Presented at the *IEEE Conference on Computer Vision and Pattern Recognition*, Ft. Collins, CO, June, 1999.

Estimating Mixture Models of Images and Inferring Spatial Transformations Using the EM Algorithm

Brendan J. Frey   Nebojsa Jojic

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

Abstract

Mixture modeling and clustering algorithms are effective, simple ways to represent images using a set of data centers. However, in situations where the images include background clutter and transformations such as translation, rotation, shearing and warping, these methods extract data centers that include clutter and represent different transformations of essentially the same data. Taking face images as an example, it would be more useful for the different clusters to represent different poses and expressions, instead of cluttered versions of different translations, scales and rotations. By including clutter and transformation as unobserved, latent variables in a mixture model, we obtain a new "transformed mixture of Gaussians", which is invariant to a specified set of transformations. We show how a linear-time EM algorithm can be used to fit this model by jointly estimating a mixture model for the data and inferring the transformation for each image. We show that this algorithm can jointly align images of a human head and learn different poses. We also find that the algorithm performs better than k-nearest neighbors and mixtures of Gaussians on handwritten digit recognition.

1 Introduction

We are interested in developing algorithms that can learn models of different types of object from unlabeled images that include background clutter and spatial transformations, such as translation, rotation, shearing and warping. For example, Fig. 1 shows 8×8 greyscale images of handwritten digits and 44×28 greyscale images of a person walking across a cluttered background while changing his pose. The original images of the handwritten digits had high contrast, so they were easily normalized for horizontal and vertical scale and translation. However, the digits were written with different amounts of shearing and this effect cannot be removed in a simple fashion. The images of the human head include background clutter that appears in multiple frames, so aligning the images is not a trivial task.

Figure 1: Images of handwritten digits, which have been written with different amounts of shearing and images of a head, which appears at different positions and has different poses.

We propose a general purpose statistical method that can jointly learn a mixture of appearances (*e.g.*, different poses of the head), suppress the background clutter and normalize for spatial transformations. Our method does *not* assume the data is ordered. In the case of time series (*e.g.*, video sequences), temporal coherence is obviously valuable for modeling the data [1–3]. In [4], we show how our technique can be

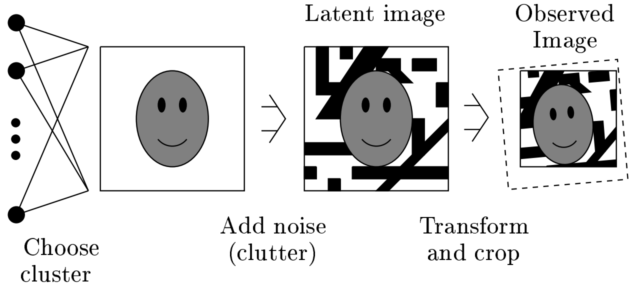

Figure 2: Generative process for a "transformed mixture of Gaussians" (TMG).

extended to take into account temporal coherence.

One approach to solving vision problems is to train a *recognition model* to accurately predict class membership from an input image or video sequence. This approach includes neighborhood-based methods such as "eigenfaces" [5, 6] and nonlinear regression techniques such as neural networks [7, 8].

In this paper, we take a Bayesian approach and train a *generative model* to accurately model the data from each class, including any known graphics that may enter the picture. Fig. 2 schematically shows a *transformed mixture of Gaussians* (TMG) that models the face as a mixture of images, models background clutter as independent Gaussian noise and models spatial transformations by a distribution over a discrete set of matrices that act on the vector of pixels.

Each stage in a TMG is probabilistic, so for a specific set of parameters (mixing proportions, cluster means and noise levels) there is a certain probability that the model will generate a given image. To generate an image, a cluster is randomly chosen according to the mixing proportions and a randomly chosen noise pattern is added to the cluster mean to obtain a *latent image*. Next, a transformation is selected at random and applied to the latent image. The result is cropped and a small amount of sensor noise is added to each pixel to obtain the final observed image.

The model can be trained using the EM algorithm, which adjusts the parameters to increase the likelihood of a set of data. For each input image, the learning procedure considers all transformations and clusters and adjusts the cluster means and noise models appropriately. In the E-Step, the responsibility (posterior probability) of each transformation - cluster pair is computed for each training case. In the M-Step, the cluster means, mixing proportions and noise levels are adjusted using the responsibilities.

The generative model can be used for pattern classification by training one model on each class of data (*e.g.*, one model for each person) in a training set and then classifying a test case by selecting the generative model that has the highest posterior probability given the test case. Each generative model $m \in \{1, \dots, M\}$ specifies a probability density $p(\mathbf{x}|m)$ for the vector of observed pixels $\mathbf{x}$. If model m has prior probability $P(m)$, then the Bayes-optimal decision for a uniform loss function is given by

$$\mathrm{argmax}_m P(m) p(\mathbf{x}|m). \tag{1}$$

In this framework, a new class of data can be added simply by training one extra generative model.

In the next section, we formally define the transformed mixture of Gaussians. Then, we describe how the model can be fit to a training set using the EM algorithm. After giving results on synthetic data, we show that the TMG can be used to extract translation invariant clusters from outdoor video sequences with cluttered backgrounds. We compare these clusters with clusters found using traditional mixture modeling and the first 10 principal components, or "eigenimages", of the data.

2 Transformed Mixtures of Gaussians (TMGs)

In a TMG, the C clusters are indexed by $c \in \{1, \dots, C\}$ and cluster c has mixing proportion $P(c) = \pi_c$. The probability density of the vector of pixel values $\mathbf{z}$ for the latent image corresponding to cluster c is

$$p(\mathbf{z}|c) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_c, \boldsymbol{\Phi}_c). \tag{2}$$

$\boldsymbol{\mu}_c$ is the mean of the latent image and $\boldsymbol{\Phi}_c$ is a diagonal covariance matrix that specifies the variability of each pixel in the latent image. The data we are interested in here is high-dimensional, so we use a diagonal covariance matrix instead of a full covariance matrix. In general, different clusters will represent different types of latent image and the corresponding noise variance maps will specify which regions are not well-modeled — *e.g.*, background clutter.

Although the true transformation variables are usually real-valued, real-valued latent variables introduce complicated integrals (instead of summations) into the EM learning algorithm. So, we will assume there is a fixed set of possible transformations and that this set is specified beforehand. Also, the algorithm is simplified by assuming that the vector of pixel values for the transformed image is obtained by multiplying the vector of pixel values for the latent image by a matrix. This permits a broad class of transformations, including translation, scale, in-plane rotation and out-of-plane rotation.

Let $\ell \in \{1, \ldots, L\}$ index the set of L transformations represented by matrices $\mathbf{\Gamma}_1, \ldots, \mathbf{\Gamma}_L$. Each transformation matrix also implements cropping simply by not specifying how to produce pixels that are not in the cropped image. For most transformations, each pixel in the observed image will depend on only a small number (say, 4) of pixels in the latent image. So, $\mathbf{\Gamma}_\ell$ is a sparse matrix with a number of columns equal to the number of pixels in the latent image and a number of rows equal to the number of pixels in the observed image.

The probability density of the vector of pixel values $\mathbf{x}$ for the image corresponding to transformation ℓ and latent image $\mathbf{z}$ is

$$p(\mathbf{x}|\ell, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{\Gamma}_\ell \mathbf{z}, \mathbf{\Psi}), \tag{3}$$

where $\mathbf{\Psi}$ is a diagonal covariance matrix that specifies the noise on the observed pixels.

The variances $\mathbf{\Phi}_c$ of the pixels in the latent image are quite different from the variances $\mathbf{\Psi}$ of the pixels in the observed image. $\mathbf{\Phi}_c$ models the noise in the pixel values for cluster c and this coheres to the latent image under transformations. In contrast, $\mathbf{\Psi}$ models noise in the observed pixels and this noise does not depend on the transformations.

The translation, scale and rotation corresponding to each ℓ are selected ahead of time so that the set of transformations effectively covers the range of possible transformations in the data and at the same time is small enough to keep the learning time reasonable. We also assume that a fixed prior probability $P(\ell) = p_\ell$ (usually uniform) is assigned to each possible transformation beforehand, although this could be learned from the training data.

The joint distribution over cluster index, transformation index, latent image and observed image is

$$\begin{aligned} p(\mathbf{x}, \ell, \mathbf{z}, c) &= p(\mathbf{x}|\ell, \mathbf{z}) P(\ell) p(\mathbf{z}|c) P(c) \\ &= \pi_c p_\ell \mathcal{N}(\mathbf{x}; \mathbf{\Gamma}_\ell \mathbf{z}, \mathbf{\Psi}) \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_c, \mathbf{\Phi}_c). \end{aligned} \tag{4}$$

In this generative model, the cluster index, transformation index and latent image are unobserved variables. Generating an image from a TMG consists of drawing a cluster index from $P(c)$, drawing a latent image from $p(\mathbf{z}|c)$, drawing a transformation index from $P(\ell)$ and then drawing an image from $p(\mathbf{x}|\ell, \mathbf{z})$.

The number of parameters in a TMG is roughly equal to the number of parameters in a standard mixture model with the same number of clusters. In Sec. 3, we show how the ML estimate of this model can be obtained using the EM algorithm.

For a given set of parameters, the model can be used to compute the likelihood for an image $\mathbf{x}$ by summing and integrating over the latent variables. It turns out that the latent image $\mathbf{z}$ can be integrated over in closed form:

$$\begin{aligned} p(\mathbf{x}, \ell, c) &= \int_{\mathbf{z}} d\mathbf{z} p(\mathbf{x}, \ell, \mathbf{z}, c) \\ &= \pi_c p_\ell \int_{\mathbf{z}} d\mathbf{z} \mathcal{N}(\mathbf{x}|\mathbf{\Gamma}_\ell \mathbf{z}, \mathbf{\Psi}) \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_c, \mathbf{\Phi}_c) \\ &= \pi_c p_\ell \mathcal{N}(\mathbf{x}; \mathbf{\Gamma}_\ell \boldsymbol{\mu}_c, \mathbf{\Gamma}_\ell \mathbf{\Phi}_c \mathbf{\Gamma}'_\ell + \mathbf{\Psi}), \end{aligned} \tag{5}$$

where " $'$ " indicates transpose. Each cluster c and transformation ℓ has a corresponding mean image $\mathbf{\Gamma}_\ell \boldsymbol{\mu}_c$ and covariance matrix $\mathbf{\Gamma}_\ell \mathbf{\Phi}_c \mathbf{\Gamma}'_\ell + \mathbf{\Psi}$. The probability density of $\mathbf{x}$ is

$$p(\mathbf{x}) = \sum_{c=1}^{C} \sum_{\ell=1}^{L} \pi_c p_\ell \mathcal{N}(\mathbf{x}; \mathbf{\Gamma}_\ell \boldsymbol{\mu}_c, \mathbf{\Gamma}_\ell \mathbf{\Phi}_c \mathbf{\Gamma}'_\ell + \mathbf{\Psi}). \tag{6}$$

2.1 Computing the Likelihood in Linear Time

If the observed image has N pixels, it turns out the computation of the likelihood from (6), *including* the computation of the inverse covariance matrices $(\mathbf{\Gamma}_\ell \mathbf{\Phi}_c \mathbf{\Gamma}'_\ell + \mathbf{\Psi})^{-1}$, uses $\mathcal{O}(CLN)$ scalar operations.

The computation is linear in N due to the sparseness of the transformation matrices. First, notice that the number of pixels in the latent image need only be $\mathcal{O}(N)$. For a translation by an integer numbers of pixels, $\mathbf{\Gamma}_\ell$ will have a single 1 in each row. For a rotation, $\mathbf{\Gamma}_\ell$ will have at most six 1's in each row. For a scaling over 1, $\mathbf{\Gamma}_\ell$ will have a single 1 in each row and for a scaling between 0.5 and 1, $\mathbf{\Gamma}_\ell$ will have at most four 1's in each row. So, each transformation matrix has $\mathcal{O}(N)$ nonzero elements.

Since $\mathbf{\Phi}_c$ and $\mathbf{\Psi}$ are diagonal, $\mathbf{\Gamma}_\ell \mathbf{\Phi}_c \mathbf{\Gamma}'_\ell + \mathbf{\Psi}$ will have $\mathcal{O}(N)$ nonzero elements and can be inverted using $\mathcal{O}(N)$ scalar operations. The inverse will also have $\mathcal{O}(N)$ nonzero elements, so the computation of $(\mathbf{x} - \mathbf{\Gamma}_\ell \boldsymbol{\mu}_c)'(\mathbf{\Gamma}_\ell \mathbf{\Phi}_c \mathbf{\Gamma}'_\ell + \mathbf{\Psi})^{-1}(\mathbf{x} - \mathbf{\Gamma}_\ell \boldsymbol{\mu}_c)$ uses $\mathcal{O}(N)$ scalar operations.

For one class and a single transformation, the TMG may remind some readers of the factor analysis model [9]. However, in factor analysis, $\mathbf{\Gamma}$ is dense, so the likelihood computation requires $\mathcal{O}(N^3)$ scalar operations.

2.2 Selecting the Number of Transformations

Although the number of scalar operations used in the likelihood computation is linear in L, it should be kept in mind that attempting to use an exhaustive set of transformations will cause L to grow polynomially with N. For n_h horizontal translations, n_v vertical translations, n_r rotations and n_s scalings, $L = n_h n_v n_r n_s$.

A TMG is meant to deal with a *moderate* number of transformations. Another algorithm (*e.g.*, a Kalman filter based face tracking algorithm) should be used to coarsely align the data before a TMG is used to extract detailed transformation-invariant clusters.

3 The EM Algorithm for Transformed Mixtures of Gaussians

In this section, we describe an iterative expectation maximization (EM) algorithm for estimating the maximum likelihood parameters of a TMG. Let θ represent a parameter in the generative model. Assuming i.i.d. data, the derivative of the log-likelihood of a training set $\mathbf{x}_1, \ldots, \mathbf{x}_T$ with respect to a parameter θ is

$$\frac{\partial \log p(\mathbf{x}_1, \ldots, \mathbf{x}_T)}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial}{\partial \theta} \log\Big(\sum_{c=1}^{C} \sum_{\ell=1}^{L} \int_{\mathbf{z}} d\mathbf{z}\, p(\mathbf{x}_t, \ell, \mathbf{z}, c) \Big). \quad (7)$$

Taking the derivative and rearranging terms, this can be rewritten

$$\frac{\partial \log p(\mathbf{x}_1, \ldots, \mathbf{x}_T)}{\partial \theta} = \sum_{t=1}^{T} \mathrm{E}\Big[\frac{\partial}{\partial \theta} \log p(\mathbf{x}_t, \ell, \mathbf{z}, c) \Big| \mathbf{x}_t \Big], \quad (8)$$

where the expectation is based on the posterior distribution,

$$p(\ell, c, \mathbf{z}|\mathbf{x}_t) = P(c, \ell|\mathbf{x}_t) p(\mathbf{z}|c, \ell, \mathbf{x}_t). \quad (9)$$

θ can be updated by setting (8) to 0, ignoring the dependence of the expectation on θ and then solving for θ to obtain a new value $\tilde{\theta}$. However, modifying θ will cause a change in the posterior distribution and the expectations and so the new log-likelihood derivative will not be equal to zero. The expectation maximization (EM) algorithm consists of iteratively computing a new set of parameters and then recomputing the expectation based on the updated parameters. This procedure consistently increases the likelihood of the training data. A fixed number of iterations can be used or the number of iterations can be determined by monitoring the log-likelihood of a validation set.

3.1 M-Step: The Maximizations

By setting (8) to 0 and solving for the new parameter values, we obtain update equations that are based on the expectations in (8). $\langle \cdot \rangle = \frac{1}{T} \sum_{t=1}^{T} (\cdot)$ indicates a statistic sufficient for the M-Step, which is computed by averaging over the training set as described in the next section; $\mathrm{diag}(\mathbf{A})$ gives a vector containing the diagonal elements of matrix $\mathbf{A}$; $\mathrm{diag}(\mathbf{a})$ gives a diagonal matrix whose diagonal contains the elements of vector $\mathbf{a}$; and $\mathbf{a} \circ \mathbf{b}$ computes the element-wise product of vectors $\mathbf{a}$ and $\mathbf{b}$. We denote the updated parameters by "$\tilde{\ }$".

$$\tilde{\pi}_k = \langle P(c = k|\mathbf{x}_t) \rangle, \quad (10)$$

$$\tilde{\boldsymbol{\mu}}_k = \frac{\langle P(c = k|\mathbf{x}_t) \mathrm{E}[\mathbf{z}|c = k, \mathbf{x}_t] \rangle}{\langle P(c = k|\mathbf{x}_t) \rangle}, \quad (11)$$

$$\tilde{\boldsymbol{\Phi}}_k = \mathrm{diag}\Big(\frac{\langle P(c = k|\mathbf{x}_t) \mathrm{E}[(\mathbf{z} - \boldsymbol{\mu}_k) \circ (\mathbf{z} - \boldsymbol{\mu}_k)|c = k, \mathbf{x}_t] \rangle}{\langle P(c = k|\mathbf{x}_t) \rangle} \Big), \quad (12)$$

$$\tilde{\boldsymbol{\Psi}} = \mathrm{diag}\big(\langle \mathrm{E}[(\mathbf{x}_t - \boldsymbol{\Gamma}_\ell \mathbf{z}) \circ (\mathbf{x}_t - \boldsymbol{\Gamma}_\ell \mathbf{z}) \big| \mathbf{x}_t] \rangle \big). \quad (13)$$

Notice that the transformation index ℓ is summed over within the above expectations. If the transformation probabilities are to be estimated, we have

$$\tilde{p}_l = \langle P(\ell = l|\mathbf{x}_t) \rangle. \quad (14)$$

In order to avoid overfitting the noise variances, it is sometimes useful to set the diagonal elements of $\boldsymbol{\Phi}_k$ and $\boldsymbol{\Psi}$ that are below some ϵ equal to ϵ.

3.2 E-Step: The Expectations

The sufficient statistics for the M-Step are computed in the E-Step during a single pass through the training set. To compute the sufficient statistics, the covariance matrices of $\mathbf{z}$ given c, ℓ and $\mathbf{x}$ are needed. Fortunately, these matrices do not depend on $\mathbf{x}$, so they can be computed once before each E-Step:

$$\boldsymbol{\Omega}_{c\ell} = \mathrm{COV}(\mathbf{z}|c, \ell, \mathbf{x}) = (\boldsymbol{\Phi}_c^{-1} + \boldsymbol{\Gamma}_\ell' \Psi^{-1} \boldsymbol{\Gamma}_\ell)^{-1}. \quad (15)$$

This expression was obtained by rearranging the terms in the exponents of (4) to obtain a distribution over $\mathbf{z}$.

During the pass through the training set, the responsibility of each cluster - transformation pair for training case $\mathbf{x}_t$ is computed:

$$\begin{aligned} P(c, \ell|\mathbf{x}_t) &= \alpha p(\mathbf{x}_t, c, \ell) \\ &= \alpha \pi_c p_\ell \mathcal{N}(\mathbf{x}_t; \boldsymbol{\Gamma}_\ell \boldsymbol{\mu}_c, \boldsymbol{\Gamma}_\ell \boldsymbol{\Phi}_c \boldsymbol{\Gamma}_\ell' + \boldsymbol{\Psi}), \end{aligned} \quad (16)$$

where α is a normalization constant computed so that $\sum_{c=1}^{C} \sum_{\ell=1}^{L} P(c, \ell|\mathbf{x}_t) = 1$. $P(c, \ell|\mathbf{x}_t)$ is then marginalized for each value of c and normalized for each value of c to obtain $P(c|\mathbf{x}_t)$ and $P(\ell|c,\mathbf{x}_t)$.

For each c and ℓ, compute

$$\mathrm{E}[\mathbf{z}|c,\ell,\mathbf{x}_t] = \mathbf{\Omega}_{c\ell}\mathbf{\Gamma}'_\ell\mathbf{\Psi}^{-1}\mathbf{x}_t + \mathbf{\Omega}_{c\ell}\mathbf{\Phi}_c^{-1}\boldsymbol{\mu}_c, \qquad (17)$$

which is derived from (4) and (15). This is marginalized to obtain the expectation in (11):

$$\mathrm{E}[\mathbf{z}|c,\mathbf{x}_t] = \sum_{\ell=1}^{L} P(\ell|c,\mathbf{x}_t)\mathrm{E}[\mathbf{z}|c,\ell,\mathbf{x}_t]. \qquad (18)$$

The expectation in (12) is computed from

$$\begin{aligned}\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c)\circ(\mathbf{z}-\boldsymbol{\mu}_c)|c,\mathbf{x}_t] &= \sum_{\ell=1}^{L} P(\ell|c,\mathbf{x}_t)\\ \cdot\big\{(\mathrm{E}[\mathbf{z}|c,\ell,\mathbf{x}_t]-\boldsymbol{\mu}_c)&\circ(\mathrm{E}[\mathbf{z}|c,\ell,\mathbf{x}_t]-\boldsymbol{\mu}_c)\\ &+\mathrm{diag}(\mathbf{\Omega}_{c\ell})\big\} \qquad (19)\end{aligned}$$

and the expectation in (13) is computed from

$$\begin{aligned}\mathrm{E}[(\mathbf{x}_t-\mathbf{\Gamma}_\ell\mathbf{z})\circ(\mathbf{x}_t-\mathbf{\Gamma}_\ell\mathbf{z})|\mathbf{x}_t] &= \sum_{c=1}^{C}\sum_{\ell=1}^{L} P(c,\ell|\mathbf{x}_t)\\ \cdot\big\{(\mathbf{x}_t-\mathbf{\Gamma}_\ell\mathrm{E}[\mathbf{z}|c,\ell,\mathbf{x}_t])&\circ(\mathbf{x}_t-\mathbf{\Gamma}_\ell\mathrm{E}[\mathbf{z}|c,\ell,\mathbf{x}_t])\\ &+\mathrm{diag}(\mathbf{\Gamma}_\ell\mathbf{\Omega}_{c\ell}\mathbf{\Gamma}'_\ell)\big\}. \qquad (20)\end{aligned}$$

3.3 Performing an EM Iteration in Linear Time

It turns out that each EM iteration uses $\mathcal{O}(CLNT)$ scalar operations, where N is the number of pixels in the observed image. The same reasoning that was used in Scn. 2.1 can be used to show that the matrix $\mathbf{\Omega}_{c\ell} = (\mathbf{\Phi}_c^{-1} + \mathbf{\Gamma}'_\ell\Psi^{-1}\mathbf{\Gamma}_\ell)^{-1}$ is computed using $\mathcal{O}(N)$ scalar operations and contains $\mathcal{O}(N)$ nonzero elements. It follows that multiplying a vector by $\mathbf{\Omega}_{c\ell}$ takes $\mathcal{O}(N)$ scalar operations. So, for each training case, computing the statistics in (17), (18), (19) and (20) uses $\mathcal{O}(CLN)$ scalar operations.

4 Extracting Shapes From Synthetic Data

Fig. 3a shows 100 examples from a training set of 200 cases of 9×9 images. Each image contains one of four shapes: a large square, a large circle, a small filled square or a small "pac-man". The background was produced by randomly selecting pixel intensities independently from a uniform distribution. In addition, the background includes a fixed distraction in the form of two pixels that are always set to have maximum intensity.

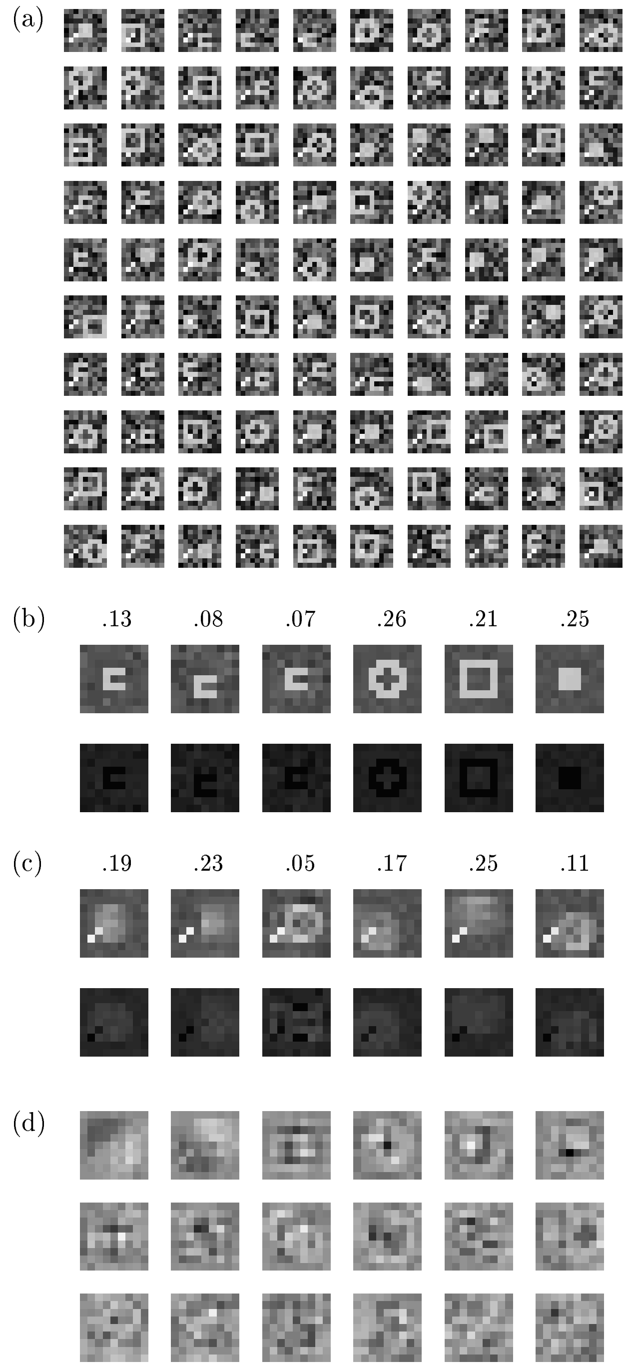

Figure 3: Extracting transformation invariant structure from synthetic data using a TMG. (a) Training examples, which include background clutter and a fixed distraction. (b) Mixing proportions, means and variances for a 6-cluster TMG. (c) Mixing proportions, means and variances for a 6-cluster mixture of Gaussians. (d) First 18 principal components (eigenimages).

(a) (b)

Figure 4: (a) Images of a person walking across a cluttered background while looking into the camera. (b) The cluster means and variances for a TMG trained using 20 iterations of the EM algorithm.

We trained a TMG containing $C = 6$ clusters and $L = 25$ translation transformations (5 horizontal shifts and 5 vertical shifts) using 20 iterations of the EM algorithm. The weights were initialized to small, random values and the mixing proportions were initialized to be equal. Fig. 3b shows the mixing proportions, cluster means $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_6$ and the diagonal elements of the cluster covariance matrices $\boldsymbol{\Phi}_1, \ldots, \boldsymbol{\Phi}_6$. Since the TMG had 2 extra clusters than necessary, it used the first 3 clusters to model the "pac-man". The remaining 3 clusters model the remaining shapes. Notice that for a given cluster, the variances indicate which pixels are background pixels (light, for high variance) versus foreground pixels (dark, for low variance).

Fig. 3c shows the parameters learned using 20 iterations of the EM algorithm for a traditional mixture model with 6 clusters. This model can be viewed as a special type of TMG that uses just the identity transformation. The shapes are severely blurred and the model fixates on the distraction. If the number of clusters is increased, the model can capture different transformations using different clusters. However, for 4 shapes and 25 transformations, there are 100 distinct clusters in the training set of 200 patterns. Training a mixture model with 100 clusters on 200 patterns would result in severely overfitting the noise.

Fig. 3d shows the first 18 principal components, or "eigenimages" [5, 6], of the training data. It is difficult to imagine how these components can be used to reconstruct the data accurately.

5 Extracting Translation Invariant Structure from Outdoor Video Sequences with Cluttered Backgrounds

Fig. 4a shows 15 examples from 100-frame head-and-shoulder video sequences of a person walking across a highly cluttered background while looking into the camera.

(a) (b) (c)

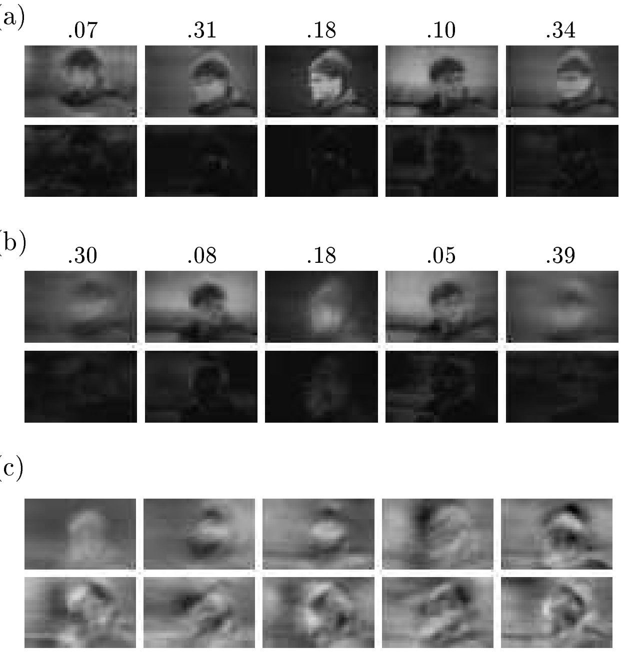

Figure 5: (a) Mixing proportions, cluster means and cluster variances for a TMG trained using 40 iterations of the EM algorithm. Note that clusters 2, 3 and 5 account for most of the data. (b) Mixing proportions, cluster means and cluster variances for a traditional mixture model. (c) First 10 principal components of the training data.

We trained a TMG containing $C = 2$ clusters and $L = 81$ translation transformations (9 horizontal shifts and 9 vertical shifts) using 20 iterations of the EM algorithm. Just like for the synthetic problem, the weights were initialized to small, random values and the mixing proportions were initialized to be equal. Fig. 4b shows the cluster means and variances. The background has been suppressed, while the detail in the faces has been retained.

Fig. 1 shows 25 examples from a 400-frame video sequence where the individual changed his pose. We trained a TMG containing $C = 5$ clusters and $L = 121$ translation transformations (11 horizontal shifts and 11 vertical shifts) using 40 iterations of the EM algorithm. Fig. 5a shows the resulting mixing proportions, cluster means and cluster variances. Most clusters have suppressed the background clutter, despite significant movements of the object in the field of vision. The mean for cluster 4 includes part of the background, but this cluster also has a low mixing proportion.

Fig. 5b shows the mixing proportions, cluster means and cluster variances for a traditional mixture

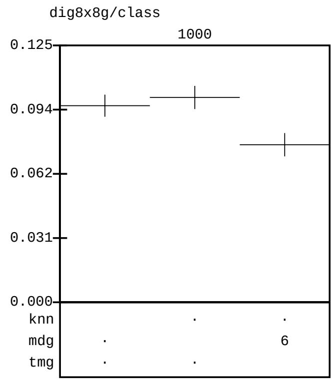

Figure 6: Estimated error rates and confidence intervals for k-nearest neighbors, mixtures of Gaussians and transformed mixtures of Gaussians on the task of handwritten digit recognition. 4 completely disjoint training set / test set partitions were used to obtain these estimates and each image was used in only one of the 4 experiments. Each training set had 100 examples of each digit (0 to 9) and each test set had 75 examples of each digit. The numbers in the lower box give the p-values (in percent) for a paired t-test on the null hypothesis that the corresponding two methods have identical performance. A dot indicates the p-value was above 9%. A low p-value indicates the two methods are different with statistical significance.

model trained using 40 iterations of the EM algorithm. The cluster means are significantly more blurred in comparison with the means found by the TMG.

Fig. 5c shows the first 10 principal components of the training data. Most of the components shown appear to account for ways to perturb the mean of the data to obtain horizontal, vertical and diagonal shifts.

6 Handwritten Digit Recognition

We trained one TMG on each class of handwritten digit from a small subset of the CEDAR CDROM archive and used (1) to classify new test cases. We used 29 transformations that captured various shearing-translation combinations. For each class, the number of clusters was determined by setting 1/3 of the training data aside for validation. To make the problem more difficult, each model was trained using only 100 images. Fig. 6 compares the performances of k-nearest neighbors, a traditional nontransformed mixture of Gaussians and the TMG classifier, which performs the best.

7 Summary

We introduced a new type of Gaussian mixture model that is invariant to transformations and have presented an efficient EM algorithm for learning the model parameters. The algorithm is capable of suppressing background clutter and learning translation invariant structure from synthetic and natural images.

This general statistical method can be applied to a variety of types of data, including nonvisual data, and it can be extended to take into account temporal coherence in time series [4]. We are also exploring "transformed component analysis", where the mixture model is replaced by a probabilistic form of PCA [10] and ways of using approximate inference so that the number of computations per iteration of EM scales better.

References

[1] A. Jepson and M. J. Black, "Mixture models for optical flow computation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1993, pp. 760–761.

[2] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing, Special Issue: Image Sequence Compression*, vol. 3, no. 5, pp. 625–638, September 1994.

[3] Y. Weiss, "Smoothness in layers: Motion segmentation using nonparametric mixture estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1997, pp. 520–527.

[4] N. Jojic and B. J. Frey, "Dynamic transformed mixtures of Gaussians," Submitted to *Neural Information Processing Systems Conference 1999*, 1999.

[5] M. Turk and A Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, 1991.

[6] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696–710, July 1997.

[7] K.-K. Sung and T. Poggio, "Example-based learning for view-based human face detection," MIT AI Memo 1521, CBCL Paper 112, 1994.

[8] H. A. Rowley, S. Baluja, and T. Kanade, "Rotation invariant neural network-based face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1998, pp. 38–44.

[9] D. Rubin and D. Thayer, "EM algorithms for ML factor analysis," *Psychometrika*, vol. 47, no. 1, pp. 69–76, 1982.

[10] B. J. Frey and N. Jojic, "Transformed component analysis: Joint estimation of spatial transformations and image components," Submitted to the *International Conference on Computer Vision*, 1999.