Modelling the Manifolds of Images of Handwritten Digits

Geoffrey E. Hinton (hinton@cs.toronto.edu)
Peter Dayan[1] (dayan@ai.mit.edu)
Michael Revow (revow@cs.toronto.edu)

Department of Computer Science
University of Toronto
6 Kings College Road
Toronto, Ontario Canada M5S 3H5
Phone (416) 978 7453 Fax (416) 978 1525

Keywords: principal components, factor analysis,
autoencoder, minimum description length,
density estimation.

IEEE trans. on Neural Networks 8(1) 65-74, 1997.

[1] Present Address: Center for Biological and Computational Learning, Department of Brain and Cognitive Sciences, E25-229, MIT, Cambridge, MA 02139, phone (617) 252 1693, fax (617) 253 2964

Abstract

This paper describes two new methods for modelling the manifolds of digitised images of handwritten digits. The models allow *a priori* information about the structure of the manifolds to be combined with empirical data. Accurate modelling of the manifolds allows digits to be discriminated using the relative probability densities under the alternative models. One of the methods is grounded in principal components analysis, the other in factor analysis. Both methods are based on locally linear, low-dimensional approximations to the underlying data manifold. Links with other methods that model the manifold are discussed.

I Introduction

A standard *discriminative* approach to digit recognition is to train a classifier to output one of the ten classes based on the input image. The classifier could, for instance, be a multilayer feedforward neural network [1]. However, it is also possible to discriminate by fitting a separate probability density model to each class and then picking the class of the model that assigns the highest density to a test image. This *relative density* approach typically requires more computation during recognition, and it can devote a lot of parameters to modelling aspects of the image that are irrelevant for discrimination, but it has several advantages:

- During training, each model need only consider training examples of its own class, saving an order of magnitude in computation. If the models are correct, this saving is achieved with no reduction in discriminative performance.
- After training, it is possible to add a new class without retraining the previous class models.
- It is possible to fit far more parameters before overfitting occurs because the input vectors contain much more information than the class label. Assuming the same number of examples in each class, a class label only contains $\log_2 10$ bits so each example only provides 3.3 bits of constraint on the function that maps inputs to class labels[1]. However, it takes many more bits to specify the input image, so each example provides far more constraint on the parameters

[1] One way to see why, is to imagine a table look-up scheme in which each input vector is randomly mapped to a different word of memory. To fit a training set perfectly, we need as many words of memory as training examples and each word only needs enough bits to specify the correct label.

of a density model. For the 8×8 real-valued images we use, $7,000$ training examples are sufficient to fit about $70,400$ parameters.

- Aspects of the image that are irrelevant for discrimination between classes may nevertheless be very relevant for detecting occasional bad images that do not fall into any of the given classes. Relative density methods have a natural rejection criterion when *all* the densities are low.
- The density models we describe can be fitted by methods like singular value decomposition (SVD) and expectation-maximisation (EM) that are considerably more efficient than gradient descent.

We are not claiming that the relative density approach is necessarily better than the discriminative approach, just that it is an alternative worth considering.

Certain discriminative methods can be seen in terms of relative densities. For instance, kernel density estimation [2], [3] is a popular non-parametric modelling technique. For this, the probability density for a particular digit is the weighted sum of a collection of kernel functions. The functions all have the same shape, but each is centred on one of the patterns in that class in the training set. Each kernel function typically integrates to 1, and the weights in the sum are usually $1/M$, where M is the number of the patterns in the training set, so the overall kernel density estimate is correctly normalised. Having built ten such models, one for each digit class, the class to which a new image belongs is inferred by evaluating the density under each of the models at the location of the new image, and reporting the one that is the highest. If the kernel functions are radially symmetric, monotonically decreasing, and have unbounded extent (e.g a Gaussian), then relative density estimation becomes identical to nearest neighbour classification as the width parameter of the kernel goes to zero.

This is because the model that contains the kernel function closest to the data will, in the limit, have infinitely higher density than any other model, even if other models have many kernel functions that are almost as close.

Kernel density estimators are convenient models in the case that there is ample data, ample computational time for inference (it is effectively a memory-based technique), and in which there is little *a priori* information about the nature of the data. The latter two are not true for images of handwritten digits. We sought to build better models for such images on the basis of such information. Elastically deformable templates [4], [5] are one example, and have been shown to model non-normalised images of characters well [6]. Unfortunately, they are also computationally too expensive for normal use. We therefore turned to Gaussian blended linear models, which are computationally much cheaper but are also appropriate for such images.

Simard *et al* [7] pointed out the locally low-dimensional linear structure underlying these images. Take the 64-dimensional space of all (normalised) 8×8 images and consider the subset of images of any particular digit, say the digit 2. Since small changes to the image of a 2 preserve its identity, this subset will have some properties of a surface – it will mostly be continuous and differentiable. In particular, affine transformations (translations, rotations, scalings, and shearing) as well as manipulations to the thickness of the strokes of a digit preserve its identity. Considering the effect on an image of small (*ie* sub-pixel) transformations like these suggests that the surface is locally at least 7 dimensional and probably somewhat more. Different styles of 2 will likely generate separated continuous patches. Simard *et al* [7] used a nearest neighbour method (which, as pointed out, is equivalent to a limiting case of a relative density method) in the space of these 7 dimensional planes, where the distance between two points in the space is the closest distance between the underlying planes in image space. Using this as the distance rather than the simple Euclidean distance between two images substantially improved recognition performance.[2]

Unfortunately, Simard's technique is founded on a nearest neighbour method, and therefore recognition is again computationally expensive[3]. We can build much cheaper models by combining information from many examples about the local structure of the manifold in image space representing a digit. Combining information in this way should have the added advantage of averaging out some of the noise in the estimate of its local structure that arises from the gradient operators that Simard used to extract the 7 dimensional subspace. We are also not limited to only the 7 *a priori* dimensions listed above, but learn the local structure from the training examples, which may contain other invariances. Although the manifold seems to be locally linear and low dimensional, there is no reason for the manifold to be globally linear. Different styles of digits, and even affine transformations that are not confined to a sub-pixel regime, will lead to different local linear patches. We were therefore forced to mix together numbers of linear models for images of each digit, *ie* to use a blended linear approximation to the surface (figure 1) [9], [10], [11], [12], [13]. The mixture is fit using either an expectation-maximization (EM) based algorithm [14] or the k-means algorithm, which is actually a limiting case of EM. The expectation (E) phase involves assigning to the linear models responsibilities for the training examples; the maximization (M) phase involves re-estimating the parameters of the linear models in the light of this assignment.

[2] This distance is not a metric since it does not satisfy the triangle inequality.

[3] Significant performance improvements can be achieved by using various speedup mechanisms, for example [8]

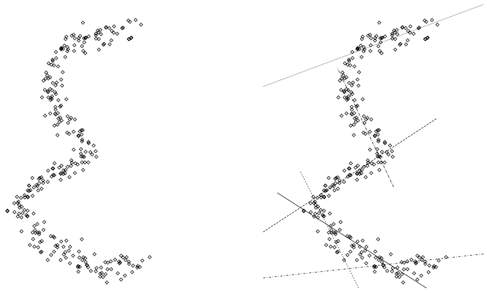


Figure 1: Illustration of how a highly non-linear one-dimensional surface (left panel) is captured by 6 locally linear models (right panel)

A convenient framework to describe our models comes from the version of neural nets called autoencoders. An autoencoder is a feedforward neural network with a single hidden layer that attempts to reconstruct its input activities at its output. Hinton & Zemel [15] and Zemel [16] show how to understand the relationship between statistical modelling and autoencoders. The code for an example is given by the combination of the activations of the hidden units and the output error, which is the information necessary to reconstruct the output given these hidden unit activations and the weights between the hidden units and the output units. The cost of this code is counted in bits, and is based on empirical prior distributions for the activations of the hidden units and the reconstruction errors. The main link between modelling and autoencoders is that this cost function can be considered as the negative log probability of the data under a particular generative model – so maximum likelihood model fitting and minimum cost are equivalent. We give cost functions below which include or exclude various elements of the code cost.

Two well established linear models are principal components analysis (PCA) and factor analysis (FA). Performing PCA requires nothing more than singular value decomposition of the covariance matrix of the examples. Performing FA is computationally more challenging. However, FA offers a sounder statistical model of examples, and one might expect it to be more proficient. Section II describes principal components analysis; section III describes factor analysis; section IV shows how to incorporate some of the tangent information that Simard *et al* [7] use to such good effect; and section V shows how the models perform on a large-scale digit recognition task.

II Mixtures of Principal Component Analysers

The r principal components of N examples $\mathbf{x}^i = \{x_1^i, x_2^i, \ldots x_n^i\}$ of n-dimensional data (assumed, without loss of generality, to have 0 mean) are the r orthogonal directions in n-space which capture the greatest variation in the examples. Put another way, if the examples are projected onto r-dimensional subspaces and the resulting variance of the projected examples is measured, then the principal components define a subspace such that this 'captured' variance is the highest. Alternatively, assuming that the examples come from a multivariate Gaussian distribution, the information conveyed by the magnitude of the projections onto these r directions is the greatest. These properties, together with the computational simplicity of performing PCA (involving no more than finding the top r eigenvectors of the covariance matrix of the examples) make it an obvious candidate for the linear model in the mixture. The r principal components define the local surface assumed for the manifold. In the context of an autoencoder, the projections of the input along the r directions are the activities of the hidden units $\mathbf{h}^i = \{h_1^i, h_2^i, ..h_r^i\}$ for input pattern i. This can be written as:

$$\mathbf{h}^i = R\mathbf{x}^i \tag{1}$$

As discussed above, the activities of the hidden units are part of the code for an example. The activities of the output units are generated by a linear combination of the hidden units:

$$\mathbf{y}^i = G\mathbf{h}^i \tag{2}$$

The other part of the code is the difference between the image itself and these output activities[4]. The resulting squared reconstruction error ($E^i = \|\mathbf{x}^i - \mathbf{y}^i\|^2$) is a measure of how well the model fits the image – the smaller it is, the better the image was captured.

As mentioned above, either an EM or a k-means procedure is used to assign the N examples among m different PCA models (we call these sub-models) for each digit. During the E step, responsibility for each pattern is assigned amongst the sub-models; during the M-step PCA is performed, altering the parameters of a sub-model appropriately to minimise the reconstruction cost of the data for which it is responsible. In the *soft*, EM version, the responsibility of sub-model a for example i is calculated as $q_a^i = e^{-E_a^i/2\sigma^2}/(\sum_b e^{-E_b^i/2\sigma^2})$ where E_b^i is the squared reconstruction error and σ^2 acts like a temperature parameter. In the *hard*, k-means version (which can be viewed as the limiting case as $\sigma^2 \to 0$), example i is allocated to the sub-model a for which E_a^i is smallest. Note that the M step is more complicated and powerful than just taking the means of those examples for which responsibility is taken.

Formally, the k-means version of the algorithm is:

```
1. Choose initial assignments among the sub-models for each example in the
   training set (typically at random, or using samples from the initial data);
```

[4] Strictly speaking there is a third component, the *model-cost*, which encodes the cost of specifying the weights in each model [16]. We assume that this cost is the same for all models.

```
2. Perform PCA separately for each sub-model;

3. Reassign patterns to the sub-model that reconstructs them the best;

4. Stop if no patterns have changed sub-model, otherwise return to step 2.
```

For the soft version, in step 2, the examples are weighted for the PCA by the responsibilities, and convergence is assessed by examining the change in the overall log-likelihood (which is equivalent to negative code-cost using Shannon optimal coding) of the data at each iteration. This log-likelihood is based on a model for the image under which the (incorrectly normalised) log probability of image i is [15]:

$$-\frac{1}{2\sigma^2}\sum_{a=1}^{m} q_a^i E_a^i - \sum_{a=1}^{m} q_a^i \log q_a^i \tag{3}$$

Minus this quantity can be considered as a cost function for learning the assignments of responsibilities and the principal components. The value of σ^2 is somewhat arbitrary.

Altogether, either procedure generates a set of local linear models for each digit. Given a test pattern we evaluate the reconstruction errors against all the models for all the digits. We use a hard method for classification – determining the identity of the pattern only by the model which reconstructs it best. The absolute quality of the best reconstruction and the relative qualities of slightly sub-optimal reconstructions are available to reject ambiguous cases.

Somewhat similar methods have been used for modelling images of lips for lip reading [10], cartoon-like drawings [12], digit and character recognition [13], [17] and data compression [11].

III Mixtures of Factor Analysers

Unfortunately, as noted above, PCA does not provide a correct statistical model for the images because it is not properly normalisable. Components of the image that lie in the directions of the principal components generate no reconstruction error, and therefore can be added at will without changing the cost. Poggio & Sung [12] suggested using as a component of the cost a quantity they called the normalised Mahalanobis distance in the subspace of the principal components. This is:

$$\mathcal{C} - \sum_{s=1}^{r} \frac{|h_{as}^{i}|^{2}}{\lambda_{as}} - \frac{1}{2}\ln\lambda_{as}$$

where $\mathcal{C}$ is a constant, h_{as}^{i} is the activity of hidden unit s for example i of sub-model a and λ_{as} is the eigenvalue associated with that component. This amounts to employing a zero mean hyper-elliptical Gaussian prior aligned in the direction of the principal components. This prior is used to code the activation of the hidden units in the autoencoder. Choosing the relative weighting in the overall cost function for the squared reconstruction error E_a^i and the coding cost is somewhat arbitrary.

Factor analysis (FA) is a different way of analysing the covariance matrix of the inputs (see [18], for an excellent introduction) which starts from a proper probabilistic model, and correctly blends the reconstruction cost and a term playing a similar role to this normalised Mahalanobis distance. FA emerges for the *same* linear autoencoder network if the cost of coding the hidden units is taken into account, using as a prior a specified multivariate Gaussian (often just the identity matrix and traditionally known as the prior for the factor loadings), and if the reconstruction errors are coded according to an elliptical multivariate Gaussian whose axes are aligned with the input dimensions. In terms of the notation introduced above, the standard factor analysis model is written as [18]:

$$\begin{aligned} \mathbf{y}^i &= G\mathbf{h}^i + \xi^i \qquad (4) \\ \mathbf{h}^i &\sim \mathcal{N}(0, \Phi), \quad \xi^i \sim \mathcal{N}(0, \Psi) \end{aligned}$$

The hidden-output weights, G, are the factor loadings and the activities of the hidden units, $\mathbf{h}^i$, are the factors (the prior over which is Gaussian). A key assumption of the FA model is that the observed variables are conditionally independent of each other, given the factors. This is equivalent to noting that the individual components of the residuals, ξ^i_j, are also independent of each other, or that Ψ is a diagonal matrix with variances $\{\sigma_1^2, \ldots, \sigma_n^2\}$ along the diagonal. It is common to take Φ to be the identity matrix. Model (4) implies that the covariance of the observed variables is given by:

$$C(G, \Psi) = GG^T + \Psi \qquad (5)$$

Under the model, the sample covariance matrix (S) follows a Wishart distribution [19] about C and Everitt [18] introduces the function:

$$F(S, C(G, \Psi)) = \ln|C| + \mathrm{trace}(SC^{-1}) - \ln|S| - n \qquad (6)$$

which, up to some constant factors, is the likelihood. Maximum likelihood FA fits the parameters of the model G and Ψ by maximising equation (6). Unfortunately, there is no technique as computationally cheap as singular value decomposition for determining the factors from a collection of images.

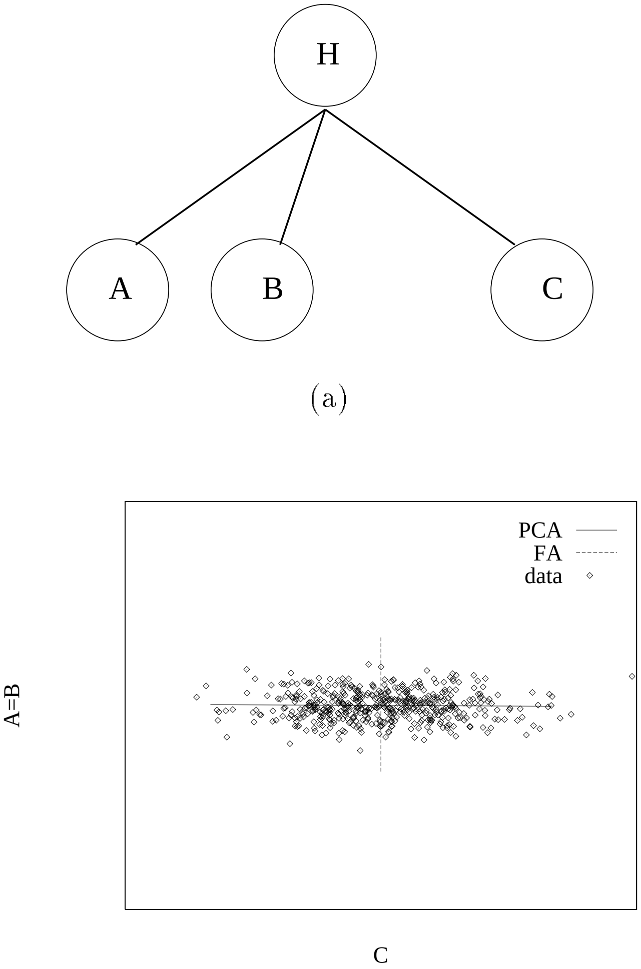
Consider the example in figure 2 in which inputs A and B are perfectly correlated but have low variance and input C is independent but has much higher variance. Consider what happens if we allow only one hidden unit. The principal component will align perfectly with C and be orthogonal to A and B. The factor, however, will align perfectly with A and B and will be orthogonal to C. The difference between the two methods is easy to understand in coding terms. PCA is equivalent to minimizing the description length of the data if we make the following simplifying assumptions:

1. Ignore the cost of communicating the model (*i.e.* the directions of the principal components)
2. Ignore the cost of communicating the projections of each data point onto the principal components (*i.e.* the cost of conveying the activities of the hidden units).
3. Use a Gaussian distribution with the same variance on each dimension as an agreed prior for communicating the residual errors when each dimension of the data point is reconstructed from the projections onto the principal components.

Assumption 2 makes it much cheaper to communicate input C by first copying it to the principal component. But this is only because the cost of communicating this component is ignored. Factor analysis has a somewhat more realistic coding interpretation. It still ignores the cost of communicating the factor loadings, but it takes into account the cost of communicating the projections onto each factor. Given a data point, there is a multivariate posterior probability distribution across the factor values and this distribution is typically not spherical. The cost of communicating the factor values is the Kullback-Leibler divergence between the posterior distribution and the spherical prior distribution for the factor values. Nothing is gained by having a factor that is just a copy of C because communicating this factor value is just as expensive as communicating the full value of C as a residual error.

Assumption 3 is another important difference between the two methods. Factor analysis allows different variances to be used for coding the residual errors on different input dimensions. If the residuals are coded using a fixed variance prior the average information required to convey a residual is linear in its variance. If, however, the variance of the prior is adapted to match the variance of the residual the information is proportional to the log of the variance. This dramatically reduces the cost of high variance residuals as compared with low variance ones and can make it better to decrease two low variance residuals by a little rather than decreasing one high variance residual by a lot.

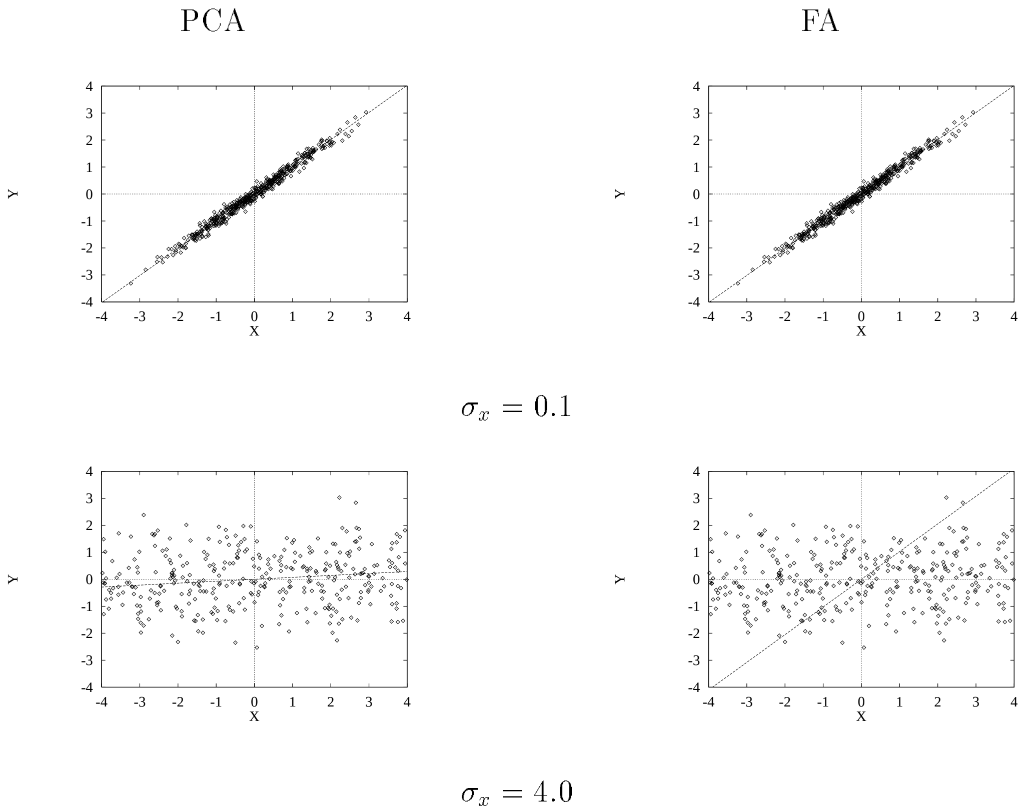
Figure 3 illustrates how the use of different noise models for different input variables can allow factor analysis to extract a more sensible model than PCA. The task in figure 3 is to infer a measured length from 3 noisy measurements. Clearly this is a situation where we wish to extract correlations amongst the inputs. In the example, three dimensional data were generated according to the rule shown, and the results of performing PCA and FA are illustrated by the projection of the generative (hidden-output) weight vector (solid line) onto the $x - y$ plane. (We have shown the 2-dimensional projections, for clarity. Because of the symmetry in the way the data was generated, the $x - z$ projection is similar, while the $y - z$ projection does not illustrate the point being made here.) In the low noise case, PCA and FA extract similar models, the generative weights show that equal attention is paid to both the x and y dimensions. However, in the high noise case, PCA must have a large weight to generate the large variance along the x dimension. On the other hand, FA correctly has recognized that all output dimensions have identical dependencies on the hidden variable, s, and so sets the generational weights accordingly. Of course, the σ_x component in Ψ (equation (4)) is inflated to account for the large variance in this input. This example illustrates how FA can model *covariance* amongst input dimensions separately from variance whereas PCA cannot.



(a)

(b)

Figure 2: (a) Shows three inputs and a single hidden unit. Inputs A and B have low variance but are perfectly correlated. Input C has high variance but is independent of A and B. The hidden unit behaves quite differently in principal component analysis and factor analysis. (b) Scatterplot of the data with the vector of incoming weights of the hidden unit for PCA (solid line) and FA (dotted line).



$\sigma_x = 0.1$

$\sigma_x = 4.0$

Figure 3: Noisy observations of a signal s, $x = s + n_x$, $y = s + n_y$, $z = s + n_z$ where s, n_x, n_y and n_z are normal random variables with zero mean. The standard deviation of s and n_y n_z are held constant at 1.0 and 0.1 respectively on all three panels, while that of n_x is as shown. The projection of 500 training examples onto the $x - y$ plane are shown by the dots, while the solid lines are the projections of hidden-output weight vectors onto this plane. See text for further explanation.

For modelling digits, FA should be immune to the fact that different dimensions might have different intrinsic amounts of noise and look for shared structure in digits. The second difference is that PCA is rotationally symmetric whereas FA is not. For FA, the particular dimensions used to describe the image are special in the sense that the noise corrupting them is taken to be mutually independent. This restriction seems reasonable for images, since the axes defined by the input pixels are indeed privileged.

Following analysis by Neal and Dayan [20] on the relationship between factor analysis, PCA, autoencoders and the Helmholtz machine [21], [22], we actually used an autoencoder to implement factor analysis. The linearity of the model implies that the posterior distribution of the factors $\mathbf{h}^i$ given the input $\mathbf{x}^i$ is Gaussian:

$$\mathbf{h}^i \sim \mathcal{N}(R\mathbf{x}^i, \Sigma) \tag{7}$$

As Rubin & Thayer [23] show in their discussion of the use of EM for FA, the correct values of R and Σ are determined by G and Ψ as:

$$\begin{aligned} R &= (G^T\Psi^{-1}G + \mathcal{I})^{-1}G^T\Psi^{-1} \\ \Sigma^{-1} &= \mathcal{I} + G^T\psi^{-1}G \end{aligned} \tag{8}$$

Following the Helmholtz machine, we call R and Σ parameters of the *recognition* model, since they are responsible for producing the bottom-up receptive fields of the hidden units. Studying the form of these recognition parameters can give insight into the elements of the images that the factors code.

Preliminary experiments with this model suggested that it was prone to overfitting in a very particular manner. Take a pixel on the outskirts of the 8×8 grid. It can easily occur that for some digit the activity of this pixel is always 0 in the *training* set for some sub-model. The factor analysis model might correctly decide that this pixel shares nothing in common with the other pixels and furthermore that its intrinsic noise level is 0. If, in the *test* set, by some quirk of the noise, there is activity in this pixel, then the likelihood of the image under this sub-model will be 0. This is unreasonable, since the pixel is only conveying noise. One way to regularise learning is to impose a minimum allowable intrinsic variance (this is a conventional way of regularising mixtures of Gaussian models); another is to add diagonal terms to the covariance matrix of the examples so it is as if they all suffer from extra independent noise. These regularisation methods were roughly equally efficacious, and we adopt the latter for the empirical studies below.

The full non-linear model uses a mixture of local factor analysers in the same way that the non-linear PCA model used a mixture of principal component analysers. The same EM-based method can be used to fit the combined model, with the E-phase assigning responsibilities for images to the factor analysers in a hard or soft manner, and the M-phase adapting the generative model within in analyser according to the new covariance matrix of the data for which it takes responsibility. Because factor analysis is a genuine generative model we avoid the arbitrary choice of σ^2 that is required to apply EM to a mixture of PCA models.

IV Tangent Information

Nearest neighbour methods offer simple, non-parametric, ways of discriminating between the digits. However, the metric that is used to judge proximity can make a substantial difference to the quality of the resulting inference. There are discriminative and maximum likelihood ways to look at this issue. For instance, Hastie and Tibshirani [24] choose a metric for the nearest neighbours at a point based on information from local linear discriminant analysis – emphasizing directions in which the images from the different classes differ and downplaying directions in which they are similar.

On the other hand, Simard *et al* [7], in the method described in the introduction, used an approach that owes more to modelling the local structure of the classes. Their approach is based on prior information, known to be valid for the particular task of digit modelling, that certain local manipulations of an image preserve the identity of the digit that is described. Effectively, each point is replaced by a local low dimensional linear manifold, deformations in the direction of which incur no cost. Equivalently, each image is modelled as a local linear surface. Note that these local surfaces are chosen to *model* the images of each digit as best as possible, and not to support the best possible *discrimination* between them. This local low-dimensional and linear behaviour is what motivated our linear models. Schwenk and Milgram [17], [25] take a slightly different approach and compile down all knowledge about a character into a single prototype trained to directly minimize the distance between the prototype and the tangent planes around each of the training examples.

Figure 4 illustrates the idea. Imagine that the four points 1-4 portray in image space different examples of the same digit, subject to some smooth transformation. As in tangent distance, one could represent this curve using the points and their local tangents (thick lines). However one might do better splitting it into three local linear models rather than four – model ‘a’ (just a line in this simple case) averages the upper part of the curve more effectively than the combination of the two tangents at ‘1’ and ‘2’. Given just the points, one might also construct model ‘b’ for ‘3’ and ‘4’, which would be unfortunate. Incorporating information about the tangents as well would encourage the separation of these segments. Care should be taken in generalising this picture to high dimensional spaces.

For both PCA and FA, the output is linearly related to the input, *ie* $\mathbf{y}^i = A\mathbf{x}^i$ with $A = RG$. In order for the models to be tolerant to distortions along tangent direction[5]

[5] j indexes the direction, for example, horizontal and vertical translations, scalings and shears

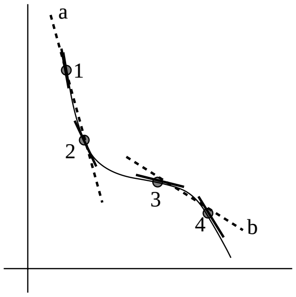


Figure 4: Didactic example of tangent distance and local linear models. See text for details.

$\mathbf{t}^{i,j}$ requires that both $\mathbf{x}^i + \mu_j \mathbf{t}^{i,j}$ and $\mathbf{x}^i - \mu_j \mathbf{t}^{i,j}$ be reconstructed well, where μ_j is a parameter that indicates how far along the tangent vector $\mathbf{t}^{i,j}$ good reconstruction should be enforced. The overall reconstruction error is then proportional to

$$E^i \sim |\mathbf{x}^i - A\mathbf{x}^i|^2 + \sum_j \mu_j^2 |\mathbf{t}^{i,j} - A\mathbf{t}^{i,j}|^2 \qquad (9)$$

and so the model has to reconstruct the tangent vectors as well as the underlying images. The relative importance of the latter with respect to the former is controlled by μ_j^2. A similar expression governs the code cost for the hidden units in FA.

An alternative way of viewing this prior is in terms of adding additional examples to the training set that are generated by these manipulations. Each example would be replaced by a Gaussian 'cloud' of fictitious examples. Adding these examples is straightforward using methods such as PCA and FA which are based on just the covariance matrix of the inputs – it amounts to adding into the covariance matrix the *a priori* tangent vectors, weighted by an amount, μ_j, that trades off the importance of the underlying pattern and the importance of the invariances. The one difference from just adding extra examples to the dataset is that we set the responsibilities of each model for the tangent vectors according to how well the model reconstructed just the 'parent' image, since the intention is to force the local linear models to capture the invariances themselves, using the tangents to shape the local structure within a submodel. If the database of training examples had just been expanded, this constraint would not have been applied.

There are two differences between this use of the tangent vectors and that in Simard *et al* [7]. One is that for us, the effect of these tangents has a limited spatial extent, whereas for them, the linear manifold about each image extends to infinity. In practice, in high dimensional image spaces, it is unlikely that images will have very large projections within the tangent space and small projections off it. The second difference is that Simard *et al* consider tangent manipulations to the test image as well as the training images – this requires finding the closest approach between the linear manifold about the test image and the linear manifolds about all the training images. It would be straightforward to do this during recognition for both PCA and FA; however doing it during learning is computationally more tricky [13].

Simard *et al*'s metric would be irrelevant in the limit of very large numbers of training images, since the database itself would contain all the transformations that actually preserve digit identity. In the same limit, the local linear PCA and FA methods would also not benefit from the tangents.

V Experimental Results

We have evaluated the efficacy of PCA and FA based density models at classifying images of handwritten digits. For both PCA and FA models, with and without tangents, we proceed as follows: During the training phase, only images of one class (say images of 2s) are presented to the mixture model. The learning algorithm outlined in section II is executed and builds a mixture of linear sub-models for each class. In order to save time, we initialize the search for FA models from the PCA models. For PCA models the coding cost is just the reconstruction cost, while for the FA models, one form of the complete code cost for example i is obtained as the sum of coding the reconstruction error and the code cost of the factors. Using Gaussian models for both, this can be expressed as:

$$\begin{aligned} E^i &= \frac{1}{2}\left[\log\frac{|\mathcal{I}|}{|\Sigma|} + \mathrm{trace}(\Sigma) + (R\mathbf{x}^i)^T R\mathbf{x}^i\right] + \\ & \sum_{j=1}^{n}\frac{1}{2\sigma_j^2}\left[(x_j^i - g_j R x_j^i)^2 + g_j^T \Sigma g_j\right] + \frac{1}{2}\log\sigma_j^2 + \mathcal{C} \end{aligned} \qquad (10)$$

where g_j is the j^{th} row of G. A test image $\mathbf{x}$ is presented to all 10 density models and estimates of the class conditional probabilities, $P(\mathbf{x}|k)$, $k = 1 \ldots 10$ are obtained. If all incorrect classifications are equally costly and if the prior belief is that all classes are equally likely then the Bayes decision rule stipulates that we should assign $\mathbf{x}$ to the class k^* where:

$$k^* = \mathrm{argmax}_k P(\mathbf{x}|k)$$

We used images from the CEDAR CDROM 1 database of Cities, States, ZIP Codes, Digits, and Alphabetic Characters [26]. The *br* training set of binary segmented digits was subdivided into two sets of size 7,000 and 4,000 respectively. The former subset was used to train the density models and the latter subset was used as a cross-validation set to allow us to choose various parameters such as the number of local linear sub-models to use in the mixtures and the number of principal components (factors) in the PCA (FA) models. When building models that use tangent information we are also free to specify the relative weightings (μ_j in equation (9)). In reality, we did not perform an exhaustive search for optimal values for all these parameters, but simply chose values that did reasonably well on the cross-validation images. For the results reported here, we allowed up to 10 principal components[6] (or factors) in each sub-model. There were also 10 sub-models in each mixture.

The CEDAR database includes two designated test sets. The *goodbs* (2213 images) set is a subset of the *bs* (2711 images) set containing only well segmented digits. After picking all the parameter values, we used all $11,000$ images to train a final version of the models which was used to evaluate performance on the test sets.

The binary images in the data set are of varying sizes, so we first scaled them to lie on an 8×8 pixel grid and then smoothed with a Gaussian filter with a standard deviation of half a pixel.

Figure 5 illustrates reconstructions of two very different styles of 2's. Sub-models which have specialized for one style reconstruct poorly images of the other style. Examples of the PCA and FA models' weights are shown in figures 6 and 7 respectively. In the PCA models, the recognition and generative weight matrices are simply transposes of each other[7] while for the FA models they differ.

The performance of the different methods are presented in Table 1. There are no significant differences between the performances of the different methods at the $p < 0.05$ level on the *bs* test set when compared pairwise using a two-tailed *McNemar's test* [27]. In comparing the columns of table 1, it is important to note that the *goodbs* is a carefully chosen subset of the *bs* test data when poorly segmented dig-

[6] This was usually sufficient to explain at least 95% of the training set variance assigned to that sub-model. Principal components that explained in excess of 95% were discarded to avoid overfitting.

[7] This follows because the weight vectors are mutually orthogonal and so the the generative model inverse is simply its transpose.

its were manually removed [26]. The training data was also manually screened. It is therefore reasonable to conclude by comparing the validation and *goodbs* performances that the models did not overfit.

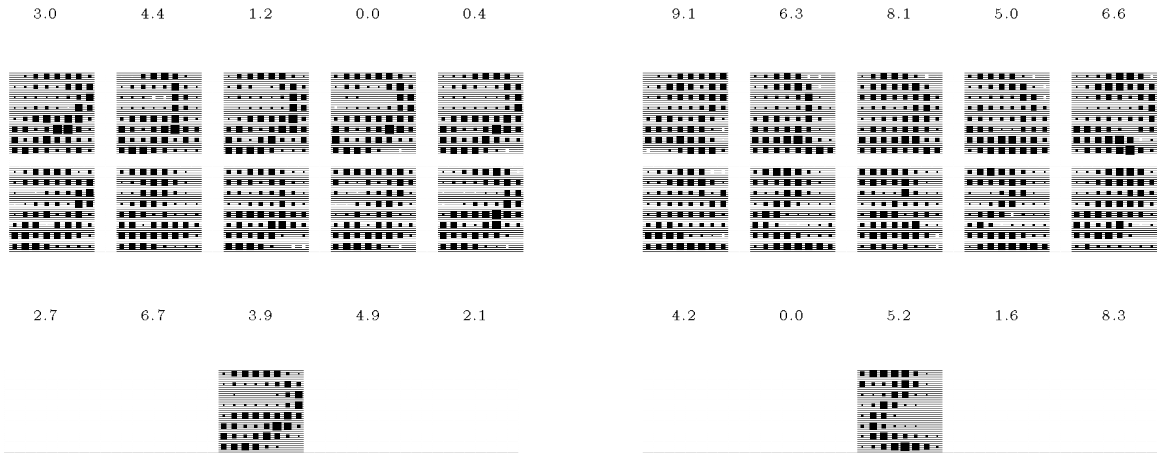


Figure 5: Operation of the mixture of PCA models on two different styles of 2's. The 8×8 image is shown at the bottom. The first two rows show the reconstructions from each of the 10 sub-models in the density estimator trained on images of 2's. Attached to each sub-model's reconstruction is its reconstruction cost relative to the best sub-model (which has cost of zero)

.

As a comparison, other state of the art methods obtain about 3% error rates [6] on the original data which had a mean size of around 45×60 pixels. Thus the images we used had areas about 40 times smaller and so could well have lost information. As a rough guide k-nearest neighbour[8] has an error rate of 5.4% on the *bs* test set. This is significantly worse ($p < 0.05$) than the performance obtained with the PCA with tangents method or the FA methods. The k-nearest neighbour method requires

[8] $k = 4$ chosen on the basis of the validation set

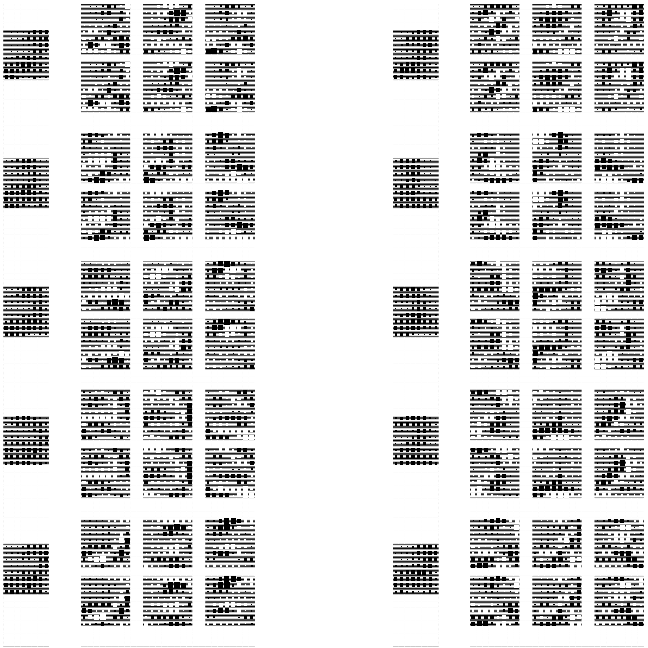


Figure 6: Weights in the 10 sub-models of the PCA density model for the digit 2. On the left are the means. The upper row immediately to the right of the mean shows the input-hidden weights (recognition weights), while the lower row gives the hidden-output weights (generative weights).

of the order of $N = 1100$, n dimensional (in this case $n = 64$) dot products to classify an image in one of the last 2 columns in table 1. Our local models (PCA or FA) require of the order $2 \times m \times r$ n-dimensional dot products for each density model. Since there are 10 digits and we used $m = r = 10$, the density models require of the $10 \times 2 \times 10 \times 10 = 2000$ or about a factor of five times less computation than a straightforward implementation of the k-nearest neighbour method. The number of dot products for the tangent distance method depends on the number of tangent directions [7]. If 7 tangent directions are used on these data then of the order of $96 \times N$ dot products are needed, or about a factor of 525 times the local model version.

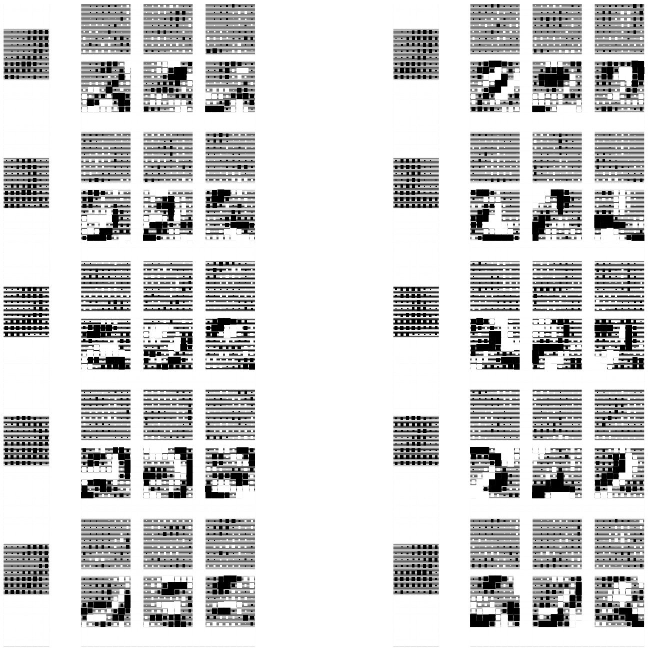


Figure 7: Weights in the 10 sub-models of the FA density model for the digit 2. The layout is the same as in figure 5.

VI Discussion

We have constructed two different sorts of locally linear models of images of digits, one based on principal components analysis and one on factor analysis, and have shown that they can both perform well at digit discrimination. These models illustrate three major points. First, linear models with just a few dimensions can be used to good effect for representing the local structure of the high dimensional complex manifolds of the pixel images of digits. Second, allowing multiple components in the mixtures can be very beneficial. In general, this could be both for capturing grossly different styles of the digits (such as 2s with and without loops in their tails) and for the effects of affine transformations of the digits by more than about a single pixel, which have

	Validation Set	*goodbs* test set	*bs* test set
PCA	1.95	2.35	4.91
FA	1.68	2.17	4.68
PCA + Tangents	1.83	2.17	4.72
FA + Tangents	1.63	2.49	5.05

Table 1: Percentage of images incorrectly classified by each of the methods. The validation results were obtained when the density models were trained using the $7,000$ training examples. The results in the last two columns were obtained with models trained on all $11,000$ training images.

highly non-linear effects on the observed images.[9] Third, *a priori* information about the local structure of the manifolds that comes from knowledge about invariances of digit identities over certain transformations [7] is very easy to incorporate into these linear models. Note that FA is just a particular way of limiting the number of parameters that define the covariance matrix used to model data.

It is clear that if the manifold of the images is mostly differentiable, then local linear models will do well at representing them. Using the EM scheme for iterative competitive clustering finds the sets of images that respect the local structure and finds the natural 'separation' points between the distinct parts. Nearest neighbour

[9] Since we did not have enough data to fit a very large number of local models for each digit, and the digits were reasonably well normalised, the different local models were mostly due to different styles.

schemes are the logical limit of such schemes, and should work best if one has so many examples that the local structure of the manifold can be inferred by finding the nearest point or points and fitting a linear surface to them. There are two potential advantages to the EM scheme. First, it is using information from somewhat distant images to determine the local directions in the manifold, allowing it to average away the substantial noise that corrupts the images. Of course, if too much averaging is done, then the directions could be systematically biased. Second, at recognition time, rather than having to search the entire training set to find the patterns closest to the new input, knowledge about these patterns has been pre-compiled into the limited number of centres and the limited number of principle component or factor directions associated with them. Recognition can therefore be quite fast.

In this study, we did not find that much difference between our two sorts of models for the covariance structure within a linear patch. One might have expected factor analysis to have had better performance, since its prior model of the image generation process is more reasonable. For FA, given the factors, any discrepancies between the model and the image are independent from one pixel to the next. PCA uses a spherical Gaussian in the directions not covered by the retained components, and this can amount to a complicated distribution over the pixels themselves. Also FA explicitly models covariance structure, whereas PCA models both variance and covariance. This advantage may have been nullified by our normalisation and regularisation procedures. On the other hand, it is computationally much cheaper to perform PCA during the learning phase, although they are equally expensive during recognition. Both models perform soundly on these data. Finding a better way to regularise the FA model against irrelevant pixels is important. Given its model, FA is completely correct to assign a zero variance to outlying pixels that are silent throughout the training set. Our prior knowledge that this might happen can be used to specify a more complicated prior that makes it possible that pixels that are generally inactive can occasionally be corrupted by noise.

We also found that the inclusion of tangent vectors did not substantially improve the performance. Our use of tangent vectors is essentially an instantiation of tangent-prop [28], which constrains the output of the network to satisfy appropriate invariances through its directional derivatives. Since our networks are linear, these directional derivatives are particularly simple, allowing the tangent vectors just to be added into the covariance matrices. If the tangents about input $\mathbf{x}^k$ were perfectly captured by a linear model centred at that point, then the reconstruction error E^i for input $\mathbf{x}^i$ would be exactly the one-sided tangent distance from $\mathbf{x}^i$ to the tangent space of input $\mathbf{x}^k$. Hastie & Simard [13], developed a locally linear mixture model analogous to the one described here, except using two-sided tangent distances during the whole of learning. Tangents would be expected not to help if there are enough data points that they express directly all the actual invariances.

It is challenging to model the low dimensional manifolds of high dimensional pixel images of digits in a computationally tractable manner. Our locally linear models are designed to capture aspects of the short-range structure of the manifold and to respect other knowledge about the digit modelling problem, such as the fact that there are different styles of handwriting even for the same digit. The models are conceptually and computationally straightforward.

Acknowledgments

This research was funded by Apple and by the Ontario Information Technology Research Centre. We thank Patrice Simard, Zoubin Ghahramani, Rob Tibshirani and Yann Le Cun for helpful discussions. Geoffrey Hinton is the Nesbitt-Burns fellow of the Canadian Institute for Advanced Research.

References

[1] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network", in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed., Denver, 1990, vol. 2, pp. 396–404, Morgan Kaufmann, San Mateo.

[2] D. E Specht, "Probabilistic neural networks", *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[3] C. Bishop, *Neural networks for pattern recognition*, Clarendon Press, 1995.

[4] B. Widrow, "The 'rubber-mask' technique–I. Pattern Measurement and Analysis", *Pattern Recognition*, vol. 5, pp. 175–197, 1973.

[5] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures", *IEEE Trans. Computers*, vol. C-22, no. 1, pp. 67–92, 1973.

[6] M. Revow, C. K. I. Williams, and G. E. Hinton, "Using generative models for handwritten digit recognition", *IEEE Transactions Pattern Analysis and Machine Intellegince*, vol. 18, no. 6, pp. 592–606, 1996.

[7] P. Simard, Y. Le Cun, and J. Denker, "Efficient pattern recognition using a new transformation distance.", in *Advances in Neural Information Processing Systems 5*, J. D. Cowan S. J. Hanson and C. L. Giles, Eds., pp. 50–58. Morgan Kaufmann, 1993.

[8] P. Simard, "Efficient computation of complex distance metrcis using hierarchial filtering", in *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J Alspector, Eds., pp. 168–175. Morgan Kaufmann, 1994.

[9] G. E. Hinton, M. Revow, and P. Dayan, "Recognizing handwritten digits using mixtures of linear models", in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds., pp. 1015–1022. MIT Press, Cambridge MA, 1995.

[10] C. Bregler and S. M. Omohundro, "Nonlinear image interpolation using manifold learning", in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds., pp. 971–980. MIT Press, 1995.

[11] N. Kambhatla and T. K. Leen, "Fast non-linear dimension reduction", in *Advances in Neural Information Processing Systems 6*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., pp. 152–159. Morgan Kaufmann, 1994.

[12] K. Sung and T. Poggio, *Example based learning for view-based human face detection*, AI Memo 1521, CBCL paper 112, 1995.

[13] T. Hastie and P. Simard, "Learning prototype models for tangent distance", in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds., pp. 999–1006. MIT Press, 1995.

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Proceedings of the Royal Statistical Society*, vol. B-39, pp. 1–38, 1977.

[15] G. E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy", in *Advances in Neural Information Processing Systems 6*, J. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan Kauffmann, 1994.

[16] R. S. Zemel, *A minimum description length framework for unsupervised learning*, PhD Thesis, Department of Computer Science, University of Toronto, 1993.

[17] H. Schwenk and M. Milgram, "Transformation invariant autoassociation with application to handwritten character recognition", in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds., pp. 991–998. MIT Press, 1995.

[18] B. S. Everitt, *An introduction to latent variable models*, Chapman and Hall, 1984.

[19] K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate analysis*, Academic Press, 1979.

[20] R. M. Neal and Dayan P., "Factor analysis using delta-rule wake-sleep learning", Tech. Rep., No. 9607. Dept. of Statistics, University of Toronto. Available from ftp://ftp.cs.toronto.edu/pub/radford/ws-fa.ps.Z, 1996.

[21] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The wake-sleep algorithm for unsupervised neural networks", *Science*, vol. 268, pp. 1158–1161, 1995.

[22] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, "The Helmholtz machine", *Neural computation*, vol. 7, no. 7, pp. 889–904, 1995.

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

[24] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor clasification", *Accepted; IEEE Transactions Pattern Analysis and Machine Intellegince*, 1995.

[25] H. Schwenk and M. Milgram, "Learning discriminant tangent models for handwritten character recognition", in *ICANN*95*. Springer Verlag, 1995.

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

[27] J. L. Fleiss, *Statistical methods for rates and proportions*, Second edition. Wiley, 1981.

[28] P. Simard, B. Victorri, Y. LeCun, and J. Denker, "A formalism for specifying selected invariances in an adaptive network", in *Advances in Neural Information Processing Systems 4*, J.E. Moody, S.J. Hanson, and S. J. Lippmann, Eds., pp. 895–903. Morgan Kaufmann, 1992.