Transformation-Invariant Clustering and Dimensionality Reduction Using EM

Brendan J. Frey and Nebojsa Jojic

Abstract

Clustering and dimensionality reduction are simple, effective ways to derive useful representations of data, such as images. These procedures often are used as preprocessing steps for more sophisticated pattern analysis techniques. (In fact, these procedures often perform as well as or better than more sophisticated pattern analysis techniques.) However, in situations where each input has been randomly transformed (*e.g.*, by translation, rotation and shearing in images), these methods tend to extract cluster centers and submanifolds that account for variations in the input due to transformations, instead of more interesting and potentially useful structure. For example, if images of a human face are clustered, it would be more useful for the different clusters to represent different poses and expressions, instead of different translations and rotations. We describe a way to add transformation invariance to mixture models, factor analyzers and mixtures of factor analyzers by approximating the nonlinear transformation manifold by a discrete set of points. In contrast to linear approximations of the transformation manifold, which assume the amount of transformation is small, our method works well for large levels of transformation. We show how the expectation maximization algorithm can be used to jointly learn a set of clusters, a subspace model, or a mixture of subspace models and at the same time infer the transformation associated with each case. After illustrating this technique on some difficult contrived problems, we compare the technique with other methods for filtering noisy images obtained from a scanning electron microscope, clustering images of faces into different categories of identification and pose, subspace modeling of facial expressions, subspace modeling of images of handwritten digits for handwriting classification, and unsupervised classification of images of handwritten digits.

I. 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 and shearing. For example, Fig. 1 shows several 140×56 pixel greyscale images obtained from a scanning electron microscope. The electron detectors and the high-speed electrical circuits randomly translate the images and add noise [1]. Standard filtering techniques are not appropriate here, since the images are not aligned. Due to the high level of noise, it is difficult to properly align them by hand and this requires human effort.

Fig. 2 shows several 44×28 pixel greyscale head-and-shoulder images of a person walking outdoors. The camera did not track the persons head perfectly, so the head appears at different locations. The images include variation in the pose of the head, as well as background clutter – some of which appears in multiple images. Aligning the images without a model of the person's appearance and without temporal information is difficult. Even with temporal information, (a video sequence) standard blob-tracking methods do not work well due to the presence of coherent background clutter.

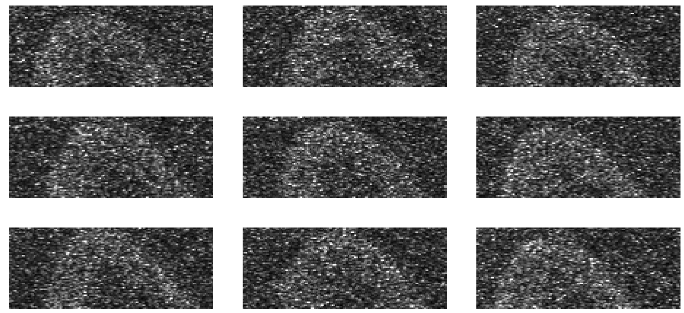

Fig. 1. Several images of size 140×56 pixels, taken by a scanning electron microscope. The electron detectors and high-speed electrical circuits introduce random translations.

Fig. 2. Some images of size 44×28 pixels, of a person walking outdoors. The head has different poses and appears at different positions in the field of view. In addition, the background is highly cluttered and there is variation in lighting conditions.

Fig. 3 shows preprocessed greyscale images of handwritten digits from postal envelopes [2]. Unlike the microscope images described above, in this case the boundaries of the digits on the envelopes were more easily identifiable, so the digits are normalized for horizontal and vertical scale and translation before sampling the 8×8 pixel images shown in the figure. However, the digits are written at different writing angles (c.f. the vertical stroke in different versions of "7"), which can be thought of as

B. J. Frey is faculty in Computer Science at the University of Waterloo and adjunct faculty in Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

N. Jojic is a doctoral candidate in the Image Formation and Processing group at the Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

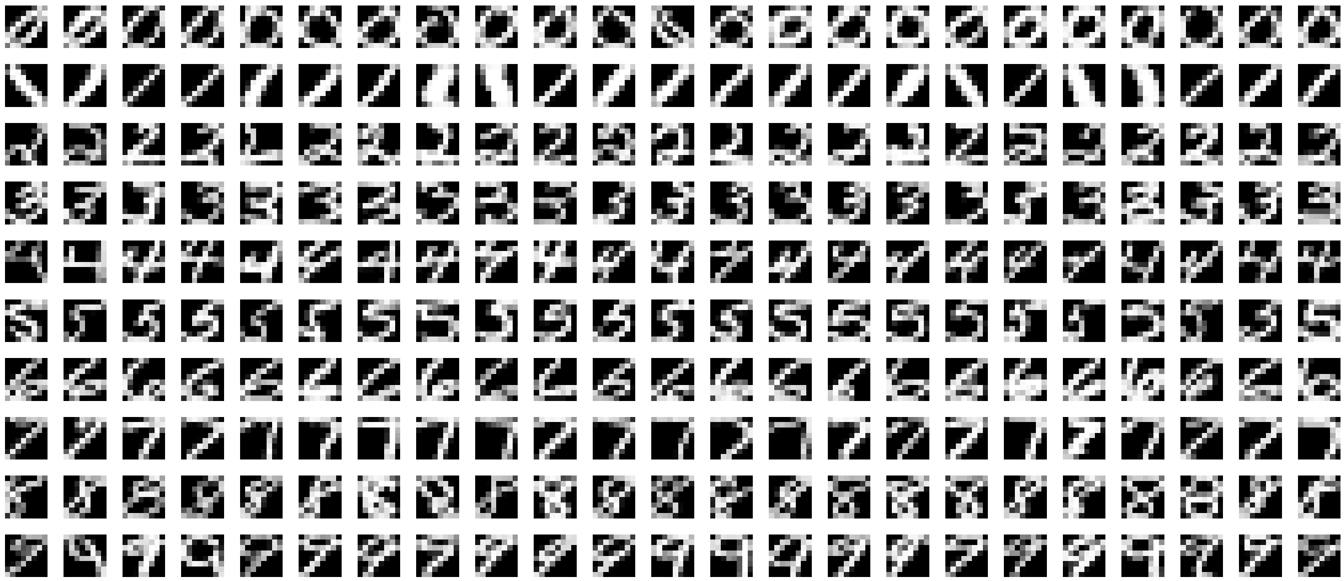

Fig. 3. Images of handwritten digits, normalized for horizontal and vertical scale and translation and sampled on an 8×8 pixel grid. Different writing angles introduce different levels of shearing in each image.

randomly selected levels of horizontal shearing. The appropriate level of shearing needed to normalize for writing angle depends on the identity of the digit (compare "0"s with "1"s), so normalizing for shearing is not straightforward as a preprocessing step.

We propose a general purpose statistical method that can jointly normalize out transformations that occur in training data, while learning a density model of the normalized data [3, 4]. In this paper, we do not assume the data is ordered. Clearly, temporal coherence provides useful cues for modeling time-series data such as video sequences [5–7]. In [8–10], we show how the techniques introduced in this paper can be extended to discrete-state dynamic models (hidden Markov models).

One approach to data modeling and machine learning is to use labeled data to train a recognition model to accurately predict class membership from the input. This supervised learning approach includes nonlinear regression techniques such as classification and regression trees [11], neural networks [12–14], Gaussian process regression [15], support vector classifiers [16], and nearest-neighborhood type methods, including "eigen-space" methods that compute distances within subspaces [17, 18].

In contrast, the approach we take here is to use *unlabeled data* to train a probability density model of the data (or *generative model*), in an unsupervised fashion. Two common data processing techniques that can be viewed in this way are clustering and linear dimensionality reduction (principal components analysis). These procedures correspond to estimation of the following density models: the mixture of Gaussians [19] and factor analysis [20]. By restricting these density models in various ways, maximum likelihood estimation corresponds to standard non-probabilistic algorithms. For example, by setting the covariance matrices of the Gaussians in a mixture of Gaussians to be $\epsilon\mathbf{I}$, $\epsilon \to 0$, k-means clustering is obtained. By restricting the factor loading matrix and sensor variances in the factor analyzer, principal components analysis is obtained. However, the probabilistic versions of these techniques have distinct advantages [19].

Unsupervised learning is useful for summarizing data (*e.g.*, finding 5 common head poses in the data from Fig. 2), filtering data (*e.g.*, denoising the images from Fig. 1), estimating density models used for data compression, and as a preprocessing step for supervised methods (*e.g.*, removing the shearing from the handwritten digits in Fig. 3 before training a classifier in a supervised fashion).

By thinking of unsupervised learning as maximum likelihood estimation of a density model of the data, we can incorporate extra knowledge about the problem. One way to do this is to include extra *latent variables* (unobserved variables) in the model. The model we present extends the mixture of Gaussians, the factor analyzer and the mixture of factor analyzers to include "transformation" as a latent variable. The model can be trained using the expectation maximization (EM) algorithm.

In the next section, we describe two computationally efficient approaches to modeling transformations. Then, we describe how these approaches can be incorporated into the generative models for a mixture of Gaussians (clustering), a factor analyzer (linear dimensionality reduction) and a mixture of factor analyzers (clustering and dimensionality reduction). We refer to these models as "transformation-invariant models". We describe how the transformation-invariant models can be fit to a training set using the expectation maximization (EM) algorithm. After illustrating the models on some difficult contrived problems, we compare them with other methods for filtering noisy images obtained from a scanning electron microscope, clustering images of faces into different categories of identification and pose, subspace modeling of facial expressions, subspace modeling of images of handwritten digits for handwriting classification, and unsupervised classification of images of handwritten digits. We focus on vision problems, but the methods can be applied to any type of data.

II. Discrete and Linear Approximations to the Transformation Manifold

To make data models invariant to a known type of transformation in the input, we would like to make all transformed versions of a particular input equivalent. Suppose an N-element input undergoes a transformation with 1 degree of freedom – for example, an N-pixel greyscale image undergoes translation in the x-direction, with wrap-around. Imagine what happens to the point in the N-dimensional pixel intensity space while the object is translated. Due to pixel mixing, a very small amount of subpixel translation will move the point only slightly, so translation will trace a continuous 1-dimensional curve in the space of pixel intensities. As illustrated in Fig. 4, extensive levels of translation will produce a highly nonlinear curve (consider translating a thin vertical line), although the curve can be approximated by a straight line locally. If K types of continuous transformation are applied, the manifold will be K-dimensional.

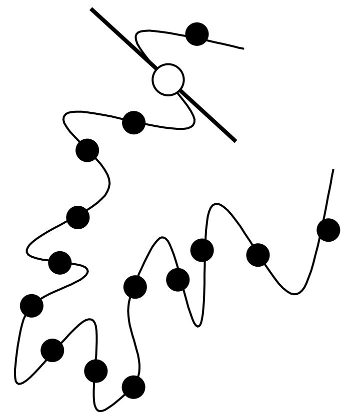

Fig. 4. An N-element input vector is represented by a point (unfilled disc) in an N-dimensional space. When the input undergoes a continuous transformation with 1 degree of freedom, a 1-dimensional manifold is traced. For transformation-invariant data modeling, we want all inputs on this manifold to be equivalent in some sense. Locally, the curve is linear, but high levels of transformation may produce a highly nonlinear curve. We approximate the manifold by discrete points (filled discs) indexed by ℓ.

Linear approximations of the transformation manifold have been used to significantly improve the performance of supervised classifiers such as nearest neighbors [21] and multilayer perceptrons [22].

Linear generative models (factor analysis, mixtures of factor analysis) have also been modified using linear approximations of the transformation manifold to build in some degree of transformation invariance [23].

In general, the linear approximation is accurate for transformations that couple neighboring pixels, but is inaccurate for transformations that couple nonneighboring pixels. In some applications (*e.g.*, handwritten digit recognition), the input can be blurred so that the linear approximation becomes valid for more severe transformations [21]. A multiresolution version of the linear approximation is proposed in [24].

In general, for significant levels of transformation, the nonlinear manifold can be better modeled using a discrete approximation. For example, the curve in Fig. 4 can be represented by a set of points (filled discs). In this approach, a discrete set of possible transformations is specified beforehand and parameters are learned so that the model is invariant to the set of transformations. This approach has been used in the supervised framework to design "convolutional neural networks" that are trained using labeled data [25].

We describe how invariance to a discrete set of transformations (like translation in images) can be built into a generative density model and we show how an EM algorithm for the original density model can be extended to the new model by computing expectations over the set of transformations.

III. Transformation as a Discrete Latent Variable

In this section, we show how to incorporate the discrete and linear approximations described above into various generative models. Conditioning on the discrete variables, all of the models presented here are jointly Gaussian, so inference is computationally efficient. Although many expressions may look complicated, they are "straightforward" linear algebra. We have posted MATLAB routines for these algorithms on our webpage, www.cs.toronto.edu/∼frey.

For the sake of clarity, we now focus on image data and transformations such as translation.

We represent the ℓth transformation by a sparse transformation matrix $\mathbf{G}_\ell$ that operates on a vector of image pixel intensities. For example, integer-pixel translations of an image can be represented by permutation matrices. Although other types of transformation matrix may not be accurately represented by permutation matrices, many useful types of transformation can be represented by sparse transformation matrices. For example, rotation and blurring can be represented by matrices that have a small number of nonzero elements per row (*e.g.*, at most 6 for rotations). Alternatively, these transformations can be approximated using permutation matrices.

The observed image $\mathbf{x}$ is linked to the nontransformed *latent image* $\mathbf{z}$ and the transformation index $\ell \in \{1, \ldots, L\}$ as follows:

$$p(\mathbf{x}|\ell, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{G}_\ell \mathbf{z}, \mathbf{\Psi}), \tag{1}$$

where $\mathbf{\Psi}$ is a diagonal matrix of sensor noise variances. It is sometimes advantageous to set $\mathbf{\Psi} = \mathbf{0}$, as described below.

Since the probability of a transformation may depend on the latent image, the joint distribution over the latent image $\mathbf{z}$, the transformation index ℓ and the observed image $\mathbf{x}$ is

$$p(\mathbf{x}, \ell, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{G}_\ell \mathbf{z}, \mathbf{\Psi}) P(\ell|\mathbf{z}) p(\mathbf{z}). \tag{2}$$

The corresponding graphical model is shown in Fig. 5a.

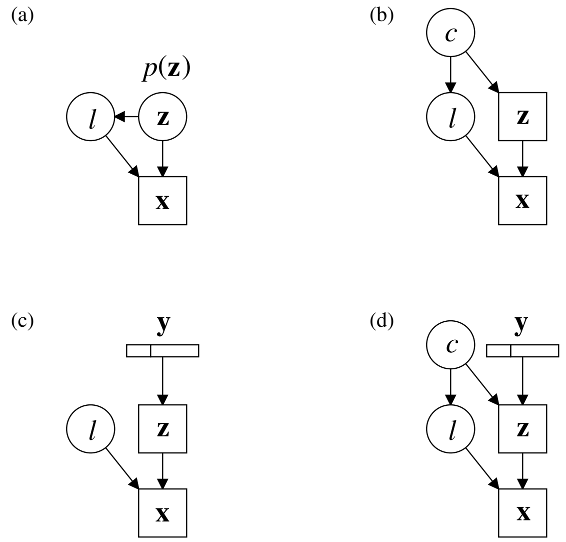

Fig. 5. (a) A graphical model showing how a discrete transformation variable ℓ can be added to a density model $p(\mathbf{z})$ for a *latent image* $\mathbf{z}$ to model the observed image $\mathbf{x}$. The Gaussian pdf $p(\mathbf{x}|\ell,\mathbf{z})$ captures the ℓth transformation plus a small amount of pixel noise. (We use a box to represent variables that have Gaussian conditional pdfs.) We have explored (b) transformed mixtures of Gaussians, where c is a discrete cluster index; (c) transformed component analysis (TCA), where $\mathbf{y}$ is a vector of Gaussian factors, some of which may model locally linear transformation perturbations; and (d) mixtures of transformed component analyzers, or transformed mixtures of factor analyzers.

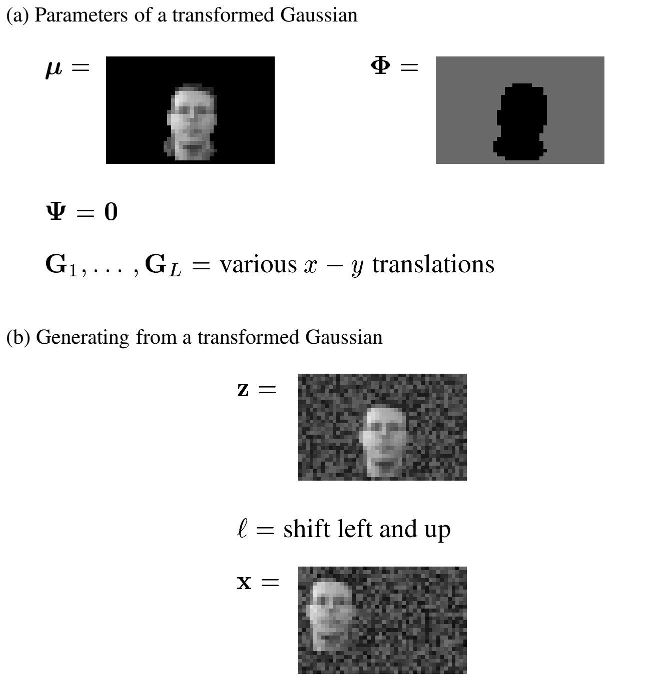

Fig. 6. A hand-crafted model illustrates how a discrete transformation index is incorporated into a Gaussian model. Whereas $\boldsymbol{\Phi}$ models additive, Gaussian noise that gets transformed, $\boldsymbol{\Psi}$ models additive, Gaussian noise that is not transformed.

A. Transformed Gaussians

To model noisy transformed images of just one shape, we choose $p(\mathbf{z})$ to be a Gaussian distribution:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Phi}), \tag{3}$$

where $\boldsymbol{\mu}$ is the mean of the Gaussian and $\boldsymbol{\Phi}$ is the covariance matrix. We usually take $\boldsymbol{\Phi}$ to be diagonal to reduce the number of parameters that need to be estimated.

For simplicity, we assume that in the absence of any observations, ℓ is independent of $\mathbf{z}$: $P(\ell|\mathbf{z}) = P(\ell)$. So, the joint distribution is

$$p(\mathbf{x}, \ell, \mathbf{z}) = \mathcal{N}(\mathbf{x}; \mathbf{G}_\ell \mathbf{z}, \boldsymbol{\Psi}) \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Phi}) \rho_\ell, \tag{4}$$

where ρ_ℓ is the probability of transformation ℓ.

The two parameters $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ represent two very different types of noise. The noise modeled by $\boldsymbol{\Phi}$ is added *before* the transformation is applied, whereas the noise modeled by $\boldsymbol{\Psi}$ is added *after* the transformation is applied. In images, large values on the diagonal of $\boldsymbol{\Phi}$ indicate regions in the latent image that are not accurately predicted by $\boldsymbol{\mu}$. These regions may correspond to background clutter or parts of an object that appear noisy (*e.g.*, blinking eyes).

Fig. 6a shows hand-crafted parameters of a transformed Gaussian that models a face appearing at different positions in the frame. $\boldsymbol{\mu}$ is shown in raster-scan format. $\boldsymbol{\Phi}$ is diagonal and the figure shows the diagonal elements of $\boldsymbol{\Phi}$ in raster-scan format, with large variances painted bright and small variances painted dark. The variance map indicates that the head region is modeled accurately by $\boldsymbol{\mu}$, whereas the surrounding region is not. Fig. 6b shows one configuration of the variables in the model, drawn from the above joint distribution. First, $\mathbf{z}$ is drawn by adding independent Gaussian noise to $\boldsymbol{\mu}$, with variances given by $\boldsymbol{\Phi}$. Next, a transformation index ℓ is drawn. Finally, transformation $\mathbf{G}_\ell$ is applied to $\mathbf{z}$ and independent Gaussian noise with variances $\boldsymbol{\Psi}$ ($\mathbf{0}$ in this case) is added to the pixels to produce $\mathbf{x}$.

To compute the density of the image under a particular transformation, we integrate over $\mathbf{z}$:

$$\begin{aligned} p(\mathbf{x}|\ell) &= \int_{\mathbf{z}} d\mathbf{z} p(\mathbf{x}, \ell, \mathbf{z}) \\ &= \int_{\mathbf{z}} d\mathbf{z} \mathcal{N}(\mathbf{x}|\mathbf{G}_\ell \mathbf{z}, \boldsymbol{\Psi}) \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Phi}) \\ &= \mathcal{N}(\mathbf{x}; \mathbf{G}_\ell \boldsymbol{\mu}, \mathbf{G}_\ell \boldsymbol{\Phi} \mathbf{G}_\ell^\mathsf{T} + \boldsymbol{\Psi}), \end{aligned} \tag{5}$$

where "$^\mathsf{T}$" indicates matrix transpose. Each transformation ℓ has a corresponding mean image $\mathbf{G}_\ell \boldsymbol{\mu}$ and covariance matrix $\mathbf{G}_\ell \boldsymbol{\Phi} \mathbf{G}_\ell^\mathsf{T} + \boldsymbol{\Psi}$.

The conditional density $p(\mathbf{x}|\ell)$ looks like the likelihood for a mixture of factor analyzers [26]. However, whereas the likelihood computation for N latent pixels takes order N^3 time in a mixture of factor analyzers, it takes *linear* time, order N, in the models considered here, because $\mathbf{G}_\ell \boldsymbol{\Phi} \mathbf{G}_\ell^\mathsf{T} + \boldsymbol{\Psi}$ is sparse.

The probability density of $\mathbf{x}$ under the model is

$$p(\mathbf{x}) = \sum_{\ell=1}^{L} \rho_\ell \mathcal{N}(\mathbf{x}; \mathbf{G}_\ell \boldsymbol{\mu}, \mathbf{G}_\ell \boldsymbol{\Phi} \mathbf{G}_\ell^\mathsf{T} + \boldsymbol{\Psi}). \tag{6}$$

Notice that if $\mathbf{G}_\ell \boldsymbol{\Phi} \mathbf{G}_\ell^\mathsf{T}$ is full rank for all ℓ, we can set $\boldsymbol{\Psi} = \mathbf{0}$ if we wish. We may do this, for example, to reduce the number of parameters that need to be estimated. Typically, $\mathbf{\Phi}$ is full rank, so $\mathbf{G}_\ell\mathbf{\Phi}\mathbf{G}_\ell^\mathsf{T}$ is full rank if $\mathbf{G}_\ell$ has rank N, where N is the number of pixels in the observed image $\mathbf{x}$.

For a given input image, we can compute the probability of each transformation:

$$P(\ell|\mathbf{x}) = p(\mathbf{x}|\ell)\rho_\ell/p(\mathbf{x}). \tag{7}$$

Transformation normalization ("stabilization") can be performed by computing the expected value of the latent image $\mathbf{z}$, given the observed image $\mathbf{x}$. Since $\mathbf{z}$ and $\mathbf{x}$ are jointly Gaussian given ℓ, we first compute $\mathrm{E}[\mathbf{z}|\ell,\mathbf{x}]$. After some linear algebra, we obtain

$$\mathrm{E}[\mathbf{z}|\ell,\mathbf{x}] = \mathbf{\Omega}_\ell\mathbf{G}_\ell^\mathsf{T}\mathbf{\Psi}^{-1}\mathbf{x} + \mathbf{\Omega}_\ell\mathbf{\Phi}^{-1}\boldsymbol{\mu}, \tag{8}$$

where $\mathbf{\Omega}_\ell$ is the covariance of $\mathbf{z}$ given ℓ and $\mathbf{x}$:

$$\mathbf{\Omega}_\ell = \mathrm{COV}(\mathbf{z}|\ell,\mathbf{x}) = (\mathbf{\Phi}^{-1} + \mathbf{G}_\ell^\mathsf{T}\mathbf{\Psi}^{-1}\mathbf{G}_\ell)^{-1}. \tag{9}$$

The normalized image is then computed from

$$\mathrm{E}[\mathbf{z}|\mathbf{x}] = \sum_{\ell=1}^{L} P(\ell|\mathbf{x})\mathrm{E}[\mathbf{z}|\ell,\mathbf{x}]. \tag{10}$$

If we set $\mathbf{\Psi} = \mathbf{0}$, then $\mathrm{COV}(\mathbf{z}|\ell,\mathbf{x}) = \mathbf{0}$ and

$$\mathrm{E}[\mathbf{z}|\ell,\mathbf{x}] = (\mathbf{G}_\ell^\mathsf{T}\mathbf{G}_\ell)^{-1}\mathbf{G}_\ell^\mathsf{T}\mathbf{x}. \tag{11}$$

B. Transformed mixtures of Gaussians (TMG)

Fig. 5b shows the graphical model for a transformed mixture of Gaussians (TMG) [27,28], where different clusters may have different transformation probabilities. Cluster c has mixing proportion π_c, mean $\boldsymbol{\mu}_c$ and covariance matrix $\mathbf{\Phi}_c$. We usually take $\mathbf{\Phi}_c$ to be diagonal to reduce the number of parameters that need to be estimated. The joint distribution is

$$p(\mathbf{x},\ell,\mathbf{z},c) = \mathcal{N}(\mathbf{x};\mathbf{G}_\ell\mathbf{z},\mathbf{\Psi})\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_c,\mathbf{\Phi}_c)\rho_{\ell c}\pi_c, \tag{12}$$

where the probability of transformation ℓ for cluster c is $\rho_{\ell c}$. Marginalizing over the latent image gives the cluster/transformation conditional likelihood,

$$p(\mathbf{x}|\ell,c) = \mathcal{N}(\mathbf{x};\mathbf{G}_\ell\boldsymbol{\mu}_c,\mathbf{G}_\ell\mathbf{\Phi}_c\mathbf{G}_\ell^\mathsf{T} + \mathbf{\Psi}), \tag{13}$$

which can be used to compute $p(\mathbf{x})$,

$$p(\mathbf{x}) = \sum_{c=1}^{C}\sum_{\ell=1}^{L} p(\mathbf{x}|\ell,c)\rho_{\ell c}\pi_c, \tag{14}$$

and the cluster/transformation responsibility $P(\ell,c|\mathbf{x})$,

$$P(\ell,c|\mathbf{x}) = p(\mathbf{x}|\ell,c)\rho_{\ell c}\pi_c/p(\mathbf{x}). \tag{15}$$

C. Transformed component analyzer (TCA)

Fig. 5c shows the graphical model for transformation-invariant factor analysis, or "transformed component analysis" (TCA). The latent image is modeled using linearly combined isotropic Gaussian variables, $\mathbf{y}$.

The joint distribution is

$$p(\mathbf{x},\ell,\mathbf{z},\mathbf{y}) = \mathcal{N}(\mathbf{x};\mathbf{G}_\ell\mathbf{z},\mathbf{\Psi})\mathcal{N}(\mathbf{z};\boldsymbol{\mu}+\mathbf{\Lambda}\mathbf{y},\mathbf{\Phi})\mathcal{N}(\mathbf{y};\mathbf{0},\mathbf{I})\rho_\ell, \tag{16}$$

where $\boldsymbol{\mu}$ is the mean of the latent image, $\mathbf{\Lambda}$ is a matrix of latent image components (the factor loading matrix) and $\mathbf{\Phi}$ is a diagonal noise covariance matrix for the latent image. Marginalizing over the factor variables and the latent image gives the transformation conditional likelihood,

$$p(\mathbf{x}|\ell) = \mathcal{N}(\mathbf{x};\mathbf{G}_\ell\boldsymbol{\mu},\mathbf{G}_\ell(\mathbf{\Lambda}\mathbf{\Lambda}^\mathsf{T}+\mathbf{\Phi})\mathbf{G}_\ell^\mathsf{T} + \mathbf{\Psi}), \tag{17}$$

which can be used to compute $p(\mathbf{x})$ and the transformation responsibility $p(\ell|\mathbf{x})$.

In general, $|\mathbf{G}_\ell(\mathbf{\Lambda}\mathbf{\Lambda}^\mathsf{T}+\mathbf{\Phi})\mathbf{G}_\ell^\mathsf{T} + \mathbf{\Psi}|$ cannot be computed in time that is linear in N. However, the determinant *can* be computed in linear time if we set (or assume) $\mathbf{\Psi} = \mathbf{0}$, in which case

$$\begin{aligned}|\mathbf{G}_\ell(\mathbf{\Lambda}\mathbf{\Lambda}^\mathsf{T}+\mathbf{\Phi})\mathbf{G}_\ell^\mathsf{T} + \mathbf{\Psi}| &= |\mathbf{G}_\ell(\mathbf{\Lambda}\mathbf{\Lambda}^\mathsf{T}+\mathbf{\Phi})\mathbf{G}_\ell^\mathsf{T}| \\ &= |\mathbf{G}_\ell\mathbf{G}_\ell^\mathsf{T}||\mathbf{\Lambda}\mathbf{\Lambda}^\mathsf{T}+\mathbf{\Phi}| \\ &= |\mathbf{G}_\ell\mathbf{G}_\ell^\mathsf{T}||\mathbf{\Lambda}^\mathsf{T}\mathbf{\Phi}^{-1}\mathbf{\Lambda}+\mathbf{I}||\mathbf{\Phi}^{-1}|.\end{aligned} \tag{18}$$

Each of these determinants can be computed in time that is linear in N. In the experiments reported below, we set $\mathbf{\Phi} = \mathbf{0}$.

By setting columns of $\mathbf{\Lambda}$ equal to the derivatives of $\boldsymbol{\mu}$ with respect to continuous transformation parameters, a TCA can accommodate *both* a local linear approximation and a discrete approximation to the transformation manifold, as described in Scn. III-E.

D. Mixtures of transformed component analyzers (MTCA).

A combination of a TMG and a TCA can be used to jointly model clusters, linear components and transformations. Alternatively, a mixture of Gaussians that is invariant to a discrete set of transformations *and* locally linear transformations can be obtained by combining a TMG with a TCA whose components are all set equal to transformation derivatives, as described in Scn. III-E.

The joint distribution for the combined model in Fig. 5d is

$$\begin{aligned}&p(\mathbf{x},\ell,\mathbf{z},c,\mathbf{y}) = \\ &\quad\mathcal{N}(\mathbf{x};\mathbf{G}_\ell\mathbf{z},\mathbf{\Psi})\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_c+\mathbf{\Lambda}_c\mathbf{y},\mathbf{\Phi}_c)\mathcal{N}(\mathbf{y};\mathbf{0},\mathbf{I})\rho_{\ell c}\pi_c.\end{aligned} \tag{19}$$

The cluster/transformation likelihood is

$$p(\mathbf{x}|\ell,c) = \mathcal{N}(\mathbf{x};\mathbf{G}_\ell\boldsymbol{\mu}_c,\mathbf{G}_\ell(\mathbf{\Lambda}_c\mathbf{\Lambda}_c^\mathsf{T}+\mathbf{\Phi}_c)\mathbf{G}_\ell^\mathsf{T} + \mathbf{\Psi}), \tag{20}$$

which can be computed in linear time if we set (or assume) $\mathbf{\Phi} = \mathbf{0}$, as for TCA.

E. Incorporating the linear approximation to the transformation manifold

It turns out there is a simple way to now incorporate *both* the linear approximation and the global, nonlinear approximation of the transformation manifold.

Suppose we would like to model that data as a mixture of Gaussians with latent variables for both local and global $x-y$ translations. We can do this by using a mixture of transformed component analyzers (MTCA), where the number of factor variables is 2 and each of these variables corresponds to either x or y translation.

Given a vector of pixel intensities $\mathbf{x}$, we can resample it at a higher resolution, apply a small amount of x translation, subsample at the original resolution, and form a difference image, $\boldsymbol{\delta}_x(\mathbf{x})$. This is an approximation to the tangent vector to the transformation manifold. We can do the same for the y-direction and then set

$$\boldsymbol{\Lambda}_c = [\boldsymbol{\delta}_x(\boldsymbol{\mu}_c) \quad \boldsymbol{\delta}_y(\boldsymbol{\mu}_c)], \tag{21}$$

in the MTCA model.

The joint distribution for this MTCA is

$$p(\mathbf{x},\ell,\mathbf{z},c,\mathbf{y}) = \mathcal{N}(\mathbf{x};\mathbf{G}_\ell\mathbf{z},\boldsymbol{\Psi}) \\ \mathcal{N}(\mathbf{z};\boldsymbol{\mu}_c + \boldsymbol{\delta}_x(\boldsymbol{\mu}_c)y_1 + \boldsymbol{\delta}_y(\boldsymbol{\mu}_c)y_2, \boldsymbol{\Phi}_c)\mathcal{N}(\mathbf{y};\mathbf{0},\mathbf{I})\rho_{\ell c}\pi_c. \tag{22}$$

The standard normal variable y_1 will account for small x-translations, while the standard normal variable y_2 will account for small y-translations.

Other continuous transformations can be accounted for locally in a similar way.

F. 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, $L = n_h n_v n_r n_s$. If each of these components is large, an approximate inference algorithm should be used [29].

IV. Estimating Transformation-Invariant Models Using the Expectation Maximization Algorithm

We present an EM algorithm for the general MTCA model described above. The EM algorithm for TMG emerges by setting the number of factor variables to 0. The EM algorithm for TCA emerges by setting the number of clusters to 1.

Conditioned on the discrete latent variables (transformation index, and mixture component index for mixture models) all remaining variables in the above models are jointly Gaussian. Partial marginals of Gaussians are also Gaussian, so the distribution of the observed variables, $\mathbf{x}$ given the discrete latent variables ℓ and c is just a Gaussian (with a particular parameterization of the mean and covariance matrix). Aside from the model that incorporates the linear approximation to the transformation manifold, all other models are linear in the parameters. So, the EM algorithm for these models is essentially just a constrained, reparameterized version of the EM algorithm for a standard mixture of Gaussians [19]. In a later section, we describe how to estimate the model that incorporates the linear approximation to the transformation manifold.

A. M-Step

We use $\langle\cdot\rangle = \frac{1}{T}\sum_{t=1}^{T}(\cdot)$ to denote a sufficient statistic computed by averaging over the training set. These sufficient statistics are computed as shown in Scn. IV-B. Using $\mathrm{diag}(\mathbf{A})$ to denote a vector containing the diagonal elements of matrix $\mathbf{A}$; $\mathrm{diag}(\mathbf{a})$ to denote a diagonal matrix whose diagonal contains the elements of vector $\mathbf{a}$; $\mathbf{a}\circ\mathbf{b}$ to denote the element-wise product of vectors $\mathbf{a}$ and $\mathbf{b}$; and "˜" to denote the updated parameters, the M-Step for the MTCA updates the parameters as follows:

$$\tilde{\pi}_c = \langle P(c|\mathbf{x}_t)\rangle, \qquad \tilde{\rho}_{\ell c} = \langle P(\ell|\mathbf{x}_t,c)\rangle, \tag{23}$$

$$\tilde{\boldsymbol{\mu}}_c = \frac{\langle P(c|\mathbf{x}_t)\mathrm{E}[\mathbf{z}-\boldsymbol{\Lambda}_c\mathbf{y}|\mathbf{x}_t,c]\rangle}{\langle P(c|\mathbf{x}_t)\rangle}, \tag{24}$$

$$\tilde{\boldsymbol{\Phi}}_c = \frac{\mathrm{diag}\big(\langle P(c|\mathbf{x}_t)\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c-\boldsymbol{\Lambda}_c\mathbf{y})\circ(\mathbf{z}-\boldsymbol{\mu}_c-\boldsymbol{\Lambda}_c\mathbf{y})|\mathbf{x}_t,c]\rangle\big)}{\langle P(c|\mathbf{x}_t)\rangle}, \tag{25}$$

$$\tilde{\boldsymbol{\Psi}} = \mathrm{diag}\big(\langle \mathrm{E}[(\mathbf{x}_t-\mathbf{G}_\ell\mathbf{z})\circ(\mathbf{x}_t-\mathbf{G}_\ell\mathbf{z})|\mathbf{x}_t]\rangle\big), \tag{26}$$

$$\tilde{\boldsymbol{\Lambda}}_c = \langle P(c|\mathbf{x}_t)\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c)\mathbf{y}^{\mathrm{T}}|\mathbf{x}_t]\rangle\langle P(c|\mathbf{x}_t)\mathrm{E}[\mathbf{y}\mathbf{y}^{\mathrm{T}}|\mathbf{x}_t]\rangle^{-1}. \tag{27}$$

To reduce the number of parameters, we may assume that $\rho_{\ell c}$ does not depend on c or even that $\rho_{\ell c}$ is held constant at a uniform distribution.

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

B. E-Step

The sufficient statistics for the M-Step are computed in the E-Step using sparse linear algebra during a single pass through the training set. In what follows, it is important to keep in mind that the matrix $\mathbf{G}_\ell$ is very sparse (usually, a permutation matrix), so that matrices like $\mathbf{G}_\ell'\boldsymbol{\Psi}^{-1}\mathbf{G}_\ell$ are also very sparse.

Before making a pass through the training set to compute the sufficient statistics, the following matrices are computed:

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

$$\boldsymbol{\beta}_{\ell,c} = \mathrm{COV}(\mathbf{y}|\mathbf{x},\ell,c) \\ = (\mathbf{I} + \boldsymbol{\Lambda}_c'\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Lambda}_c - \boldsymbol{\Lambda}_c'\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Omega}_{\ell,c}\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Lambda}_c)^{-1}. \tag{28}$$

Then, each case in the training set is processed. For case $\mathbf{x}_t$, $P(c,\ell|\mathbf{x}_t)$ is first computed for each combination of c, ℓ, as described in Scn. III. Then, the following expectations are computed:

$$\mathrm{E}[\mathbf{y}|\mathbf{x}_t,\ell,c] = \boldsymbol{\beta}_{\ell,c}\boldsymbol{\Lambda}_c'\boldsymbol{\Phi}_c^{-1}[\boldsymbol{\Omega}_{\ell,c}\mathbf{G}_\ell'\boldsymbol{\Psi}^{-1}\mathbf{x}_t - (\mathbf{I}-\boldsymbol{\Omega}_{\ell,c}\boldsymbol{\Phi}_c^{-1})\boldsymbol{\mu}_c],$$

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

$$\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c)\circ(\mathbf{z}-\boldsymbol{\mu}_c)|\mathbf{x}_t,\ell,c] = (\mathrm{E}[\mathbf{z}|\mathbf{x}_t,\ell,c]-\boldsymbol{\mu}_c)\circ(\mathrm{E}[\mathbf{z}|\mathbf{x}_t,\ell,c]-\boldsymbol{\mu}_c)$$
$$+\mathrm{diag}(\boldsymbol{\Omega}_{\ell,c})+\mathrm{diag}(\boldsymbol{\Omega}_{\ell,c}\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Lambda}_c\boldsymbol{\beta}_{\ell,c}\boldsymbol{\Lambda}_c'\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Omega}_{\ell,c}),$$

$$\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c)\mathbf{y}'|\mathbf{x}_t,\ell,c] = (\mathrm{E}[\mathbf{z}|\mathbf{x}_t,\ell,c]-\boldsymbol{\mu}_c)\mathrm{E}[\mathbf{y}|\mathbf{x}_t,\ell,c]'$$
$$+\boldsymbol{\Omega}_{\ell,c}\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Lambda}_c\boldsymbol{\beta}_{\ell,c}. \qquad (29)$$

The expectations needed to accumulate the sufficient statistics in (24)-(27) are then computed from

$$P(c|\mathbf{x}_t)\mathrm{E}[\mathbf{z}-\boldsymbol{\Lambda}_c\mathbf{y}|\mathbf{x}_t,c]$$
$$= \sum_\ell P(c,\ell|\mathbf{x}_t)(\mathrm{E}[\mathbf{z}|\mathbf{x}_t,\ell,c] - \boldsymbol{\Lambda}_c\mathrm{E}[\mathbf{y}|\mathbf{x}_t,\ell,c]),$$

$$P(c|\mathbf{x}_t)\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c-\boldsymbol{\Lambda}_c\mathbf{y})\circ(\mathbf{z}-\boldsymbol{\mu}_c-\boldsymbol{\Lambda}_c\mathbf{y})|\mathbf{x}_t,c]$$
$$= \sum_\ell P(c,\ell|\mathbf{x}_t)\big\{\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c)\circ(\mathbf{z}-\boldsymbol{\mu}_c)|\mathbf{x}_t,\ell,c]$$
$$+\mathrm{diag}(\boldsymbol{\Lambda}_c\boldsymbol{\beta}_{\ell,c}\boldsymbol{\Lambda}_c') - 2\mathrm{diag}(\boldsymbol{\Lambda}_c\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu}_c)\mathbf{y}'|\mathbf{x}_t,\ell,c]')$$
$$+(\boldsymbol{\Lambda}_c\mathrm{E}[\mathbf{y}|\mathbf{x}_t,\ell,c])\circ(\boldsymbol{\Lambda}_c\mathrm{E}[\mathbf{y}|\mathbf{x}_t,\ell,c])\big\},$$

$$\mathrm{E}[(\mathbf{x}_t-\mathbf{G}_\ell\mathbf{z})\circ(\mathbf{x}_t-\mathbf{G}_\ell\mathbf{z})|\mathbf{x}_t]$$
$$= \sum_{c,\ell} P(c,\ell|\mathbf{x}_t)\big\{(\mathbf{x}_t-\mathbf{G}_\ell\mathrm{E}[\mathbf{z}|\mathbf{x}_t,\ell,c])\circ(\mathbf{x}_t-\mathbf{G}_\ell\mathrm{E}[\mathbf{z}|\mathbf{x}_t,\ell,c])$$
$$+\mathrm{diag}(\mathbf{G}_\ell\boldsymbol{\Omega}_{\ell,c}\mathbf{G}_\ell') + \mathrm{diag}(\mathbf{G}_\ell\boldsymbol{\Omega}_{\ell,c}\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Lambda}_c\boldsymbol{\beta}_{\ell,c}\boldsymbol{\Lambda}_c'\boldsymbol{\Phi}_c^{-1}\boldsymbol{\Omega}_{\ell,c}\mathbf{G}_\ell')\big\},$$

$$P(c|\mathbf{x}_t)\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu})\mathbf{y}'|\mathbf{x}_t,c]$$
$$= \sum_\ell P(c,\ell|\mathbf{x}_t)\mathrm{E}[(\mathbf{z}-\boldsymbol{\mu})\mathbf{y}'|\mathbf{x}_t,\ell,c],$$

$$P(c|\mathbf{x}_t)\mathrm{E}[\mathbf{y}\mathbf{y}'|\mathbf{x}_t,c] = \sum_\ell P(c,\ell|\mathbf{x}_t)\boldsymbol{\beta}_{\ell,c}$$
$$+\sum_\ell P(c,\ell|\mathbf{x}_t)\mathrm{E}[\mathbf{y}|\mathbf{x}_t,\ell,c]\mathrm{E}[\mathbf{y}|\mathbf{x}_t,\ell,c]'. \qquad (30)$$

C. *Learning models that incorporate the linear approximation to the transformation manifold*

A simple approach is to treat $\boldsymbol{\delta}_x(\boldsymbol{\mu}_c)$ and $\boldsymbol{\delta}_y(\boldsymbol{\mu}_c)$ as constants while updating the $\boldsymbol{\mu}_c$'s as described above. After the $\boldsymbol{\mu}_c$'s are updated, we can use the new values of $\boldsymbol{\delta}_x(\boldsymbol{\mu}_c)$ and $\boldsymbol{\delta}_y(\boldsymbol{\mu}_c)$ to compute the sufficient statistics.

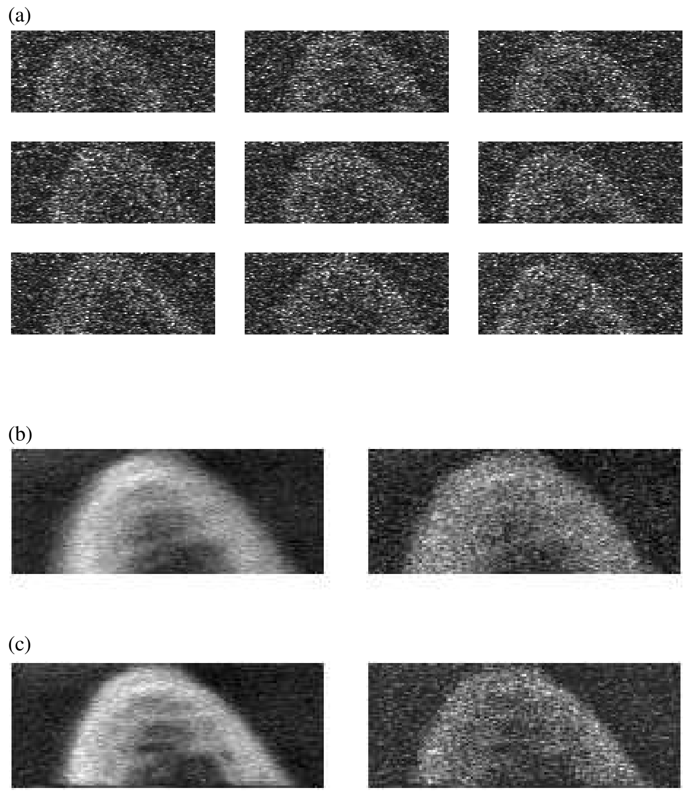

Fig. 7. (a) 140×56 pixel SEM images. (b) The mean and variance of the image pixels. (c) The mean and variance found by a TMG reveal more structure and less uncertainty.

V. Experiments

A. *Filtering images from a scanning electron microscope (SEM).*

SEM images (*e.g.*, Fig. 7a) can have a very low signal to noise ratio due to a high variance in electron emission rate and modulation of this variance by the imaged material [1]. To reduce noise, multiple images are usually averaged and the pixel variances can be used to estimate certainty in rendered structures. Fig. 7b shows the estimated means and variances of the pixels from 230 140×56 SEM images like the ones in Fig. 7a. In fact, averaging images does not take into account spatial uncertainties and filtering in the imaging process introduced by the electron detectors and the high-speed electrical circuits.

We trained a single-cluster TMG with 5 horizontal shifts and 5 vertical shifts on the 230 SEM images using 30 iterations of EM. To keep the number of parameters almost equal to the number of parameters estimated using simple averaging, the transformation probabilities were not learned and the pixel variances in the observed image were set equal after each M step. So, TMG had 1 more parameter. Fig. 7c shows the mean and variance learned by the TMG. Compared to simple averaging, the TMG finds sharper, more detailed structure. The variances are significantly lower, indicating that the TMG produces a more confident estimate of the image.

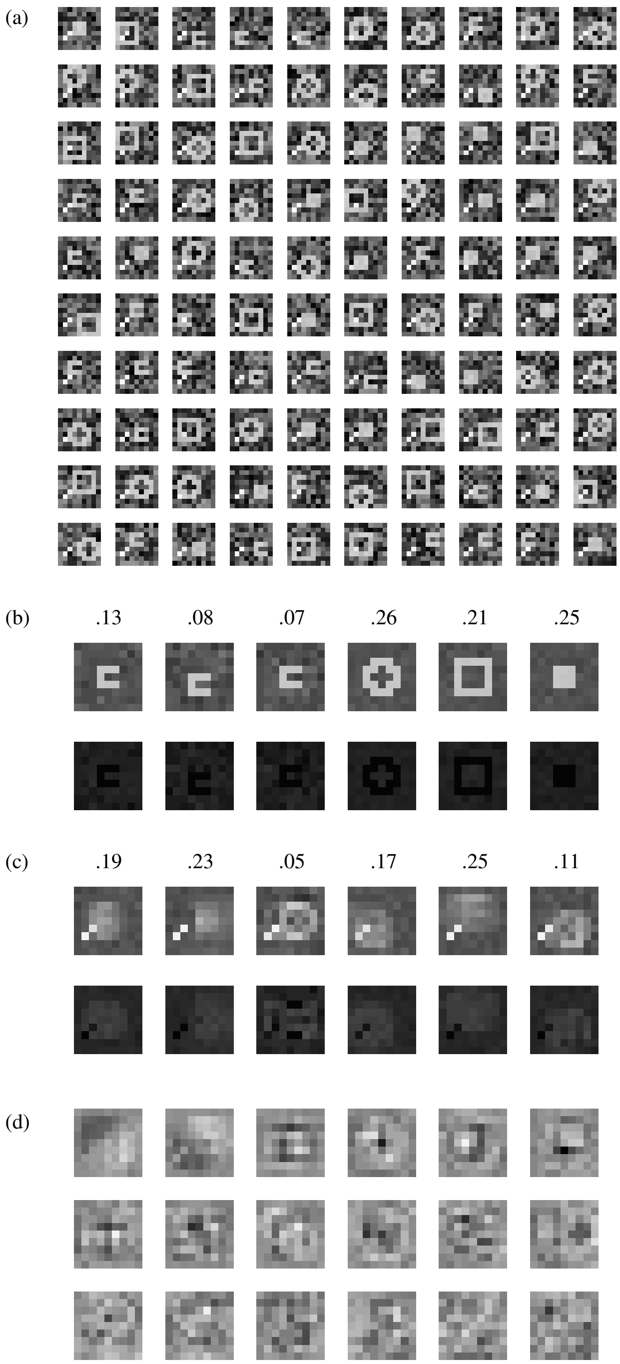

Fig. 8. Extracting transformation invariant structure from synthetic data using a TMG. (a) Training examples, which include background clutter and a fixed distraction. (b) Means and variances for a 6-cluster TMG. (c) Means and variances for a 6-cluster Gaussian mixture model. (d) 18 principal components (eigenimages).

B. Extracting clusters from synthetic data.

Fig. 8a 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.

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. 8b 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. 8c shows the parameters learned using 20 iterations of the EM algorithm for a traditional mixture model with 6 cluster. 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. 8d shows the first 18 principal components, or "eigenimages" [17, 18], of the training data. It is difficult to imagine how these components can be used to reconstruct the data accurately.

C. Clustering faces and facial poses.

Fig. 9a shows examples from a training set of 400 jerky images of two people walking across a cluttered background. We trained a TMG with 4 clusters, 11 horizontal shifts and 11 vertical shifts using 15 iterations of EM after initializing the weights to small, random values. The loop-rich MATLAB script executed in 40 minutes on a 500MHz Pentium processor. Fig. 9b shows the cluster means, which include two sharp representations of each person's face, with the background clutter suppressed. Fig. 9c shows the much blurrier means for a mixture of Gaussians trained using 15 iterations of EM.

Fig. 10a shows examples from a training set of 400 jerky images of one person with different poses. We trained a TMG with 5 clusters, 11 horizontal shifts and 11 vertical shifts using 40 iterations of EM. Fig. 10b shows the cluster means, which capture 4 poses and mostly suppress the background clutter. The mean for cluster 4 includes part of the background, but this cluster also has a low mixing proportion of 0.1. A traditional mixture of Gaussians trained using 40 iterations of EM finds blurrier means, as shown in Fig. 10c. The first 4 principal components mostly try to account for lighting and translation, as shown in Fig. 10d.

D. Learning shape and lighting representations from noisy unaligned images of an object.

Fig. 11a shows a training set of 144 9×9 noisy images of a uniformly colored pyramid (gray) at randomly selected positions and illuminated by parallel light rays with randomly selected angle and intensity. A cluttered background was simu-

Fig. 9. (a) Frontal face images of two people. (b) Cluster means learned by a TMG and (c) a mixture of Gaussians.

lated by randomly selecting pixel values from a uniform distribution.

The first 8 principal components of the training data, scaled by the standard deviation of the projected data, are shown in Fig. 11b. It appears the components implement a multiresolution approximation to model shifts of the object.

We trained a TCA with 3 components and 81 transformations implementing 9 horizontal and 9 vertical shifts using 10 iterations of the EM algorithm. To initialize the parameters, the mean and variance of each pixel was first computed from the training data. The parameters were then initialized to random values, using the mean and variance as a 1st order guide. The transformation probabilities p_ℓ were set equal.

Fig. 11c shows the mean latent image $\boldsymbol{\mu}$, the 3 columns of $\boldsymbol{\Lambda}$ (shown as 3 images), the latent image noise $\boldsymbol{\Phi}$ (shown as an image where the pixel intensity is equal to 4 times the standard deviation) and the observed image noise $\boldsymbol{\Psi}$. The mean clearly shows that the outline of the object has been determined and that the uniform coloring has been determined (except at the point

Fig. 10. (a) Images of one person with different poses. (b) Cluster means learned by a TMG. (c) Less detailed cluster means learned by a mixture of Gaussians. (d) Mean and first 4 principal components of the data, which mostly model lighting and translation.

of the pyramid). Linear combinations of the 3 components produce different lighting conditions (see the following paragraph) which implies that the 3-element *rows* of $\boldsymbol{\Lambda}$ are proportional to the object surface normals, up to some rotation in 3-dimensional space. The variance map for the latent image shows that the model predicts low variance for pixels belonging to the object, but high variance for other pixels (the background clutter). Finally, the variance map for the observed image accounts for the small amount of noise that is present in the images.

The TCA can be simulated in a noise-free, transformation-free fashion, by drawing a subspace representation from $p(\mathbf{y})$, computing $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\Lambda}\mathbf{y}$ and then computing $\mathbf{x} = \mathbf{G}_1\mathbf{z}$. Fig. 11d shows 144 examples simulated in this way. These "fantasies" show that the TCA can simulate the different lighting conditions.

E. Learning a subspace representation of facial expressions from imperfectly aligned images.

Fig. 12 shows a training set of 100 16×24 images of automatically aligned faces with different expressions. The accuracy of the face detection algorithm used to align the images is +/-2 pixels in each direction.

(a)
(b)
(c)
(d)

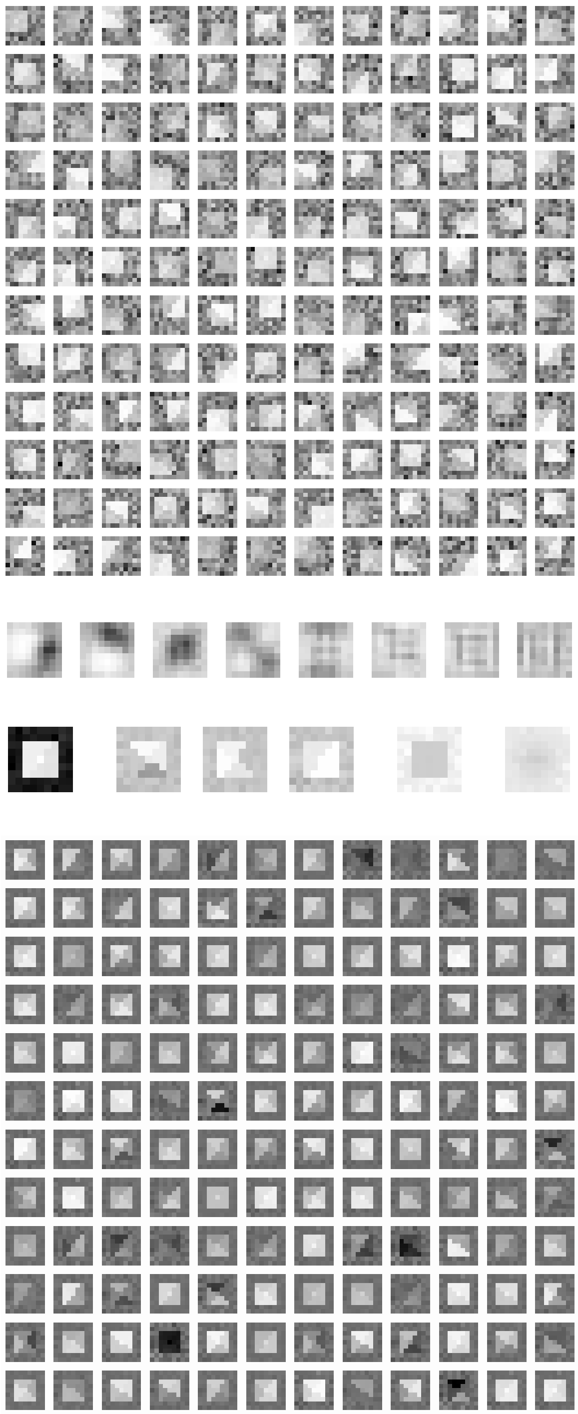

Fig. 11. (a) Noisy images of a pyramid at different locations and under different lighting conditions. (b) The first 8 scaled principal components. A pixel colored gray – halfway between black and white – corresponds to a component value of 0. (c) The mean, components, and noise deviation of a TCA with 3 components, after 10 iterations of EM. Pixels for the mean and the noise deviations are colored using the same scale as the training images. (d) Examples simulated from the TCA, without noise and without transformations.

Fig. 13a shows the mean of the training data and the first 10 principal components, scaled by the standard deviation of the projected data. The first 5 components obviously account for vertical, horizontal and diagonal shifts in the data and the remaining components are very blurred.

Fig. 12. Imperfectly aligned images of faces with different expressions.

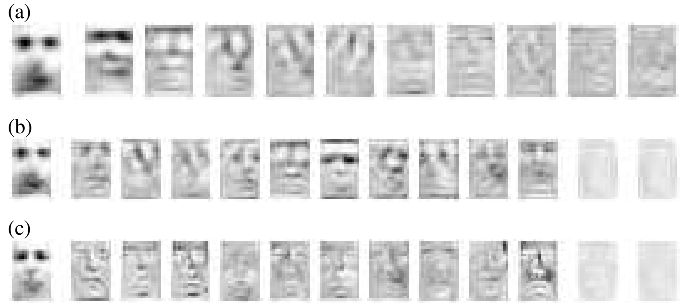

Fig. 13. (a) The mean and first 10 scaled principal components of the face data. (b) The mean, 10 components and noise deviations found by FA (TCA with only the identity transformation). (c) The mean, 10 components and noise deviations found by TCA.

Fig. 13b shows the parameters for a FA model (a TCA with only the identity transformation) trained using 70 iterations of EM. The parameters were initialized using the mean and variance of each pixel in the training data. The sum of the two images on the far right of Fig. 13b gives the variance map for FA. In contrast to PCA, different components represent similar amounts of energy (variance). This is because FA does not find a preferred set of basis vectors (factors) for the subspace. Like PCA, FA finds very blurred components.

We trained a TCA with 10 components and 25 transformations implementing 5 horizontal and 5 vertical shifts using 70

(a)

(b)

Fig. 14. Examples of the face data simulated using (a) the PCA subspace and (b) the factor analyzer.

iterations of the EM algorithm. The parameters were initialized

Fig. 15. Examples of the face data simulated using the TCA model.

using the mean and variance of each pixel in the training data. The transformation probabilities p_ℓ were set equal.

Fig. 13c shows the mean, components and variance maps. Unlike PCA and FA, TCA extracts clear components. The first component appears to expose some teeth, the second component appears to raise the eyebrows, raise the upper lip and expose a "tongue", and so on. The components found by TCA are unique up to a unitary transformation, so each component often includes more than one feature. A further processing step can be applied to find a unitary transformation that produces components with spatially localized energy.

To see how well the PCA subspace represents the data, we can draw a subspace point from an axis-aligned Gaussian with variances determined from the projected training data, and then use the principal components to map the point to image space. 100 examples simulated in this manner are shown in Fig. 14a. Although the "faces" do appear to be shifted around the field of vision, they are also severely blurred. Fig. 14b shows examples simulated using the FA model, without adding sensor noise. They appear similar to the examples simulated using the PCA model.

Fig. 15 shows 100 examples of images simulated using the TCA, without the latent image and observed image noise and without randomly selected transformations. The images are much clearer than those simulated using the PCA subspace and the factor analyzer. The expressions in the training set are repro-

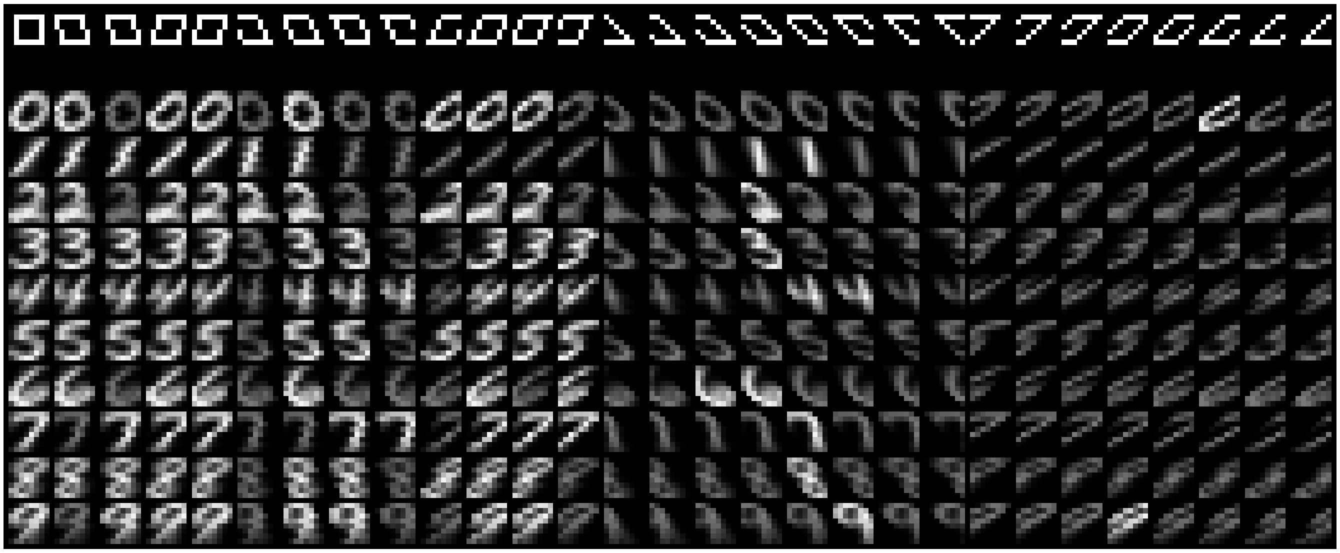

Fig. 16. The means (left column of images) and their sheared and translated versions (remaining columns) found by training 10 TCA models on 10 sets of handwritten digits, containing 200 examples each. The top row of pictures illustrates how a test pattern is deformed by the translation/shearing transformations. Those translation/shearing transformations that have low probability $\rho_{\ell c}$ after learning are dimmed.

duced and the model also generates novel realistic expressions that are not present in the training set, such as the one in the 5th column of the 1st row and the one in the 1st column of the 3rd row.

F. Modeling handwritten digits.

We performed both supervised and unsupervised learning experiments on 8×8 greyscale versions of 2000 digits from the CEDAR CDROM (Hull, 1994). Although the preprocessed images fit snugly in the 8×8 window, there is wide variation in "writing angle" (*e.g.*, the vertical stroke of the 7 is at different angles). So, we produced a set of 29 shearing+translation transformations (see the top row of Fig. 16) to use in transformed density models.

In our supervised learning experiments, we trained one 10-component TCA on each class of digit using 30 iterations of EM. Fig. 17 shows the mean and 10 components for each of the 10 models. The lower 10 rows of images in Fig. 16 show the sheared and translated means. In cases where the transformation probability is below 1%, the image is dimmed. We also trained one 10-component factor analyzer on each class of digit using 30 iterations of EM. The means and components are shown in Fig. 18. The means found by TCA are sharper and whereas the components found by factor analysis often account for writing angle (*e.g.*, see the components for 7) the components found by TCA tend to account for line thickness and arc size. Fig. 19 shows images that were randomly generated from the TCA models and Fig. 20 shows images that were randomly generated from the factor analyzer models. Since different components in the factor analyzers account for different stroke angles, the simulated digits often have an extra stroke, whereas digits simulated from the TCAs contain fewer spurious strokes.

To test recognition performance, we trained 20-component factor analyzers and TCAs on 200 examples of each digit using 50 iterations of EM. The noise variances were not allowed to drop below 10^{-2} to prevent overfitting a pixel that happens to always be off in the training data. Each set of models used Bayes rule to classify 1000 test patterns. The results are summarized in Table I and are compared with a standard feedforward method, k-nearest neighbors, where k was chosen using leave-one-out cross validation. TCA has a lower error rate than the other two methods.

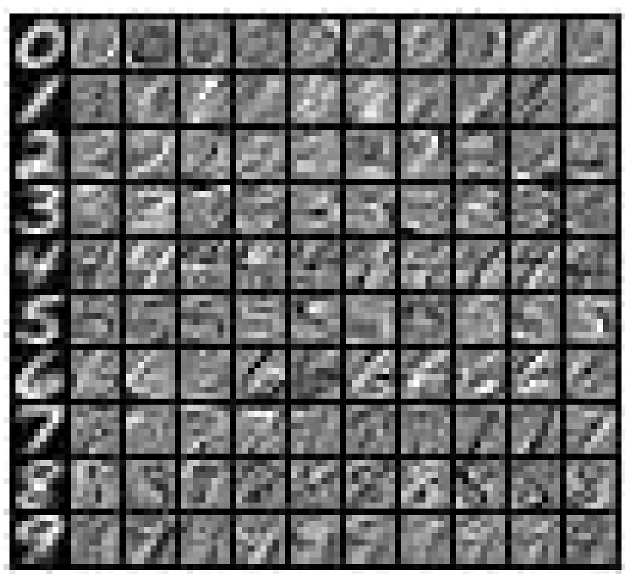

Fig. 17. The means (left column of images) and 10 components (remaining 10 columns) for the model from Fig. 16.

The probability of each transformation p_ℓ in TCA was learned and we believe there was some overfitting. For example, some of the sheared image means in Fig. 16 that are faded (have average responsibilities less than 0.01) are good generalizations. The p_ℓ's can be regularized quite easily, by blurring them after

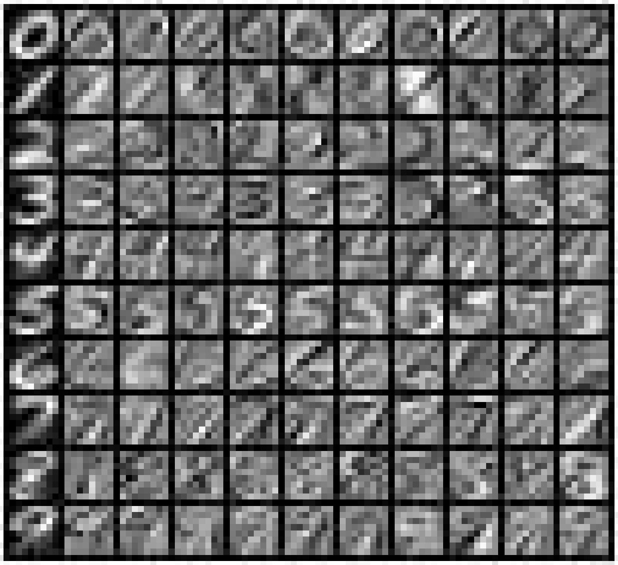

Fig. 18. The means (left column of images) and 10 components (remaining 10 columns) found by training 10 factor analysis models on the same 10 sets of data as were used to train the TCA models above. Factor analysis produces blurrier means than TCA.

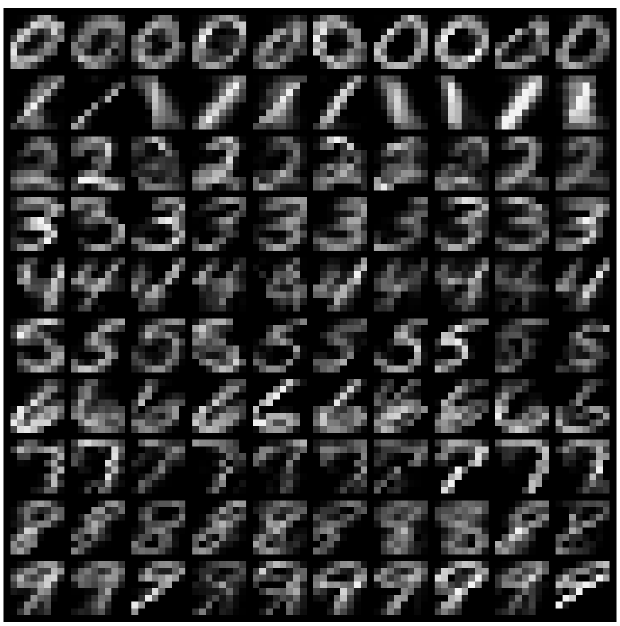

Fig. 19. Digits randomly generated from the 10 TCA models.

each M-Step, using the appropriate topology. For example, if the transformations are 1-D translation, 1-D blurring is applied; if the transformations are 2-D translations, 2-D blurring is applied; *etc.*

In our unsupervised learning experiments, we fit 10-cluster mixture models to the entire set of 2000 digits to see which models could identify all 10 digits. We tried a mixture of 10 Gaussians, a mixture of 10 factor analyzers and a 10-cluster TMG. In each case, 10 models were trained using 100 iterations of EM and the model with the highest likelihood was selected and is shown in Fig. 21. Compared to the TMG, the first two methods found blurred and repeated classes. After identifying each cluster with its most prevalent class of digit, we found that the first two methods had error rates of 53% and 49%, but the TMG had a much lower error rate of 26%.

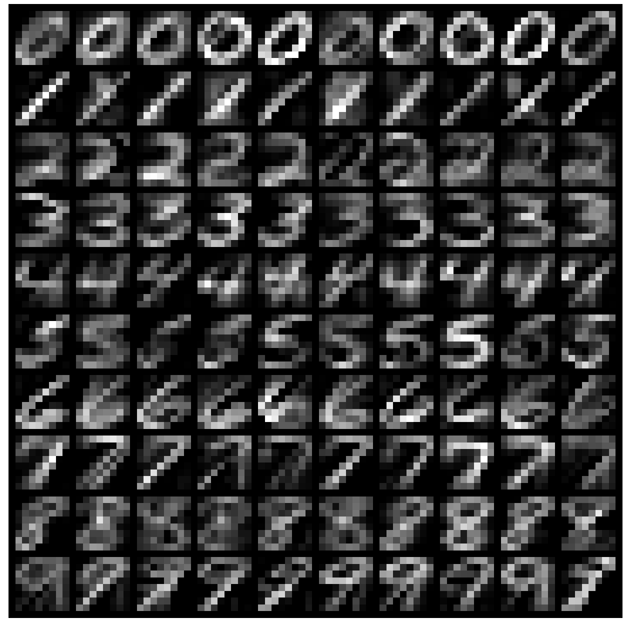

Fig. 20. Digits randomly generated from the 10 factor analysis models.

TABLE I

Handwritten digit recognition rates, for a training set of 2000 8×8 images and a test set of 1000 images.

Method	Error rate
k-nearest neighbors	7.6%
Factor analysis	3.2%
Transformed component analysis	2.7%

VI. Summary

In many learning applications, we know beforehand that the data includes transformations of an easily specified nature (*e.g.*, shearing in images of handwritten digits). If a density model is learned directly from the data, the model must account for both the transformations in the data and the more interesting and potentially useful structure.

We introduce a way to make standard density models for clustering and linear dimensionality reduction invariant to local and global transformations in the input. The result is a latent variable model containing continuous and discrete variables. Given the discrete variables, the distribution over the other variables is jointly Gaussian, so inference and estimation (via the expectation maximization algorithm) is efficient.

The algorithms are able to jointly normalize input data for transformations (*e.g.*, translation, shearing and rotation in images) and learn models of the normalized data.

We illustrate the algorithms on a variety of difficult tasks. For example, the transformation-invariant mixture of Gaussians is able to learn different facial poses from a set of out-

Fig. 21. Clustering handwritten digits. Three different methods were used to cluster a training set of 2000 cases containing digits from *all* classes. (a) The means found by a mixture of 10 Gaussians. (b) The means found by a mixture of 10 factor analyzers. (c) The means found by a transformed mixture of 10 Gaussians. In each case, 9 models were learned and the model with the highest likelihood on the training set was selected.

door images showing a person walking across a cluttered background with varying lighting conditions. MATLAB scripts for transformation-invariant clustering and dimensionality reduction are available on our webpage, www.cs.toronto.edu/∼frey.

We focus on translational transformations in this paper, but other types of transformation can be used, such as rotation, scale, out-of-plane rotation and warping in images. Other domains may have quite different types of transformation.

In the case of time-series data, the transformations at the neighboring time steps influence which transformations are likely in the current time step. In [8–10], we show how the techniques presented here can be extended to time series.

The number of computations needed for exact inference scales exponentially with the dimensionality of the transformation manifold. If there are n_1 transformations of the first type, n_2 transformations of the second type, and so on, exact inference and learning takes time that scales as $\prod_i n_i$. We are currently exploring the performance of a faster, variational inference and learning method that takes time that scales as $\sum_i n_i n_{i+1}$. This exponential speedup is achieved by inferring different types of transformation index separately. Inference of a particular index is coupled to inference of the other indices using variational parameters that represent average influences of other indices.

We believe the approach presented here will prove to be useful for applications that require transformation-invariant clustering and dimensionality reduction.

REFERENCES

[1] R. Golem and I. Cohen, "Scanning electron microscope image enhancement," Technical Report Chool of Computer and Electrical Engineering Project Report, Ben-Gurion University, 1998.

[2] J. J. Hull, "A database for handwritten text recognition research," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, pp. 550–554, 1994.

[3] B. J. Frey and N. Jojic, "Estimating mixture models of images and inferring spatial transformations using the EM algorithm," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1999, pp. 416–422.

[4] B. J. Frey and N. Jojic, "Transformed component analysis: Joint estimation of spatial transformations and image components," in *Proceedings of the IEEE International Conference on Computer Vision*, September 1999.

[5] 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.

[6] 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.

[7] 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.

[8] N. Jojic, N. Petrovic, B. J. Frey, and T. S. Huang, "Transformed hidden markov models: Estimating mixture models of images and inferring spatial transformations in video sequences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2000.

[9] B. J. Frey and N. Jojic, "Transformation-invariant filtering using expectation maximization," in *Proceedings of the IEEE Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control*, 2000.

[10] N. Jojic and B. J. Frey, "Video summarization and filtering using transformation-invariant hidden markov models," Submitted to *International Journal on Computer Vision*, 2001.

[11] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Wadsworth, Blemont CA., 1984.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

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

[14] 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.

[15] R. M. Neal, "Regression and classification using gaussian process priors," in *Bayesian Statistics 6*, J. M. Bernardo *et al.*, Ed., pp. 475–501. Oxford University Press, 1998.

[16] V. Vapnik, *Statistical Learning Theory*, John Wiley, New York NY., 1998.

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

[18] 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.

[19] C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press Inc., New York NY., 1995.

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

[21] P. Y. Simard, Y. Le Cun, and J. Denker, "Efficient pattern recognition using a new transformation distance," in *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds. Morgan Kaufmann, San Mateo CA., 1993.

[22] P. Y. Simard, B. Victorri, Y. Le Cun, and J. Denker, "Tangent prop – a formalism for specifying selected invariances in an adaptive network," in *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, San Mateo CA., 1992.

[23] G. E. Hinton, P. Dayan, and M. Revow, "Modeling the manifolds of images of handwritten digits," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 65–74, 1997.

[24] N. Vasconcelos and A. Lippman, "Multiresolution tangent distance for affine-invariant classification," in *Advances in Neural Information Processing Systems 10*, M. I. Jordan, M. I. Kearns, and S. A. Solla, Eds. MIT Press, Cambridge MA., 1998.

[25] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[26] Z. Ghahramani and G. E. Hinton, "The EM algorithm for mixtures of factor analyzers," University of Toronto Technical Report CRG-TR-96-1, 1997.

[27] B. J. Frey and N. Jojic, "Estimating mixture models of images and inferring spatial transformations using the EM algorithm," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1999, pp. 416–422.

[28] N. Jojic and B. J. Frey, "Topographic transformation as a discrete latent variable," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K.-R. Müller, Eds. MIT Press, Cambridge MA., 2000.

[29] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*, MIT Press, Cambridge MA., 1998, See `http://www.cs.utoronto.ca/∼frey`.