NISTIR 5209

Comparison of Handprinted Digit Classifiers

Patrick J. Grother
Gerald T. Candela

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Advanced Systems Division
Gaithersburg, MD 20899

NIST

NISTIR 5209

Comparison of Handprinted Digit Classifiers

Patrick J. Grother
Gerald T. Candela

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Advanced Systems Division
Gaithersburg, MD 20899

June 1993

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arati Prabhakar, Director

Comparison of Handprinted Digit Classifiers

Patrick J Grother
Gerald T. Candela

Image Recognition Group
National Institute of Standards and Technology

May 24, 1993

Abstract

We report recognition results for several pattern classifiers trained and tested on disjoint sets of 30620 digits selected from the first 500 writers of NIST Special Database 3. The classifiers are ubiquitous in traditional pattern recognition literature (minimum distance, maximum *a posteriori*, nearest neighbor) as well as neural network literature (multilayer perceptron, radial basis functions, probabilistic neural network). For the purpose of valid comparison of classifiers fixed sets of Karhunen-Loève Transforms, were used as features. These were produced from images preprocessed using the fixed methods for size and orientation normalization, The "K-means" clustering algorithm is used to produce subclasses thereby supervising training and aiding recognition. Graphical displays of classification and associated confidences illustrate classifier complexity. Recognition error rates for all the classifiers are tabulated as a function of feature vector dimension. Computational and memory requirements of the different classifiers are also compared.

1 Introduction

Optical Character Recognition (OCR) has been a popular focus of Pattern Recognition research since at least the 1960's. The ready availability of image samples and the continuing challenge of commercially viable recognition has meant that OCR research is ongoing. However classification of loosely constrained handwritten digits, at least, is essentially a solved problem [1].

A good review of OCR is found in [2]. The huge quantity of research from academia and industry has yielded a multitude of algorithms for normalization [3] [4], feature extraction [5], and classification [6] [7] [8] [9], that are capable of digit OCR. The popularity of OCR research was maintained with the advent of neural network paradigms applicable to feature extraction and classification. The advantage of many neural network classifiers, once trained, is their efficiency, and despite advances in computational resources, future commercial segmentation *and* recognition efforts [10] [11] will be precluded from using numerous techniques from the literature because of their algorithmic computational requirements. The trade-off between classification performance and computational requirements has prompted this study of digit classifier efficacy. The reader should see Kimura et al. [12] for a similar survey.

2 NIST OCR Databases

The classifiers described in this report were trained and tested using feature vectors derived from the digit images of NIST Special Database 3 [13]. This database consists of binary 128 by 128 pixel raster images segmented from

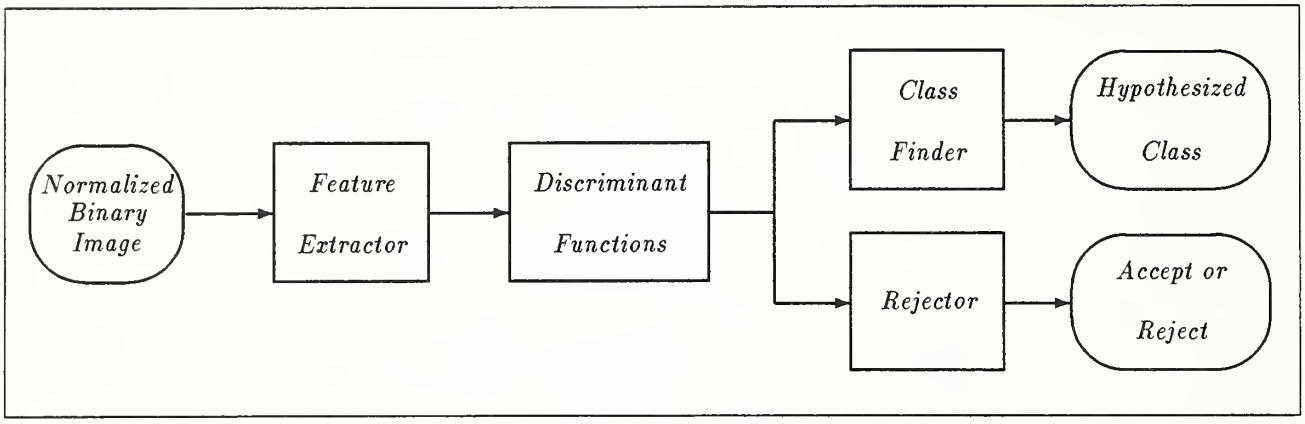


Figure 1: Components of Classification System

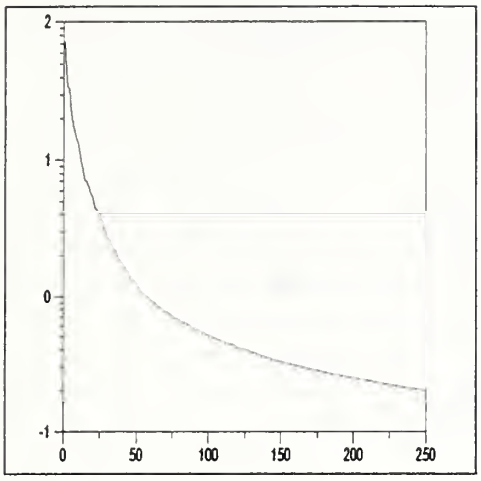
the sample forms of 2100 writers published on CD as [14]. External results on segmentation and recognition of this database have been reported [15]. The relative difficulties of the NIST OCR databases have been discussed in [16]. For this study samples are drawn randomly from the first 250 writers to yield a training set of 7480 digits with *a priori* class probabilities all equal to 0.1. Even for digits, depending on the application, certain classes may be more prevalent; in banking tasks, for example, "0" is more common. The test set is similarly constructed from the second 250 writers yielding 23140 samples. The images are size normalized by pixel deletion, stroke width is bounded by binary erosion and dilation, and consistent orientation is effected by row shearing.

3 Components of Classification System

Each experimental OCR classifier notionally comprises the modules of Figure 1. The ovals indicate inputs and outputs, the rectangles represent processing modules, and the arrows show logical procedure. A 32 pixel square raster contains the normalized digit image. The feature extraction module linearly transforms the binary data yielding reduced dimensionality classifiable features. The next module is the bank of "Discriminant Functions". There are as many discriminant functions as there were clusters in the training set. For several classifiers more than one cluster per class was used. Each function maps an n dimensional extracted feature vector to a scalar which adopts a large value if the unknown input is of the cluster corresponding to that particular discriminant function. The values produced by the bank of discriminant functions are sent, finally, to both the "Class Finder" and the "Rejector". The class finder infers the *hypothesized class* from the discrimant values associated with each cluster, yielding the classifier's best guess of the class. The rejector module computes a "confidence function" of the discriminant values, compares the result with a specified threshold, and thereby indicates whether the unknown should be accepted or rejected. Rejection implies the hypothesized class is not trustworthy and such examples are either ignored or reclassified more robustly.

4 Feature Extractor

The normalized input images are 2^5 pixels high and the width is less than or equal to the height. Raster dimensionality is 2^{10}. A lower dimensionality feature vector is obtained as the incomplete Karhunen-Loève (K-L) transformation of the image, typically of useful dimension $\leq 2^6$. The K-L transformation is an orthonormal function and corresponds to projection of the images onto the eigenvectors of the covariance matrix of the image data. (The production of this transform is also known as *principal factors* or *principal components* analysis.) The covariance matrix is diagonalized (using, for example, EISPACK serial Fortran routines) producing the largest eigenvalues and corresponding eigenvectors. Feature extraction is thus the application of an affine function to



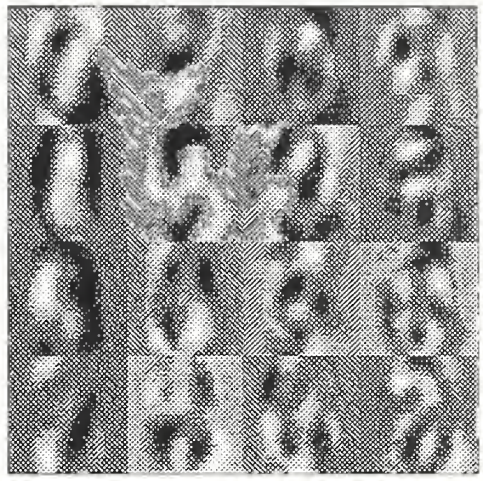


Figure 2: Eigenvalues and Eigenvectors. The latter are shown in eigenvalue order top to bottom then left to right. Note the central support on the x-axis for the basis functions due to the tight constraints imposed by the size normalization. It is visually apparent that several eigenvectors will have large correlations and anti-correlations with several digit classes.

the image; the mean image vector is subtracted and the result is premultiplied by the matrix whose rows are the eigenvectors. Thus each element of the K-L transform is the projection of the image onto a basis vector. The sample variance of that element is the respective eigenvalue, which is therefore a measure of its "size". Only 64 of the eigenvectors were retained with eigenvalues ranging from 72.9 down to 0.9. Figure 2 shows the eigenvalues on a log scale; their rapid decline implies that the information content of the K-L features is concentrated in the first few features. This variance ordering of the features provides a consistent method for reducing the dimensionality of the feature sets, the less variant coefficients being discarded first. Figure 2 also shows the leading eigenvectors.

Figure 3 shows locations of the 748 training examples per class represented using the first two K-L features. Although graphical representation of high dimsensional spaces is the perennial problem for pattern recognition it is apparent that two features are insufficient to separate all classes. Despite "0"s and "1"s, and "6"s and "9"s separating reasonably, no classifier, in two dimensions, acheived better than 50% test set error.

Only one type of feature has been considered for this comparsion of classifiers. Although many other feature types are known to be classifiable for OCR [17] [18], the Karhunen-Loève transform is, among the unitary transforms, an optimally compact signal representation of the original data. These features have the further benefit that reconstruction of the images is possible and the K-L transform is optimal, for n coefficients of a unitary transform, at mean square error between reconstruction and original. The variance ordering is useful for comparing classifiers at reduced dimensionalities.

5 Feature Clustering

For several classifiers described below it is well known [19] that splitting class prototypes into clusters yields improved performance. Indeed clustering algorithms can be used for unsupervised classification [20]. One readily available method is the "K-means" algorithm [21] [22]. The examples are iteratively split into clusters as follows. The first iteration starts with the cluster of all prototypes and computes the distances from each to the cluster center. The second iteration divides the cluster into two, moving cases from one to the other until no further movements decrease the distances between each case and the center of its assigned cluster. For subsequent

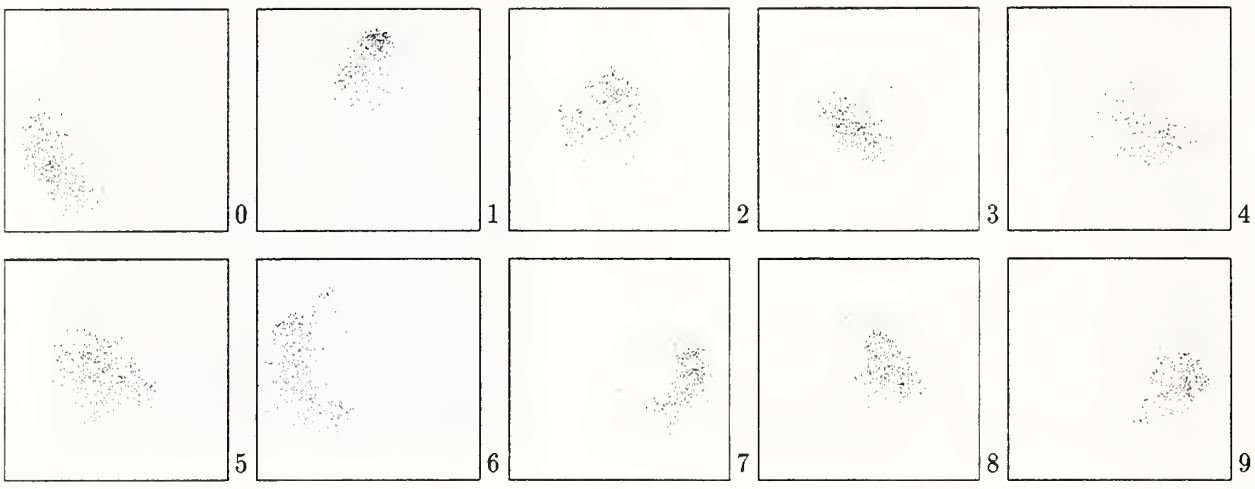


Figure 3: Locations by class of the training samples. Compare these with the EMD classification map of figure 4. The horizontal and vertical axes correspond to the first and second eigenvectors respectively.

iterations, the cluster with the largest variance is split and its prototypes are assigned to the cluster whose mean is least distant. The means are then updated and reassignment continues until no movements are made for an iteration. It is important to note that this distance clustering is classless such that clusters could be formed from neighbors of different class. Supervison is enforced by application of the algorithm to each class independently.

6 Classifiers and Discriminant Functions

Each classifer consists of a bank of discriminant functions. The classifiers are separated into three categories. It is, however, notable that the category names are somewhat arbitrary ant that some classifiers have attributes of more than one category. In the statistical pattern recognition literature [23] the *Parametric* classifiers use variables such as the expected means and covariances to express the class density functions. In assuming, for example, linear and quadratic forms for our discriminant functions we categorize simple Euclidean Minimum Distance (EMD), the more advanced Quadratic Minimum Distance (QMD) and the Normal (NRML) classifier as parametric classifiers. The *Non-Parametric* classifiers do not adopt a structured expression of the density functions; two Nearest Neighbor classifiers, the popular K-NN and an improvement termed Weighted Several Nearest Neighbors (WSNN) are considered. Finally, the Neural Net category contains the Multi-Layer Perceptron (MLP), Radial Basis Functions classifiers of two types (RBF1 and RBF2), and the Probabilistic Neural Net (PNN).

For each type of discriminant function, one or more diagrams are provided showing the resulting hypothetical class regions in two-dimensional feature space. These diagrams show the hypothesized classifications of regularly spaced feature vectors sampled over the square region centered on (0,0) and with extent large enough to contain the training vectors. Restriction of this graphical representation to two dimensions is undeniably, but necessarily, not ideal. These class maps should be compared with the real distributions of figure 3.

6.1 Notation

The notation below will be used in the descriptions of the discriminant functions.

L = number of classes. For digits, $L = 10$

$$
\begin{aligned}
N &= \text{number of clusters, } N \geq L \\
n &= \text{dimensionality of features} \\
\mathbf{R}^n &= \text{the set of all } n\text{-tuples of real numbers} = \text{"feature space"} \\
\mathbf{x} &= \text{extracted "feature vector" of a digit } (\mathbf{x} \in \mathbf{R}^n) \\
M_i &= \text{number of training prints of cluster } i \ (1 \leq i \leq N) \\
\mathbf{x}_j^{(i)} &= \text{feature vector from } j^{\text{th}} \text{ digit of cluster } i \ (1 \leq i \leq N, 1 \leq j \leq M_i) \ (\mathbf{x}_j^{(i)} \in \mathbf{R}^n) \\
\boldsymbol{\mu}_i &= \text{mean feature vector for cluster } i \ (1 \leq i \leq N) \ (\boldsymbol{\mu}_i \in \mathbf{R}^n) \\
\mathbf{m}_i &= \text{an estimate of } \boldsymbol{\mu}_i \\
\boldsymbol{\Sigma}_i &= \text{covariance matrix for cluster } i \ (1 \leq i \leq N)(\boldsymbol{\Sigma}_i \in \mathbf{R}^{n \times n}) \\
\mathbf{S}_i &= \text{an estimate of } \boldsymbol{\Sigma}_i \\
d^2(\mathbf{x}, \mathbf{y}) &= (\mathbf{x} - \mathbf{y})^{\mathrm{T}}(\mathbf{x} - \mathbf{y}) = \text{squared Euclidean distance between } \mathbf{x} \text{ and } \mathbf{y} \ (\mathbf{x}, \mathbf{y} \in \mathbf{R}^n) \\
r^2(\mathbf{x}, \mathbf{y}, \mathbf{z}) &= \sum_{i=1}^{n} ((x_i - y_i)/z_i)^2 \\
&= \text{distance between } \mathbf{x} \text{ and } \mathbf{y} \text{ normalized by } \mathbf{z} \ (\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{R}^n) \\
D_i(\mathbf{x}) &= i^{\text{th}} \text{ discriminant function } (1 \leq i \leq N, \mathbf{x} \in \mathbf{R}^n)
\end{aligned}
$$

6.2 Parametric Classifiers

6.2.1 Euclidean Minimum Distance (EMD) Classifier

This is perhaps one of the simplest classifiers that one can design. Its discriminant functions are of the form

$$D_i(\mathbf{x}) = -d^2(\mathbf{x}, \mathbf{m}_i).$$

An unknown is assigned the class associated with the cluster of the highest-valued discriminant function. This is equivalent to using the class label of the estimated cluster-mean that is closest, in the Euclidean distance sense, to the unknown. In the one cluster per class case the hypothetical class regions are convex polygons. This classifier is essentially the Perceptron (although the actual boundaries may be different) whose linear separability limitations were described by Minsky and Papert [24]. Figure 4 shows the class regions when only two features are used. The estimated cluster mean vectors $\mathbf{m}_i$ are marked with plus signs.

6.2.2 Quadratic Minimum Distance (QMD) Classifier

The training examples of each cluster i are used to produce sample covariance matrices, $\mathbf{S}_i$, and estimated mean vectors $\mathbf{m}_i$. The following discriminants are used:

$$
\begin{aligned}
D_i(\mathbf{x}) &= -(\mathbf{x} - \mathbf{m}_i)^{\mathrm{T}} \mathbf{S}_i^{-1} (\mathbf{x} - \mathbf{m}_i) \\
&= -\mathbf{z}^{\mathrm{T}} \mathbf{z} \\
\mathbf{z} &= \boldsymbol{\Lambda}_i^{-\frac{1}{2}} \boldsymbol{\Psi}_i^{\mathrm{T}} (\mathbf{x} - \mathbf{m}_i)
\end{aligned}
$$

That is, the cluster mean, $\mathbf{m}_i$, is first subtracted from the unknown, and the result projected onto the eigenvectors Ψ_i of the cluster i covariance matrix, and finally whitened by dividing each component by the root of the corresponding eigenvalues Λ_i This can be thought of as a form intermediate between EMD and the Normal (NRML) classifier, described below. Figure 4 shows the resulting class regions; the boundary between any two clusters is quadratic. When the number of clusters per class increases the inverse covariance matrices for a given cluster are formed from a decreasing number of training examples. Computational difficulties occur when the number of cluster examples forming the covariance is small. The rank of $\mathbf{S}_i$ may then be less than n and $\mathbf{S}_i$ is singular preventing its conventional inverse from being evaluated. It should be noted that QMD is not a true

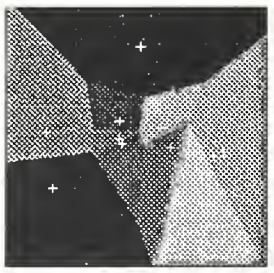
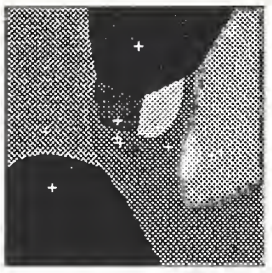
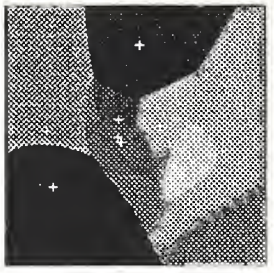

EMD

QMD

NRML

Figure 4: Parametric Classifiers. For EMD note the class boundaries are the perpendicular bisectors of segments connecting pairs of class means. For QMD and NRML note the quadratic forms of the decision boundaries. The + signs indicate the locations of the estimated class means.

parametric classifier in the sense that the estimated conditional density functions do not integrate to unity. If this condition is forced then the normal classifier results.

6.2.3 Normal (NRML) Classifier

This classifier is based on parametric density estimation that presupposes a multivariate normal distribution for each class. First, it will be useful to mention a few facts that pertain to any parametric classifier, using the following terminology:

$$\begin{aligned}
p(i) &= \textit{a priori} \text{ probability of cluster } i \\
\lambda(i|j) &= \text{loss incurred by classifying to } i \text{ a print that is of cluster } j \ (1 \leq i, j \leq N) \\
p(\mathbf{x}) &= \text{mixture density: for } S \subseteq \mathbf{R}^n,\ \textstyle\int_S p(\mathbf{x})d\mathbf{x} = P(\mathbf{x} \in S) \\
p(\mathbf{x}|i) &= \text{conditional density: for } S \subseteq \mathbf{R}^n,\ \textstyle\int_S p(\mathbf{x}|i)d\mathbf{x} = P(\mathbf{x} \in S | \mathbf{x} \text{ is from a cluster-}i \text{ print}) \\
p(i|\mathbf{x}) &= \textit{a posteriori} \text{ probability: for a particular } \mathbf{x},\ p(i|\mathbf{x}) = P(\mathbf{x} \text{ is from a cluster-}i \text{ print})
\end{aligned}$$

Given a particular loss function $\lambda(i|j)$, the optimal or "Bayesian" classifier is the one that minimizes the expected loss. Define the "symmetric" loss function in terms of the Krönecker delta:

$$\lambda(i|j) = 1 - \delta_{ij} = \begin{cases} 0 & i = j \\ 1 & \text{otherwise} \end{cases}.$$

This means that correct classifications produce no losses and that all kinds of incorrect classifications produce equal loss values of 1 unit. In this case, the Bayesian classifier is the one that classifies each unknown $\mathbf{x}$ to the cluster i for which the *a posteriori* probability $p(i|\mathbf{x})$ is highest. According to Bayes's rule [25],

$$p(i|\mathbf{x}) = \frac{p(i)p(\mathbf{x}|i)}{p(\mathbf{x})}.$$

Since the value of the mixture density $p(\mathbf{x})$ has no effect on which possible i value maximizes $p(i|\mathbf{x})$, $p(\mathbf{x})$ may be disregarded. Also for a pattern recognition problem in which the *a priori* probabilities are the same then the $p(i)$ can be ignored. The result is to classify $\mathbf{x}$ to the cluster i for which $p(\mathbf{x}|i)$ is highest. For the Normal classifier each cluster, i, is assumed to have conditional density function

$$p(\mathbf{x}|i) = (2\pi)^{-\frac{n}{2}} |\mathbf{\Sigma}_i|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^{\mathrm{T}} \mathbf{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)\right),$$

where $\boldsymbol{\mu}_i$ and $\mathbf{\Sigma}_i$ are the mean vector and covariance matrix for cluster i. For classification the $(2\pi)^{-\frac{n}{2}}$ term is constant and may be discarded. Finally by replacing the mean vectors $\boldsymbol{\mu}_i$ and covariance matrices $\mathbf{\Sigma}_i$ with their sample estimates, $\mathbf{m}_i$ and $\mathbf{S}_i$, squaring, and taking logarithms the discriminant function for the Normal classifier becomes

$$D_i(\mathbf{x}) = -\log|\mathbf{S}_i| - (\mathbf{x} - \mathbf{m}_i)^{\mathrm{T}} \mathbf{S}_i^{-1} (\mathbf{x} - \mathbf{m}_i).$$

Figure 5: Single Nearest Neighbor Classifier. Note the very intricate non-contiguous decision boundaries local to each training prototype.

The hypothetical class regions are given in figure 4. The location of the means in that figure, indicated by the + signs, shows that if $\mathbf{x} = \mathbf{m}_i$ then, in two dimensions, misclassification can result solely because of the determinant terms.

6.3 Nearest Neighbor Classifiers

Nearest-neighbor classifiers have been the subject of decades of research (see, for example, Dasarathy's collection of papers [36]). The following are simple and ubiquitous yet effective examples of such methods.

6.3.1 k-Nearest Neighbor (k-NN)

If $k = 1$, this is an elaboration of EMD; instead of using just $\mathbf{m}_i$, as a single prototype for the class, the 1-NN classifier uses *all* of the class-i training examples as prototypes for the class. The 1-NN classification of an unknown vector is simply the class of the nearest prototype. This rule is intuitively appealing, and Cover and Hart [6] have shown it to have good asymptotic behavior: under mild assumptions, its large-sample probability of error is bounded above by twice the Bayes (i.e. minimum possible) probability of error. The 1-NN discriminant functions have the form:

$$D_i(\mathbf{x}) = -\min_{1 \le j \le M_i} d^2\left(\mathbf{x}, \mathbf{x}_j^{(i)}\right).$$

Figure 5 shows the class regions. Each region is the union of many convex polygons each containing a single prototype of the class; hence, a class region is a very complicated polygon, not necessarily convex or even connected. In the more general case voting between the k nearest neighbors is used. The majority class is used as the hypothesis. The method is useful near class boundaries when the single nearest neighbor may be of the wrong class but the majority are not. If $\mathcal{S}_\mathbf{x}$ is the set of the k closest prototypes voting on the class of $\mathbf{x}$ then it is the union of the sets of voting prototypes, $\mathcal{S}_\mathbf{x}^{(i)}$, containing only prototypes of class i. The k-NN discriminant function is then simply the set size:

$$D_i(\mathbf{x}) = |\mathcal{S}_\mathbf{x}^{(i)}|.$$

6.3.2 Weighted Several Nearest Neighbors (WSNN)

A more elaborate form of the nearest neighbor method is to allow k to be a random variable such that the number of voting neighbors is different for each unknown. This classifier finds the closest prototype to the unknown, then defines the "neighboring" prototypes to be those whose squared Euclidean distance from the unknown is less than α times the squared-distance of the nearest prototype, where α is a constant. Further the number of "votes" received by class i is divided by the square root of the sum of squared-distances of class-i near neighbors from the

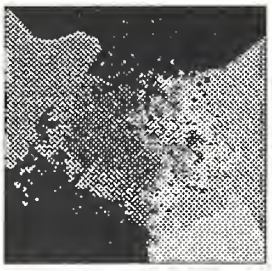
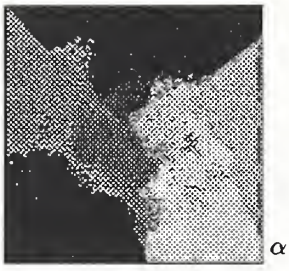
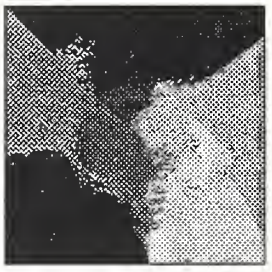

$\alpha = 50$

$\alpha = 500$

$\alpha = 1000$

Figure 6: Weighted Several Nearest Neighbors. In the limit of small α this classifier defaults to 1-NN. Note the fine grained structure throughout that is typical of nearest neighbor methods.

unknown, so as to diminish the importance of neighbors that are relatively far away compared to other neighbors. Formally,

$$\begin{aligned}
\alpha &= \text{neighborhood-size factor} \\
\mathcal{S}_{\mathbf{x}}^{(i)} &= \text{the set of indices of class-}i\text{ training vectors that are} \\
&\quad \text{in the } \alpha\text{-neighborhood of unknown vector } \mathbf{x} \\
&= \left\{ j \,\middle|\, 1 \le j \le M_i, d^2\left(\mathbf{x}, \mathbf{x}_j^{(i)}\right) < \alpha \min_{1 \le k \le N, 1 \le p \le M_k} d^2\left(\mathbf{x}, \mathbf{x}_p^{(k)}\right) \right\} \\
V_{\mathbf{x}}^{(i)} &= |\mathcal{S}_{\mathbf{x}}^{(i)}| = \text{number of "votes" for class } i
\end{aligned}$$

The discriminant functions are then

$$D_i(\mathbf{x}) = \begin{cases} V_{\mathbf{x}}^{(i)} \left(\sum_{j \in \mathcal{S}_{\mathbf{x}}^{(i)}} d^2\left(\mathbf{x}, \mathbf{x}_j^{(i)}\right) \right)^{-\frac{1}{2}} & \text{if } V_{\mathbf{x}}^{(i)} > 0 \\ 0 & \text{otherwise} \end{cases} .$$

Figure 6 shows the WSNN class regions resulting from α values of 50, 500 and 1000.

6.4 Neural Net Classifiers

6.4.1 Multi-Layer Perceptron (MLP)

This classifier is also known as a feedforward neural net. We have used an MLP with three layers (counting the inputs as a layer). It will be convenient to define the following notation:

$$\begin{aligned}
N^{(i)} &= \text{number of nodes in } i^{\text{th}} \text{ layer } (i = 0, 1, 2),\ N^{(0)} = n,\ N^{(2)} = L \\
f(x) &= 1/(1 + e^{-x}) = \text{sigmoid function} \\
b_i^{(k)} &= \text{bias weight of } i^{\text{th}} \text{ node of } k^{\text{th}} \text{ layer } (k = 1, 2; 1 \le i \le N^{(k)}) \\
w_{ij}^{(k)} &= \text{weight connecting } i^{\text{th}} \text{ node of } k^{\text{th}} \text{ layer to } j^{\text{th}} \text{ node of} \\
&\quad (k-1)^{\text{th}} \text{ layer } (k = 1, 2; 1 \le i \le N^{(k)}; 1 \le j \le N^{(k-1)})
\end{aligned}$$

The discriminant functions are then of the form

$$D_i(\mathbf{x}) = f\left(b_i^{(2)} + \sum_{j=1}^{N^{(1)}} w_{ij}^{(2)} f\left(b_j^{(1)} + \sum_{k=1}^{N^{(0)}} w_{jk}^{(1)} x_k \right) \right) .$$

For the training of the weights of this network, a reasonable procedure is the use of an optimization algorithm to minimize the mean-squared-error, over the training set, between the discriminant values actually produced and

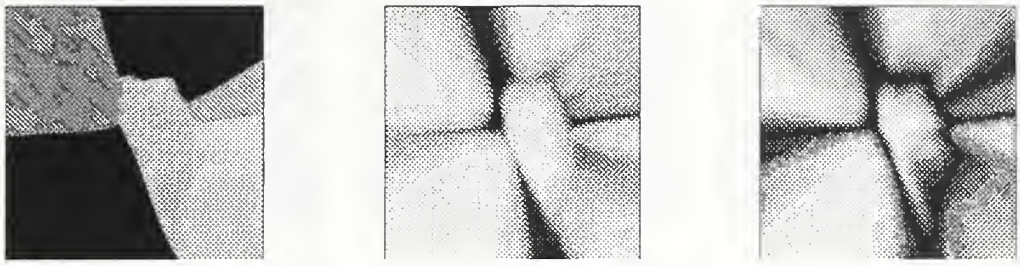


Figure 7: MLP Classification and Confidence Maps. From left; class boundaries, highest discriminant value, difference in highest two discriminant values.

"target discriminant values" consisting of the appropriate strings of 1's and 0's as defined by the actual classes of the training examples. For example, if a training feature vector is of class 2, then its target vector of discriminant values is set to (0,1,0,0,0,0,0,0,0,0). It is more feasible to minimize this kind of an "error function" than to attempt to directly minimize the number of incorrectly classified training examples, since the latter number will take on only relatively few values and is a discontinuous "step function". The error function is modified by addition of a scalar "regularization" term [26]. This equals a tunable constant, λ, multiplied by the mean square weight, $\overline{w_{ij}^2}$. This term prevents large weights which are associated with overtraining, i.e. the overfitting of the weights to the training data. This has been shown to increase the generalization ability of the network [27].

Networks of the MLP type are the most commonly used "neural nets" in use today, and they are usually trained using a "backpropagation" algorithm [28]. A "scaled conjugate gradient" training method instead [29, 30, 31, 27] has been preferred to the ubiquitous backpropagation method, speed gains of an order of magnitude being typical. Figure 7 shows MLP class regions resulting from varying the first two inputs to a trained 8 input, 48 hidden unit network.

6.4.2 Radial Basis Functions (RBF1 and RBF2)

Neural nets of the Radial Basis Functions type get their name from the fact that they are built from radially symmetric Gaussian functions of the inputs. Actually, the RBF nets discussed here use Gaussian functions that are more general than radially symmetric functions: their constant potential surfaces are ellipsoids whose axes are parallel to the coordinate axes, whereas radially symmetric Gaussian functions have spherical constant potential surfaces. However, the name Radial Basis Functions has become customary for any neural net that uses Gaussian functions in its first layer.

We have experimented with RBF networks of two types, which will be denoted RBF1 and RBF2. The following notation will be convenient:

$$
\begin{aligned}
N^{(i)} &= \text{number of nodes in } i^{\text{th}} \text{ layer } (i = 0, 1, 2) \\
\mathbf{c}^{(j)} &= \text{center vector of } j^{\text{th}} \text{ hidden node } (1 \le j \le N^{(1)})\ (\mathbf{c}^{(j)} \in \mathbf{R}^n) = (c_1^{(j)}, \ldots, c_n^{(j)})^{\mathrm{T}} \\
\boldsymbol{\sigma}^{(j)} &= \text{width vector of } j^{\text{th}} \text{ hidden node } (1 \le j \le N^{(1)})\ (\boldsymbol{\sigma}^{(j)} \in \mathbf{R}^n) = (\sigma_1^{(j)}, \ldots, \sigma_n^{(j)})^{\mathrm{T}} \\
b_j^{(k)} &= \text{bias weight to the } j^{\text{th}} \text{ node of the } k^{\text{th}} \text{ layer} \\
f(x) &= 1/(1 + e^{-x}) = \text{sigmoid function} \\
w_{ij} &= \text{weight connecting } i^{\text{th}} \text{ output node to } j^{\text{th}} \text{ hidden node } (1 \le i \le N^{(2)}; 1 \le j \le N^{(1)})
\end{aligned}
$$

Each hidden node computes a radial basis function. For RBF1, these functions are unbiased exponentials

$$\phi_j(\mathbf{x}) = \exp\left(-r^2(\mathbf{x}, \mathbf{c}^{(j)}, \boldsymbol{\sigma}^{(j)})\right),$$

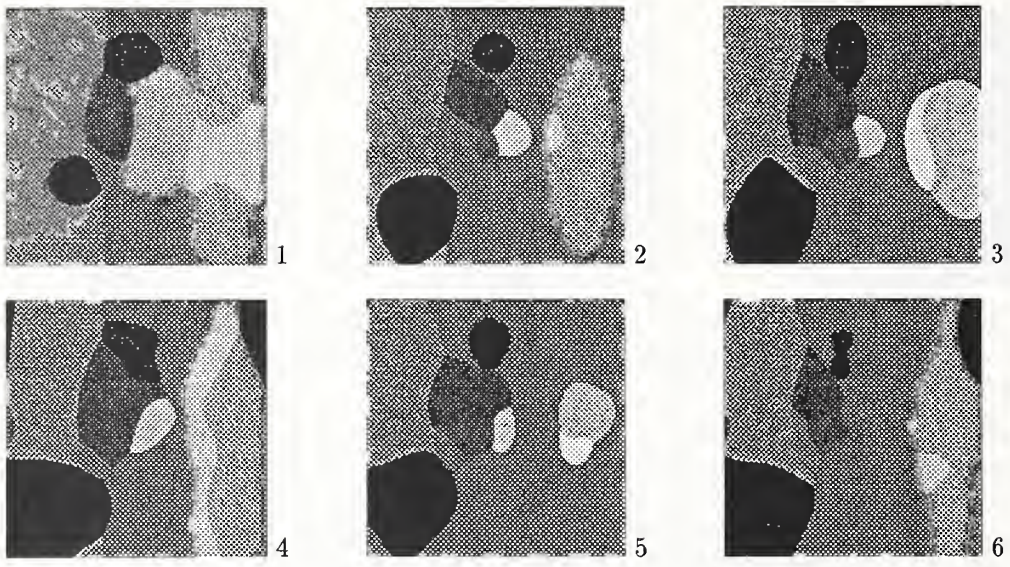


Figure 8: RBF1 Classification regions for increasing numbers of centers per class.

and for RBF2, they are of the sigmoidal form with bias

$$\phi_j(\mathbf{x}) = f\left(-b_j^{(1)} - r^2(\mathbf{x}, \mathbf{c}^{(j)}, \boldsymbol{\sigma}^{(j)})\right).$$

For either type of RBF, the i^{th} discriminant function is the following function of the radial basis functions:

$$D_i(\mathbf{x}) = f\left(b_i^{(2)} + \sum_{j=1}^{N^{(1)}} w_{ij}\phi_j(\mathbf{x})\right).$$

The centers $\mathbf{c}^{(j)}$, widths $\boldsymbol{\sigma}^{(j)}$, hidden-node bias weights $b_j^{(1)}$ (RBF2 only), output-node bias weights $b_i^{(2)}$, and output-node weights w_{ij} may be collectively thought of as the trainable "weights" of the RBF network. They are trained initially using the cluster means (from a "K-means" algorithm applied to the prototype set) as the center vectors $\mathbf{c}^{(j)}$. The width vectors $\boldsymbol{\sigma}^{(j)}$, are set to a single tunable positive value. More sophisticated methods of determining RBF parameters may be found in [32] [33]. The output layer weights are set such that each output node is connected with a positive weight to hidden nodes of its class (that is, hidden nodes whose initial center vectors are means of clusters from its class), and connected with a negative weight to hidden nodes of other classes. Training proceeds by optimization identical to that described for the MLP. Figure 8 shows RBF1 class regions resulting from the use of up to 6 hidden nodes per class, and Figure 9 shows RBF2 class regions for the same numbers of hidden nodes per class.

6.4.3 Probabilistic Neural Net (PNN)

This classifier is proposed in a 1990 paper by Specht [34]. Each training example becomes the center of a kernel function which takes its maximum at the example and recedes gradually as one moves away from the example in feature space. An unknown $\mathbf{x}$ is classified by computing, for each class i, the sum of the values of the class-i kernels at $\mathbf{x}$. Many forms are possible for the kernel functions; we have obtained our best results using radially symmetric Gaussian kernels. The resulting discriminant functions are of the form

$$D_i(\mathbf{x}) = \sum_{j=1}^{M_i} \exp\left(-\frac{1}{2\sigma^2} d^2\left(\mathbf{x}, \mathbf{x}_j^{(i)}\right)\right),$$

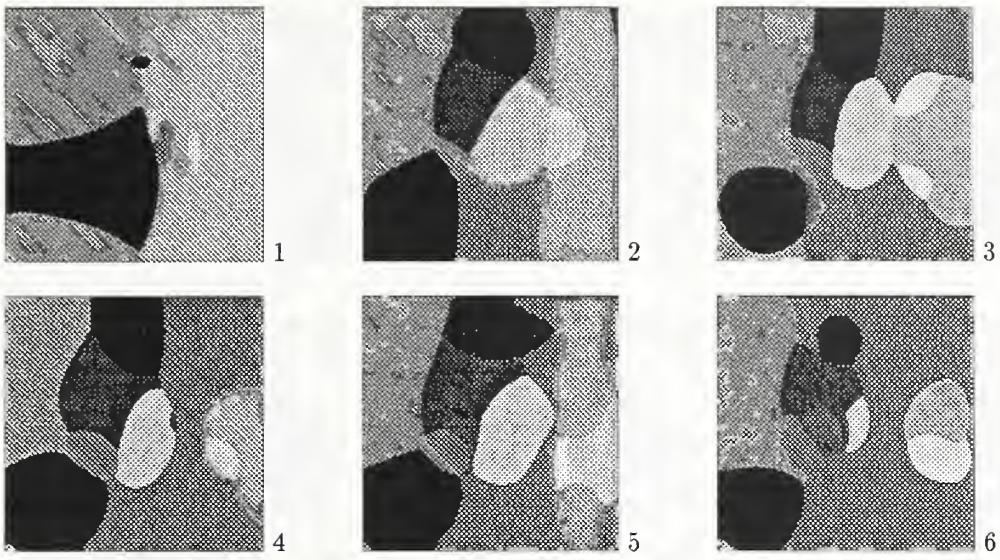


Figure 9: RBF2 Classification regions for increasing numbers of centers per class.

where σ is a scalar "smoothing parameter" that may be optimized by trial and error. Figure 10 shows the PNN class regions resulting from the use of σ values of 0.25, 1.00, and 5.00. Notice that a small σ value produces very complex class regions similar to those of 1-NN, and that as σ is increased, the regions become simpler.

7 Class Finder and Rejector

The "Class Finder" module is a function mapping the discriminant values to a single index indicating hypothesized class. In the most elementary case the index of the maximum indicates the class. The function may be more complicated; discriminant values associated with the same class may be combined in some weighted voting.

The Rejector module produces a confidence value; it is a function of the discriminant values

$$a((D_1(\mathbf{x}), \ldots, D_N(\mathbf{x})) : R^N \rightarrow R^1,$$

and it quantifies the assertiveness of the classifier for the unknown $\mathbf{x}$. The simplest use of a is to subtract from it some predefined threshold value, a_0. A negative result implies rejection of the hypothesized class of $\mathbf{x}$. This mechanism allows examplars to remain unclassified with the intent of achieving a lower substitutional error rate on those testing vectors whose classifications are accepted. This study only considers the simple strategy of using the maximum discriminant value as the confidence. An error rate versus rejection rate plot is obtained by selecting one confidence rule and varying the threshold a_0. High thresholds cause rejection of more examples but lower error rates on those accepted.

8 Comparison of the Classifiers

8.1 Accuracy

For each class i, the number of the 2314 class-i test digits that were correctly classified is denoted by c_i. Clearly, $c_i/2314$ may be used as an estimate of the conditional probability of correct classification of a print given that the actual class of the print is i. The mixture probability of correct classification is then the *a priori* weighted sum

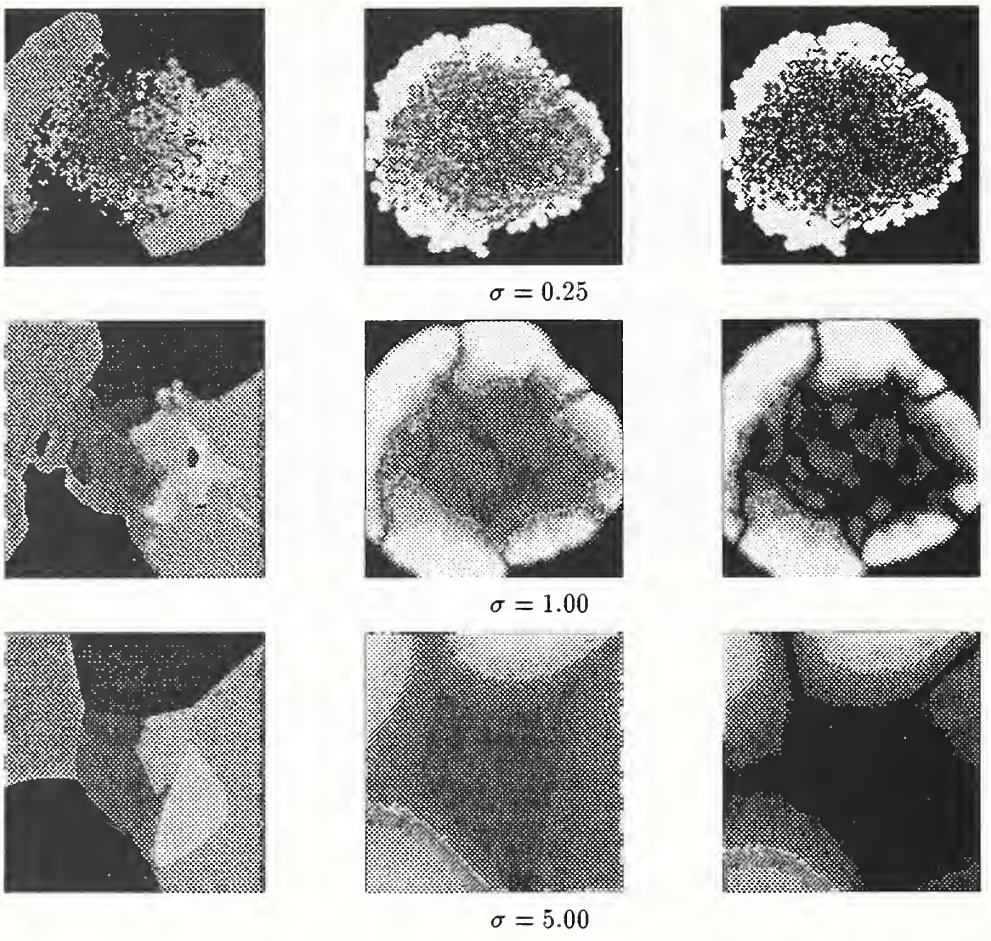


Figure 10: PNN Classification and Confidence Maps in 2 dimensions for increasing σ values. From left: class boundaries, highest discriminant value, difference in highest two discriminant values.

of the by-class probabilities. Table 8.1 shows for each classifier these estimated probabilities of *error*, expressed as percentages, for increasing dimensionality of the K-L feature set.

Note that the optimal number of features (shown in bold) is not the same for all classifiers, the parametric classifiers, QMD and NRML, being noticeably more parsimonious in the number of features required. It is also apparent that most of the classifiers attain a plateau as the number of features reaches approximately 32 thereafter only gaining several tenths of a percent. The best classifiers are the computationally expensive nearest neighbor classifiers and their relative PNN. They achieve one third less errors than the neural networks and parametric classifiers. The optimum value of $\alpha = 1.1$ for WSNN corresponds to a 1-NN scheme for most test patterns. Accordingly k-NN is seen to have a higher error rate for increasing k.

Figure 11 gives the error versus rejection profiles for the classifiers.

8.2 Computational Requirements

High training costs for a classifier can hinder experimentation, of course, and sufficiently large expense in training can ultimately preclude the use of such a classifier. Of the algorithms described here, for a fixed size training set, no classifier took more than three hours of workstation time. The neural networks are notoriously expensive to train whereas the nearest neighbor methods, including PNN, require no training. Once off-line training is

System	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
knn:1	27.0	7.9	4.6	3.5	3.1	2.9	2.7	2.7	2.7	**2.6**	2.6	2.6	2.7	2.7	2.7	2.7
knn:3	23.7	7.1	4.2	3.4	3.1	2.8	2.7	2.7	2.7	**2.6**	2.7	2.7	2.7	2.7	2.8	2.7
knn:5	22.1	6.8	4.1	3.3	3.1	2.9	2.8	2.8	**2.7**	2.8	2.8	2.8	2.8	2.8	2.8	2.8
wsnn:1.1	26.8	7.8	4.5	3.4	3.0	2.8	2.7	2.6	2.6	**2.5**	2.6	2.6	2.6	2.6	2.5	2.6
pnn:3.0	21.9	7.2	4.3	3.3	2.9	2.7	2.7	2.6	2.6	**2.5**	2.6	2.6	2.6	2.6	2.5	2.5
mlp:32	23.6	9.7	6.9	6.4	6.2	5.8	5.6	5.7	5.5	5.6	5.5	**5.3**	5.4	5.4	5.3	5.4
mlp:48	22.8	9.0	6.5	5.9	5.4	5.2	5.2	5.0	4.7	4.9	5.0	4.7	**4.6**	4.9	5.0	4.9
mlp:64	22.2	8.7	6.2	5.3	4.9	4.6	4.5	4.6	4.5	4.5	4.5	4.5	**4.3**	4.5	4.4	4.5
rbf1:1	29.8	14.3	13.2	13.0	13.4	13.2	13.1	13.9	13.0	**12.6**	13.4	12.6	13.2	13.3	13.2	13.2
rbf1:2	24.4	11.5	9.8	9.3	8.9	8.5	8.5	8.4	8.2	8.4	8.2	8.1	8.3	8.1	**7.9**	7.9
rbf1:3	22.7	10.0	8.0	61.4	7.1	6.7	6.6	6.5	6.5	6.5	6.4	6.4	**6.2**	6.4	6.2	6.3
rbf1:4	22.2	9.3	6.9	6.1	5.8	5.7	5.5	5.5	5.5	5.4	5.5	5.4	5.4	**5.3**	5.3	5.4
rbf1:5	21.4	8.9	6.4	5.5	5.0	5.0	4.7	4.9	5.0	4.9	4.8	4.7	4.9	4.9	4.7	**4.6**
rbf1:6	21.0	8.5	5.9	5.1	4.9	4.6	4.4	4.3	4.5	4.3	4.3	**4.2**	4.2	4.4	4.3	4.4
rbf2:1	28.5	71.3	11.7	9.9	9.7	8.7	9.5	9.1	9.1	9.2	**8.6**	8.8	8.8	8.9	8.9	8.9
rbf2:2	24.3	11.3	8.9	7.9	7.3	6.7	6.4	**6.1**	6.1	6.3	6.3	6.2	6.3	6.2	6.2	6.5
rbf2:3	23.0	9.9	7.2	7.0	6.1	5.6	5.5	5.0	6.0	5.4	**4.9**	5.7	4.9	5.0	5.6	5.0
rbf2:4	22.4	9.6	6.4	5.3	5.4	4.4	5.6	5.0	**4.3**	4.5	4.6	4.6	4.5	4.4	4.8	4.7
rbf2:5	21.7	8.2	6.0	5.1	5.3	4.5	4.6	4.4	4.6	4.4	4.4	4.4	4.2	4.1	4.1	**4.0**
rbf2:6	21.4	8.6	5.6	4.7	4.7	4.3	4.5	4.0	4.0	4.2	**3.9**	4.2	4.0	3.9	4.0	4.0
emd:1	37.3	19.1	17.3	16.1	15.6	15.2	15.1	15.0	15.0	14.9	14.9	**14.8**	14.8	14.8	14.8	14.8
emd:2	29.6	14.4	13.1	11.7	11.2	11.0	10.8	10.7	10.7	10.7	10.7	10.7	**10.6**	10.6	10.6	10.6
emd:3	26.8	12.7	10.8	9.3	9.0	8.8	8.8	8.7	**8.6**	8.6	8.7	8.7	8.7	8.7	8.7	8.7
emd:4	25.4	11.9	9.5	8.1	7.6	7.3	7.3	7.4	7.3	**7.1**	7.2	7.1	7.1	7.1	7.1	7.1
emd:5	25.5	11.1	8.9	7.5	6.7	6.7	6.6	6.6	6.5	6.3	6.7	6.6	**6.2**	6.2	6.2	6.3
emd:6	26.4	10.7	8.2	7.3	6.1	6.1	5.9	6.1	6.0	**5.7**	6.0	5.8	5.9	6.0	5.9	6.1
emd:7	26.3	10.3	7.6	6.1	5.9	5.6	5.3	5.5	5.3	5.2	5.4	**5.1**	5.2	5.4	5.4	5.6
qmd:1	26.2	10.0	6.3	5.1	5.0	**4.8**	4.9	5.1	5.1	5.2	5.3	5.6	5.6	5.8	5.8	5.9
qmd:2	23.6	9.2	5.8	4.9	**4.7**	4.7	4.9	4.9	5.0	5.2	5.3	5.5	5.6	5.7	5.8	5.9
qmd:3	25.8	9.1	5.4	4.5	4.1	**4.0**	4.5	4.7	4.9	5.1	5.3	5.4	5.6	5.9	6.0	6.3
qmd:4	25.5	8.9	5.7	5.0	5.0	**4.5**	4.9	5.0	5.3	5.5	6.1	6.3	6.5	6.9	7.2	7.6
nrml	26.1	9.9	6.3	5.1	5.0	**4.8**	4.9	5.0	5.0	5.2	5.3	5.5	5.6	5.5	5.5	5.6

Table 1: Dependence of Classification Error on KL Transform Dimensionality. Given with the classifier acronym are: For k-NN the value of k, for WSNN the value of α, for PNN the value of σ, for MLP networks the number of hiddens units, for RBF networks the number of centers per class, and for EMD and QMD classifiers the number of clusters per class. Bold type indicates the dimensionality yielding minimum error for each classifier.

complete the classification rate is of more interest. The dominant term in recogniton rate is not classification time but the cost of dimensionality reducing feature extraction. Pure classification rates, excluding the K-L transform time, range from 13 characters per second (cps) for the neighbor classifiers (KNN, WSNN and PNN) through 80 cps for two-cluster QMD to 130 cps for RBF and 250 cps for the MLP all on a serial workstation. Nevertheless timing is particularly difficult and the reader is encouraged to first consider algorithmic complexity.

9 Future Work

The focus in academia and commerce on OCR research is now migrating toward the more difficult problem of recognition of structured documents. At its simplest this involves segmentation and recognition of text fields. The processes may be tightly coupled as in the acse of recognition of multiple objects or cluttered field OCR. The Image Recognition Group recognizes that digit OCR is essentially a solved problem for many applications. However work will continue into upper and lower case recognition as it applies to text field processing. There is an emphasis on investigation of algorithms that conserve computational resources as required by commercial products. In

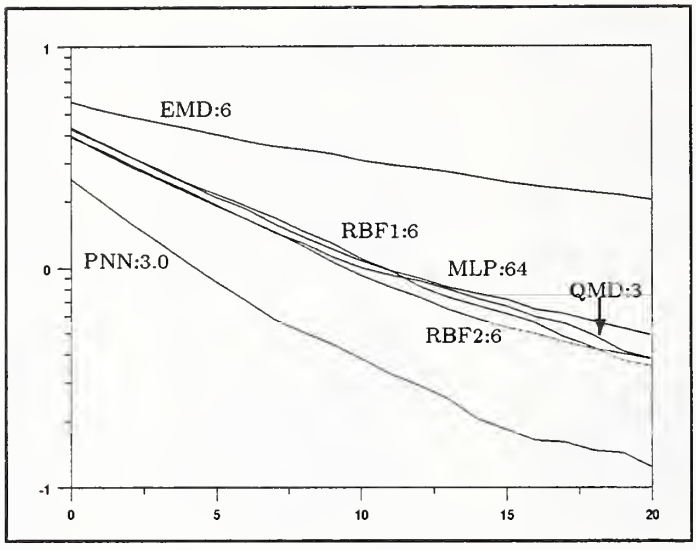


Figure 11: Error versus Rejection, log $e(r)$. The log of the classification error percent of accepted patterns as a function of the low confidence classification rejection percentage. The initial gradients of the curves $e'(r)$ range from -0.62 to -0.47.

particular, the relationship between dimensionality, prototype set size, feature type and computational expense is a candidate for investigation. For example the preservation of performance while reducing neighbor prototype set size is of obvious interest.

10 Summary and Conclusions

We have performed numerous experiments for digit OCR using "statistical" and "neural net" classification of K-L features. The result is that the EMD and QMD parametric classifiers are able to compete with the popular MLP and RBF neural architectures presented here. The lowest error rate classifier, PNN, as described here is more akin to the KNN and WSNN non-parametric neighbor methods in terms of error rate and computational expense than to the other neural network schemes. The authors maybe contacted over email using jerry@magi.ncsl.nist.gov and patrick@magi.ncsl.nist.gov.

References

[1] R. A. Wilkinson, J. Geist, S. Janet, P. J. Grother, C. J. C. Burges, R. Creecy, B. Hammond, J. J. Hull, N. J. Larsen, T. P. Vogl, and C. L. Wilson. The First Optical Character Recognition Systems Confernce. Technical Report NISTIR 4912, National Institute of Standards and Technology, August 1992.

[2] T. Pavlidis and S. Mori, editors. *Special Issue on Optical Character Recognition.* IEEE, July 1992. Volume 80, Number 7.

[3] F. L. Alt. Digital Pattern Recognition by Moments. In G. L. Fischer *et al.*, editor, *Optical Character Recognition*, pages 159–179. McGreger & Werner, 1962.

[4] R. G. Casey. Moment normalization of handprinted characters. *IBM J. Res. Dev*, page 548, 1970.

[5] G. L. Cash and M. Hatamian. Optical character recognition by the method of moments. *CVGIP*, 39:291–310, 1987.

[6] T. M. Cover and P. E. Hart. Nearest neighbour pattern classification. *IEEE Transactions on Information Processing*, IT-13:21–27, 1967.

[7] 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 D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 598–605. Morgan Kaufman, 1990.

[8] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon. Neural network recognizer for hand-written zip code digits. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1, pages 323–331. Morgan Kaufmann, 1989.

[9] K. Fukushima, T. Imagawa, and E Ashida. Character recognition with selective attention. In *Proceedings of the IJCNN*, volume I, pages 593–598, July 1991.

[10] G. Martin and J. Pittman. Recognizing handprinted letters and digits using backpropagation. *Neural Computation*, 3:258–267, 1991.

[11] G. L. Martin. Centered-object integrated segmentation and recognition for visual character recognition. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 504–511. Morgan Kaufmann, Denver, December 1991.

[12] F. Kimura and M. Shridhar et al. Statistical and neural classification of handwritten numerals: A comparable study. In *11th IAPR International Conference on Pattern Recognition*, Aug. - Sept. 1992. to be presented.

[13] M. D. Garris and R. A. Wilkinson. Handwritten segmented characters database. Technical Report Special Database 3, **HWSC**, National Institute of Standards and Technology, February 1992.

[14] C. L. Wilson and M. D. Garris. Handprinted character database. Technical Report Special Database 1, **HWDB**, National Institute of Standards and Technology, April 1990.

[15] R. G. Casey and H. Takahashi. Experience in segmenting and recognizing the nist database. In *Proceedings of the International Workshop on Frontiers of Handwriting Recognition*, France, 1991.

[16] Patrick J. Grother. Cross Validation Comparison of NIST OCR Databases. In D. P. D'Amato, editor, volume 1906. SPIE, San Jose, 1993.

[17] M. T. Y. Lai and C. Y. Suen. Automatic recognition of characters by fourier descriptors and boundary encoding. *Pattern Recognition*, 14(1–6):383–393, 1981.

[18] S. O. Belkasim, M. Shridhar, and M. Ahmadi. Pattern recognition with moment invariants: A comparative study and new results. *Pattern Recognition*, 24(12):1117–1138, 1991.

[19] R. O. Duda and P. E. Hart. *Pattern Recognition and Scene Analysis.* New York: John Wiley & Sons, Inc., 1973.

[20] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, second edition, 1988.

[21] J. A. Hartigan. *Clustering Algorithms*, pages 84–108. New York: John Wiley & Sons, Inc., 1975.

[22] D. J. Hall and G. B. Ball. Isodata: A novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, Ca., 1965.

[23] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* New York: Academic Press, second edition, 1990.

[24] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, MA, 1969.

[25] Nils J. Nilsson. *Learning Machines: Foundations of trainable pattern-classifying systems*, chapter 3, page 45. McGraw-Hill Book Company, 1965.

[26] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[27] J. L. Blue and P. J. Grother. Training Feed Forward Networks Using Conjugate Gradients. In *Conference on Character Recognition and Digitizer Technologies*, volume 1661, pages 179–190, San Jose California, February 1992. SPIE.

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 332:533–536, 1986.

[29] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[30] E. M. Johansson, F. U. Dowla, and D. M. Goodman. Backpropagation learning for multi-layer feed-forward neural networks using the conjugate gradient method. *IEEE Transactions on Neural Networks*, 1991.

[31] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. Technical Report PB-339, Aarhus University, 1990.

[32] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and K. M. Hummels. On the training of radial basis function classifiers. *Neural Networks*, 5:595–603, 1992.

[33] D. Wettschereck and T. Dietterich. Improving the performance of radial basis function networks by learning center locations. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 1133–1140, San Mateo, 1991. Morgan Kaufmann.

[34] Donald F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.

[35] I. K. Sethi. Entropy nets: from decision trees to neural networks. *Proceedings of the IEEE*, 78:1605–1613, 1990.

[36] B. V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* IEEE Computer Society Press, 1991.