A Framework for Representing Non-Stationary Data with Mixtures of Linear Models

Cynthia Archer
B.S. Electrical Engineering, University of Illinios (1982)
M.S. Computer Science, Oregon Graduate Institute (1998)

A dissertation submitted to the faculty of the
OGI School of Science & Engineering
Oregon Health & Science University
in partial fulfillment of the
requirements for the degree
Doctor of Philosophy
in
Computer Science and Engineering

October 2002

The dissertation "A Framework for Representing Non-Stationary Data with Mixtures of Linear Models" by Cynthia Archer has been examined and approved by the following Examination Committee:

Todd K. Leen
Professor
Computer Science and Engineering
Thesis Research Adviser

Andrew Fraser
Associate Professor
Systems Science
Portland State University

David Basterfield
Assistant Professor
Computer Science and Engineering

Eric Wan
Associate Professor
Electrical and Computer Engineering

Dedication

To my children, Isaac and Iris

Contents

List of Figures

Abstract

A Framework for Representing Non-Stationary Data with Mixtures of Linear Models

Cynthia Archer

Supervising Professor: Todd K. Leen

In this thesis, we present a latent data framework that facilitates formalizing observations of data behavior into statistical models. Using this framework, we derive two related models for a broad category of real-world data that includes images, speech data, and other measurements from natural processes. These models take the form of constrained Gaussian mixture models. Our statistical models lead to new algorithms for adaptive transform coding, a common method of signal compression, and adaptive principal component analysis, a technique for data modeling and analysis.

Adaptive transform coding is a computationally attractive method for compressing non-stationary multi-variate data. A classic transform coder converts signal vectors to a new coordinate basis and then codes the transform coefficient values independently with scalar quantizers. An adaptive transform coder partitions the data into regions and compresses the vectors in each region with a custom transform coder. Prior art treats the development of transform coders heuristically, chaining sub-optimal operations together. Instead of this ad hoc approach, we start from a statistical model of the data. Using this model, we derive, in closed form, a *new* optimal linear transform for coding. We incorporate this transform into a new transform coding algorithm that provides an optimal solution for non-stationary signal compression. We evaluate our adaptive transform coder on the task of image compression. Our results show that a single adaptive transform coder can compress database images with quality comparable to or better than a set of current state-of-the art coders customized to each image in the database.

Adaptive principal component analysis (PCA) is an effective modeling tool for high-dimensional data. Classic PCA models high-dimensional data by finding the closest low-dimensional hyperplane to the data. Adaptive or local PCA partitions data into regions and performs PCA on the data within each region. Prior art underestimates the potential of this method by requiring a single global target dimension for the model hyperplanes. We develop a statistical model of the data that allows the target dimension to adjust to the data structure. This formulation leads to a *new* algorithm for adaptive PCA, which minimizes dimension reduction error subject to an entropy constraint. The entropy constraint, which derives naturally from the probability model, effectively controls model complexity when training data is sparse. We evaluate our adaptive PCA models on two tasks; exploratory data analysis of salinity and temperature measurements from the Columbia River estuary and texture image segmentation. Our results show that entropy-constrained adaptive PCA conforms to the natural cluster structure of data better than state-of-the-art modeling methods.

Chapter 1

Introduction

Signal classification, or recognition, includes a wide range of information processing problems. Included in this field are diverse and compelling topics such as speech recognition, detection of machinery faults, image compression, and medical diagnosis. Computer-based solutions to these problems have generally proved difficult since the data are highly complex and closed form solutions rarely exist. Common approaches to signal classification make substantial simplifying assumptions. One such assumption is that signals are wide-sense stationary, that is, their first and second order statistical properties do not change across time or space [Hay91]. However, many signals of practical interest are not stationary. In digital images, different image regions, e.g. a region showing a tree vs. a region showing a human face, exhibit different statistical properties. In human speech, statistical properties vary between phonemes.

Solution strategies for non-stationary signal classification generally fall into one of two categories: development of complex non-linear models that capture the changing signal characteristics over the whole data space, or development of a collection of simple linear models that assume the signal is wide-sense stationary in small regions of the data space. Non-linear principal component analysis (PCA) for dimension reduction [Kra91, DC93], implemented with multi-layer auto-associative neural networks, is an example of the first approach. Local PCA for dimension reduction [KL97], which partitions the signal space into regions and performs classic PCA in each region, is an example of the second approach. The second approach has the advantage that a collection of linear models is less complex to develop yet performs better than the non-linear approach.

1.1 Overview

Adopting a statistical framework provides an effective approach to these problems, as these signals are often well characterized with mixtures of probability densities. In order to develop appropriate models, we first develop a statistical framework that allows us to formalize observations of signal behavior into a probability model of the data of interest. A latent data framework is an effective tool for describing data behavior and making model choices. In this framework the observed or measured signal is derived from some simple underlying distribution. This *latent* data is mapped to the observed signal space with some linear transformation and distorted with noise. Different choices of latent data distribution and mapping produce different statistical models of the data.

Using this latent data framework, we derive models for a broad category of real-world data that includes images, speech data, and other measurements of natural processes. Data of this sort consist of collections of several distinct low-dimensional patterns embedded in the high-dimensional observation space. These patterns can be represented mathematically as low-dimensional hyperplanes. We develop two different latent data formulations that result in constrained Gaussian mixture models (GMM) for data that is comprised of low-dimensional hyperplanes. The first has ties to data compression and leads us to an algorithm for optimal adaptive transform coder design. Adaptive transform coding is a computationally attractive method for compressing non-stationary, multivariate signals. The second has ties to local and probabilistic formulations of PCA. From this second model, we develop an algorithm for entropy-constrained adaptive PCA, which can be used for data modeling and analysis. Both the adaptive transform coding and adaptive PCA algorithms improve on comparable state-of-the-art methods for compression and modeling.

1.2 Adaptive Transform Coding

Transform coders provide a lower complexity alternative to vector quantization. They are typically used for high bit-rate compression of multidimensional signals such as images. The Joint Photographic Experts Group (JPEG) standard for image compression incorporates transform coding [Wal91].

A transform coder converts signal vectors to a new coordinate basis in order to reduce statistical redundancy between vector components. Separate scalar quantizers then code each of the transform coefficients. This combination of redundancy removal and efficient quantization produces a compressed representation but adds distortion. The goal of transform coding is to minimize coding distortion while reducing the signal representation below some target size.

Transform coders use one of two types of scalar quantizers: fixed-rate or entropy-constrained. Fixed-rate quantizers use the same number of bits to code every signal vector. Entropy-constrained or variable-rate quantizers adjust the number of bits used to code a signal vector according to its entropy. For the same amount of coding distortion, variable-rate quantizers provide more compression than do fixed-rate quantizers and are generally preferred. However, fixed-rate coding is more robust for applications requiring transmission of the compressed signal over noisy channels. For more information on signal compression, see Gersho and Gray's text [GG92].

Classic transform coding assumes that the correlations between signal vector components are the same everywhere in the signal space. However, most signals of interest, e.g. digital images, are non-stationary. To capture these local variations and thereby reduce distortion in the compressed representation, adaptive transform coding methods partition the signal space into regions and compress the signal vectors in each region with a separate, unique transform coder. However, adaptive transform coders require substantial space in which to store the coder parameters. This overhead makes adaptive transform coding ineffective for the compression of individual signals. Consequently, adaptive transform coding is limited to the compression of databases of related signals, since the overhead can be amortized over the entire database.

Much prior art treats the problem of designing adaptive transform coders heuristically, chaining sub-optimal operations together in a somewhat ad hoc manner. For example [CS77, DH95, TB99] all employ partial optimization, combining sub-optimal space partitioning methods with the PCA transform and (sometimes) optimally designed quantizers. While the PCA transform is traditionally considered the best linear coding transform, it only minimizes coding error when the data density is Gaussian.

Our development is unique in that we establish a statistical model for transform coding that leads to a generalized Lloyd algorithm for both global and adaptive transform coder design. An essential part of our work is a new orthogonal transform, the coding optimal transform (COT), that minimizes coding error. Our new design algorithm minimizes compression distortion by concurrently optimizing the partition of the signal space, the local transforms, and quantizers. We present this development in three chapters: evaluation of the coding optimal transform for global transform coding, evaluation of coding optimal partition using PCA based, fixed-rate, adaptive transform coders, and evaluation of entropy-constrained adaptive transform coding on a realistic image compression application.

1.3 Adaptive PCA

Global PCA models high-dimensional data by finding the "closest" low-dimensional hyperplane to the data. PCA minimizes the mean-squared dimension reduction error or reconstruction distance between example data and this hyperplane. To model non-stationary data, adaptive or local PCA methods partition data into regions and perform PCA on the data within each region. Adaptive PCA has the potential to be an effective tool for data modeling in situations where there is insufficient training data to develop full covariance models, yet simple spherical models provide too little modeling flexibility.

Despite their success, previous authors [KL97, HRD95, TB99] under-utilize the potential of adaptive PCA models by choosing a *single global* target dimension. This constraint neglects the variability of intrinsic data dimension that is observed in real world data. Our development is unique in that we develop a statistical model of the data that permits the local dimension to vary. This formulation leads to a new algorithm for adaptive PCA, which minimizes reconstruction distance subject to an entropy-constraint. The entropy-constraint is not introduced in an ad hoc manner, but is naturally derived from the probability model. This constraint can be used to control model complexity when training data is sparse. In addition, it allows the local dimension to adjust to the data structure. Consequently, different model forms, from spherical to full covariance, can appear in a single adaptive model. This flexibility allows us to effectively model non-stationary data.

1.4 Organization

This thesis is organized into three primary sections: the statistical framework, adaptive transform coding, and adaptive PCA. Chapter two describes our statistical framework based on a latent data description of the signal of interest. Chapters three, four, and five cover our adaptive transform coding work. Chapters six and seven cover our work with variable dimension and entropy-constrained adaptive PCA.

1.4.1 Statistical Framework (chapter two)

In this first chapter, we present a latent data framework from which we derive our adaptive transform coding and adaptive PCA algorithms. This framework is an extension of that presented by Basilevsky [Bas94], Tipping and Bishop [TB99], and Roweis and Ghahramani [RG99]. This statistical framework can be used to derive a number of common density models and related signal processing algorithms. We present three example algorithms: K-means clustering, entropy-constrained vector quantization, and global PCA.

1.4.2 Coding Optimal Transform (chapter three)

A significant contribution of our transform coding work is the coding optimal transform (COT), which minimizes coding error rather than some other cost function, such as dimension reduction error. This transform has apparently not been discussed in the compression literature, although developing an optimal transform coder design algorithm is impossible without it. We develop optimal *global* transform coder design algorithms that incorporate the COT for both fixed-rate and entropy-constrained compression. In addition, we compare compression performance of COT and PCA transform based global coders on benchmark images. We found that the COT differs enough from the PCA transform to provide up to 1 dB improvement in signal-to-noise ratio (SNR).

1.4.3 Fixed-rate Adaptive Transform Coding (chapter four)

We develop the statistical model and algorithm for optimal *fixed-rate* adaptive transform coding. However, in order to facilitate comparisons with prior work, we use PCA transform based coders rather than the COT. The PCA transform does not minimize coding error and careful implementation is required to avoid convergence problems. Image compression experiments on both video images and magnetic resonance images show that fixed-rate adaptive coders can improve SNR by 2.5 to 3.0 dB compared to global coders and by 0.5 to 3.0 dB compared to other published fixed-rate methods.

1.4.4 Entropy-Constrained Adaptive Transform Coding (chapter five)

We develop the statistical model for *entropy-constrained* transform coding and the resulting generalized-Lloyd algorithm for optimal adaptive transform coder design. We then evaluate our algorithm by using it to compress a both benchmark images and database of synthetic aperture radar (SAR) images. Compression experiments on benchmark images demonstrate that our coders improve compressed image SNR by 0.25 to 1.25 dB over adaptive coders that use the Discrete Cosine (DC) and PCA transforms. Overhead considerations limit adaptive transform coding to the compression of databases of related signals Our results from the SAR database show that a single adaptive transform coder with either DCT or COTs can compress database images with SNRs comparable to those achieved by using a customized global coder for each image in the database.

Figure 1.1 summarizes the transform coding work presented in chapters three, four, and five. The first two columns list the algorithm and the associated key concepts. Each algorithm is found by minimizing a cost function with respect to the model parameters. The equation number of this cost function is listed in column three and column four gives the section where the derivation is performed. Column five gives the figure which summarizes key evaluation results.

Algorithm	Key Concept	Cost Function	Section	Results
global transform coding (TC)	mixture model	Eqn. 3.9	3.2	...
	coding optimal transform	Eqn. 3.15	3.3.2	Fig. 3.4
	fixed-rate quantizers	Eqn. 3.24	3.4.2	Fig. 3.7
	variable-rate quantizers	Eqn. 3.21	3.4.2	Fig. 3.6
fixed-rate (local) adaptive TC	optimal partition	Eqn. 4.13	4.3.1	Fig. 4.5
variable-rate adaptive TC	coding optimal transform	Eqn. 5.21	5.3.2	Fig. 5.5
	optimal partition	Eqn. 5.13	5.3.1	Fig. 5.6

Figure 1.1: Summary of Transform Coding Chapters

1.4.5 Variable Dimension Adaptive PCA (chapter six)

We briefly discuss our early work with variable-dimension local PCA that, along with our compression work, led us to consider a new approach to local or adaptive PCA. In this work, we develop a resource allocation approach to dimension selection. Our algorithm allocates dimensions to different regions so as to minimize the dimension reduction distortion while keeping the *average* dimension below some target value. Our results show that allowing the dimension to vary from region to region, instead of selecting a single global dimension, substantially reduces dimension reduction error.

1.4.6 Entropy Constrained Adaptive PCA (chapter seven)

We present the development of our statistical model for adaptive PCA and the resulting entropy-constrained algorithm. This algorithm adjusts the model parameters to minimize the dimension reduction error between the model and sample data subject to a penalty on the entropy. We evaluate the modeling quality of our entropy-constrained adaptive PCA on several data sets: a mixture of five Gaussians, measurements of salinity and temperature in the Columbia estuary, and high-dimensional image texture data. Adaptive PCA models conform to the natural cluster structure even when the data set is too small to develop accurate full-covariance models. In addition, comparisons to an entropy-constrained vector quantizer demonstrate that adaptive PCA models can classify data as accurately as spherical models while using substantially fewer components.

Figure 1.2 summarizes the adaptive and local PCA work presented in chapters

Algorithm	Key Concept	Cost Function	Section	Results
variable dimension local PCA	allocation of dimension resources	Eqn. 6.3	6.3	Fig. 6.2
adaptive PCA	mixture model	Eqn. 7.12	7.2	...
	entropy penalty for complexity control	Eqn. 7.15	7.3	Fig. 7.12, 7.13, 7.14

Figure 1.2: **Summary of Adaptive PCA Chapters**

six and seven. The first two columns list the algorithm and the associated key concepts. Each algorithm is found by minimizing a cost function with respect to the model parameters. The equation number of this cost function is listed in column three and column four gives the section where the model parameter derivation is performed. Column five gives the figure which summarizes key evaluation results.

Chapter 2

Statistical Framework

In this chapter, we present our latent data framework from which we can derive a number of common density models and signal processing algorithms. This latent framework extends that presented by Tipping and Bishop [TB99] for PCA and both Basilevski [Bas94] and Roweis and Ghahramani [RG99] for factor analysis. This framework allows us to develop explicit statistical models of the data of interest, which facilitates the selection and development of effective processing methods. We first describe the general latent data framework and how it can be used to derive signal processing algorithms. We then present three example algorithms that fit into this framework: K-means clustering [Mac67], entropy-constrained vector quantization [CLG89], and principal component analysis (PCA).

2.1 Latent Data Model

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 that is obscured by linear transformations and noise. Our goal is to recover this underlying structure in order to reduce the size of the signal representation.

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 modeled with a *simple* mixture density of the form

$$\mathrm{p}(s) = \sum_{\alpha=1}^{M} \pi_\alpha \, \mathrm{p}(s|\alpha) \tag{2.1}$$

where π_α are the mixing coefficients and $\mathrm{p}(s|\alpha)$ is often a spherical Gaussian or delta function. The location of each latent mixture component is given by the conditional

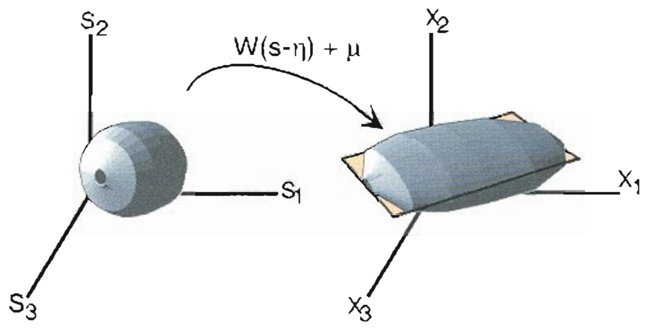


Figure 2.1: PCA Model. Structure of latent variable space, S, and mapping to observed space, X. The data density in the latent space consists of a single three dimensional Gaussian. This latent data is mapped to the observed data space by an orthogonal transform, W, which stretches and rotates the data.

mean $\eta_\alpha = \mathsf{E}[s|\alpha]$. Previous formulations (e.g. [TB99, RG99]) use a single normal distribution in the latent space, rather than this more general mixture distribution.

Linear maps with translation μ_α and rotation plus scaling transform W_α embed the latent data in the observed space, X. The embedded data is corrupted with additive Gaussian noise, $\epsilon_\alpha \sim \mathcal{N}(0, \Phi_\alpha)$ where Φ_α is diagonal for factor analysis and spherical for PCA and K-means clustering. Figure 2.1 illustrates this mapping from latent to observed space. The observed data generated from a sample s drawn from latent component α is

$$x = W_\alpha(s - \eta_\alpha) + \mu_\alpha + \epsilon_\alpha \tag{2.2}$$

with conditional densities

$$\mathrm{p}(x|s, \alpha) = \mathcal{N}(\mu_\alpha + W_\alpha(s - \eta_\alpha), \Phi_\alpha) \tag{2.3}$$

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

$$\begin{aligned} \mathrm{p}(x) &= \int \sum_\alpha \mathrm{p}(x|s,\alpha)\mathrm{p}(s|\alpha)\pi_\alpha ds \\ &= \sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x|\alpha) \end{aligned} \tag{2.4}$$

where π_α are the same mixing coefficients given in (2.1) and

$$\mathrm{p}(x|\alpha) = \mathcal{N}(\mu_\alpha, \Sigma_\alpha) \tag{2.5}$$

The form of Σ_α is constrained by the latent density and the transform W_α. We will discuss different covariance constraints and resulting models throughout this thesis.

The Expectation-Maximization algorithm (EM) [DLR77] fits parametric probability models to data by maximizing the log likelihood of the model for some training data set $\{x_n,\ n = 1 \ldots N\}$. The log likelihood of the data for this family of models is given by

$$\mathcal{L} = \sum_{n=1}^{N} \log \left(\sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x_n|\alpha) \right) \tag{2.6}$$

To simplify (2.6), we introduce the the density $z(\alpha, x_n)$ over the unknown component assignments.

$$\mathcal{L} = \sum_{n=1}^{N} \log \left(\sum_{\alpha=1}^{M} z(\alpha, x_n) \frac{\pi_\alpha \mathrm{p}(x_n|\alpha)}{z(\alpha, x_n)} \right) \tag{2.7}$$

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

$$\mathcal{L} \geq \langle \mathcal{L} \rangle = \sum_{\alpha=1}^{M} \sum_{n=1}^{N} \left(z(\alpha, x_n) \log \pi_\alpha \mathrm{p}(x_n|\alpha) - z(\alpha, x_n) \log z(\alpha, x_n) \right) \tag{2.8}$$

with equality when the $z(\alpha, x_n)$ are the posterior probabilities $\mathrm{p}(\alpha|x_n)$ [RG99, NH98]. This choice of z produces soft-clustering models.

The EM algorithm maximizes the likelihood (2.6) by iteratively finding the partition given by the posteriors (E step) and maximizing the model parameters (M step). The E step updates the posterior probabilities based on the current estimate of the model parameters.

$$\mathrm{p}(\alpha|x_n) = \frac{\pi_\alpha \mathrm{p}(x_n|\alpha)}{\sum_\beta \pi_\beta \mathrm{p}(x_n|\beta)} \tag{2.9}$$

The M step updates the model parameters so that the expected log likelihood (2.8) is maximized. The maximum likelihood estimates of the priors π_α and the component means μ_α are given by

$$\pi_\alpha = \frac{1}{N} \sum_n \mathrm{p}(\alpha|x_n) \tag{2.10}$$

$$\mu_\alpha = \frac{\sum_n \mathrm{p}(\alpha|x_n) x_n}{\sum_n \mathrm{p}(\alpha|x_n)} \tag{2.11}$$

The maximum likelihood estimates of the model covariance matrices Σ_α are found by solving

$$0 = \text{Trace}\left[\delta\Sigma_\alpha(\Sigma_\alpha^{-1} - \Sigma_\alpha^{-1}\frac{\sum_n \mathrm{p}(\alpha|x_n)(x_n-\mu_\alpha)(x_n-\mu_\alpha)^T}{\sum_n \mathrm{p}(\alpha|x_n)}\Sigma_\alpha^{-1})\right] \tag{2.12}$$

where $\delta\Sigma_\alpha$ is a matrix of small arbitrary changes in Σ_α.

2.2 Hard-clustering Algorithms

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.

The EM algorithm provides a template for deriving hard-clustering algorithms from these 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

$$z(\alpha, x_n) = \begin{cases} 1 & \mathrm{p}(\alpha|x_n) > \mathrm{p}(\gamma|x_n)\ \forall\gamma \neq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

With this hard-clustering model, the final term in the expected log likelihood (2.8) becomes zero since $z(\alpha, x_n)\ln z(\alpha, x_n) = 0\ \forall\alpha, n$. Choosing hard-clustering (2.13) and expanding $\langle\mathcal{L}\rangle$ (2.8) using (2.5) yields the cost function

$$\mathcal{C} = \sum_{\alpha=1}^{M}\sum_{n=1}^{N} z(\alpha, x_n)\left(\log\pi_\alpha - \frac{1}{2}\ln|\Sigma_\alpha| - \frac{1}{2}(x_n-\mu_\alpha)^T\Sigma_\alpha^{-1}(x_n-\mu_\alpha)\right) \tag{2.14}$$

The EM procedure inspires a generalized Lloyd algorithm that iteratively optimizes the partition and model parameters to minimize modeling cost (2.14). The hard assignments given in (2.13) lead to a partition of the data into regions R_α such that

$$\sum_{x\in R_\alpha} f(x) = \sum_{n=1}^{N} z(\alpha, x_n) f(x) \tag{2.15}$$

for any function $f(x)$. The parameter estimators are the same as the maximum likelihood estimators with the posteriors replaced by the hard assignments (2.13). The equations for the priors become

$$\pi_\alpha = \frac{1}{N} \sum_n z(\alpha | x_n) = \frac{N_\alpha}{N} \tag{2.16}$$

where N_α are the number of data items assigned to component α. The equations for the means become

$$\mu_\alpha = \frac{1}{N_\alpha} \sum_n z(\alpha | x_n) x_n = \frac{1}{N_\alpha} \sum_{x \in R_\alpha} x_n \tag{2.17}$$

The minimum cost estimates of the covariance matricies are found by solving

$$0 = \text{Trace}\left[\delta\Sigma_\alpha(\Sigma_\alpha^{-1} - \Sigma_\alpha^{-1}\frac{1}{N_\alpha}\sum_{x \in R_\alpha}(x_n - \mu_\alpha)(x_n - \mu_\alpha)^T \Sigma_\alpha^{-1})\right] \tag{2.18}$$

2.3 Algorithms from Latent Data Framework

A number of commonly used algorithms and density models fit into this latent data framework, including K-means clustering, entropy-constrained vector quantization and PCA. K-means clustering and entropy-constrained vector quantization are commonly used to design vector quantizers or to coarsely cluster data prior to other processing. PCA is a common technique for reducing the dimension of input data. In this section, we examine how these common algorithms are derived from our latent data framework.

2.3.1 K-means Clustering and Vector Quantization

Both Nowlan [Now91] and Chou [CLG89] note the correspondence between a mixture of spherical Gaussians and vector quantization (VQ) or K-means clustering. VQs code each data vector with the closest, in terms of some distance metric (e.g. Euclidean distance), of a small set of reproduction vectors [GG92]. To replicate this structure, the latent space density becomes a mixture of delta functions

$$p(s) = \sum_\alpha \pi_\alpha \, \delta(s - \eta_\alpha) \tag{2.19}$$

The single transform is the identity matrix, $W = \mathrm{I}$, and the noise is spherical $\epsilon \sim \mathcal{N}(0, \sigma_\alpha^2 \mathrm{I})$. The observed data is given by $x = s - \eta_\alpha + \mu_\alpha + \epsilon_\alpha$. The density on observed data x is therefore a mixture of Gaussians (2.4), with spherical components $\mathrm{p}(x|\alpha) = \mathcal{N}(\mu_\alpha, \sigma_\alpha^2 \mathrm{I})$. The noise variance is not fit to data, but is selected to control model complexity. In the limit that all the noise variances are identical and go to zero, the EM algorithm for fitting the mixture model reduces to K-means clustering [Mac67] or equivalently, the Linde-Buzo-Gray (LBG) algorithm for (fixed-rate) VQ design [LBG80]. Explicit hard clustering is not required, since in the E step, the posterior probabilites

$$p(\alpha|x) = \frac{1}{1 + \sum_{\gamma \neq \alpha} \exp\left(\frac{-1}{2\sigma^2}[||x_n - \mu_\gamma||^2 - 2\sigma^2 \ln \pi_\gamma) - (||x_n - \mu_\alpha||^2 - 2\sigma^2 \ln \pi_\alpha)]\right)} \tag{2.20}$$

become zero or one in the limit that the noise variance goes to zero,

$$\mathrm{p}(\alpha|x_n) \rightarrow \begin{cases} 1 & \text{if } ||x_n - \mu_\alpha||^2 < ||x_n - \mu_\gamma||^2 \ \forall\ \gamma \neq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{2.21}$$

The M step optimizes the prior probabilities π_α and component means μ_α using (2.16) and (2.17), respectively.

Entropy-constrained VQ [CLG89] is derived from a similar probability model with identical and *non-zero* noise variances for all components $\sigma_\alpha^2 = \sigma^2$, $\forall \alpha$. In this case, the observed component densities become

$$\mathrm{p}(x|\alpha) = \mathcal{N}(\mu_\alpha, \sigma^2 \mathrm{I}) \tag{2.22}$$

Expanding the cost function (2.14) using (2.22) yeilds the cost function for vector quantization

$$\mathcal{C} = \frac{1}{\sigma^2} \sum_{\alpha=1}^{M} \sum_{n=1}^{N} z(\alpha, x_n) \left((x_n - \mu_\alpha)^T (x_n - \mu_\alpha) + 2\sigma^2(-\log \pi_\alpha + \frac{1}{2} \log \sigma^2) \right) \tag{2.23}$$

The noise variance σ^2 acts as a LaGrange multiplier combining the mean-squared coding cost, $\frac{1}{N} \sum_\alpha \sum_n z(\alpha, x_n) ||x_n - \mu_\alpha||^2$, and differential entropy, $-\pi_\alpha \ln \pi_\alpha + \frac{1}{2} \log \sigma^2$. The differential entropy consists of a discrete entropy term $-\pi_\alpha \ln \pi_\alpha$ plus the log of a quantizer bin size $\frac{1}{2} \log \sigma^2$ [CT91]. Choosing σ^2 is equivalent to placing a constraint on the entropy. Note that the entropy constraint arises naturally from the probability model and is not an arbitrary addition.

The generalized-Lloyd algorithm for entropy-constrained VQ design iteratively optimizes the partition (encoder) and the model parameters (decoder) [CLG89]. The partition, or assignment of data to components, defines regions

$$R_\alpha = \{x \mid (\|x-\mu_\alpha\|^2 - 2\sigma^2 \log \pi_\alpha) < (\|x-\mu_\gamma\|^2 - 2\sigma^2 \log \pi_\gamma) \ \forall \gamma \neq \alpha\} \quad (2.24)$$

As in the fixed-rate vector quantizer above, the M step optimizes the prior probabilities π_α and component means μ_α using (2.16) and (2.17), respectively. Note that since the $\frac{1}{2}\log\sigma^2$ term does not affect the optimization of the partition or model parameters, μ_α and π_α, it can be dropped from the cost function to make the correspondence between this formulation and classic derivations exact.

2.3.2 Principal Component Analysis

Probabilistic formulations of PCA have been developed by several researchers including [TB99, Bas94, Row97]. PCA reduces the dimension of data by projecting it to the hyperplane defined by the leading eigenvectors of the data covariance matrix as illustrated in Figure 2.2. To replicate this structure, the d dimensional latent data s consists of a single Gaussian, $p(s) = \mathcal{N}(\eta, \rho^2 I)$ with mean η and variance ρ^2.

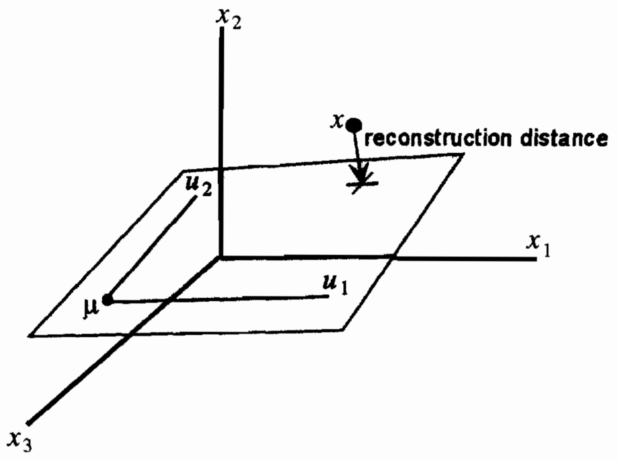


Figure 2.2: Dimension Reduction via PCA. PCA reduces the dimension of data x by projecting it to hyperplane defined by the PCA transform U. The reconstruction distance or dimension reduction error is the orthogonal distance between the data point and the hyperplane.

The observed data is generated from the latent data via

$$x = W(s-\eta) + \mu + \epsilon \quad (2.25)$$

where μ is a translation, W is a rotation and scaling, and ϵ is a noise source. The embedding transform W has two parts; an *orthogonal* transform U and a *diagonal* stretching transform Γ such that $W = U\Gamma^{\frac{1}{2}}$. Zero entries in the stretching transform Γ suppress variables so that the model dimension $q < d$. Consequently, Γ is effectively $q \times q$ and U is $d \times q$. Following the action of W, the data is smeared with spherical Gaussian noise $\mathcal{N}(0, \sigma^2 \mathrm{I})$. Figure 2.1 illustrates the data structure and mapping to the observation space.

The conditional density of x given s is

$$\mathrm{p}(x|s) = \mathcal{N}(\mu + W(s - \eta), \sigma^2 \mathrm{I}) \tag{2.26}$$

The latent density and mapping induce a density on the observed data given by

$$\begin{aligned} \mathrm{p}(x) &= \int \mathrm{p}(x|s)\mathrm{p}(s)ds \\ &= \mathcal{N}(\mu, \sigma^2\mathrm{I} + U\Gamma U^T) \end{aligned} \tag{2.27}$$

where, without loss of generality, we chose the latent variance ρ^2 to be one. We make no assumptions concerning the latent mean η.

To simplify the cost function (2.14), we first find the inverse of $\Sigma = \sigma^2\mathrm{I} + U\Gamma U^T$. Applying the Sherman-Morrison-Woodbury formula [GL89] to Σ yields

$$\Sigma^{-1} = \frac{1}{\sigma^2}(\mathrm{I} - UU^T) + U\Lambda^{-1}U^T \tag{2.28}$$

with $q \times q$ diagonal matrix $\Lambda = \Gamma + \sigma^2\mathrm{I}$. Expanding the cost (2.14) using (2.28) yields the cost function for PCA

$$\begin{aligned} \mathcal{C} = \frac{1}{\sigma^2}\Bigg(&\sum_{n=1}^{N}(x_n - \mu)^T(\mathrm{I} - UU^T)(x_n - \mu) + \\ &2\sigma^2\left[\frac{1}{2}\ln|\Lambda/\sigma^2| + \frac{1}{2}(x_n - \mu)^T U\Lambda^{-1}U^T(x_n - \mu) + \frac{d}{2}\ln\sigma^2\right]\Bigg) \end{aligned} \tag{2.29}$$

The noise variance σ^2 combines the dimension reduction distortion, $\frac{1}{N}\sum_n(x_n - \mu)^T(\mathrm{I} - UU^T)(x_n - \mu)$, and differential entropy, $\frac{1}{2}\log|\Lambda/\sigma^2| + \frac{q}{2} + \frac{d}{2}\ln\sigma^2$. The differential entropy is the sum of a discrete entropy and the log of a quantizer bin size [CT91]. The discrete entropy $\frac{1}{2}\log|\Lambda/\sigma^2| + \frac{q}{2}$ is the sum of the entropy due to coding the data with a quantizer of bin size σ and half the dimension $\frac{q}{2}$. The dimension term comes from simplifying the Mahalanobis distance $\mathrm{Trace}[\Lambda^{-1}U^T(\frac{1}{N}\sum_n(x_n - \mu)(x_n - \mu)^T)U] = q$. Choosing σ^2 is equivalent to choosing the target dimension.

The design algorithm for PCA optimizes the model parameters; there is no partitioning step, since there is only one component. The component mean μ is given by (2.17). The U transform is constrained to be orthogonal, that is, $U^TU = \mathrm{I}$. Minimizing cost (2.29) with respect to $W = U\Gamma^{\frac{1}{2}}$, while meeting the orthogonality constraint, yields the relation

$$U^TS = \Lambda U^T \tag{2.30}$$

where $S = \frac{1}{N}\sum_n(x_n - \mu)(x_n - \mu)^T$ is the data covariance [Row97]. The columns of U are the eigenvectors of the data covariance S and Λ is a diagonal matrix containing the leading q eigenvalues of S. The stretching matrix $\Gamma = \Lambda - \sigma^2\mathrm{I}$.

To find the optimal dimension q, we evaluate the change in cost due to increasing the dimension by one. If we order the entries in Λ from largest to smallest, then increasing the dimension from $q - 1$ to q results in a change of cost

$$\Delta C = \ln\frac{\lambda_q}{\sigma^2} - (\frac{\lambda_q}{\sigma^2} - 1) \tag{2.31}$$

where λ_q is the q^{th} entry in Λ. Since $\ln x \leq x - 1$, increasing the dimension will decrease the cost ($\Delta C < 0$) until $\lambda_q = \sigma^2$. In addition, the model dimension is constrained to be no larger than the number of stretching values greater than zero, so $\lambda_q > \sigma^2$. These two conditions set the optimal dimension q equal to the number of eigenvalues λ greater than the noise variance σ^2.

2.4 New Algorithms from Our Framework

In the next chapters, we develop new algorithms for adaptive transform coding and adaptive PCA using our latent data framework. Both these algorithms are based on modeling the data as a collection of hyperplanes, although the underlying latent data models differ. Adaptive transform coding is based on a *discrete* latent data model with the components constrained to lie at the vertices of a rectangular grid. Adaptive PCA is based on a mixture of spherical Gaussians latent data model. In both cases, linear transforms embed the latent data in the observation space and the data is corrupted by spherical Gaussian noise.

Our adaptive transform coding model leads to a generalized Lloyd algorithm for transform coder design. This algorithm integrates optimization of all transform coder parameters: the data partition, the transforms, and the quantizers. We describe the derivation and evaluation of our adaptive transform coding algorithm in the next three chapters.

Our adaptive PCA model leads to a generalized Lloyd algorithm for adaptive PCA. This algorithm minimizes dimension reduction distortion subject to a penalty on model entropy. The entropy-constraint provides complexity control, which allows our models to conform to the natural cluster structure of data. We describe the algorithm derivation and evaluation of adaptive PCA in the last chapters.

Chapter 3

The Coding Optimal Transform

In order to develop our adaptive transform coding algorithms, we first had to solve the problem of optimal *global* transform coding. This chapter presents our development and evaluation of a generalized-Lloyd algorithm for transform coding. Some of the material in this chapter was published at the Data Compression Conference in 2001 [AL01a].

We develop a statistical model for transform coding that leads to a new algorithm that integrates all optimization steps into a coherent and consistent framework. Each iteration of the algorithm is designed to minimize coding distortion as a function of both the transform and quantizer designs. Our algorithm is a constrained version of the generalized-Lloyd or LBG algorithm for vector quantizer design. The reproduction vectors are constrained to lie at the vertices of a rectangular grid.

A significant result of our approach is a new transform basis specifically designed to minimize mean-squared quantization distortion for both fixed-rate and entropy-constrained coding. For Gaussian distributed data, this transform reduces to the PCA transform, or equivalently, the Karhunen-Loeve transform (KLT). However, in general the coding optimal transform (COT) differs from the KLT enough to provide up to 1 dB improvement in compressed signal-to-noise ratio (SNR) on images. We describe a practical algorithm that finds the COT for a given signal. In addition, we present image compression results demonstrating the SNR improvement achieved with our algorithm relative to KLT based transform coding.

3.1 Introduction

Transform coding is a low-complexity alternative to vector quantization and is widely used for image and video compression. A transform coder compresses multi-dimensional data by first transforming the data vectors to new coordinates and then coding the transform coefficient values independently with scalar quantizers. A key goal of the transform coder is to minimize compression distortion while keeping the compressed signal representation below some target size. While quantizers are typically designed to minimize compression distortion [Llo82, FM84], this is not the case for the transform. The coordinate transform has been fixed a priori, as in the discrete cosine transform (DCT) used in the JPEG compression standard [Wal91]. The transform has also been adapted to the signal statistics using the Karhunen-Loeve transform (KLT) as in recently published transform coding work [DH95, ECG99]. These transforms are not designed to minimize compression distortion, nor are they designed (selected) in concert with quantizer development. For instance, the design goal of the KLT is to concentrate signal energy into a few components.

We develop a statistical model for transform coding. This development leads to a new algorithm for transform coder design that concurrently optimizes both transform and quantizers. Our algorithm is a constrained version of the Linde-Buzo-Gray (LBG) algorithm for vector quantizer design [LBG80]. A significant result of our approach is a new transform basis designed to minimize mean squared compression distortion. In this chapter, we derive the conditions this coding-optimal transform (COT) must satisfy to minimize distortion. In addition, we describe a simple algorithm for determining the transform. We conclude by presenting results from image compression experiments that compare the compression performance of COT-based transform coders with KLT-based transform coders.

3.2 Transform Coder Model

A transform coder converts a signal to new coordinates and then codes the transform coefficients independently of one another with scalar quantizers. One can think of a transform coder as a vector quantizer with the M reproduction vectors constrained to lie at the vertices of a rectangular grid. The grid is defined by orthogonal axes, s_J, $J = 1 \dots d$ and d sets of scalar reproduction values, one for each dimension. There are M_J possible reproduction values on the s_J axis, thus the total number of grid vertices is $M = \prod_J M_J$. Encoding a d-dimensional data vector with a vector quantizer requires $\mathcal{O}(Md)$ add/multiply operations for the distance calculations and $\mathcal{O}(M)$ compare operations. A transform coder requires $\mathcal{O}(d^2)$ add/multiply operations for the transform and naively $\mathcal{O}(\sum_J M_J)$ compare operations. However, efficient binary search techniques can be used to encode the scalar transform coefficients reducing the number of compare operations to $\mathcal{O}(\log_2 M)$.

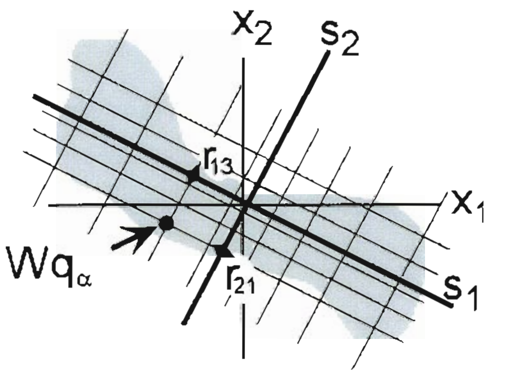


Figure 3.1: Orientation of quantizer grid in signal space. The quantizer reproduction vectors q_α, $\alpha = 1 \ldots M$, lie at the vertices of a rectangular grid. The grid is oriented to the signal vectors x (indicated by the gray area) with orthogonal transform, W.

The compression and restoration processes replace each signal vector with one of a small set of reproduction vectors. The encoder assigns the transform coefficients of a data vector to codewords. The decoder replaces each codeword with the associated reproduction value. Figure (3.1) illustrates the structure of a two-dimensional transform coder. The r values indicate the scalar reproduction values; r_{Ji} is the i^{th} value along the s_J axis. The coordinates of the reproduction *vectors*, q_α, $\alpha = 1 \ldots M$ are combinations of the scalar reproduction values $[r_{1i}, r_{2j}, \ldots, r_{dk}]^T$, $i = 1 \ldots M_1$, $j = 1 \ldots M_2$, etc. A reproduction vector q_α represents all the data vectors in region R_α of the data space. We will refer to the regions defined by the assignment of signal values to reproduction values as the *partition*.

The $d \times d$ orthogonal transform, W, defines the orientation of the quantizer grid in the data space. In the data coordinate basis, the reproduction vectors are given by Wq_α. Conversely, in the transform basis, the data vectors are $s = W^T x$.

To replicate the transform coder structure, we envision the data as drawn from a

d dimensional latent data space, S and embedded in an observation or measurement space X, also d dimensional. The density on the latent space is a mixture of delta functions

$$\mathrm{p}(s) = \sum_{\alpha=1}^{M} \pi_\alpha \delta(s - q_\alpha) \tag{3.1}$$

where the latent values, q_α, lie at the vertices of a rectangular grid as illustrated in Figure 3.2. The grid is defined by the s axes and a set of grid mark values, $\{r_{Ji}\}$, where r_{Ji} is the i^{th} grid mark along the s_J axis. There are M_J possible grid mark values on the s_J axis and the total number of grid vertices $M = \prod_J M_J$. Thus the coordinates of some q_α can be written as $[r_{1i}, r_{2j}, \ldots, r_{dk}]^T$. In addition, we constrain the mixing coefficients, π_α, to be the product of prior probabilities, p_{Ji}, so that

$$\pi_\alpha = \prod_J p_{Ji} \tag{3.2}$$

By incorporating these constraints into (3.1), we can write the density on s as product of marginal densities (for the full derivation, see Appendix A)

$$\mathrm{p}(s) = \prod_{J=1}^{d} \sum_{i=1}^{M_J} \mathrm{p}_{Ji} \delta(s_J - r_{Ji}) \tag{3.3}$$

We will use both formulations of the latent density (3.1) and (3.3) for our algorithm development.

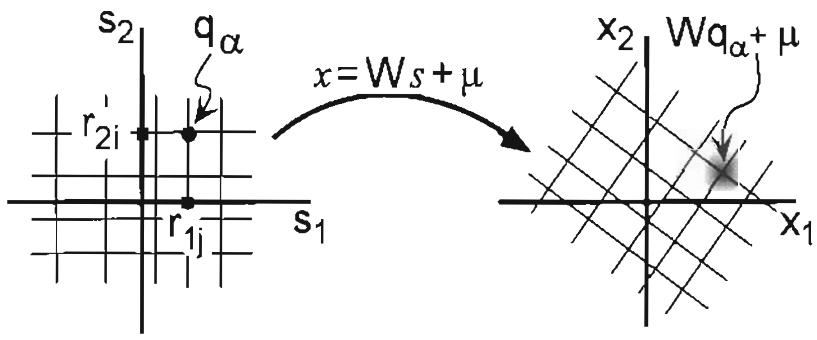


Figure 3.2: Structure of latent variable space, S, and mapping to observed space, X. The data density in the latent space consists of a mixture of delta functions where the mixture components, q_α, are constrained to lie at the vertices of a rectangular grid. This grid is mapped to the observed data space by an orthogonal transform, W, and corrupted with additive Gaussian noise.

The latent data is mapped to the observation space by an orthogonal transformation, W, and corrupted with additive Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2 \mathrm{I})$, with mean zero and variance σ^2I, as illustrated in figure 3.2. The observed data generated from a sample s drawn from latent component α is

$$x = W(s - q_\alpha) + \mu + \epsilon \tag{3.4}$$

with conditional densities

$$\mathrm{p}(x|s, \alpha) = \mathcal{N}(\mu + W(s - q_\alpha), \sigma^2 \mathrm{I}) \tag{3.5}$$

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

$$\begin{aligned} \mathrm{p}(x) &= \int \sum_\alpha \pi_\alpha \mathrm{p}(x|s, \alpha) \delta(s - q_\alpha) ds \\ &= \sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x|\alpha) \end{aligned} \tag{3.6}$$

with marginal density

$$p(x|\alpha) = \mathcal{N}(\mu + W q_\alpha, \sigma^2 \mathrm{I}) \tag{3.7}$$

The Expectation-Maximization algorithm (EM) [DLR77] fits parametric probability models to data by maximizing the log likelihood of the model for some training data set $\{x_n,\ n = 1 \ldots N\}$. The log likelihood is given by

$$\mathcal{L} = \sum_{n=1}^{N} \log \left(\sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x_n|\alpha) \right) \tag{3.8}$$

Introducing the density $z(\alpha, x_n)$ over the unknown component assignments and using Jenkin's inequality, allows us to simplify (3.8). The log likelihood $\mathcal{L}$ is bounded below by the *expected* log likelihood

$$\begin{aligned} \mathcal{L} \geq \langle \mathcal{L} \rangle &= \sum_{n=1}^{N} \sum_{\alpha=1}^{M} z(\alpha, x_n) \left[\log\ \pi_\alpha - \frac{d}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} |x_n - \mu - W q_\alpha|^2 \right] - \\ &\sum_{n=1}^{N} \sum_{\alpha=1}^{M} z(\alpha, x_n) \ln z(\alpha, x_n) \end{aligned} \tag{3.9}$$

with equality when $z(\alpha, x) = p(\alpha|x)$ is the posterior probability of component α conditioned on the data vector x [NH98].

The EM algorithm provides a template for deriving a transform coding algorithm from this probability model. To achieve hard-clustering needed for transform coding, we choose $z(\alpha, x_n)$ to be

$$z(\alpha, x_n) = \begin{cases} 1 & \mathrm{p}(\alpha|x_n) > \mathrm{p}(\gamma|x_n)\ \forall \gamma \neq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

With this hard-clustering model, the final term in the expected log likelihood (3.9) becomes zero since $z(\alpha, x_n) \ln z(\alpha, x_n) = 0\ \forall \alpha, n$. Consequently, $\langle \mathcal{L} \rangle$ reduces to the cost function

$$\mathcal{C} = \sum_{\alpha=1}^{M} \sum_{n=1}^{N} z(\alpha, x_n) \left(||x_n - \mu - W q_\alpha||^2 - 2\sigma^2 \log \pi_\alpha \right) \tag{3.11}$$

This cost function consists of two terms combined with the multiplier $2\sigma^2$: the average coding distortion $\sum_\alpha \sum_n z(\alpha, x_n) ||x_n - W q_\alpha||^2$ and the entropy $-\sum_\alpha \pi_\alpha \log \pi_\alpha$. This *entropy-constrained* cost function (3.11) is the same as that found by minimizing coding distortion subject to an *average* bit-rate constraint (e.g. [CLG89]). In the limit as the noise variance σ^2 goes to zero, *and there are a fixed number of code vectors*, we recover the cost function for *fixed-rate* transform coding.

3.3 Optimal Transform Coding

Our generalized Lloyd algorithm for transform coder design iteratively optimizes the partition (encoder) and model parameters (decoder) to minimize the coding cost (3.11). The transform coder parameters are the orthogonal transform W, the number of reproduction values in each quantizer M_J, $J = 1 \ldots d$, and the reproduction values r_{Ji}, $J = 1 \ldots d$, $i = 1 \ldots M_J$ that form the reproduction *vectors* $q_\alpha, \alpha = 1 \ldots M$. We first describe optimizing the partition followed by transform and then quantizer optimization.

3.3.1 Partition Optimization

To optimize the partition or encoder, each data vector is assigned to the reproduction vector q_α that represent is with the lowest cost. This assignment partitions the data into regions R_α such that

$$\sum_{x \in R_\alpha} f(x) = \sum_{n=1}^{N} z(\alpha, x_n) f(x) \tag{3.12}$$

for any function $f(x)$. The regions R_α are defined by subregions R_{J_i} associated with the scalar reproduction values. The partition defines subregions R_{J_i} such that each transform coefficient $s_J = W^T x$, belongs to the scalar reproduction value r_{Ji} that represents it with the lowest entropy-penalized distortion,

$$R_{Ji} = \left\{ s_J \mid (\|s_J - r_{J_i}\|^2 + 2\sigma^2 l_{Ji}) < (\|s_J - r_{Jk}\|^2 + 2\sigma^2 l_{Jk}) \; \forall k \neq i \right\} \tag{3.13}$$

where $l_{Ji} = -\log \mathrm{p}_{Ji}$ is the code word length. For fixed rate coding, the partition is given by (3.13) with σ^2 set to zero. Figure 3.3 demonstrates the transform and coding process.

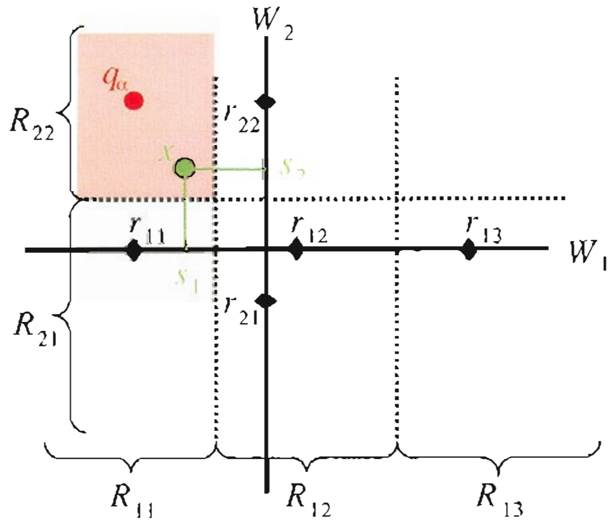


Figure 3.3: Transform Coding a Data Vector. Projecting data vector x with transform W yields coefficient values $W_1^T x = s_1$ and $W_2^T x = s_2$. The data space is partitioned into subregions with boundaries indicated by dotted lines. Coefficient s_1 is in subregion R_{11} and s_2 is in subregion R_{22}, hence x is represented by reproduction vector $q_\alpha = [r_{11}, r_{22}]^T$. The region R_α associated with q_α (shaded) is the intersection of R_{11} and R_{22}.

3.3.2 Transform Optimization

To optimize the transform, we find the center μ and orientation W of each quantizer grid that minimizes the coding cost function (5.13). The minimum cost estimators for the grid center place the grid at the mean of the data.

$$\mu = \frac{1}{N}\sum_x x \tag{3.14}$$

To optimize the transform, we find the orientation of the quantizer grid which minimizes distortion (3.11). The transform W is constrained to be orthogonal, that is $W^TW = \mathbf{I}$. The cost function to be minimized is thus

$$C = \sum_{\alpha=1}^{M}\sum_{x\in R_\alpha} \|x - \mu - \sum_{J=1}^{n} W_J q_{\alpha J}\|^2 + \sum_{K=1}^{n}\sum_{L=1}^{n} \gamma_{KL}(W_K^T W_L - \delta_{K,L}) \tag{3.15}$$

where W_J is the J^{th} column vector of W, q_K is the K^{th} coordinate of reproduction vector q, and γ_{KL} is a Lagrange multiplier. Minimizing C with respect to the transform matrix element W_{KJ} yields

$$\sum_\alpha q_{\alpha J}\sum_{x\in R_\alpha}(x-\mu)^T W_K = \sum_\alpha q_{\alpha K}\sum_{x\in R_\alpha}(x-\mu)^T W_J \tag{3.16}$$

If we define the outer-product matrix Q

$$Q = \sum_\alpha q_\alpha \sum_{x\in R_\alpha}(x-\mu)^T \tag{3.17}$$

then (3.16) requires $QW = W^TQ^T$. This symmetry condition along with the orthogonality condition uniquely defines the coding optimal transform (COT) W.

By using the conditions for the coding optimal transform, we can determine this transform for two cases of interest, Gaussian data and high-resolution coding. Gersho and Gray [GG92] and Mallat [Mal99] have shown, by using high-resolution distortion approximations, that the optimal coding transform for Gaussian data is the KLT. Using (3.16) it is possible to show that this is the case, *regardless of bit-rate.* For a alternate approach to this proof, see [GZV00]. The product of Q (3.17) and W is given by

$$QW = \sum_{\alpha=1}^{M} q_\alpha \int_{R_\alpha} s^T \mathrm{p}_s(s)\mathrm{d}^d s \tag{3.18}$$

where $s = W^T(x-\mu)$ and W is orthogonal. We need two results to show QW is symmetric when W is the KLT or PCA transform. First we note that for Gaussian $p_x(x) = \mathcal{N}(0,\Sigma)$, W diagonalizes the covariance Σ, hence $p_s(s)$ is the product of marginals

$$\mathrm{p}_s(s) = \prod_J \mathrm{p}_J(s_J) \tag{3.19}$$

Second, the reproduction values which minimize mean-squared distortion are given by

$$q_{\alpha K} = \frac{\int_{R_{\alpha K}} s_K \mathrm{p}_K(s_K) ds_K}{\int_{R_{\alpha K}} \mathrm{p}_K(s_K) ds_K} \tag{3.20}$$

where $R_{\alpha K}$ is subregion associated with R_α and the s_K axis. Substituting (3.19) and (3.20) into (3.18), it is straightforward to show that QW is symmetric, hence the KLT is the coding optimal transform when the data is Gaussian. Note that the partition (encoder) need not minimize mean squared error, so this result applies to entropy-constrained and uniform quantizers, as well as fixed-rate quantizers.

In the case of high-resolution coding, the reproduction values are so numerous and closely spaced that the data density in each region R_α is uniform, $p_x(x|x \in R_\alpha) =$ constant. When the reproduction values are given by minimum error quantizers (3.20), QW is symmetric for *any* orthogonal W. Consequently, in the high-resolution limit, distortion does not depend on the orientation of the quantizer grid.

3.3.3 Quantizer Optimization

Quantizer optimization is most conveniently performed in the transform coordinates. To rewrite the cost in terms of the transform coefficients $s_J = W^T(x - \mu)$, $J = 1 \ldots d$, we start the derivation from the product of scalars formulation (3.3) for the latent density instead of (3.1). The resulting cost function, which is equivalent to (3.11), is

$$\mathcal{C} = \sum_{J=1}^{d} \sum_{i=1}^{M_J} \sum_{s_J \in R_{Ji}} \left(|s_J - r_{Ji}|^2 - 2\sigma^2 l_{Ji} \right) \tag{3.21}$$

where the $l_{Ji} = -\log \mathrm{p}_{Ji}$ is commonly interpreted as the code word length.

To optimize the quantizer reproduction values, we adjust the number of reproduction values in each coordinate M_J and the value of each r_{Ji} to minimize the cost (3.21). Minimizing the cost (3.21) with respect to the reproduction values places each reproduction value at the mean of the transform coefficients $s_J = W_J^T(x - \mu)$ in R_{Ji}.

$$r_{Ji} = \frac{1}{N_{Ji}} \sum_{s_J \in R_{Ji}} s_J \tag{3.22}$$

where N_{Ji} are the number of transform coefficients in R_{Ji}. The entropy term does not affect this optimization, so (3.22) specifies optimal reproduction values for both fixed-rate and entropy-constrained transform coding. The prior probabilities p_{Ji} are given by

$$\mathrm{p}_{Ji} = N_{Ji}/N \tag{3.23}$$

Determining the quantizer sizes is performed differently for entropy-constrained and fixed rate transform coding. For *entropy-constrained* transform coding, selecting the noise variance or Lagrange multiplier is equivalent to selecting an entropy constraint. The entropy constraint determines the number of reproduction values M_J in each scalar quantizer. The entropy terms in (3.21) move the partition away from the minimium distortion solution, so that reproduction values with low prior probabilities may have no data items assigned to them. Reproduction values with $\mathrm{p}_{Ji} = 0$ can be removed from the coder, reducing the value of M_J. Consequently, selecting a large value for σ^2 produces small quantizers and low bit-rate coders.

For fixed-rate coding, we set the noise variance to zero and constrain the number of coding bits. The number of coding bits per block $\sum_J M_J$ is kept below some target rate B. The fixed-rate cost function is therefore

$$\mathcal{C} = \sum_{J=1}^{d}\sum_{i=1}^{M_J}\sum_{s_J \in R_{Ji}} \left(|s_J - r_{Ji}|^2\right) + \lambda\left(\sum_{J=1}^{d}\log M_J - B\right) \tag{3.24}$$

where λ is a Lagrange multiplier. Allocating the B coding bits where they minimize coding distortion the most determines the optimal values for $M_J,\ J = 1\ldots d$ [SG88, Ris91].

3.4 Implementation

The algorithm for optimal transform coder design is a constrained version of the Linde-Buzo-Gray (LBG) algorithm for vector quantizer design [LBG80]. It alternates between improving the transform and improving the quantizers until the constrained distortion measure reaches a local minimum.

3.4.1 Coding Optimal Transform Algorithm

The COT algorithm finds the orientation of the current quantizer grid that minimizes compression distortion (3.11). The quality of the final solution is sensitive to the bit allocation determined at the initial quantizer grid orientation. To insure a good starting point, we initialize W to the KL transform. At each iteration, we calculate the QW matrix from the transform coefficients and the reproduction values. To minimize distortion, we find the W that makes the QW matrix symmetric (3.16). We quantify how far the matrix is from symmetric with the sum squared difference between transposed matrix elements

$$A = \sum_{K=1}^{n-1} \sum_{J=K+1}^{n} (a_{KJ} - a_{JK})^2 . \tag{3.25}$$

where a_{KJ} is the K^{th} row and J^{th} column element of QW. We apply Givens rotations [GL89], $G(K, J, \theta)$, to minimize A. Multiplication by $G(K, J, \theta)$ applies a rotation of θ radians to the (K, J) coordinate plane. For a $n \times n$ matrix, there are $\frac{n^2-n}{2}$ such planes. Minimizing (3.25) with respect to rotation $G(K, J, \theta)$ yields a solution for θ that is quartic in $\tan\theta$. However, when the angle is small, so that $\tan^2\theta \ll 1$, the solution simplifies to

$$\tan\theta \approx \frac{(a_{KK} + a_{JJ})(a_{KJ} - a_{JK}) - \sum_{I \neq K,J}(a_{JI}a_{IK} - a_{JI}a_{IK})}{\sum_{I \neq K,J}(a_{JI}a_{IJ} + a_{KI}a_{IK}) + (a_{KK} + a_{JJ})^2 - (a_{KJ} - a_{JK})^2} \tag{3.26}$$

In image compression experiments, we consistently found that the rotation angles were small. We find the rotation angle (3.26) for each coordinate plane and apply these rotations to the current transform matrix. This process is *repeated* until $A/||QW||_F$, where $||QW||_F$ is the Frobenius norm, is less than a threshold ($A \approx 0$). The new W will orient the quantizer grid so that compression distortion is minimized.

3.4.2 Quantizer Algorithms

Quantizer optimization defines scalar quantizers that represent the data with minimal distortion given a constraint on the compressed bit-rate. To develop the quantizers, we first transform the signal vectors to the the current transform basis W. We consider both entropy-constrained and fixed-rate compression cases.

For *entropy-constrained* compression, each quantizer is trained to optimally represent the transform coefficients using an entropy-constrained quantizer algorithm [FM84]. We initialize with ten-bit uniform quantizers. If the entropy $H = \sum_J \sum_i p_{Ji} \log_2 p_{Ji}$ is too far from the target rate ($|H - B| > 0.1$ bit), we adjust the Lagrange multiplier, $\lambda = 2\sigma^2$, which enforces the rate constraint. The change in λ

is $(B - H)/\frac{\partial H}{\partial \lambda}$ where $\frac{\partial H}{\partial \lambda}$ is estimated from the previous two values of H and λ. For the k^{th} adjustment, $\frac{\partial H}{\partial \lambda^{(k)}} = (H^{(k-1)} - H^{(k-2)})/(\lambda^{(k-1)} - \lambda^{(k-2)})$ with $\lambda^{(0)} = 0$ and $\lambda^{(1)} = 1$. The quantizers are then retrained with the new λ and the entropy is re-evaluated. This process repeats until the rate constraint is satisfied.

For *fixed-rate* compression, we use optimal bit allocation [Ris91] to determine quantizer sizes. In [Ris91], Riskin defines two bit allocation methods; we use the method that does *not* require convexity of the rate-distortion function. We maintain one through ten bit quantizers for each coordinate. Each quantizer is trained to optimally represent the transform coefficients using the Lloyd algorithm [Llo82]. One then calculates the distortion for each quantizer size and coordinate. Starting from zero bits in each coordinate, one allocates one, two, or more bits at a time to the coordinate where the additional allocation will reduce the distortion per coding bit the most. This method results in an allocation of coding bits that is at or close to the desired number of bits B and that is on the convex hull of possible rate-distortions.

3.5 Experimental Results

We illustrate the difference between the KLT and COT using two dimensional data that is sampled from two intersecting Gaussian distributions: $\mathcal{N}(0, U_1^T \Sigma_1 U_1)$ with

$$U_1 = \begin{bmatrix} -.6 & .8 \\ .8 & .6 \end{bmatrix} \text{ and } \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & .16 \end{bmatrix}$$

and $\mathcal{N}(0, U_2^T \Sigma_2 U_2)$ with

$$U_2 = \begin{bmatrix} .6 & .8 \\ .8 & -.6 \end{bmatrix} \text{ and } \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & .16 \end{bmatrix}$$

Figure 3.4 contains a plot of this data overlaid with a one by two bit quantizer grid. The KLT aligns the grid along the dominant high-variance Gaussian, consequently data from the lower variance Gaussian is poorly represented. The COT rotates the quantizer grid so that the reproduction vectors better represent all the data. The compressed data signal-to-noise ratio (SNR) is 0.46 dB higher when the COT orients the quantizer.

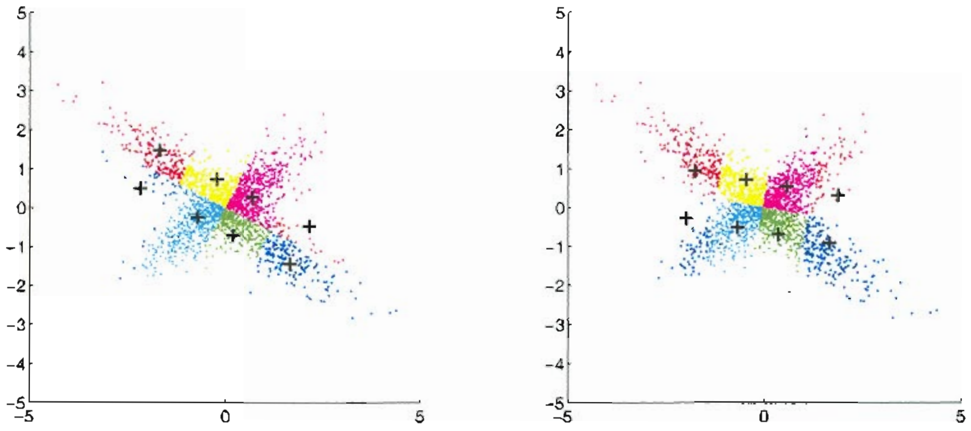
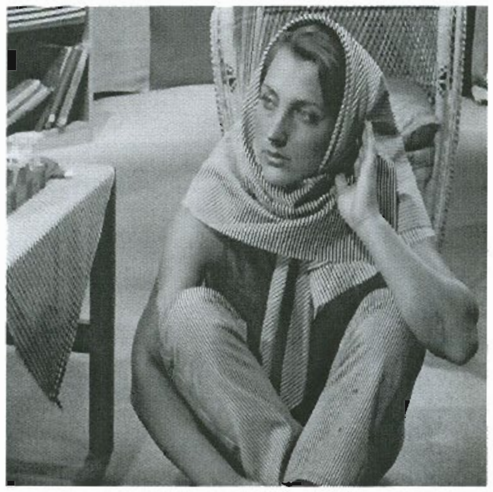

(a) KLT (b) COT

Figure 3.4: Comparison of COT and KLT. The quantizer on the left is oriented with the KLT, the one of the right with the COT. Data vectors are indicated with ·'s. and the reproduction vectors are indicated with +'s.

We also exercised our transform coders on image data. In plots 3.6 and 3.7, we show SNR results for two classic test images, Barbara and Goldhill shown in Figure 3.5. These images are available from the University of Waterloo website [oW98]. The plots in figure 3.6 are for entropy-constrained compression; entropy coding was not performed. The plots in figure 3.7 are for fixed-rate compression.

For *entropy-constrained* compression, our experiments show that using the COT instead of KLT increases SNR by 0.3 to 1.2 dB for entropies in the range of 0.25 to 1.25 bits per pixel (bpp). Of the images tested, Barbara showed the largest SNR improvement when the COT is used and Goldhill showed the smallest improvement. Other tested image types (e.g. frames from natural image video, magnetic resonance images) showed similar SNR improvements.

For *fixed-rate* compression and low bit rates, using the COT instead of KLT increases image SNR very little. The high bit-rate COT basis vectors are essentially the same as the high-variance KLT basis vectors, so when only a few coordinates are coded there is little difference in SNR. However, the SNR improvement due to using the COT increases as more coordinates are coded, since for image data the mid-variance KLT basis vectors differ from the mid and low bit-rate COT vectors. At 1.0 bpp orienting the quantizer grid with the COT instead of KLT increases SNR

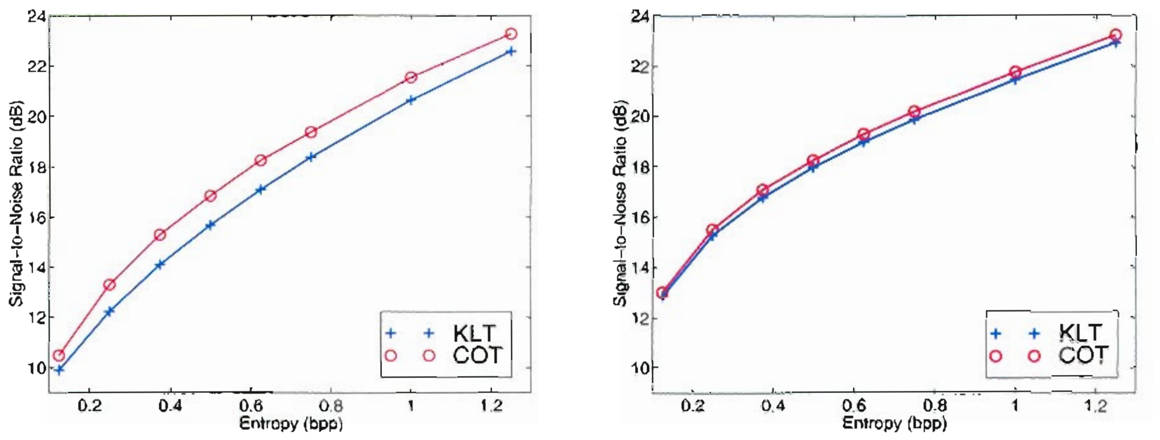

(a) Barbara

(b) Goldhill

Figure 3.5: Classic image compression benchmark images. Barbara is a photograph of a seated women wearing striped clothing. Goldhill is a photograph of a row of houses in a hillside village..

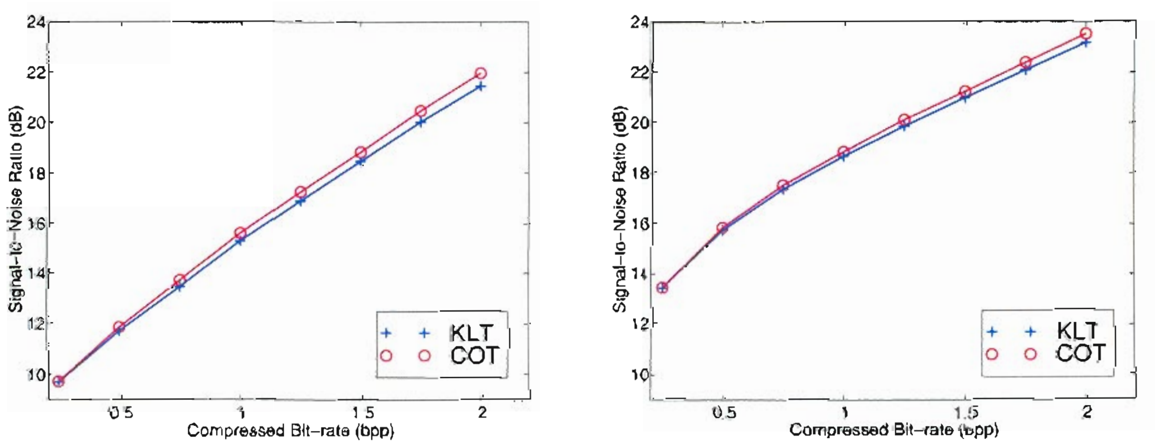
by 0.2 to 0.35 dB.

COT-based transform coding is no worse than KLT-based coding in terms of storage overhead and encode/decode time. The storage overhead, which includes storing and transmitting the transform matrix and quantizer reproduction values, is the same for both methods. Encoding and decoding times are also the same. However, in our variable-rate compression experiments, the COT-based coders required typically 3 to 4 times longer to train than did the KLT-based coders. For the Barbara image, the KLT-based coders required 130 to 135 seconds to train on a Sun SPARC Ultra2 with approximately 95% of the training time due to developing the quantizers. The COT-based coders required 225 to 680 seconds, depending on the number of training iterations. Transform optimization accounted for 30% to 60% of the training time.



(a) Barbara (b) Goldhill

Figure 3.6: Entropy-constrained compression: SNR versus entropy for Barbara and Goldhill test images.



(a) Barbara (b) Goldhill

Figure 3.7: Fixed-rate compression: SNR versus bit-rate for Barbara and Goldhill test images.

3.6 Discussion

Transform coders are often constructed by concatenating an ad hoc choice of transform with bit allocation and quantizer design. Instead, we start from a statistical model of the data from which we derive a new algorithm for transform coder design. This algorithm is a constrained version of the LBG algorithm for vector quantizer design, with reproduction vectors constrained to lie at the vertices of a rectangular grid. In addition, our derivation leads to a new transform basis, the coding optimal transform (COT), which unlike the KLT, is specifically designed to minimize compression distortion. Variable-rate image compression experiments show that using our COT instead of the KLT increases SNR by 0.3 to 1.2 dB. We have shown that the COT reduces to the KLT for Gaussian sources.

Like the KLT, the COT is a data dependent transform. Consequently, it suffers from the same drawbacks as the KLT; the transform must be calculated from the input signal and stored with the compressed signal. Because of these drawbacks, we expect COT-based transforms coders to be most effective in an adaptive or universal transform coding framework. An adaptive transform coder consists of several different transform coders, each optimized to compress a different signal type. Each signal vector is compressed using the transform coder that represents it with the least distortion. Our transform coding algorithm could be incorporated into an *adaptive* framework, such as that developed by ourselves [AL01b] or Effros, et al. [ECG99], to create a constrained LBG algorithm for adaptive transform coding.

Chapter 4

Fixed-Rate Adaptive Transform Coding

In this chapter, we investigate the application of local Principal Component Analysis (PCA) to transform coding for *fixed-rate* image compression. Some of this material was published at the Neural Information Processing Systems (NIPS) Conference [AL01b] and Neural Networks for Signal Processing (NNSP) Workshop [AL00], both in 2000.

Local PCA transform coding adapts to differences in correlations between signal components by partitioning the signal space into regions and compressing signal vectors in each region with a local transform coder. Previous researchers optimize the signal space partition and transform coders independently and consequently underestimate the potential advantage of using adaptive transform coding methods. We propose a new algorithm, derived from a statistical model of the data, that concurrently optimizes the signal space partition and local transform coders. The resulting algorithm is simply a constrained version of the LBG algorithm for vector quantizer design.

Image compression experiments show that adaptive PCA transform coders designed with our integrated algorithm compress an image with less distortion than previous related methods. We saw improvements in compressed image signal-to-noise ratio of 0.5 to 2.0 dB compared to other tested adaptive methods and 2.5 to 3.0 dB compared to global PCA transform coding.

4.1 Introduction

Compression algorithms for image and video signals often use *transform coding* as a low-complexity alternative to vector quantization (VQ). Transform coders compress multi-dimensional data by *transforming* the signal vectors to new coordinates and *coding* the transform coefficients independently of one another with scalar quantizers. One can think of a transform coder as a vector quantizer with the M reproduction vectors constrained to lie at the vertices of a rectangular grid. For d dimensional data, a transform coder requires $\mathcal{O}(d^2)$ add/multiply operations for the transform operation, whereas a vector quantizer requires $\mathcal{O}(Md)$ add/multiply operations for distance calculations. A transform coder requires $\mathcal{O}(\log_2 M)$ compare operations for encoding, much less than the $\mathcal{O}(M)$ compare operations required for vector quantization.

Noting that many types of real-world signals are not wide-sense stationary, several researchers have extended the idea of global transform coding to adapt to non-stationary signals [DH95, TB99, AL99]. In these adaptive transform coders, the signal space is partitioned into disjoint regions and a transform and set of scalar quantizers are designed for each region. In our own previous work [AL99], we use k-means partitioning to define the regions. Dony and Haykin [DH95] partition the space to minimize dimension-reduction error. Tipping and Bishop [TB99] use partitioning according to a probabilistic rule that reduces, in the appropriate limit, to partitioning by dimension-reduction error, as defined by Kambhatla and Leen [KL97]. These systems do not partition the signal space with the goal of minimizing compression distortion.

The ad hoc construction of transform coders contrasts sharply with vector quantizer design, which uses algorithms that minimize coding error. These algorithms can be derived from statistical models of the data. Nowlan [Now91] develops a probabilistic framework for VQ by demonstrating the correspondence between a VQ and a mixture of spherically symmetric Gaussians. In the limit as the mixture component variance goes to zero, the Expectation-Maximization (EM) procedure for fitting the mixture model to data becomes identical to the Linde-Buzo-Gray (LBG) algorithm [LBG80] for vector quantizer design.

This paper develops a statistical model for *fixed-rate* adaptive transform coding. We define a constrained mixture of Gaussians model that provides a framework for transform coder design. Our new design algorithm is a *constrained* version of the LBG algorithm. It iteratively optimizes the signal space partition, the local transforms, the allocation of coding bits, and the scalar quantizer reproduction values until it reaches a local distortion minimum. This approach leads to a new method of partitioning the signal space designed to minimize coding error.

To evaluate the compression performance of our algorithm, we give results from compressing two types of gray-scale digital images: Magnetic Resonance Images (MRI) and gray-scale natural images of traffic moving through street intersections. We compare compressed image quality using our local PCA transform coding to that of using global PCA transform coding and two other adaptive transform coding methods similar to previous work [AL99, DH95].

4.2 Probability Models for Transform Coding

In this section, we develop constrained mixture of Gaussians models that provide a statistical model for fixed-rate adaptive transform coding. A transform coder converts a signal to new coordinates and then codes the coordinate values *independently* of one another with scalar quantizers. An *adaptive* transform coder consists of a collection of transform coders, each specialized to optimally compress data from different regions of the data space.

To develop a model for adaptive transform coding, we envision the d dimensional observed data as drawn from a structured discrete latent data space S, also with dimension d. The latent data lies at the vertices, $q_\alpha^{(\mathsf{m})}$ of one of M rectangular grids centered at $\eta^{(\mathsf{m})}$. Grid m is defined by the s axes and a set of grid mark values $\{r_{Ji}^{(\mathsf{m})}\}$, where r_{Ji} is the i^{th} grid mark along the s_J axis. The coordinates of each $q_\alpha^{(\mathsf{m})}$ can be written as some $[r_{1i}^{(\mathsf{m})}, r_{2j}^{(\mathsf{m})}, \ldots, r_{dk}^{(\mathsf{m})}]^T$. There are $\mathcal{K}_J^{(\mathsf{m})}$ possible grid mark values along the s_J axis in the m^{th} grid and the total number of grid vertices is $\mathcal{K}_\mathsf{m} = \prod_J \mathcal{K}_J^{(\mathsf{m})}$. Each grid has the same number of components $\mathcal{K}_\mathsf{m} = \mathcal{K}$, $\forall \mathsf{m}$, however the number and spacing of the grid mark values on each axis can differ. Figure 4.1 illustrates the structure of a single grid.

The density due to a single grid consists of a *mixture* of delta functions

$$\mathsf{p}(s|\mathsf{m}) = \sum_{\alpha=1}^{\mathcal{K}} p(\alpha|\mathsf{m})\, \delta(s - \eta^{(\mathsf{m})} - q_\alpha^{(\mathsf{m})}) \qquad (4.1)$$

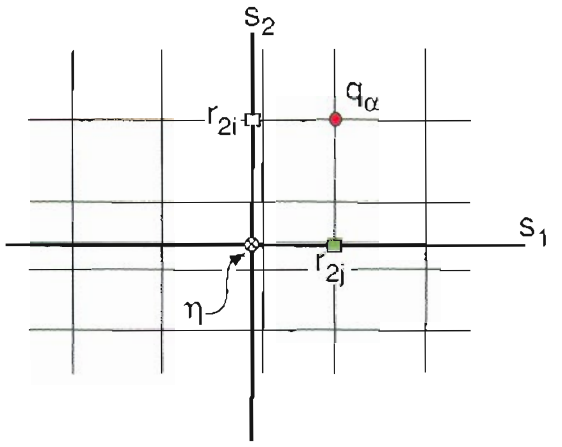


Figure 4.1: Structure of latent variable space, S, with single grid. The density on s consists of a mixture of delta functions where the mixture components, q_α, are constrained to lie at the vertices of a rectangular grid. The grid is centered at η and is defined by the s axes and a set of grid mark values $\{r_{Ji}\}$, where r_{Ji} is the i_{th} grid mark along the s_J axis.

The local mixing coefficients $\mathsf{p}(\alpha|\mathsf{m})$ are the product of prior probabilities $\mathsf{p}(r_{Ji}|\mathsf{m})$ so that

$$\mathsf{p}(\alpha|\mathsf{m}) = \prod_{J=1}^{d} \mathsf{p}(r_{Ji}^{(\mathsf{m})}|\mathsf{m}) \tag{4.2}$$

Consequently, we can write the local density for grid m as a product of marginal densities (for the derivation, see Appendix A)

$$\mathsf{p}(s|\mathsf{m}) = \prod_{J=1}^{d} \sum_{i=1}^{\mathcal{K}_J^{(\mathsf{m})}} \mathsf{p}(r_{Ji}^{(\mathsf{m})}|\mathsf{m})\delta(s_J - \eta_J^{(\mathsf{m})} - r_{Ji}^{(\mathsf{m})}) \tag{4.3}$$

where $\eta_J^{(\mathsf{m})}$ is the J^{th} coordinate value of $\eta^{(\mathsf{m})}$. We will use both formulations for the latent density (4.1) and (4.3) for our algorithm development.

The density on the whole latent space consists of a *mixture* of delta function mixtures

$$\mathsf{p}(s) = \sum_{\mathsf{m}=1}^{M} \pi_\mathsf{m} \sum_{\alpha=1}^{\mathcal{K}} p(\alpha|\mathsf{m})\, \delta(s - \eta^{(\mathsf{m})} - q_\alpha^{(\mathsf{m})}) \tag{4.4}$$

where π_m are mixing coefficients. The latent data from each grid m is mapped to the observation space by its own orthogonal transform $W^{(\mathsf{m})}$. The data is then corrupted with additive Gaussian noise, $\epsilon \sim \mathcal{N}(0, \sigma^2\mathrm{I})$. The observed data generated from some sample s drawn from latent component (α, m) is

$$x = W^{(\mathsf{m})}(s - \eta^{(\mathsf{m})} - q_\alpha^{(\mathsf{m})}) + \mu^{(\mathsf{m})} + \epsilon^{(\mathsf{m})} \tag{4.5}$$

with conditional densities

$$\mathrm{p}(x|s,\alpha,\mathsf{m}) = \mathcal{N}(\mu^{(\mathsf{m})} + W^{(\mathsf{m})}(s - \eta^{(\mathsf{m})} - q_\alpha^{(\mathsf{m})}), \sigma^2 \mathrm{I}) \tag{4.6}$$

Figure 4.2 illustrates this mapping from a two grid latent space.

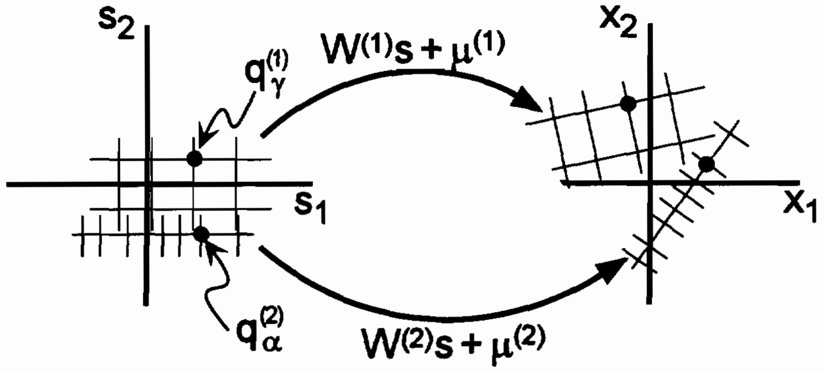


Figure 4.2: Fixed-rate adaptive model : Structure of latent variable space, S, and mapping (in hard clustering limit) to observed space, X. The mixture components, $q_\alpha^{(\mathsf{m})}$, are constrained to lie at the vertices of the M^{th} grid. Each grid has the same number of components, $\mathcal{K} = 8$. The Latent data is mapped to the observation space by orthogonal transforms, $W^{(m)}$ and corrupted with additive Gaussian noise.

The latent density and mapping induce a mixture of constrained Gaussian mixtures density on x of the form

$$\begin{aligned} \mathrm{p}(x) &= \int \sum_{\mathsf{m}=1}^{M} \pi_\mathsf{m} \sum_{\alpha=1}^{\mathcal{K}} \mathrm{p}(\alpha|\mathsf{m})\mathrm{p}(x|s,\alpha,\mathsf{m})\delta(s - \eta^{(\mathsf{m})} - q_\alpha^{(\mathsf{m})})ds \\ &= \sum_{\mathsf{m}} \pi_\mathsf{m} \sum_{\alpha=1}^{\mathcal{K}} \mathrm{p}(\alpha|\mathsf{m})\mathrm{p}(x|\alpha,\mathsf{m}) \end{aligned} \tag{4.7}$$

with the marginal densities

$$p(x|\alpha,\mathsf{m}) = \mathcal{N}(\mu^{(\mathsf{m})} + W^{(\mathsf{m})} q_\alpha^{(\mathsf{m})}, \sigma^2 \mathrm{I}) \tag{4.8}$$

The Expectation-Maximization algorithm (EM) [DLR77] fits parametric probability models to data by maximizing the log likelihood of the model for some training data set $\{x_n,\ n = 1 \ldots N\}$. The log likelihood is given by

$$\mathcal{L} = \sum_{n=1}^{N} \log\left(\sum_{\mathsf{m}=1}^{M} \pi_\mathsf{m} \sum_{\alpha=1}^{\mathcal{K}} \mathrm{p}(\alpha|\mathsf{m})\mathrm{p}(x_n|\alpha,\mathsf{m}) \right) \tag{4.9}$$

The generating component α of each data item x_n is unknown, so we introduce the density $z(\alpha, \mathsf{m}, x_n)$ over these unknown component assignments. The log likelihood $\mathcal{L}$ is bounded below by the *expected* log likelihood

$$\begin{aligned}\mathcal{L} \geq \langle\mathcal{L}\rangle \;=\; & \sum_{n=1}^{N}\sum_{\mathsf{m}=1}^{M}\sum_{\alpha=1}^{\mathcal{K}} z(\alpha,\mathsf{m},x_n)\,\log\left(\pi_{\mathsf{m}}\mathrm{p}(\alpha|\mathsf{m})\right) + \\ & \sum_{n=1}^{N}\sum_{\mathsf{m}=1}^{M}\sum_{\alpha=1}^{\mathcal{K}} z(\alpha,\mathsf{m},x_n)\big(-\frac{d}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\|x_n - \mu^{(\mathsf{m})} - W^{(\mathsf{m})}q_\alpha^{(\mathsf{m})}\|^2\big) - \\ & \sum_{n=1}^{N}\sum_{\mathsf{m}=1}^{M}\sum_{\alpha=1}^{\mathcal{K}} z(\alpha,\mathsf{m},x_n)\ln z(\alpha,\mathsf{m},x_n) \end{aligned} \tag{4.10}$$

with equality when $z(\alpha, \mathsf{m}, x) = p(\alpha, \mathsf{m}|x)$ is the posterior probability of component α in grid m conditioned on the data vector x [NH98]. The posterior probabilities are given by

$$\mathrm{p}(\alpha,\mathsf{m}|x) = \frac{1}{1+\sum_{\hat{\mathsf{m}}\neq\mathsf{m}}\sum_{\gamma\neq\alpha}\exp\left(\frac{-1}{2\sigma^2}[D_\gamma^{(\hat{\mathsf{m}})}(x)+2\sigma^2 l_\gamma^{(\hat{\mathsf{m}})}] - [D_\alpha^{(\mathsf{m})}(x)+2\sigma^2 l_\alpha^{(\mathsf{m})}]\right)} \tag{4.11}$$

where $D_\alpha^{(\mathsf{m})}(x) = \|x_n - \mu^{(\mathsf{m})} - W^{(\mathsf{m})}q_\alpha^{(\mathsf{m})}\|^2$ and $l_\alpha^{(\mathsf{m})} = -\log(\pi_{\mathsf{m}}\mathrm{p}(\alpha|\mathsf{m}))$.

The EM algorithm provides a template for deriving a transform coding algorithm from this probability model. To achieve hard-clustering needed for transform coding, we take the noise variance σ^2 to zero. In the limit that $\sigma^2 \to 0$, the $l_\alpha^{(\mathsf{m})}$ terms become insignificant and the posteriors (4.11) collapse to one or zero

$$z(\alpha,\mathsf{m},x) \to \begin{cases} 1 & D_\alpha^{(\mathsf{m})}(x) > D_\gamma^{(\hat{\mathsf{m}})}(x)\ \forall\gamma\neq\alpha \text{ and } \hat{\mathsf{m}}\neq\mathsf{m} \\ 0 & \text{otherwise}\end{cases} \tag{4.12}$$

In this hard-clustering limit, the final term in the expected log likelihood (4.10) becomes zero. Consequently, $\langle\mathcal{L}\rangle$ reduces to the cost function

$$\mathcal{C} = \frac{1}{N}\sum_{\mathsf{m}=1}^{M}\sum_{\alpha=1}^{\mathcal{K}}\sum_{n=1}^{N} z(\alpha,\mathsf{m},x_n)\|x_n - \mu^{(\mathsf{m})} - W^{(\mathsf{m})}q_\alpha^{(\mathsf{m})}\|^2 \tag{4.13}$$

with M and $\mathcal{K}$ selected so that $\log_2(M\mathcal{K})$ equals the desired coding bit-rate B. When the number of grids $M = 1$ we recover the cost function for fixed-rate *global* transform coding.

4.3 Adaptive Transform Coding Algorithm

In this section, we present a new algorithm for fixed-rate adaptive transform coder design that integrates optimization of the transform coder parameters: the data space partition, transforms, and quantizers. This generalized-Lloyd algorithm fits the parameters to data so that coding distortion (4.13) is minimized. Like all such algorithms, the optimization process is iterative. It alternately partitions the data space into local regions and then optimizes the transform and quantizers for each region.

4.3.1 Partition Optimization

To optimize the partition or encoder, each data vector is assigned to the reproduction vector $q_\alpha^{(\mathsf{m})}$ of transform coder m that represents it with the lowest distortion. To partition the data, we compress each d dimensional data vector x with each local transform coder $\mathsf{m} = 1 \ldots M$. To compress x, we first find the transform coefficients, $s_J^{(\mathsf{m})} = (W_J^{(\mathsf{m})})^T(x - \mu^{(\mathsf{m})})$, $J = 1 \ldots d$, where W_J is the J^{th} basis (column) vector of the W transform matrix. Each $s_J^{(\mathsf{m})}$ is then assigned to the scalar quantizer reproduction value, $r_{Ji}^{(\mathsf{m})}$ that represents it with the least entropy penalized distortion. Figure 4.3 demonstrates this transform and coding process.

We assign x to transform coder $\hat{\mathsf{m}}$ such that

$$\hat{\mathsf{m}} = \operatorname*{argmin}_m \sum_{J=1}^{d} \|(W_J^{(\mathsf{m})})^T(x - \mu^{(\mathsf{m})}) - r_{Ji}^{(\mathsf{m})}\|^2 \tag{4.14}$$

Hence, the data space partition defines regions $R^{(\mathsf{m})}$ such that each x belongs to the transform coder that compresses it lowest distortion,

$$R^{(\mathsf{m})} = \left\{x \mid \|(W_J^{(\mathsf{m})})^T(x - \mu^{(\mathsf{m})}) - r_{Ji}^{(\mathsf{m})}\|^2 < \|(W_J^{(\hat{\mathsf{m}})})^T(x - \mu^{(\hat{\mathsf{m}})}) - r_{Jk}^{(\hat{\mathsf{m}})}\|^2 \; \forall \hat{\mathsf{m}} \neq \mathsf{m}\right\} \tag{4.15}$$

In addition, the partition defines subregions $R_{Ji}^{(\mathsf{m})}$ such that each local transform coefficient $s_J^{(\mathsf{m})} = (W_J^{(\mathsf{m})})^T x$, $x \in R^{(\mathsf{m})}$, belongs to the scalar reproduction value $r_{Ji}^{(\mathsf{m})}$ that represents it with the lowest distortion,

$$R_{Ji}^{(\mathsf{m})} = \left\{s_J^{(\mathsf{m})} \mid \|s_J^{(\mathsf{m})} - r_{Ji}^{(\mathsf{m})}\|^2 < \|s_J^{(\mathsf{m})} - r_{Jk}^{(\mathsf{m})}\|^2 \; \forall k \neq i\right\} \tag{4.16}$$

Figure 4.4 illustrates the relationship between the the transform coder regions, $R^{(\mathsf{m})}$ and subregions $R_{Ji}^{(\mathsf{m})}$.

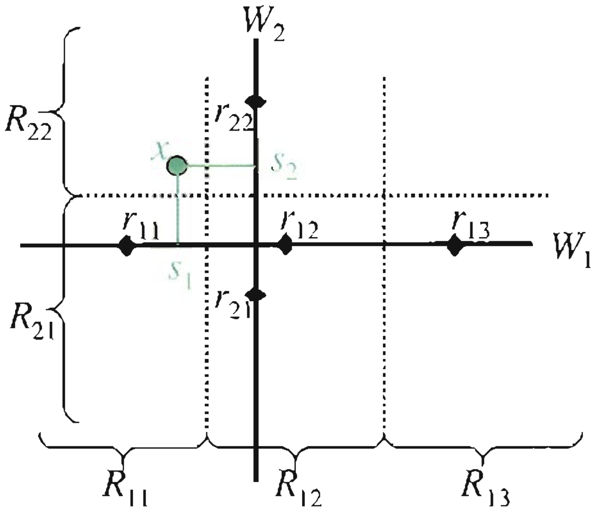


Figure 4.3: Transform Coding a Data Vector. Projecting data vector x with transform W yields coefficient values $W_1^T x = s_1$ and $W_2^T x = s_2$. The data space is partitioned into subregions with boundaries indicated by dotted lines. Coefficient s_1 is in subregion R_{11} and s_2 is in subregion R_{22}, hence x is represented by reproduction vector $q = [r_{11}, r_{22}]^T$.

The prior probabilities $\mathsf{p}(r_{Ji}^{(\mathsf{m})}|\mathsf{m})$ and π_{m} are estimated from the number of data values in each region. The transform coder prior $\pi_{\mathsf{m}} = N_{\mathsf{m}}/N$, where N are the total number of data vectors and N_{m} are the number of vectors in $R^{(\mathsf{m})}$. The reproduction value priors $\mathsf{p}(r_{Ji}^{(\mathsf{m})}|\mathsf{m}) = N_{Ji}^{(\mathsf{m})}/N_{\mathsf{m}}$ where $N_{Ji}^{(\mathsf{m})}$ are the number of transform coefficients in $R_{Ji}^{(\mathsf{m})}$.

4.3.2 Transform Optimization

To optimize the transform, we find the shift μ and orientation W of each quantizer grid that minimizes the coding cost function (5.13). The minimum cost estimators for the grid shift place each grid at the mean of the data assigned to its region

$$\mu^{(\mathsf{m})} = \frac{1}{N_{\mathsf{m}}} \sum_{x \in R^{(\mathsf{m})}} x \tag{4.17}$$

The grid orientation or transform W is constrained to be orthogonal, that is $W^T W = \mathbf{I}$. The cost function for transform coder m is thus

$$C = \frac{1}{N_{\mathsf{m}}} \sum_{\alpha=1}^{K_{\mathsf{m}}} \sum_{x \in R_{\alpha}^{(\mathsf{m})}} \|x - \mu^{(\mathsf{m})} - \sum_{J=1}^{d} W_J^{(\mathsf{m})} q_{\alpha J}^{(\mathsf{m})}\|^2 + \sum_{K=1}^{d} \sum_{L=1}^{d} \gamma_{KL}((W_K^{(\mathsf{m})})^T W_L^{(\mathsf{m})} - \delta_{K,L}) \tag{4.18}$$

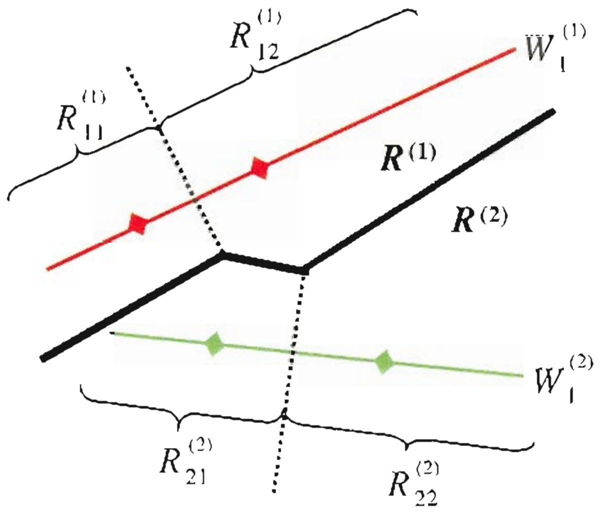


Figure 4.4: Data Space Partition. Partition of a two dimensional data space with two coders. Both coders consists of a 2×1 grids The boundary between the two coders, which partitions the data space into $R^{(1)}$ and $R^{(2)}$, is shown by the heavy black line. Subregion boundaries are indicated with dotted lines. The diamonds along the transform axes indicate placement of reproduction values.

where W_J is the J^{th} column vector of W, $q_{\alpha K}$ is the K^{th} coordinate of reproduction vector q_α, and γ_{KL} is a Lagrange multiplier. The partition assigns each data vector x to a quantizer reproduction *vector* $q_\alpha^{(m)}$ defining local regions $R_\alpha^{(m)}$. Figure 4.1 illustrates the relationship between the reproduction vectors q and scalar quantizer reproduction values or grid marks r.

Minimizing the local cost function (4.18) with respect to the transform matrix element $W_{KJ}^{(m)}$ yields

$$\sum_\alpha q_{\alpha J}^{(m)} \sum_{x \in R_\alpha^{(m)}} (x - \mu^{(m)})^T W_K^{(m)} = \sum_\alpha q_{\alpha K}^{(m)} \sum_{x \in R_\alpha^{(m)}} (x - \mu^{(m)})^T W_J^{(m)} \tag{4.19}$$

where $p(\alpha|m)$ is the prior probability of $q_\alpha^{(m)}$. If we define the outer-product matrix Q

$$Q = \sum_\alpha q_\alpha^{(m)} \sum_{x \in R_\alpha^{(m)}} (x - \mu^{(m)})^T \tag{4.20}$$

then (4.19) requires $QW = W^T Q^T$. This symmetry condition along with the orthogonality condition uniquely defines the Coding Optimal Transform (COT). Appendix A contains the detailed derivation of the COT.

Our global transform coding trials [AL01a] indicate that there is little difference between the PCA transform and the COT for *fixed-rate* quantizers. In addition, prior work in the area of transform coding uses either the PCA transform or DCT. Consequently, for the results presented in this chapter, we replace the COT with the PCA transform. However, the PCA transform is not optimal for coding, so careful implementation is required to achieve convergence to a local distortion minimum.

4.3.3 Quantizer Optimization

To optimize the quantizers, we adjust the number of reproduction values in each coordinate and coder $\mathcal{K}_J^{(\mathsf{m})}$ and the value of each reproduction value $r_{Ji}^{(\mathsf{m})}$ to minimize the cost function (4.13). This optimization is most conveniently performed in the local transform coordinates defined by $W^{(\mathsf{m})}$. Deriving the transform coding cost function using the product formulation of the latent density (4.3), instead of (4.1), yields the coding cost in terms of transform coefficients $s_J^{(\mathsf{m})} = (W_J^{(\mathsf{m})})^T(x-\mu^{(\mathsf{m})})$. In addition, we have the flexibility to adjust the number of components in each scalar quantizer as long as the total number of reproduction vectors in each grid equals $\mathcal{K}$, that is $\prod_J \mathcal{K}_J^{(\mathsf{m})} = \mathcal{K}$. We incorporate this constraint using Lagrange multiplier λ, so the cost function for transform coder m becomes

$$\mathcal{C} = \frac{1}{N_{\mathsf{m}}} \sum_{J=1}^{d} \sum_{i=1}^{\mathcal{K}_J^{(\mathsf{m})}} \sum_{s_J \in R_{Ji}^{(\mathsf{m})}} |s_J^{(\mathsf{m})} - r_{Ji}^{(\mathsf{m})}|^2 - \lambda^{(\mathsf{m})} \left(\sum_{J=1}^{d} \log \mathcal{K}_J^{(\mathsf{m})} - \log \mathcal{K} \right) \tag{4.21}$$

Minimizing the cost (4.21) with respect to the reproduction values places each reproduction value at the mean of the transform coefficients assigned to it.

$$r_{Ji}^{(\mathsf{m})} = \frac{1}{N_{Ji}^{(\mathsf{m})}} \sum_{s_J^{(\mathsf{m})} \in R_{Ji}^{(\mathsf{m})}} s_J^{(\mathsf{m})} \tag{4.22}$$

where $N_{Ji}^{(\mathsf{m})}$ are the number of transform coefficients in $R_{Ji}^{(\mathsf{m})}$.

For fixed-rate coding, the number of reproduction vectors per grid $\sum_J \mathcal{K}_J^{(\mathsf{m})}$ is kept below some target number $\mathcal{K}$, which corresponds to a target bit-rate $\hat{B} = B - \log_2 M$. Allocating the $\hat{B}$ coding bits where they minimize coding distortion the most determines the optimal values for $\mathcal{K}_J^{(\mathsf{m})}$, $J = 1 \ldots d$ [SG88, Ris91]. For a recent comprehensive review of quantization methods see [GN98].

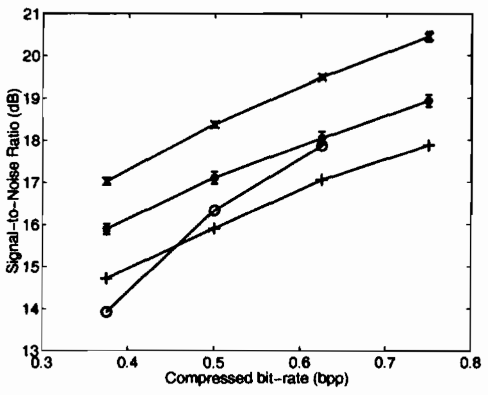
4.4 Adaptive Transform Coding Results

We find the adaptive transform coder for a set of images by applying our constrained LBG algorithm (CLBG) to a training image. The data vectors are 8×8 image pixel blocks. Then we compress a test image using the resulting transform coder. We measure compressed *test* image quality with signal-to-noise ratio, $\text{SNR} = 10 \log_{10}(\text{pixel variance}/\text{MSE})$, where MSE is the per pixel mean-squared coding error.

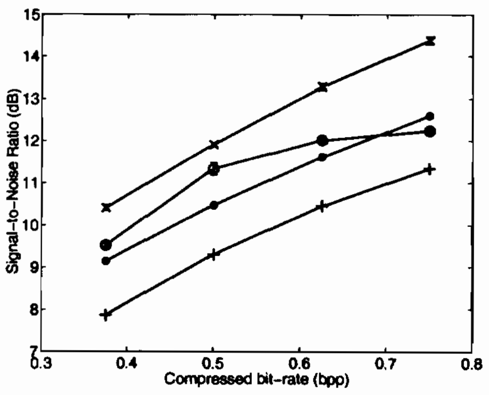
Our implementation modifies codebook optimization to reduce computational requirements and facilitate comparisons to other published methods. First, instead of using optimal bit allocation, we use a greedy algorithm [GG92], which allocates bits one at a time to the coordinate with the largest distortion. In global transform coding trials (0.375 to 0.75 bpp), this substitution reduced SNR by less than 0.1 dB. Second, instead of using the coding optimal transform, we use the PCA transform. In global transform coding trials (0.25 to 0.75 bpp), this substitution reduced SNR by 0.05 to 0.27 dB.

Classic global PCA transform coding is our baseline compression method. We also evaluate the compression performance of two other adaptive transform coders that use different methods to partition the signal space. The first method, Euclidean Distance Partition (EDP), clusters image blocks into regions so that the Euclidean distance between the blocks and the region means is minimized [AL99]. The second method, Reconstruction Distance Partition (RDP), clusters image blocks into regions so that the *reconstruction distance* is minimized [KL97]. The reconstruction distance is the mean squared error between an image block and its dimension-reduced reconstruction. The RDP method is similar to that used by Dony and Haykin [DH95]. For the RDP method, we selected a target dimension of eight, since at 0.5 bits per pixel (bpp) this dimension gave us the best test image SNR.

We report on compression experiments using two types of images, Magnetic Resonance Images (MRI) and gray-scale natural images of traffic moving through street intersections [oK98]. These MRI images were used by Dony and Haykin in [DH95] and we duplicate their image pre-processing. One MRI image is decomposed into overlapping 8×8 blocks to form 15,625 training vectors; a second image is used for testing. The traffic images are frames from two video sequences. We use frames from the first half of both sequences for training and frames from the last halves for testing.



(a) MRI test image SNR. All adaptive coders have 16 regions.



(b) Traffic test image SNR. All adaptive coders have 32 regions.

Figure 4.5: Compressed Image SNR. The × is our coding optimal partition (CLBG), ∘ local PCA partition with dimension eight (RDP), • k-means clustering (EDP), and + is global PCA. The dotted line values are local PCA results from [DH95]. Errorbars indicate standard deviation of 8 trials.

Figure 3 shows compressed test image SNR for four compressed bit-rates and four compression methods. The quoted bit-rates *include* the bits necessary to specify region assignments. The × results are for our CLBG transform coder. Our system increases SNR compared to global PCA (+) by 2.3 to 3.0 dB, EDP method (•) by 1.1 to 1.8 dB and RDP method (∘) by 0.5 to 2.0 dB. In addition, our system yields image SNRs 1.6 to 3.0 dB higher that Dony and Haykin's local PCA transform coder (dimension eight) [DH95]. Their local PCA coder does not use optimal bit allocation or Lloyd quantizers, which further reduces compressed image SNR.

The SNR improvement seen with the RDP method rolls-off at higher bit-rates. This method requires that we select the number of retained dimensions *before* training. We retain only eight dimensions, consequently at higher bit-rates directions that should be coded are discarded, which reduces compressed image quality.

We also evaluated the effect of changing the numbers of regions on compressed image SNR. Figure 4.6 shows test image SNR for compression to 0.5 bpp with 8, 16, and 32 region adaptive transform coders. For all three methods, SNR increases as the numbers of regions increase, assuming there is enough representative training data to prevent over-training. This is the expected result, since if all coding bits are

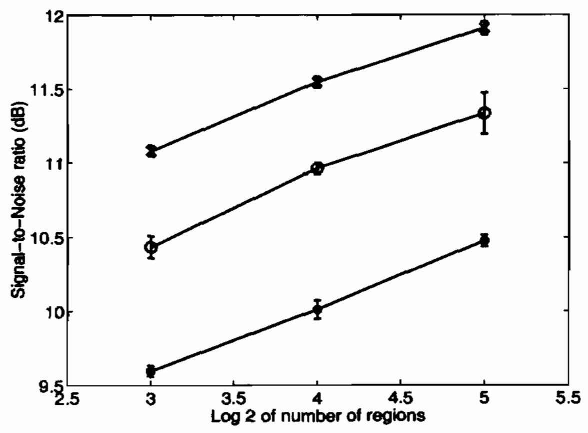


Figure 4.6: Test image compressed to 0.5 bpp. The • is EDP transform coding, ∘ is RDP transform coding, × is CLBG transform coding. Errorbars indicate standard deviation of eight trails.

used to represent the region designation, these algorithms produce an unconstrained LBG vector quantizer.

The enhanced image quality resulting from CLBG transform coding is also evident in the restored images. Figure 4.7 shows sections from a test image compressed to 0.5 bpp. The figure includes the original image and restored images from global PCA transform coding and our CLBG transform coding. The global PCA transform coded image is significantly degraded compared to the original. For example, the trolley tracks, the lines on the road, and edges of the cars are blurred and broken. In addition, pixel block edges are readily apparent throughout much of the image. When the image is compressed with CLBG transform coding, the blocking effect is less severe and image details are less blurred.

4.5 Summary

In this chapter, we cast the design of both conventional and adaptive transform coders as a constrained optimization procedure. We derive a new algorithm for transform coder design from the EM procedure for fitting a mixture of mixtures model to data. In contrast to standard transform coder design, all operations: partitioning the signal space (for the adaptive case), transform design, allocation of coding bits, and quantizer design, are coupled together to minimize compression distortion. This approach leads to a new transform basis that is optimized for coding. The coding optimal transform is in general different from PCA. Our approach also leads to a method of data space partitioning that is optimized for coding. This method assigns each signal vector to the coder the compresses it with the least distortion.

We evaluated our CLBG algorithm by using it to compress digital gray-scale images. To reduce computational requirements, our implementation approximates optimal local transform coder design by using the PCA transform and a greedy bit allocation procedure. At compression ratios in the range of 10:1 to 20:1, tests using our method demonstrate compressed image signal-to-noise ratios up to 3.0 dB higher than global PCA transform coding. When the same images were compressed with adaptive transform coders similar to previously implemented systems [DH95, AL99], the resulting image SNRs are 0.5 to 2.0 dB lower than those obtained with our system. Our integrated algorithm produces more efficient transform coders than previous local PCA methods that design the signal space partition and coefficient quantizers separately.

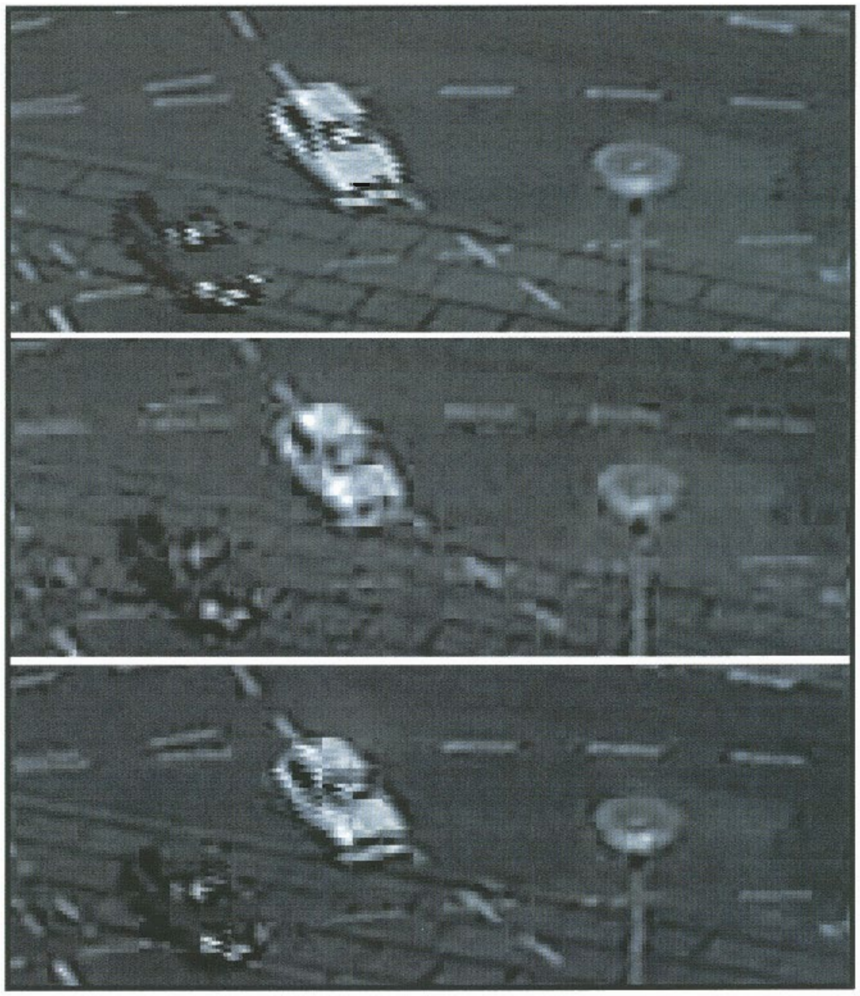


Figure 4.7: Sections from a test image compressed to 0.5 bpp. From top to bottom, original image, image compressed using global PCA transform coding, and CLBG transform coding with 32 regions.

Chapter 5

Entropy-Constrained Adaptive Transform Coding

In this chapter, we establish a probabilistic framework for adaptive transform coding that leads to a generalized Lloyd algorithm for entropy-constrained transform coder design. Transform coders are often constructed by concatenating an ad hoc choice of transform with suboptimal bit allocation and quantizer design. Instead, we start from a probabilistic latent variable model in the form of a mixture of constrained Gaussian mixtures. From this model we derive an optimal transform coding algorithm, which integrates optimization of all transform coder parameters. An essential part this algorithm is our introduction of a new transform basis, which unlike other transforms (PCA, DCT, etc.) is optimal for *coding.*

Compression experiments on benchmark images demonstrate that these optimal coders improve compressed image signal-to-noise ratio (SNR) by 0.25 to 1.25 dB over transform coders based on the DCT and PCA transforms. Optimal *adaptive* coders improve SNR by about 1 dB relative to global coders. In addition, our results from compressing a set of synthetic aperture radar images indicate that adaptive transform coders can be used effectively to compress databases of interest for real-world applications.

5.1 Introduction

Transform coding is a computationally attractive alternative to vector quantization that is widely used for image and video compression. A transform coder compresses multi-dimensional data by first transforming the data vectors to new coordinates and then coding the transform coefficient values independently with scalar quantizers. A key goal of the transform coder is to minimize compression distortion while keeping the compressed signal representation below some target size. While quantizers have typically been designed to minimize compression distortion [Llo82, FM84], this has not been the case for the transform portion of the coder. The transform has either been fixed a priori, as in the discrete cosine transform (DCT) used in the JPEG compression standard [Wal91], or adapted to the signal statistics using the Karhunen-Loeve transform (KLT) as in recently published transform coding work [DH95, ECG99]. These transforms are not designed to minimize compression distortion, nor are they designed (selected) in concert with quantizer development to deliver the best compression performance.

Classic transform design assumes that correlations between signal components are the same everywhere in the signal space. This assumption is valid only for wide-sense stationary data. Noting that signals such as images and speech are not wide sense stationary, several researchers have extended global transform coding to adapt to changing signal characteristics [ECG99, DH95, TB99, AL99]. In *adaptive* transform coding, the signal space is partitioned into disjoint regions and a set of basis functions (transforms) and scalar quantizers are designed for each region. In our own previous work [AL99], we use k-means clustering [Mac67] to define these regions. Dony and Haykin [DH95] partition the space to minimize dimension-reduction error. Tipping and Bishop [TB99] use partitioning according to a probabilistic rule that reduces, in the appropriate limit, to partitioning by dimension-reduction error as defined by Khambatla and Leen in [KL97]. These last two techniques are optimal for the task of dimension-reduction, but not for compression. Effros, et. al. [ECG99] correctly partition the signal space to minimize entropy-constrained coding error, but then use sub-optimal transform coders to compress the data in each region. None of these systems integrate optimization of all the transform coder parameters nor design these parameters to produce a coder that minimizes coding error.

In contrast to the ad hoc construction of transform coders, vector quantizers (VQ) are designed with algorithms [LBG80, CLG89] that minimize coding error. VQ algorithms all derive from a probabilistic model of the signal data to be compressed. Nowlan [Now91] presents this probabilistic framework by demonstrating the correspondence between a VQ and a mixture of spherically symmetric Gaussians. In the limit that the variance of the mixture components goes to zero, the Expectation-Maximization (EM) procedure [DLR77] for fitting the mixture model to data reduces to the K-means algorithm [Mac67] or, equivalently, the Linde-Buzo-Gray (LBG) algorithm [LBG80] for vector quantizer design. In addition, Chou et. al. [CLG89] note that the design algorithm for a entropy-constrained VQ (ECVQ) is a hard-clustering version of this same EM algorithm, but with non-zero component variance.

We make use of this probabilistic framework to construct transform coders that share the same optimal characteristics as VQs, yet maintain their advantage in computability. This paper develops an optimal design approach for both global and adaptive (local) transform coding. We first define a constrained mixture of Gaussians model that provides a framework for optimal transform coder design. Using this framework, we develop a new constrained generalized-Lloyd algorithm for transform coders that integrates optimization of the signal space partition, the local transforms, and the scalar quantizers. We conclude by demonstrating compression performance of our algorithms on both benchmark images and a database of synthetic aperture radar (SAR) images.

5.2 Probability Models for Transform Coding

In this section, we develop constrained mixture of Gaussians models that provide a statistical model for adaptive transform coding. A transform coder converts a signal to new coordinates and then codes the coordinate values *independently* of one another with scalar quantizers. An *adaptive* transform coder consists of a collection of transform coders, each specialized to optimally compress data from different regions of the data space.

To develop a model for adaptive transform coding, we envision the d dimensional observed data as drawn from a structured discrete latent data space S, also with dimension d. The latent data lies at the vertices, $q_\alpha^{(\mathsf{m})}$ of one of M rectangular grids centered at $\eta^{(\mathsf{m})}$. Grid m is defined by the s axes and a set of grid mark values $\{r_{Ji}^{(\mathsf{m})}\}$, where r_{Ji} is the i^{th} grid mark along the s_J axis. The coordinates of each $q_\alpha^{(\mathsf{m})}$ can be written as some $[r_{1i}^{(\mathsf{m})}, r_{2j}^{(\mathsf{m})}, \ldots, r_{dk}^{(\mathsf{m})}]^T$. There are $\mathcal{K}_J^{(\mathsf{m})}$ possible grid mark

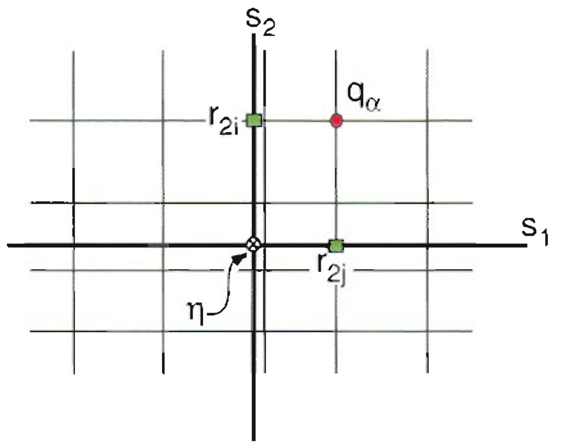


Figure 5.1: Structure of latent variable space, S, with single grid. The density on s consists of a mixture of delta functions where the mixture components, q_α, are constrained to lie at the vertices of a rectangular grid. The grid is centered at η and is defined by the s axes and a set of grid mark values $\{r_{Ji}\}$, where r_{Ji} is the i_{th} grid mark along the s_J axis.

values along the s_J axis in the mth grid and the total number of grid vertices is $\mathcal{K}_m = \prod_J \mathcal{K}_J^{(m)}$. Each grid can have a different number of components $\mathcal{K}_m$. Figure 5.1 illustrates the structure of a single grid.

The density due to a single grid consists of a *mixture* of delta functions

$$p(s|m) = \sum_{\alpha=1}^{\mathcal{K}^{(m)}} p(\alpha|m)\, \delta(s - \eta^{(m)} - q_\alpha^{(m)}) \tag{5.1}$$

The local mixing coefficients $p(\alpha|m)$ are the product of prior probabilities $p(r_{Ji}|m)$ so that

$$p(\alpha|m) = \prod_J p(r_{Ji}^{(m)}|m) \tag{5.2}$$

Consequently, we can write the local density for grid m as a product of marginal densities (for the derivation, see Appendix A)

$$p(s|m) = \prod_{J=1}^{d} \sum_{i=1}^{\mathcal{K}_J^{(m)}} p(r_{Ji}^{(m)}|m)\delta(s_J - \eta_J^{(m)} - r_{Ji}^{(m)}) \tag{5.3}$$

where $\eta_J^{(m)}$ is the J^{th} coordinate value of $\eta^{(m)}$. We will use this product of marginal densities formulation later in our algorithm development.

The density on the whole latent space consists of a *mixture* of delta function mixtures

$$p(s) = \sum_{m=1}^{M} \pi_m \sum_{\alpha=1}^{\mathcal{K}^{(m)}} p(\alpha|m)\, \delta(s - \eta^{(m)} - q_\alpha^{(m)}) \qquad (5.4)$$

where π_m are mixing coefficients. The latent data from each grid m is mapped to the observation space by its own orthogonal transform $W^{(m)}$. The data is then corrupted with additive Gaussian noise, $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. The observed data generated from some sample s drawn from latent component (α, m) is

$$x = W^{(m)}(s - \eta^{(m)} - q_\alpha^{(m)}) + \mu^{(m)} + \epsilon^{(m)} \qquad (5.5)$$

with conditional densities

$$p(x|s, \alpha, m) = \mathcal{N}(\mu^{(m)} + W^{(m)}(s - \eta^{(m)} - q_\alpha^{(m)}), \sigma^2 I) \qquad (5.6)$$

Figure 5.2 illustrates this mapping from a two grid latent space.

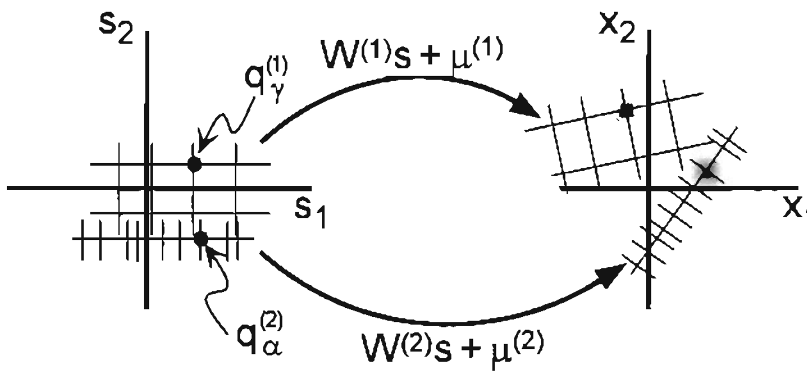


Figure 5.2: Nonstationary data model : Structure of latent variable space, S, and mapping (in hard clustering limit) to observed space, X. The mixture components, $q_\alpha^{(m)}$, are constrained to lie at the vertices of the M^{th} grid. Latent data is mapped to the observation space by orthogonal transforms, $W^{(m)}$ and corrupted with additive Gaussian noise.

The latent density and mapping induce a mixture of constrained Gaussian mixtures density on x of the form

$$\begin{aligned} p(x) &= \int \sum_{m=1}^{M} \pi_m \sum_{\alpha=1}^{\mathcal{K}_m} p(\alpha|m) p(x|s, \alpha, m) \delta(s - \eta^{(m)} - q_\alpha^{(m)}) ds \\ &= \sum_{m} \pi_m \sum_{\alpha=1}^{\mathcal{K}_m} p(\alpha|m) p(x|\alpha, m) \qquad (5.7) \end{aligned}$$

with the marginal densities

$$p(x|\alpha, \mathsf{m}) = \mathcal{N}(\mu^{(\mathsf{m})} + W^{(\mathsf{m})}q_\alpha^{(\mathsf{m})}, \sigma^2 \mathrm{I}) \tag{5.8}$$

The Expectation-Maximization algorithm (EM) [DLR77] fits parametric probability models to data by maximizing the log likelihood of the model for some training data set $\{x_n,\ n = 1 \dots N\}$. The log likelihood is given by

$$\mathcal{L} = \sum_{n=1}^{N} \log \left(\sum_{\mathsf{m}=1}^{M} \pi_\mathsf{m} \sum_{\alpha=1}^{\mathcal{K}_\mathsf{m}} \mathrm{p}(\alpha|\mathsf{m})\mathrm{p}(x_n|\alpha, \mathsf{m}) \right) \tag{5.9}$$

In order to simplify (5.9), we introduce the density $z(\alpha, \mathsf{m}, x_n)$ over the unknown component assignments.

$$\mathcal{L} = \sum_{n=1}^{N} \log \left(\sum_{\alpha=1}^{M} z(\alpha, x_n) \frac{\pi_\alpha \mathrm{p}(x_n|\alpha)}{z(\alpha, x_n)} \right) \tag{5.10}$$

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

$$\begin{aligned} \mathcal{L} \geq \langle \mathcal{L} \rangle \quad = \quad & \sum_{n=1}^{N} \sum_{\mathsf{m}=1}^{M} \sum_{\alpha=1}^{\mathcal{K}_\mathsf{m}} z(\alpha, \mathsf{m}, x_n) \log \left(\pi_\mathsf{m} \mathrm{p}(\alpha|\mathsf{m}) \right) - \\ & \sum_{n=1}^{N} \sum_{\mathsf{m}=1}^{M} \sum_{\alpha=1}^{\mathcal{K}_\mathsf{m}} z(\alpha, \mathsf{m}, x_n) (\frac{d}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} \|x_n - \mu^{(\mathsf{m})} - W^{(\mathsf{m})} q_\alpha^{(\mathsf{m})}\|^2) - \\ & \sum_{n=1}^{N} \sum_{\mathsf{m}=1}^{M} \sum_{\alpha=1}^{\mathcal{K}_\mathsf{m}} z(\alpha, \mathsf{m}, x_n) \ln z(\alpha, \mathsf{m}, x_n) \end{aligned} \tag{5.11}$$

with equality when $z(\alpha, \mathsf{m}, x) = p(\alpha, \mathsf{m}|x)$ is the posterior probability of component α in grid m conditioned on the data vector x [NH98].

The EM algorithm provides a template for deriving a transform coding algorithm from this probability model. To achieve the hard-clustering needed for transform coding, we choose $z(\alpha, \mathsf{m}, x_n)$ to be one or zero

$$z(\alpha, \mathsf{m}, x_n) = \begin{cases} 1 & \mathrm{p}(\alpha, \mathsf{m}|x_n) > \mathrm{p}(\gamma, \hat{\mathsf{m}}|x_n)\ \forall \gamma \neq \alpha \text{ and } \hat{\mathsf{m}} \neq \mathsf{m} \\ 0 & \text{otherwise} \end{cases} \tag{5.12}$$

With this hard-clustering model, the final term in the expected log likelihood (5.11) becomes zero since $z(\alpha, \mathsf{m}, x_n) \ln z(\alpha, \mathsf{m}, x_n) = 0\ \forall \alpha, \mathsf{m}, n$. Consequently, $\langle \mathcal{L} \rangle$ reduces to the cost function

$$\mathcal{C} = \frac{1}{N} \sum_{\mathsf{m}=1}^{M} \sum_{\alpha=1}^{\mathcal{K}_\mathsf{m}} \sum_{n=1}^{N} z(\alpha, \mathsf{m}, x_n) \left(\|x_n - \mu^{(\mathsf{m})} - W^{(\mathsf{m})} q_\alpha^{(\mathsf{m})}\|^2 - 2\sigma^2 \log(\pi_\mathsf{m} \mathrm{p}(\alpha|\mathsf{m})) \right) \tag{5.13}$$

This cost function consists of two terms combined with the Lagrange multiplier $2\sigma^2$: the average coding distortion

$$\mathcal{D} = \frac{1}{N} \sum_{\mathsf{m}} \sum_{\alpha} \sum_{n} z(\alpha, \mathsf{m}, x_n) ||x_n - \mu^{(\mathsf{m})} - W^{(\mathsf{m})} q_\alpha^{(\mathsf{m})}||^2 \tag{5.14}$$

and the discrete entropy

$$\mathcal{H} = -\sum_{\mathsf{m}} \sum_{\alpha} (\pi_{\mathsf{m}} \mathsf{p}(\alpha|\mathsf{m})) \log(\pi_{\mathsf{m}} \mathsf{p}(\alpha|\mathsf{m})) \tag{5.15}$$

This *entropy penalized* cost function (5.13) is the same as that found by minimizing coding distortion subject to an *average* bit-rate constraint (e.g. [CLG89]). In the limit that the noise variance σ^2 goes to zero, *and we limit the number of code vectors,* we recover the cost function for *fixed-rate* transform coding. When the number of grids $M = 1$ we recover the cost function for *global* or classic transform coding.

5.3 Adaptive Transform Coding Algorithm

In this section, we present a new algorithm for adaptive transform coder design that integrates optimization of the transform coder parameters: the data space partition, transforms, and quantizers. This generalized-Lloyd algorithm fits the parameters to data so that entropy penalized coding distortion (5.13) is minimized. Like all such algorithms, the optimization process is iterative. It alternately partitions the data space into local regions and then optimizes the transform and quantizers for each region. Each such iteration reduces (or at least does not increase) the value of the cost function. Generalized-Lloyd type algorithms converge to a local minimum of the cost function.

5.3.1 Partition Optimization

To optimize the partition or encoder, each data vector is assigned to the reproduction vector $q_\alpha^{(\mathsf{m})}$ of transform coder m that represents it with the least entropy-constrained distortion. To partition the data, we compress each d dimensional data vector x with each local transform coder $\mathsf{m} = 1 \ldots M$. To compress x, we first find the transform coefficients, $s_J^{(\mathsf{m})} = (W_J^{(\mathsf{m})})^T (x - \mu^{(\mathsf{m})})$, $J = 1 \ldots d$, where W_J is the J^{th} basis (column) vector of the W transform matrix. Each $s_J^{(\mathsf{m})}$ is then assigned to the scalar quantizer reproduction value, $r_{Ji}^{(\mathsf{m})}$ that represents it with the least entropy penalized distortion. Figure 5.3 demonstrates this transform and coding process.

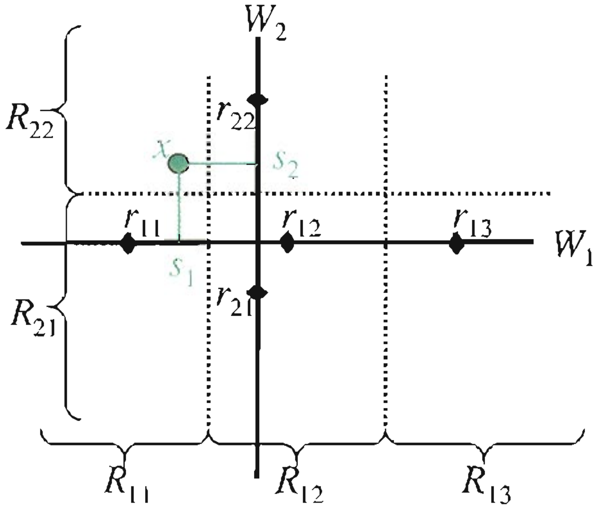


Figure 5.3: Transform Coding a Data Vector. Projecting data vector x with transform W yields coefficient values $W_1^T x = s_1$ and $W_2^T x = s_2$. The data space is partitioned into subregions with boundaries indicated by dotted lines. Coefficient s_1 is in subregion R_{11} and s_2 is in subregion R_{22}, hence x is represented by reproduction vector $q = [r_{11}, r_{22}]^T$.

The cost of assigning x to transform coder m is

$$C^{(\mathsf{m})}(x) = \sum_{J=1}^{d} \left(\|(W_J^{(\mathsf{m})})^T (x - \mu^{(\mathsf{m})}) - r_{Ji}^{(\mathsf{m})}\|^2 + 2\sigma^2 l_{Ji}^{(\mathsf{m})} \right) \tag{5.16}$$

where $l_{Ji}^{(\mathsf{m})} = -\log p(r_{Ji}^{(\mathsf{m})}|\mathsf{m})$ is the code word length. We then assign x to transform coder $\hat{\mathsf{m}}$ such that

$$\hat{\mathsf{m}} = \operatorname*{argmin}_{m} C^{(\mathsf{m})}(x) - 2\sigma^2 \log \pi_{\mathsf{m}} \tag{5.17}$$

Hence, the data space partition defines regions $R^{(\mathsf{m})}$ such that each x belongs to the transform coder that compresses it with the least entropy penalized distortion,

$$R^{(\mathsf{m})} = \left\{ x \mid (C^{(\mathsf{m})}(x) - 2\sigma^2 \log \pi_{\mathsf{m}}) < (C^{(\hat{\mathsf{m}})}(x) - 2\sigma^2 \log \pi_{\hat{\mathsf{m}}}) \; \forall \hat{\mathsf{m}} \neq \mathsf{m} \right\} \tag{5.18}$$

In addition, the partition defines subregions $R_{Ji}^{(\mathsf{m})}$ such that each local transform coefficient $s_J^{(\mathsf{m})} = (W_J^{(\mathsf{m})})^T x$, $x \in R^{(\mathsf{m})}$, belongs to the scalar reproduction value $r_{Ji}^{(\mathsf{m})}$ that represents it with the lowest coding cost,

$$R_{Ji}^{(\mathsf{m})} = \left\{ s_J^{(\mathsf{m})} \mid (\|s_J^{(\mathsf{m})} - r_{Ji}^{(\mathsf{m})}\|^2 + 2\sigma^2 l_{Ji}^{(\mathsf{m})}) < (\|s_J^{(\mathsf{m})} - r_{Jk}^{(\mathsf{m})}\|^2 + 2\sigma^2 l_{Jk}^{(\mathsf{m})}) \; \forall k \neq i \right\} \tag{5.19}$$

Figure 5.4 illustrates the relationship between the the transform coder regions, $R^{(m)}$ and subregions $R^{(m)}_{Ji}$. Consequently, the new data space partition minimizes the coding cost function (5.13) for the current transform and quantizer values.

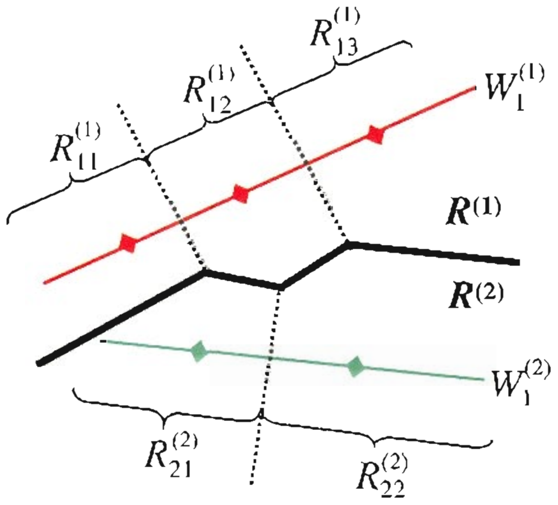


Figure 5.4: Data Space Partition. Partition of a two dimensional data space with two coders. Coder 1 consists of a 3×1 grid and coder 2 consists of a 2×1 grid. The boundary between the two coders, which partitions the data space into $R^{(1)}$ and $R^{(2)}$, is shown by the heavy black line. Subregion boundaries are indicated with dotted lines. The diamonds along the transform axes indicate placement of reproduction values.

The prior probabilities $p(r^{(m)}_{Ji}|m)$ and π_m are estimated from the number of data values in each region. The transform coder prior $\pi_m = N_m/N$, where N are the total number of data vectors and N_m are the number of vectors in $R^{(m)}$. The reproduction value priors $p(r^{(m)}_{Ji}|m) = N^{(m)}_{Ji}/N_m$ where $N^{(m)}_{Ji}$ are the number of transform coefficients in $R^{(m)}_{Ji}$.

5.3.2 Transform Optimization

To optimize the transform, we find the center μ and orientation W of each quantizer grid that minimizes the coding cost function (5.13). The minimum cost estimators for the grid center place each grid at the mean of the data assigned to its region

$$\mu^{(m)} = \frac{1}{N_m} \sum_{x \in R^{(m)}} x \tag{5.20}$$

The grid orientation or transform W is constrained to be orthogonal, that is $W^TW = \mathbf{I}$. The cost function for transform coder m is thus

$$\mathcal{C}_{\mathsf{m}} = \frac{1}{N_{\mathsf{m}}} \sum_{\alpha=1}^{\mathcal{K}_{\mathsf{m}}} \sum_{x \in R_\alpha^{(\mathsf{m})}} \|x - \mu^{(\mathsf{m})} - \sum_{J=1}^{d} W_J^{(\mathsf{m})} q_{\alpha J}^{(\mathsf{m})}\|^2 + \sum_{K=1}^{d} \sum_{L=1}^{d} \gamma_{KL}((W_K^{(\mathsf{m})})^T W_L^{(\mathsf{m})} - \delta_{K,L}) \tag{5.21}$$

where W_J is the J^{th} column vector of W, $q_{\alpha K}$ is the K^{th} coordinate of reproduction vector q_α, and γ_{KL} is a Lagrange multiplier. The partition assigns each data vector x to a quantizer reproduction *vector* $q_\alpha^{(\mathsf{m})}$ defining local regions $R_\alpha^{(\mathsf{m})}$. Figure 5.1 illustrates the relationship between the reproduction vectors q and scalar quantizer reproduction values or grid marks r.

Minimizing the local cost function (5.21) with respect to the transform matrix element $W_{KJ}^{(\mathsf{m})}$ yields

$$\sum_\alpha q_{\alpha J}^{(\mathsf{m})} \sum_{x \in R_\alpha^{(\mathsf{m})}} (x - \mu^{(\mathsf{m})})^T W_K^{(\mathsf{m})} = \sum_\alpha q_{\alpha K}^{(\mathsf{m})} \sum_{x \in R_\alpha^{(\mathsf{m})}} (x - \mu^{(\mathsf{m})})^T W_J^{(\mathsf{m})} \tag{5.22}$$

where $\mathrm{p}(\alpha|\mathsf{m})$ is the prior probability of $q_\alpha^{(\mathsf{m})}$. If we define the outer-product matrix Q

$$Q = \sum_\alpha q_\alpha^{(\mathsf{m})} \sum_{x \in R_\alpha^{(\mathsf{m})}} (x - \mu^{(\mathsf{m})})^T \tag{5.23}$$

then (5.22) requires $QW = W^TQ^T$. This symmetry condition along with the orthogonality condition uniquely defines the Coding Optimal Transform (COT). Appendix A contains the detailed derivation of the COT.

To minimize distortion, the COT orients the quantizer grid so that the QW matrix is symmetric (5.22). We can quantify how far the matrix is from symmetric with the sum squared differences between transposed matrix elements

$$A = \sum_{K=1}^{n-1} \sum_{J=K+1}^{n} (a_{KJ} - a_{JK})^2 . \tag{5.24}$$

where a_{KJ} is the K^{th} row and J^{th} column element of QW. We apply Givens rotations [GL89], $G(K, J, \theta)$, to minimize A. Multiplication by the $G(K, J, \theta)$ matrix applies a rotation of θ radians to the (K, J) coordinate plane. For a $n \times n$ matrix, there are $\frac{n^2-n}{2}$ such planes. Minimizing (5.24) with respect to rotation $G(K, J, \theta)$ yields a solution for θ that is quartic in $\tan\theta$. However, when the angle is small ($\tan^2\theta \ll 1$), the solution simplifies to

$$\tan\theta \approx \frac{(a_{KK}+a_{JJ})(a_{KJ}-a_{JK}) - \sum_{I\neq K,J}(a_{JI}a_{IK}-a_{JI}a_{IK})}{\sum_{I\neq K,J}(a_{JI}a_{IJ}+a_{KI}a_{IK}) + (a_{KK}+a_{JJ})^2 - (a_{KJ}-a_{JK})^2} \tag{5.25}$$

Since the COT reduces to the PCA transform when the data is Gaussian [AL01a, GZV00], we expect that starting the optimization from the PCA transform will keep the rotation angles small. This approach worked well in practice, allowing us to use this simpler form for the rotation angle. We find the rotation angle (5.25) for each coordinate plane and apply these rotations to the current transform matrix. This process is *repeated* until $A/\|QW\|_F$, where $\|QW\|_F$ is the Frobenius norm, is less than a threshold ($A \approx 0$). This new W will orient the quantizer grid so that compression distortion is minimized.

5.3.3 Quantizer Optimization

To optimize the quantizers, we adjust the number of coders M, number of reproduction values in each coordinate $\mathcal{K}_J^{(\mathsf{m})}$ and the value of each reproduction value $r_{Ji}^{(\mathsf{m})}$ to minimize the cost function (5.13). This optimization is most conveniently performed in the local transform coordinates defined by $W^{(\mathsf{m})}$. We rederive the transform coding cost function using the product formulation for the latent density (5.3), instead of (5.1). This derivation yields the coding cost in terms of transform coefficients $s_J^{(\mathsf{m})} = (W_J^{(\mathsf{m})})^T(x-\mu^{(\mathsf{m})})$. The cost function for transform coder m is

$$\mathcal{C} = \frac{1}{N_{\mathsf{m}}}\sum_{J=1}^{d}\sum_{i=1}^{\mathcal{K}_J^{(\mathsf{m})}}\sum_{s_J\in R_{Ji}^{(\mathsf{m})}}\left(|s_J^{(\mathsf{m})}-r_{Ji}^{(\mathsf{m})}|^2 - 2\sigma^2 l_{Ji}^{(\mathsf{m})}\right) - 2\sigma^2\pi_{\mathsf{m}}\log\pi_{\mathsf{m}} \tag{5.26}$$

where the $l_{Ji}^{(\mathsf{m})} = -\log \mathrm{p}(r_{Ji}^{(\mathsf{m})}|\mathsf{m})$ is commonly interpreted as the code word length.

Minimizing the cost (5.26) with respect to the reproduction values places each reproduction value at the mean of the transform coefficients assigned to it.

$$r_{Ji}^{(\mathsf{m})} = \frac{1}{N_{Ji}^{(\mathsf{m})}}\sum_{s_J^{(\mathsf{m})}\in R_{Ji}^{(\mathsf{m})}} s_J^{(\mathsf{m})} \tag{5.27}$$

where $N_{Ji}^{(\mathsf{m})}$ are the number of transform coefficients in $R_{Ji}^{(\mathsf{m})}$.

For entropy-constrained transform coding, selecting the noise variance is equivalent to selecting a target entropy. The target entropy determines the number of transform coders M and the number of reproduction values $\mathcal{K}_J^{(m)}$ in each scalar quantizer. The entropy terms in (5.26) move the partition away from the minimium distortion solution, so that reproduction values with low prior probabilities may have no data items assigned to them. Reproduction values with $p(r_{Ji}^{(m)}|m) = 0$ can be removed from the coder, reducing the value of $\mathcal{K}_J^{(m)}$. Likewise, coders with low priors may have no data items assigned to them, allowing the number of coders to be reduced. Consequently, selecting a large value for σ^2 produces low bit-rate coders. For a recent comprehensive review of quantization methods see [GN98].

5.4 Algorithm Evaluation

We evaluate our adaptive transform coding algorithm on benchmark images and a database of synthetic aperture radar (SAR) images. We compare compression performance of our method to that of classic transform coders based on the PCA transform (also known as the Karhunen-Loeve Transform or KLT) and the Discrete Cosine Transform (DCT). We also compare performance to that of PCA transform and DCT based adaptive transform coders. All coders use optimal entropy-constrained quantizers [FM84]. We report compression performance as signal-to-noise ratio (SNR), in dB, versus entropy, in bits per pixel (bpp). No entropy coding is performed.

In this evaluation, we also compare our emthod to compression with local PCA transform based coding, such as the iterative algorithm developed by Effros, et. al. [ECG99] or that presented in our previous work with *fixed-rate* adaptive transform coding [AL01b]. However, we found that the local PCA based algorithm has convergence issues. The PCA transform is only optimal for coding if the data is Gaussian. Consequently, the PCA transform update portion of this algorithm does not, in general, reduce the coding cost. That is, using PCA to define the transform does *not* yield a generalized Lloyd algorithm. In practice, we found that the coding cost almost always increased when the transform was updated and then decreased when the quantizers and partition were optimized. To handle these frequent cost increases, we monitored the *absolute* change in coding cost and stopped the optimization when this absolute change became small.

5.4.1 Evaluation on Benchmark Images

We illustrate the performance advantages of our optimal adaptive transform coding algorithm over classic transform coders and DCT based adaptive coding by compressing the benchmark images Barbara and Goldhill [oW98] shown in Figure 3.5. Barbara is a photograph of a seated women wearing striped clothing. Goldhill is a photograph of a row of houses in a hillside village. We compressed these images to entropies in the range of 0.25 to 1.25 bpp. The plot in Figure 5.5 displays SNR versus entropy results for global transform coders on the Barbara image. The differences in SNR were smaller for the goldhill image. Our global transform coding experiments on both images show that using our COT-based coder improves SNR by 0.3 to 1.2 dB relative to PCA transform based coders and 0.1 to 0.7 dB relative to DCT based coders. We give additional global transform coding results in [AL01a]. The plot in Figure 5.6 displays SNR versus entropy results for global versus adaptive transform coders on the Barbara image. In adaptive transform coding experiments with sixteen coders, our COT-based coder improves SNR by 0.4 to 1 dB relative to DCT-based adaptive coders. In addition, adaptive transform coding improves SNR by 0.4 to 1 dB relative to the corresponding global transform coder. Adaptive coders that use the PCA transform had SNRs comparable to DCT based adaptive coders at high bit rates and 0.2 to 0.5 dB higher at low bit rates.

To reconstruct a compressed image, the decompression engine must have the transform coder parameters. Therefore, these parameters must be transmitted with the compressed image, effectively increasing the compressed size. The storage space required for the transform coder parameters is referred to as *overhead*. For the tested transform coders, the overhead was 10 bits (3 decimal digits) for each transform element and 18 bits (5 decimal digits + sign) for each reproduction value and each associated prior probability. Since the DCT is a fixed for all images, we hard-coded it into the decompression software. When overhead is included as part of the compressed image, we find that the Barbara image compressed with the global DCT based coder has a SNR approximately 0.5 dB higher than when compressed with the global COT based coder. The SNR improvement provided by the COT is not enough to compensate for the increased overhead. The deleterious effect of overhead on compression performance is greater for adaptive transform coders, making it impractical to develop such coders for individual images.

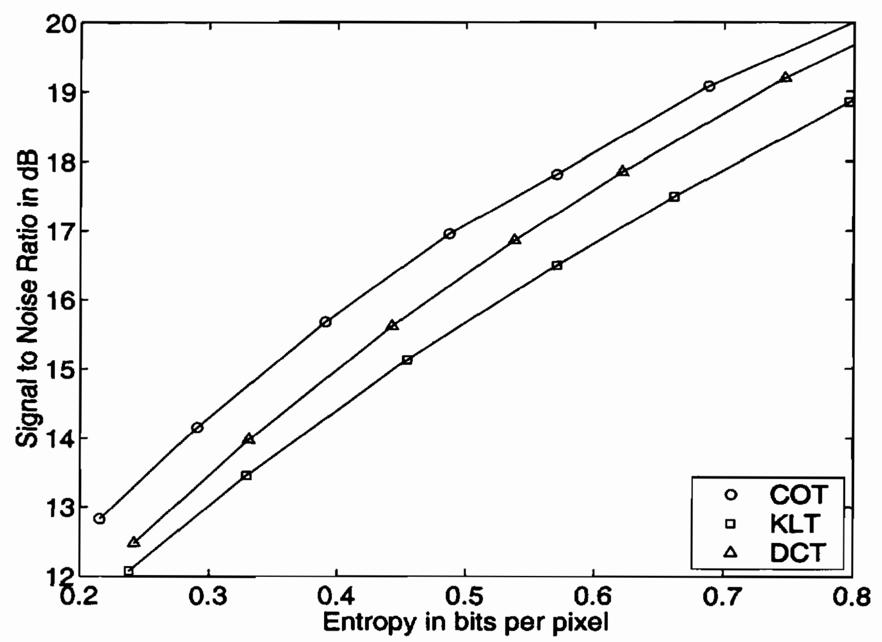


Figure 5.5: Comparison of performance of global transform coders. Plot shows signal-to-noise ratio versus entropy for Barbara image for global transform coders with entropy-constrained quantizers and the COT (circle), PCA transform (square), or DCT(triangle).

5.4.2 Evaluation on Image Database

Database compression provides an important and practical application for adaptive transform coding. While the data contained in an individual data file, such as an image, is non-stationary, the characteristics of the different files within the database are often similar. Consequently, one *adaptive* transform coder can be developed and subsequently used to compress all files within the database. This allows us to incorporate the transform coder parameters into the decompression engine, alleviating the overhead problem. For large databases, a single adaptive coder can require less overhead storage than separate coders for each image. As an added advantage, no new coders need to be developed when new items are added to the database. Although published work [DH95, ECG99, TB99, AL99] demonstrates the performance gain of adaptive transform coding over global coders *developed on the same training image*, it does not address how much compression performance is lost relative to a *set* of global coders, one for each image in the database. In this section, we compare performance of adaptive transform coders developed on a training image to global coders developed specifically for the test image.

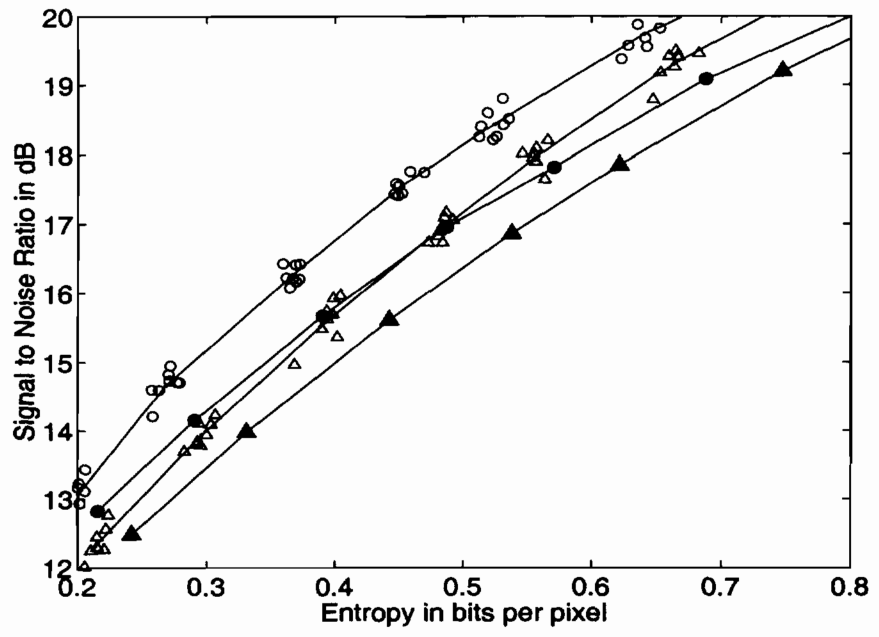


Figure 5.6: Comparison of compression performance of adaptive coding with 16 local transform coders to global transform coding. Signal-to-Noise Ratio versus entropy for Barbara image where adaptive COT based coder results are indicated with open circles, global COT with solid circles, adaptive DCT based coder with open triangles and global DCT with solid triangles.

We evaluated the adaptive transform coders on a small database (18 MByte) of synthetic aperture radar (SAR) images [Fre00]. Our database consists of eleven images acquired via space-borne radar by the space shuttle Endeavor [Lab02]. Each image contains three pseudo-color channels: red is L-band (24cm) horizontally transmitted and received, green is L-band horizontally transmitted and vertically received, and blue is C-band (6 cm) horizontally transmitted and received. Prior to compression, each image is decomposed into its three channels and the pixels in each channel are divided into 8×8 blocks to form 64 dimensional data vectors. SAR images of Belgrade, Taipei, and San Diego constitute the training set (5.9MBtyes) used to optimize the transform coder parameters. We evaluated compression performance on eight SAR images chosen for their diversity of land uses and terrain types. The test images were acquired over Athens, Boston, Hampton, Honolulu, Laughlin (Colorado River), Lisbon, Phnom Penh, and Ventura.

We developed both global and adaptive transform coders for the three image training set for four entropies in the range of 0.2 bpp to 0.7 bpp. Seven adaptive coders for each bit-rate were trained starting from 64 regions and different random initializations. Since the cost function contains an entropy constraint, local coders that do not represent any data well will see their prior probabilities go to zero during the training process and are consequently discarded. The number of final coders ranged from 36 to 63 with an average of 57. We report SNR results in terms of entropy. The coder parameters can be included in the decompression engine, so overhead is not included in the bit-rate. We developed eleven global coders for each bit rate, one for each of the images in our database. Overhead is included in the customized global coder bit-rate, since coder parameters are different for every image.

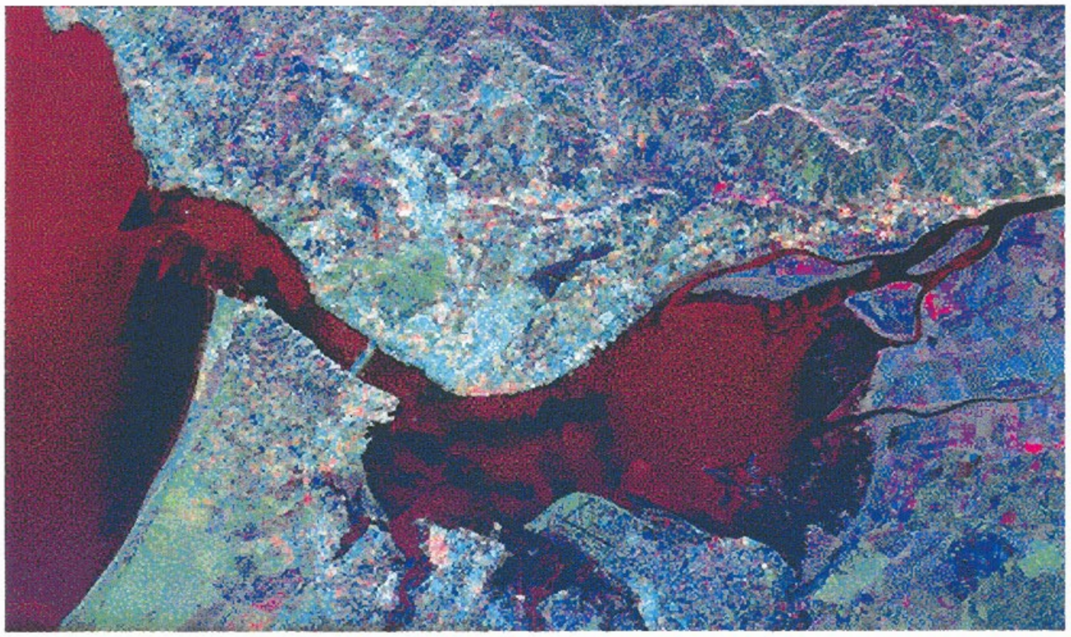


Figure 5.7: Pseudo-color SAR image of Lisbon, Portugal

Our results show that a single adaptive transform coder can perform as well on a database of related images as a set of global coders customized for each image in the database. Compression results for the Lisbon image, plotted in figure 5.8, demonstrate the relative performance of the COT and DCT based compression methods. The Lisbon image, Figure 5.7, includes a wide variety of land uses and terrain, making it representative of other images in the database. The global DCT and COT based coders have nearly identical overhead adjusted performance. The adaptive coders, developed for a training image, not only match global coder performance, but have slightly *better* SNRs than a global coder developed for the Lisbon image. The SNR improvement of the COT based adaptive coder is 0.14 dB and the improvement of the DCT based coder is 0.36 dB at an entropy of 0.466 bpp. The PCA transform based adaptive coder had test image SNRs comparable to the COT based coder. Note that the adaptive COT based coder generalizes less well than the adaptive DCT based coder.

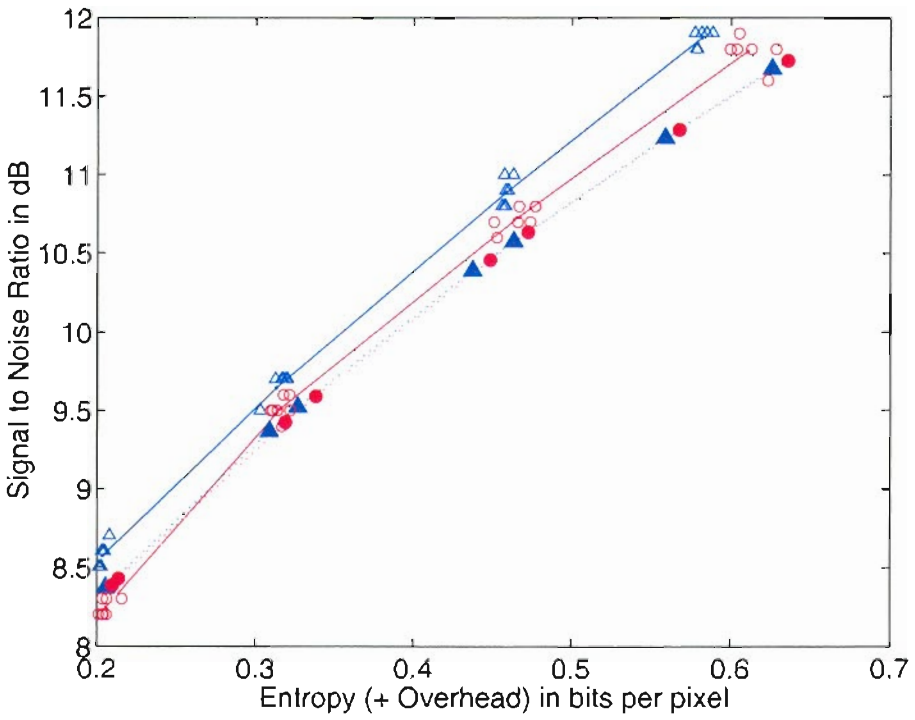


Figure 5.8: Signal-to-Noise Ratio versus entropy for Lisbon image. Compression performance of adaptive coding with approximately 57 local transform coders to global transform coders developed for the test image. Solid circles are for COT based global coder and solid triangles indicate the DCT based global coder. The bit-rate for the global coders is the entropy plus overhead. Open circles are the COT based adaptive coder and open triangles are the DCT based adaptive coder. The lines pass through the means of the seven trials at each bit-rate. The bit-rate for the adaptive coders is just the entropy.

We saw similar results for the eight test images and for the three images included in the training set. At entropies of 0.5 bpp, the eleven image SNRs ranged from 8 dB to 12 dB. The adaptive COT based coder had higher SNR than the customized global coders for all but two test images (Ventura and Laughlin) and had an average improvement of 0.10 dB. The adaptive DCT based coder had higher SNR than the customized global coders for all but one test image (Ventura) and had an average improvement of 0.25 dB. The adaptive DCT based coder generalized better than the COT based coder with consistently better SNRs (average 0.18 dB) on the test images.

Another important aspect of compression is time: training time, encoding or compression time, and restoration or decompression time. An adaptive transform coder requires more time for training and encoding than does a comparable global transform coder, although the restoration times are about the same. Training and encoding are done once for the items in a database, making the larger processing time less important than the reconstruction time. Our adaptive COT based coders required an average training time of 900 minutes and our adaptive DCT based coders an average time of 500 minutes on a 750 MHz Sun Ultra-SPARC III. The individual global transform coders required an average of 36 minutes and 4.3 minutes for the COT and DCT versions, respectively. Adaptive coders also require longer encode times. The adaptive COT based coder required 352 seconds to encode the Lisbon image compared to 4.1 seconds for the global coder. The adaptive DCT based coder required 76.5 seconds compared to 3.3 seconds for the global coder. However, the differences in reconstruction time are small for the different methods. Adaptive COT based coders require 3.9 seconds to decompress the Lisbon image compared to 3.8 seconds for global COT, 3.2 seconds for adaptive DCT, and 3.1 seconds for the global DCT based coders.

5.5 Summary

This chapter describes the culmination of our research into optimal global and adaptive transform coder design. Existing transform coding design algorithms are constructed by concatenating separately designed and often suboptimal transforms and quantizers. In contrast to this approach, we developed a probabilistic framework for both global and adaptive transform coding. Using this probability model, we derived a generalized-Lloyd algorithm for optimal transform coder design. A significant and necessary part of this work is a new transform, the COT, that minimizes mean-squared coding error. Definition of this transform made possible our development of an algorithm that integrates optimization all transform coder parameters: the signal space partition, the transform, and the quantizers.

We evaluated our adaptive transform coder on benchmark images and a small database of SAR images. Our results on the benchmark images demonstrate the performance advantage of our new algorithm over existing transform coding methods. Global COT based transform coders have SNRs 0.3 to 1.2 dB higher than coders based on the PCA transform and 0.1 to 0.7 dB higher than DCT based coders. Adaptive coders have SNRs about 1 dB higher than global coders. However, practical considerations of training time and overhead storage limit adaptive coders to applications with a large number of similar data files that will likely be compressed once and restored many times.

Adaptive coders have been referred to as "universal coders" [ECG99], since with enough local coders, they can theoretically adapt to a variety of input signals. Our results on the SAR image database indicate that a single adaptive transform coder can be used effectively to compress databases. Adaptive transform coders compressed test images with SNRs as good as or better than global transform coders developed individually for each test image. DCT based adaptive coders appear to generalize best as they had better test image SNRs than either COT or PCA transform based coders. Note that generalization capability is largely dependent on the training set. For instance, both the COT and DCT based adaptive coders had SNRs lower than the corresponding global coder for the Ventura image, indicating that this image contains regions with characteristics that were not in the training set. An important practical consideration for developing robust coders is the construction of a training set that contains adequate representation of all expected signal types.

Adaptive transform coding provides a mechanism for developing custom compression engines for large scientific databases. Possible applications besides image databases include geophysical data or data from simulations of environmental processes. Our optimal adaptive transform coding algorithm provides a benchmark for systematic tradeoff between complexity, overhead, and performance gain of custom coder components. For instance, our experiments show that while our optimal algorithm produces coders than outperform existing transform coding methods on training data, adaptive DCT based coders generalize better to test *image* data.

Similar evaluations can be performed for types of data for which the DCT may not be a good transform choice. Similarly, one could constrain the quantizers to have uniform spacing (uniform quantizers have low overhead) and evaluate compression performance against optimal, but higher overhead, entropy-constrained quantizers. In summary, our new algorithm gives transform coding the same grounding as vector quantization, allowing systematic development of custom adaptive transform coders and filling a void in the compression literature.

Chapter 6

Variable Dimension Local PCA

This chapter addresses the problem of resource allocation in local linear models for non-linear principal component analysis. Local PCA models partition the data into regions and perform PCA in each region. Prior formulations of local PCA underutilize the potential of these models by requiring a single global target dimension. We propose a resource allocation approach to local dimension selection. Evaluations using our variable dimension local PCA to reduce the dimension of blocks of image data substantially increases dimension reduced image quality compared to fixed dimension approaches. Some of the material in this chapter was published at the International Joint Conference for Neural Networks in 1999.

6.1 Introduction

Local Principal Component Analysis (PCA) models, such as those developed by Kambhatla and Leen [KL97] and Hinton et. al. [HRD95], are alternatives to non-linear PCA models such as the five-layer, non-linear autoassociators developed by Kramer [Kra91] and Demers and Cottrell [DC93]. The latter construct smooth curved manifolds that are close to the data. Local PCA partitions the data space into regions and performs PCA in each region. Geometrically, such models approximate the data manifold by a set of local PCA hyperplanes. When used for dimension reduction, local PCA models exhibit a clear performance advantage over simple PCA. They are faster to fit than five-layer, nonlinear autoassociators, and often outperform them.

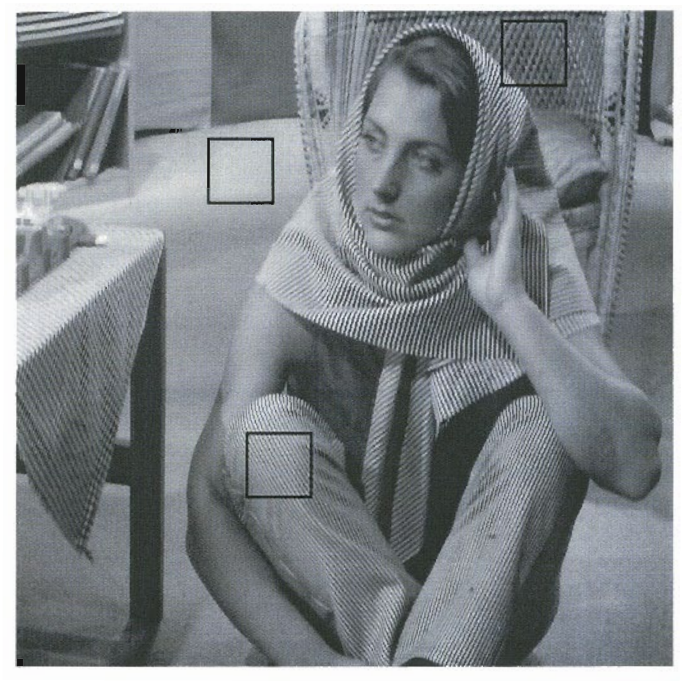
Despite their success, previous studies [KL97, HRD95, DH95, TB99] under-utilize the potential of these models. These authors choose a *global* target dimension, and hence neglect the variability in intrinsic data dimension from region to region in the data space. Here we construct a Langrangian-based algorithm that allows the model's dimension to be adjusted locally in order to decrease distortion, while the *average* dimension is constrained to a particular value.

6.2 Background

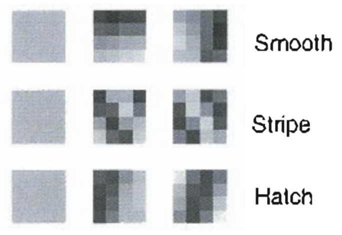
While transform coding depends on both the transform and the coding of the transform coefficients, dimension reduction dispenses with coding. It is therefore more purely a window into the performance of the transform. Dimension reduction can be used to preprocess data for other signal processing tasks, such as compression, classification and detection, or density estimation. It is vital for visualization of high-dimensional data.

Previous work on dimension reduction operates almost exclusively with reduction to a single, globally defined dimension. The dimension may be chosen by a fidelity requirement that places an upper bound on the allowed average distortion. Alternatively, several authors [WK85, HPF90] have applied minimum description length (MDL) criteria to PCA for estimating signal dimension. We argue that there is no compelling argument to require a single, global dimension, although a fixed-dimension approach may be more convenient for data visualization.

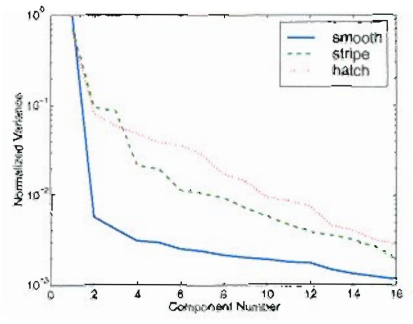
A simple exploration of the local correlation structure of images shows that different regions of the data space have different dimension. We extracted three small regions from the Barbara image, a common benchmark image for image compression evaluations. Each region was divided into 4×4 blocks to produce sixteen dimensional vectors. The principal eigendirections and variances for these three regions are shown beside the image in Figure 6.1. Note that the second and third principal eigenvectors show different patterns that are representative of the blocks in each region. Equally important, the numbers of high variance directions, that is the local dimensions, are different for each region.



(a) Barbara Image



(b) Leading Eigenvectors



(c) Variances

Figure 6.1: Non-stationarity of image data demonstrated with Barbara benchmark image. The characteristics for three regions, *smooth* taken from the floor, *stripe* taken from the women's slacks, and *hatch* taken from the wicker chair, are shown on the right. The principal eigendirection is the same for all three regions, but the second and third eigenvectors capture patterns that are representative of each region. The number of high-variance eigendirections, which determines the local dimension, varies between regions.

6.3 Variable Dimension Local PCA

Local PCA algorithms cluster the input data into regions and perform PCA on the data that falls within each region. Kambhatla and Leen define two clustering methods for local PCA in [KL97]. One method partitions the space using a coarse vector quantizer. Data vectors are assigned to the region whose center has the smallest Euclidean distance to the data vector. The second method partitions the data in order to minimize reconstruction distance or, equivalently, dimension reduction error. Both methods require the user to choose a single, global target dimension. The results presented in [KL97] show little performance difference in the two clustering methods.

For the local PCA work described here, clustering is done using the simpler vector quantizer method described in [KL97]. We perform PCA on the data assigned to each region to identify the local eigenvectors and eigenvalues. However, we do not choose a single global target dimension, but propose a *resource allocation* approach to local dimension assignment. Given some number of regions M and target *average* dimension d_o, our algorithm assigns dimension d_α to region R_α so as to minimize the expected dimension reduction distortion while keeping the average dimension below d_o. The algorithm can potentially assign a different dimension to each region of the signal space.

Our algorithm is motivated by a Lagrangian formulation in which we minimize dimension reduction distortion subject to a constraint on the average dimension. For a model with M regions, the prior probability of region R_α is π_α and the average dimension is

$$\bar{d} = \sum_{\alpha=1}^{M} \pi_\alpha d_\alpha \tag{6.1}$$

For a region with N_α data vectors, the average dimension reduction distortion is given by

$$D_\alpha(d_\alpha) = \frac{1}{N_\alpha} \sum_{x \in R_\alpha} (x - \mu_\alpha)(\mathrm{I} - U_\alpha U_\alpha^T)(x - \mu_\alpha) \tag{6.2}$$

where μ_α is the region mean and U_α is a matrix containing the leading d_α eigenvectors of the local data. The cost function to be minimized is thus

$$\mathcal{C} \;=\; \sum_{\alpha=1}^{M} \pi_\alpha \, D(d_\alpha) + \gamma\,(d_o - \sum_{\alpha=1}^{M} \pi_\alpha \, d_\alpha) \tag{6.3}$$

In general, realizing the theoretical minimization of this cost function (6.3) is not possible. However the following simple heuristic does allow one to reach an empirical minimum at approximately the desired average dimension:

1. For all regions R_α, initialize $d_\alpha = 0$.
2. Find region with the largest discarded eigenvalue $\lambda_{d_\alpha+1}$.
3. Allocate one additional dimension to that region, $d_\alpha^{new} = d_\alpha^{old}+1$.
4. Calculate the new average dimension $\bar{d}$. If $\bar{d} \geq d_o$, stop. Otherwise, loop to step (2).

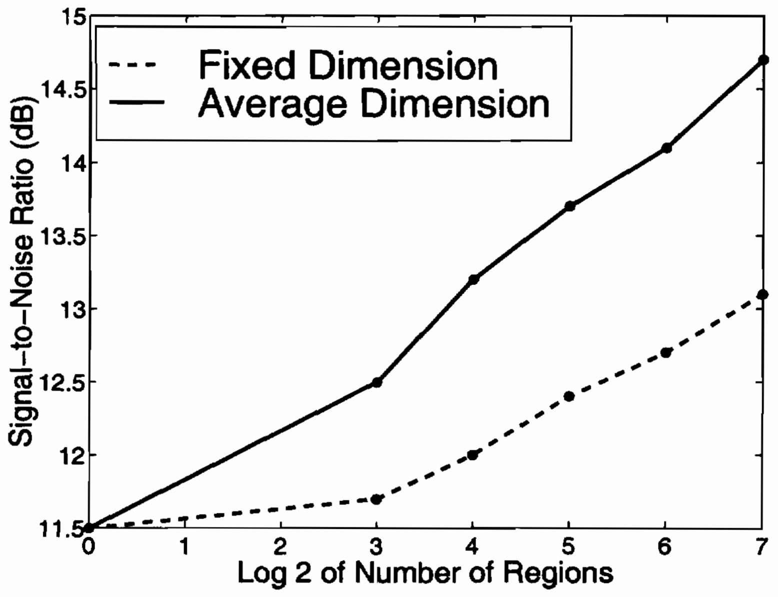


Figure 6.2: Dimension reduction of image blocks from 64 to 8 dimensions. Plot shows SNR for local PCA with different numbers of regions. Results for fixed local dimension are shown with dashed line and variable local dimension with solid line.

6.4 Experimental Results

We compared the dimension reduction performance of our variable dimension local PCA algorithm to that of fixed dimension local PCA [KL97] on a database of video frame images. The database consists of 50 frames each from two video sequences of city street intersections [oK98]. Each image was decomposed into 8 $\times$ 8 blocks to form 64 dimensional vectors. The training set consists of eight frames from the first half of each sequence. Frames from the second half of each sequence were used for testing.

Allowing the local dimension to adjust to the data results in substantial increases in the signal-to-noise ratio (SNR) of dimension reduced signals compared to fixed dimension methods. Figure 6.2 shows results for reduction of 64-dimensional blocks of image pixel values to both eight average and fixed dimensions. When using 32 PCA regions, the local dimension assigned by our algorithm varies between 3 and 23 among the different regions. The corresponding image SNR improvement is 1.3 dB relative to assigning all regions dimension eight and 2.2 dB relative to global PCA. For 128 region local PCA, the SNR increases about 1.5 dB in going from global

PCA (1 region) to the 128 region fixed dimension local PCA and increases another 1.5 dB by incorporating local dimension allocation.

6.5 Summary

Despite their success, previous formulations of local PCA under-utilize the potential of these models. These models incorporate a single global target dimension, neglecting the variability in intrinsic data dimension throughout the data space. We developed a variable dimension PCA algorithm that minimizes dimension reduction distortion subject to a constraint on the *average* dimension. The fidelity gains achieved by our variable dimension local PCA algorithm relative to fixed dimension methods are comparable to the advantage of used fixed dimension local PCA over global PCA.

This work with variable dimension local PCA indicates that to achieve good modeling quality, we should allow the dimension to adapt to the data structure. Our work developing adaptive transform coding models shows that constraining entropy is an effective way to limit model complexity, while giving the model the flexibility to adjust to the data structure. For transform coding, the entropy constraint arises naturally from the associated statistical model when we choose the noise variance to be the same everywhere in the data space. A similar construction for local or adaptive PCA whould provide an effective way to allow the dimension to conform to the data, while limiting model complexity. Our new entropy-constrained adaptive PCA algorithm is described in the next chapter.

Chapter 7

Entropy-Constrained Adaptive PCA

In this final chapter, 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.

7.1 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 [OT96], 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 [KL97, HRD95, DH95] have developed effective dimension reduction methods using adaptive PCA. In addition, Tipping and Bishop [TB99] and Ghararamani and Hinton [GH96] 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 [AL99], we found that dimension reduction performance of local PCA methods could be substantially improved by allowing the dimension to vary. Meinicke and Ritter [MR01] 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 [AL01b, AL01a], 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 [MR01]. 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 [TB99, MR01], 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.

7.2 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 [TB99] for probabilistic PCA and Basilevsky [Bas94] and Roweis and Ghahramani [RG99] 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.

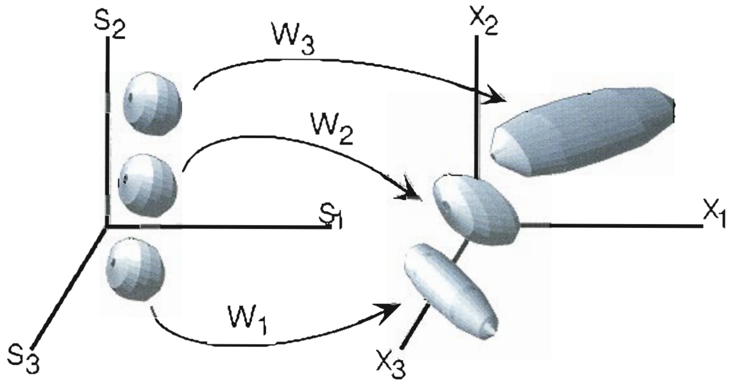


Figure 7.1: Adaptive PCA Model. Structure of latent variable space, S, and mapping to observed space, X. The data density in the latent space consists of a three Gaussians. This latent data is mapped to the observed data space by orthogonal transform, W, which stretch and rotate the data.

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 modeled with a simple *mixture* density of the form

$$p(s) = \sum_{\alpha=1}^{M} \pi_\alpha \, p(s|\alpha) \tag{7.1}$$

where π_α are the mixing coefficients and the components are spherical Gaussians $p(s|\alpha) = \mathcal{N}(\eta_\alpha, \rho^2 \mathrm{I})$ 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 7.1 illustrates this mapping from latent to observed space.

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

$$x = W_\alpha(s - \eta_\alpha) + \mu_\alpha + \epsilon_\alpha \tag{7.2}$$

with conditional densities

$$\mathrm{p}(x|s,\alpha) = \mathcal{N}(\mu_\alpha + W_\alpha(s - \eta_\alpha), \sigma_\alpha^2 \mathrm{I}) \tag{7.3}$$

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

$$\begin{aligned} \mathrm{p}(x) &= \int \sum_\alpha \mathrm{p}(x|s,\alpha)\mathrm{p}(s|\alpha)\pi_\alpha ds \\ &= \sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x|\alpha) \end{aligned} \tag{7.4}$$

where π_α are the same mixing coefficients given in (7.1) and $\mathrm{p}(x|\alpha) = \mathcal{N}(\mu_\alpha, \Sigma_\alpha)$. The covariance is constrained such that

$$\Sigma_\alpha = \sigma_\alpha^2 \mathrm{I} + U_\alpha \Gamma_\alpha U_\alpha^T \tag{7.5}$$

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) [DLR77] fits parametric probability models to data by maximizing the log likelihood of the model for some training data set $\{x_n,\ n = 1 \ldots N\}$. For additional information on mixture model fitting see chapter two of [Bis95]. The log likelihood for this model is given by

$$\mathcal{L} = \sum_{n=1}^{N} \log \left(\sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x_n|\alpha) \right) \tag{7.6}$$

To simplify the log likelihood equation (7.6), we introduce the density $z(\alpha, x_n)$ over the unknown component assignments. $\mathcal{L}$ is then bounded below by the *expected* log likelihood

$$\mathcal{L} \geq \sum_{\alpha=1}^{M} \sum_{n=1}^{N} \left(z(\alpha, x_n) \log \pi_\alpha \mathrm{p}(x_n|\alpha) - z(\alpha, x_n) \log z(\alpha, x_n) \right) \tag{7.7}$$

with equality when the $z(\alpha, x_n)$ are the posterior probabilities $\mathrm{p}(\alpha|x_n)$ [RG99, NH98]. This choice of z produces soft-clustering models.

Researchers have recently developed two different probability models for PCA [TB99, MR01], which can be derived from this framework. In Tipping and Bishop's model [TB99] 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 [KL97] for clustering by reconstruction distance. In this hard-clustering limit, the posterior probabilities become zero or one, that is

$$\mathrm{p}(\alpha|x) \rightarrow \begin{cases} 1 & \text{if } (x-\mu_\alpha)^T(\mathrm{I}-U_\alpha U_\alpha^T)(x-\mu_\alpha) < (x-\mu_\gamma)^T(\mathrm{I}-U_\gamma U_\gamma^T)(x-\mu_\gamma) \\ 0 & \text{otherwise} \end{cases} \tag{7.8}$$

for all $\gamma \neq \alpha$. In addition, the expected log likelihood (7.7) reduces to the cost function for local PCA

$$\mathcal{C} = \frac{1}{N}\sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\left((x_n-\mu_\alpha)^T(1-U_\alpha U_\alpha^T)(x_n-\mu_\alpha)\right) \tag{7.9}$$

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 [AL01b, AL01a]. 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 [MR01], 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 (7.5) become

$$\Sigma_\alpha = \sigma^2\mathrm{I} + U_\alpha\Gamma_\alpha U_\alpha^T \tag{7.10}$$

In the limit that σ^2 goes to zero, each local covariance matrix (7.10) 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 (7.7) for our adaptive PCA model, we first invert Σ_α (7.10) using the Sherman-Morrison-Woodbury formula [GL89]

$$\Sigma_\alpha^{-1} = \frac{1}{\sigma^2}(\mathrm{I} - U_\alpha U_\alpha^T) + U_\alpha\Lambda_\alpha^{-1}U_\alpha^T \tag{7.11}$$

with diagonal $d_\alpha \times d_\alpha$ matrix $\Lambda_\alpha = \Gamma_\alpha + \sigma^2 \mathrm{I}$. Using (7.11) to expand (7.7) gives the expected data log likelihood

$$\begin{aligned}
\mathcal{L} \;=\; & \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\ \ln\ \pi_\alpha - \\
& \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\frac{1}{2\sigma^2}\left((x_n - \mu_\alpha)^T(\mathrm{I} - U_\alpha U_\alpha^T)(x_n - \mu_\alpha)\right) - \\
& \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\frac{1}{2}(x_n - \mu_\alpha)^T U_\alpha \Lambda_\alpha^{-1} U_\alpha^T (x_n - \mu_\alpha) - \\
& \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\frac{1}{2}\left(\ln|\Lambda_\alpha| + (d - d_\alpha)\ln\sigma^2\right) - \\
& \sum_{n=1}^{N}\sum_{\alpha=1}^{M} z(\alpha, x_n)\ln z(\alpha, x_n) \qquad (7.12)
\end{aligned}$$

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.

7.3 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.

7.3.1 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.

$$z(\alpha, x_n) = \begin{cases} 1 & \mathrm{p}(\alpha|x_n) > \mathrm{p}(\gamma|x_n)\ \forall\gamma \neq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{7.13}$$

The hard assignments given in (7.13) partition the data space into regions R_α such that

$$\sum_{x \in R_\alpha} f(x) = \sum_{n=1}^{N} z(\alpha, x_n) f(x) \tag{7.14}$$

for any function $f(x)$. By choosing hard clustering with z given by (7.13), the expected log likelihood (7.12) reduces to the entropy-constrained cost function for adaptive PCA

$$\begin{aligned} \mathcal{C} &= \frac{1}{\sigma^2} \sum_{n=1}^{N} \sum_{\alpha=1}^{M} z(\alpha, x_n) \Big[(x_n - \mu_\alpha)^T (1 - U_\alpha U_\alpha^T)(x_n - \mu_\alpha) - 2\sigma^2 \ln \pi_\alpha + \\ & 2\sigma^2 \left(\frac{1}{2}(x_n - \mu_\alpha)^T U_\alpha \Lambda_\alpha^{-1} U_\alpha^T (x_n - \mu_\alpha) + \frac{1}{2} \ln |\Lambda_\alpha / \sigma^2| + \frac{d}{2} \ln \sigma^2 \right) \Big] \end{aligned} \tag{7.15}$$

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

$$D_\alpha(x) = (x - \mu_\alpha)^T (1 - U_\alpha U_\alpha^T)(x - \mu_\alpha). \tag{7.16}$$

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

$$H_\alpha(x) = -\ln \pi_\alpha + \frac{1}{2} \ln |\Lambda_\alpha / \sigma^2| + \frac{1}{2}(x - \mu_\alpha)^T U_\alpha \Lambda_\alpha^{-1} U_\alpha^T (x - \mu_\alpha) \tag{7.17}$$

and the log of a quantizer bin size $\frac{d}{2} \ln \sigma^2$ [CT91]. 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 (7.17), since

$$\text{Trace}\left[U_\alpha^T(\sum_n z(\alpha, x_n)(x_n - \mu_\alpha)(x_n - \mu_\alpha)^T)U_\alpha\Lambda_\alpha^{-1}\right] = d_\alpha \tag{7.18}$$

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.

7.3.2 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 (7.15). 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 (7.13). The partition consists of regions R_α such that

$$R_\alpha = \{x \mid D_\alpha(x) + 2\sigma^2 H_\alpha(x) < D_\gamma(x) + 2\sigma^2 H_\gamma(x) \ \forall\ \gamma \neq \alpha\} \tag{7.19}$$

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(7.15) for the current partition. The equations for the priors are

$$\pi_\alpha = \frac{1}{N}\sum_n z(\alpha|x_n) = \frac{N_\alpha}{N} \tag{7.20}$$

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

$$\mu_\alpha = \frac{1}{N_\alpha}\sum_{x \in R_\alpha} x \tag{7.21}$$

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

$$U_\alpha^T S_\alpha = \Lambda_\alpha U_\alpha^T \tag{7.22}$$

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

$$S_\alpha = \frac{1}{N_\alpha}\sum_{x \in R_\alpha}(x - \mu_\alpha)(x - \mu_\alpha)^T \tag{7.23}$$

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

$$\Delta C = \ln\frac{\lambda_q}{\sigma^2} - \left(\frac{\lambda_q}{\sigma^2} - 1\right) \tag{7.24}$$

where λ_q is the q^{th} entry in Λ_α. By Jensen's inequality $\ln\theta \leq \theta - 1$, therefore increasing the dimension will decrease the cost ($\Delta C < 0$) 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 (7.15) 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.

An outstanding issue concerns the selection of an appropriate entropy constraint via the noise variance σ^2. We attempted to select an optimal σ^2 by determining the value the minimized the cost for a hold-out or validation data set. However, this selection of σ^2 results in under-constrained models with nearly full dimension. However, our early evaluations show that models which contain several low-dimensional components conform better to the data structure than those with few high-dimensional components. Further work is needed to refine this observation into a principaled method of selecting an appropriate noise variance. To facilitate this work, 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.

7.4 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 [RGF93]. 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.

7.4.1 Deterministic Annealing

Deterministic annealing is motivated by viewing clustering as a minimization of free energy [Ros98]. 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 [RGF93]

$$z(x,y) = \frac{\exp(-\hat{d}(x,y)/T)}{\sum_y \exp(-\hat{d}(x,y)/T)} \tag{7.25}$$

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

$$d(\hat{x,} y) = D_\alpha(x) + 2\sigma^2(H_\alpha(x) + \frac{d}{2}\ln\sigma^2) \tag{7.26}$$

where D_α is given by (7.16) and H_α is given by (7.17). Substituting (7.26) into (7.25) and using (7.25) to expand the free energy F yields

$$F = -T\sum_x \ln \sum_\alpha \exp\left(- \ [D_\alpha(x) + 2\sigma^2(H_\alpha(x) + \frac{d}{2}\ln\sigma^2)]/T\right) \tag{7.27}$$

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 (7.6). As T approaches zero, the assignments becomes hard, as in (7.13), and each x is assigned to a single cluster. In this hard-clustering limit, F reduces to the adaptive PCA cost function (7.15).

The free energy formulation of adaptive PCA (7.27) allows implementation of deterministic annealing using the template described by Rose [RGF93]. It does not require the two-stage training process proposed by Meinicke and Ritter [MR01]. 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 [MR01, Ros98]. 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 $T_{\text{new}} = 0.9\ T_{\text{old}}$ 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.

7.4.2 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.

7.4.3 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.

7.4.4 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 7.2 contains a scatterplot of the 400 point test data projected to the two leading global 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 [CLG89] and adaptive PCA models using all three methods. Figure 7.3 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

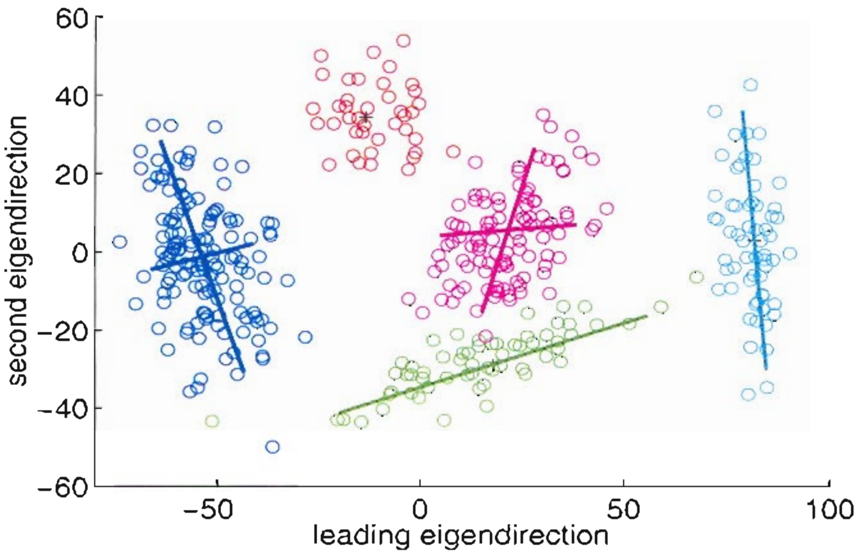
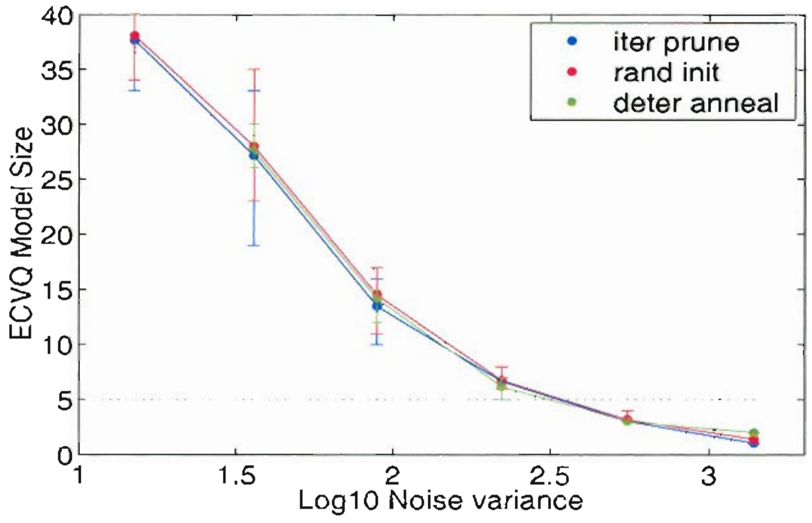


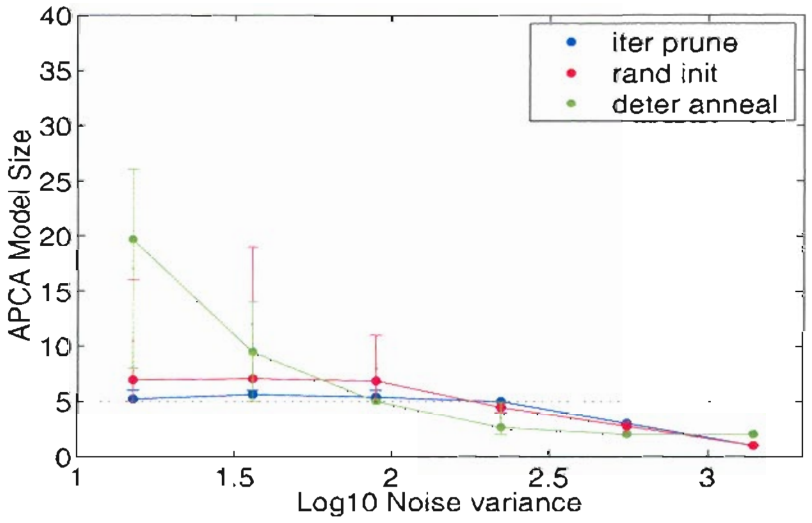
Figure 7.2: Mixture of Five Gaussians Test Data. Scatterplot of artificial data set used for testing. Data consists of 400 three-dimensional points drawn from a mixture of five Gaussians. Colored lines indicate the principal eigenvectors of each component and the number of lines corresponds to the dimension. Data is projected to the two leading eigendirections.

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 7.4 shows examples of five and six component models. Each model component matches one of the generating clusters in Figure 7.2 and no components bridge multiple clusters.

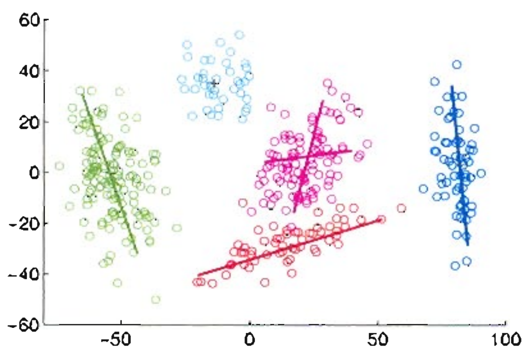


(a) ECVQ

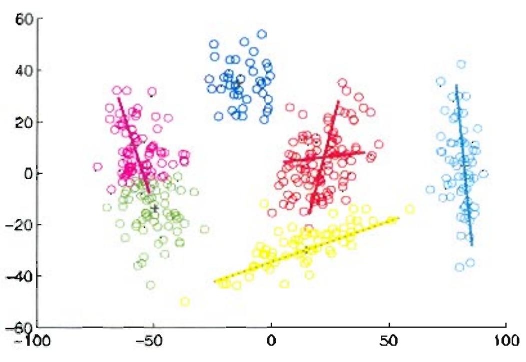


(b) APCA

Figure 7.3: 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).

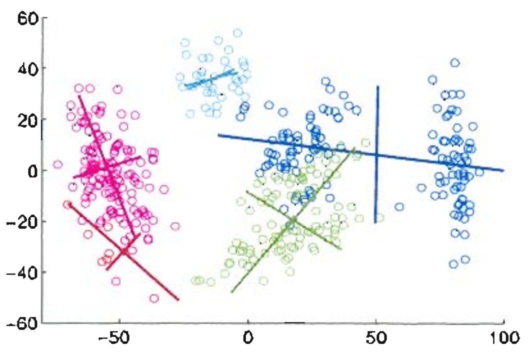


(a) Five Components

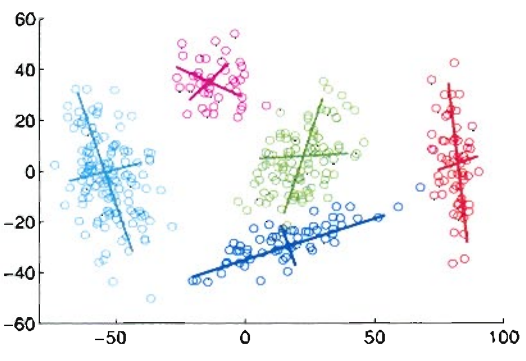


(b) Six Components

Figure 7.4: 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 7.5: 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.

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 7.5 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 chapter, we use our iterative pruning method for model training, since it produces better quality and more consistent models than those developed from random initializations.

7.5 Evaluation

We compare the modeling performance of our entropy-constrained adaptive PCA algorithm (APCA) to an entropy-constrained VQ (ECVQ) [CLG89] 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.

7.5.1 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 of the data. 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 the clusters is given by

$$I(clus, \alpha) = H(\alpha) + H(clus) - H(\alpha, clus) \tag{7.28}$$

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.

$$NMI(clus, \alpha) = 1 - \frac{H_p + H_a}{H(\alpha) + H(clus)} \tag{7.29}$$

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.

7.5.2 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 or class and where a component covers all or part of several different classes. Here we present clustering results on a real world data set, salinity and temperature measurements gathered in the Columbia River Estuary.

Columbia River Data

Sensors deployed in the Columbia River Estuary by environmental scientists at Oregon Health & Science University [BWP+99] 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 [ABL02]. 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 7.6 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.

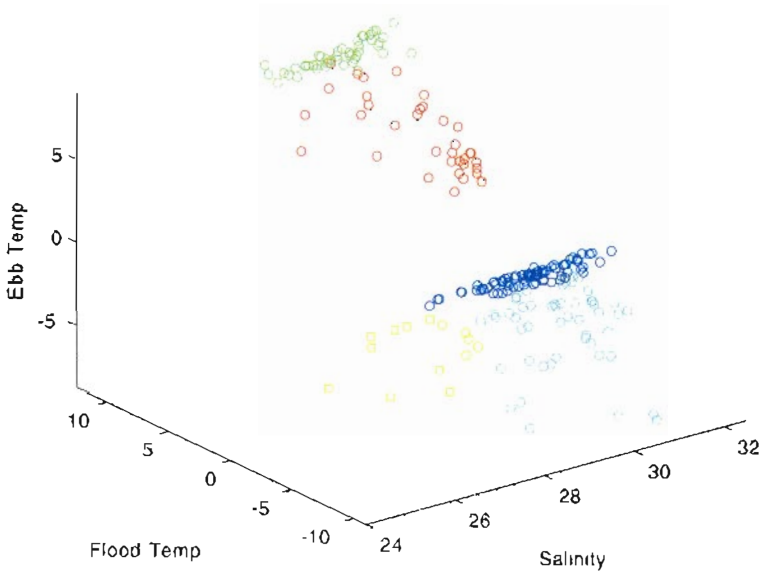


Figure 7.6: Columbia River Salinity and Temperature Data. Scatterplot of salinity and temperature at largest diurnal tidal flood and temperature at deepest diurnal ebb. Temperatures have been normalized by the estimated difference between the ocean and river temperatures. Colors indicate different classes identified from visual examination of the time series. The red and cyan regions may contain more than one class.

River Data Analysis

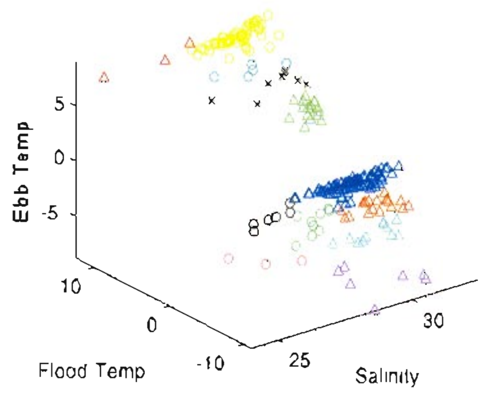
We use several modeling methods to cluster the salinity and temperature data into classes, including our entropy-constrained adaptive PCA (APCA), entropy-constrained vector quantization (ECVQ), a hard-clustering version of a full covariance GMM (HGMM), and local PCA (LPCA). Local PCA [KL97] partitions 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 each model separated the test data into classes. Scatterplots are from the model that had the lowest validation set cost.

The ECVQ models partition the space into many small spherical classes. Figure 7.7a shows clustering by the ECVQ model at noise variance 0.5. This model had thirteen components, over twice the number of our subjective estimate. All natural clusters are represented by several components. Since it does not identify the number of unique classes nor regimes, we found ECVQ to be a poor choice for this data analysis.

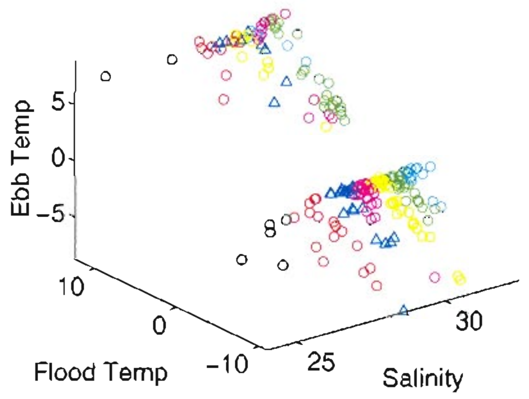
The LPCA models partition the space into many one-dimensional subspaces. Figure 7.7b shows clustering by a seven component LPCA model. Model size selection using the validation set indicated that the number of components should be at least forty (largest size tested). This large a model is not instructive, so we present results for a model size of seven. Model components consistently cut across the natural cluster structure of the data. We found the LPCA modeling method to be inappropriate for clustering or data analysis.

The HGMM models partition the space into five regions, but they do not match the classes show in Figure 7.6. Figure 7.7c 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.

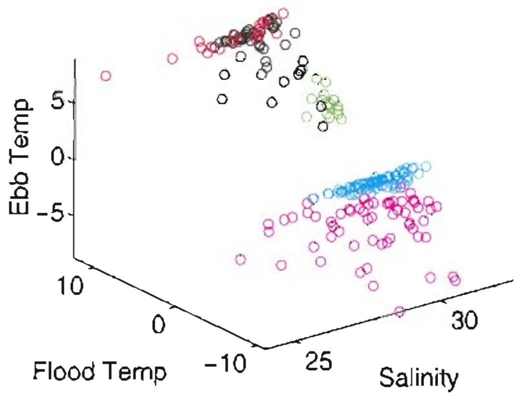
Unlike the previous three model types, APCA models are well-matched to the natural cluster structure of the data. Figure 7.7d shows clustering by the APCA model. This model correctly identifies the summer (red), low salinity (blue), and equal river and ocean temperature (cyan) classes. It identifies two classes within the temperature transition region (pink and green) and two classes within the winter region (yellow and black). Of the methods tested, the APCA model corresponded most closely to the natural cluster structure.



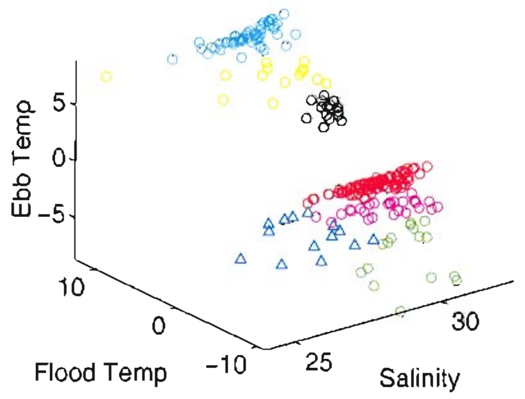
(a) ECVQ



(b) LPCA



(c) HGMM



(d) APCA

Figure 7.7: 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.

7.5.3 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.

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 7.8 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.

Model Size Analysis

As part of our analysis, we evaluated the model size selected for each method. Figure 7.9 includes model sizes for ECVQ, APCA, and HGMM. The full covariance HGMM method can model the data with one or two components, so the size is consistently too small. At high σ^2, the ECVQ and APCA models have heavy entropy penalties and select too few components. The ECVQ model size increases as σ^2 decreases, as more components are needed to cover the space as the component size shrinks. In contrast, the APCA model selects the close to the correct number of classes for a range of noise variances. For these moderate noise variances (1000 to 8000), the local dimension is small and several components are required to model the data

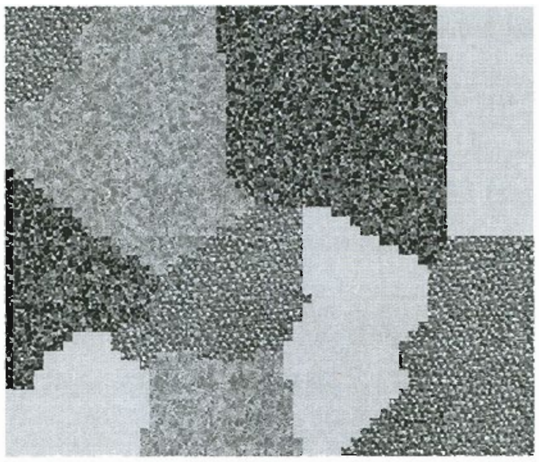


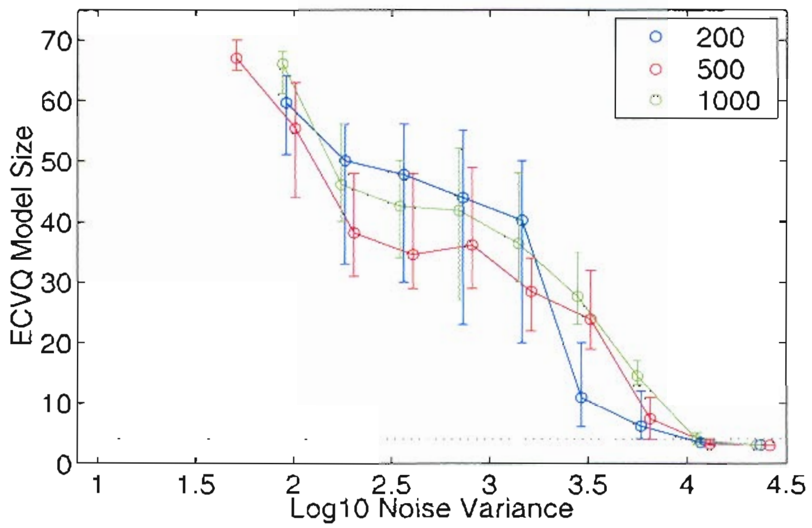
Figure 7.8: Texture Test Data. Data consists of 81-dimensional vectors formed into 9×9 blocks and organized into a map for visualization. Each block sampled is from one of four textures, dense leaves (dark), cloth (coarse texture), marble (gray and white), and paper (light).

accurately. As σ^2 drops, the local dimensions increase, so fewer components are needed and the model size decreases. When σ^2 is very small, the APCA models are nearly full dimension and have too few components like the HGMM models.

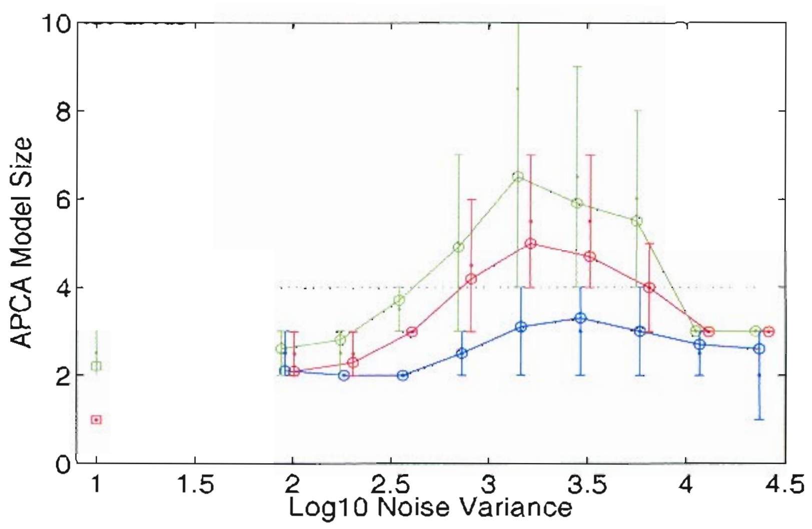
Quantitative Quality Analysis

Correctly identifying the model size does not tell us how accurately the model matches the data. To evaluate goodness of fit, we normalized mutual information on the test data for models developed on each of the training data sets. Normalized mutual information incorporaates measures of component impurity, H_p in Figure 7.10 and component abundance, H_a in Figure 7.11. Figure 7.12 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. Note that the classification ability as measured by H_p is very good for noise variances less than 5000. HGMM has very poor purity values, since the models had fewer than four components, so the normalized mutual information is low. Data from the different texture classes lies close together in the 81-dimensional space, so the HGMM model with one or two full dimensional components covers the data space. Adding components does not reduce the modeling cost as there is no penalty for high-dimensional components.



(a) ECVQ



(b) APCA & HGMM

Figure 7.9: Selected model size for different training set sizes. Plot (a) shows average model size for ECVQ and plot (b) shows average model size for APCA (circles) and HGMM (squares). The HGMM results are plotted at 1 for comparison purposes. The 500 and 200 vector training results for HGMM are identical. Models were trained using 1000 vector (green), 500 vector (red) and 200 vector (blue) set sizes. Error bars indicate maximum and minimum model size for ten different initializations. The correct model size of four is indicated by the dotted line.

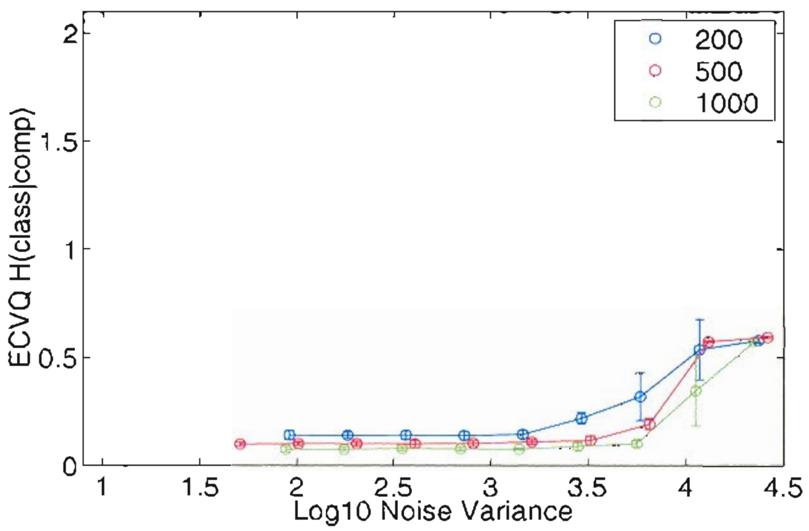
Even when the model size was hand selected to be at least four, most test blocks were attributed to two components and purity values did not improve.

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. Adaptive PCA models have low impurity values and good model sizes for noise variances in the range $1200 < \sigma^2 < 8000$ for all training set sizes. For the larger training sets, the model quality remained good for σ^2 down to 600 for the 500 vector set and 300 for the 1000 vector set. With larger training sets, we can fit more covariance parameters accurately and the models continue to conform to the data structure as we lower the noise variance. For all training set sizes, the APCA models reveal more about the natural cluster structure of the data than either full covariance or spherical models.

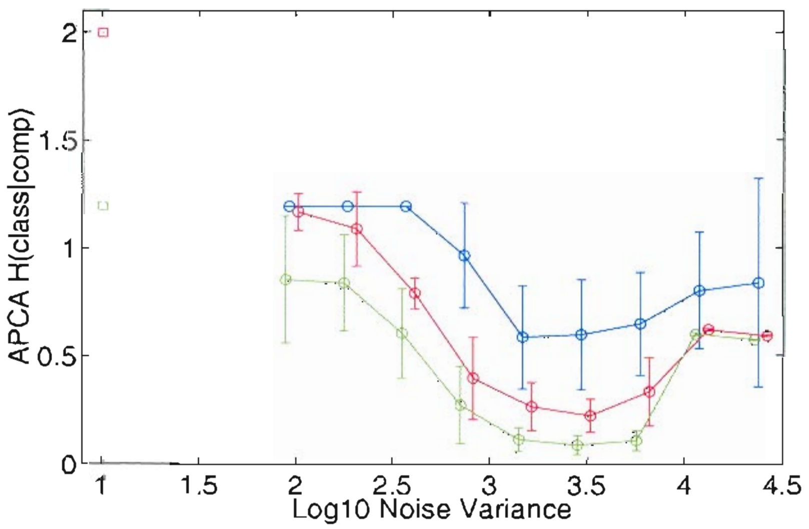
Visual Quality Analysis

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 7.13 and 7.14 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 (see Figure 7.9). The selected σ^2 was 2928 for the 200 vector training set and 2802 for the 1000 vector set.

The test image segmentation results shown in Figure 7.13 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.

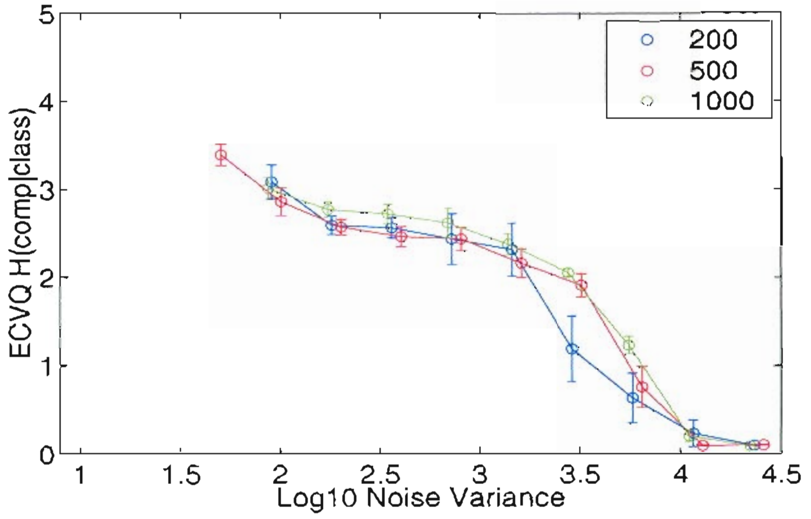


(a) ECVQ

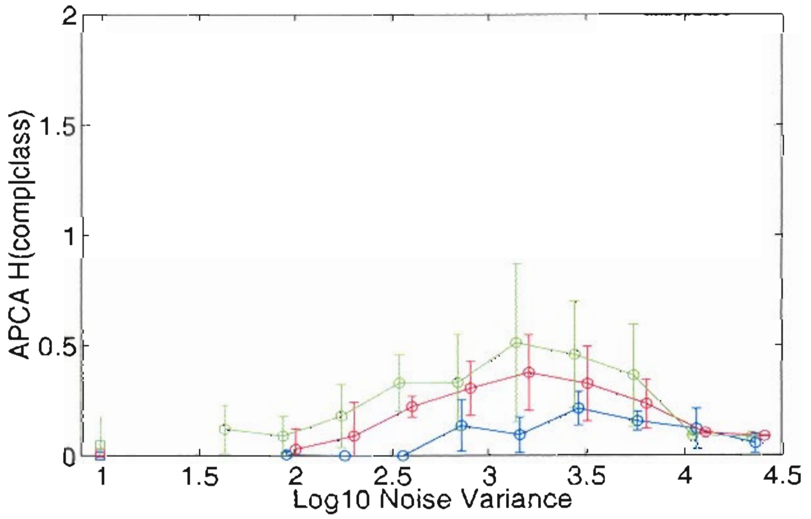


(b) APCA & HGMM

Figure 7.10: Component impurity for different training set sizes. Plot (a) shows H_p for ECVQ and plot (b) shows H_p 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 over ten different initializations. The 500 vector and 200 vector HGMM results are identical.

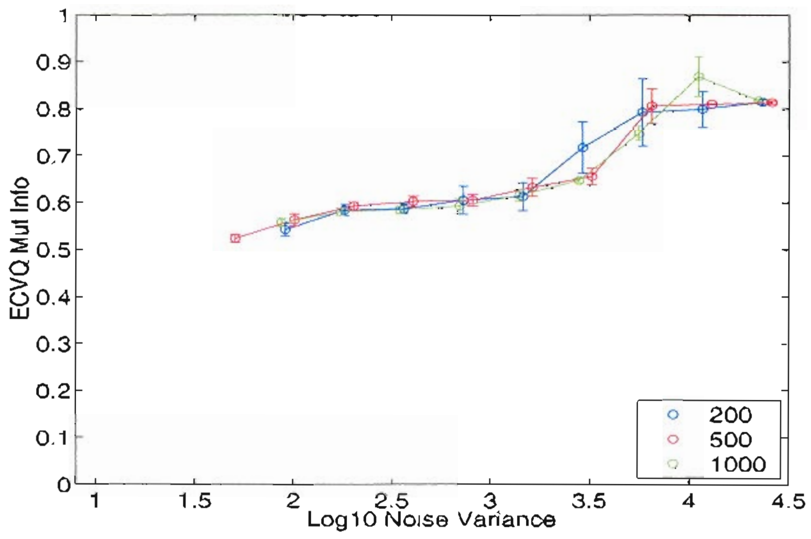


(a) ECVQ

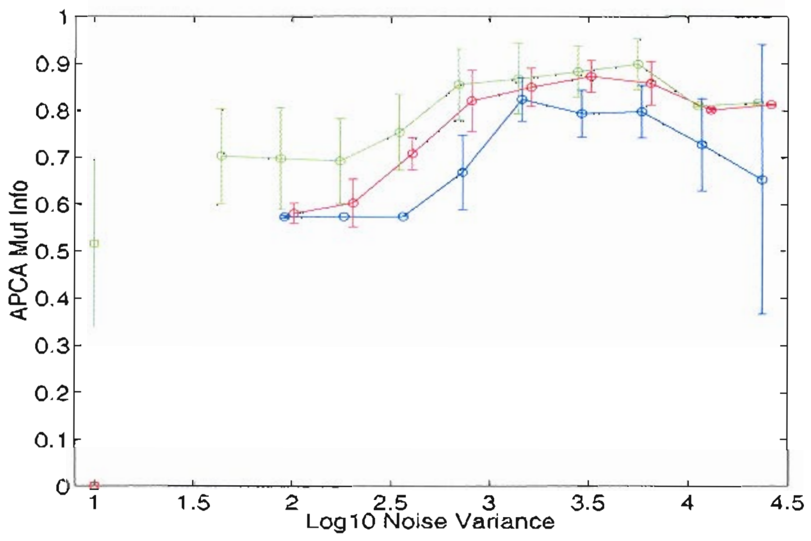


(b) APCA & HGMM

Figure 7.11: Component Abundance for different training set sizes. Plot (a) shows H_a for ECVQ and plot (b) shows H_a 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 results are identical.

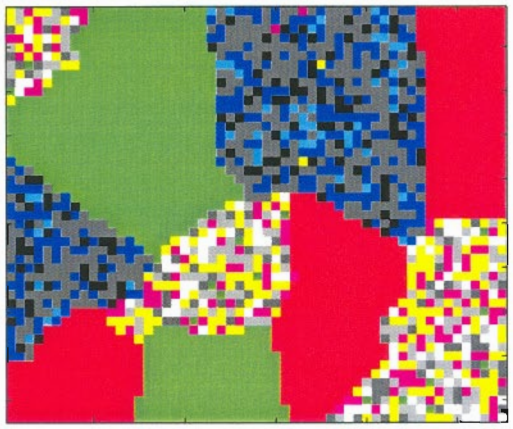


(a) ECVQ



(b) APCA & HGMM

Figure 7.12: 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.


(a) ECVQ

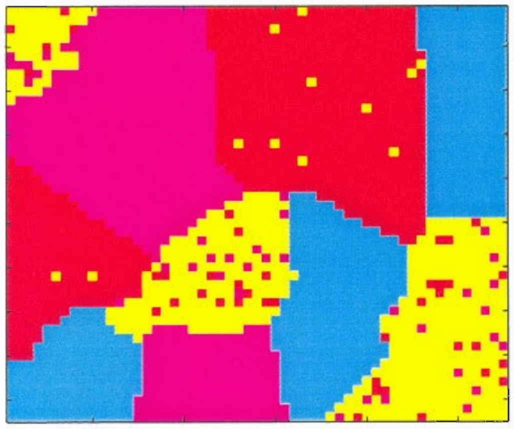
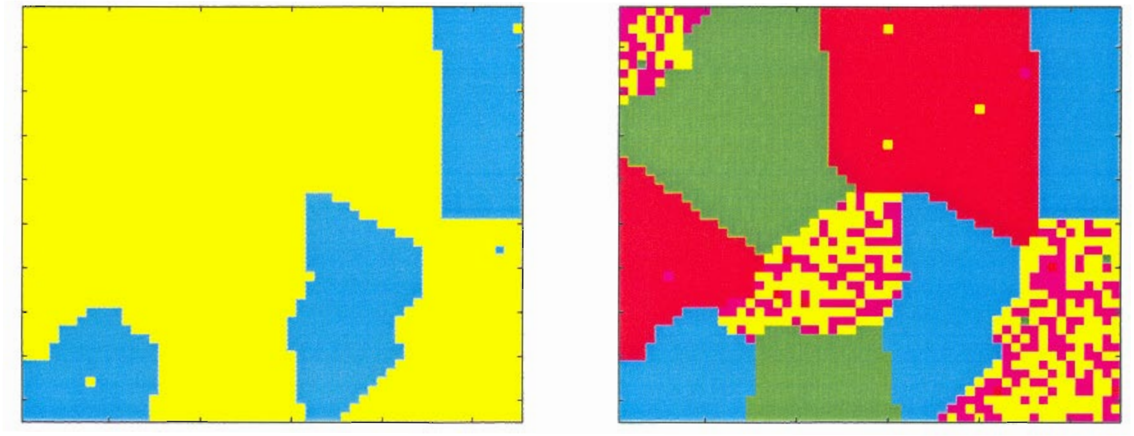

(b) APCA

Figure 7.13: 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.

The segmentation results in Figure 7.14 are for models developed on the 1000 vector training set with $\sigma^2 = 2802$. The ECVQ model (not shown) has 28 components with $H_p = 0.076$ 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 one 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.

7.6 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 GMMs have their own limitations. They either require



(a) HGMM (b) APCA

Figure 7.14: 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.

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.

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 first attempted to selected 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 [MR01] 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. That is, 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. For the work in this chapter, we selected the noise variance at the point where the average model size was largest.

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 provides 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.

Chapter 8

Summary

Statistical signal classification encompasses solution strategies for a number of compelling, practical problems. The list of such problems continues to grow as industry seeks to apply computer-based analysis and decision-making to more real-world phenomena, both man-made and natural. The problems selected for our research, image compression and data modeling, are interesting in their own right. They are also good representatives of this problem space as a whole.

In this thesis, we developed a latent data framework, which facilitates formalizing observations of data behavior into a statistical model. We showed how one can use this framework to develop processing algorithms from statistical models of the data. Using this framework, we developed new algorithms for adaptive transform coding and PCA-based data modeling.

Our primary contributions in the area of adaptive transform coding include the derivation, in closed form, of the optimal linear transform for coding. Using this definition, we develop a new transform coding algorithm that provides an optimal compression solution for non- stationary signals. One useful outcome is that this algorithm provides a standard against which one can evaluate the "goodness" of the PCA (or other) transform for compression.

Our primary contributions in the area of adaptive PCA modeling include development of a new adaptive PCA modeling method that allows model parameters to adjust to local data structure. This algorithm incorporates an entropy penalty, which permits the model to conform to the data structure, when training data is sparse. Our adaptive PCA models match the natural cluster structure of data better than spherical or full covariance modeling methods.

Our work has resulted in new understanding in both the theoretical limits of computer-based processing in these areas and in the relationships between common processing algorithms. We have also demonstrated computationally practical implementations with better performance than existing methods. These same approaches are applicable to a number of problems in the field, e.g. handwritten character recognition, helicopter transmission fault detection, and image segmentation. The improved performance and practical computability of these local model based techniques will enable computer-based solutions for many difficult classification problems.

Bibliography

[ABL02] C. Archer, A. Baptista, and T. K. Leen. Fault detection for salinity sensors in the Columbia estuary. *submitted to Water Resources Research*, 2002.

[AL99] 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*, volume 2, pages 916–920, July 1999.

[AL00] C. Archer and T.K. Leen. Adaptive transform coding as constrained vector quantization. In *Neural Networks for Signal Processing, Proceedings of the IEEE Workshop*, pages 308–317, December 2000.

[AL01a] C. Archer and T.K. Leen. The coding-optimal transform. In Storer and Cohn, editors, *Proceedings Data Compression Conference*, pages 381–390. IEEE Computer Society, March 2001.

[AL01b] 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*, pages 925–931. MIT Press, 2001.

[Bas94] A. Basilevsky. *Statistical Factor Analysis and Related Methods.* John Wiley and Sons, Inc., 1994.

[Bis95] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford Press, 1995.

[BWP$^+$99] 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.

[CLG89] P. Chou, T. Lookabaugh, and R. Gray. Entropy-constrained vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):31–41, 1989.

[CS77] W. Chen and C. Smith. Adaptive coding of monochrome and color images. *IEEE Transactions on Communications*, 25(11):1285–1292, 1977.

[CT91] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley and Sons, Inc., 1991.

[DC93] D. DeMers and G. Cottrell. Non-linear dimensionality reduction. In Giles, Hanson, and Cowan, editors, *Advances in Neural Information Processing Systems 5*, pages 580–587. Morgan Kaufmann, 1993.

[DH95] R. Dony and S. Haykin. Optimally adaptive transform coding. *IEEE Transactions on Image Processing*, 4(10):1358–1370, 1995.

[DLR77] 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.

[ECG99] M. Effros, P. Chou, and R. Gray. Weighted universal image compression. *IEEE Transactions on Image Processing*, 8(10):1317–1328, 1999.

[FM84] N. Farvardin and J. Modestino. Optimum quantizer performance for a class of non-gaussian memoryless sources. *IEEE Transactions on Information Theory*, IT-30(3):485–497, 1984.

[Fre00] T. Freeman. What is Imaging RADAR? Jet Propulsion Laboratories. Available at http://southport.jpl.nasa.gov/desc/imagingradarv3.html, revised February 2000.

[GG92] A. Gersho and R. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic, 1992.

[GH96] 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.

[GL89] G. Golub and C. Van Loan. *Matrix Computations.* John Hopkins University Press, 1989.

[GN98] R. Gray and D. Neuhoff. Quantization. *IEEE Transactions on Information Theory*, 44(6):2325–2383, 1998.

[GZV00] V. Goyal, J. Zhuang, and M. Vetterli. Transform coding with backward adaptive updates. *IEEE Transactions on Information Theory*, 46(4):1623–1633, 2000.

[Ham62] M. Hamermesh. *Group Theory and Its Application to Physical Problems.* Addison-Wesley, 1962.

[Hay91] S. Haykin. *Adaptive Filter Theory.* Prentice Hall, 1991.

[HPF90] T. Hediger, A. Passamante, and M.E. Farrell. Characterizing attractors using local intrinsic dimensions calculated by singular-value decomposition and information-theoretic criteria. *Physical Review*, A 41(10):5325–5332, 1990.

[HRD95] 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.

[KL97] N. Kambhatla and T. K. Leen. Optimal dimension reduction by local PCA. *Neural Computation*, 9(7):1493–1516, 1997.

[Kra91] M. Kramer. Nonlinear prinipal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, February 1991.

[Lab02] Jet Propulsion Laboratories. Space radar images of earth SIR-C/X-SAR. Available at http://www.jpl.nasa.gov/radar/sircxsar, Viewed February 2002.

[LBG80] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[Llo82] S. Lloyd. Least square optimization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. LeCam and J. Neyman, editors, *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[Mal99] S. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 1999.

[MR01] P. Meinicke and H. Ritter. Resolution-based complexity control for gaussian mixture models. *Neural Computation*, 13(2):453–475, 2001.

[NH98] 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.

[Now91] S. Nowlan. *Soft Competitive Adaptation: Neural Network Learning Algorithms based on fitting statistical Mixtures.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1991.

[oK98] University of Karlsruhe. Image sequence server: Durlacher Tor and Ettlinger Tor. Available at i21www.ira.uka.de/image_sequences, Viewed April 1998.

[OT96] 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.

[oW98] University of Waterloo. Waterloo repository image greyset2. Available at http://links.uwaterloo.ca/greyscale2.base.html, Viewed January 1998.

[RG99] S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):306–345, 1999.

[RGF93] 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.

[Ris91] E. Riskin. Optimal bit allocation via the generalized BFOS algorithm. *IEEE Transactions on Information Theory*, 37(2):400–402, 1991.

[Ros98] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.

[Row97] Sam Roweis. EM algorithms for PCA and SPCA. In Solla Jordan, Kearns, editor, *Advances in Neural Information Processing Systems 10*, pages 626–632. MIT Press, 1997.

[SG88] Y. Shoham and A. Gersho. Efficient bit allocation for an arbitrary set of quantizers. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(9):1445–1453, 1988.

[TB99] M. Tipping and C. Bishop. Mixture of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–483, 1999.

[Wal91] G. Wallace. Overview of JPEG (ISO/CCITT) still image compression standard. *Communications of the ACM*, 4(4):30–40, 1991.

[WK85] M. Wax and T. Kailath. Detection of signals by information theoretic criteria. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-33(2):387–392, 1985.

Appendix A

Entropy-Constrained Transform Coder Derivation

This appendix contains the full derivation of our statistical model for global transform coding. We first present the probability model, followed by the optimization of model parameters. We then discuss the correspondence between the probability model and a hard-clustering transform coder. The final section in this appendix includes the derivation of the coding optimal transform for Gaussian data.

A.1 Probability Model for Transform Coder

A transform coder converts a signal to new coordinates and then codes the coordinate values *independently* of one another with scalar quantizers. To replicate this structure, we envision the data as drawn from a latent data space, S, in which the data density can be written as a product of marginal densities

$$\mathrm{p}(s) = \mathrm{p}(s_1, s_2, \ldots, s_d) = \prod_{J=1}^{d} \mathrm{p}_J(s_J)$$

where s_J is the J^{th} component of the d-dimensional vector s.

The latent space is discrete with values $\{q_\alpha,\ \alpha = 1 \ldots M\}$. We model the data density in the latent space with a *constrained* mixture densities,

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

where π_α are the mixing coefficients and $\mathrm{p}(s|\alpha) = \delta(s - q_\alpha)$. We constrain the mixture component means, q_α, to lie at the vertices of a rectangular grid centered at the η. Without loss of generality, we take η to be zero. The grid is defined by the s axes and a set of grid mark values, $\{r_{Ji_J}\}$, where r_{Ji_J} is the i_J^{th} grid mark along the s_J axis. There are M_J possible grid mark values on the s_J axis and the total number of grid vertices $M = \prod_J M_J$. Thus the coordinates of q_α are $[r_{1i_1}, r_{2i_2}, \dots, r_{di_d}]^T$. Conversely, α is a function of $i_1, i_2, \dots, i_d$. Figure A.1 illustrates the latent data space structure.

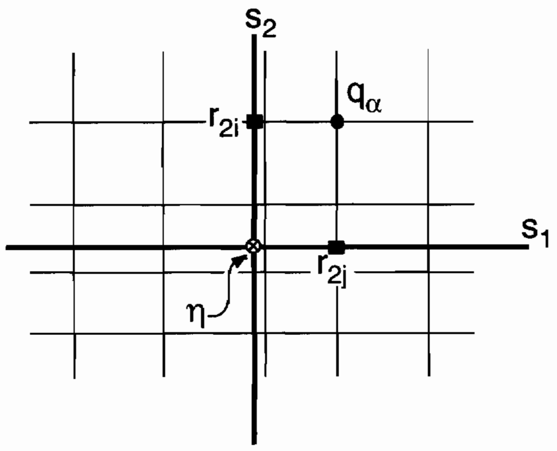


Figure A.1: Structure of latent variable space, S, with single grid. The density on s consists of a mixture of delta functions where the mixture components, q_α, are constrained to lie at the vertices of a rectangular grid. The grid is centered at η and is defined by the s axes and a set of grid mark values $\{r_{Ji}\}$, where r_{Ji} is the i_{th} grid mark along the s_J axis.

The marginal densities are $\mathrm{p}_J(s_J|i_J) = \delta(s_J - r_{Ji_J})$. We write the density of s conditioned on mixture component α as

$$\mathrm{p}(s|\alpha) = \mathrm{p}(s_1, \dots, s_d|\alpha(i_1, \dots, i_d)) = \prod_{J=1}^{d} \mathrm{p}_J(s_J|i_J). \tag{A.2}$$

We also constrain the mixing coefficients, π_α, to be the product of prior probabilities, p_{Ji_J}

$$\pi_{\alpha(i_1,\dots,i_d)} = \prod_J \mathrm{p}_{Ji_J}. \tag{A.3}$$

By combining (A.1), (A.2), and (A.3) the density on s becomes a sum of products of marginal densities

$$\mathrm{p}(s) = \sum_\alpha \prod_J \mathrm{p}_{Ji_J}\ \mathrm{p}_J(s_J|i_J).$$

The sum over the mixture components α is equivalent to sums over all grid mark values for all coordinates

$$\sum_{\alpha=1}^{M} = \sum_{i_1=1}^{M_1} \sum_{i_2=1}^{M_2} \cdots \sum_{i_d=1}^{M_d}$$

so the density on s becomes

$$\begin{aligned} \mathrm{p}(s) &= \sum_{i_1=1}^{M_1} \sum_{i_2=1}^{M_2} \cdots \sum_{i_d=1}^{M_d} \prod_J \mathrm{p}_{Ji_J}\, \mathrm{p}_J(s_J|i_J) \\ &= \sum_{i_1=1}^{M_1} \mathrm{p}_{1i_1}\, \mathrm{p}_1(s_1|i_1) \ldots \sum_{i_d=1}^{M_d} \mathrm{p}_{di_d}\, \mathrm{p}_d(s_d|i_d) \end{aligned}$$

We now write the probability model for s in the desired form of a product of marginal densities

$$\mathrm{p}(s) = \prod_{J=1}^{d} \sum_{i_J=1}^{M_J} \mathrm{p}_{Ji_J}\, \mathrm{p}_J(s_J|i_J). \tag{A.4}$$

The latent data is mapped to the observation space by an orthogonal transformation W and translated by μ. After the mapping, the data is corrupted with additive Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ with mean 0 and variance $\sigma^2\mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Figure A.2 illustrates this mapping from latent to observation space. The observed data generated from a sample s drawn from latent component α is

$$x = W(s - q_\alpha) + \mu + \eta \tag{A.5}$$

with conditional densities

$$\mathrm{p}(x|s,\alpha) = \mathcal{N}(\mu + W(s - q_\alpha), \sigma^2\mathrm{I}) \tag{A.6}$$

The latent density and mapping induces a mixture of constrained Gaussian densities on x of the form

$$\begin{aligned} \mathrm{p}(x) &= \int \sum_\alpha \pi_\alpha \mathrm{p}(x|s,\alpha)\delta(s - q_\alpha)ds \\ &= \sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x|\alpha) \end{aligned} \tag{A.7}$$

with marginal densities

$$\mathrm{p}(x|\alpha) = \mathcal{N}(\mu + Wq_\alpha, \sigma^2\mathrm{I}) \tag{A.8}$$

and the mixing coefficients π_α is given by (A.3).

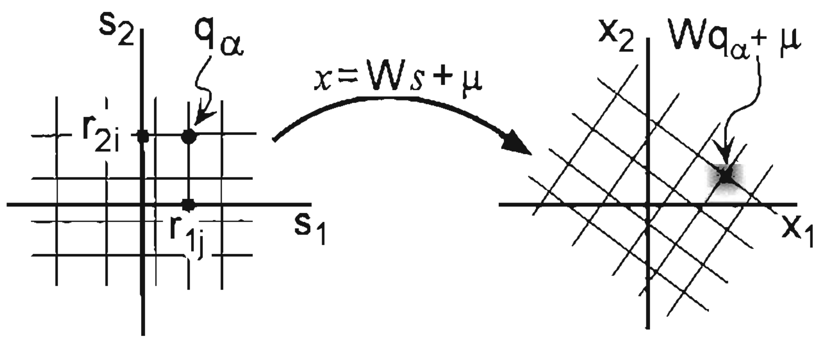


Figure A.2: Structure of latent variable space, S, and mapping to observed space, X. The data density in the latent space consists of a mixture of delta functions where the mixture components, q_α, are constrained to lie at the vertices of a rectangular grid. This grid is mapped to the observed data space by an orthogonal transform, W, and corrupted with additive Gaussian noise.

The expectation-maximization algorithm (EM) [DLR77] fits parametric probability models, such as (A.7), to data by maximizing the data log likelihood for some training set of N data vectors, $\{x_n,\ n = 1 \ldots N\}$.

$$\mathcal{L} = \sum_{n=1}^{N} \log \left(\sum_{\alpha=1}^{M} \pi_\alpha \mathrm{p}(x_n|\alpha) \right) \tag{A.9}$$

To simplify (A.9) we introduce the density $z(\alpha, x_n)$ over these unknown component assignments. Using Jenkin's inequality to bring the sum over α outside the log gives the *expected* log likelihood. The log likelihood $\mathcal{L}$ is bounded below by the expected log likelihood

$$\begin{aligned} \mathcal{L} \geq \langle \mathcal{L} \rangle &= \sum_{n=1}^{N} \sum_{\alpha=1}^{M} z(\alpha, x_n) \log \pi_\alpha - \\ &= \sum_{n=1}^{N} \sum_{\alpha=1}^{M} z(\alpha, x_n) \left(\frac{d}{2} \log(2\pi\sigma^2) + \frac{1}{2\sigma^2} |x_n - \mu - W q_\alpha|^2 \right) - \\ & \sum_{n=1}^{N} \sum_{\alpha=1}^{M} z(\alpha, x_n) \ln z(\alpha, x_n) \end{aligned} \tag{A.10}$$

with equality when $z(\alpha, x) = \mathrm{p}(\alpha|x)$ is the posterior probability of component α conditioned on the data vector x [NH98]. The posterior probabilites are given by

$$\mathrm{p}(\alpha|x_n) = \frac{\pi_\alpha \mathrm{p}(x_n|\alpha)}{\sum_\beta \pi_\beta \mathrm{p}(x_n|\beta)} \tag{A.11}$$

A.2 Parameter Estimation

The EM algorithm fits our model to data by finding values for the model parameters that maximize the expected data log likelihood (A.10). The model parameters are:

1. the grid translation, μ,
2. the transformation matrix, W, which specifies the orientation of the grid,
3. the grid mark values, r_{Ji_J}, $J = 1 \ldots d$ and $i_J = 1 \ldots M_J$ that determine the component means, q_α, $\alpha = 1 \ldots M$,
4. the priors, p_{Ji_J}, $J = 1 \ldots d$ and $i_J = 1 \ldots M_J$ that determine the mixing coefficients π_α, $\alpha = 1 \ldots M$, and
5. the number of grid mark values in each direction, M_J, $J = 1 \ldots d$.

The noise variance σ^2 is not fit to data, by is selected to control model complexity as measured by the entropy

$$H = -\sum_\alpha \pi_\alpha \ln \pi_\alpha = -\sum_J \sum_{i_J} \mathrm{p}_{Ji_J} \ln \mathrm{p}_{ji_J} \tag{A.12}$$

Choosing σ^2 large will produce low complexity models with low H values. Conversely, choosing σ^2 small will produce high entropy models with many components.

A.2.1 Transform Optimization

To optimize the transform related parameters, we find the center μ and orientation W of each quantizer grid that maximizes the likelihood (A.10). The minimum cost estimators for the grid center place the grid at the mean of the data.

$$\mu = \frac{1}{N} \sum_x x \tag{A.13}$$

The optimal transform W_0 is a member of the special orthogonal group in d dimensions, SO(d). For background on group theory see [Ham62]. The group has $(d^2 - d)/2$ parameters. Consider a curve in the group $W(\lambda)$, $\lambda \in R$ that passes through W_0 at $\lambda = 0$, but is otherwise arbitrary. Maximizing the likelihood (A.10) along such a curve yields

$$-\frac{1}{2\sigma^2}\sum_\alpha\sum_n \mathrm{p}(\alpha|x_n)\,(x_n-\mu)^T\,\frac{\mathrm{d}W}{\mathrm{d}\lambda}\bigg|_{\lambda=0} q_\alpha \;=\; 0 \tag{A.14}$$

where we have used the orthogonality of W to simplify the expression. Defining the matrix

$$Q = \sum_\alpha q_\alpha\left(\sum_n \mathrm{p}(\alpha|x_n)\,(x_n-\mu)^T\right) \tag{A.15}$$

allows us to write (A.14) as

$$\mathrm{Trace}\left[\frac{\mathrm{d}W}{\mathrm{d}\lambda}\bigg|_{\lambda=0} Q\right] \;=\; 0\,. \tag{A.16}$$

The derivative along $W(\lambda)$ is

$$\frac{\mathrm{d}W}{\mathrm{d}\lambda}\bigg|_{\lambda=0} = \sum_{j=1}^{\frac{d^2-d}{2}} c_j \mathrm{L}_j W_0 \tag{A.17}$$

where the L_j are the generators of the Lie algebra so(d) tangent to the group at the identity. The c_j are scalars determined by the specific curve. Postmultiplying L_j by W_0 moves the tangent vectors from the identity to W_0. Substituting (A.17) into (A.16) and using the fact that we consider *any* curve through W_0, hence arbitrary c_j, we recover

$$\mathrm{Trace}[\,\mathrm{L}_j W_0\, Q\,] = 0 \quad \text{for } j = 1\ldots\frac{d^2-d}{2}\,. \tag{A.18}$$

Since the L_j's are antisymmetric, (A.18) requires that $W_0 Q$ is symmetric. This symmetry condition and the orthogonality condition, $W^T W = \mathbf{I}$, uniquely determine W.

Alternately, we can derive the symmetry condition by examining how the likelihood changes with changes in elements of W. W is constrained to be orthogonal, so we minimize the constrained likelihood (dropping terms that do not depend on W)

$$\mathcal{L}_c = \sum_{n=1}^{N}\sum_{\alpha=1}^{M} \mathrm{p}(\alpha|x_n)\frac{-1}{2\sigma^2}\,|(x_n-\mu) - \sum_{K=1}^{d} W_K q_{\alpha K}|^2 + \sum_{K=1}^{d}\sum_{J=1}^{d}\lambda_{KJ}(W_K^T W_J - \delta_{KJ}) \tag{A.19}$$

where W_K is the K^{th} column of the W matrix and $q_{\alpha K}$ is the K^{th} coordinate of q_α. The change in likelihood with respect to a change in one or more elements of W_L is

$$\begin{aligned}\delta\mathcal{L}_c &= \sum_n\sum_\alpha \mathrm{p}(\alpha|x_n)\,\frac{-1}{2\sigma^2}\Bigg(((x_n-\mu)-\sum_K W_K q_{\alpha K})^T(-\delta W_L\, q_{\alpha L}) + \\ &\qquad (-\delta W_L\, q_{\alpha L})^T((x_n-\mu)-\sum_K W_K q_{\alpha K})\Bigg) + \sum_K(\lambda_{KL}+\lambda_{LK})W_K^T\,\delta W_L\end{aligned}$$

where $\delta W_L = [\delta W_{1L}, \delta W_{2L}, \ldots, \delta W_{dL}]^T$ is a vector of small, arbitrary changes in W. Since W is orthogonal, $W_K^T\,\delta W_L + \delta W_K^T\, W_L = 0$. Consquently, the terms containing $q_{\alpha K}q_{\alpha L}$ cancel and the above equation simplifies to

$$\delta\mathcal{L}_c = \left(\sum_n\sum_\alpha \mathrm{p}(\alpha|x_n)\frac{1}{\sigma^2}q_{\alpha L}(x_n-\mu)^T + \sum_K(\lambda_{KL}+\lambda_{LK})W_K^T\right)\delta W_L \qquad \text{(A.20)}$$

At a maximum of the likelihood, $\delta\mathcal{L}_c$ is zero. Since the change in W_L is arbitrary, this means the term in parenthesis must be zero. Post-multiplying by W_J and using the orthgonality of W yields

$$\sum_n\sum_\alpha \mathrm{p}(\alpha|x_n)\; q_{\alpha L}\;(x_n-\mu)^T W_J = -\sigma^2(\lambda_{JL}+\lambda_{LJ}) = \sum_n\sum_\alpha \mathrm{p}(\alpha|x_n)\; q_{\alpha J}\;(x_n-\mu)^T W_L \qquad \text{(A.21)}$$

or QW is symmetric

$$QW = W^T Q^T \qquad \text{(A.22)}$$

where Q is given by (A.15).

The symmetry condition (A.22) along with the orhogonality condition uniquely defines the coding optimal transform (COT). The transform W contains $d \times g$ elements, where d is the data dimenision and $g \leq d$ are the number of scalar quantizers with *more* than one reproduction value. Therefore, we require dg equations to uniquely specify W. The symmetry condition (A.22) provides $g(g-1)/2 + (d-g)g$ equations and the orthogonality condition, $W^TW = \mathbf{I}$, provides $g(g+1)/2$ equations for the rquired total of dg equations.

A.2.2 Likelihood in Transform Coordinates

To determine the rest of the model parameters, it is easier to work in the transform coordinates defined by W. Hence, we rewrite the likelihood using $y^{(n)} = W^T(x_n - \mu)$

$$\mathcal{L} = \sum_{n=1}^{N}\sum_{\alpha=1}^{M} \mathrm{p}(\alpha|y^{(n)})\left[\ln\pi_\alpha + \ln\mathrm{p}(y^{(n)}|\alpha) - \ln\mathrm{p}(\alpha|y^{(n)})\right] \qquad \text{(A.23)}$$

where $\mathrm{p}(y|\alpha) = \mathcal{N}(q_\alpha, \sigma^2\mathbf{I})$. Note that in the limit that the noise variance goes to zero, $\mathrm{p}(y) = \mathrm{p}(s)$. Since $W^TW = \mathbf{I}$, the two forms for the data log likelihood, (A.10) and (A.23), are equivalent.

Next, we write $\mathrm{p}(\alpha|y)$ in terms of the priors of the grid mark values and the transform coefficients. The transform coefficients are $y_J^{(n)} = W_J^T x_n$, where W_J is the J^{th} column vector of W.

$$\begin{aligned}
\mathrm{p}(\alpha|y^{(n)}) &= \frac{\pi_\alpha \mathrm{p}(y^{(n)}|\alpha)}{\mathrm{p}(y^{(n)})} \\
&= \frac{\prod_J \mathrm{p}_{Ji_J}\mathrm{p}_J(y_J^{(n)}|i_J)}{\prod_K \sum_{i_K} \mathrm{p}_{Ki_K}\mathrm{p}_K(y_K^{(n)}|i_K)} \\
&= \prod_{J=1}^{d} \frac{\mathrm{p}_{Ji_J}\mathrm{p}(y_J^{(n)}|i_J)}{\mathrm{p}_J(y_J^{(n)})} \\
&= \prod_{J=1}^{d} \mathrm{p}_J(i_J|y_J^{(n)}) \qquad \text{(A.24)}
\end{aligned}$$

Replacing the sums over α in (A.23) with corresponding sums over the grid values yields

$$\begin{aligned}
\mathcal{L} &= \sum_{n=1}^{N}\sum_{i_1=1}^{M_1}\cdots\sum_{i_d=1}^{M_d}\prod_K \mathrm{p}_K(i_K|y_K^{(n)}) \left[\ln(\prod_J \mathrm{p}_{Ji_J}) + \ln(\prod_J \mathrm{p}_J(y_J^{(n)}|i_J))\right] \\
&= \sum_{n=1}^{N}\sum_{i_1=1}^{M_1}\cdots\sum_{i_d=1}^{M_d}\prod_K \mathrm{p}_K(i_K|y_K^{(n)}) \left[\sum_J \ln \mathrm{p}_{Ji_J} + \sum_J \ln \mathrm{p}_J(y_J^{(n)}|i_J)\right] \\
&= \sum_{n=1}^{N}\sum_{i_1=1}^{M_1}\mathrm{p}_1(i_1|y_1^{(n)})\cdots\sum_{i_d=1}^{M_d}\mathrm{p}_d(i_d|y_d^{(n)}) \left[\sum_J \ln \mathrm{p}_{Ji_J} + \sum_J \ln \mathrm{p}_J(y_J^{(n)}|i_J)\right] \qquad \text{(A.25)}
\end{aligned}$$

To keep the equations short in the derivation below, we will show just the first term in the square brackets above. The second term is treated identically. Breaking out the terms for the last coordinate, d, gives us

$$\begin{aligned}
\mathcal{L} &= \sum_{n=1}^{N}\sum_{i_1=1}^{M_1}\mathrm{p}_1(i_1|y_1^{(n)})\cdots\sum_{i_{d-1}=1}^{M_{d-1}}\mathrm{p}_{d-1}(i_{d-1}|y_{d-1}^{(n)})\sum_{i_d=1}^{M_d}\mathrm{p}_d(i_d|y_d^{(n)}) \left(\sum_{J=1}^{d-1}\ln \mathrm{p}_{Ji_J} + \ln \mathrm{p}_{di_d}\right) \\
&= \sum_{n=1}^{N}\sum_{i_1=1}^{M_1}\mathrm{p}_1(i_1|y_1^{(n)})\cdots\sum_{i_{d-1}=1}^{M_{d-1}}\mathrm{p}_{d-1}(i_{d-1}|y_{d-1}^{(n)})\left(\sum_{J=1}^{d-1}\ln \mathrm{p}_{Ji_J}\right)\left(\sum_{i_d=1}^{M_d}\mathrm{p}_d(i_d|y_d^{(n)})\right) + \\
&\quad \sum_{n=1}^{N}\sum_{i_d=1}^{M_d}\mathrm{p}_d(i_d|y_d^{(n)})\ln \mathrm{p}_{di_d}\left(\sum_{i_1=1}^{M_1}\mathrm{p}_1(i_1|y_1^{(n)})\cdots\sum_{i_{d-1}=1}^{M_{d-1}}\mathrm{p}_{d-1}(i_{d-1}|y_{d-1}^{(n)})\right) \qquad \text{(A.26)}
\end{aligned}$$

Since $\sum_{i_J=1}^{M_J} \mathrm{p}_J(i_J|y_J^{(n)}) = 1$ for every coordinate $J = 1 \dots d$, (A.26) becomes

$$\mathcal{L} = \sum_{n=1}^{N} \sum_{i_1=1}^{M_1} \mathrm{p}_1(i_1|y_1^{(n)}) \cdots \sum_{i_{d-1}=1}^{M_{d-1}} \mathrm{p}_{d-1}(i_{d-1}|y_{d-1}^{(n)}) \left(\sum_{J=1}^{d-1} \ln \mathrm{p}_{Ji_J} \right) + \sum_{n=1}^{N} \sum_{i_d=1}^{M_d} \mathrm{p}_d(i_d|y_d^{(n)}) \ln \mathrm{p}_{di_d} \tag{A.27}$$

Repeating the above simplification for each remaining coordinate, $J = d-1 \dots 1$, yields

$$\begin{aligned} \mathcal{L} &= \sum_{n=1}^{N} \sum_{J=1}^{d} \sum_{i_J=1}^{M_J} \mathrm{p}_J(i_J|y_J^{(n)}) \left[\ln \mathrm{p}_{Ji_J} + \ln \mathrm{p}_J(y_J^{(n)}|i_J) \right] \\ &= \sum_{n=1}^{N} \sum_{J=1}^{d} \sum_{i_J=1}^{M_J} \mathrm{p}_J(i_J|y_J^{(n)}) \left[\ln \mathrm{p}_{Ji_J} - \frac{1}{2} \ln 2\pi\sigma^2 - \frac{1}{2\sigma^2} |y_J^{(n)} - r_{Ji_J}|^2 \right] \end{aligned} \tag{A.28}$$

A.2.3 Quantizer Optimization

Maximizing the log likelihood (A.28) with respect to grid value, r_{Kk} yields

$$\frac{\partial \mathcal{L}}{\partial r_{Kk}} = \sum_n \mathrm{p}_K(k|y_K^{(n)}) \, \frac{1}{\sigma^2} \, (y_K^{(n)} - r_{Kk}) = 0$$

$$r_{Kk} = \frac{\sum_n \mathrm{p}_K(k|y_K^{(n)}) \, y_K^{(n)}}{\sum_n \mathrm{p}_K(k|y_K^{(n)})} \tag{A.29}$$

The reproduction vectors, $q_{\alpha(i_1,\dots,i_d)}$ are given by $[r_{1i_1}, \dots, r_{di_d}]^T$.

The number of grid values M_K in each coordinate is influenced by the choice of component or noise variance σ^2. The number of components in each dimension is related to the ratio of the data variance in that dimension to the component variance. When σ^2 is small, many components are required to cover the data space, so the likelihood maximum will occur at large values of M and M_K. When σ^2 is large, fewer components are needed to maximize the likelihood.

Next we maximize the likelihood (A.28) with respect to the prior probabilites of the grid values. The priors for each coordinate must sum to one. Consequently, we maximize the constrained likelihood

$$\mathcal{L}_c = \mathcal{L} + \sum_J \gamma_J \left(\sum_{i_J} \mathrm{p}_{Ji_J} - 1 \right) \tag{A.30}$$

Maximizing (A.30) yields

$$\frac{\partial \mathcal{L}_c}{\partial \mathrm{p}_{Kk}} = \sum_n \mathrm{p}_K(k|y_K^{(n)}) \frac{1}{\mathrm{p}_{Kk}} + \gamma_K = 0.$$

Applying the constraint, we find that $\gamma_K = -N$. Replacing γ_K in the above equation yields

$$\mathrm{p}_{Kk} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{p}_K(k|y_K^{(n)}) \tag{A.31}$$

The mixing coefficients are calculated from the grid value priors, $\pi_{\alpha(i_1,\ldots,i_d)} = \prod_J \mathrm{p}_{Ji_J}$.

A.3 Controling Model Complexity

Instead of finding the maximum likelihood estimator for the noise variance σ^2, we adjust the variance to control complexity as measured by the entropy (A.12). We select a value of σ^2 that keeps the model entropy below some target value H_0. Entropy is influenced by the values of all the model parameters, the transform, the grid mark values, the grid mark prior probabilities, and the variance. To keep the solution on the constraint surface, $H = H_0$, small changes in the parameter values should cancel each other.

$$\sum_{J=1}^{d}\sum_{i_J=1}^{M_J} \frac{\partial H}{\partial \mathrm{p}_{Ji_J}} \mathrm{dp}_{Ji_J} + \sum_{J=1}^{d}\sum_{i_j=1}^{M_J} \frac{\partial H}{\partial r_{Ji_J}} \mathrm{d}r_{Ji_J} + \sum_{J=1}^{d}\sum_{K=1}^{d} \frac{\partial H}{\partial W_{KJ}} \mathrm{d}W_{KJ} + \frac{\partial H}{\partial \sigma^2}\mathrm{d}\sigma^2 = 0 \tag{A.32}$$

The variance σ^2 is an implicit function of W, the p_{Ji_J}'s and r_{Ji_J}'s, $J = 1 \ldots d$ and $i_J = 1 \ldots M_J$. So, for given W, p_{Ji_J}s and r_{Ji_J}s, we can estimate the (first-order) change in σ^2 necessary to make $H = H_0$.

$$\mathrm{d}\sigma^2 = \frac{H_0 - H}{\partial H/\partial \sigma^2} \tag{A.33}$$

The derivative of H with respect to σ^2 is

$$\frac{\partial H}{\partial \sigma^2} = -\frac{1}{N}\sum_J\sum_i \frac{\partial \hat{p}_{Ji}}{\partial \sigma^2}\left(1 + \ln \hat{p}_{Ji}\right). \tag{A.34}$$

where hatted parameters ($\hat{\cdot}$) are those found using the current posterior values and non-hatted parameters are those used to calculate the current posterior values. The values of $\hat{p}_{Ji}$ are given by (A.31), so (A.34) becomes

$$\frac{\partial H}{\partial \sigma^2} = -\frac{1}{N}\sum_J\sum_i\sum_n \frac{\partial \mathrm{p}_J(i|s_J^{(n)})}{\partial \sigma^2}\left(1 + \ln \hat{p}_{Ji}\right). \tag{A.35}$$

The $\mathrm{p}_J(i|y_J)$, $i = 1 \ldots M_J$, are given by

$$\mathrm{p}_J(i|y_J) = \frac{\mathrm{p}_{Ji}\ \exp\left(\frac{-1}{2\sigma^2}|y_J - r_{Ji}|^2\right)}{\sum_j \mathrm{p}_{Jj}\ \exp\left(\frac{-1}{2\sigma^2}|y_J - r_{Jj}|^2\right)} \tag{A.36}$$

where $y_J = W_J^T x$. Taking the derivative with respect to σ^2,

$$\begin{aligned}
\frac{\partial \mathrm{p}_J(i|y_J)}{\partial\sigma^2} &= \frac{\mathrm{p}_{Ji}\,|y_J - r_{Ji}|^2\,\frac{1}{2\sigma^4}\exp\,\frac{-1}{2\sigma^2}|y_J - r_{Ji}|^2}{\sum_j \mathrm{p}_{Jj}\ \exp\,\frac{-1}{2\sigma^2}|y_J - r_{Jj}|^2} - \\
&\quad \frac{\mathrm{p}_{Ji}\ \exp\,\frac{-1}{2\sigma^2}|y_J - r_{Ji}|^2}{\left(\sum_j \mathrm{p}_{Jj}\ \exp\,\frac{-1}{2\sigma^2}|y_J - r_{Jj}|^2\right)^2}\sum_k \frac{\mathrm{p}_{Jk}|y_J - r_{Jk}|^2}{2\sigma^4}\exp\left(\frac{-1}{2\sigma^2}|y_J - r_{Jk}|^2\right) \\
&= \mathrm{p}_J(i|y_J)\ |y_J - r_{Ji}|^2\frac{1}{2\sigma^4} - \mathrm{p}_J(i|y_J)\sum_k \mathrm{p}_J(k|y_J)\ |y_J - r_{Jk}|^2\frac{1}{2\sigma^4} \\
&= \frac{1}{2\sigma^4}\mathrm{p}_J(i|y_J)\left(|y_J - r_{Ji}|^2 - \sum_k \mathrm{p}_J(k|y_J)\ |y_J - r_{Jk}|^2\right)
\end{aligned} \tag{A.37}$$

Substituting (A.37) into (A.35) and simplifying

$$\frac{\partial H}{\partial\sigma^2} = \frac{1}{2\sigma^4}\frac{1}{N}\sum_{J=1}^{d}\sum_{i=1}^{M_J}\sum_n \mathrm{p}_J(i|y_J)\ \ln\hat{p}_{Ji}\left(\sum_j \mathrm{p}_J(j|y_J^{(n)})\ |y_J^{(n)} - r_{Jj}|^2 - |y_J^{(n)} - r_{Ji}|^2\right) \tag{A.38}$$

Substituting this result into (A.33) yields

$$\mathrm{d}\sigma^2 = \frac{2\sigma^4(H_0 - H)}{\frac{1}{N}\sum_J\sum_i\sum_n \mathrm{p}_J(i|y_J^{(n)})\ \ln\hat{p}_{Ji}\left(\sum_j \mathrm{p}_J(j|y_J^{(n)})\ |y_J^{(n)} - r_{Jj}|^2 - |y_J^{(n)} - r_{Ji}|^2\right)} \tag{A.39}$$

A.4 Model and Transform Coder Correspondence

The EM algorithm provides a template fo deriving a transform coding algorithm from this probability model. To achieve hard clustering, we choose choose $z(\alpha, x_n)$ to be

$$z(\alpha, x_n) = \begin{cases} 1 & \mathrm{p}(\alpha|x_n) > \mathrm{p}(\gamma|x_n)\ \forall\gamma \neq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{A.40}$$

In this hard clustering model the responsibility for a data value x_n is assigned to the mixture component with the largest posterior. The final term in the expected log likelihood (A.10) becomes zero since $z(\alpha, x_n) \ln z(\alpha, x_n) = 0 \; \forall \alpha, n$. Consequently, $\langle \mathcal{L} \rangle$ reduces to the cost function

$$\mathcal{C} = \sum_{\alpha=1}^{M} \sum_{n=1}^{N} z(\alpha, x_n) \left(\|x_n - \mu - Wq_\alpha\|^2 - 2\sigma^2 \log \pi_\alpha \right) \tag{A.41}$$

This cost function consists of two terms combined with the Lagrange multiplier $2\sigma^2$: the average coding distortion $\sum_\alpha \sum_n z(\alpha, x_n) \|x_n - \mu - Wq_\alpha\|^2$ and the entropy $-\sum_\alpha \pi_\alpha \log \pi_\alpha$. This *entropy-constrained* cost function (A.41) is the same as that found by minimizing coding distortion subject to an *average* bit-rate constraint, as we show below.

This partition (A.40) defines regions

$$R_\alpha = \{x \mid |x - Wq_\alpha|^2 - 2\sigma^2 \ln \pi_\alpha \leq |x - Wq_\gamma|^2 - 2\sigma^2 \ln \pi_\gamma \; \forall \, \gamma\}. \tag{A.42}$$

Rewriting these region definitions in terms of the transform coefficients, $y_J = W_J^T(x - \mu)$, yields

$$R_{\alpha(i_1,\ldots,i_d)} = \{x \mid y_1 \in R_{1i_1} \text{ and } y_2 \in R_{2i_2} \text{ and } \cdots \text{ and } y_d \in R_{di_d}\} \tag{A.43}$$

where

$$R_{Jj} = \{y_J \mid (y_J - r_{Jj})^2 - 2\sigma^2 \ln \mathrm{p}_{Jj} \leq (y_J - r_{Jl})^2 - 2\sigma^2 \ln \mathrm{p}_{Jl} \;\; \forall \, l\} \tag{A.44}$$

This is also the optimal partition of the signal space for an entropy-constrained transform coder.

An entropy-constrained transform coder limits compressed signal size to some target bit-rate, H_0.

$$l = -\sum_{J=1}^{d} \sum_{i_J=1}^{M_J} \mathrm{p}_{Ji_J} \, l_{Ji_J} \leq H_0 \tag{A.45}$$

where l_{Ji_J} is the length of the code word for reproduction value r_{Ji_J}.

The cost of compressing a signal is

$$D = \sum_{J=1}^{d} \sum_{i_J=1}^{M_J} \mathrm{p}_{Ji_J} \left(\frac{1}{N_{Ji_J}} \sum_{y_J \in R_{Ji_J}} |y_J - r_{Ji_J}|^2 \right) - \lambda \left(\sum_{J=1}^{d} \sum_{i_J=1}^{M_J} \mathrm{p}_{Ji_J} \, l_{Ji_J} - H_o \right) \tag{A.46}$$

where N_{Ji_J} are the number of y_J in bin R_{Ji_J}. In the hard clustering model, finding the model parameters that maximize the likelihood (A.28) is the same as finding the transform coder parameters that minimize constrained distortion (A.46). The model grid values (A.29) correspond to the optimal scalar quantizer reproduction values

$$r_{Kk} = \frac{1}{N_{Kk}} \sum_{y_K \in R_{Kk}} y_K \tag{A.47}$$

The model grid value priors (A.31) correspond to the prior probabilites of the bins, $\mathrm{p}_{Kk} = N_{Kk}/N$.

The length of each code word is the logarithm of the associated prior probability. Kraft's inequality states that a uniquely decodable code must satisfy

$$\sum_{i_J=1}^{M_J} 2^{-l_{Ji_J}} \leq 1.$$

We add this constraint to (A.46), using Lagrange multipliers γ_J

$$C = D + \sum_J \gamma_J \left(\sum_{i_J} 2^{-l_{Ji_J}} - 1 \right) \tag{A.48}$$

If we do not constrain the code lengths to be integers, we can take the derivative of the cost with respect to the code lengths.

$$\frac{\partial C}{\partial l_{Kk}} = \mathrm{p}_{Kk} - \gamma_K \ln 2 \; 2^{-l_{Kk}} \; = \; 0$$

Applying Kraft's inequality yields $\gamma_K = 1/\ln 2$. The optimal code lengths are

$$l_{Kk} = -\log_2 \mathrm{p}_{Kk} \tag{A.49}$$

Using optimal code lengths, the average bit-rate becomes

$$l = -\sum_J \sum_{i_J} \mathrm{p}_{Ji_J} \log_2 \mathrm{p}_{Ji_J},$$

which is the entropy. The entropy constraint determines the number of reproduction values K_J in each scalar quantizer. The rate or entropy terms in (A.46) move the partition away from the minimium distortion solution, so that reproduction values with low prior probabilities may have no data items assigned to them. Reproduction values with $\mathrm{p}_{Ji} = 0$ can be removed from the coder, reducing the value of K_J. Consequently, selecting a small entropy H_0 produces small quantizers and low bit-rate coders.

Substituting the solution for l_{Ji} into the cost equation (A.46) yields

$$D = \sum_{J=1}^{d} \sum_{i_J=1}^{M_J} \sum_{y_J \in R_{Ji_J}} |y_J - r_{Ji_J}|^2 + \lambda \log \mathrm{p}_{Ji_J} \tag{A.50}$$

where we select the Lagrange multiplier λ to enforce the entropy constraint. The variance σ^2 in (A.41) corresponds to the Lagrange multiplier λ.

A.5 COT for Gaussian Data

The optimal coding transform for Gaussian data is the PCA transform or KLT. For zero-mean d dimensional Gaussian data, x, $\mathrm{p}(x) = \mathcal{N}(0, \Sigma)$, where Σ is the covariance matrix. The density of the transform coefficients, $y = W^T(x - \mu)$, is

$$\mathrm{p}(y) = \mathcal{N}(0, W^T \Sigma W) \tag{A.51}$$

To proceed, we need to find QW in terms of the grid mark values, r, and transform coefficients, y. QW is given by

$$QW = \sum_{\alpha} q_\alpha \left(\int_{R_\alpha} \mathrm{p}(x) \mathrm{d}^d x \; x^T \right) W \tag{A.52}$$

The value in the K_{th} row and J^{th} column, $\forall K \neq J$ is

$$(QW)_{KJ} = \sum_{\alpha=1}^{M} q_{\alpha K} \int_{R_\alpha} \mathrm{p}(x) \mathrm{d}^d x W_J^T x \tag{A.53}$$

where $q_{\alpha K}$ is the K^{th} coordinate of q_α. Now $q_{\alpha K} = r_{Ki_K}$ and $W_J^T(x - \mu) = y_J$. Making these substitutions and replacing the sums over α with the corresponding sums over i_J and the integrals over R_α with the corresponding integrals over R_{Ji_J}, for $J = 1 \ldots d$, yields

$$\begin{aligned}
(QW)_{KJ} &= \sum_{i_1} \cdots \sum_{i_d} \int_{R_{1i_1}} \cdots \int_{R_{di_d}} r_{Ki_K} y_J \mathrm{p}(y_1, \ldots, y_d) \mathrm{d}y_1 \ldots \mathrm{d}y_d \\
&= \sum_{i_K} r_{Ki_K} \sum_{i_J} \int_{R_{Ji_J}} y_J \int_{R_{Ki_K}} \sum_{i_1} \int_{R_{1i_1}} \cdots \sum_{i_d} \int_{R_{di_d}} \mathrm{p}(y_1, \ldots, y_d) \mathrm{d}y_1 \ldots \mathrm{d}y_d \\
&= \sum_{i_K} r_{Ki_K} \sum_{i_J} \int_{R_{Ji_J}} y_J \int_{R_{Ki_K}} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \mathrm{p}(y_1, \ldots, y_d) \mathrm{d}y_1 \ldots \mathrm{d}y_d \mathrm{d}y_K \mathrm{d}y_J \\
&= \sum_{i_K} r_{Ki_K} \sum_{i_J} \int_{R_{Ji_J}} y_J \int_{R_{Ki_K}} \mathrm{p}(y_J, y_K) \mathrm{d}y_K \mathrm{d}y_J
\end{aligned} \tag{A.54}$$

Repeating this process for $K = J$, we find the tha diagonal elements of QW are

$$(QW)_{JJ} = \sum_{i_J} r_{Ji_J} \int_{R_{Ji_J}} y_J \mathrm{p}(y_J) \mathrm{d}y_J \tag{A.55}$$

If W is the PCA transform, it diagonalizes Σ and the density on y is

$$\mathrm{p}(y) = \prod_{J=1}^{d} \mathrm{p}_J(y_J) \tag{A.56}$$

Substituting (A.56) into the equation for an element of QW (A.54) yields

$$\begin{aligned} (QW)_{KJ} &= \sum_{i_K} r_{Ki_K} \sum_{i_J} \int_{R_{Ji_J}} y_J \mathrm{p}_J(y_J) \mathrm{d}y_J \int_{R_{Ki_K}} \mathrm{p}_K(y_K) \mathrm{d}y_K \\ &= \sum_{i_K} \mathrm{p}_{Ki_K} r_{Ki_K} \int_{-\infty}^{\infty} y_J \mathrm{p}_J(y_J) \mathrm{d}y_J \\ &= \bar{y}_J \sum_{i_K} \mathrm{p}_{Ki_K} r_{Ki_K} \end{aligned} \tag{A.57}$$

where $\bar{y}_J$ is the mean of the J^{th} coefficient.

The r_{Ki_K} that minimize the coding distortion (A.41) are given by

$$r_{Ki_K} = \frac{\int_{R_{Ki_K}} y_K \mathrm{p}_K(y_K) \mathrm{d}y_K}{\int_{R_{Ki_K}} \mathrm{p}_K(y_K) \mathrm{d}y_K} \tag{A.58}$$

Subsituting (A.58) into (A.57) yields

$$\begin{aligned} (QW)_{KJ} &= \bar{y}_J \sum_{i_K} \int_{R_{Ki_K}} y_K \mathrm{p}_K(y_K) \mathrm{d}y_K \\ &= \bar{y}_J \int_{-\infty}^{\infty} y_K \mathrm{p}_K(y_K) \mathrm{d}y_K \\ &= \bar{y}_J \bar{y}_K \\ &= (QW)_{JK} \end{aligned} \tag{A.59}$$

Since QW is symmetric, the PCA transform is the optimal transform for coding Gaussian data. No high resolution approximations are required to achieve this result. In addition, the partition or encoder need not minimize mean squared coding error, so this result also holds when the quantizers are designed using entropy penalized cost functions.

Appendix B

Selecting Entropy for Adaptive PCA

Adaptive PCA models data as a collection of low-dimensional hyperplanes embedded in a high dimensional observation space and corrupted with spherical noise. We control model complexity by selecting a single global variance, σ^2, for the noise contribution. Choosing this noise variance is equivalent to selecting an entropy penalty. Our evaluations show that entropy-constrained adaptive PCA models have the potential to accurately model high-dimensional real-world data, even when training data is sparse.

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.

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

$$x = Ws + \mu + \epsilon \tag{B.1}$$

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 the value of ρ^2. However, the eigenvalues of real-world data typically do not reach some plateau. Consequently, the validation set method 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. The source textures are images of dense leaves, coarse cloth, marble, and paper. Figure B.1 shows the eigenvalues for these four source images. We calculated these eigenvalues by first dividing each source image into 9×9 blocks, then removing the data mean and performing singular value decomposition (SVD).

The artificial texture data was generated according to the PCA model (B.1) using the means, eigenvectors, and eigenvalues of the source 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 SVD on the image 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 floor ρ^2 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. Figure B.1 also shows the eigenvalues of the artificial data. The floor 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

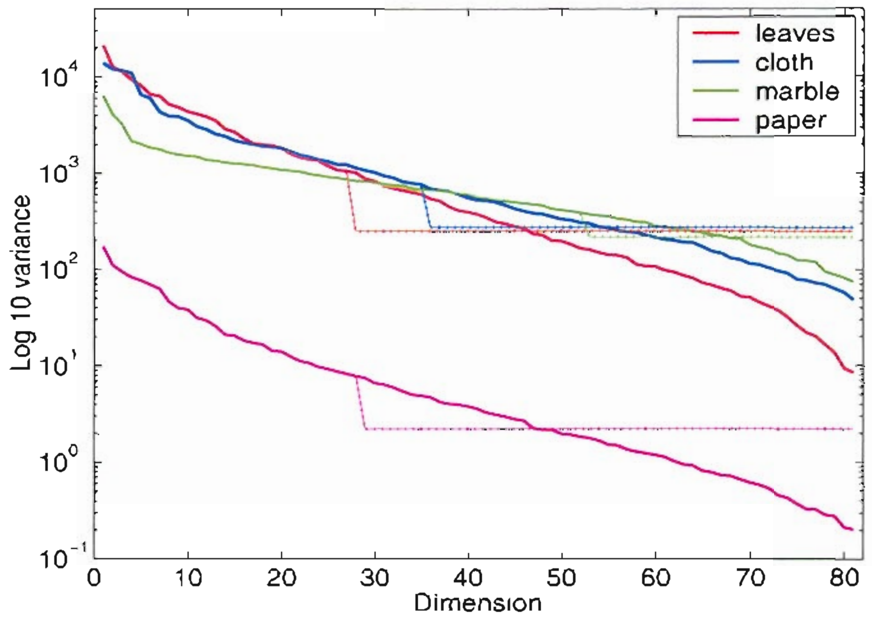
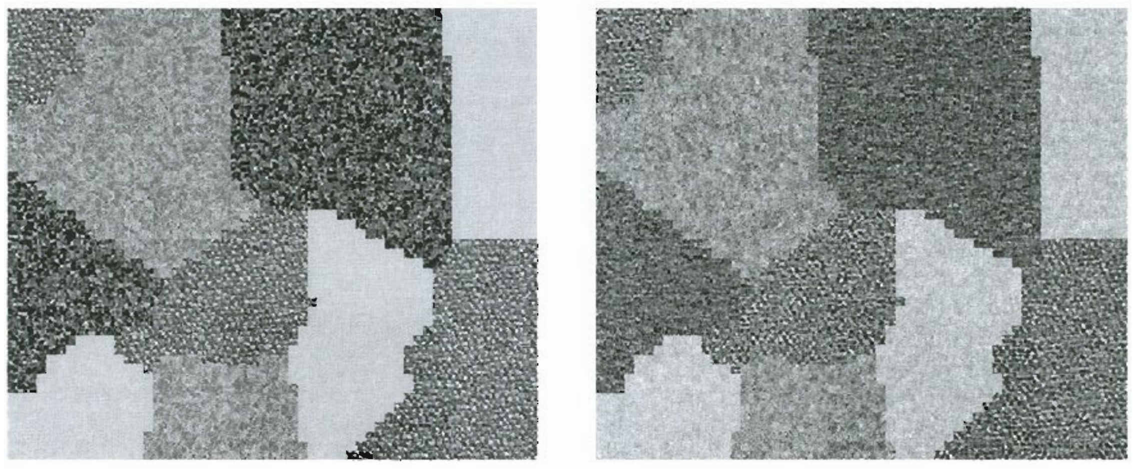


Figure B.1: Eigenvalues of real and artificial textures. Real (sampled) texture eigenvalues are shown by solid line. The leading eigenvalues of the artificial textures are the same as the real texture values. The trailing eigenvalues, shown by dotted line, have been replaced by a noise variance equal to their mean value.

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 B.2 shows the test data organized into maps for visualization.

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 B.3. 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



(a) Sampled (b) Artificial

Figure B.2: 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.

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.

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. However, evaluations performed on texture data indicate that models conform to the data structure when the noise variances is chosen at or slightly below the point where the model size is largest. Figure B.4, which contains a plot of component impurity H_p and abundance H_a for the real texture data, shows that classification ability is best at noise variances where the components are most abundant. Investigating the relationship between model size and component dimension for different choices of noise variance may lead to effective methods of entropy penalty selection.

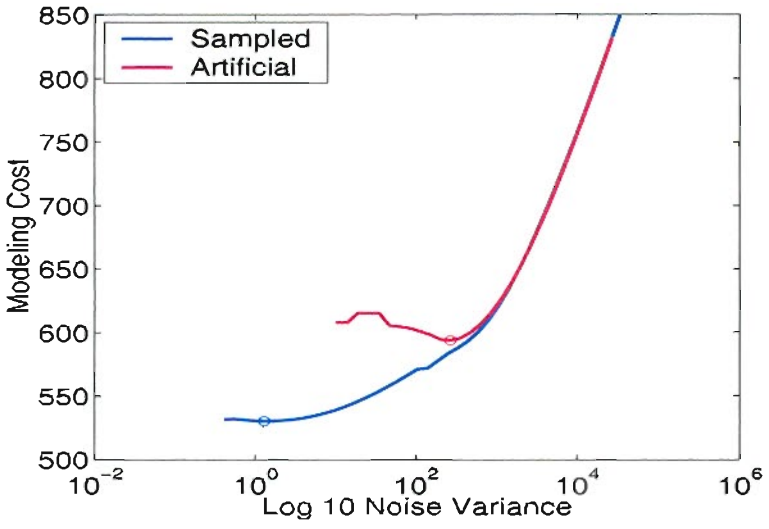


Figure B.3: Validation set cost for artificial and sampled texture data. Validation set cost plotted against $\log_{10}\sigma^2$ with artificial data cost in red and sampled data cost in blue. Circles indicate cost minima.

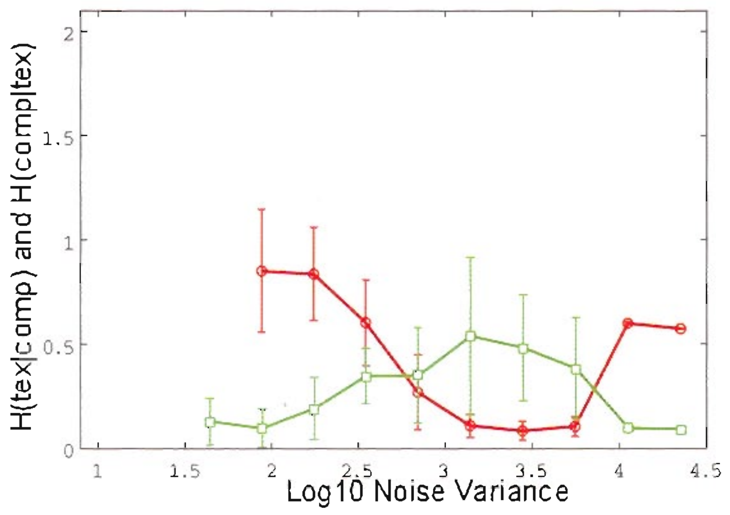


Figure B.4: Component impurity (red) and abundance (green) for real texture data. Circles indicate means and errorbars indicate standard deviation of ten models trained from different initializations.

Biographical Note

Cynthia Archer was born in Rockford, Illinois, U.S.A on July 20. 1960. In 1978, she entered the electrical engineering program at the University of Illinois, Urbana-Champaign. She completed her Bachelor of Science degree in 1982, graduating with honors. After graduation, she joined Raytheon's Submarine Signal Division working on sonar transducer research. In 1983, Cynthia joined General Telephone and Electric (GTE) working in their Government Systems Division. While at GTE, she developed custom electronic hardware, firmware, and software for secure satellite communications equipment. After ten years in industry, Cynthia returned to school part-time at Oregon Graduate Institute of Science and Technology(OGI) to earn a degree in computer science. After graduating with her Master of Science degree in 1998, she returned to school full time to pursue her doctoral degree in the area of machine learning with emphasis on adaptive signal processing and pattern recognition. In 2001, she was one of two recipients of the annual student achievement award given by OGI. She is the principal author on four published conference papers and two journal papers currently under review.