CAR-TR-691 October 1993
CS-TR-3162

Evaluation of Pattern Classifiers for Fingerprint and OCR Applications

J.L. Blue[1]
G.T. Candela[1]
P.J. Grother[1]
R. Chellappa[2]
C.L. Wilson[1]

[1]National Institute of Standards and Technology
Gaithersburg, MD 20899

[2]Department of Electrical Engineering
Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742-3275

Abstract

In this paper we evaluate the classification accuracy of four statistical and three neural network classifiers for two image based pattern classification problems. These are fingerprint classification and optical character recognition (OCR) for isolated handprinted digits. The evaluation results reported here should be useful for designers of practical systems for these two important commercial applications. For the OCR problem, the Karhunen-Loève (K-L) transform of the images is used to generate the input feature set. Similarly for the fingerprint problem, the K-L transform of the ridge directions is used to generate the input feature set. The statistical classifiers used were Euclidean minimum distance, quadratic minimum distance, normal, and k-nearest neighbor. The neural network classifiers used were multi-layer perceptron, radial basis function, and probabilistic. The OCR data consisted of 7,480 digit images for training and 23,140 digit images for testing. The fingerprint data consisted of 2,000 training and 2,000 testing images. In addition to evaluation for accuracy, the multi-layer perceptron and radial basis function networks were evaluated for size and generalization capability. For the evaluated datasets the best accuracy obtained for either problem was provided by the probabilistic neural network, where the minimum classification error was 2.5% for OCR and 7.2% for fingerprints.

1 Introduction

Over the last thirty years, significant progress has been made in the theory and design of pattern classifiers for a number of practical problems drawn from character recognition, fingerprint classification, biomedical applications, and automatic target recognition. Prior to the re-emergence of Neural Network (NN) techniques, the dominant paradigms were statistical, structural, and syntactic. Since the mid-eighties, NN techniques have raised the possibility of realizing fast, adaptive systems for pattern classification. In spite of all these advances, very little has been done on evaluating the different classifiers for one or more applications. Only recently, a probabilistic model has been proposed [1] for the evaluation of NN classifiers, with results on synthetic data. Known theoretical results on error bounds and probabilities are often based on ideal distributions of class conditional densities and/or on infinite samples. Thus a purely theoretical evaluation of a wide variety of classifiers for finite samples is very difficult. However, one can make such an evaluation by empirically observing the performance of a number of classifiers on a very large dataset. As of now, such studies have not been widely published. The reasons may be one or more of the following: a) inadequate computational resources, b) non-availability of large datasets, and c) a general dislike for doing evaluations or comparative studies. Given that pattern classification is a mature area and that several NN approaches have emerged, the time is ripe for an evaluation of the different classifiers for a specific application domain. Such a study should involve a large dataset and also should be unbiased.

Over the last year, the Image Recognition Group at the National Institute of Standards and Technology (NIST) has undertaken such evaluation tasks for two important problems: OCR and fingerprint classification. This paper is a report on this evaluation study. We evaluate four statistical classifiers and three NN classifiers. The statistical classifiers are Euclidean Minimum Distance (EMD), Quadratic Minimum Distance (QMD), Normal (NRML), and k-Nearest Neighbor (k-NN). The three neural classifiers included in the evaluation are the Multi-Layer Perceptron (MLP), Radial Basis Functions (RBF), and a Probabilistic Neural Network (PNN). For a given application, all the classifiers were given the same feature sets. Misclassification errors are tabulated as a function of feature dimension and classifier parameters such as the number of hidden units, etc. These tables should be very useful for designers of OCR and fingerprint classification systems. Although we have used moderately large datasets, we refrain from drawing conclusions on the performance of classifiers since the classifiers have not been optimized in terms of best features, code efficiency, etc. We hope that with the availability of enormous computational resources, more such evaluation studies on large datasets can be undertaken. A limited performance evaluation study using a statistical classifier and a backpropagation algorithm for the recognition of handwritten numerals can be found in [2].

The organization of the paper is as follows. After a brief introduction to the OCR and fingerprint classification problems in Section 1, we discuss, in Section 2, the datasets used in the experiments. The system components are described in Section 3. The structures of the various classifiers used are described in Section 4. Section 5 provides the results of classification experiments. In addition, variation of classification accuracy with respect to the size and generalization capability of NN is studied for RBF and PNN in Section 6.

1.1 The OCR problem

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 kept OCR research ongoing. Classification of loosely constrained handwritten digits, at least, is largely a solved problem [3].

A good review of OCR can be found in [4]. A huge quantity of research from academia and industry has yielded a multitude of algorithms for normalization [5, 6], feature extraction [7], and classification [8, 9, 10, 11], that are capable of OCR of digits. The popularity of OCR research has increased with the advent of NN paradigms applicable to feature extraction and classification. The advantage of many NN classifiers, once trained, is their efficiency. In future commercial segmentation *and* recognition efforts [12, 13], lack of efficiency will preclude using numerous techniques from the literature because of their computational requirements. The trade-off between classification performance and computational requirements has prompted this study of digit classifier efficacy.

1.2 The fingerprint problem

At least three major approaches have been taken to automatic fingerprint classification. These are the structural, syntactic, and artificial neural network (ANN) approaches. In the structural approach [14, 15, 16], one extracts features based on minutiae and represents the features using a graph data structure. Structural matching is done by exploiting the topology of the features. In the syntactic approach [17], one typically approximates ridge patterns [18] as strings of primitives and models the plausible strings from a class such as Tented Arch using production rules as in grammar. Depending on the type of grammar used one can see a significant drop in the number of production rules required. For example, a context-free grammar would require fewer production rules than a regular grammar. When a new fingerprint arrives whose identification is sought, one extracts the string of primitives and passes it through a parser. A successful parser output indicates the class to which the fingerprint belongs. Whether the parsing is successful or not, a description of the input string is always generated. The more general the grammar is, the more complex the parser tends to be. Applications of the syntactic approach using more complex grammars such as stochastic grammars [19] (where probabilities are associated with the production rules), tree grammars [20], and programmed grammars [21] have been considered for the fingerprint classification problem. The main stumbling block of the syntactic approach is that mechanisms for the inference of grammars from training samples have not been well understood [22, 23]. Recent advances in learning the structure of a grammar from training samples using neural nets [24] look promising.

The Image Recognition Group at NIST has recently implemented a massively parallel NN fingerprint classification system using a parallel computer of the SIMD variety (single-instruction-stream, multiple-data-stream) [25]. This system uses image-based ridge-valley features. Using K-L transforms, a significant reduction in feature vector dimensions is achieved. A MLP network trained using a conjugate gradient method is used for classification. It takes about 2.7 seconds to preprocess and classify a fingerprint using a massively parallel computer. The system is capable of 93% classification accuracy with 10% rejects. More recently [26], a fingerprint matching and classification system that uses a hierarchical pyramid structure has been reported. The matching module takes two images as inputs and outputs a number between 0 and 1. This number, which reflects the degree of belief that the two images are from the same finger, is estimated using a probabilistic Bayesian approach. The network parameters (about 104 in number) are trained using a steepest descent procedure that minimizes a cross entropy measure. Impressive matching results are reported. It is interesting to note that the receptive fields of the learning filter at the bottom of the pyramid appear to be edge or ridge orientation detectors, but sometimes correspond to minutiae detectors. This finding supports the choice of ridge direction components used as features in the NIST system.

1.3 Generalization

The focus of most NN applications has been on error minimization. A standard method of error minimization for real world problems is backpropagation [27] although more powerful methods of optimization have also been used [28, 29]. In addition to the problem of error reduction, effective generalization also requires that the information content of the network be reduced to some minimum value [30, 31, 32]. The resulting reduced network has the advantage of increased speed achieved by using fewer connections and is more effective in terms of the use of information capacity to achieve a specified pattern recognition accuracy.

The optimization strategy used in this research focuses on information content and the efficiency of information transferred to the network from the training set. This results in a smaller network with a very high information content that allows the use of a reasonably small training set. We have used the Boltzmann method as a secondary method of optimization to prune the networks used here [30, 31]. The method can be used in conjunction with a primary method of optimization such as a scaled conjugate gradient scheme [29]. The resulting optimized MLP network has been used for both fingerprint pattern classification and OCR.

In the case of RBF networks, the explicit network pruning used on MLP's is unnecessary. RBF networks are self-pruning to some degree. Unimportant connections are effectively pruned away by the training process learning a large width; each large width effectively deletes one connection from an input to one RBF and reduces the number of active parameters by two. More pruning is done with small training sets than with large ones, and more with large networks than with small ones.

2 Databases

2.1 OCR

The classifiers described in this report were trained and tested using feature vectors derived from the digit images of NIST Special Database 3 [33]. This database consists of binary 128 by 128 pixel raster images segmented from the sample forms of 2100 writers published on CD as [34]. Other results on segmentation and recognition of this database have been reported [35]. The relative difficulties of the NIST OCR databases have been discussed in [36]. For this study samples were 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 bounded by binary erosion and dilation, and consistent orientation is effected by shearing rows by an amount determined by the leftmost and rightmost pixels in the first and last rows defining a vertical line.

2.2 Fingerprints

The classifiers described in this report were trained and tested using feature vectors derived from the fingerprint images of NIST Special Database 4 [37]. This database consists of 8 bit per pixel gray level raster images of two inked impressions ("rollings") of each of 2000 different fingers. The feature vectors used to train the classifiers were made from the 2000 first-rollings, and those used to test the classifiers were made from the 2000 second-rollings. Every fingerprint in the database has an associated class label, assigned by experts. The two rollings of any finger have the same class, since the class of a fingerprint is not affected by variations that occur between different rollings of the finger.

Fingerprints as they naturally occur are not distributed equally into the five classes. We have taken a summary of the NCIC classes[1] of fingerprints from more than 22.2 million cards and reduced the numbers contained therein to estimates of the true frequencies or *a priori* probabilities of the five fingerprint classes. The estimated probabilities are .037, .029, .338, .317, and .279, for the classes Arch, Tented Arch, Left Loop, Right Loop, and Whorl, respectively.

The 2000 fingers represented in Special Database 4 are equally divided among the five classes. The database was produced this way, rather than by using a natural distribution, so as to increase the representation of the relatively rare, and also difficult, Arch and Tented Arch classes. This provides trainable classifiers with more examples with which to learn these difficult classes. The training and testing sets have equally many prints of each class.

3 System Components

Each of our experimental classifiers consists of a set of components as shown in Figure 1. The ovals represent input and output data, the rectangles represent processing components, and the arrows represent the flow of data. The components do not necessarily correspond to separate devices or programs; they merely represent a separation of the processing into conceptual units, so that the overall structure may be discerned. The inputs for OCR and fingerprints are extracted from the appropriate NIST databases. OCR images are a 32 pixel square binary raster containing a hand printed digit image extracted from a document. The feature extraction performs a K-L transform on the normalized character image. The fingerprint image is a raster of 512 by 512 8-bit grayscale pixels, produced by scanning the fingerprint card with a CCD camera. The fingerprint classifiers described in this report take as their input a small vector of numerical features derived from a fingerprint raster image. The fingerprint is reduced to 112 features (not all of which need be used) as follows. First, it is subjected to an FFT-based filter that increases the relative power of dominant frequencies, increasing the ratio of signal (fingerprint ridges) to noise. The local orientations

[1]The NCIC classification system separates fingerprints into numerous classes, which are basically the same as the classes used in the Henry system. The NCIC method for producing a card's class from the classes of its individual fingerprints is different than the summarizing method used in the Henry system.

of the ridges at 840 equally-spaced locations (a 28 by 30 grid) are then measured, using an orientation finder described in [25]. The orientation finder is based on a "ridge-valley" fingerprint binarizer described in [38]. It computes an orientation at the location of each pixel, then averages these basic orientations in nonoverlapping 16×16-pixel squares to produce the grid of 840 orientations. These are used as input to a translational registration module that attempts to standardize core location. The K-L transform of these modified ridge directions is taken as a compact classifiable representation of the fingerprint.

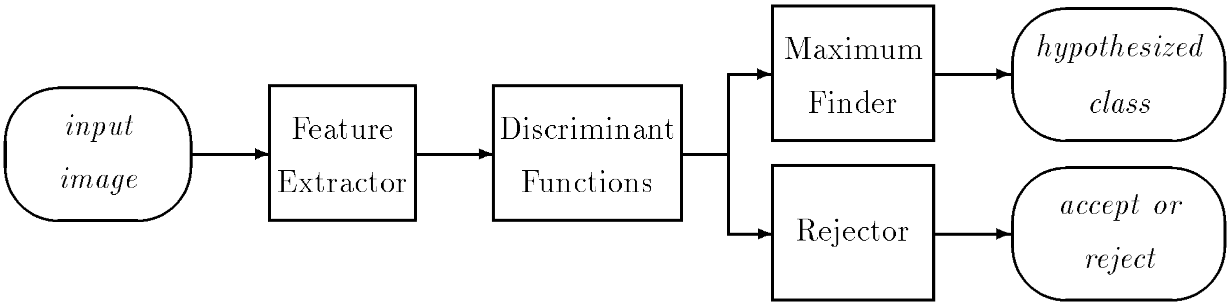


Figure 1: Components of Classification System

For both OCR and fingerprint classification the next component of the system is the bank of discriminant functions. There is one discriminant function for each class. Each one produces a single floating-point number, which tends to have a large value if the input image is of the corresponding class. The n-tuple of values produced by the bank of discriminant functions is sent to two final components, the maximum finder and the rejector. The maximum finder finds which one of the discriminant values is highest, and assigns its class as the *hypothesized class* of the image, that is, the classification system's best guess as to the class.

In the experiments reported here, the reject option was not exercised. Experiments including rejectors of the following form are reported in [39]. The outputs of the discriminant functions are fed to a "confidence function", which produces a number that is treated as if it were a measure of reliability of the classification decision made by the maximum finder. In other words, images that produce high values of the confidence function are considered to be more likely to have been assigned correct classes than those that produce lower confidence values. The following confidence function often produces good results: define its value to be the highest discriminant function value minus the second highest value. This is intuitively reasonable, since it will assign low confidence to examples that are near a hypothetical class boundary. Rejection of an image means that the classification system refuses to assign a class, because it cannot be sufficiently certain as to the correct class. As different classifiers can be optimized using different rejectors, we decided not to include rejectors in our evaluation tests. This could be a topic for future study.

4 Classifiers

Each classifier 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 and that some classifiers have attributes of more than one category. In the statistical pattern recognition literature [40] *parametric* classifiers use variables such as the estimated means and covariances to express the class density functions. The decision surfaces implicit in the EMD classifier are linear. Those of the QMD and NRML classifiers are quadratic. We categorize the EMD, QMD, and NRML methods as parametric classifiers. *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 WSNN, were considered. Finally, the neural net category contains MLP, RBF classifiers of two types (RBF1 and RBF2), and 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. The figures refer only to the OCR problem but it is notable that the corresponding figures for fingerprint are similar in form. Restriction of this graphical representation to two dimensions is undeniably, but necessarily, not ideal.

4.1 Notation

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

$$
\begin{aligned}
L &= \text{number of classes. For digits, } L = 10\text{, for fingerprints, } L = 5 \\
N &= \text{number of clusters, } N \geq L \\
p(i) &= \textit{a priori} \text{ probability of cluster } i \\
\hat{p}(i) &= \text{an estimate of } p(i) \\
n &= \text{dimensionality of features} \\
\mathbf{R}^n &= \text{the set of all } n\text{-tuples of real numbers} = \text{“feature space”} \\
\mathbf{x} &= \text{extracted “feature vector” of a image } (\mathbf{x} \in \mathbf{R}^n) \\
\mathbf{x}_j^{(i)} &= \text{feature vector from } j^{\text{th}} \text{ image of cluster } i \ (1 \leq i \leq N, 1 \leq j \leq M_i) \ (\mathbf{x}_j^{(i)} \in \mathbf{R}^n) \\
M_i &= \text{number of training images of cluster } i \ (1 \leq i \leq N) \\
\boldsymbol{\mu}_i &= \text{mean feature vector for cluster } i \ (1 \leq i \leq N) \ (\boldsymbol{\mu}_i \in \mathbf{R}^n) \\
\mathbf{m}_i &= \text{an estimate of } \boldsymbol{\mu}_i \\
\boldsymbol{\Sigma}_i &= \text{covariance matrix for cluster } i \ (1 \leq i \leq N) \ (\boldsymbol{\Sigma}_i \in \mathbf{R}^{n \times n}) \\
\mathbf{S}_i &= \text{an estimate of } \boldsymbol{\Sigma}_i \\
d^2(\mathbf{x}, \mathbf{y}) &= (\mathbf{x} - \mathbf{y})^{\mathrm{T}}(\mathbf{x} - \mathbf{y}) = \text{squared Euclidean distance between } \mathbf{x} \text{ and } \mathbf{y} \ (\mathbf{x}, \mathbf{y} \in \mathbf{R}^n) \\
r^2(\mathbf{x}, \mathbf{y}, \mathbf{z}) &= \sum_{i=1}^{n} ((x_i - y_i)/z_i)^2 \\
&= \text{distance between } \mathbf{x} \text{ and } \mathbf{y} \text{ normalized by } \mathbf{z} \ (\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{R}^n) \\
D_i(\mathbf{x}) &= i^{\text{th}} \text{ discriminant function } (1 \leq i \leq N, \mathbf{x} \in \mathbf{R}^n)
\end{aligned}
$$

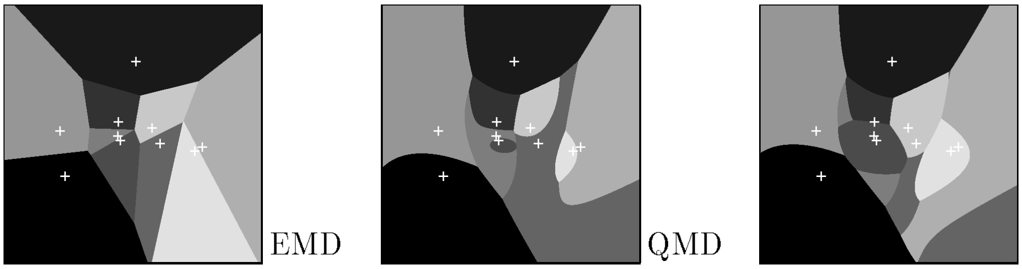
4.2 Parametric Classifiers

4.2.1 Euclidean Minimum Distance Classifier

This is perhaps one of the simplest classifiers that one can design. The 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. Each cluster region is bounded by a convex polygon. In the one cluster per class case these regions are the hypothetical class decision regions. This classifier suffers from the same linear separability limitations as the Perceptron critiqued by Minsky and Papert [41]. In the many clusters per class case the union of the cluster regions defines the class decision region whose boundary is then piecewise linear. Figure 2 shows the class for regions when only two features and one cluster per class are used. The estimated cluster mean vectors $\mathbf{m}_i$ are marked with plus signs.



EMD QMD NRML

Figure 2: Parametric classifiers. For EMD note the perpendicular bisectors. For QMD note the quadratic forms of the decision boundaries. The + signs indicate the locations of the estimated class means.

4.2.2 Quadratic Minimum Distance 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:

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

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 NRML classifier, described below. Figure 2 shows the resulting class regions. The form of these figures is similar for the fingerprint classification problem. As in the EMD classifier, one can use several prototypes to represent each class. 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, preventing its conventional inverse from being evaluated.

4.2.3 Normal Classifier

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

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

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 incorrect classifications produce equal loss values of 1. In this case, the Bayesian classifier is the one that classifies each unknown $\mathbf{x}$ to the class i for which the *a posteriori* probability $p(i|\mathbf{x})$ is highest. According to Bayes' rule [42],

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

Since the value of the mixture density $p(\mathbf{x})$ has no effect on which possible i value maximizes $p(i|\mathbf{x})$, $p(\mathbf{x})$ can always be omitted. Further, for a pattern recognition problem in which the *a priori* probabilities are the same, $p(i)$ can be ignored. The result is to classify $\mathbf{x}$ to the cluster i for which $p(i)p(\mathbf{x}|i)$ 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 we are able to define the discriminant function for the Normal classifier as

$$D_i(\mathbf{x}) = 2 \log \hat{p}(i) - \log |\mathbf{S}_i| - (\mathbf{x}-\mathbf{m}_i)^{\mathrm{T}} \mathbf{S}_i^{-1} (\mathbf{x}-\mathbf{m}_i)$$

since the squaring and taking of logarithms has no effect on which i maximizes the discriminant. The hypothetical class regions are given in Figure 2.

4.3 Nearest Neighbor Classifiers

Nearest-neighbor classifiers have been the subject of decades of research (see, for example, Dasarathy's collection of papers [43]). The following are simple and ubiquitous yet effective examples of such methods.

4.3.1 k-Nearest Neighbor

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 [8] 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 3 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 maybe 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:

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

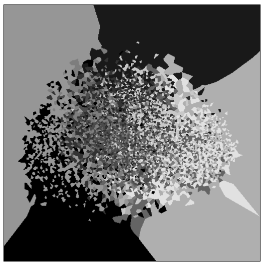

1-NN

Figure 3: Single nearest neighbor classifier. Note the very intricate non-contiguous decision boundaries local to each training prototype.

4.3.2 Weighted Several Nearest Neighbors

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

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

$$V_{\mathbf{x}}^{(i)} = |\mathcal{S}_{\mathbf{x}}^{(i)}| = \text{number of "votes" for class } i$$

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

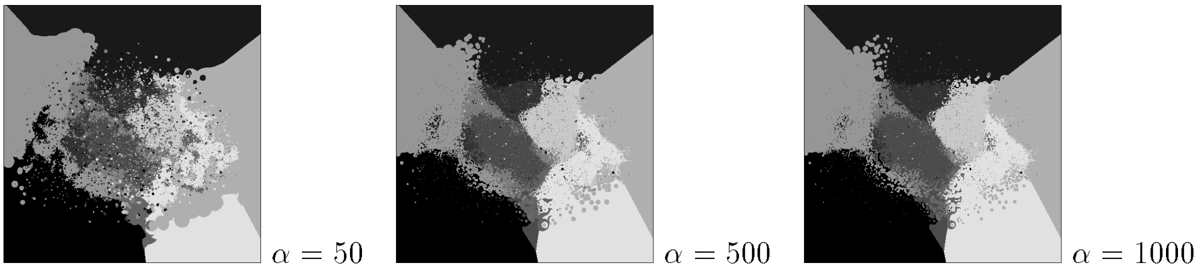


Figure 4: Weighted several nearest neighbors. In the limit of small α this classifier defaults to 1-NN. Note the fine grained structure throughout that is typical of nearest neighbor methods.

4.4 Neural Net Classifiers

4.4.1 Multi-Layer Perceptron

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:

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

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 "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). 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 the addition of a scalar "regularization" term [44]. 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 [29].

Networks of the MLP type are the most commonly used "neural nets" in use today, and they are usually trained using a "backpropagation" algorithm [45]. A "scaled conjugate gradient" training method [46, 47, 28, 29] was used in our research instead of the ubiquitous backpropagation method, training speed gains of an order of magnitude being typical. Figure 5 shows MLP class regions resulting from varying the first two inputs to a trained 8 input, 48 hidden unit network.

Figure 5: MLP classification and confidence maps. From left: class boundaries, highest discriminant value, difference in highest two discriminant values.

4.4.2 Radial Basis Functions

Neural nets of the RBF 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 RBF 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 referred to as RBF1 and RBF2. The following notation will be convenient:

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

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

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

and for RBF2, they are of the biased sigmoidal form

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

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 can be found in [48, 49]. 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 6 shows RBF1 class regions resulting from the use of 1, 2, 4, and 6 hidden nodes per class, and Figure 7 shows RBF2 class regions for the same numbers of hidden nodes per class.

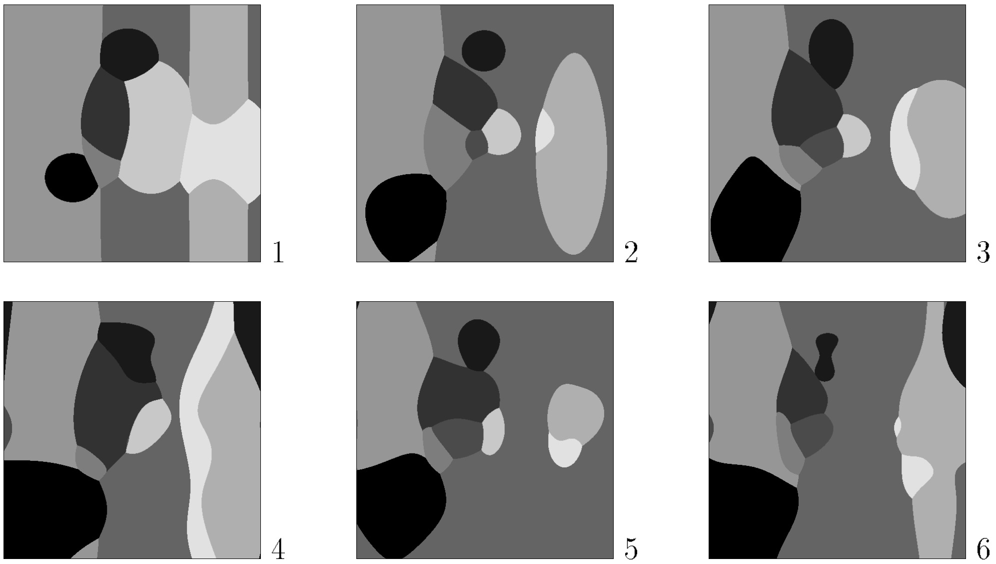


Figure 6: RBF1 classification regions for increasing numbers of centers per class.

4.4.3 Probabilistic Neural Net

This classifier was proposed in a recent paper by Specht [50]. 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

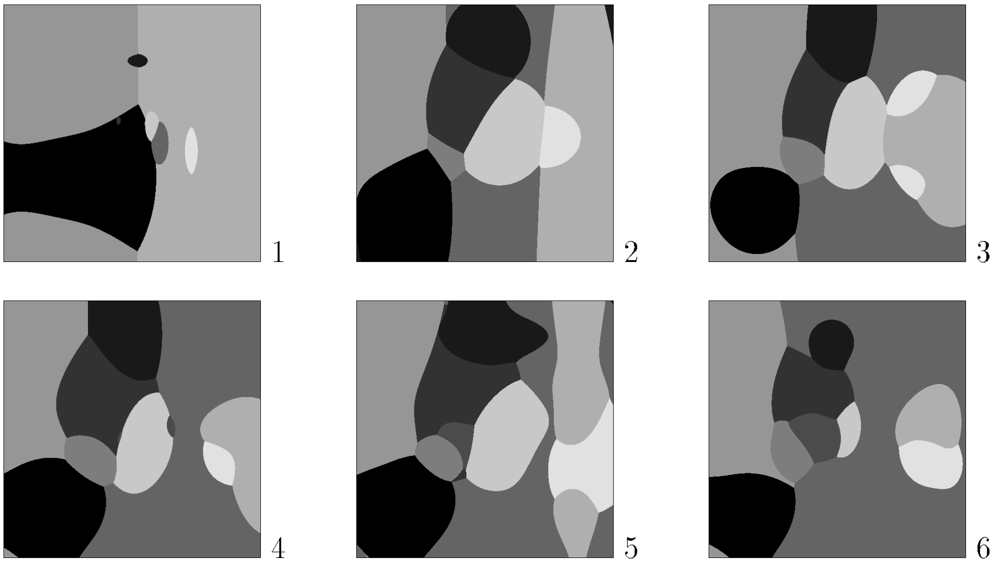


Figure 7: RBF2 classification regions for increasing numbers of centers per class.

computing, for each class i, the sum of the values of the class-i kernels at $\mathbf{x}$, multiplying these numbers by compensatory factors involving the estimated *a priori* probabilities, and picking the class whose resulting discriminant value is highest. Many forms are possible for the kernel functions; we obtained our best results using radially symmetric Gaussian kernels. The resulting discriminant functions are of the form

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

where σ is a scalar "smoothing parameter" that can be optimized by trial and error. Figure 8 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 and are similar to those produced by parametric statistical and RBF classifiers.

5 Results of Classification

5.1 OCR

Table 1 shows for each classifier the estimated probabilities of *error*, expressed as percentages, for increasing dimensionality of the K-L feature set. Note that the optimal number of features

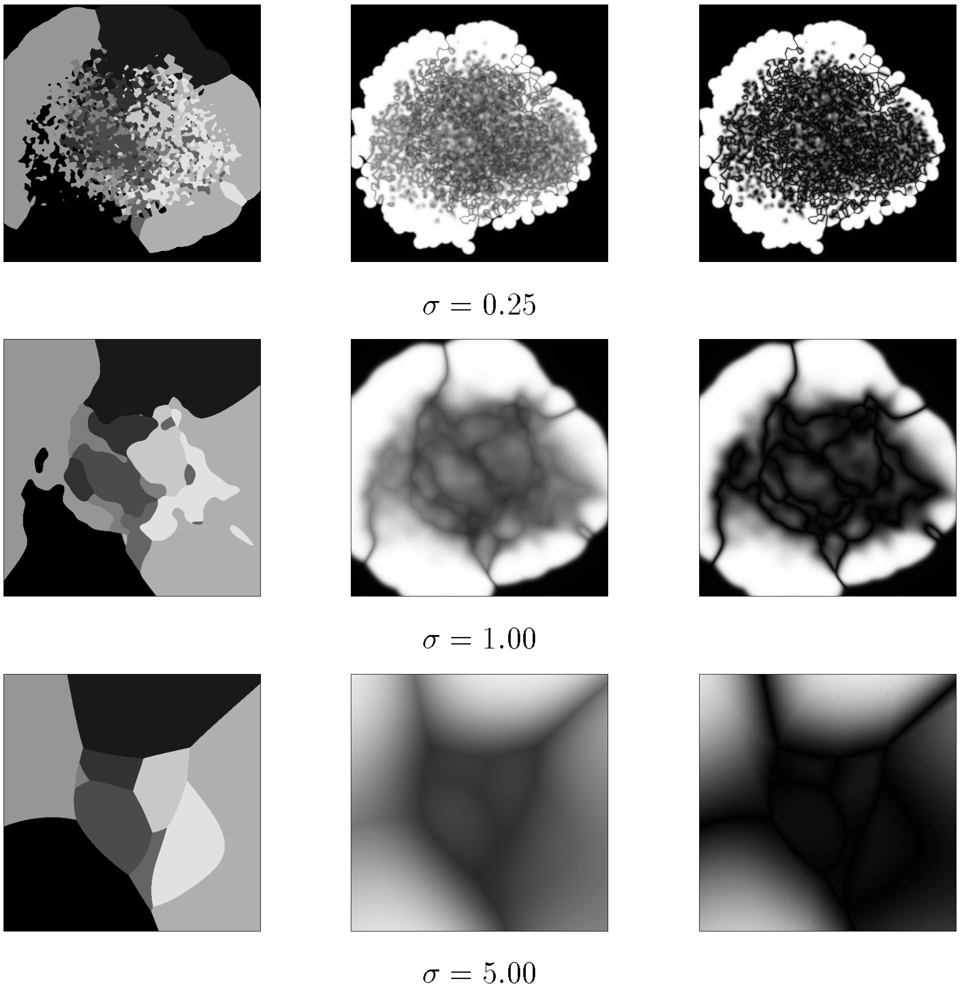


Figure 8: PNN classification and confidence maps in two dimensions for increasing σ values. From left: class boundaries, highest discriminant value, difference in highest two discriminant values.

yielding lowest classification error (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 essentially attain a plateau as the number of features reaches approximately 32, thereafter gaining only a few tenths of a percent. The best classifiers are the computationally expensive nearest neighbor classifier and the related PNN. They achieve one third fewer errors than the neural networks and parametric classifiers. The optimum value $\alpha = 1.1$ 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.

Two caveats may be stated about the table. First, the MLP and RBF results depend on the initial guesses for the parameters. Often a number of different random guesses are tried to assess the effect of the initial guess; for this table, because of the magnitude of the calculation necessary, only one initial guess was used. Second, the RBF calculations were done with a sigmoidal output layer; the later work summarized in Section 6.2 suggests that better results can be obtained with a linear output layer.

5.2 Fingerprints

The test set used in these experiments—the second rollings of Special Database 4—consists of an equal number (400) of each of the five classes of fingerprints. Because naturally occurring prints have a very unequal distribution into classes, it would be a mistake to use the percentage of this test set incorrectly classified as an estimate of the probability that a classifier will incorrectly classify a naturally occurring print. Instead, the following calculations are used to produce the test "score" (estimated probability of incorrect classification). For each class i, the number of the 400 class-i test prints (i.e. test prints whose actual class is i) that were incorrectly classified was counted; this count is denoted by w_i. Clearly, $w_i/400$ can be used as an estimate of the conditional probability of incorrect classification of a print given that the actual class of the print is i. Therefore,

$$\sum_{i=1}^{N} \hat{p}(i) w_i / 400$$

Table 1: Dependence of classification error on KL transform dimensionality for digit recognition. 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 hidden units; for RBF networks, the number of centers per class; and for EMD and QMD, the number of clusters per class. Boldface indicates optimum dimensionality for each classifier.

System	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64
k-NN: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
k-NN: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
k-NN: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

can be used as an estimate of the probability of incorrect classification. The accuracy "scores" mentioned below are these estimated probabilities, expressed as percentages.

Table 2 shows the lowest error rate that was obtained for each type of classifier. Also listed, for each classifier type, are the optimal settings that were found for other parameters: number of K-L features used; type of training set—the full "balanced" set of 400 prints of each class, or a "natural" set produced by discarding some of the training prints so as to cause the frequencies to be approximately equal to the estimated *a priori* probabilities; and, for some of the classifier types, another adjustable parameter or a number of hidden nodes. Table 3 shows, for each classifier type, the lowest error rates that were obtained for each of several numbers of features; it is clear that the optimal number of features is not the same for all of these classifier types, as was the case for characters.

Table 2: Lowest error percentages for the various classifier types, and the parameters that produced them for the fingerprint problem.

Classifier	Error %	No. of features	Training set	Other parameter values
EMD	26.2	80	balanced	–
QMD	12.8	16	balanced	–
NRML	11.3	28	balanced	–
1-NN	9.0	96	natural	–
WSNN	8.9	96	natural	$\alpha = 1.09$
MLP	8.2	64	natural	64 hidden nodes
RBF1	8.3	112	natural	70 hidden nodes
RBF2	8.1	64	natural	110 hidden nodes
PNN	7.2	112	balanced	$\sigma = 2.26$

6 Generalization Experiments

6.1 Information Based Methods

In the MLP generalization experiments, K-L features were used to train MLP's using different methods of statistical size reduction. Only the training and recognition parts of the system were involved in the test. For the OCR problem two sets of 10,000 K-L features derived from characters taken from NIST Special Database 3 [33] were used. For the fingerprint problem

Table 3: Fingerprint classification error percentages as a function of feature dimensionality. NRML produced a smaller error percentage for a number of features not in the table: 11.3, for 28 features.

Classifier	Number of features						
	16	32	48	64	80	96	112
EMD	26.9	26.6	26.4	26.3	26.2	26.3	26.3
QMD	12.8	15.6	18.0	20.1	20.7	21.6	23.0
NRML	13.5	12.8	16.8	18.1	19.7	20.7	23.0
1-NN	10.7	9.6	9.7	9.3	9.1	9.0	9.3
WSNN	10.3	9.3	9.1	9.1	8.9	8.9	9.0
MLP	9.1	8.8	8.6	8.2	8.2	8.4	8.5
RBF1	9.8	8.6	9.1	8.8	8.8	8.5	8.3
RBF2	10.7	9.5	10.7	8.1	8.8	8.4	8.2
PNN	9.0	7.9	7.5	7.6	7.4	7.3	7.2

two sets of 2,000 K-L features derived from fingerprints from NIST Special Database 4 [37] were used.

The scaled conjugate gradient (SCG) method of [29] is used to obtain a starting network for the Boltzmann weight pruning algorithm. For the OCR problem the network has an input layer with 48 nodes, a hidden layer with 64 nodes, and an output layer with 10 nodes. For the fingerprint problem, the network has an input layer with 128 input nodes, a hidden layer with 128 nodes, and an output layer with five nodes. In both cases the initial network is a fully connected network. The pruning using the Boltzmann method was carried out by selecting a normalized temperature, T, and removing weights based on a probability of removal

$$P_i = \exp(-w_i^2/T).$$

The values of P_i are compared to a set of uniformly distributed random numbers, R_i, on the interval $[0,1]$. If the probability P_i is greater than R_i then the weight is set to zero. The process is carried out for each iteration of the SCG optimization process and is dynamic. If a weight is removed it may subsequently be restored by the SCG algorithm; the restored weight may survive if it has sufficient magnitude in subsequent iterations.

This method can be modified to include information about the strength of the input features so that

$$P_i = \exp(-\lambda_j w_i^2/T),$$

where λ_j is the eigenvalue associated with the jth K-L feature for weights connected to these features in the input layer and $\lambda_j = 1$ for weights connecting the hidden and output layers. This method of pruning is referred to as eigenvalue-weighted pruning.

During this optimization process two important measures of information content are calculated [51]. The information capacity of the network, C, is given by

$$C = N_{wts}(\log_2|w_{\max}| - \log_2|w_{\min}| + 1),$$

where N_{wts} is the number of non-zero weights, $w_{\max}$ is the weight with the largest magnitude, and $w_{\min}$ is the weight with the smallest magnitude. The entropy is given by

$$H = C - \sum_{i=1}^{N_{wts}} \log_2|w_i| + N_{wts}(1 - \log_2|w_{\min}|).$$

The effect on the information content of the network can be evaluated by examining the distribution of weights in the network as a function of temperature or by evaluation of the information capacity of the network.

The results of using Boltzmann and eigenvalue-weighted pruning during the training of a network for the solution of the OCR problem are shown in Table 4. The results of using Boltzmann and eigenvalue-weighted pruning during the training of a network for the solution of the fingerprint problem are shown in Table 5. The statistical evaluation of each network was carried out using the equations provided in the previous section. Examination of these results shows two distinct results. The OCR problem is easier to solve than the fingerprint problem and the efficiency of information transfer in both cases is improved by the eigenvalue weighting of the pruning.

In every statistical measure of network capacity and accuracy, the OCR network pruned with the eigenvalue-weighted pruning function is superior to the Boltzmann pruned network. Recognition accuracy is higher at all temperatures for both testing and training. At the two critical temperatures accuracy is 93.5% for the Boltzmann case and 93.6% for the eigenvalue case. The critical temperature, T_c, is the temperature where the loss of information from pruning is equal to the information gained by the CG optimization. The number of weights used is 1186 in the Boltzmann case and 1065 in the eigenvalue-weighted case. The capacity-error product is lower in the eigenvalue case and the bits per weight are higher. This indicates that the information transfer during training is more efficient for eigenvalue-weighted training.

Table 4: Parameters of the pruned network for the OCR problem using Boltzmann pruning and eigenvalue-weighted Boltzmann pruning.

Parameter	Boltzmann	Eigen
T_c	0.07	0.77
Weights	1186	1065
Maximum weights	3786	3786
Capacity (bits)	11146	10281
Maximum capacity (bits)	41646	41646
Accuracy (%)	93.5	93.6
Maximum accuracy (%)	94.8	94.8
Minimum error $\times$ capacity	658	560
Bits per weight	8.00	9.79

Table 5: Parameters of the pruned network for the fingerprint problem using Boltzmann pruning and eigenvalue-weighted Boltzmann pruning.

Parameter	Boltzmann	Eigen
T_c	0.404	0.737
Weights	667	1046
Maximum weights	17157	17157
Capacity (bits)	4120	6632
Maximum capacity (bits)	171570	171570
Accuracy (%)	71.8	78.1
Maximum accuracy (%)	84.3	84.3
Minimum error $\times$ capacity	1177	1447
Bits per weight	6.34	7.14

The result for the fingerprint problem are more complex. Some statistical measures of network capacity and accuracy for the fingerprint network pruned with the eigenvalue-weighted pruning function are superior to the Boltzmann pruned network and some are not. Recognition accuracy is higher for the eigenvalue-weighted case at all temperatures for both testing and training. At the two critical temperatures accuracy is 71.8% for the Boltzmann case and 78.1% for the eigenvalue case. The number of weights used is 667 in the Boltzmann case and 1046 for the eigenvalue-weighted case. The Boltzmann pruned network is smaller but less accurate. The capacity-error product is high in the eigenvalue case both because there are more weights and because the number of bits per weight is higher. This indicates that the information transfer during training is more efficient for eigenvalue-weighted training and that more information is retained.

6.2 Sample Size Based Methods

The RBF generalization experiments used training and test sets, each containing 10,000 character images from NIST Special Database 3 [33]. Each feature set was a truncated K-L expansion of a 32 by 32 pixel binary image of a hand-printed digit. The RBF networks considered have 24 to 48 inputs, a hidden layer consisting of from one to four RBFs per digit, and an output layer of 10 linear or sigmoidal nodes. Also considered are some standard MLP networks, with sigmoidal hidden and output layers.

The supervised learning minimized the standard objective function, the sum of squares of the output errors. For networks with a sigmoidal output layer, a small constant times the sum of squares of the output layer weights was added to the objective function as a regularization, i.e. to minimize over-training. In order to simplify the gradient calculation, the inverses of the widths, $s_{ij} = 1/\sigma_{ij}$, were used as variables.

The optimization (training) was done using a combination of scaled conjugate gradients [28, 29] and a limited-memory quasi-Newton algorithm [52]. The program was written to allow any combination of the centers, widths, and weights to be learned, and the remainder to stay fixed. Training was done with varying training set sizes, from 156 patterns to the entire 10,000 patterns; testing was done on the entire 10,000 pattern testing set.

The initial values for the RBF centers were obtained from a k-means algorithm [53]. The widths produced by the k-means algorithm were not directly useful. Instead, uniform widths, several times the typical widths from the k-means algorithm, were used. It proved better to make the Gaussians much too broad than too narrow; the exact value used is unimportant as long as it is large enough. The importance of large widths may be understood by the following argument.

Suppose a pattern $\mathbf{x}$ has all its r^2 values so large that their activations are essentially zero. Then the contribution of $\mathbf{x}$ to the gradient of the objective function will be essentially zero for all centers, and the pattern will not influence the training at all. An extreme case of this behavior can be seen by taking all widths much too small. Then all RBFs produce zero for all patterns, and all the optimization can do is to adjust the bias terms in the output layer; the process converges rapidly to a poor local minimum.

For the same reason, random output layer weights do not work well for RBF networks. In the work reported here, the initial weights used were "sensible": positive for the weights from the centers to their corresponding output nodes, zero for the remaining weights and for the biases.

The number of free parameters in the experiments reported here ranged from 570 to 4250. The objective function has multiple local minima and is sensitive to details of the initial values; a relatively small change in the initial values for the parameters generally results in finding a different local minimum. For each network, ten different sets of initial conditions were used; for RBFs, it proved adequate to use a random $\pm 5\%$ variation on the widths. For MLPs, initial weights were chosen from a uniform random distribution in $(-0.5, +0.5)$.

Two strategies were used in training. The first is to train on successively larger subsets of the 10,000 pattern training set: 156, 312, 625, 1250, 2500, 5000, and finally 10,000 patterns. Training on the smallest sets goes quickly, and each set of parameters is a good initial guess for training on the next larger training set, but there is a possibility of wasting some work. The second strategy is to train only on the full training set.

The first strategy was, on average, faster, but not drastically faster. It has the added advantage of providing extra information, as seen in Figures 9 and 10. Especially for larger networks, the first strategy, on average, provides better training and testing.

Keeping the centers and widths fixed and learning appropriate weights resulted in poor training and poor testing in networks with only one or a few RBFs per class. Accordingly, centers and widths were also learned. Using initial widths from the k-means algorithm also resulted in poor training and testing; the optimization got stuck in a poor local minimum. Accordingly, all initial widths were then set to the same reasonably large value, with a small

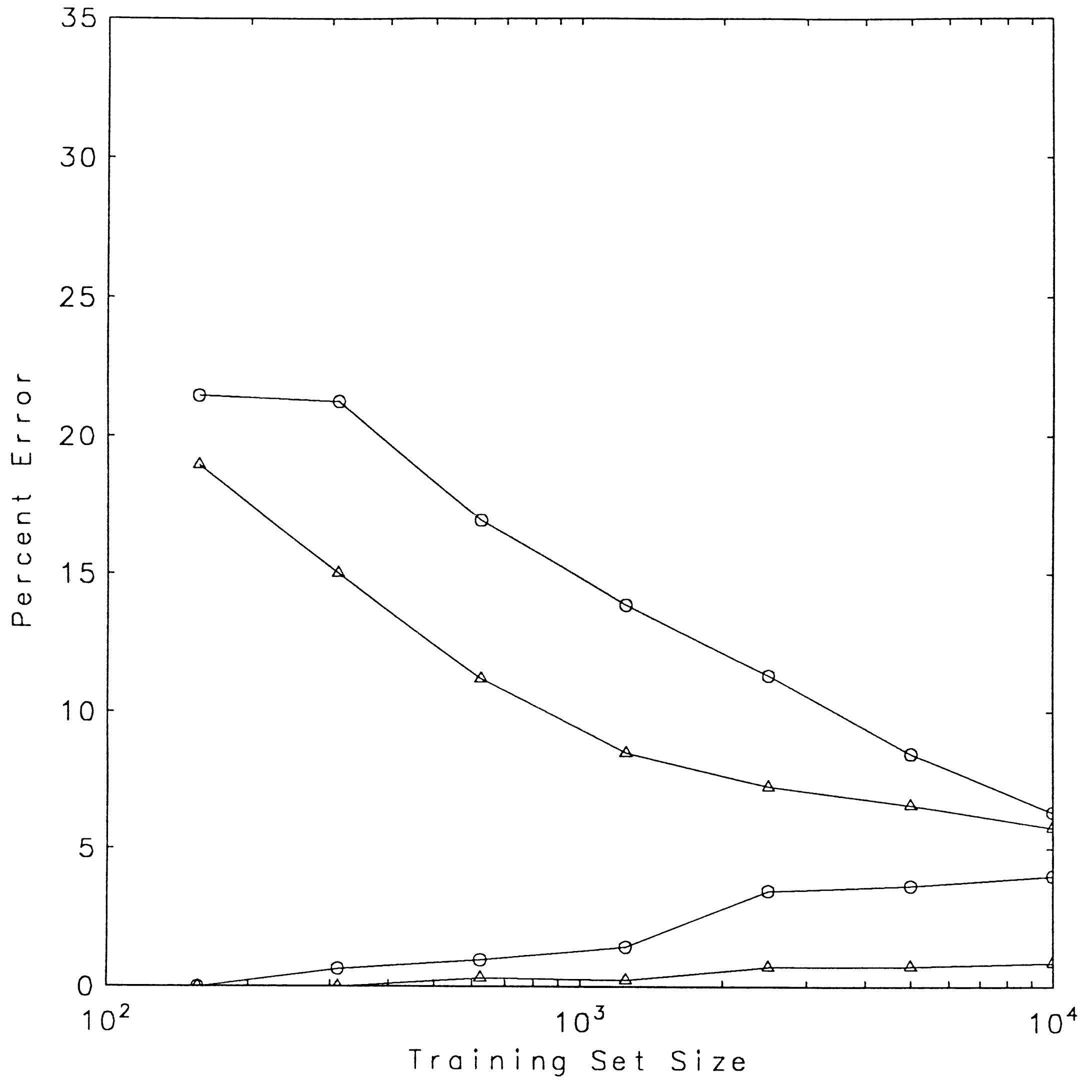


Figure 9: Testing error (top curves) and training error (bottom curves) versus training set size for MLP networks. The results are shown for the random start with the best testing error when trained with the full training set. 48-18-10 (◯) 48-36-10 (△).

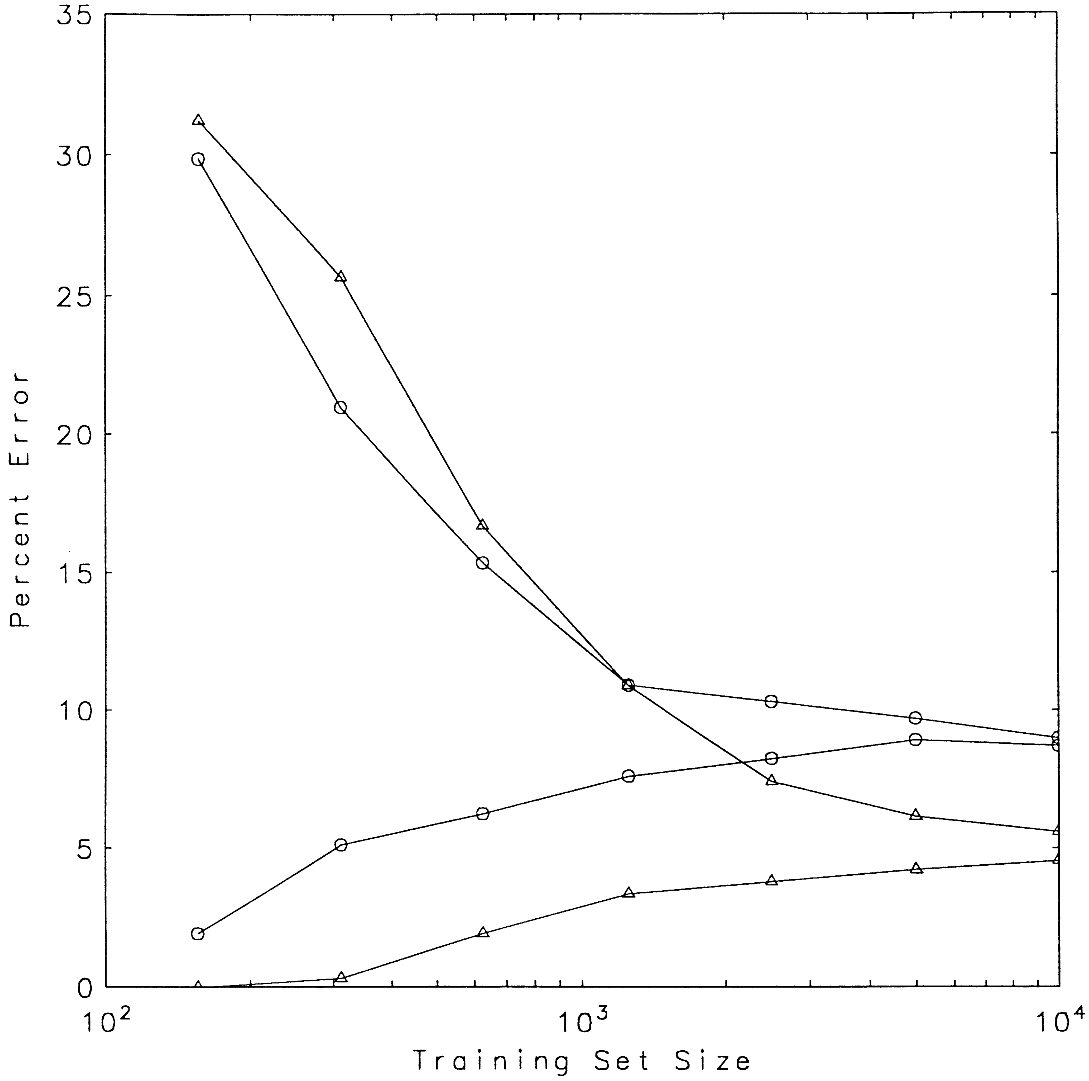


Figure 10: Testing error (top curves) and training error (bottom curves) versus training set size for RBF type 1 networks with a linear output layer. The results are shown for the random start with the best testing error when trained with the full training set. 48-20-10 (◯) 48-40-10 (△).

random variation.

In general, RBF networks with sigmoidal output layers trained significantly more slowly than RBF networks with linear output layers, and gave somewhat worse training and testing errors.

In general, RBF2 networks trained more slowly than RBF1 networks and gave slightly poorer training and testing errors. However, RBF2 networks with sigmoidal output layers have the useful property that the output weights can be fixed at reasonable values, rather than learned, with little or no worsening of the training and testing error.

RBF networks are self-pruning to some degree. Unimportant connections are effectively pruned away by the training process learning a large width, σ_{ij}; each large width effectively deletes one connection from an input to one RBF and reduces the number of active parameters by two. More pruning is done with small training sets than with large ones, and more with large networks than with small ones. Some results are shown in Table 6.

Table 6: Networks used and free parameters for each; for RBF networks, active number used in best solution found training on the full training set.

RBF/MLP	Structure	Parameters	Active
R	24-10-10	590	566
R	24-20-10	1170	1014
R	24-30-10	1750	1448
R	36-10-10	830	766
R	36-20-10	1650	1364
R	48-10-10	1070	982
R	48-20-10	2130	1650
R	48-30-10	3190	2122
R	48-40-10	4250	2870
M	24-16-10	570	
M	24-24-10	850	
M	24-36-10	1270	
M	48-18-10	1072	
M	48-36-10	2134	

The remainder of this section refers only to RBF1 networks with linear output layers.

Compared to MLP networks of a similar size (i.e., similar numbers of free parameters to optimize), RBF networks in general train at about the same rate. Their behavior versus training set size is different, though. Figure 1 gives results for a small (24-16-10, 570 parameters) and a large (48-36-10, 2130 parameters) MLP network. The large network gives quite accurate training results, much better than the small one, but the testing error is not much different for large training set sizes.

For comparison, Figure 10 gives results for a small (24-10-10, 590 parameters) and a medium (24-30-10, 1750 parameters) RBF network. The large network does not train as accurately, but there is much less difference in training and testing accuracy than for the MLP networks. In other words, the RBF networks are less likely to overfit the training data.

Figure 11 summarizes many hours of computation for MLP and RBF networks. Training and testing results from ten random starts are shown for each network.

The RBF results are closer to the diagonal, the diagonal being as good as one could ever expect. The smaller networks are closer to the diagonal than the larger ones; 10,000 training patterns are sufficient to train the small networks as well as they can be trained, but more patterns are needed for the larger networks.

The MLP results are farther from the diagonal. Increasing the network size gives better training error, but no better testing error. Many more training patterns are needed.

Figure 12 shows testing error versus number of free parameters. For a small number of free parameters, MLP networks do better.

7 Conclusions

Examination of Table 1 and Table 2 shows that even on problems as diverse as OCR and fingerprint classification the ranking of the methods is similar. The neighbor-based methods are the most accurate with PNN being the best of them. The comparison of MLP and RBF methods show that RBF is usually the better method. Generalization experiments on MLP and RBF networks (see Tables 4, 5, 6) demonstrate that fully connected MLP networks contain far too many free parameters for efficient information use. RBF networks can be constructed which are self-pruning and which achieve better accuracy for a given training

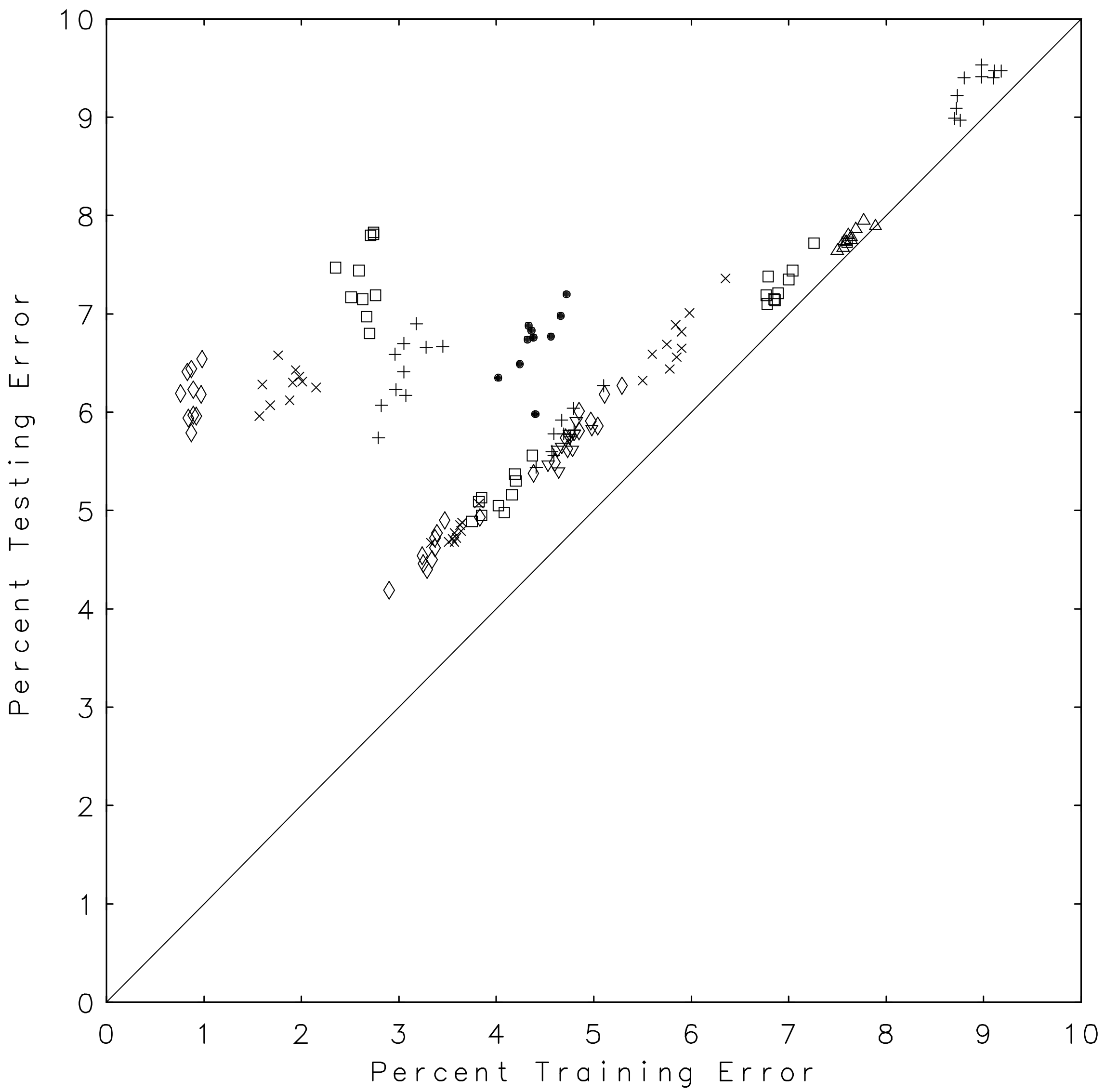


Figure 11: Training error versus testing error for different networks. All training is on the full training set. The symbols near the upper left of the figure are MLP networks: 24-16-10 (●), 24-24-10 (+), 24-36-10 (×), 48-18-10 (□), and 48-36-10 (◇). The symbols nearer the diagonal line are RBF type 1 networks with a linear output layer: 24-10-10 (+), 24-20-10 (×), 24-30-10 (lower group of +), 24-40-10 (lower group of ×), 36-10-10 (△), 36-20-10 (▽), 48-10-10 (□), 48-20-10 (◇), 48-30-10 (lower group of □), and 48-40-10 (lower group of ◇).

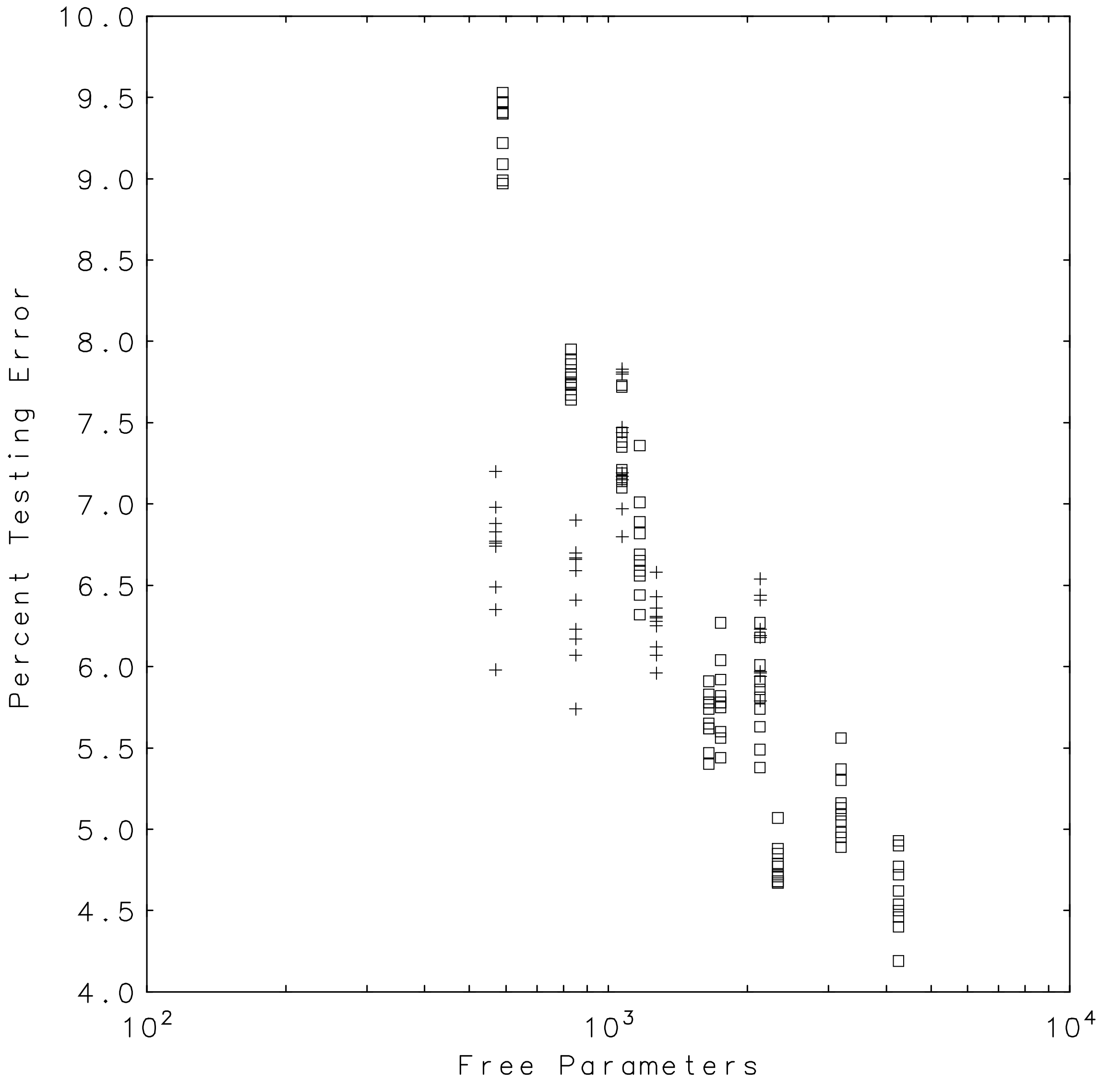


Figure 12: Testing error versus number of free parameters for MLP networks (+) and for RBF networks (□).

set size. When MLP and RBF methods are compared to multicluster EMD and QMD methods the NN methods are more straightforward to implement but do not show a clear accuracy advantage. All of the experiments presented here also suggest that the training set sizes used, although large, are not sufficient to fully saturate most of the machine learning methods studied here.

Acknowledgement

The authors would like to thank Jon Geist for assistance in interpretation of the results of the classifier comparisons.

References

[1] M.T. Musavi, K.H. Chan, D.M. Hummels, K. Kalantri, and W. Ahmed. A probabilistic model for evaluation of neural network classifiers. *Pattern Recognition*, 25:1241–1251, 1992.

[2] J. Cao, M. Shridhar, F. Kimura, and M. Ahmadi. Statistical and neural classification of handwritten numerals: A comparative study. In *11th IAPR International Conference on Pattern Recognition*, pages 643–646, The Hague, The Netherlands, 1992.

[3] 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 Conference. Technical Report NISTIR 4912, National Institute of Standards and Technology, August 1992.

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

[5] F.L. Alt. Digital pattern recognition by moments. In G.L. Fischer *et al.*, editors, *Optical Character Recognition*, pages 159–179. McGregor & Werner, 1962.

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

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

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

[9] 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 Kaufmann, 1990.

[10] 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.

[11] K. Fukushima, T. Imagawa, and E Ashida. Character recognition with selective attention. In *Proceedings of the IJCNN*, volume 1, pages 593–598, 1991.

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

[13] 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, 1991.

[14] M.K. Sparrow and P.J. Sparrow. A Topological Approach to the Matching of Single Fingerprints: Development of Algorithms for Use on Rolled Impressions. Technical Report Special Publication SW-124, National Bureau of Standards, Washington, DC, October 1985.

[15] P.K. Isenor and S.A. Zapy. Fingerprint identification using graph matching. *Pattern Recognition*, 19:113–122, 1986.

[16] A.K. Hrechak and J.A. McHugh. Automated fingerprint identification using structured matching. *Pattern Recognition*, 23:893–904, 1990.

[17] K.S. Fu. *Syntactic Pattern Recognition*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[18] B. Moayer and K.S. Fu. A syntactic approach to fingerprint pattern recognition. *Pattern Recognition*, 7:1–23, 1975.

[19] B. Moayer and K.S. Fu. An application of stochastic languages to fingerprint pattern recognition. *Pattern Recognition*, 8:173–179, 1976.

[20] B. Moayer and K.S. Fu. A tree system approach for fingerprint pattern recognition. *IEEE Transactions on Computers*, 25:262–274, 1976.

[21] K. Rao and K. Balck. Type classification of fingerprints: A syntactic approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:223–231, 1980.

[22] K.S. Fu and T.L. Booth. Grammatical inference: Introduction and survey—Part I. *IEEE Transactions on Systems, Man, and Cybernetics*, 5:95–111, 1975.

[23] K.S. Fu and T.L. Booth. Grammatical inference: Introduction and survey—Part II. *IEEE Transactions on Systems, Man, and Cybernetics*, 5:409–423, 1975.

[24] C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun, and Y.C. Lee. Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*, 4:380, 1992.

[25] C.L. Wilson, G.T. Candela, P.J. Grother, C.I. Watson, and R.A. Wilkinson. Massively Parallel Neural Network Fingerprint Classification System. Technical Report NISTIR 4880, National Institute of Standards and Technology, July 1992.

[26] P. Baldi and Y. Chauvin. Neural networks for fingerprint matching and classification (abstract). In *Proceedings, Neural Information Processing Systems Conference*, 1992.

[27] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed*

Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, Chapter 8, pages 318–362. MIT Press, Cambridge, MA, 1986.

[28] M.F. Møller. A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. Technical Report PB-339, University of Aarhus, November 1990.

[29] J.L. Blue and P.J. Grother. Training feed forward networks using conjugate gradients. In *Conference on Character Recognition and Digitizer Technologies*, SPIE volume 1661, pages 179–190, San Jose, CA, February 1992.

[30] O.M. Omidvar and C.L. Wilson. Optimization of neural network topology and information content using Boltzmann methods. In *Proceedings of the IJCNN*, Volume 4, pages 594–599, June 1992.

[31] O.M. Omidvar and C.L. Wilson. Topological separation versus weight sharing in neural network optimization. In S.S. Chen, editor, *Neural and Stochastic Methods in Image and Signal Processing*, SPIE volume 1766, San Diego, CA, 1992.

[32] I. Guyon, V.N. Vapnick, B.E. Boser, L.Y. Botton, and S.A. Solla. Structural risk minimization for character recognition. In R. Lippmann, editor, *Advances in Neural Information Processing Systems*, volume 4, pages 471–479. Morgan Kaufmann, 1992.

[33] M.D. Garris and R.A. Wilkinson. Handwritten Segmented Characters Database. Technical Report Special Database 3, National Institute of Standards and Technology, February 1992.

[34] C.L. Wilson and M.D. Garris. Handprinted Character Database. Special Database 1, National Institute of Standards and Technology, April 1990.

[35] 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*, 1991.

[36] P.J. Grother. Cross validation comparison of NIST OCR databases. In D.P. D'Amato, editor, SPIE volume 1906, San Jose, CA, 1993.

[37] C.I. Watson and C.L. Wilson. Fingerprint database. Special Database 4, National Institute of Standards and Technology, April 1992.

[38] R.M. Stock and C.W. Swonger. Development and evaluation of a reader of fingerprint minutiae. Cornell Aeronautical Laboratory, Technical Report XM-2478-X-1:13–17, 1969.

[39] G.T. Candela and R. Chellappa. Comparative Performance of Classification Methods for Fingerprints. Technical Report NISTIR 5163, National Institute of Standards and Technology, April 1993.

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

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

[42] N.J. Nilsson. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, Chapter 3, page 45. McGraw-Hill, New York, 1965.

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

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

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

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

[47] 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.* To be published.

[48] 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.

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

[50] D.F. Specht. Probabilistic neural networks. *Neural Networks*, 3:109–118, 1990.

[51] J.J. Atick. Could information theory provide an ecological theory of sensory processing? *Networks*, 3:213–251, 1992.

[52] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[53] J.A. Hartigan. *Clustering Algorithms*. Wiley, New York, 1975.