Metrics and Models for Handwritten Character Recognition*

Trevor Hastie† Patrice Y. Simard‡

December 8, 1997

Abstract

A digitized handwritten numeral can be represented as a binary or greyscale image. An important pattern recognition task that has received much attention lately is to automatically determine the digit, given the image.

While many different techniques have been pushed very hard to solve this task, the most successful and intuitively appropriate is due to Simard (Simard, LeCun & Denker 1993). Their approach combined nearest-neighbor classification with a subject-specific invariant metric that allows for small rotations, translations, and other natural transformations. We report on Simard's classifier, and compare it to other approaches. One important negative aspect of near-neighbor classification is that all the work gets done at lookup time, and with around 10,000 training images in high dimensions this can be exorbitant.

In this paper we develop rich models for representing large subsets of the prototypes. One example is a low-dimensional hyperplane defined by a point and a set of basis or tangent vectors. The components of these models are learned from the training set, chosen to minimize the average *tangent distance* from a subset of the training images — as such they are similar in flavor to the Singular Value Decomposition (SVD), which finds closest hyperplanes in Euclidean distance. These models are either used singly per class, or as basic building blocks in conjunction with the K-means clustering algorithm.

keywords: Nearest Neighbor, Classification, Invariance

1 Introduction

Figure 1 shows some handwritten digits taken from US envelopes. Each image consists of 16×16 pixels of greyscale values ranging from $0-255$. These 256 pixel values are regarded as a feature vector X to be used as input to a classifier, which will automatically assign a digit class based on the pixel values. We denote the digit class by the 10 level categorical variable C.

This particular database has been used in many different studies, and a variety of techniques have been attempted. There are $7,291$ training images and an "official" test set of 2007 images. We review some of these approaches, starting with a natural and classical procedure for producing linear decision boundaries between the classes.

*To appear in *Statistical Science*, 1998

†Statistics Department, Stanford University, CA94305. Trevor Hastie was partially supported by grant DMS-9504495 from the National Science Foundation, and grant ROI-CA-72028-01 from the National Institutes of Health.

‡AT&T Research Laboratories, 100 Schultz Dr, Red Bank, NJ 07701

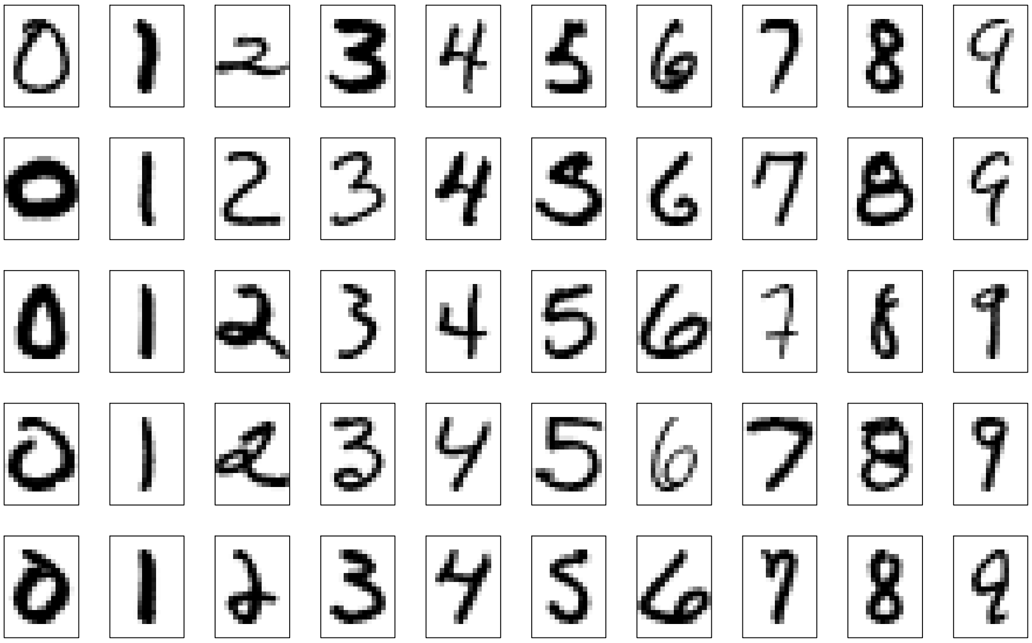

Figure 1: *Some examples of digitized handwritten numerals, all rescaled and normalized to a* 16×16 *greyscale image.*

1.1 Linear Discriminant Analysis

This simple model is often successful in classification tasks. The model assumes that the feature vector X has a multivariate gaussian distribution in each class, each with a different centroid μ_k but sharing a common covariance matrix Σ. Since X has 256 components, this is a high dimensional problem. Each of the 10 centroids have 256 parameters, and the common covariance matrix has $256 \times 257/2 = 32,896$ parameters. In addition we specify the *prior probabilities* for the classes $\pi_k = P(C = k)$. These parameters are then used to form the posterior probability estimates $\hat{P}(C|X)$, and the estimated Bayes classifier for this model assigns to the class k for which $\hat{P}(C = k|X)$ is largest. It is easily seen that the quadratic terms in X drop out in this calculation, and that

$$\begin{aligned} \log \hat{P}(C = k|X) &= -\hat{\mu}_k^T \hat{\Sigma}^{-1} X + \frac{1}{2}\hat{\mu}_k^T \hat{\Sigma}^{-1} \hat{\mu}_k + \frac{1}{2} \log \pi_k + q(X) \\ &= \hat{\beta}_k^T X + \hat{\beta}_{k0} + q(X), \end{aligned} \tag{1}$$

where $q(X)$ is the same for all k, and hence the *decision boundaries* between classes are linear. $\delta_k(X) = -\log \hat{P}(C = k|X)$ is known as the *discriminant function*, and hence the name linear discriminant analysis or LDA. We also see that the actual parameters needed for the discriminant functions are far fewer - in fact $(p+1)(K-1)$ for a K class problem in p dimensions.

LDA achieves a test error rate of around 11% on this problem, which might seem good at first glance, but does not compare well with competitors. LDA suffers from excessive bias and variance for this problem. The bias stems from using linear decision boundaries between the classes. We see later on that more flexible decision boundaries pay big rewards. The excess variance stems from the fact that once committed to linear decision boundaries, there are more efficient methods for estimating the linear parameters in (1).

Neighboring pixels tend to have strong positive correlation, and hence the corresponding discriminant coefficients will be negatively correlated. Different variants of LDA have been proposed to account for this spatial correlation and hence *borrow strength* from neighboring pixels. Hastie, Buja & Tibshirani (1995) used a penalized discriminant analysis where the discriminant coefficients are constrained to be spatially smooth. This technique brought the error rate down to 8.2%, by *effectively shrinking* the dimension of the space from 256 down to 40.

One way to reduce the bias is to represent each class by a mixture of several gaussian distributions. Hastie & Tibshirani (1996) developed such an approach, with up to 5 gaussians in each class. Their models could also accommodate the regularization used in penalized discriminant analysis. The error rates were only a slight improvement over penalized discriminant analysis.

Boser, Guyon & Vapnik (1992) fit optimal margin hyperplanes, also known as *support vector machines* between each class and the rest (Vapnik 1996). These techniques are related to the preceding, in that they fit linear decision boundaries (typically in a space augmented by basis expansions of the original pixels.) The idea is to find hyperplanes that either separate or approximately separate the data well. Boser et al. (1992) claim that they can search around in high-dimensional feature spaces without using exorbitant numbers of parameters. These models are certainly interesting and deserve closer scrutiny by the statistics community. They achieve error rates in the mid-4% range on these problems, and so outperform the other linear techniques.

1.2 Neural Network Classifiers

This digit recognition problem was tackled vigorously by the neural network community. A single hidden layer neural network model for this problem can be written as

$$\log \frac{P(C=k|X)}{P(C=K|X)} = \beta_{0k} + \beta_k^T z_j, \ k = 1, \ldots, K-1 \tag{2}$$

$$z_j = \sigma(a_{0j} + a_j^T X), \ j = 1, \ldots, M \tag{3}$$

where $K = 10$ is the number of classes and M is the number of hidden units z_j. The *activation* function σ creates nonlinear basis functions along projections $a_{0j} + a_j^T X$, and the most common such function is $\sigma(x) = 1/(1+e^{-x})$. Figure 2 depicts a single layer network, often known as a *perceptron* due to early work on models for the brain, where there are two outputs ($K = 2$), three inputs ($p = 3$), and four hidden units ($M = 4$).

This model can be seen as an extension of the polychotomous logistic regression model, and is similar in structure and flavor to the projection pursuit regression models of Friedman & Stuetzle (1981). There is a large literature on such models (Ripley 1996, Bishop 1995), with many possibilities for fitting, regularizing and sizing the networks.

While the addition of possibly many nonlinear basis functions will reduce the bias of linear methods in this problem, the number of effective parameters grows rapidly along with the variance. Many variants have been proposed to reduce this effect. One such class uses the concept of *local connectivity with shared weights (parameters)* at hidden units. The idea is to have more than one layer of hidden units. Each layer serves as a *feature extractor* from the previous layer, with each unit connected to a localized region of its input image e.g. a 4×4 block. The same set of weights are used over all such blocks, and hence they serve as a filter for extracting a particular local feature of the image.

Although neural networks are often regarded as automatic "black box" classifiers, they do require some tuning. The number of hidden layers and units within a layer need to be determined. The algorithms for fitting the networks require tuning as well, and are

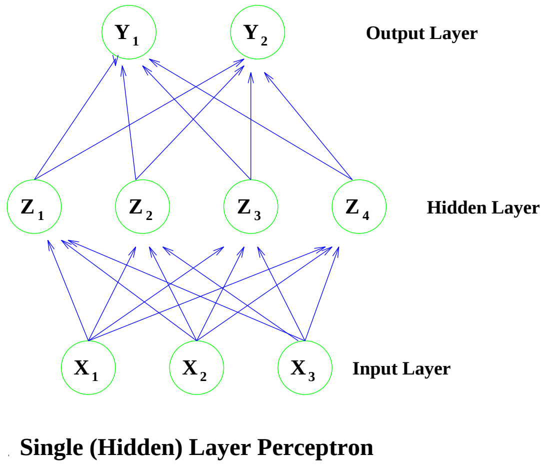

Figure 2: A single layer neural network with 4 hidden units and 2 output units

sensitive to learning rates, regularization parameters, and initialization. Many different neural network configurations have been tailored for this particular application along the lines outlined above (Le Cun, Boser, Denker, Henderson, Howard, Hubbard & Jackel 1990). The error rates are typically around 5%.

A word of caution is needed when tackling a popular problem of this kind. Although there is an official test set of data to be used to evaluate different methods, it can be over-used. For example, a group may attempt tens or hundreds of different configurations, but only report the results of the best. These caveats hold for any technique with tunable parameters, but are especially pertinent for neural networks which have many.

1.3 Methods that incorporate invariances

One of the problems with all the methods discussed so far is that they are insensitive to the spatial organization of the pixels. If we rotate the image a few degrees before digitization, the nature of the pixel representation can change dramatically, while the nature of the image (and the ability of the human to identify it) have not changed much at all. While it can be argued that the techniques that require spatial smoothness create such sensitivities by *blurring* the images, these were insufficient on their own to make dramatic improvements. Ideally we want techniques that are insensitive to small *natural transformations* such as location shifts, rotation, horizontal and vertical scaling, and shear. These are known as the *affine* transformations. We will see that other natural invariances are also desirable for digit recognition.

Hastie & Tibshirani (1993) proposed a prototype method, where each digit was represented by one or more piece-wise linear curves. The images were represented by a point set in two-dimensional *horizontal/vertical* coordinates, obtained by thresholding the greyscale values of each pixel. Finally, each prototype was fit to the points by affine-invariant least squares. This allowed transformations of the prototype such as rotations, shifts, scale changes etc, and care was taken to limit the range of these transformations. They then used lack of fit statistics as a basis for a classifier, and achieved just under 5% errors. The methods described in this paper are similar in spirit, but implemented in quite a different way (with far greater success).

Nearest neighbor classifiers are extremely simple, and always worth trying as a benchmark with any classification task. The one-nearest neighbor classifier (1-NN) is the most simple: a new object is classified by assigning the class of the closest training object. "Closest" is in feature space, and implies a metric; almost always euclidean distance is used, although this is not always the most reasonable choice. On these data euclidean-metric 1-NN achieves a test error rate of 5.3%. This is a rather remarkable feat given the high dimensionality of the data—we learn that the *curse of dimensionality* makes near-neighborhoods very large in high dimensions. We conjecture that the reasons for the success of 1-NN are:

- Invariances are built in, since with 700 "threes" for example, there are likely to be slightly rotated versions, different sizes and so on.
- There is evidence that the data lie clustered around low dimensional manifolds, so the effective dimension is much lower than 256.

Simard et al. (1993) recognized the power of 1-NN in this context, and showed that its performance could be further improved by incorporating invariance to specific transformations in the underlying distance metric — the so called *tangent distance*. They achieved the best performance on these data with a test error rate of 2.6%. We review tangent distance in section 2.

Being a memory based technique, nearest neighbor classification can be computationally expensive to classify new observations (here we have $7,291$ training observations, and a partial sort is required for each classification). This is exacerbated due to the additional computations required for the tangent-distance metric. While techniques have been proposed for editing and thinning large data bases for nearest neighbor rules, this has up to now not been successfully implemented for these data. So while the tangent metric achieves the best results, the computationally prohibitive lookup costs make it infeasible for routine use.

In this paper we address this problem for the tangent distance algorithm, by developing rich models for representing large subsets of the prototypes. Our leading example of a prototype model is a low-dimensional (12-dim) hyperplane defined by a point and a set of basis or tangent vectors. The components of these models are learned from the training set, chosen to minimize the average *tangent distance* from a subset of the training images — as such they are similar in flavor to the Singular Value Decomposition (SVD), which finds closest hyperplanes in Euclidean distance. These models are either used singly per class, or as basic building blocks in conjunction with the K-means clustering algorithm to produce a set of prototypes per class. Our results show that not only are the models effective, but they also have meaningful interpretations. In handwritten character recognition, for instance, the main tangent vector learned for the the digit "2" corresponds to addition/removal of the loop at the bottom left corner of the digit; for the 9 the fatness of the circle. We can therefore think of some of these learned tangent vectors as representing additional invariances derived from the training digits themselves. Each learned prototype model therefore represents very compactly a large number of prototypes of the training set.

2 Overview of Tangent Distance

When we look at handwritten characters, we are easily able to allow for simple transformations such as rotations, small scalings, location shifts, and character thickness when identifying the character. Any reasonable automatic scheme should similarly be insensitive to such changes.

Simard et al. (1993) finessed this problem by generating a parametrized 7-dimensional manifold for each image, where each parameter accounts for one such invariance. Consider a single invariance dimension: rotation. If we were to rotate the image by an angle θ prior to digitization, we would see roughly the same picture, just slightly rotated (see figure 3.) Our images are 16×16 grey-scale pixelmaps, which can be thought of as points in a 256-

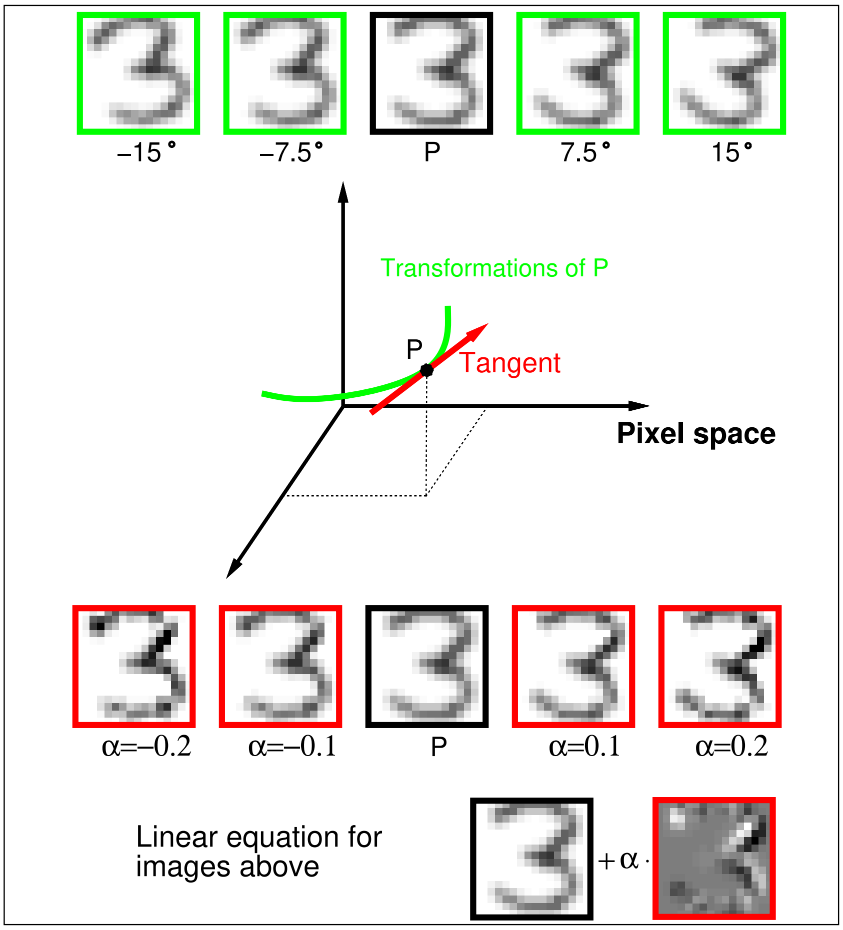

Figure 3: *A series of rotated versions of the image of a three, approximated by points along the tangent to the rotation curve. The tangent approximation starts to degrade as the angle gets large. This tangent family can be represented by a parametrized line, where the unit direction vector can itself be usefully displayed as an image.*

dimensional Euclidean space. The rotation operation traces out a smooth one-dimensional curve $X_i(\theta)$ with $X_i(0) = X_i$, the image itself. Instead of measuring the distance between two images as $D(X_i, X_j) = \|X_i - X_j\|$ (for any norm $\|\cdot\|$), the idea is to use instead the rotation-invariant $D^I(X_i, X_j) = \min_{\theta_i,\theta_j} \|X_i(\theta_i) - X_j(\theta_j)\|$. Simard et al. (1993) used 7 dimensions of invariance, accounting for horizontal and vertical location and scale, rotation and shear and character thickness.

Deriving the manifold exactly is impossible, given a digitized image, and would be impractical anyway. They approximated the manifold instead by its tangent plane at the image itself, leading to the tangent model $\tilde{X}_i(\theta) = X_i + T_i\theta$, and the *tangent distance* $D^T(X_i, X_j) = \min_{\theta_i,\theta_j} \left\|\tilde{X}_i(\theta_i) - \tilde{X}_j(\theta_j)\right\|$. Here we use θ for the 7-dimensional parameter, and for convenience drop the tilde. Notice that the metric allows movement in the tangent

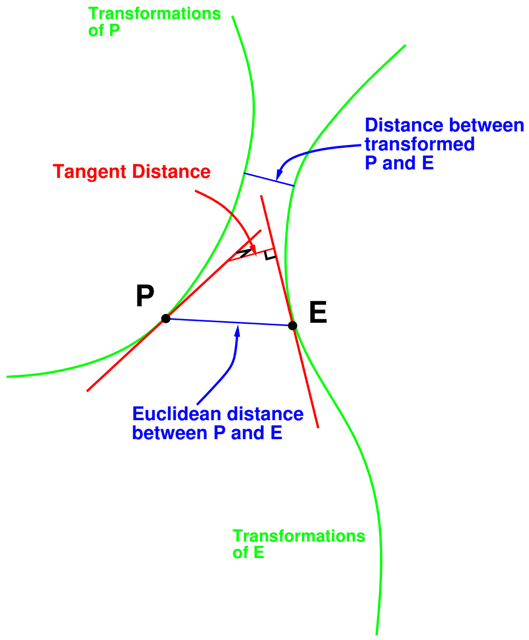

Figure 4: *Associated with each image is a manifold of dimension 7 corresponding to the the seven transformations such as rotation, scaling, etc. In principal we would like to compute the shortest distance between the manifolds of two images. Tangent distance approximates these manifolds by their tangent hyperplanes, which simplifies the distance calculations dramatically.*

spaces of both the prototype and the test image. Figure 4 illustrates the approximation. The approximation is valid locally, and thus permits local transformations. Non-local transformations are not interesting anyway (we don't want to flip 6s into 9s, or shrink all digits down to nothing (Säckinger 1992)). Simard et al. (1993) report that they found it unnecessary to restrict the transformations to be local, since the degradation of the linear approximation far from the origin produced images that were extremely distorted.

The computations involved in the approximation are relatively straightforward. We give some details on the computation of T_i in the appendix. If $\|\cdot\|$ is the Euclidean norm, computing the tangent distance is a simple least-squares problem, with solution the square-root of the residual sum-of-squares in the regression with response $X_i - X_j$ and predictors $(-T_i : T_j)$.

Simard et al. (1993) used D^T to drive a 1-NN classification rule, and achieved the best rates so far—2.6%—on the official test set (2007 examples) of the USPS data base. Unfortunately, 1-NN is expensive, especially when the distance function is non-trivial to compute; for each new image classified, one has to compute the tangent distance to each of the training images, and then classify as the class of the closest. Our goal in this paper is to reduce the training set dramatically to a small set of prototype models; classification is then performed by finding the closest prototype.

3 Prototype Models

The centroid of a set of N points in d dimensions minimizes the average squared norm from the points:

$$M = \frac{1}{N}\sum_{i=1}^{N} X_i = \arg\min_M \sum_{i=1}^{N} \|X_i - M\|^2 \tag{4}$$

In this section we explore some ideas for generalizing the concept of a mean or centroid for a set of images, taking into account the tangent families. Such a centroid model can be used on its own, or else as a building block in a K-means clustering algorithm at a higher level. We will interchangeably refer to the images as points (in 256 space).

Tangent centroid

One could generalize definition (4) and ask for the point M that minimizes the average squared *tangent distance*:

$$M_T = \arg\min_M \sum_{i=1}^{N} D^T(X_i, M)^2 \tag{5}$$

This appears to be a difficult optimization problem, since computation of tangent distance requires not only the image M but also its tangent basis T_M. Thus the criterion to be minimized is

$$D(M) = \sum_{i=1}^{N} \min_{\gamma_i,\theta_i} \|M + T(M)\gamma_i - X_i - T_i\theta_i\|^2$$

where $T(M)$ produces the tangent basis from M (see appendix for details). All but the location tangent vectors in $T(M)$ are nonlinear functionals of M, and even without this nonlinearity, the problem to be solved is a difficult inverse functional.

The following iterative algorithm is motivated on intuitive grounds:

Tangent Centroid Algorithm

Initialize: Set $M^0 = \frac{1}{N}\sum_{i=1}^{N} X_i$, let $T_M^0 = T(M^0)$ be the derived set of tangent vectors, and $D^0 = \sum_i D^T(X_i, M^0)$. Denote the current tangent centroid (tangent family) by $M^0(\gamma) = M^0 + T_M^0\gamma$.

Iterate:
1. For each i find a γ_i^ℓ and θ_i^ℓ that solve $\min_{\gamma,\theta}\left\|M^{\ell-1} + T_M^{\ell-1}\gamma - X_i(\theta)\right\|$
2. Set $M^\ell \leftarrow \frac{1}{N}\sum_{i=1}^{N}(X_i(\theta_i^\ell) - T_M^{\ell-1}\gamma_i)$ and compute the new tangent subspace $T_M^\ell = T(M^\ell)$.
3. Compute $D^\ell = \sum_i D^T(X_i, M^\ell)$

Until: D^ℓ converges.

Given the current guess for $M = M^{\ell-1}$, in step 1 we locate the closest member of its tangent family, namely $M(\gamma_i^\ell)$, to the tangent family of of X_i, namely $X_i(\theta_i^\ell)$. In step 2 it might seem natural to replace M by the average of the $X_i(\theta_i^\ell)$. Instead we treat T_M as fixed, and pick M^ℓ to minimize the norm. Note that the first step in **Iterate** is available from the computations in the third step. The algorithm divides the parameters into two sets: M in the one, and then T_M, γ_i and θ_i for each i in the other. It alternates between the two sets, although the computation of T_M given M is not the solution of an optimization problem. It seems very hard to say anything precise about the convergence or behavior of this algorithm, since the tangent vectors depend on each new version of M in a nonlinear way. Our experience has always been that it converges fairly rapidly (< 6 iterations). A potential drawback of this model is that the T_M are not learned, but are implicit in M. The next proposal attends to this deficiency.

Tangent subspace

Rather than define the model as a point and have it generate its own tangent subspace, we can include the subspace as part of the parametrization: $M(\gamma) = M + V\gamma$. Then we define this *tangent subspace model* as the minimizer of

$$D(M,V) = \sum_{i=1}^{N} \min_{\gamma_i,\theta_i} \|M + V\gamma_i - X_i(\theta_i)\|^2 \tag{6}$$

over M and V. Note that V can have an arbitrary number $0 \le r \le 256$ of columns, although it does not make sense for r to be too large. An iterative algorithm similar to the tangent centroid algorithm is available, which hinges on the SVD decomposition for fitting affine subspaces to a set of points. We briefly review the SVD in this context.

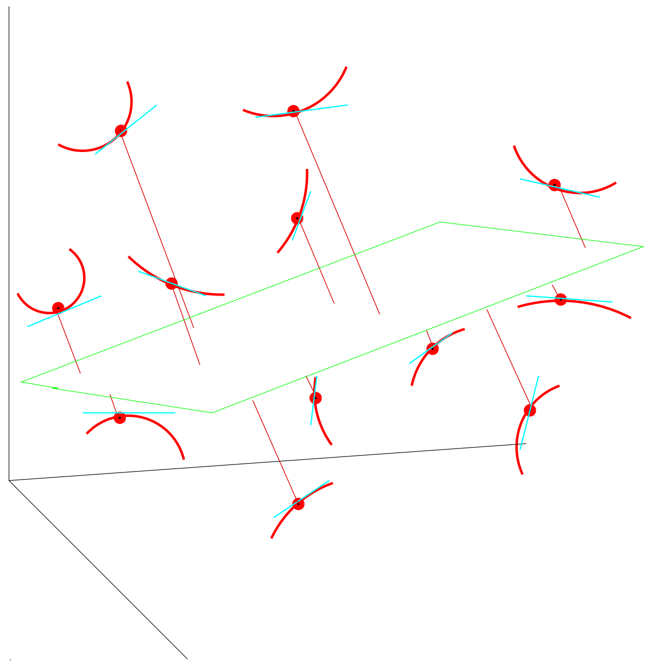

Figure 5: *The SVD finds a hyperplane of a given dimension that minimizes the average squared distance to a set of points. In this case the points are the pixel values of greyscale images in 256 dimensional space. The* tangent subspace *model finds the hyperplane closest in* tangent distance *to a set of images; this approximates a collection of (linearized) manifolds by a hyperplane.*

Let $\mathcal{X}$ be the $N \times 256$ matrix with rows the vectors $X_i - \bar{X}$ where $\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i$. Then $SVD(\mathcal{X}) = UDV^T$ is a unique decomposition with $U_{N\times R}$ and $V_{256\times R}$ the orthonormal *left* and *right* matrices of *singular vectors*, and $R = \operatorname{rank}(\mathcal{X})$. $D_{R\times R}$ is a diagonal matrix of decreasing positive *singular* values. Some properties of the SVD that are pertinent here are:

1. If we replace D by $D^{(r)}$, which is D with the last $R-r$ entries replaced by zero, then $UD^{(r)}V^T$ is the *best rank* r approximation to $\mathcal{X}$, in the least-squares sense.

2. Consider finding the closest affine, rank-r subspace to a set of points, or

$$\min_{M,V^{(r)},\{\gamma_i\}} \sum_{i=1}^{N} \left\| X_i - M - V^{(r)}\gamma_i \right\|^2$$

   where $V^{(r)}$ is $256 \times r$ orthonormal. The solution is given by the SVD above, with $M = \bar{X}$ and $V^{(r)}$ the first r columns of V.

3. The total residual squared distance of the solution is $\sum_{j=r+1}^{R} D_{jj}^2$.

4. The optimal γ_i indexes the orthogonal projection of X_i onto the subspace: $\gamma_i = {V^{(r)}}^T(X_i - \bar{X})$, and $\hat{X}_i = \bar{X} + V^{(r)}{V^{(r)}}^T(X_i - \bar{X})$

5. The $V^{(r)}$ are also the largest r principal components or eigenvectors of the covariance matrix of the X_i. They give in sequence directions of maximum spread, and for a given digit class can be thought of as class-specific invariances.

We now present our *Tangent subspace algorithm* for solving (6); for convenience we assume V is rank r for some chosen r, and drop the superscript.

Tangent subspace algorithm

Initialize: Set $M^0 = \frac{1}{N}\sum_{i=1}^{N} X_i$ and let V^0 correspond to the first r right singular vectors of $\mathcal{X}$. Set $D^0 = \sum_{j=r+1}^{R} D_{jj}^2$, and let the current tangent subspace model be $M^0(\gamma) = M^0 + V^0\gamma$. The SVD supplies γ_i^0, and $\theta_i^0 = 0$.

Iterate:

1. For each i find a θ_i^ℓ (and γ_i) that solve $\min_{\gamma_i,\theta_i} \left\| M^{\ell-1}(\gamma) - X_i(\theta) \right\|$
2. Set $M^\ell \leftarrow \frac{1}{N}\sum_{i=1}^{N} X_i(\theta_i^\ell)$ and replace the rows of $\mathcal{X}$ by $X_i(\theta_i^\ell) - M^\ell$. The SVD of $\mathcal{X}$ gives V^ℓ (the first r right singular vectors), and γ_i^ℓ.
3. Compute $D^\ell = \sum_{j=r+1}^{R} D_{jj}^2$

Until: D^ℓ converges.

The algorithm alternates between

1. finding the closest point in the tangent subspace of each image to the current tangent subspace model, and

2. computing the SVD for these closest points.

These two steps alternate between *overlapping parameter spaces*: $S_1 = \{\gamma_i, \theta_i\}$ and $S_2 = \{M, V, \gamma_i\}$. In the tangent centroid model, we took advantage of the γ_i^ℓ learned in step 1 in updating M. Here γ_i is optimized in both steps 1 and 2; in step 1 optimizing both γ_i and θ_i allows for a better choice of θ_i, and the final choice of γ_i is made in step 2.

Each step of the alternation decreases the criterion (decreases or leaves alone). Suppose after iteration ℓ the parameters are $\{M^\ell, V^\ell, \theta_i^\ell, \gamma_i^\ell\}$ and the squared distance is

$$D(M^\ell, V^\ell) = \sum_{j=r+1}^{R} D_{jj}^2 = \sum_{i=1}^{N} \left\| M^\ell + V^\ell \gamma_i^\ell - X_i(\theta_i^\ell) \right\|^2,$$

- Step 1 reduces each of the N components of the sum in $D(M, V)$ by N separate optimizations for each of θ_i and γ_i; only the θ_i are retained.
- Step 2 fixes θ_i and updates M, V and γ_i by the SVD. Since this is a least squares procedure, and the values used in step 1 are acceptable candidates, the criterion again decreases.

Since each step either reduces D or leaves it alone, and D is positive, it converges. In all our examples we found that 12 complete iterations were sufficient to achieve a relative convergence ratio of 0.001. Figure 5 illustrates the idea behind the tangent subspace model.

The algorithm is stationary for any solution to (6), since it is easy to see that any such solution must be the SVD of the closest tangent points. Some degeneracies can occur. For example if any of the tangent vectors for any of the images lie in the span of V, then the θ_i and γ_i for that image will not be unique. In such cases we set the aliased components of θ_i and γ_i to zero, to eliminate any unwanted influence at extremes of their ranges. All these statements do not quite amount to proof that the algorithm converges to a stationary point of the criterion. One would need to show, for example, that when the criterion failed to decrease, the gradient with respect to all the parameters was zero. In light of the possible degeneracies outlined above, this need not be the case. To date we have no convergence proof.

An alternative approach we are currently exploring is to eliminate all the θ_i once and for all. Let $H_i = T_i(T_i^T T_i)^{-1} T_t^T$, the projection operator onto the tangent subspace T_i of X_i. Then we can reduce the objective function (6) to

$$D(M, V) = \sum_{i=1}^{N} \min_{\gamma_i} \|(I - H_i)(M + V\gamma_i - X_i)\|^2 \tag{7}$$

$$= \sum_{i=1}^{N} \min_{\gamma_i} \|(M + V\gamma_i - X_i)\|_{I-H_i}^2 \tag{8}$$

This is again cast as a generalization of the closest-hyperplane problem, where each point carries its own metric. In this case the metric is defined by a positive semi-definite matrix. The first author has encountered two unrelated problems leading to a similar criterion, and current research is focussed on efficient algorithms for optimizing such criteria.

One advantage of the tangent subspace model is that we need not restrict ourselves to a seven-dimensional V — indeed, we have found 12 dimensions has produced the best results. The basis vectors found for each class are interesting to view as images. Figure 6 shows some examples of the basis vectors found, and what kinds of invariances in the images they account for. These are digit specific features; for example, a prominent basis vector for the family of 2s accounts for big versus small loops. Each of the examples shown accounts for a similar digit specific invariance. None of these changes are accounted for by the 7-dimensional tangent models, which were chosen to be digit non-specific. Note that the SVD without tangent distance would tend to mix the affine invariances with these digit specific invariances.

To classify a new image, its tangent distance is computed to each of the subspace models, and assigned to the class of the closest. For the USPS data we achieved 4.1% errors using a single subspace model per class; see table 1.

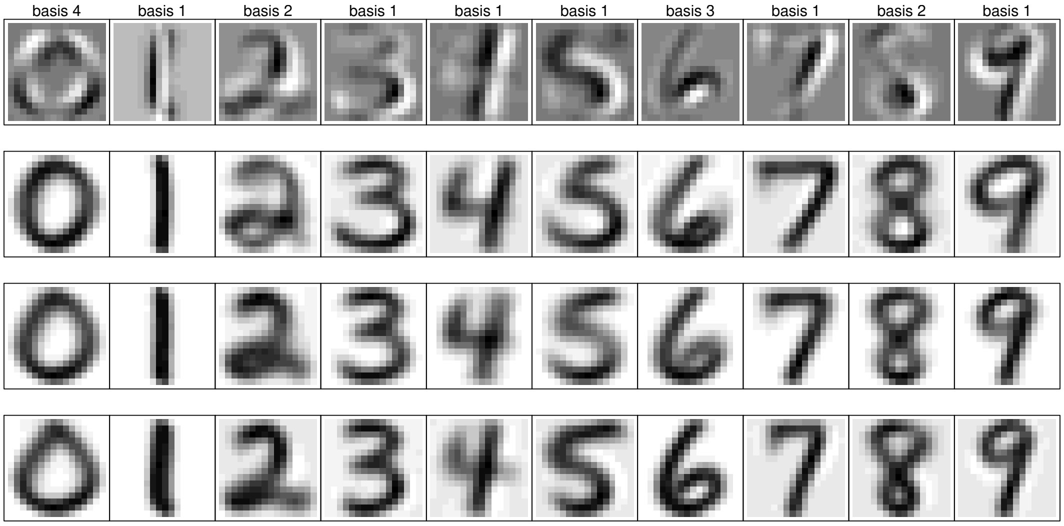

Figure 6: *Each column corresponds to a particular tangent subspace basis vector for the given digit. The top image is the basis vector itself, and the remaining 3 images correspond to the* 0.1, 0.5 *and* 0.9 *quantiles for the projection indices for the training data for that basis vector, showing a range of image models for that basis, keeping all the others at* 0.

4 Subspace models and K-means clustering

A natural extension of these single prototype-per-class models, is to use them as centroid modules in a K-means algorithm. The extension is obvious, and we summarize it in algorithmic form for the tangent subspace model (the cluster algorithm for the tangent centroid model is trivially similar). Note that a model of this kind is fit for each of the 10 digit classes.

Tangent subspace K-means algorithm

Initialize:
1. Choose a value for K.
2. Fit a regular K-means cluster model to the raw images, filtered down via a 64 dimensional smooth basis (we use 10 independent starts and pick the best solution)
3. Partition the data into K classes depending on which K-means centroid is closest.

Iterate:
1. For each of the K-classes fit a separate tangent subspace model.
2. Compute the tangent distance of each observation to the K subspace models, and reassign their class memberships to the closest model. Let $D_i^{\min}$ be the tangent distance to the closest model.
3. Compute $D = \sum_{i=1}^{N} D_i^{\min}$.

Until: D converges.

In the initialization step, we replace the images by their coordinates in a smooth 64 dimensional tensor-product basis of splines. The smoothing tends to smear pixels, which is a poor-man's noisy way of building in invariances. This allows us to rapidly try several starting solutions for the K-means algorithm. Each of the cluster centers require iteration, and these get computed repeatedly, often with very few membership changes. We limit the number of iterations, and by the time the whole algorithm has converged, all these cluster centers have converged as well. In a similar way the tangent centroid or subspace models can be used to seed LVQ algorithms (Kohonen 1989), but so far we have not much experience with them.

Table 1: *Test errors for a variety of situations. In all cases the training data were 7291 USPS handwritten digits, and the test data the "official" 2007 USPS test digits. Each entry describes the model used in each class, so for example in row 5 there are 5 models per class, hence 50 in all.*

Model	Prototype	Metric	# Prototypes/Class	Error Rate
1	1-NN	Euclidean	≈ 700	0.053
2	12 dim SVD subspace	Euclidean	1	0.055
3	12 dim SVD subspace	Tangent	1	0.045
4	12 dim Tangent subspace	Tangent	1	0.041
5	12 dim Tangent subspace	Tangent	3	0.038
6	12 dim Tangent subspace	Tangent	5	0.038
7	Tangent centroid	Tangent	20	0.038
8	(5) $\cup$ (7)	Tangent	23	0.034
9	1-NN	Tangent	≈ 700	0.026

5 Results

Table 1 summarizes the results for some of these models. Models 1 and 9 are both 1-NN, and the latter uses tangent distance and achieves the best error rate. Model 2 and 3 correspond to a SVD model for the images fit by ordinary least squares rather than least tangent squares. Model 2 classifies using Euclidean distance, model 3 using tangent distance. Model 4 fits a single 12-dimensional *tangent subspace* model per class, while models 5 and 6 use 12-dimensional tangent subspaces as cluster centers within each class. We tried other dimensions in a variety of settings, but 12 seemed to be generally the best. Model 7 corresponds to the *tangent centroid* model used as the centroid in a 20-means cluster model per class; the performance compares with with K=3 for the subspace model. Model 8 combines 5 and 7, and reduces the error even further. These limited experiments suggest that the tangent subspace model is preferable, since it is more compact and the algorithm for fitting it is on firmer theoretical grounds.

Notice that the performance can deteriorate (or at least not improve) if we continue to add more prototypes, or increase the dimension of the tangent subspace models. This is again the bias/variance tradeoff in operation. Adding more parameters creates models that fit the training data better. For parametric families of models this improved fit can be achieved in a rather aggressive way, and lead to models that do not generalize well when tested on independent data.

Figure 7 shows some of the misclassified examples in the test set. Despite all the matching, it seems that Euclidean distance still fails us in the end in some of these cases.

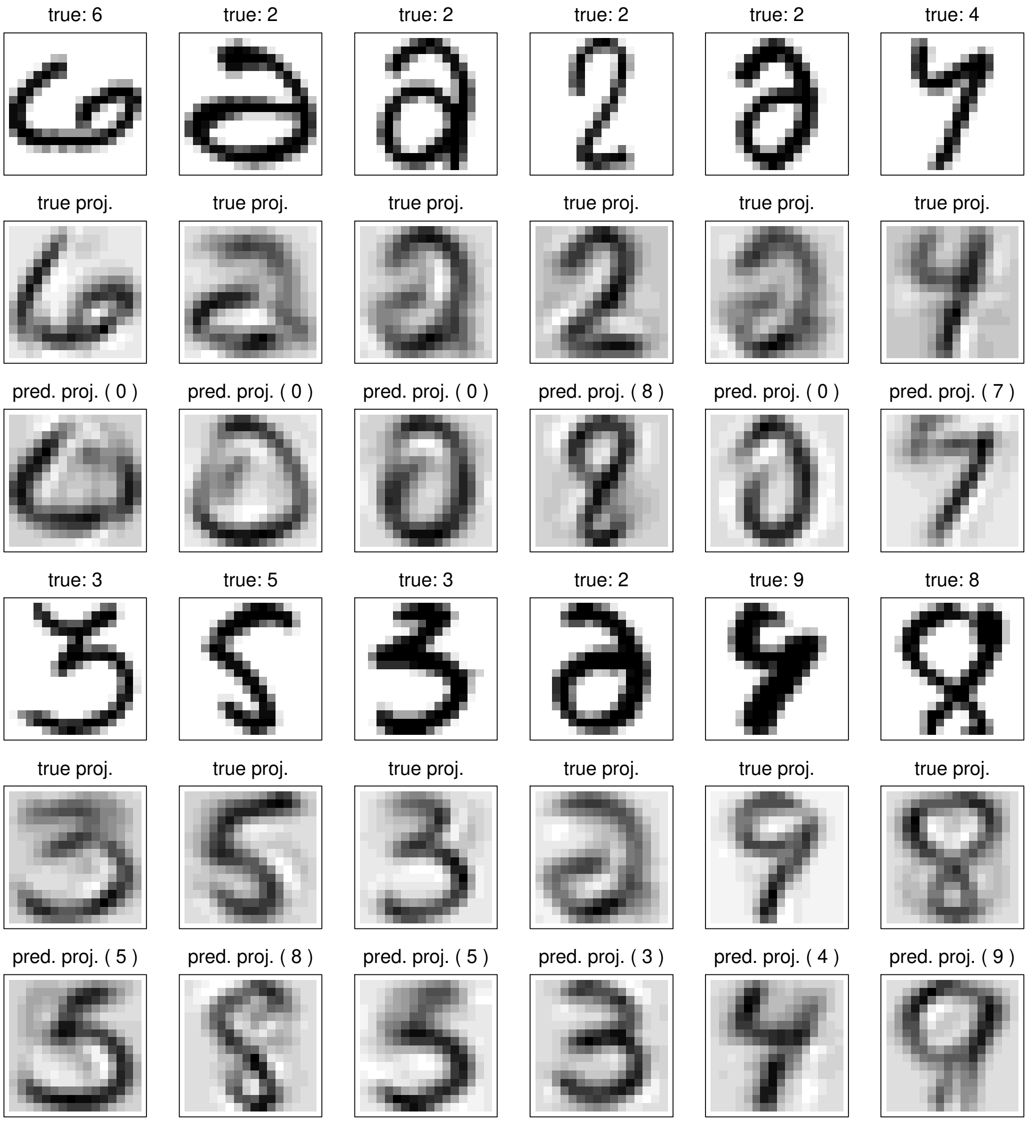

Figure 7: *Some of the errors for the test set corresponding to line (3) of table 1. Each case is displayed as a column of three images. The top is the true image, the middle the tangent projection of the true image onto the subspace model of its class, the bottom image the tangent projection of the image onto the winning class. The models are sufficiently rich to allow distortions that can fool Euclidean distance.*

6 Other approaches

We tried other approaches that exploited the tangent distance, but were unsuccessful on these test data. We briefly outline some of these, since they may be useful in other settings.

Stochastic image families

One can think of θ in the model $X_i(\theta) = X_i + T_i\theta$ as being a random variable, and hence generating a stochastic family of deformed versions of the image X_i. This would typically generate a cloud of images centered at X_i, confined to the subspace spanned by T_i. This model can be used to generate more images for each digit if these are required. One use of such a construction is to fit centroid models that get close to the image families in an average sense rather than a minimum sense. For example, the criterion for the tangent subspace model would be

$$\sum_{i=1}^{N} E_{\theta|X_i} \min_{\gamma(\theta)} \|M + V\gamma(\theta) - X_i(\theta)\|^2 \tag{9}$$

Assuming further that $\theta|X_i \sim N(0, \Sigma_\theta)$, and without any loss in generality that $\frac{1}{N}\sum_{i=1}^{N} X_i = 0$, a closed form solution is available: $M = 0$ and V is given by the appropriate number of eigenvectors of

$$\sum_{i=1}^{N} E_{\theta|X_i}(X_i + T_i\theta)(X_i + T_i\theta)^T = \sum_{i=1}^{N}\left(X_i X_i^T + T_i \Sigma_\theta T_i^T\right) \tag{10}$$

We tried fitting a single model of this kind for each of the digit classes, using $\Sigma_\theta = \sigma^2 I$ for various values of σ^2 and dimensions. Given the solution subspace, we classified new observations as before, using tangent distance. The best performance (in classifying the test data) was attained for $\sigma^2 = 0$ and a dimension of 12, which shows empirically that this approach only does worse than even the ordinary SVD! We do not fully understand this phenomenon, and can only deduce that the spherical prior for θ is inappropriate.

Other metrics

We are using Euclidean distance in conjunction with tangent distance. Since neighboring pixels are correlated, one might expect that a metric that accounted for the correlation might do better. We tried a few such approaches for the model with the 20 clusters with tangent centroids described in table 1. For each digit class we computed the *pooled within cluster covariance matrix* S_j, $j = 0, 1, \ldots, 9$, from the training data. We then modified the definition of tangent distance to accommodate this metric:

$$D^T(X_i, M_{jk}) = \min_{\theta_i, \theta_{jk}} \left(X_i(\theta_i) - M_{jk}(\theta_{jk})\right)^T S_i^{-1} \left(X_i(\theta_i) - M_{jk}(\theta_{jk})\right),$$

where M_{jk} is the kth tangent centroid for the jth digit class. In our investigations, we only used this distance for classifying the test data (i.e we did not re-learn the cluster models). The performance was worse than Euclidean distance. We also tried 2×2 variants in which

- We replaced S_i by a *regularized version* $S_i + \Omega$ to enforce stability and spatial smoothness of the metric (Hastie, Tibshirani & Buja 1994).
- We corrected each distance for the *size* of the covariance in a way consistent with Gaussian likelihood-ratio tests, by adding the term $\log|S_i|$ to the distance.

Neither of these gave any improvements either.

We also tried to incorporate information about where the images project in the tangent subspace models into the classification rule. We thus computed two distances: 1) tangent distance to the subspace, and 2) Mahalanobis distance within the subspace to the centroid for the subspace. Again the best performance was attained by ignoring the latter distance.

7 Discussion

Gold, Mjolsness & Rangarajan (1994) independently had the idea of using "domain specific" distance measures to seed K-means clustering algorithms. Their setting was slightly different from ours, and they did not use subspace models. The idea of classifying points to the closest subspace is found in the work of Oja (1989), but of course not in the context of tangent distance.

Our models are fit separately in each class, without any concerns of overlap. Here we remind the reader of the distinction between gaussian discriminant analysis and logistic regression; the latter are fit by conditional maximum likelihood, often termed *discriminative learning*. An alternative approach in the context of subspace models might be to embed them in a polychotomous logistic regression model. We are currently exploring models of this kind, and more generally discriminative versus non-discriminative learning in a variety of different contexts. Our preliminary experience is that we do not see any improvement, but the jury is still out.

In conclusion, learning tangent centroid and subspace models is an effective way to reduce the number of prototypes (and thus the cost in speed and memory) at a slight expense in the performance. In the extreme case, as little as one 12 dimensional tangent subspace per class and the tangent distance is enough to outperform classification using ≈ 700 prototypes per class and the Euclidean distance (4.1% versus 5.3% on the test data).

References

Bishop, C. (1995), *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford.

Boser, B., Guyon, I. & Vapnik, I. (1992), A training algorithm for optimal margin classifiers, *in* 'Proceedings of COLT II', Philadelphia, Pa.

Friedman, J. & Stuetzle, W. (1981), 'Projection pursuit regression', *Journal of the American Statistical Association* **76**, 817–823.

Gold, S., Mjolsness, E. & Rangarajan, A. (1994), Clustering with a domain specific distance measure, *in* 'Advances in Neural Information Processing Systems', Morgan Kaufman, San Mateo, CA.

Hastie, T., Buja, A. & Tibshirani, R. (1995), 'Penalized discriminant analysis', *Annals of Statistics* **23**(1), 73–102.

Hastie, T. & Tibshirani, R. (1993), Handwritten digit recognition via deformable prototypes. unpublished manuscript.

Hastie, T. & Tibshirani, R. (1996), 'Discriminant analysis by gaussian mixtures', *J. Royal Statist. Soc. (Series B)* **58**, 155–176.

Hastie, T., Tibshirani, R. & Buja, A. (1994), 'Flexible discriminant analysis by optimal scoring', *Journal of the American Statistical Association* **89**, 1255–1270.

Kohonen, T. (1989), *Self-Organization and Associative Memory (3rd edition)*, Springer-Verlag, Berlin.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R., Hubbard, W. & Jackel, L. (1990), Handwritten digit recognition with a back-propogation network, *in* D. Touretzky, ed., 'Advances in Neural Information Processing Systems', Vol. 2, Morgan Kaufman, Denver, CO.

Oja, E. (1989), 'Neural networks, principal components, and subspaces', *International Journal Of Neural Systems* **1**(1), 61–68.

Ripley, B. D. (1996), Pattern recognition and neural networks. Cambridge University Press.

Säckinger, E. (1992), Recurrent networks for elastic matching in pattern recognition, Technical report, AT&T Bell Laboratories.

Simard, P. Y., LeCun, Y. & Denker, J. (1993), Efficient pattern recognition using a new transformation distance, *in* 'Advances in Neural Information Processing Systems', Morgan Kaufman, San Mateo, CA, pp. 50–58.

Vapnik, V. (1996), *The Nature of Statistical Learning*, Springer-Verlag.

A Tangent Models

In this section we give a derivation of the tangent model different from those that have appeared before. We use a functional approach, and then view the digitized images as discretized versions of these.

Suppose we represent an image prior to digitization as a differentiable function $F : R^2 \mapsto R$; i.e. $F(z)$ gives the greyscale value at spatial location z. The family of functions generated by the six dimensional affine transformations can be represented as

$$\begin{aligned} F^I(z,\mu,A) &= F(z_0+\mu+A(z-z_0)) \\ &= F(Z(z,z_0,\mu,A)) \end{aligned} \tag{11}$$

where

- μ accounts for location shifts,
- z_0 is the center of rotation, scaling and shear, and
- A is a 2×2 transformation matrix with factorization $A = R(\theta)T$, where R is a rotation matrix and T an upper-triangular scale/shear matrix.

These affine transformations act by altering the points $z = (x, y)$ at which we reference F. The first order Taylor series approximation to this family about suitable *null* transformations has the form

$$F^T(z,\mu,A) = F(z) + \sum_{\alpha\in\{\mu,\theta,T\}} \frac{\partial F}{\partial \alpha}(\alpha-\alpha_0) \tag{12}$$

$$= F(z) + \nabla F(z)^T \sum_{\alpha\in\{\mu,\theta,T\}} \frac{\partial Z(z,z_0,\mu,A)}{\partial \alpha}(\alpha-\alpha_0) \tag{13}$$

This leads to the following six *derivative (tangent)* functions $F_\alpha(z)$:

- *x-location:* $\alpha = \mu_1$ and $F_\alpha = F_x(z) = \frac{\partial F(z)}{\partial x}$
- *y-location:* $\alpha = \mu_2$ and $F_\alpha = F_y(z) = \frac{\partial F(z)}{\partial y}$
- *x-scale:* $\alpha = T_{11}$ and $F_\alpha = (x - x_0)F_x(z)$
- *y-scale:* $\alpha = T_{22}$ and $F_\alpha = (y - y_0)F_y(z)$
- *Rotation:* $\alpha = \theta$ and $F_\alpha = (y - y_0)F_x(z) - (x - x_0)F_y(z)$
- *Shear:* $\alpha = T_{12}$ and $F_\alpha = (y - y_0)F_x(z) + (x - x_0)F_y(z)$

Finally, based on entirely intuitive grounds, the thickness derivative ($\alpha =$*thickness*) is given by $F_\alpha(z) = F_x(z)^2 + F_y(z)^2$ (when displayed as an image this function look like the outline of F.)

A digitized image can be thought of as F sampled at a lattice of points $z_{ij} = (x_i, y_j)$ (or integrated over rectangles defined by them). As we move from functions to digitized functions, the Taylor approximation becomes a tangent subspace to the digitized manifold F^I. To implement the approximation, we need the derivatives F_x and F_y evaluated at the same set of lattice points. Several approaches can be used to approximate these derivatives:

1. Use first differences in each direction
2. Convolve the image with a smooth bivariate kernel, and then differentiate. In practice this implies differentiating the kernel first (separately in x and y) and then convolving. We used the kernel $k_h(z_1, z_2) = (1/h)\exp(-\|z_1 - z_2\|^2/2h)$.
3. Smooth the image first, but then use first differences as in (1)

All these techniques have approximately the same performance.