Memory Based Character Recognition Using a Transformation Invariant Metric

Patrice Y. Simard Yann Le Cun
John S. Denker

AT&T Bell Laboratories
Holmdel, NJ 07733

Abstract

Memory-based classification algorithms such as Radial Basis Functions or K-nearest neighbors often rely on simple distances (Euclidean distance, Hamming distance, etc.), which are rarely meaningful on pattern vectors. More complex, better suited distance measures are often expensive and rather ad-hoc (elastic matching, deformable templates). We propose a new distance measure which (a) can be made locally invariant to any set of transformations of the input and (b) can be computed efficiently. We tested the method on large handwritten character databases provided by the Post Office and the NIST. Using invariances with respect to translation, rotation, scaling, skewing and line thickness, the method outperformed all other systems on small (less than 10,000 patterns) database and was competitive on our largest (60,000 patterns) database.

1 INTRODUCTION

Distance-based classification algorithms such as radial basis functions or K-nearest neighbors often rely on simple distances (such as Euclidean distance, Hamming distance, etc.). As a result, they suffer from high sensitivity to simple transformations of the input patterns that should leave the classification unchanged (e.g. translation or scaling for 2D images). This is illustrated in Fig. 1 where an unlabeled image of a "9" must be classified by finding the closest prototype image out of two images representing respectively a "9" and a "4". According to the Euclidean distance (sum of the squares of the pixel to pixel differences), the "4" is closer even though the "9" is much more similar once it has been rotated and thickened. The result is an incorrect classification. The key idea is to construct a distance measure which is invariant with respect to some chosen transformations such as translation, rotation and others. The special case of linear transformations has been well studied in statistics and is

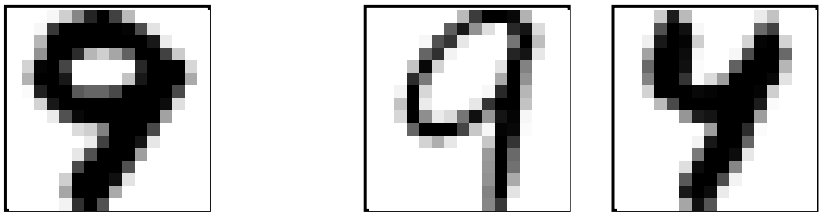

Figure 1: What is a good similarity measure? According to the Euclidean distance the pattern to be classified is more similar to prototype B. A better distance measure would find that prototype A is closer because it differs mainly by a rotation and a thickness transformation, two transformations which should leave the classification invariant.

sometimes referred to as Procrustes analysis (Sibson, 1978). It has been applied to signature verification (Hastie et al., 1991) and on-line character recognition (Sinden and Wilfong, 1992).

This paper considers the more general case of non-linear transformations such as geometric transformations of gray-level images. Remember that even a simple image translation corresponds to a highly non-linear transformation in the high-dimensional pixel space[1]. In previous work (Simard et al., 1992b), we showed how a neural network could be trained to be invariant with respect to selected transformations of the input. We now apply similar ideas to distance-based classifiers.

When a pattern P is transformed (e.g. rotated) with a transformation s that depends on one parameter α (e.g. the angle of the rotation), the set of all the transformed patterns $S_P = \{x \mid \exists \vec{\alpha} \text{ for which } x = s(\vec{\alpha}, P)\}$ is a one-dimensional curve in the vector space of the inputs (see Fig. 2). In certain cases, such as rotations of digitized images, this curve must be made continuous using smoothing techniques (see (Simard et al., 1992b)). When the set of transformations is parameterized by n parameters α_i (rotation, translation, scaling, etc.), S_P is a manifold of at most n dimensions. The patterns in S_P that are obtained through *small* transformations of P, i.e. the part of S_P that is close to P, can be approximated by a plane tangent to the manifold S_P at the point P. Small transformations of P can be obtained by adding to P a linear combination of vectors that span the tangent plane (tangent vectors). The images at the bottom of Fig. 2 were obtained by that procedure. Tangent vectors for a transformation s can easily be computed by finite difference (evaluating $\partial s(\alpha, P)/\partial \alpha$); more details can be found in (Simard et al., 1992b; Simard et al.,

[1] If the image of a "3" is translated vertically upward, the middle top pixel will oscillate from black to white three times.

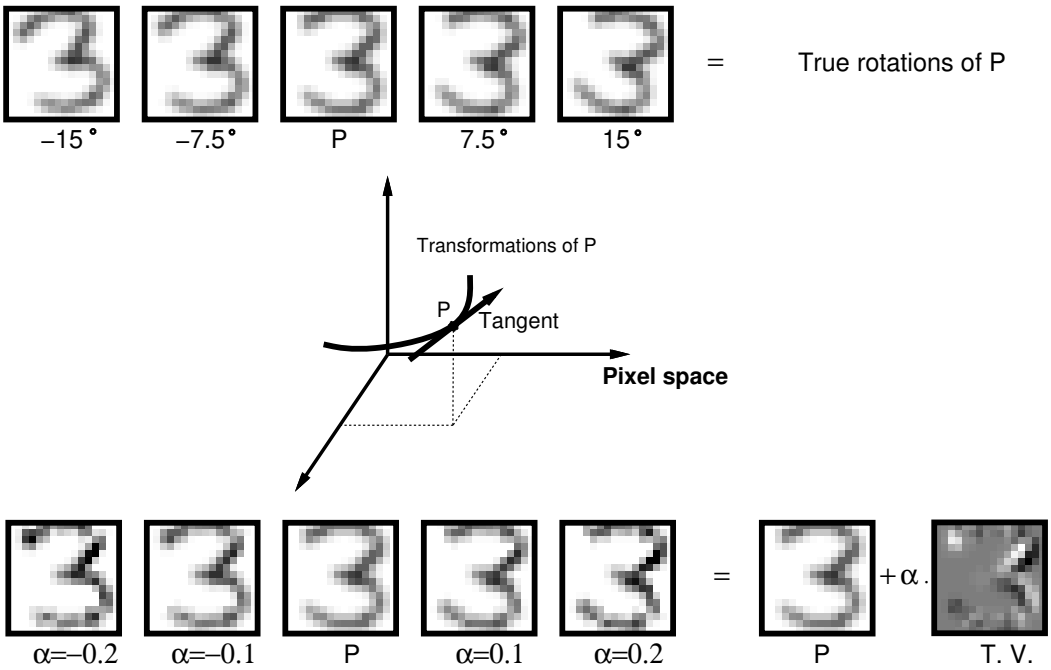

Figure 2: Top: Small rotations of an original digitized image of the digit "3". Middle: Representation of the effect of the rotation in pixel space (if there were only 3 pixels). Bottom: Images obtained by moving along the tangent to the transformation curve for the same original digitized image P by adding various amounts (α) of the tangent vector (T.V.).

1992a).

As we mentioned earlier, the Euclidean distance between two patterns P and E is often sub-optimal because it is sensitive to irrelevant transformations of P and of E. In contrast, the distance $\mathcal{D}(E, P)$ defined to be the minimal distance between the two manifolds S_P and S_E is truly invariant with respect to the transformations used to generate S_P and S_E. Unfortunately, these manifolds have no analytic expression in general, and finding the distance between them is a hard optimization problem with multiple local minima. Besides, true invariance is not necessarily desirable since a rotation of a "6" into a "9" does not preserve the correct classification.

Our approach consists of computing the minimum distance between the linear surfaces that best approximate the non-linear manifolds S_P and S_E. This solves three problems at once: 1) linear manifolds have simple analytical expressions which can be easily computed and stored, 2) finding the minimum distance between linear manifolds is a simple least squares problem which can be solved efficiently and, 3) this distance is locally invariant but not globally invariant.

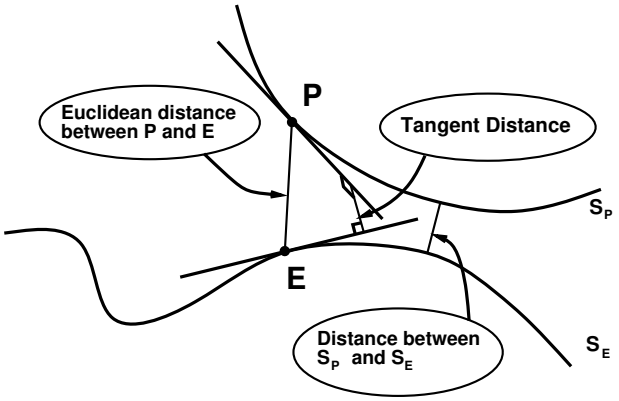

Figure 3: Illustration of the Euclidean distance and the tangent distance between P and E

Thus the distance between a "6" and a slightly rotated "6" is small but the distance between a "6" and a "9" is large. The different distances between P and E are represented schematically in Fig. 3.

The figure represents two patterns P and E in N-dimensional space. The manifolds generated by s are represented by one-dimensional curves going through E and P respectively. The linear approximations to the manifolds are represented by lines tangent to the curves at E and P. These lines do not intersect in N dimensions and the shortest distance between them (uniquely defined) is $D(E, P)$. The distance between the two non-linear transformation curves $\mathcal{D}(E, P)$ is also shown on the figure.

An efficient implementation of the tangent distance $D(E, P)$ will be given in the next section. Although the tangent distance can be applied to any kind of patterns represented as vectors, we have concentrated our efforts on applications to image recognition. Comparison of tangent distance with the best known competing method will be described. Finally we will discuss possible variations on the tangent distance and how it can be generalized to problems other than pattern recognition.

2 IMPLEMENTATION

In this section we describe formally the computation of the tangent distance. Let the function s transform the input point u to $s(\vec{\alpha}, u)$ according to the parameter $\vec{\alpha}$. We require s to be differentiable with respect to $\vec{\alpha}$ and u, and require $s(0, u) = u$. If u is a 2 dimensional image for instance, $s(\vec{\alpha}, u)$ could be a rotation of u by the angle $\vec{\alpha}$. If we are interested in all transformations of images which conserve distances (isometry), $s(\vec{\alpha}, u)$ would be a rotation by α_r followed by a translation by α_x, α_y of the image u. In this case $\vec{\alpha} = (\alpha_r, \alpha_x, \alpha_y)$ is a vector of parameters of dimension 3. In general, $\vec{\alpha} = (\alpha_0, \ldots, \alpha_{m-1})$ is of

dimension m.

Since s is differentiable, the set $S_u = \{x \mid \exists \vec{\alpha} \text{ for which } x = s(\vec{\alpha}, u)\}$ is a differentiable manifold which can be approximated to the first order by a hyperplane T_u. This hyperplane is tangent to S_u at u and is generated by the columns of matrix

$$L_u = \left.\frac{\partial s(\vec{\alpha}, u)}{\partial \vec{\alpha}}\right|_{\vec{\alpha}=\vec{0}} = \left[\frac{\partial s(\vec{\alpha}, u)}{\partial \alpha_0}, \ldots, \frac{\partial s(\vec{\alpha}, u)}{\partial \alpha_{m-1}}\right]_{\vec{\alpha}=\vec{0}} \tag{1}$$

which are vectors tangent to the manifold. If E and P are two patterns to be compared, the respective tangent planes T_E and T_P can be used to define a new distance D between these two patterns. The tangent distance $D(E, P)$ between E and P is defined by

$$D(E, P) = \min_{x \in T_E, y \in T_P} \|x - y\|^2 \tag{2}$$

The equation of the tangent planes T_E and T_P is given by:

$$E'(\vec{\alpha}_E) = E + L_E \vec{\alpha}_E \tag{3}$$
$$P'(\vec{\alpha}_P) = P + L_P \vec{\alpha}_P \tag{4}$$

where L_E and L_P are the matrices containing the tangent vectors (see Eq. 1) and the vectors $\vec{\alpha}_E$ and $\vec{\alpha}_P$ are the coordinates of E' and P' in the corresponding tangent planes. The quantities L_E and L_P are attributes of the patterns so in many cases they can be precomputed and stored.

Computing the tangent distance

$$D(E, P) = \min_{\vec{\alpha}_E, \vec{\alpha}_P} \|E'(\vec{\alpha}_E) - P'(\vec{\alpha}_P)\|^2 \tag{5}$$

amounts to solving a linear least squares problem. The optimality condition is that the partial derivatives of $D(E, P)$ with respect to $\vec{\alpha}_P$ and $\vec{\alpha}_E$ should be zero:

$$\frac{\partial D(E, P)}{\partial \vec{\alpha}_E} = 2(E'(\vec{\alpha}_E) - P'(\vec{\alpha}_P))^\top L_E = 0 \tag{6}$$
$$\frac{\partial D(E, P)}{\partial \vec{\alpha}_P} = 2(P'(\vec{\alpha}_P) - E'(\vec{\alpha}_E))^\top L_P = 0 \tag{7}$$

Substituting E' and P' by their expressions yields to the following linear system of equations, which we must solve for $\vec{\alpha}_P$ and $\vec{\alpha}_E$:

$$L_P^\top (E - P - L_P \vec{\alpha}_P + L_E \vec{\alpha}_E) = 0 \tag{8}$$
$$L_E^\top (E - P - L_P \vec{\alpha}_P + L_E \vec{\alpha}_E) = 0 \tag{9}$$

The solution of this system is

$$(L_{PE} L_{EE}^{-1} L_E^\top - L_P^\top)(E - P) = (L_{PE} L_{EE}^{-1} L_{EP} - L_{PP})\vec{\alpha}_P \tag{10}$$
$$(L_{EP} L_{PP}^{-1} L_P^\top - L_E^\top)(E - P) = (L_{EE} - L_{EP} L_{PP}^{-1} L_{PE})\vec{\alpha}_E \tag{11}$$

where $L_{EE} = L_E^\top L_E$, $L_{PE} = L_P^\top L_E$, $L_{EP} = L_E^\top L_P$ and $L_{PP} = L_P^\top L_P$. LU decompositions of L_{EE} and L_{PP} can be precomputed. The most expensive part in solving this system is evaluating L_{EP} (L_{PE} can be obtained by transposing L_{EP}). It requires $m_E \times m_P$ dot products, where m_E is the number of tangent vectors for E and m_P is the number of tangent vectors for P. Once L_{EP} has been computed, $\vec{\alpha}_P$ and $\vec{\alpha}_E$ can be computed by solving two (small) linear system of respectively m_E and m_P equations. The tangent distance is obtained by computing $\|E'(\vec{\alpha}_E) - P'(\vec{\alpha}_P)\|$ using the value of $\vec{\alpha}_P$ and $\vec{\alpha}_E$ in equations 3 and 4. If n is the length of vector E (or P), the algorithm described above requires roughly $n(m_E + 1)(m_P + 1) + 3(m_E^3 + m_P^3)$ multiply-adds. Approximations to the tangent distance can be computed more efficiently, as we now discuss.

3 OPTIMIZATION

Memory based algorithms are generally limited by the extensive computational requirements due to the large number of distance computations. If no optimization technique is used, the computational cost is given in equation 12.

$$\text{computational cost} \approx \begin{array}{c}\text{number of}\\ \text{prototypes}\end{array} \times \begin{array}{c}\text{distance}\\ \text{complexity}\end{array} \tag{12}$$

Fortunately these algorithms are well suited for optimization using hierarchies of prototypes, and that this is even more true when the distance complexity is high. In this section, we increase the recogntion speed in two ways: 1) Finding the closest prototype can be done by recursively searching included subsets of the database using distances of increasing complexity. This is done by using a hierarchy of tangent distances (increasing the number of tangent vectors from 0 to its maximum) and multiresolution (using wavelets). 2) A confidence level can be computed for each distance. If the confidence in the classification is above a threshold early on, there is no need to compute the more expensive distances.

3.1 FILTERING USING A HIERARCHY OF DISTANCES

Our goal is to compute the distance from one unknown pattern to every prototype in a large database in order to determine which one is the closest. Some patterns are so different from each other that a very crude approximation of our distance can tell us so. There is a wide range of variation in computation time (and performance) depending on the choice of the distance. For instance, computing the Euclidean distance with n pixels is n/k times as expensive as computing it with k pixels. Similarly, at a given resolution, computing the tangent distance with m tangent vectors is $(m+1)^2$ times as expensive as computing the Euclidean distance ($m = 0$ tangent vectors).

This observation provided us with a hierarchy of about a dozen different distances ranging in computation time from 4 multiply-adds (Euclidean distance on a 2×2 averaged image) to 20,000 multiply-adds (tangent distance,

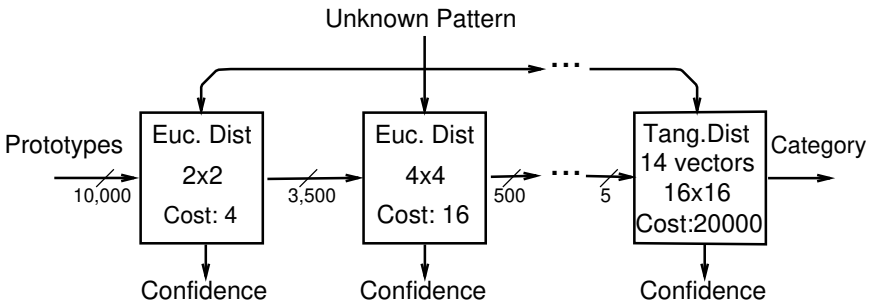

Figure 4: Pattern recognition using a hierarchy of distance. The filter proceed from left (starting with the whole database) to right (where only a few prototypes remain). At each stage distances between prototypes and the unknown pattern are computed, sorted and the best candidate prototypes are selected for the next stage. As the complexity of the distance increases, the number of prototypes decreases, making computation feasible. At each stage a classification is attempted and a confidence score is computed. If the confidence score is high enough, the remaining stages are skipped.

7 tangent vectors, 16×16 pixel images). The resulting filtering algorithm is straightforward and is diagrammed in Figure 4.

The general idea is to store the database of prototypes several times at different resolutions and with different tangent vectors. Each of these resolutions and groups of tangent vectors defines a distance d_i. These distances are ordered in increasing accuracy and complexity. The first distance d_1 is computed on all (K_0) prototypes of the database. The closest K_1 patterns are then selected and identified to the next stage. This process is repeated for each of the distances; i.e. at each stage i, the distance d_i is computed on each K_{i-1} patterns selected by the previous stage. Of course, the idea is that as the complexity of the distance increases, the number of patterns on which this distance must be computed decreases. At the last stage, the most complex and accurate distance is computed on all remaining patterns to determine the classification.

The only difficult part is to determine how many K_i prototypes should be selected at each stage to maximize speed without compromising accuracy. Note that if the last distance used is the most accurate distance, setting all K_i to the number of patterns in the database will give optimal performance (at the most expensive cost). Increasing K_i always improves the performance in the sense that it allows to find patterns that are closer for the next distance measure d_{i+1}.

The simplest way to determine K_i is by selecting a validation set and plotting the performance on this validation set as a function of K_i. The optimal K_i is then determined graphically.

This method is very useful when the performance is not degraded by choosing small K_i. In this case, the distance evaluation is done using distance metrics which are relatively inexpensive to compute. The computation cost becomes:

$$\text{computational cost} \approx \sum_i \begin{array}{l}\text{number of}\\ \text{prototypes}\\ \text{at stage } i\end{array} \times \begin{array}{l}\text{distance}\\ \text{complexity}\\ \text{at stage } i\end{array} \tag{13}$$

3.2 PRUNING THE SEARCH USING CONFIDENCE SCORES

If a confidence score is computed at each stage of the distance evaluation, it is possible for certain patterns to avoid computing the most expensive distances for certain patterns. In the extreme case, if the Euclidean distance between two patterns is 0, there is really no need to compute the tangent distance. A crude but effective way to compute a confidence score at a given stage i is to find the closest prototype (for distance d_i) in each of the possible classes. The distance difference between the closest class and the next closest class gives an approximation of a confidence of this classification. A simple algorithm is then to compare at stage i the confidence score c_{ip} of the current unknown pattern p to a threshold θ_i, and to stop the classification process for this pattern as soon as $c_{ip} > \theta_i$. The classification will then be determined by the closest prototype at this stage. The computation time will therefore be different depending on the pattern to be classified. Easy patterns will be recognized very quickly while difficult patterns will need to be compared to some of the prototypes using the most complex distance. The total computation cost is therefore:

$$\text{computational cost} \approx \sum_i \begin{array}{l}\text{number of}\\ \text{prototypes}\\ \text{at stage } i\end{array} \times \begin{array}{l}\text{distance}\\ \text{complexity}\\ \text{at stage } i\end{array} \times \begin{array}{l}\text{probability}\\ \text{to reach}\\ \text{stage } i\end{array} \tag{14}$$

Note that if all θ_i are high, the performance is maximized but so is the cost. We therefore wish to find the smallest value of θ_i which does not degrade the performance (increasing θ_i always improves the performance). As in the previous section, the simplest way to determine the optimal θ_i is graphically with a validation set.

3.3 DISCUSSION

Several hierarchies of distance are possible for optimizing the search process. One of the main advantages of the multiresolution approach is that it is easily implemented with wavelet transforms (Mallat, 1989), and that in the wavelet

space, the tangent distance is conserved (with orthonormal wavelet bases). Furthermore, the multiresolution decomposition is completely orthogonal to the tangent distance decomposition. In our experiments, the Haar transform was used.

An incremental nearest neighbor search algorithm based on k-d tree (Broder, 1990) was also implemented. The k-d tree structure was interesting because it can potentially be used with tangent distance. It turned out however that in large dimensional space, the distance from a point to a hyperplane is almost always smaller than the distance between any pair of points. As a result, the unknown pattern must be compared to many prototypes to have a reasonable accuracy. The speed up factor was comparable to our multiresolution approach in the case of Euclidean distance (about 10), but we have not been able to obtain both good performance and high speedup with the k-d tree algorithm applied to tangent distance. This algorithm was not used in our final experiments.

Our most successful hierarchy consisted of adding tangent vectors one by one, on both sides. Even though this implies solving a new linear system at each stage, the computational cost is dominated by computing dot products between tangent vectors. These dot products are then reused in the subsequent stages to create larger linear systems (involving more tangent vectors). This hierarchy has the advantage that the first stage is only twice as expensive, yet much more accurate, than the Euclidean distance. Each subsequent stage brings a lot of accuracy at a reasonable cost. (The cost increases quicker toward the later stages since solving the linear system grows with the cube of the number of tangent vectors.) The last stage is exactly the full tangent distance.

Obviously, the tangent vectors can be added in different orders. We did not try to find the optimal order. For character recognition applications, adding translations first, followed by hyperbolic deformations, scalings, thickness deformations and rotations yielded good performance.

Using the multiresolution scheme and adding the tangent vectors incrementally gave a total speed-up of about 500 (i.e. 3 characters per second on a 10,000 prototype database). The computation was spread among the various stages (more than 50% of the computation was done at resolutions of 8×8 or smaller, with Euclidean distance).

4 RESULTS

Before giving the results of handwritten digit recognition experiments, we would like to demonstrate the property of "local invariance" of tangent distance. A 16×16 pixel image similar to the "3" in Fig 2 was translated by various amounts. The tangent distance (using the tangent vector corresponding to horizontal translations) and the Euclidean Distance between the original image and its translated version were measured as a function of the magnitude k (in pixels) of the translation. The result is plotted in Fig. 5. It is clear that the Euclidean

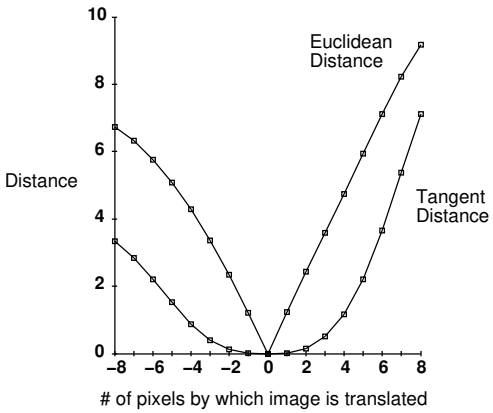

Figure 5: Euclidean and tangent distances between a 16x16 handwritten digit image and its translated version as a function of the amount of translation measured in pixels.

Distance increases steeply and more-or-less linearly with k while the tangent distance remains very small for translations as large as two pixels. This indicates that, while Euclidean Distance is not invariant to translation, tangent distance is locally invariant. The extent of the invariance can be increased by smoothing the original image, but significant features may be blurred away, leading to confusion errors. The figure is not symmetric for large translations because the translated image is truncated to the 16×16 pixel field of the original image. In the following experiments, smoothing was done by convolution with a Gaussian of standard deviation $\sigma = 0.75$. This value, which was estimated visually, turned out to be nearly optimal (but not critical).

4.1 Handwritten Digit Recognition

Experiments were conducted to evaluate the performance of tangent distance for handwritten digit recognition. An interesting characteristic of digit images is that we can readily identify a set of local transformations which do not affect the identity of the character, while covering a large portion of the set of possible *instances* of the character. Seven such image transformations were identified: X and Y translations, rotation, scaling, two hyperbolic transformations (which can generate shearing and squeezing), and line thickening or thinning. The first six transformations were chosen to span the set of all possible linear coordinate transforms in the image plane (nevertheless, they correspond to highly non-linear transforms in pixel space). Additional transformations have been tried with less success.

US Postal Service database: In the first experiment, the database consisted of 16×16 pixel size-normalized images of handwritten digits, coming from US mail envelopes. The entire training set of 9709 examples of was used as the

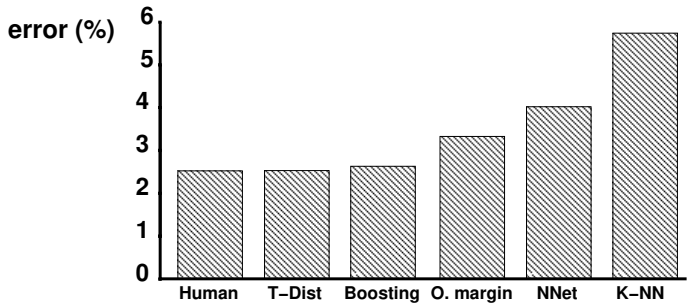

Figure 6: Comparison of the error rate of tangent nearest neighbors and other methods on two handwritten digit databases

.

prototype set. The test set contained 2007 patterns. The best performance was obtained with the "one nearest neighbor" rule. The results are plotted in Fig. 6. The error rate of the method is 2.5%. Two members of our group labeled the test set by hand with an error rate of 2.5% (using one of their labelings as the truth to test the other also yielded 2.5% error rate). This is a good indicator of the level of difficulty of this task[2].

The performance of a standard neural network (called "leNet 1", (Le Cun et al., 1990)) is 4% raw error while our best neural network using boosting (Drucker, Schapire and Simard, 1993) achieves 2.6%. The optimal margin classifier yielded 3.3% while the performance of one nearest neighbor with the Euclidean distance was 5.9%. These results show that tangent distance performs as well or better than both standard K-nearest neighbor and neural networks on small database.

NIST database: The second experiment was performed on a large database provided by the National Institute of Standards and Technology. NIST had divided the data into two sets which unfortunately had different distributions (the original training and testing sets). In our experiments we conbined these two sets 50/50 to make a training set of 60,000 patterns and a testing set of 10,000 patterns, both having the same charateristics.

On these sets we tested several algorithms. The description of each of these algorithms and their performance are reported in a companion paper (Bottou et al., 1994). Tangent distance was used with 3 nearest neighbors and the same set of tangent vectors and optimizations as with the USPS experiments. Remarquably, large neural networks, optimal margin classifier and tangent distance all tied at 1.1% raw error performance. Boosting, however, yielded better performance with 0.7% raw error.

[2]This is an extremely difficult test set. Procedures that achieve less than 0.5% error on other handwritten digit tasks barely achieve 4% on this one

A close examination of the errors made by tangent distance showed that many of the errors were due to pairing of testing pattern with prototypes which were very close in pixel space exept for a small differences such as open/closed loop ("5" versus "6", "9 versus 8", etc...). This suggest that if tangent distance were computed in a better feature space than the pixel space, the performance would improve.

5 DISCUSSION

The tangent distance algorithm described in the implementation section can be improved/adjusted in at least four ways: 1) choosing a better feature space than pixel space, 2) modifying the distance function itself, 3) changing the set of transformations/tangent vectors, and 4) using the tangent distance with classification algorithms other than K-nearest neighbors, perhaps in combination, to minimize the number of prototypes. We will discuss each of these aspects in turn.

Feature space: All the experiments discribed in the result section were computing distance in pixel space. We are well aware that this is one of the worst possible spaces to compare images. One very good candidate algorithm for better performance would be to use a trained neural network to do the preprocessing and feature extraction, and then use tangent distance in this feature space. The drawback of this method is that the tangent vectors in such a feature space cannot be computed from the new image in that space. They must either be stored (at additional cost in memory) or be propagated through the network (at additional cost in recognition time).

New distance function: Tangent distance can be viewed as one iteration of a Newton-type algorithm which finds the points of minimum distance on the true transformation manifolds. The vectors $\vec{\alpha_E}$ and $\vec{\alpha_P}$ are the coordinates of the two closest points in the respective tangent spaces, but they can also be interpreted for real (non-linear) transformations. If $\vec{\alpha_{E,i}}$ is the amount of the translation tangent vector that must be added to E to make it as close as possible to P, we can compute the true translation of image E by $\vec{\alpha_{E,i}}$ pixels. In other words, $E'(\alpha_E)$ and $P'(\alpha_P)$ are projected onto close points of S_E and S_P. This involves a resampling but can be done efficiently. Once this new image has been computed, the corresponding tangent vectors can be computed for this new image and the process can be repeated. Eventually this will converge to a local minimum in the distance between the two transformation manifold of P and E. The tangent distance needs to be normalized for this iteration process to work.

Other tangent vectors: The *a priori* knowledge embodied in the tangent vectors depends greatly on the application. For character recognition, thickness was one of the most important transformations, reducing the error rate from 3.3% to 2.6%. Such a transformation would be meaningless in, say, speech or

face recognition. Other transformations such as local rubber sheet deformations may be interesting for character recognition. Transformations can be known *a priori* or learned from the data.

Other algorithms, reducing the number of prototypes: Tangent distance is a general method that can be applied to problems other than image recognition, with classification methods other than K-nearest neighbors. Many distance-based classification schemes could be used in conjunction with tangent distance, among them LVQ (Kohonen, 1984), and radial basis functions. Since all the operators involved in the tangent distance are differentiable, it is possible to compute the partial derivative of the tangent distance (between an object and a prototype) with respect to the tangent vectors, or with respect to the prototype. Therefore the tangent distance operators can be inserted in gradient-descent based adaptive machines (of which LVQ and RBF are particular cases). The main advantage of learning the prototypes or the tangent vectors is that fewer prototypes may be needed to reach the same (or superior) level of performance as, say, regular K-nearest neighbors.

In conclusion, tangent distance can greatly improve many of the distance-based algorithms. Tangent distance used with K-nearest neighbors outperformed all existing techniques on a standard classification task. This success is probably largely due to the fact that *a priori* knowledge can be very effectively expressed in the form of tangent vectors. Many other algorithms are based on computing distances and can be adapted to express *a priori* knowledge in a similar fashion. Promising candidates include Parzen windows, learning vector quantization and radial basis functions.

References

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., and Vapnick, V. (1994). Comparison of Classifier Methods: A Case Study in Handwritten Digit Recognition. In *International conference on pattern recogntion (submitted)*.

Broder, A. J. (1990). Strategies for Efficient Incremental Nearest Neighbor Search. *Pattern Recognition*, 23:171–178.

Drucker, H., Schapire, R., and Simard, P. Y. (1993). Boosting Performance in Neural Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7, No. 4:705–719.

Hastie, T., Kishon, E., Clark, M., and Fan, J. (1991). A Model for Signature Verification. Technical Report 11214-910715-07TM, AT&T Bell Laboratories.

Kohonen, T. (1984). Self-organization and Associative Memory. In *Springer Series in Information Sciences*, volume 8. Springer-Verlag.

Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2 (NIPS*89)*, Denver, CO. Morgan Kaufman.

Mallat, S. G. (1989). A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11, No. 7:674–693.

Sibson, R. (1978). Studies in the Robustness of Multidimensional Scaling: Procrustes Statistices. *J. R. Statist. Soc.*, 40:234–238.

Simard, P. Y., LeCun, Y., Denker, J., and Victorri, B. (1992a). An Efficient Method for Learning Invariances in Adaptive classifiers. In *International Conference on Pattern Recognition*, volume 2, pages 651–655, The Hague, Netherlands.

Simard, P. Y., Victorri, B., LeCun, Y., and Denker, J. (1992b). Tangent Prop – A formalism for specifying selected invariances in an adaptive network. In *Neural Information Processing Systems*, volume 4, pages 895–903, San Mateo, CA.

Sinden, F. and Wilfong, G. (1992). On-line Recognition of Handwritten Symbols. Technical Report 11228-910930-02IM, AT&T Bell Laboratories.