EM in High-Dimensional Spaces

Bruce A. Draper, *Member, IEEE*, Daniel L. Elliott, Jeremy Hayes, and Kyungim Baek

***Abstract*—This paper considers fitting a mixture of Gaussians model to high-dimensional data in scenarios where there are fewer data samples than feature dimensions. Issues that arise when using principal component analysis (PCA) to represent Gaussian distributions inside Expectation-Maximization (EM) are addressed, and a practical algorithm results. Unlike other algorithms that have been proposed, this algorithm does not try to compress the data to fit low-dimensional models. Instead, it models Gaussian distributions in the $(N-1)$-dimensional space spanned by the N data samples. We are able to show that this algorithm converges on data sets where low-dimensional techniques do not.**

***Index Terms*—Author, please supply your own keywords or send a blank e-mail to keywords@ieee.org to receive a list of suggested keywords.**

I. Introduction

Expectation-Maximization (EM) is a well-known technique for unsupervised clustering. Formally, EM is an algorithm for finding the maximum likelihood estimate (MLE) of the parameters of an underlying distribution from a data set with hidden or missing values. In practice, EM is used to fit mixtures of distributions to data sets, most often mixtures of Gaussians. The hidden variables reflect soft membership in the clusters. As a result, EM is used both to fit maximally likely distributions to data and to assign data samples to the most likely cluster.

This paper is about practical issues that arise when applying EM to a mixture of Gaussians in very high-dimensional feature spaces. In particular, we are interested in the scenario where there are far more feature dimensions than data samples. This scenario is common in appearance-based computer vision, where the samples are images and the features are pixels. Since even a modest-sized image has tens of thousands of pixels, the number of dimensions usually exceeds the number of samples. It should be noted, however, that this scenario also occurs in other domains with high-dimensional data, such as genomics.

The key issues are how to represent Gaussian distributions in high dimensions, how to estimate the probability of a point in a high-dimensional Gaussian distribution, and how to implement soft membership. The standard approach of representing a Gaussian distribution by its sample covariance matrix is not recommended when there are more features than samples, since the sample covariance matrix will be singular. A singular covariance matrix makes it difficult to estimate the probability of a point, since the covariance matrix cannot be inverted. Instead, we represent Gaussian distributions as the eigenvalues and eigenvectors of a principal component analysis (PCA) decomposition of the sample covariance matrix. This is not a novel idea (see Section III for related work). We find, however, that other methods in the literature do not converge on very high-dimensional data. We believe this is because of practical issues of exactly how high-dimensional Gaussian distributions are represented, how probabilities are estimated, and how soft membership is implemented.

Manuscript received August 1, 2003; revised July 30, 2004. This paper was recommended by Associate Editor Bir Bhanu.

B. A. Draper, D. L. Elliott, and J. Hayes are with the Computer Science Department, Colorado State University Fort Collins, CO 80523 USA (e-mail: draper@cs.colostate.edu).

K. Baek is with the Department of Bioengineering, Columbia University, New York, NY 10027 USA.

Digital Object Identifier 10.1109/TSMCB.2005.846670

We therefore explore these issues and propose a version of EM with PCA (called HD5) for fitting high-dimensional Gaussian mixture models to small data sets. The most surprising element of HD5 is that it retains eigenvectors associated with zero eigenvalues. We show that as a result of keeping these "extra" eigenvectors, HD5 converges on high-dimensional data sets where naive approaches and low-dimensional approaches (i.e., approaches based on compression) do not.

II. Expectation-Maximization

EM has become a standard technique for fitting mixtures of Gaussian models to data. Formally, EM is an algorithm for finding the MLE of the parameters of an underlying distribution from a data set with hidden or missing values. When clustering using a mixture model, the distribution usually is a mixture of Gaussians, and the hidden values represent the likelihoods that samples belong to clusters. EM consists of two steps: an expectation step and a maximization step. The expectation (E) step is to calculate $Q(\Theta, \Theta^{(i-1)}) = E[\log p(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{(i-1)}]$. Here, $\mathcal{X}$ is the observed data, $\mathcal{Y}$ is the hidden or missing data, and Θ is the model parameters. In other words, the E step calculates the probability that the data came from the current model, given the current estimates of the hidden parameters and the observed data. The maximization (M) step finds the values of Θ that maximize the probability calculated in the E step. This is defined as $\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(i-1)})$.

A full discussion of EM is beyond the scope of this paper. For a brief tutorial on EM, we recommend [1]; for the original source, see [2]. It is important to note for the purposes of this paper, however, that $\Theta = [\theta_1, \ldots, \theta_k]$ represents the parameters of k Gaussians. When the data has more samples than features, the parameters of the individual Gaussians $\theta_j = (\mu_j, \Sigma_j)$ are usually represented as their means and covariance matrices. Given a set of parameters θ_j, the probability of a sample x can then be written as

$$p(x|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{(d/2)}|\Sigma_j|^{(1/2)}} e^{-(1/2)(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)}.$$

The probability $p(x|\mu_j, \Sigma_j)$ is calculated in the E step to estimate the likelihood that sample x came from cluster j, and this likelihood is used in the M step to update the Gaussian parameters θ_j. Of course, the probability $p(x|\mu_j, \Sigma_j)$ cannot be computed according to the equation above when there are more features than samples because the sample covariance matrix Σ_j will be singular.

III. Related Work

In 1996, Ghahramani and Hinton derived a version of EM for clustering mixtures of reduced dimension factor analyzers [3]. Their explicit goal was to combine clustering with dimensionality reduction. Their technique, which is based on latent variable models, clustered data to maximize the likelihood of the data given a low-dimensional factor loading matrix. This technique was quickly exploited by Kambhatla and Leen for image compression [4] and by Frey *et al.* for face recognition [5]. More recently, it was used by Baek and Draper to suppress background pixels prior to object recognition [6].

The disadvantage to Ghahramani's approach is that factor analyzers separate the common variance from the unique variance and represent only the common variance. While this may be useful for some domains (see [6]), it is not a true representation of an underlying Gaussian distribution. In 1999, Tipping and Bishop built on Ghahramani's work to develop an EM algorithm for mixtures of reduced dimension principal component analyzers [7].

It is well known that for Gaussian data, PCA eigenvectors represent the axes of the underlying distribution, whereas the eigenvalues represent the variance along these axes (see [8], among others). As a result, Tipping and Bishop's algorithm fits a Gaussian mixture model to high-dimensional data. The problem is that Tipping and Bishop fit a *low-dimensional* Gaussian representation to high-dimensional data. They optimize for Q-dimensional principal component analyzers, where Q is much less than the ambient dimensionality D of the data (or the number of samples N). As such, their process model assumes that data samples are drawn from Q-dimensional Gaussian distributions in a D-dimensional space, which are then corrupted by high-dimensional white noise.

It has been hypothesized that in some domains (most notably human faces), only a few dimensions are needed [9]. If so, Tipping and Bishop's PCA analyzers should be well suited to those domains. In other domains, however, a few dimensions are not enough, and eigenvectors associated with very small eigenvalues may be significant. In these cases, the difference between a low- and a high-dimensional Gaussian representation is significant, and there is no guarantee that algorithms for low-dimensional Gaussians will converge on high-dimensional data.

In general, N data samples will inhabit at most an $N-1$-dimensional subspace of the D ambient dimensions. The goal of this paper is to fit $N-1$-dimensional Gaussians to every cluster. Other researchers fit high-dimensional models by other means. Sakuma and Kobayashi propose a heuristic kernel method for fitting high-dimensional mixtures of Gaussians to data [10], although this method lacks the probabilistic rigor of EM. Lu *et al.* fit support vector machine models under similar assumptions of small sample sizes but assume the training data is already labeled [11]. Chan *et al.* fit a mixture of low-dimensional independent component analyzers to data [12].

IV. EM With PCA

The goal of this paper is to use EM to fit a mixture of high-dimensional Gaussian distributions to small data sets. The basic idea is to use the eigenvalues and eigenvectors from PCA decompositions of weighted sample covariance matrices to represent high-dimensional Gaussian distributions. An abstract (pseudo-code) description of the algorithm is shown in Fig. 1.

We assume the data contains N samples, each of which is a point in a D-dimensional space, where D is the number of ambient feature dimensions. We assume that $N_c < D$, $c \in \{1 \dots C\}$, although this algorithm may also be useful and will run properly in situations where N_c is larger than D, but the number of points in any given cluster is less than D.

Given N samples in a D-dimensional space, the data cannot span more than an $N-1$ dimensions of the original space. This implies that we cannot fit Gaussian models represented by more than $N-1$ dimensions to the data.

To relate Fig. 1 to standard EM, steps 1 and 2 bootstrap the process by generating an initial estimate for Θ and $p(x^{(i)}|\theta_j)$. The initial covariance is assumed to be spherical. Steps 3 and 4 are the M steps; step 3 maximizes the means of the clusters, and step 4 computes the eigenvectors and eigenvalues that represent the covariance structure. Step 5 is the E step: It calculates the probabilities of the data samples given the clusters.

The algorithm as presented above calculates the means and eigenvalues/eigenvectors of the Gaussian distributions that maximize the likelihood of observing the data. If the goal is also to assign data samples to clusters, the maximum likelihood assignment is

$$\forall\, l \neq j, p(x^{(i)}|\theta_j) > p(x^{(i)}|\theta_l) \rightarrow x^{(i)} \in \mathcal{X}^{(j)}. \tag{2}$$

1) Randomly initialize centers.
2) Calculate initial probabilities. This is done by calculating $p(x^{(i)}|\theta_j) = \frac{1}{\|x^{(i)}-\mu_j\|}$, which is the inverse euclidean distance. The probabilities are then normalized across clusters, so that $\sum_j p(x^{(i)}|\theta_j) = 1$.
3) Update the locations of the cluster centers through a weighted average:
$$\mu_j = \frac{\sum_i p(x^{(i)}|\theta_j)x^{(i)}}{\sum_i p(x^{(i)}|\theta_j)}. \tag{1}$$
4) Perform PCA on weighted data ($\mathcal{X}^{(c)} = [p(x^{(1)}|\theta_j)(x^{(1)}-\mu_j)|\dots|p(x^{(n)}|\theta_j)(x^{(n)}-\mu_j)]$).
5) Compute $p(x^{(i)}|\theta_j)$ using equation 3 below, and then normalize.
6) Iterate over steps three through five until convergence.

Fig. 1. EM with PCA.

As discussed in the introduction, there are three issues that arise when using PCA to fit high-dimensional Gaussian mixture models to data: 1) how to represent high-dimensional Gaussian distributions; 2) how to approximate the value of a sample given a distribution; and 3) how to maximize distribution parameters given soft membership (likelihood) values. Section IV-A address these issues in the context of EM with PCA.

A. Soft Membership and Weighted PCA

PCA extracts the eigenvectors and eigenvalues of a sample covariance matrix. It is well known that when data samples are drawn from a Gaussian distribution, the PCA eigenvectors are an MLE of the principle axes of the Gaussian distribution, whereas the eigenvalues represent the variance along these axes [8]. As a result, PCA is a method for estimating the covariance parameters of a Gaussian distribution from a set of data points where the PCA eigenvectors and eigenvalues are a representation of that maximally likely distribution.

Before describing probabilistically weighted PCA, we should review the mechanics of how standard PCA fits Gaussian distributions to unweighted data. Let $\mathcal{X}$ be a set of (unweighted) samples, and let $\overline{x}$ be the average sample in $\mathcal{X}$. Define $\hat{X}$ to be the mean-subtracted data matrix, such that the ith column of $\hat{X}$ is $x^{(i)} - \overline{x}$. Then, PCA is the singular value decomposition of the sample covariance matrix $\hat{X}\hat{X}^T$.

To compute a probabilistically weighted PCA for cluster θ_j, the data matrix $\hat{X}$ must be adjusted. Define $\overline{\mathcal{X}^{(j)}}$ to be the weighted average of the samples, where $p(x^{(i)}|\theta_j)$ is the weight of sample i. Then, define $\hat{X}^{(j)}$ to be the weighted mean-subtracted data matrix whose ith column is $p(x^{(i)}|\theta_j)(x^{(i)} - \overline{x}^{(j)})$. Now, the SVD of the matrix $\hat{X}^{(j)}\hat{X}^{(j)^T}$ provides the probabilistically weighted principal axes and variances. (Note that this corresponds to steps #3 and #4 of Fig. 1.)

B. Representing High-Dimensional Gaussians

Again, this paper is concerned with using PCA to represent Gaussian distributions with covariance inside an EM algorithm; the key representational issue of which revolves around how many dimensions to keep for each cluster. Low-dimensional techniques (such as the Mixture of Principal Component Analyzers) assume that the underlying Gaussians

TABLE I

RECOGNITION ACCURACIES (PERCENTAGES) FOR FOUR VERSIONS OF EM WITH PCA ON NINE SYNTHETIC DATA SETS. THE FOUR VARIANTS OF EM WITH PCA DIFFER IN TERMS OF THE NUMBER OF EIGENVECTORS USED TO REPRESENT EACH CLUSTER, AND THE EQUATION USED TO APPROXIMATE $p(x^{(i)}|\theta_j)$, THE PROBABILITY OF A SAMPLE GIVEN A CLUSTER. LOW-DIMENSIONAL (LD) VERSIONS REPRESENT CLUSTERS USING A FIXED NUMBER OF EIGENVECTORS Q; LD ALGORITHMS ARE TESTED WITH FIVE, 30, AND 60 EIGENVECTORS, RESPECTIVELY. HIGH-DIMENSIONAL VERSIONS (HD) REPRESENT CLUSTERS WITH $N-1$ EIGENVECTORS (IN THE CASE OF HD5) OR THE SET OF ALL EIGENVECTORS ASSOCIATED WITH NONZERO EIGENVALUES (IN THE CASE OF HD3). LD3 AND HD3 APPROXIMATE $p(x^{(i)}|\theta_j)$ USING (3), WHEREAS LD4 APPROXIMATE $p(x^{(i)}|\theta_j)$ USING (4), AND HD5 APPROXIMATES IT WITH (5). THE DATA SETS DIFFER ACCORDING TO THE NUMBER OF CLUSTERS K, WHICH IS EITHER TWO, FIVE, OR TEN, AND THE DISTRIBUTION OF DISTANCES BETWEEN THE CLUSTER MEANS, WHICH HAS A MEAN OF 5.0 (EASY), 2.5 (MODERATE), OR 0.0 (HARD)

Data Set	Versions of EM with PCA							
(K, Δ-mean)	HD5	HD3	LD3 (Q=5)	LD3 (Q=30)	LD3 (Q=60)	LD4 (Q=5)	LD4 (Q=30)	LD4 (Q=60)
K=2, Δ-mean=5.0	**89.7(±12.1)**	50(±0.0)	66.3(±11.8)	60.0(±7.3)	50.0(±0.0)	83.5(±19.5)	80.7(±25.0)	50.0(±0.0)
K=5, Δ-mean=5.0	**79.7(±13.5)**	20.0(±0.0)	20.0(±0.0)	23.6(±11.3)	20.0(±0.0)	20.0(±0.0)	28.0(±25.3)	20.0(±0.0)
K=10, Δ-mean=5.0	**79.3(±5.2)**	10.0(±0.0)	10.0(±0.0)	10.0(±0.0)	10.0(±0.0)	30.1(±32.5)	10.0(±0.0)	10.0(±0.0)
K=2, Δ-mean=2.5	72.9(±20.7)	50.0(±0.0)	75.7(±8.5)	58.7(±7.6)	50.0(±0.0)	**85.0(±17.1)**	77.8(±20.5)	50.0(±0.0)
K=5, Δ-mean=2.5	**78.5(±9.3)**	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)
K=10, Δ-mean=2.5	**76.0(±5.4)**	10.0(±0.0)	10.0(±0.0)	10.0(±0.0)	10.0(±0.0)	59.1(±33.9)	10.0(±0.0)	10.0(±0.0)
K=2, Δ-mean=0.0	52.3(±2.3)	50.0(±0.0)	**64.5(±9.1)**	55.2(±5.6)	50.0(±0.0)	53.5(±4.1)	52.0(±5.5)	50.0(±0.0)
K=5, Δ-mean=0.0	28.4(±2.4)	20.0(±0.0)	**33.7(±2.7)**	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)	20.0(±0.0)
K=10, Δ-mean=0.0	**19.2(±1.9)**	10.0(±0.0)	10.7(±02.1)	10.0(±0.0)	10.0(±0.0)	10.0(±0.0)	10.0(±0.0)	10.0(±0.0)

can be described in Q dimensions $Q \ll (N-1)$, and they compute just Q eigenvectors for every cluster. In essence, any variance outside of the first Q dimensions is modeled as white noise. Unfortunately, estimating the value of Q is not easy [13], and since each cluster is generated by a different Gaussian process, there is no reason to believe that a single value of Q exists for all clusters. The other possibility advocated here is to keep all $N-1$ eigenvectors. In this way, we model each Gaussian in as many dimensions as the data will support.

The dimensionality of a cluster varies during clustering. When clustering with hard assignments, every sample is assigned to one cluster. As a result, every cluster has a data-dependent number of samples and therefore a data-dependent number of nonzero eigenvalues. In a soft assignment algorithm like EM, this problem goes away in theory but not in practice. In theory, $p(x^{(i)}|\theta_j) \neq 0$ for all samples and clusters since Gaussian distributions have infinite tails. Therefore, every cluster spans the same number of dimensions. In high dimensions, however, the probabilities $p(x^{(i)}|\theta_j)$ become so small that they are within round-off of zero. Weights with zero value lead to a data-dependent number of nonzero eigenvalues, just as in the hard assignment case.

The solution is to represent every Gaussian with $N-1$ eigenvectors, *even if some of those eigenvectors have zero eigenvalues.* The underlying model is that there is a minimum amount of variance ϵ in all $N-1$ dimensions and that eigenvalues smaller than ϵ are an artifact of the sample size.

C. Probability Function

The final issue is how to estimate the probability of a data sample given a high-dimensional Gaussian distribution, as required for step #5 of Fig. 1. Let R_j be the matrix of eigenvectors for cluster j, and let $\Lambda = [\lambda^{(1)}, \lambda^{(2)}, \ldots, \lambda^{(n)}]$ be the associated eigenvalues. Then, for any data sample $x^{(i)} \in \mathcal{X}$, $y^{(ij)} = R_j x^{(i)}$ is the projection of $x^{(i)}$ into the subspace of cluster j. If $y_i^{(ij)}$ is the ith element of $y^{(ij)}$, then the probability of sample $x^{(i)}$ being generated by cluster θ_j is

$$p(x^{(i)}|\theta_j) = \frac{e^{-(1/2)\sum_{k=1}^{Q}((y_k^{(ij)^2})/(\lambda^{(i)}))}}{(2\pi)^{(Q/2)}\prod_{k=1}^{Q}\lambda^{(k)(1/2)}} \tag{3}$$

where Q is the number of eigenvectors in R_j (and, therefore, the number of elements in $y^{(ij)}$). Note that this is just the standard probability equation for a decorrelated multivariate Gaussian distribution.

Equation (3) assumes, however, that all of the eigenvalues $\lambda^{(i)}$ are nonzero and that $y^{(ij)} = R_j x_i$ is a lossless projection (the latter is equivalent to assuming that $\|y^{(ji)}\| = \|x^{(i)}\|$). Algorithms that fit low-dimensional Gaussian distributions to data violate the second assumption since some dimensions are dropped, and in general, $\|y^{(ji)}\| < \|x^{(i)}\|$. In contrast, our approach keeps all the dimensions but violates the first assumption that all eigenvalues are nonzero. We therefore need to approximate $p(x^{(i)}|\theta_j)$ instead of computing it exactly.

The simplest approach is to use a low-dimensional Gaussian representation ($Q \ll (N-1)$) and ignore the dropped dimensions. As discussed by Moghaddam and Pentland [14], this greatly overestimates $p(x^{(i)}|\theta_j)$ since the dropped terms are all between zero and one and should be multiplied with the product of the first Q dimensions. To compensate, they include a second term

$$p(x^{(i)}|\theta_j) \approx \frac{e^{-(1/2)\sum_{k=1}^{Q}((y_k^{(ij)^2})/(\lambda^{(i)}))}}{(2\pi)^{(Q/2)}\prod_{k=1}^{Q}\lambda^{(k)(1/2)}} * \frac{e^{-(1/2\sigma)(\|x^{(i)}\|^2 - \|y^{(ij)}\|^2)}}{(2\pi\sigma)^{(((N-1)-Q)/2)}} \tag{4}$$

where σ is the average of the dropped eigenvalues, and N is the number of samples; therefore, $N-1$ is the maximum number of dimensions that the data can span.

This equation effectively assigns an average eigenvalue σ to all the dimensions in the null space of R_j and assumes that the energy projected into the null space of R_j is evenly divided among the dimensions of the null space. Equation (4) is used, among other places, by Tipping and Bishop [7].

Unfortunately, Moghaddam and Pentland's approximation still overestimates the probability $p(x^{(i)}|\theta_j)$. $\|x^{(i)} - y^{(ij)}\|$ is the magnitude of the projection of $x^{(i)}$ into the null space of R_j. Equation (4) implicitly assumes that the magnitude of the projection of x is evenly distributed. It approximates $p(x^{(i)}|\theta_j)$ by using the average null space magnitude for every null-space dimension. Of course, the probability

drops off rapidly with distance, and if the projections of x onto the various null-space dimensions are not equal to each other, the true probability $p(x^{(i)}|\theta_j)$ may be significantly lower than the one estimated by (4).

Computers are getting faster, however, and there is often no need to discard dimensions. We approximate $p(x^{(i)}|\theta_j)$ by keeping all $N-1$ dimensions and assigning a minimal eigenvalue of ϵ to every dimension. As a result, $\|y^{(ij)}\| = \|x^{(i)}\|$, and

$$p(x^{(i)}|\theta_j) \approx \frac{e^{-(1/2)\sum_{k=1}^{Q}((y_k^{(ij)2})/(\max(\epsilon,\lambda^{(i)})))}}{(2\pi)^{(Q/2)}\prod_{k=1}^{Q}\max(\epsilon,\lambda^{(k)(1/2)})}. \quad (5)$$

We show in Section V that the accuracy of the estimate of $p(x^{(i)}|\theta_j)$ matters. The underestimates caused by (3) (with $Q < (N-1)$) and (4) will prevent EM from converging on high-dimensional data sets.

V. Experiments on Synthetic Data

This paper presents an algorithm for fitting mixtures of high-dimensional Gaussian distributions to small data sets, under the assumptions that 1) the data will be more accurately modeled by high-dimensional Gaussians than low-dimensional Gaussians, and 2) (5) is a better approximation to $p(x^{(i)}|\theta_j)$ than (4) [or (3) with $Q < (N-1)$]. In this section, we test these assumptions by comparing the performance of various versions of EM with PCA on synthetic data.

In particular, we test four versions of EM:

- **High-dimensional Gaussians with** (5). This is the version we advocate. All clusters are represented by $N-1$ eigenvectors, and $p(x^{(i)}|\theta_j)$ is approximated using (5). This algorithm is labeled High Dimensional with (5) (HD5) in Table I.
- **High-dimensional Gaussians with** (3). Clusters are represented by up to $N-1$ eigenvectors, but eigenvectors associated with zero eigenvalues are discarded, and $p(x^{(i)}|\theta_j)$ is approximated using (3). This algorithm is labeled HD3.
- **Low-dimensional Gaussians with** (3). Clusters are represented by Q eigenvectors, $Q < (N-1)$, and $p(x^{(i)}|\theta_j)$ is approximated using (3) (effectively ignoring the discarded dimensions). This algorithm is labeled LD3.
- **Low-dimensional Gaussians with** (4). Clusters are represented by Q eigenvectors, $Q < (N-1)$, and $p(x^{(i)}|\theta_j)$ is approximated using (4) to compensate for the discarded dimensions. This algorithm is labeled LD4.

All four versions of EM are tested on synthetic data sets with 500 dimensions. The data sets are generated by randomly drawing a small number of samples from 500 dimensional Gaussian processes.

The data sets are created by first randomly generating Gaussian processes from a set of hyper-priors. For the experiments reported in this paper, we generated either two, five, or ten Gaussian processes per data set and sampled each process 50 times, for a total of 100, 250, or 500 samples per data set. The hyper-priors specify that the standard deviation of the standard deviations of the principal axes is one. For the easiest data sets, the distribution of distances between process means has a mean of 5.0 and a standard deviation of 1.0; for the moderate data sets, the distribution of means has a mean signed distance of 2.5; and for the hardest data sets (shown in the bottom three lines of Table I), the mean signed distance between cluster centers is zero.

Mixtures of Gaussians are evaluated by their classification accuracy. Clustering algorithms are run until convergence, and then, every sample is assigned to its most likely cluster. Individual Gaussians are labeled according to the process that generated the plurality of their samples, and the accuracy of the mixture is measured as the percent of samples assigned to their correct clusters.

Table I shows the classification accuracy of all four variants of EM with PCA over nine synthetic problems. Since the low-dimensional versions of EM with PCA are parameterized by the number of subspace dimensions Q, they are tested with three different values of Q (five, 30, 60) on each problem. In addition, since EM is nondeterministic (due to the random selection of initial cluster centers), each test is repeated ten times. Table I shows both the mean recognition accuracy and the standard deviations of the accuracies across runs. On some data sets, many versions of EM consistently converge on solutions where a single cluster accounts for all of the data. In these cases, the recognition accuracy is $(1/K)$, and the standard deviation of the recognition rate is zero. The maximum recognition rates for each data set are shown in boldface.

In general, HD5 outperforms other versions of EM with PCA; it has the highest recognition rate in six of nine trials. In addition, in general, the recognition rate drops with the distance between the cluster centers. This makes sense; it is difficult to reliably separate points drawn from distributions with very similar means.

More interestingly, HD5 always outperforms the other forms of EM with PCA when the number of clusters is large (in this case, ten). When the number of clusters is small, the low-dimensional techniques also perform well, assuming that they are restricted to a very small number of dimensions (e.g., five). Apparently, it is possible to separate points from two clusters by looking at just a few of the most widely varying dimensions, but this strategy seems to break down as the number of clusters increases or as the number of subspace dimensions increases.

These experiments suggest that HD5 is the best option for fitting mixtures of high-dimensional Gaussians to small data sets. This is the version of the algorithm that represents every cluster with $(N-1)$ eigenvectors, even if some of the eigenvalues are zero, and estimates $p(x^{(i)}|\theta_j)$ using (5). In essence, it assumes a minimum variation of ϵ along all $N-1$ dimensions.

VI. Explanation of Experimental Results

Why does HD5 empirically outperform the other three versions of EM with PCA? To explain why this happens, we created the two-dimensional example in the top panel of Fig. 2. This example shows samples drawn from two underlying processes, labeled with x's and y's. Ideally, EM should converge on a solution where the x's are in one cluster and the y's in another. The cluster of y's has no variance in the horizontal dimension, however. It therefore has only one eigenvector with a nonzero eigenvalue, and this eigenvector runs vertically through the cluster.

The HD3 algorithm will begin to converge on the correct solution, putting all the x's in one category and all the y's in the other. As it does this, however, the y cluster is left with only one nonzero eigenvalue. When (3) estimates the probability that an x sample is in cluster y, it does so by projecting the x sample onto the horizontal eigenvector of the y cluster and, therefore, grossly overestimates this probability, in effect adding the x sample into the y cluster.

In this example, the low-dimensional algorithms would also be limited to representing each cluster with a single eigenvector, leading to similar confusion. In effect, the problem occurs when high-dimensional representations collapse to smaller numbers of dimensions because of eigenvectors with zero eigenvalues or when critical information is in dimensions not included in low-dimensional representations.

This simple, two-dimensional example has at least three differences from the high-dimensional cases encountered in practice. First, the dimensionality of the cluster description only becomes smaller in the two-dimensional case because we artificially constructed a data set with no variance in the horizontal dimension. When the number of samples

exceeds the number of ambient data dimensions, however, the dimensionality of a cluster is limited by the number of samples in the cluster, and as clusters converge on subsets of samples, their dimensionality is inevitably reduced. Second, since EM uses soft assignment, every sample should always have a nonzero probability of belonging to any cluster; therefore, in theory, the data dimensionality of every cluster is $N - 1$. In practice, however, the probabilities in high-dimensional spaces are so small that they round to zero;[1] therefore, samples are effectively eliminated from clusters, and the dimensionality of those clusters drops. Third and finally, Moghaddam and Pentland's approximation for $p(x^{(i)}|\theta_j)$ (4) would have solved this example because there is only one null space dimension. As the dimensionality increases, however, the difference between the projection onto the average null space dimension and the projection onto a particular null space dimension increases, and (4) overestimates the probability of a given principal component analyzer generating a data point that was actually generated by a different PCA. As a result, the phenomenon illustrated in Fig. 2 almost always occurs in high-dimensional data.

The HD5 version of EM with PCA solves this problem by modeling every cluster with $N - 1$ eigenvectors, including eigenvectors associated with zero eigenvalues. This ensures that no sample will ever lie in the null space of any other cluster. The probabilities $p(x^{(i)}|\theta_j)$ are then estimated using (5).

To further illustrate our point that keeping too few subspace dimensions results in inferior, if not entirely useless, classifications, we performed two additional experiments. The first experiment relates the number of subspace dimensions kept to the classification accuracy. The second experiment projects the data samples into the inherent and null spaces of each cluster and measures the relative magnitudes of the two projections. Both experiments use data generated using the MPPCA generative process. Therefore, the number of inherent dimensions of the generative process is selected at random from between one and 499. Each data set has an ambient dimensionality of 500, 500 data points, and ten clusters. The means are chosen from a uniform distribution with a range of ± 5. The covariance of each cluster is also selected from a uniform distribution with a range of $[0, 1]$.

The results from the first of these experiments are shown in Fig. 3. The values on the x-axis represent the number of subspace dimensions kept while the y-axis represents classification accuracy. These results were computed by running the *LD3* version of our algorithm with increasing values of Q on the same 30 data sets. As can be seen, retaining a large number of subspace dimensions has a beneficial effect on classification accuracy. It should be noted that classification accuracy reaches its highest point before $Q = 499$. Most likely, this is a result of the random starting locations between runs of each version. This suggests that keeping all $N - 1$ subspace dimensions is not entirely necessary. However, choosing a lower value can be risky since the choice will be arbitrary and domain-specific. Therefore, we still recommend keeping all $N - 1$ dimensions.

The second experiment projected data samples into the null and inherent spaces of each cluster. Table II shows the average magnitude over all samples and clusters of these projections, measured as a percentage of the original sample length. The table is divided into intra- and inter-cluster projections, depending on whether the sample is being projected into the most likely cluster (intra-cluster) or another cluster (inter-cluster). The values in the second column are also known as the reconstruction errors and may be computed by $\|x - U^T x\|$ [15]. In this experiment, the inherent space was taken to the be the $N_j - 1$ dominant eigenvectors, where N_j is the number of data points belonging to cluster j.

[1]It does not matter how much precision is used here; very small probabilities lead to very small eigenvalues, which in turn create even smaller probabilities until eventually, both the eigenvalues and the probabilities round to zero.

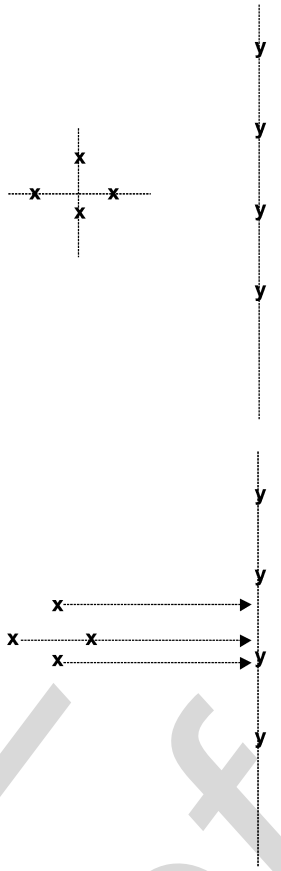

Fig. 2. Two-dimensional scenario in which EM may not converge. The top frame shows two sets of points (x's and y's), with dashed lines showing the eigenvectors when they are correctly clustered. The lower frame shows what happens when the y cluster has only one nonzero eigenvalue: the x's project near the middle of the single eigenvector associated with the y cluster, causing the probability that the x's are in the y cluster to be greatly overestimated.

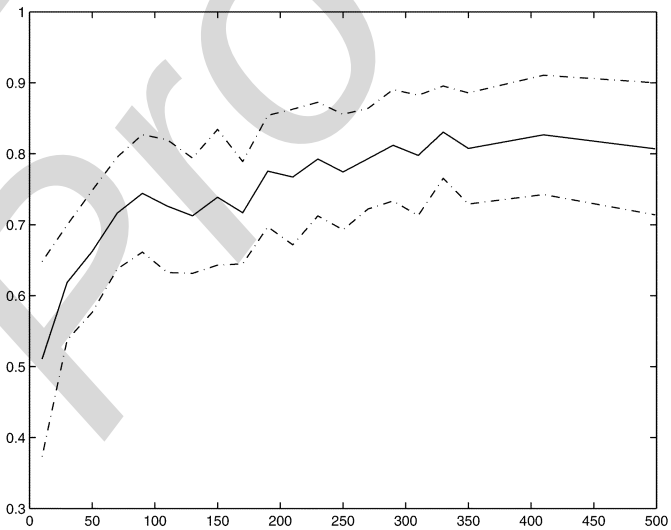

Fig. 3. Plot comparing number of dimensions kept (x-axis) to the classification accuracy (y-axis).

As can be seen across the top row of Table II, nearly all of each sample's magnitude lies in the inherent space of its most likely cluster, with virtually no variance. This suggests that the probability of a sample belonging to a cluster can be accurately estimated by simply using (3) when the true likelihood of membership is high.

TABLE II
RESULTS OF THE CALCULATIONS OF THE PERCENT MAGNITUDE THAT FALLS INTO THE INHERENT AND NULL SPACES OF EACH CLUSTER FOR INTER- AND INTRA-CLUSTER PROJECTIONS

	Inherent Space	Null Space
Intra-cluster	$0.99999 \pm 9.42 \times 10^{-5}$	$5.53677 \times 10^{-6} \pm 9.42 \times 10^{-5}$
Inter-cluster	0.30790 ± 0.13	0.69313 ± 0.13

Unfortunately, the bottom row shows that, on average, well over half of a sample's magnitude lies in the null space of the other clusters. As a result, the likelihood of a sample belonging to one of these clusters may be grossly overestimated, leading to assignment errors and/or instability. This is exactly the type of error illustrated in Fig. 2.

The data sets used to generate Table II neglect one pathological case in which the means of the underlying Gaussians differ, but their covariance matrices are the same. In this rare case, all of the techniques considered above should work about equally well, although *HD5* would be more computationally expensive than the other options.

VII. QUALITATIVE EXPERIMENTS ON REAL IMAGES

As computer vision researchers, we are suspicious when results are only presented on synthetic data. It may be that an algorithm works well on strictly Gaussian data but not on real data, which tends to come from messier distributions. In this case, the history was the opposite. Based on the recommendations in [14], we first implemented EM with Q dimensions per Gaussian, estimating probabilities according to (4). When this did not converge on real image sets, we began exploring other options.

To demonstrate that HD5 converges on real images, we ran it on a data set of 119 images: 60 images of cat faces and 59 images of dog faces. Each image consists of 64×64 pixels, resulting in 4096 dimensions. A run using four clusters is shown in Figs. 4–6. Fig. 4 shows the images belonging to each cluster. Fig. 5 shows the mean value (μ_j) for each cluster, whereas Fig. 6 shows the first five eigenvectors for each cluster. The eigenvectors are sorted in decreasing order with respect to their eigenvalues. Therefore, the images on the left in Fig. 6 show the eigenvectors with the greatest variance in their respective dimensions.

The interpretation of exactly what is being clustered is subjective. Looking at the images in Fig. 4, we hypothesize that clusters one and two are clustering dark and light cats, respectively. Clusters three and four appear to cluster dark and light dogs. This conjecture is supported by data displayed in Fig. 5. Looking at the mean images suggests that the clusters are separating data by animal and level of brightness. The eigenvector data shown in Fig. 6 is more difficult to interpret. For example, the eigenvectors associated with the first cluster (dark cats) shows that the primary source of variance comes from the color of the background. The second eigenvector appears to account for the varying darkness of the ears of the cat as well as whether or not the nose area has a patch of bright fur. The third eigenvector appears to be primarily accounting for the presence of a patch of bright fur near the nose.

VIII. DISCUSSION

We have presented an algorithm for fitting a mixture of Gaussians model to high-dimensional data using EM. Like previous algorithms, we represent Gaussian clusters in high dimensions through the eigenvectors and eigenvalues of their PCA decomposition. Unlike previous algorithms, we do not compress the data to find low-dimensional representations of clusters. Instead, we represent clusters in the $(N-1)$ dimensional subspace spanned by the N data samples.

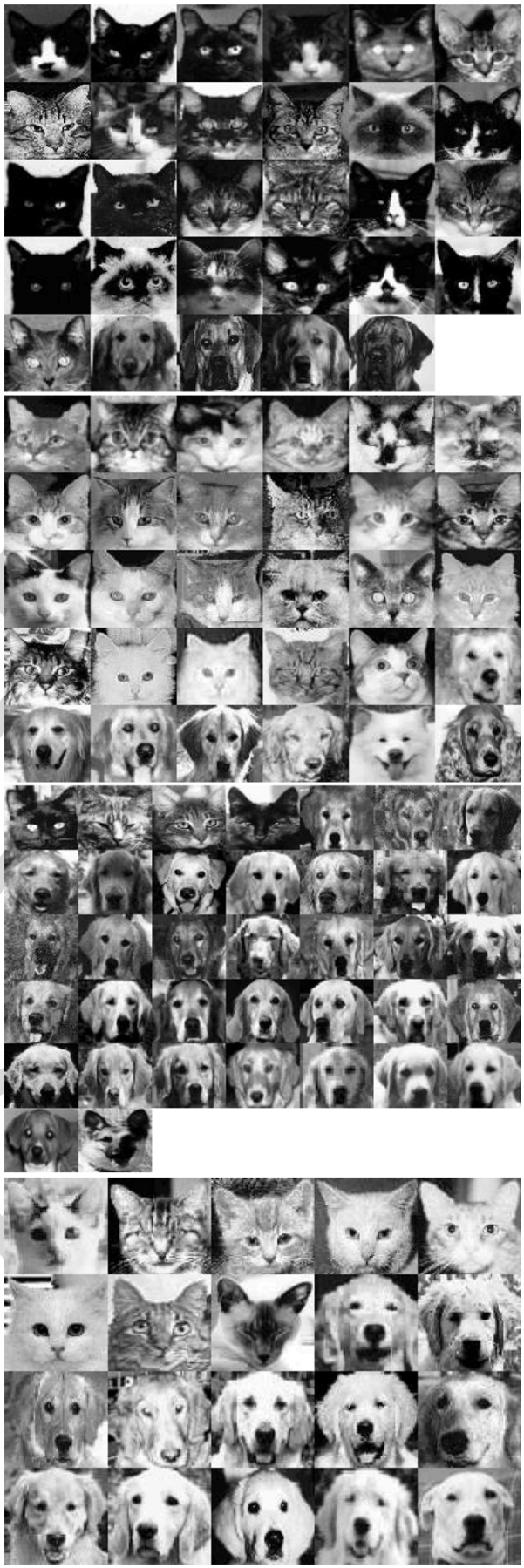

Fig. 4. Images showing cluster membership for a run of HD5 with four clusters.

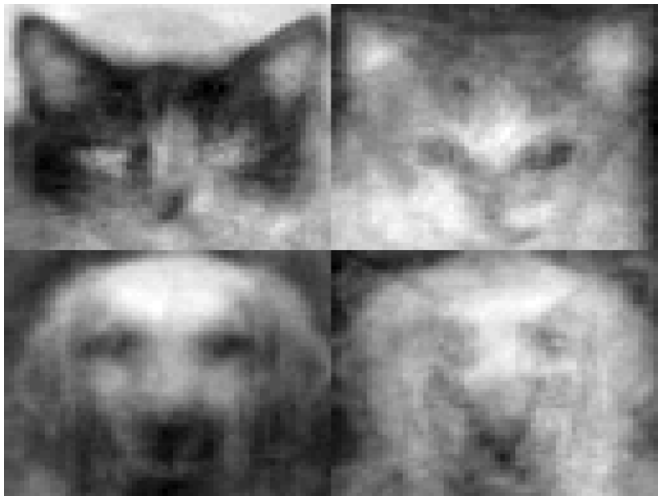

Fig. 5. Cluster centers (μ_j) for the clusters shown in Fig. 4.

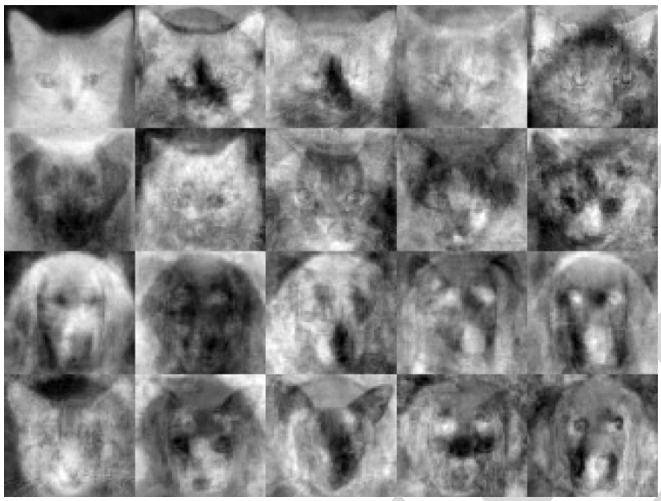

Fig. 6. First five eigenvalues for each of the clusters in Fig. 4.

Although the basic idea of using PCA to fit Gaussian distributions to small data sets is not surprising, we find that EM will only perform well if all clusters are represented by $N-1$ eigenvectors, *even if some of those eigenvectors are associated with zero eigenvalues*. We therefore keep the complete set of eigenvectors for every cluster and estimate $p(x^{(i)}|\theta_j)$ by assigning a minimal value of ϵ to every eigenvalue, as shown in (5). The result is a stable, albeit computationally expensive, algorithm for clustering data in high-dimensional spaces. In addition to the experiments described here, this algorithm has been tested and shown to converge on as few as 90 samples with as many as 10 000 dimensions.

REFERENCES

[1] J. Bilmes. (1997) A Gentle Tutorial on the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. [Online]. Available: citeseer.nj.nec.com/article/bilmes98gentle.html

[2] A. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist. Soc.*, vol. 39, pp. 1–38, 1977.

[3] Z. Ghahramani and G. E. Hinton. (1996) The EM Algorithm for Mixtures of Factor Analyzers. [Online]. Available: citeseer.nj.nec.com/ghahramani97em.html

[4] N. Kambhatla and T. Leen, "Dimension reduction by local PCA," *Neural Comput.*, vol. 9, pp. 1493–1516, 1997.

[5] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*. Cambridge, MA: MIT Press, 1998.

[6] K. Baek and B. A. Draper, "Factor analysis for background suppression," in *Proc. Int. Conf. Pattern Recogn.*, 2002.

[7] M. E. Tipping and C. M. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Comput.*, vol. 11, no. 2, pp. 443–482, Feb. 1999.

[8] I. Jolliffe, *Principal Component Analysis*. New York: Springer, 2002.

[9] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.

[10] J. Sakuma and S. Kobayashi, "Non-parametric expectation-maximization for gaussian mixtures," in *Proc. Workshop Inform.-Based Induction Syst.*, 2002.

[11] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos, "Face recognition using kernel direct discriminant analysis algorithms," *IEEE Trans. Neural Networks*, vol. 14, no. 1, pp. 117–126, Jan. 2003.

[12] K. Chan, T.-W. Lee, and T. Sejnowski, "Variational learning of clusters of undercomplete nonsymmetric independent components," *J. Machine Learning Res.*, vol. 3, pp. 99–114, 2002.

[13] "Mass. INst. Technol. Media Lab. Perceptual Comput. Section Tech. Rep. 514,", Cambridge, MA, 2000.

[14] B. Moghaddam and A. Pentland. (1997, Jul.) Probabilistic visual learning for object representation. *IEEE Trans. Pattern Anal. Machine Intell.*. [Online], vol (7), pp. 696–710

[15] M. Kirby, *Geometric Data Analysis*. New York: Wiley, 2001.