaptive PCA model behavior.

5.3.2 Quantitative Quality Analysis

To evaluate goodness of model fit, we measure normalized mutual information on the test data for models developed on each of the training data sets. Normalized mutual information

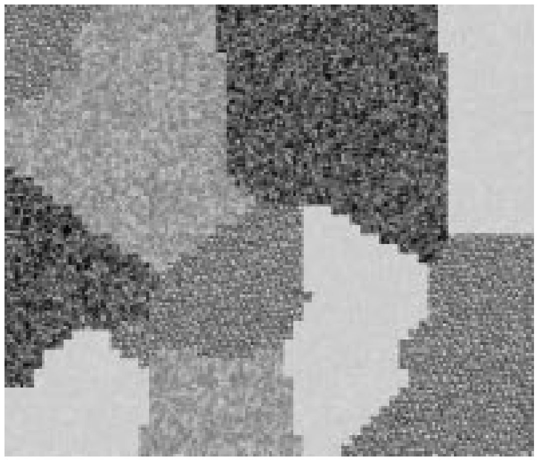


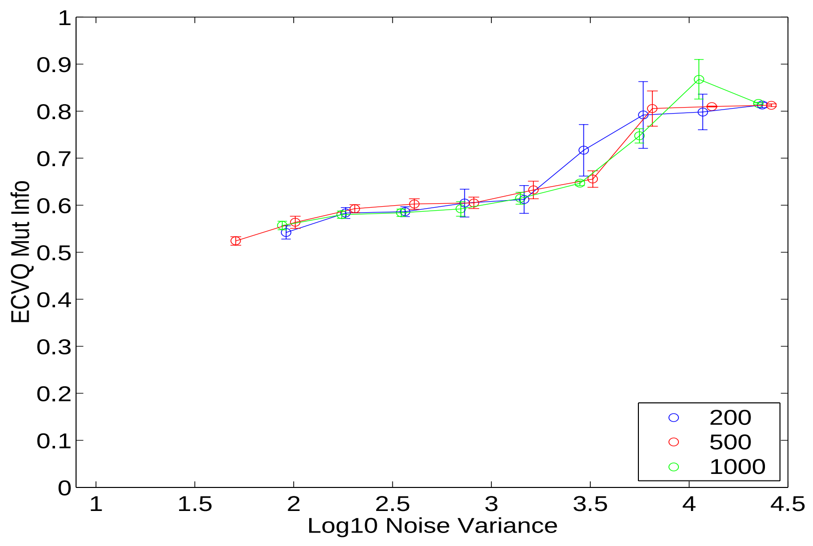
Figure 8: Texture Test Data. Data consists of 81-dimensional vectors formed into 9×9 blocks and organized into a map for visualization. Each block sampled is from one of four textures, dense leaves (dark), cloth (coarse texture), marble (gray and white), and paper (light).

incorporaates measures of component impurity, H_p, and component abundance, H_a. Figure 9 shows normalized mutual information for the different models and training set sizes.

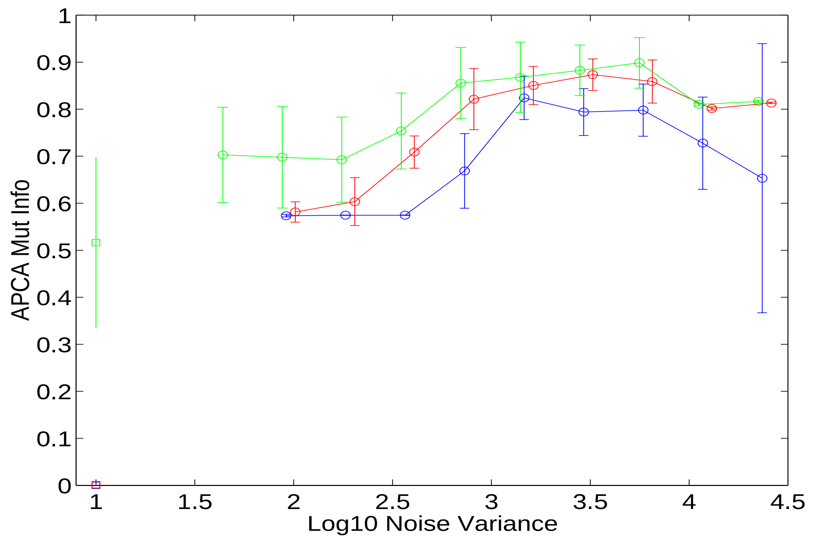
For ECVQ, model components become overabundant as σ^2 decreases, hence the normalized mutual information decreases. HGMM has very poor purity values, since the models had fewer than four components, so the normalized mutual information is low. We found that when the APCA models have noise variances in the right range, the component purity is close that that of the ECVQ models, but the models are more concise. Consequently, the values of normalized mutual information are higher than for the other modeling methods. The APCA models reveal more about the natural cluster structure of the data than either full covariance or spherical models.

5.3.3 Visual Quality Analysis

We also evaluated model accuracy by visually examining how well the model segmented the test texture image. A perfect model would use four components and attribute all the data blocks from one texture to one component. Figures 10 and 11 shows an examples of the assignment of test data blocks to model components. Each color in these images represents a different model component. For these examples, we selected the noise variance σ^2 that produced the largest average model size. The selected σ^2 was 2928 for the 200 vector training set and 2802 for the 1000 vector set.



(a) ECVQ



(b) APCA & HGMM

Figure 9: Normalized Mutual Information for different training set sizes. Plot (a) shows NMI for ECVQ and plot (b) shows NMI for APCA (circles) and HGMM (squares). HGMM results are plotted at 1 for comparison purposes. Models were trained using 1000 vector (green), 500 vector (red) and 200 vector (blue) set sizes. Error bars indicate standard deviation for ten different initializations. The 500 vector and 200 vector HGMM models each have a single component, so the NMI values are both zero.

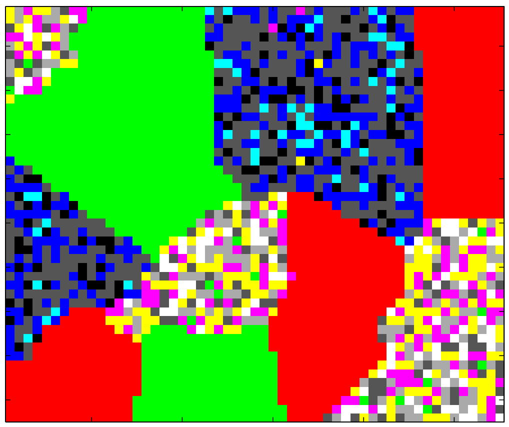
The test image segmentation results shown in Figure 10 use models developed on the 200 vector training set with $\sigma^2 = 2928$. The ECVQ model has ten components, with $H_p = 0.170$ bits, $H_a = 1.051$ bits, and $NMI = 0.75$. This model correctly classified 96.1% of the image blocks. The APCA model has four components, with $H_p = 0.202$ bits, $H_a = 0.195$ bits, and $NMI = 0.90$. It correctly identified the texture for 96.5% of the image blocks. The HGMM model (not shown) had only one component, consequently, it was unable to segment the image. The ECVQ model uses 4 and 5 components to represent the cloth and leaf textures respectively, whereas the APCA model uses a single component for each class. The APCA model segments the texture image as accurately as the ECVQ model, even though it has many fewer components.

The segmentation results in Figure 11 are for models developed on the 1000 vector training set with $\sigma^2 = 2802$. The ECVQ model (not shown) has 28 components, $H_p = 0.076$ bits, $H_a = 2.11$ bits, and $NMI = 0.65$. This model correctly classifies 98.7% of the test blocks. The HGMM model, with $H_p = 1.20$ bits, $H_a = 0.012$ bits, and $NMI = 0.57$, has only two components and segments the image poorly with 52.5% correct classification. The APCA model with $H_p = 0.050$ bits, $H_a = 0.282$ bits and $NMI = 0.92$, models the cloth texture with two components and the remaining textures with one component each. The APCA model, with five components and correct classification of 99.3%, segments the texture image more accurately than either the ECVQ or HGMM models.

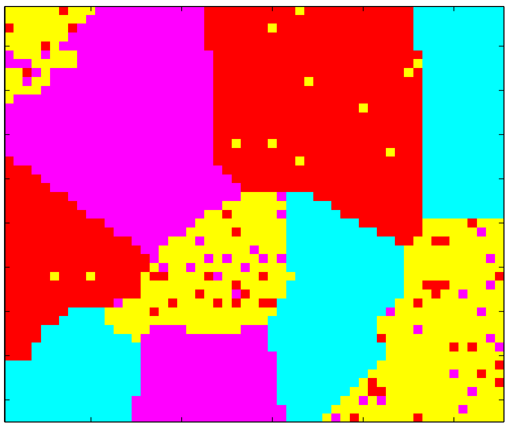
6 Selecting the Entropy Penalty

An important aspect of adaptive PCA modeling is the selection of an appropriate noise variance, or equivalently, the entropy penalty. Evaluations performed over a range of noise variances show that adaptive PCA models accurately model data by conforming to the natural cluster structure at appropriate choices of noise variance. However, when the noise variance is selected too large, the models do not have the flexibility to conform to the data structure. When the noise variance is selected too small for the available training data, the models have only a few high-dimensional components that bridge natural clusters.

Initially, we attempted to select an optimal value of σ^2 by measuring modeling cost on a validation data set, which consists of data examples not in the training set. We tested models developed using different values of σ^2. The model with the lowest validation set cost indicates the optimal noise variance.

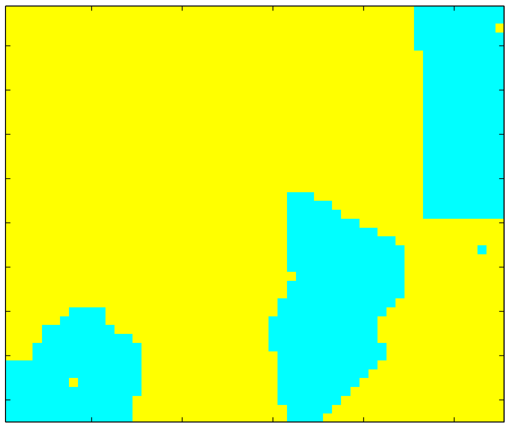


(a) ECVQ

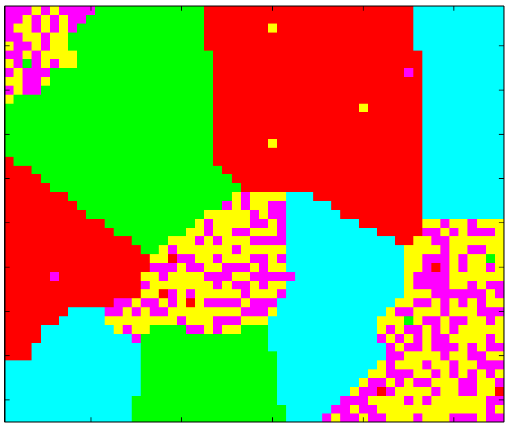


(b) APCA

Figure 10: Texture Segmentation with 200 vector training set. Map (a) on the left shows ECVQ model segmentation and Map (b) show APCA model segmentation. Both models were developed on a 200 point training set.



(a) HGMM



(b) APCA

Figure 11: Texture Segmentation with 1000 vector training set. Map (a) on the left shows HGMM model segmentation and Map (b) show APCA model segmentation. Both models were developed on a 1000 point training set.

This method for selecting σ^2 performs well on artificial data that is generated from a PCA model, but poorly on real-world data. The PCA model is based on the assumption that the measured data x is generated from a low-dimensional source s, embedded in the observation space with translation μ and transform W, and corrupted with additive Gaussian noise. Consequently, the observed data is given by

$$x = Ws + \mu + \epsilon \tag{31}$$

where $s \sim \mathcal{N}(0, \mathbf{I})$ and $\epsilon \sim \mathcal{N}(0, \rho^2\mathbf{I})$. If we order the eigenvalues of x from largest to smallest, beyond some dimension d they will plateau at the variance ρ^2 of ϵ. The validation set method selects noise variances σ^2 close to ρ^2. However, the eigenvalues of real-world data typically do not reach some plateau. Consequently, the validation set method correctly selects noise variances that are small, but which result in nearly full-dimensional and under-constrained models.

To illustrate the different behaviors of artificial and real-world data, we perform texture segmentation tasks on both types of data. The real texture data consists of 9×9 pixel blocks sampled from four texture source images, as discussed in the earlier evaluation section. The artificial texture data was generated according to the PCA model (31) using the means, eigenvectors, and eigenvalues of the source textures.

To acquire the necessary information to create the artificial textures, we decomposed each source imaged into 9×9 blocks to form 81 dimensional vectors. The translation μ is the mean of these blocks. We then removed the mean and performed singular value decomposition (SVD) on the mean-removed blocks to calculate the matrix of eigenvectors U and eigenvalues Λ. To determine the dimension, d, we retained the largest eigenvalues to account for 90% of the total variance. The variance ρ^2 for the discarded dimensions is calculated as the mean of the discarded eigenvalues. The transform $W = \hat{U}\Gamma^{\frac{1}{2}}$ where the columns of $\hat{U}$ are the eigenvectors associated with the leading d eigenvalues and $\Gamma = \Lambda - \rho^2\mathbf{I}$ are the d leading eigenvalues minus ρ^2. The variances and dimensions for the four textures are: leaves $\rho^2 = 250$, $d = 27$, cloth $\rho^2 = 268$, $d = 35$, marble $\rho^2 = 213$, $d = 52$, and paper $\rho^2 = 2.25$, $d = 28$.

For both the real and artificial data sets, we generated a 1000 vector training file, 500 vector validation file, and 2500 vector test file. For the real data, we sampled blocks from the source images starting at random offsets. To generate artificial texture blocks, we drew an s from a d dimensional unit variance Gaussian distribution, then transformed and translated it using W and μ. We then added Gaussian noise drawn from an 81-dimensional spherical

Gaussian distribution with variance ρ^2 to form a data vector x. Figure 12 shows the test data organized into maps for visualization.

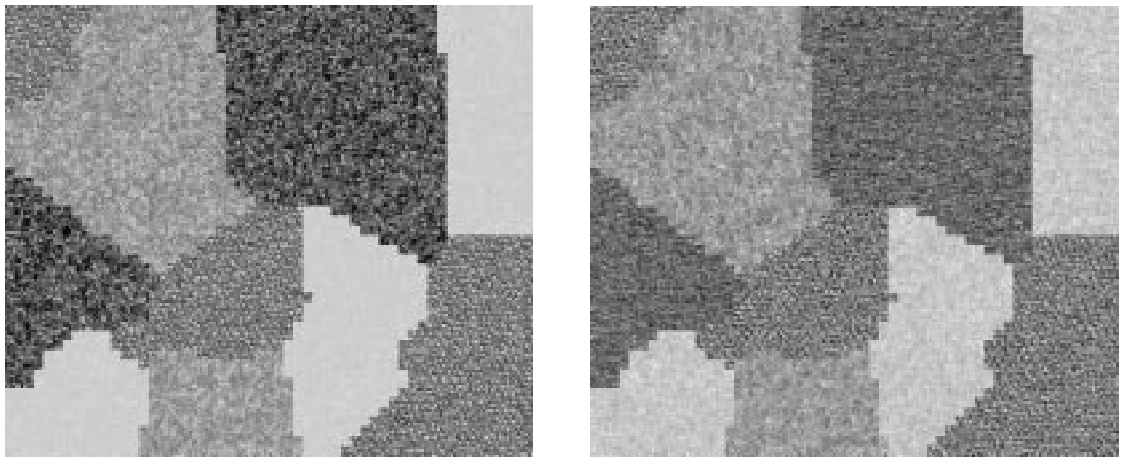

(a) Sampled (b) Artificial

Figure 12: Texture Test Data. Figure (a) shows data sampled from one of four textures, dense leaves (dark), cloth (coarse texture), marble (gray and white), and paper (light). Figure (b) shows artificial data generated from PCA model and first and second order statistics of texture images.

We trained adaptive PCA models for a range of noise variances σ^2 using the validation set to select model size. For each model, we recorded the validation set modeling cost; these are shown in Figure 13. The adaptive PCA algorithm produced accurate models with the correct number of components and good segmentation accuracy for a range of noise variances, $5000 \geq \sigma^2 \geq 150$ for the artificial texture data and $3000 \geq \sigma^2 \geq 450$ for the sampled texture data.

For the artificial texture data, the validation set cost has a minimum at $\sigma^2 = 270$ (next larger value tested was 360 and next smaller value was 202), which agrees well with the noise variances ρ^2 of three of the textures (268, 250, 213). The model at this noise variance has four components and segments the test image accurately with correct texture classification of over 95%. For the sampled texture data, the validation set cost minimum is at $\sigma^2 = 1.3$ (next larger value tested was 1.7 and next smaller value was 0.75). At this low noise variance, the model has two components: one 81 dimensional component that represents the cloth, leaves, and marble textures and one 52 dimensional component that represents the paper texture. However, at higher noise variances, $3000 \geq \sigma^2 \geq 450$, the model has four components and segments the image with better than 95% accuracy.

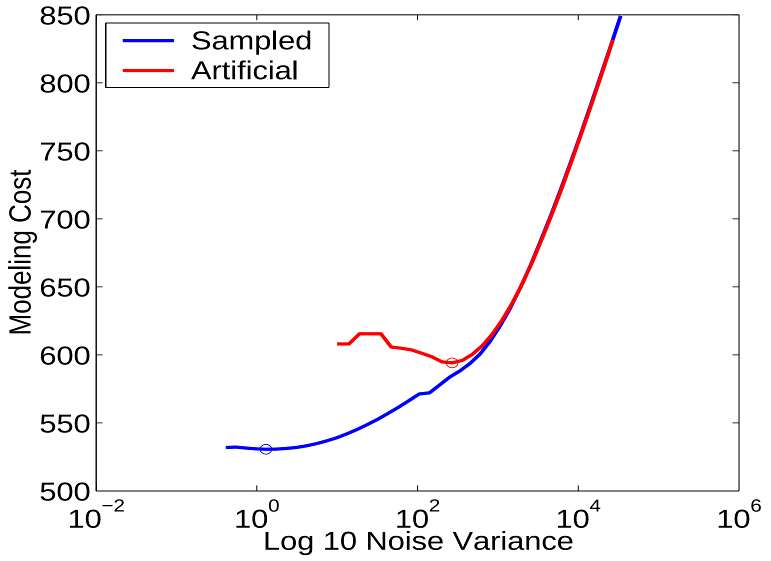


Figure 13: Validation set cost for artificial and sampled texture data. Validation set cost for range of noise variances σ^2 with artificial data cost in red and sampled data cost in blue. Circles indicate cost minima.

For entropy-constrained adaptive PCA to reach its full potential as a clustering, modeling, and analysis tool, we need a method that selects an appropriate noise variance. Selecting the noise variance to minimize modeling cost of a separate validation set results in values that are too small when the data eigenvalues do not plateau at some noise floor. Evaluations performed on texture data indicate that models conform best to the data structure when the noise variance is at or slightly below the point where the model size or component abundance is largest. Figure 14 contains a plot of component impurity H_p and abundance H_a for the real texture. This plot shows that classification ability is best at noise variances where the model size is largest. At lower noise variances, where the local dimensions are higher and there are fewer components, the classification ability of the model decreases. Investigating the relationship between model size and component dimension for different choices of noise variance may lead to effective methods of entropy penalty selection.

7 Summary

Adaptive models, which partition the signal space into regions and then model the data within each region with simple linear models, can effectively represent non-stationary data. However, standard adaptive modeling methods, such as K-means clustering and full-covariance

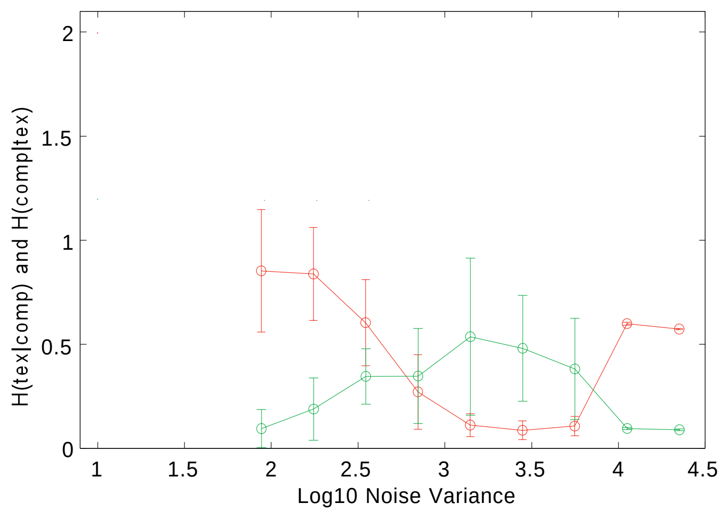


Figure 14: Component impurity (red) and abundance (green) for real texture data for range of noise variances σ^2. Circles indicate means and errobars indicate standard deviations of ten models trained from different initializations.

GMMs have their own limitations. They either require large amounts of training data to produce robust models, like GMMs, limiting their practical usefulness or they are geometrically constrained, like K-Means clustering, which limits their ability to adjust component parameters to the data structure. In this paper, we developed a new modeling method, entropy-constrained adaptive PCA, which strikes a balance between these two methods.

Using a latent data framework, we derived a statistical model for a broad category of non-stationary data, in which the data consists of a collection of hyperplanes. From this model, we develop our adaptive PCA algorithm. Adaptive PCA adjusts each component's eigenvectors, eigenvalues, and *dimension* to the local data structure. In addition, an entropy penalty provides complexity control, which allows accurate modeling of even sparse training data. Unlike some constrained modeling methods, this entropy penalty arises naturally from the statistical model.

We used our adaptive PCA algorithm for texture segmentation and for the preliminary analysis of salinity and temperature measurements from the Columbia River estuary. We evaluated how well adaptive PCA models matched the natural data structure in comparison to entropy-constrained VQs, a hard-clustering version of a full covariance GMM, and local PCA. Spherical models, such as VQs, use many small clusters to model the data. While such models can classify the data accurately, they provide little insight into the data structure. Local PCA produces consistently poor clusters that cut across the natural data structure. Hard-clustering GMM produces models with too few components that bridge the natural clusters. In contrast, adaptive PCA models consistently conform to the natural data structure with classification accuracy comparable to spherical models that have many more components.

An outstanding research issue concerns the selection of an appropriate entropy penalty via the noise variance. The noise variance should be relatively high when data is sparse and lower when data is abundant. We attempted to select the noise variance by determining the value that minimized the cost for a validation data set. However, this selection resulted in models with nearly full covariance matrices. Consequently, these models were under-constrained and exhibited the same poor modeling behavior as full covariance models. Similar validation set methods used by Meinicke and Ritter [8] also resulted in models with nearly full dimension.

Evaluation of component impurity and abundance for models with different noise variances suggest a different approach for noise variance selection. Models conform best to the natural data structure at noise variances where the component abundance is highest. The best models contain several low-dimensional components rather than very few high-dimensional components. Further work is needed to refine this observation into a theoretically motivated way of selecting an appropriate noise variance.

Acknowledgements

This work was supported under NASA contract NCC 2-1264 and NSF grants ECS-9976452 and CCR-0082736.

References

[1] D. Ormoneit and V. Tresp. Improved gaussian mixture density estimates using bayesian penalty terms and network averaging. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 542–548. The MIT Press, 1996.

[2] N. Kambhatla and T. K. Leen. Optimal dimension reduction by local PCA. *Neural Computation*, 9(7):1493–1516, 1997.

[3] G. Hinton, M. Revow, and P. Dayan. Recognizing handwritten digits using mixtures of linear models. In Tesauro, Touretzky, and Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 1015–1022. MIT Press, 1995.

[4] R. Dony and S. Haykin. Optimally adaptive transform coding. *IEEE Transactions on Image Processing*, 4(10):1358–1370, 1995.

[5] M. Tipping and C. Bishop. Mixture of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–483, 1999.

[6] Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Department of Computer Science, University of Toronto, 1996.

[7] C. Archer and T.K. Leen. Optimal dimension reduction and transform coding with mixture principal components. In *Proceedings of International Joint Conference on Neural Networks*, July 1999.

[8] P. Meinicke and H. Ritter. Resolution-based complexity control for gaussian mixture models. *Neural Computation*, 13(2):453–475, February 2001.

[9] C. Archer and T.K. Leen. From mixtures of mixtures to adaptive transform coding. In Leen, Dietterich, and Tresp, editors, *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[10] C. Archer and T.K. Leen. The coding-optimal transform. In Storer and Cohn, editors, *Proceedings Data Compression Conference*. IEEE Computer Society, March 2001.

[11] A. Basilevsky. *Statistical Factor Analysis and Related Methods*. John Wiley and Sons, Inc., 1994.

[12] S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):306–345, 1999.

[13] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[14] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford Press, 1995.

[15] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic, 1998.

[16] G. Golub and C. Van Loan. *Matrix Computations*. John Hopkins University Press, 1989.

[17] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley and Sons, Inc., 1991.

[18] K. Rose, E. Gurewitz, and G. Fox. Constrained clustering as on optimization method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):785–794, 1993.

[19] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[20] P. Chou, T. Lookabaugh, and R. Gray. Entropy-constrained vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):31–41, 1989.

[21] A. Baptista, M. Wilkin, P. Pearson, P. Turner, C. McCandlish, and P. Barrett. Costal and estuarine forecast systems: A multipurpose infrastructure for the Columbia river. *Earth System Monitor*, 9(3), 1999.

[22] C. Archer, A. Baptista, and T. K. Leen. Fault detection for salinity sensors in the Columbia estuary. *submitted to Water Resources Research*, 2002.