

available at <URL:<http://www.cis.hut.fi/~jorma/papers/>>

## Comparison of Neural and Statistical Classifiers— Theory and Practice

Lasse Holmström  
Rolf Nevanlinna Institute

Petri Koistinen  
Rolf Nevanlinna Institute

Jorma Laaksonen  
Laboratory of Computer and Information Science,  
Helsinki University of Technology

Erkki Oja  
Laboratory of Computer and Information Science,  
Helsinki University of Technology

Research Reports A13  
January 1996

Rolf Nevanlinna Institute  
P.O. Box 4 (Yliopistonkatu 5)  
00014 University of Helsinki, Finland

ISBN 952-9528-32-9  
ISSN 0787-8338  
YLIOPISTOPAINO

## Abstract

Pattern classification using neural networks and statistical methods is discussed. We first give a tutorial overview that groups popular classifiers according to their underlying mathematical principles into several distinct categories. Starting from the Bayes classifier, one division is whether the classifier is explicitly estimating class conditional densities, or directly estimating the posterior probabilities by regression. Another criterion is the flexibility of the architecture in the sense of how rich the discriminant function family is. Still one dimension is neural vs. nonneural learning: neural learning is characterized by simple local computations in a number of real or virtual processing elements.

Based on these comparisons, a number of classification methods were selected for a case study that uses handwritten digit data. An effort was made to get fair estimates of their true classification performance, thus training set cross-validation was extensively used to design the various classifiers. The classification errors were estimated with an independent testing set.

The performance of a number of most typical neural and statistical classifiers was compared. Also, four methods of our own were used in the comparisons: the Reduced Kernel Discriminant Analysis (RKDA), the Learning  $k$ -Nearest Neighbor Classifier, the Averaged Learning Subspace Method (ALSM), and a modified version of Kernel Discriminant Analysis. Also, committee classifiers and classification with rejection were considered. In these experiments, the Local Linear Regression (LLR) method, although computationally prohibitively heavy, was the best classifier from the point of view of classification accuracy, with the Averaged Learning Subspace Method (ALSM) following close behind. For methods having both a learning and a non-learning version, error correcting learning seemed to give an advantage.

## Keywords

Classification methods, neural networks, comparison, handwritten digit recognition.

# 1 Introduction

The main application of artificial neural networks is pattern recognition. Understood in the sense of “artificial perception,” this involves deep problems in cognitive science and artificial intelligence. Realistic systems for computer vision, even for intrinsically two-dimensional problems, are hybrids of many methodologies like signal processing, classification, and relational matching. It seems that neural networks can be used to an advantage in certain subproblems, especially in feature extraction and classification [74]. These are also problems amenable to statistical techniques, because the data representations are real vectors of measurements or feature values, and it is possible to collect training samples on which regression analysis or density estimation become feasible. Thus, in many cases neural techniques and statistical techniques are seen as alternatives, or in fact neural networks are seen as a subset of statistics. This approach has led on one hand to a fruitful analysis of existing neural networks, on the other hand brought new viewpoints to current statistical methods, and sometimes produced a useful synthesis of the two fields. It must be emphasized, however, that outside some well-defined and focused application fields like multivariate classification, the two fields are different and the long term goals of neural networks in designing autonomous machine intelligence still remain.

In a narrow sense, the pattern recognition problem can be parceled to data acquisition and preprocessing, feature extraction, and classification. In addition to general references such as [17, 117, 14, 102, 25, 68, 84, 24] questions of pattern recognition are extensively covered in the literature of many specific application areas such as speech and image recognition.

The performance of statistical and neural network methods is usually discussed in connection with feature extraction and classification. Feature extraction involves the determination of those characteristics of the data that make subsequent reliable classification possible. Often one is able to take advantage of problem dependent features of the data but several general purpose approaches are available as well [14]. Efficient feature extraction is crucial for reliable classification and, if possible, these two subsystems should be matched optimally in the design of a complete pattern recognition system. In this article we concentrate on classification only. Further, only supervised classification using statistical and neural network methods is considered thus leaving out both unsupervised classification (clustering [35]) and syntactic pattern recognition [24]. In statistics literature supervised classification is often called discriminant analysis [68].

A popular application of pattern recognition is offline character recognition, where research during the last two decades has focused on handwritten characters [99, 65,

50, 71, 101, 100, 96, 30, 107, 1]. This is mainly due to the fact that while recognition of machine printed characters is considered a solved problem, the reliability achieved with handwritten text has not yet reached the level required in practical applications. Handwritten character recognition has become a popular application area for neural network classifiers [28, 32, 63], which through their adaptive capabilities have often been able to achieve better reliability than classical statistical or knowledge-based structural recognition methods.

Recently, many benchmark and comparison studies have been published on neural and statistical classifiers [81, 9, 49, 5]. One of the most extensive was the Statlog project in which statistical methods, machine learning and neural networks were compared [69]. As a general conclusion of that study, good statistical classifiers included the  $k$ -Nearest Neighbor ( $k$ -NN) rule, and good neural network classifiers included Learning Vector Quantization (LVQ) and Radial Basis Function (RBF) classifiers. One of the databases used in that study consisted of handwritten digits. The good performance of nearest neighbor classifiers for handwritten digits was also confirmed by Blue et al. [4], who however only compared the result to Radial Basis Function and Multi-Layer Perceptron (MLP) neural classifiers. A kernel discriminant analysis type method, the Probabilistic Neural Network, also did very well in that study.

In our case study, we also use handwritten digits and compare four methods of our own to a carefully selected subset of neural and statistical classifiers. These four methods are: Reduced Kernel Discriminant Analysis, Learning  $k$ -Nearest Neighbors classifier, the Averaged Learning Subspace Method, and a variant of kernel discriminant analysis that uses smoothing parameter selection based on error rate minimization. Further, Local Linear Regression has been rarely used previously in a classification context. Besides a different set of classifiers selected for comparison, the present study also differs from [4] in our systematic use of training set cross-validation in all classifier design. The final performance estimates are based on an independent testing set that had no role in classifier construction, including the choice of optimal pattern vector dimension.

In Section 2 we begin with an overview of classification methods that groups classifiers into a few main categories according to their underlying mathematical principles. The specific methods compared are described in Section 3. Some classifier design issues, including cross-validation, implementation of the reject option, and committee classifiers are discussed in Section 4. The performance of the various classifiers in handwritten digit recognition is discussed in Section 5 and the results of the paper are summarized in Section 6.

## 2 A Taxonomy of Classifiers

In the following overview we will assume that the classification problem involves patterns that are stochastically independent and identically distributed, although taking advantage of context dependent information in such applications as optical character recognition certainly might be beneficial.

To fix the notation, a  $d$ -dimensional feature or pattern vector is denoted by  $\mathbf{x} = [x_1, \dots, x_d]^T \in \mathbb{R}^d$ . In supervised classification each pattern is thought to originate from a specific class  $j \in \{1, \dots, c\}$ , where the number of possible classes  $c$  is known and fixed. A classifier can be regarded as a function  $g : \mathbb{R}^d \rightarrow \{1, \dots, c\}$  that classifies a given pattern  $\mathbf{x}$  to the class  $g(\mathbf{x})$ .

A pattern with its associated class is stochastically modeled as a pair  $(\mathbf{X}, J)$ , where  $\mathbf{X}$  is a random vector and  $J$  is a random variable with possible values  $1, \dots, c$ . The *a priori* probability of class  $j$  is  $P_j = P(J = j)$  and the probability density function of class  $j$  is denoted by  $f_j$ . The pooled data with all the classes combined then has the density function  $f = \sum_{j=1}^c P_j f_j$ . Sometimes the class-conditional densities  $f_j$  are supported by disjoint regions of the space  $\mathbb{R}^d$  and it is then possible to construct a classifier with zero classification error. More often the individual classes overlap and the smallest achievable error, the Bayes classification error, is positive. The Bayes classifier that minimizes the misclassification error [14] is defined by

$$(1) \quad g_{\text{BAYES}}(\mathbf{x}) = \operatorname{argmax}_{j=1, \dots, c} P_j f_j(\mathbf{x}).$$

Here “argmax” denotes the maximizing argument, i.e.,  $\mathbf{x}$  is classified to the class that maximizes the product  $P_j f_j(\mathbf{x})$ . An alternative way is to consider the *a posteriori* probability  $q_j(\mathbf{x}) = P(J = j \mid \mathbf{X} = \mathbf{x})$  of class  $j$  given  $\mathbf{x}$  and use the rule

$$(2) \quad g_{\text{BAYES}}(\mathbf{x}) = \operatorname{argmax}_{j=1, \dots, c} q_j(\mathbf{x}).$$

The rules (1) and (2) are equivalent since

$$(3) \quad q_j(\mathbf{x}) = P(J = j \mid \mathbf{X} = \mathbf{x}) = \frac{P_j f_j(\mathbf{x})}{f(\mathbf{x})}.$$

However, in practice the classifiers (1) and (2) have to be estimated from training data  $(\mathbf{x}_1, j_1), \dots, (\mathbf{x}_n, j_n)$  of pattern vectors with known classes and then two distinct approaches emerge. The use of rule (1) requires explicit estimation of the class-conditional probability density functions  $f_j$ . For (2), some regression technique can be used to estimate the posterior probabilities  $q_j$  directly without considering the class-conditional densities separately. We elaborate upon these two approaches further in Sections 2.1 and 2.2 below.

Other criteria than minimum classification error can be important in practice. Sometimes a class-dependent misclassification cost is needed. In certain types of applications it is essential to use Neyman-Pearson style classification [117, 46] where the classification error of one class is minimized given a fixed error for another class. The use of a *reject class* can help reduce the misclassification rate in tasks where exceptional handling (e.g. by a human expert) of particularly ambiguous cases is feasible. The patterns which cannot be assigned to any one class with reasonable certainty are put into the reject class for further processing. The decision to reject a pattern  $\mathbf{x}$  can be based on its probability of being misclassified. With the Bayes classification rule this probability is given by

$$(4) \quad e(\mathbf{x}) = 1 - \max_{j=1, \dots, c} q_j(\mathbf{x}).$$

The highest misclassification probability occurs when the posterior probabilities  $q_j(\mathbf{x})$  are equal and then  $e(\mathbf{x}) = 1 - 1/c$ . One can therefore select a rejection threshold  $0 \leq \theta \leq 1 - 1/c$  and reject  $\mathbf{x}$  if

$$(5) \quad e(\mathbf{x}) > \theta.$$

There is then a trade-off between the classification error and the size of the reject class: decreasing the threshold  $\theta$  makes the misclassification error for the patterns that we agree to classify smaller, but at the expense of a larger reject class that needs separate handling.

## 2.1 Density Estimation

In the density estimation approach one needs estimates for the prior probabilities  $P_j$  as well as the class-conditional densities  $f_j$  in (1). The prior probabilities may either be known or they can be estimated from the relative frequencies of the classes among the training data. The difficult part is to estimate the class-conditional densities. A classical approach is to model the class-conditional densities as multivariate normal distributions. The logarithm of  $P_j f_j(\mathbf{x})$  is then a quadratic function of  $\mathbf{x}$  and the name Quadratic Discriminant Analysis (QDA) or quadratic classifier is often used. In case of equal class covariance matrices a linear function results and the term Linear Discriminant Analysis (LDA) is used. A recent development is Regularized Discriminant Analysis (RDA) [21] which interpolates between LDA and QDA. A thorough account of these and other classifiers is provided by McLachlan [68].

Modeling the class-conditional densities as multivariate normals is an example of parametric density estimation, where the densities are assumed to belong to a

family of functions described by a finite set of parameters. In nonparametric density estimation no such fixed family of possible densities is assumed. Kernel or Parzen estimates as well as  $k$ -nearest neighbor methods (at least when  $k$  is large) are examples of popular nonparametric density estimation methods [90, 86, 108]. They give rise to such classification methods as Kernel Discriminant Analysis (KDA) [18, 33, 89] and  $k$ -Nearest Neighbor ( $k$ -NN) rules [18, 14]. The Probabilistic Neural Network (PNN) [95] is the neural network counterpart of KDA. In another approach the densities are estimated as finite mixtures of some standard probability densities by using the expectation-maximization (EM) algorithm or some other method [79, 103, 77, 78, 38]. Such an approach can be viewed as an economized KDA or an instance of the RBF approach [3]. The Self-organizing Reduced Kernel Density Estimator introduced in [45] estimates densities in the spirit of radial basis functions and we refer to the corresponding classification method as Reduced Kernel Discriminant Analysis (RKDA).

It should be pointed out that good density estimates are not always necessary for the design of an efficient classifier. In fact, only the optimal decision boundaries in  $\mathbb{R}^d$  between the classes need to be estimated accurately. It is therefore not uncommon that a quadratic or a linear classifier works well even though the underlying assumptions of normality do not hold. In Kernel Discriminant Analysis the smoothing parameter of the estimator (see Section 3) that gives best classification performance may not be optimal from the point of view of density estimation.

## 2.2 Regression

In the second approach to classification the class posterior probabilities  $P(J = j \mid \mathbf{X} = \mathbf{x})$  are directly estimated using some regression technique. Different methods correspond to the classes of functions and the types of fitting criteria used in the estimation procedure. In parametric regression one models the posterior probabilities using a family of functions described by a finite number of parameters. Examples are linear and logistic regression, as well as a Multi-Layer Perceptron (MLP) [40, 3] with a fixed architecture, i.e., fixed layer sizes. In nonparametric regression no fixed functional form is assumed. Examples of nonparametric methodologies are Projection Pursuit [23, 19], additive models [37], Multivariate Adaptive Regression Splines (MARS) [22], the Nadaraya–Watson kernel regression estimator [108, 56] and Local Linear Regression (LLR) [11, 12, 108]. The Nadaraya–Watson kernel regression estimator is also called the General Regression Neural Network [94]. Other neural network approaches include Multi-Layer Perceptrons with a flexible architecture and Radial Basis Function expansions [40, 3].

One way to fit a regression model is to use binary least squares regression. In a two class case we define the response  $y_i = 1$  or  $0$  depending on whether the  $i$ th training vector  $\mathbf{x}_i$  comes from class 1 or 2. The posterior probability is then estimated in an indirect way as follows. One first defines a family of functions  $\mathcal{R}$  that typically has the form  $\mathcal{R} = \{r(\cdot, \mathbf{t}) : \mathbf{t} \in T\}$ , where  $T$  is a finite-dimensional parameter space, and then selects a function  $r$  that minimizes the sum of squared errors criterion,

$$(6) \quad \frac{1}{n} \sum_{i=1}^n (y_i - r(\mathbf{x}_i))^2 = \min_{r \in \mathcal{R}}$$

The error (6) can be regarded as an estimate of the expected value  $E[(Y - r(\mathbf{X}))^2]$ . Ideally, this expectation is minimized by the conditional expectation function  $m(\mathbf{x}) = E(Y | \mathbf{X} = \mathbf{x})$  which in this case is equal to the posterior probability  $P(J = 1 | \mathbf{X} = \mathbf{x})$ . However, the conditional expectation  $m$  may or may not belong to the family  $\mathcal{R}$ , and besides, sampling variation will anyhow prevent us from estimating  $m$  exactly even when it does belong to  $\mathcal{R}$  [110, 80].

While the least squares fitting criterion can be motivated by the above reasoning, perhaps a more natural approach is to use logistic regression methodology [68, Ch. 8] and model the posterior probability directly as

$$(7) \quad P(J = 1 | \mathbf{X} = \mathbf{x}) = \frac{\exp(r(\mathbf{x}, \mathbf{t}))}{1 + \exp(r(\mathbf{x}, \mathbf{t}))}.$$

One can then estimate the unknown parameter  $\mathbf{t}$  by maximizing the conditional log-likelihood of  $y_1, \dots, y_n$  given that  $\mathbf{X}_1 = \mathbf{x}_1, \dots, \mathbf{X}_n = \mathbf{x}_n$ . When  $r(\mathbf{x}, \mathbf{t})$  is linear in its parameters, one can use the methodology developed for generalized linear models [66] to estimate the parameter vector. Note that the estimated posterior probability then takes values in the interval  $[0, 1]$  as a probability should. Note also that the least squares fitting criterion (6) can be thought to rise from using the maximum likelihood principle to estimate a regression model where errors are distributed normally, whereas the logistic approach uses binomially distributed error, clearly the statistically correct model.

These techniques are easily generalized to situations with more than two classes. Using one-of- $c$  coding, let the response  $\mathbf{y}_i$  be the unit vector  $[0, \dots, 0, 1, 0, \dots, 0]^T \in \mathbb{R}^c$  with 1 in the  $j$ th place. Then in the least squares approach one tries to minimize

$$(8) \quad \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^c (\mathbf{y}_i^{(j)} - \mathbf{r}^{(j)}(\mathbf{x}_i))^2 = \min_{\mathbf{r} \in \mathcal{R}}$$

over a family  $\mathcal{R}$  of  $\mathbb{R}^c$ -valued functions  $\mathbf{r}$ , where we denote the  $j$ th component of a vector  $\mathbf{z}$  by  $\mathbf{z}^{(j)}$ . One natural way of generalizing the logistic regression approach

is to model the posterior probabilities as the softmax [7] of the components of an  $\mathbb{R}^c$ -valued function  $\mathbf{r}$ ,

$$(9) \quad P(J = j \mid \mathbf{X} = \mathbf{x}) = q_j(\mathbf{x}) = \frac{\exp(\mathbf{r}^{(j)}(\mathbf{x}))}{\sum_{k=1}^c \exp(\mathbf{r}^{(k)}(\mathbf{x}))}.$$

Then a natural fitting criterion is to maximize the conditional log-likelihood

$$(10) \quad \sum_{i=1}^n \sum_{j=1}^c \mathbf{y}_i^{(j)} \log q_j(\mathbf{x}_i) = \sum_{i=1}^n \log q_{j_i}(\mathbf{x}_i) = \max_{\mathbf{r} \in \mathcal{R}}!$$

In the case of two classes this approach is equivalent to the use of the cross-entropy fitting criterion [3].

A very natural approach would be a regression technique that uses the number of misclassified patterns as the fitting criterion to be minimized. This provides for example one viable way of selecting the smoothing parameters in a classifier based on kernel regression. While such techniques have been proposed also in parametric regression settings [41] they are seldom used in practice. Classification and Regression Trees (CART) are an example of a nonparametric technique that estimates the posterior probabilities directly but uses neither the binary nor the logistic regression approach [6].

### 2.3 Other Classifiers

In addition to using the Bayes rule combined with either density estimation or regression, a third popular approach is to model the classes using prototypes and classify according to the shortest distance (defined in a suitable sense) to a prototype. Examples are Learning Vector Quantization (LVQ) [55] and the  $k$ -Nearest Neighbor classifiers with a small  $k$ . A distinct family of discrimination methods are the subspace classifiers [73] that model the individual classes using linear subspaces and classify a pattern according to its shortest distance to the model subspaces. Examples include the classical CLAFIC (CLAss-Featuring Information Compression) method [109] and the Averaged Learning Subspace Method (ALSM) that features incremental learning [73].

### 2.4 What are neural classifiers?

In the previous discussion we characterized some popular classification techniques in terms of the mathematical principles they are based on. In this general view many neural networks can be seen as representatives of certain larger families of statistical

techniques. However, this abstract point of view fails to identify some key features of neural networks that characterize them as a distinct methodology.

From the very beginning of neural network research [67, 82, 83] the goal was to demonstrate problem-solving without explicit programming. The neurons and networks were supposed to learn from examples and store this knowledge in a distributed way among the connection weights. The original methodology was exactly opposite to the goal-driven or top-down design of statistical classifiers in terms of explicit error functions. In neural networks, the approach has been bottom-up: starting from a very simple linear neuron that computes a weighted sum of its inputs, adding a saturating smooth nonlinearity, and constructing layers of similar parallel units, it turned out that “intelligent” behavior like speech synthesis [87] emerged by simple learning rules. The computational aspect has always been central. At least in principle, everything that the neural network does should be accomplished by a large number of simple local computations using the available input and output signals, like in real neurons, but unlike heavy numerical algorithms involving such operations as matrix inversions. Perhaps the best example of a clean-cut neural network classifier is the LeNet system [62, 5] for handwritten digit recognition. Such a computational model supports well the implementation in regular VLSI circuits.

In the current research on neural networks, these original views are clearly becoming vague as some of the simplest neural networks like the one hidden layer MLP or RBF networks have been shown to have very close connections to statistical techniques. The goal remains, however, of building much more complex artificial neural systems for demanding tasks like speech recognition [54] or computer vision [61], in which it is difficult or eventually impossible to state the exact optimization criteria for all the consequent processing stages.

Figure 1 is an attempt to assess the neural characteristics of some of the classification methods discussed earlier. For definitions of these classifiers, we refer to Section 3. The horizontal axis measures the flexibility of a classifier architecture in the sense of the richness of the discriminant function family encompassed by a particular method. High flexibility of architecture is a property often associated with neural networks. In some cases (MLP, RBF, CART, MARS) the flexibility can also include algorithmic model selection during learning.

In the vertical dimension, the various classifiers are categorized on the basis of how they are designed from a training sample. Training is considered nonneural if the training vectors are used as such in classification (e.g.  $k$ -NN, KDA), or if some statistics are first estimated in batch mode and the discriminant functions are computed from them (e.g. QDA, CLAFIC). Neural learning is characterized by simple local computations in a number of real or virtual processing elements. Neural learning

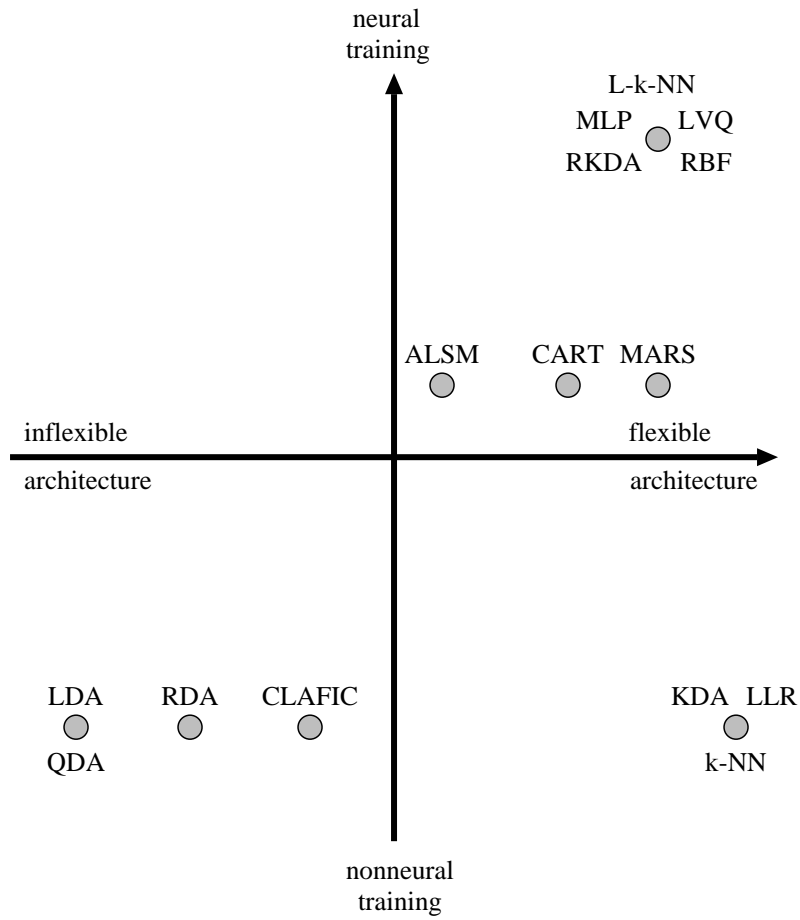


Figure 1: A schematic depiction of the neural characteristics of some classification methods.

algorithms are typically of the error correction type; for some such algorithms, not even an explicit cost function exists. Typically, the training set is used several times (epochs) in an on-line mode. Note, however, that for some neural networks (MLP, RBF) the current implementations in fact often employ sophisticated optimization techniques which would justify moving them downwards in our map to the lower half plane.

In this schematic representation the classical LDA and QDA methods are seen as least neural with the RDA and CLAFIC possessing at least some degree of flexibility in their architecture. The architecture of KDA,  $k$ -NN, and LLR is extremely flexible. Compared to CLAFIC, the ALSM method allows for both incremental learning and flexibility of architecture in the form of subspace dimensions that can change during learning. In this view, neural classifiers are well exemplified in particular by such methods as MLP, RBF, RKDA, LVQ, and Learning  $k$ -NN (L- $k$ -NN), but also to some degree by ALSM, CART, and MARS.

### 3 Methods Described

For our case study we chose a set of methods that includes typical representatives of the various categories discussed in Section 2. We will next give a short description of each method tested.

#### 3.1 QDA, LDA and RDA

Quadratic Discriminant Analysis (QDA) is based on the assumption that pattern vectors from class  $j$  are normally distributed with mean vector  $\boldsymbol{\mu}_j$  and covariance matrix  $\boldsymbol{\Sigma}_j$ . Following the density estimation approach then leads to the rule

$$(11) \quad g_{\text{QDA}}(\mathbf{x}) = \operatorname{argmax}_{j=1,\dots,c} \left[ \log \hat{P}_j - \frac{1}{2} \log \det \hat{\boldsymbol{\Sigma}}_j - \frac{1}{2} (\mathbf{x} - \hat{\boldsymbol{\mu}}_j)^T \hat{\boldsymbol{\Sigma}}_j^{-1} (\mathbf{x} - \hat{\boldsymbol{\mu}}_j) \right]$$

(after taking the logarithm of  $\hat{P}_j \hat{f}_j(\mathbf{x})$  and canceling a common term). Here  $\hat{\boldsymbol{\mu}}_j$  and  $\hat{\boldsymbol{\Sigma}}_j$  denote the natural estimates of the corresponding theoretical quantities, namely, the sample mean and the sample covariance of those pattern vectors in the training set which originate from class  $j$ , respectively.

If one assumes that the different classes are normally distributed with different mean vectors but with a common covariance matrix  $\boldsymbol{\Sigma}$ , then the previous formula simplifies to the Linear Discriminant Analysis (LDA) rule

$$(12) \quad g_{\text{LDA}}(\mathbf{x}) = \operatorname{argmax}_{j=1,\dots,c} \left[ \log \hat{P}_j + \hat{\boldsymbol{\mu}}_j^T \hat{\boldsymbol{\Sigma}}^{-1} (\mathbf{x} - \frac{1}{2} \hat{\boldsymbol{\mu}}_j) \right],$$

where a natural estimate for  $\Sigma$  is the pooled covariance matrix estimate

$$(13) \quad \widehat{\Sigma} = \sum_{j=1}^c \widehat{P}_j \widehat{\Sigma}_j.$$

Friedman’s Regularized Discriminant Analysis (RDA) [21] is a compromise between LDA and QDA. The decision rule is otherwise the same as (11) but instead of  $\widehat{\Sigma}_j$  one uses regularized covariance estimates  $\widehat{\Sigma}_j(\lambda, \gamma)$  with two regularizing parameters. Parameter  $\lambda$  controls the shrinkage of the class conditional covariance estimates toward the pooled estimate and  $\gamma$  controls the shrinkage toward a multiple of the identity matrix, see [21] for the exact formulas. One can obtain the QDA and LDA with certain choices of the regularizing parameters.

### 3.2 KDA

In the Kernel Discriminant Analysis (KDA) approach one forms kernel estimates  $\hat{f}_j$  of the class conditional densities and then applies rule (1). To define the kernel estimates, let  $\mathbf{x}_{ij}, i = 1, \dots, n_j$ , be the training data from class  $j$ . Then the estimate of the  $j$ th class-conditional density is

$$(14) \quad \hat{f}_j(\mathbf{x}) = \frac{1}{n_j} \sum_{i=1}^{n_j} K_{h_j}(\mathbf{x} - \mathbf{x}_{ij}), \quad j = 1, \dots, c,$$

where  $K$  is a fixed probability density function, called the kernel,  $h_j > 0$  is the smoothing parameter of class  $j$ , and  $K_h$  denotes the scaled kernel

$$(15) \quad K_h(\mathbf{x}) = h^{-d} K(\mathbf{x}/h), \quad \mathbf{x} \in \mathbb{R}^d.$$

This scaling ensures that  $K_h$  and hence also each  $\hat{f}_j$  is a probability density. A popular choice which was also used in the present study is the Gaussian kernel

$$(16) \quad K(\mathbf{x}) = (2\pi)^{-d/2} \exp(-\|\mathbf{x}\|^2/2).$$

The choice of suitable values for the smoothing parameters is crucial in kernel density estimation and KDA. Several approaches have been proposed in the literature, see [90, 86, 68, 108]. We used two methods based on cross-validated error count. In the first method, KDA1, we restrict all the smoothing parameters  $h_j$  to be equal to a parameter  $h$ , and evaluate the number of misclassified vectors using cross-validation (see Section 4.1). We then vary  $h$  and select that value which minimizes the cross-validated error count. In the second method, KDA2, we let the smoothing parameters vary separately and again seek to minimize the cross-validated error count.

Both of the methods lead to optimization problems where the object function is piecewise constant. In the first method the search space is one-dimensional, and the optimization problem can be solved simply by evaluating the object function on a suitable grid of  $h$ -values. In the second method the nonsmoothness of the object function causes more trouble. Instead of minimizing the error count directly, we found it advantageous to minimize a smoothed version of it. The particular smoothing described below is inspired by that used in [105]. Let  $\hat{q}_j(\mathbf{x}, h_1, \dots, h_c)$  be the posterior probability estimate at  $\mathbf{x}$  corresponding to the current smoothing parameters and define the  $c$ -component vector function  $\mathbf{u}$  by

$$(17) \quad \mathbf{u}^{(j)}(\mathbf{x}) = \exp(\gamma \hat{q}_j(\mathbf{x}, h_1, \dots, h_c)) / \sum_{k=1}^c \exp(\gamma \hat{q}_k(\mathbf{x}, h_1, \dots, h_c)),$$

where  $\gamma > 0$  is a parameter. Then the smoothed error count is given by

$$(18) \quad n - \sum_{i=1}^n \mathbf{u}^{(j_i)}(\mathbf{x}_i).$$

As  $\gamma \rightarrow \infty$ , this converges towards the true error count. In the experiments we fixed  $\gamma = 100$ . Since the smoothed error count is a differentiable function of the smoothing parameters, one can use a gradient-based minimization method for the optimization. As an initial point for the iteration, we used the vector  $[h, \dots, h]^T$ , where  $h$  is the value selected in KDA1.

### 3.3 RKDA

The standard kernel density estimate suffers from the curse of dimensionality: as the dimension  $d$  of data increases, the size of a sample  $\mathbf{x}_1, \dots, \mathbf{x}_n$  required for an accurate estimate of an unknown density  $f$  grows quickly. On the other hand, even if there are enough data for accurate density estimation, the application at hand may limit the complexity of the classifier one can use in practice. A kernel estimate with a large number of terms may be computationally too expensive to use. One solution is to *reduce* the estimate, that is, to use fewer kernels but to place them at optimal locations. One can also introduce kernel dependent weights and smoothing parameters. Various reduction approaches were described in [26, 27, 31, 77, 78, 103, 92, 112]. In some cases the methods suggested have a close connection with radial basis function and mixture density estimation techniques.

The Self-organizing Reduced Kernel Density Estimate [45] has the form

$$(19) \quad \hat{f}(\mathbf{x}) = \sum_{k=1}^{\ell} w_k K_{h_k}(\mathbf{x} - \mathbf{m}_k),$$

where  $\mathbf{m}_1, \dots, \mathbf{m}_\ell$  are the reference vectors of a Self-Organizing Map [55],  $w_1, \dots, w_\ell$  are nonnegative weights with  $\sum_{k=1}^{\ell} w_k = 1$ , and  $h_k$  is a smoothing parameter associated with the  $k$ th kernel. In order to achieve substantial reduction one takes  $\ell \ll n$ . The kernel locations  $\mathbf{m}_k$  are obtained by training the Self-Organizing Map using the whole available sample  $\mathbf{x}_1, \dots, \mathbf{x}_n$  from  $f$ . The weights  $w_k$  are computed iteratively and they reflect the number of training data in the Voronoi regions of the corresponding reference vectors. The smoothing parameters are optimized via stochastic gradient descent that attempts to minimize a Monte Carlo estimate of the integrated squared error  $\int (\hat{f} - f)^2$ . Simulations have shown that when the underlying density  $f$  is multimodal, the use of the feature map algorithm gives better density estimates than  $k$ -means clustering, the approach proposed in [64]. Reduced Kernel Discriminant Analysis (RKDA) constitutes using estimates (19) for the class-conditional densities in the classifier (1). A drawback of RKDA in the present application is that the smoothing parameters of the class-conditional density estimates used in the approximate Bayes classifier are optimized from the point of view of integrated squared error and not discrimination performance which is the true focus of interest.

### 3.4 MLP

We used a standard Multi-Layer Perceptron with  $d$  inputs,  $\ell$  hidden units and  $c$  output units and with the logistic activation function in the hidden and output units [40, 3]. Such a network has  $(d + 1)\ell + (\ell + 1)c$  adaptable weights which in our experiments were determined by minimizing the sum of squared errors criterion using a conjugate gradient method [88]. Using the notation of Section 2.2, we used the vector  $0.1 + 0.8\mathbf{y}_i$  as the desired output for input  $\mathbf{x}_i$ , i.e., the vectors  $\mathbf{y}_i$  were scaled to better fit within the range of the logistic function. Then the scaled outputs  $1.25(\mathbf{r}^{(j)}(\mathbf{x}) - 0.1)$  of the optimized network can be regarded as estimating the posterior probabilities  $P(J = j \mid \mathbf{X} = \mathbf{x})$ . We employed the heuristic of starting the local optimizations from many random initial points and keeping the weights yielding the minimum value for the sum of squared errors to prevent the network from converging to a shallow local minimum.

### 3.5 LLR

Local Linear Regression (LLR) (or more generally local polynomial regression) is a nonparametric regression method which has its roots in classical methods proposed for the smoothing of time series data, see [12]. Such estimators have received more attention recently, see e.g. [36]. The particular version described below is also called

LOESS [11, 12].

Local Linear Regression models the regression function in the neighborhood of each point  $\mathbf{x}$  by means of a linear function  $\mathbf{z} \mapsto \mathbf{b}_0 + \mathbf{B}(\mathbf{z} - \mathbf{x})$ . Given training data  $(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)$ , the fit at point  $\mathbf{x}$  is calculated as follows. First one solves the weighted linear least squares problem

$$(20) \quad \sum_{i=1}^n \|\mathbf{y}_i - \mathbf{b}_0 - \mathbf{B}(\mathbf{x}_i - \mathbf{x})\|^2 w((\mathbf{x}_i - \mathbf{x})/h(\mathbf{x})) = \min_{\mathbf{b}_0, \mathbf{B}}$$

and then the fit at  $\mathbf{x}$  is given by the vector  $\mathbf{b}_0$ . As the function  $w$  we used the tricube weight function [11]

$$(21) \quad w(u) = \max((1 - |u|^3)^3, 0).$$

The local bandwidth  $h(\mathbf{x})$  is controlled by a neighborhood size parameter  $0 < \alpha \leq 1$ : one takes  $k$  equal to  $\alpha n$  rounded to the nearest integer and then takes  $h(\mathbf{x})$  equal to the distance to the  $k$ th closest neighbor of  $\mathbf{x}$  among the vectors  $\mathbf{x}_1, \dots, \mathbf{x}_n$ . If the components of  $\mathbf{x}$  are measured in different scales, then it is advisable to select the metric for the nearest neighbor calculation carefully. However, in our application we simply used the Euclidean metric. At a given  $\mathbf{x}$ , the weighted linear least squares problem can be reduced to inverting a  $(d + 1) \times (d + 1)$  matrix, where  $d$  is the dimensionality of  $\mathbf{x}$ , see e.g. [108, Ch. 5].

It would also be possible to use a bandwidth  $h$  which does not depend on  $\mathbf{x}$  and the weight function could be chosen differently. This would lead to local linear (or local polynomial) kernel regression [108, Ch. 5]. There is an interesting connection between local constant kernel regression and KDA. First of all, the local constant regression with kernel weights is identical with the Nadaraya-Watson kernel regression estimator. When the responses  $\mathbf{y}_i$  are formed as explained in Section 2.2 and the prior probabilities are equal, then the local constant regression estimator gives the same posterior probability estimates and hence the same classifier as the KDA provided the bandwidths  $h$  are the same and that the local constant regression estimator uses the kernel  $K$  of the KDA classifier as its weight function.

### 3.6 Tree classifier, MARS and FDA

The introduction of tree based models in statistics dates back to [70] although their current popularity is largely due to the seminal book [6]. For Euclidean pattern vectors  $\mathbf{x} = [x_1, \dots, x_d]^T$ , a classification tree is a binary tree where at each node the decision to branch either to left or right is based on a test of the form  $x_i \geq \lambda$ . The cut-off values  $\lambda$  are chosen to optimize a suitable fitting criterion. The tree

growing algorithm recursively splits the pattern space  $\mathbb{R}^d$  into hyper rectangles while trying to form maximally pure nodes, that is, subdivision rectangles that ideally contain training vectors from one class only. Stopping criteria are used to keep the trees reasonably sized although the commonly employed strategy is to first grow a large tree that overfits the data and then use a separate pruning stage to improve its generalization performance. A terminal node is labeled according to the class with the largest number of training vectors in the associated hyper rectangle. The tree classifier therefore uses the Bayes rule with the class posterior probabilities estimated by locally constant functions.

The particular tree classifier used here is available as part of the S-Plus statistical software package [2, 8, 106]. This implementation uses a likelihood function to select the optimal splits [10]. Pruning was performed by the minimal cost-complexity method. The cost of a subtree  $T$  is taken to be

$$(22) \quad R_\alpha(T) = R(T) + \alpha \cdot \text{size}(T),$$

where  $R(T)$  is an estimate of the classification error of  $T$ , size of  $T$  is measured by the number of its terminal nodes, and  $\alpha > 0$  is a cost parameter. An overfitted tree is pruned by giving  $\alpha$  increasingly large values and selecting nested subtrees that minimize  $R_\alpha(T)$ .

MARS [22] is a regression method that shares features with tree based modeling. MARS estimates an unknown function  $r$  using an expansion

$$(23) \quad \hat{r}(\mathbf{x}) = a_0 + \sum_{k=1}^M a_k B_k(\mathbf{x}),$$

where the functions  $B_k$  are multivariate splines. The algorithm is a two stage procedure, beginning with a forward stepwise phase which adds basis functions to the model in a deliberate attempt to overfit the data. The second stage of the algorithm is standard linear regression backward subset selection. The maximum order of variable interactions (products of variables) allowed in the functions  $B_k$ , as well the maximum value of  $M$  allowed in the forward stage, are parameters that need to be tuned experimentally. Backward model selection uses the generalized cross-validation criterion (GCV) introduced in [13].

$$\text{GCV}(M) = \frac{(1/n) \sum_{i=1}^n (y_i - \hat{r}(\mathbf{x}_i))^2}{[1 - C(M)/n]^2}.$$

Here the numerator is the lack-of-fit based on the training data and the denominator imposes a penalty for increasing the model complexity,  $C(M)$ . This complexity function is taken to be  $C(M) = M \cdot (\alpha/2 + 1)$ , where  $M$  is the number of basis functions

in the proposed model and  $\alpha$  represents an additional cost for fitting the parameters associated with the included additional basis functions.

The original MARS algorithm fits only scalar valued functions and it is therefore not well-suited to discrimination tasks with more than two classes. A recent proposal called Flexible Discriminant Analysis (FDA) [39] with its publicly available S-Plus implementation in the StatLib program library contains vector valued MARS as one of its ingredients. However, FDA is not limited to just MARS as it allows the use of other regression techniques as its building blocks as well. In FDA one can first train  $c$  separate MARS models  $\mathbf{r}^{(j)}$  with equal basis function sets but different coefficients  $a_k$  to map training vectors  $\mathbf{x}_i$  to the corresponding unit vectors  $\mathbf{y}_i$  (cf. Section 2.2). Then a linear map  $A$  is constructed to map the regression function output space  $\mathbb{R}^c$  onto a lower dimensional feature space  $\mathbb{R}^\ell$  in a manner that optimally facilitates prototype classification based on the transformed class means  $A(\mathbf{r}(\hat{\boldsymbol{\mu}}_j))$  and a weighted Euclidean distance function.

### 3.7 $k$ -NN and Learning $k$ -NN Classifiers

In a  $k$ -NN classifier [14] each class is represented by a set of prototype vectors. The  $k$  closest neighbors of a pattern vector are found from among all the prototypes and the class label is decided by the majority rule. A possible tie of two or more classes is broken by decreasing  $k$  by one and re-voting.

Recently, two of the authors have introduced a set of adaptation rules that can be used in iterative training of a  $k$ -NN classifier [58]. The learning rules of the proposed Learning  $k$ -NN resemble those of LVQ but at the same time the classifier utilizes the improved classification accuracy provided by majority voting. The performance of the standard  $k$ -NN classifier depends on the quality and size of the training set and the performance of the classifier decreases if the available computing resources limit the number of training vectors one can use. In such a case the Learning  $k$ -NN rule is better able to utilize the available data by using the whole size  $n$  training set to optimize the smaller set of  $\ell < n$  prototype vectors.

### 3.8 Variations of LVQ

The Learning Vector Quantization (LVQ) algorithm [55] produces a set of prototype or codebook pattern vectors that can be used in a 1-NN classifier. Training consists of moving a fixed number  $\ell$  of codebook vectors iteratively toward or away from the training samples  $\mathbf{x}_i$ . The variations of the LVQ algorithm—including LVQ1, OLVQ1, LVQ2, LVQ2.1, LVQ3, and LVQ4—differ in the way the codebook vectors

are updated. The LVQ learning process can be interpreted either as an iterative movement of the decision boundaries between neighboring classes, or as a way to generate a set of codebook vectors whose density reflects the shape of the function  $s$  defined as

$$(24) \quad s(\mathbf{x}) = P_j f_j(\mathbf{x}) - \max_{k \neq j} P_k f_k(\mathbf{x}),$$

where  $j = g_{\text{BAYES}}(\mathbf{x})$ . Note that the zero set of  $s$  consists of the Bayes optimal decision boundaries.

### 3.9 CLAFIC and ALSM

The motivation for the subspace classifiers originates from compression and optimal reconstruction of multidimensional data. The use of linear subspaces as class models is based on the assumption that the data within each class approximately lies on a lower-dimensional subspace of the pattern space  $\mathbb{R}^d$ . A vector from an unknown class can then be classified according to its shortest distance from the class subspaces.

The sample mean  $\hat{\boldsymbol{\mu}}$  of the whole training set is first subtracted from the pattern vectors. For each class  $j$  the correlation matrix  $\widehat{\mathbf{R}}_j$  is estimated and its first few eigenvectors  $\mathbf{u}_{1j}, \dots, \mathbf{u}_{\ell_j j}$  are used as columns of a basis matrix  $\mathbf{U}_j$ . The classification rule of the CLAFIC [109] algorithm can then be expressed as

$$(25) \quad g_{\text{CLAFIC}}(\mathbf{x}) = \operatorname{argmax}_{j=1, \dots, c} \|\mathbf{U}_j^T \mathbf{x}\|^2.$$

The Averaged Learning Subspace Method (ALSM) introduced by one of the authors [73] is an iterative learning version of CLAFIC in which the unnormalized sample class correlation matrices  $\widehat{\mathbf{S}}_j = \sum_{i=1}^{n_j} \mathbf{x}_{ij} \mathbf{x}_{ij}^T$  are slightly modified according to the correctness of the classifications,

$$(26) \quad \widehat{\mathbf{S}}_j(k+1) = \widehat{\mathbf{S}}_j(k) + \alpha \sum_{i \in A_j} \mathbf{x}_i \mathbf{x}_i^T - \beta \sum_{i \in B_j} \mathbf{x}_i \mathbf{x}_i^T.$$

Here  $\mathbf{x}_{1j}, \dots, \mathbf{x}_{n_j j}$  is the training sample from class  $j$ ,  $\alpha$  and  $\beta$  are small positive constants,  $A_j$  is the set of indices  $i$  for which  $\mathbf{x}_i$  comes from class  $j$  but is classified erroneously to a different class and  $B_j$  consists of those indices for which  $\mathbf{x}_i$  is classified to  $j$  although it actually originates from a different class. The basis matrices  $\mathbf{U}_j$  are recalculated after each training epoch from the dominant eigenvectors of the modified  $\widehat{\mathbf{S}}_j$ . In our experiments we chose equal numbers  $\ell_1 = \dots = \ell_c = D$  of basis vectors both in CLAFIC and in ALSM, and for ALSM,  $\alpha$  was chosen to be equal to  $\beta$ .

## 4 Classifier Design: Some General Issues

### 4.1 Cross-Validation

In order to get reliable estimates of classifier performance, the available data should first be divided into two separate parts: the training sample and the testing sample. The whole process of classifier design should then be based strictly on the training sample only. In addition to parameter estimation, the design of some classifiers involves the choice of various tuning parameters and model or architecture selection. To utilize the training sample efficiently, cross-validation [98] (or “rotation”, cf. [14, Ch. 10.6.4]) can be used. In  $v$ -fold cross-validation the training sample is first divided into  $v$  disjoint subsets. One subset at a time is then put aside, a classifier is designed based on the union of the remaining  $v - 1$  subsets and then tested for the subset left out. Cross-validation approximates the design of a classifier using all the training data and then testing it on an independent set of data, which enables defining a reasonable object function to be optimized in classifier design. E.g., for a fixed classifier, the dimension of the pattern vector can be selected so that it minimizes the cross-validated error count. After optimization, one can obtain an unbiased estimate of the performance of the optimized classifier by means of the separate testing sample. Notice that the performance estimates might become biased if the testing sample were in any way used during the training of the classifier.

### 4.2 Implementation of the Reject Option

Except for the prototype classifiers  $k$ -NN and Learning  $k$ -NN, LVQ, and the subspace classifiers, all other methods provide for a natural way to implement the reject option as their decision rules can be interpreted in terms of the class posterior probabilities. For small  $k$ , neither the Learning  $k$ -NN nor the standard  $k$ -NN classifier can be convincingly interpreted as a density estimation method but the very definitions of these classifiers allow for a simple rejection rule. For  $k = 3$ , say, we can reject unless all the three nearest prototypes are from the same class. Second, we may reject only those patterns whose nearest prototypes belong to three different classes. Third, we may decide never to reject. We are therefore able to select between three points on the rejection-reliability tradeoff curve. For subspace classifiers, relative projection lengths on the class subspaces can be used to guide rejection.

### 4.3 Committee Classifiers

In practice one is usually able to classify a pattern using more than one classifier. It is then quite possible that combining the opinions of several parallel methods results in improved classification performance. Such hybrid classifiers, classifier ensembles, or *committees*, have been studied intensively in recent years [75].

Besides improved classification performance, there are other reasons to use a committee classifier. The pattern vectors may be composed of components that originate from very diverse domains. E.g. in digit classification, some features may be statistical quantities like moments, while others are discrete structural descriptors like numbers of endpoints, loops, etc. There may not be an obvious way to concatenate the various components into a single pattern vector suitable for any single classifier type. In some situations the computational burden can be reduced either during training or in the recognition phase if the classification is performed in several stages.

Various methods exist for forming a committee of classifiers even when their output information is of different types. In the simplest case a classifier only outputs its decision about the class of an input pattern but sometimes also some measure of the certainty of the decision is provided. The classifier may also propose a set of classes in the order of decreasing certainty or a measure of decision certainty may be associated with all possible classes. Various ways to combine classifiers with such different types of output information are analyzed in [116, 43, 44, 47].

The simplest decision rule is to use a majority rule among the classifiers in the committee, possibly ignoring the opinion of some of the classifiers [115]. Two or more classifiers using different sets of features may be combined to implement the rejection of ambiguous patterns [72, 53, 101, 59]. A genetic algorithm can search for optimal weights to combine the classifier outputs [60]. Theoretically more advanced methods may be derived from the EM-algorithm [43, 44, 113, 114, 52] or from the Dempster-Shafer theory of evidence [116, 20].

The outputs of several classifiers may be combined linearly [51] or nonlinearly [104] to reduce the variance of the posterior probability estimates. A more general case is variance reduction in continuous function estimation. A set of MLPs can be combined into a committee classifier with reduced output variance and thus smaller classification error [76, 34, 57, 111]. A separate confidence function may also be incorporated in each of the MLPs [91].

Given a fixed feature extraction method one can either use a common training set to design a number of different types of classifiers [48] or alternatively use different training sets to design several versions of one type of classifier [16, 15, 42, 85, 93].

## 5 Case Study: Classification of Handwritten Digits

### 5.1 Data and its Preprocessing

We tested the performance of the classifiers defined previously on a realistic pattern recognition problem, the classification of handwritten digits. To obtain the test material, we used a set of forms. They were filled out by randomly chosen Finnish people and the data were collected for experimental purposes only. The total number of forms was 894 and each contained two handwritten examples of each of the ten digits. The images were scanned in binary form with the resolution of 300 pixels per inch in both directions. In this resolution each digit was printed approximately in a  $75 \times 100$  pixel rectangle.

The bounding boxes of the digits were normalized to  $32 \times 32$  pixels and the slant of the images was removed. No normalization of the line width was performed. The preprocessing procedure closely resembles that of [29], [4], and it produced a sample of 17880 binary vectors, each 1024-dimensional. Some examples are displayed in the leftmost column of Figure 2. The sample was then divided into two sets of equal size, one for training and one for testing the classifiers.

### 5.2 Feature Selection

As pattern vectors were used Karhunen-Loève (KL) transforms [25] of the original preprocessed images. The KL transform was derived from the estimated covariance matrix of the training set. This approach is similar to that adopted in [29], [4], and it is based on the assumption that the essential information needed in classification is concentrated along the directions which correspond to the largest variations in the data. In practice, it is possible that the classification accuracy in fact improves when the dimensionality of the data is reduced. This happens if the larger estimation errors in classifier design overshadow the potential extra discrimination information provided by higher dimensional pattern vectors.

The magnitude of the covariance matrix eigenvalues can be used to measure the usefulness of the corresponding KL transform components in classification. However, for the digit data, the eigenvalues decreased rather steadily in magnitude and it was not possible to decide on any natural reduced pattern vector dimension. On the other hand, higher dimensions imply more computations both in classifier design and application. In our tests we used a maximum pattern dimension of 64. As explained in Section 4.1, cross-validation can then be used to select the best dimension  $d \leq 64$  for

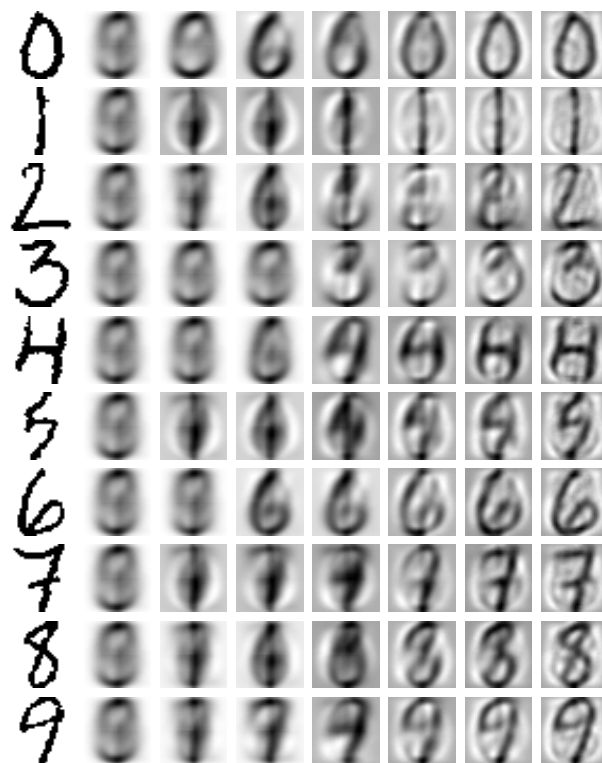


Figure 2: In the leftmost column, some normalized  $32 \times 32$  digit images. In the second column, mean of the training set and in the remaining columns, reproductions by adding projections in the principal directions up to dimension 1, 2, 5, 16, 32, and 64.

each classifier separately. The columns of Figure 2 show how the normalized images can be reproduced from their KL transforms with progressively greater accuracy as the transform dimension is increased.

### 5.3 Performance Comparisons and Discussion

In our experiments, the selection of classifier architectures as well as the choice of the values for different regularizing and smoothing parameters was based on training set cross-validation. For some methods, such as KDA, even  $n$ -fold cross-validation where  $n$  is the size of the training set could readily be used whereas for methods such as MLP this would be impractical as the design of the classifier involves a rather time consuming training stage. Besides, in our application such leave-one-out cross-validation seemed to deliver optimistically biased error rates which probably is due to the fact that in the training sample there were several exemplars of each digit from each writer. As a reasonable compromise, 10-fold cross-validation was used for most methods, with the exception of RKDA and MLP where 2-fold and 4-fold cross-validation was used, respectively.

Choosing an optimal pattern vector dimension is part of architecture or model selection and for most classifiers this was done using cross-validation. Cross-validation was also employed to determine the regularization parameters of RDA, the smoothing parameters of KDA and LLR, the number of reference vectors in RKDA, the size of the hidden layer of the MLP classifier, the codebook sizes for the Learning  $k$ -NN method and LVQ, the subspace dimension  $D$  for CLAFIC and ALSM, as well as the learning coefficients of ALSM. During cross-validation of the MLP classifier, each optimization was started from two random initial points with the values for the adaptable weights drawn independently from the standard normal distribution. Thus eight optimization runs were needed for each  $(d, \ell)$ -pair tried. The number of training epochs of Learning  $k$ -NN, LVQ, and ALSM was also determined by cross-validation. The tree classifier can use cross-validation to evaluate the error  $R(T)$  of (22) when pruning the overfitted model. Depending on the computational costs associated with the various methods, parameter search grids of different coarseness were used. However, as explained in Section 3.2, in KDA2 we used multivariate optimization instead of a grid search.

Figure 3 displays an example of architecture selection using cross-validation, in this case for the CLAFIC subspace classifier. The subspace dimension is selected according to the smallest cross-validated error giving the optimum  $D = 29$ . Note how the cross-validated error follows closely the testing error while the training error is optimistically biased and also suggests a somewhat higher optimal subspace dimension.

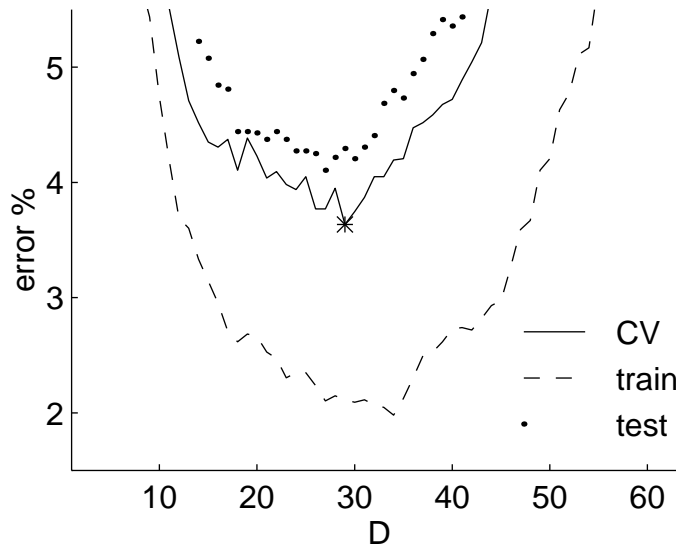


Figure 3: Choosing the optimal CLAFIC subspace dimension using cross-validation. Shown are the cross-validated (CV) error, the training set error, and the testing set error. The optimal dimension 29 is marked with a star.

The MLP, LVQ, Learning  $k$ -NN and RKDA algorithms need initialization that makes their test set performance depend either on a set of randomly generated real numbers or the particular subset of training data used for initialization. To account for this, each of these methods was evaluated by computing the average performance and its standard deviation in ten independent trials. During testing, the MLP optimizations were started from four random initial points with the values for the weights drawn independently from the standard normal distribution. The MLP having the smallest sum of squared errors on the training set was then tested on the testing set and this procedure was repeated ten times to obtain the results.

Table 1 shows the testing set classification errors for all the classifiers considered, when no rejection was allowed. The optimal cross-validated parameters are also shown for reference. To assess the statistical accuracy of the results, one can obtain a nominal estimate for the standard deviation of the error percentage by using the binomial distribution, cf. [14, Ch. 10.3]. In the range of 2.5–4.0%, where the results for our best classifiers lie, this gives a standard deviation of approximately 0.2 percentage points. Then, e.g., a nominal 95% confidence interval for the error rate of the LLR classifier would be given by  $2.8 \pm 0.4\%$ . Notice, however, that this confidence interval is nominal since it has been calculated under the assumption that the testing sample is independent of the training sample and consists of independent and identically

Classifier	error %	cross-validated parameters
LDA	9.8	$d = 64$
QDA	3.7	$d = 47$
RDA	3.4	$d = 61, \gamma = 0.25, \lambda = 0$
KDA1	3.7	$d = 32, h = 3.0$
KDA2	3.5	$d = 36, h_1, \dots, h_{10}$
RKDA	5.2 (0.1)	$d = 32, \ell = 35$
MLP	7.0 (0.4)	$d = 36, \ell = 35$
LLR	2.8	$d = 36, \alpha = 0.1$
Tree classifier	16.8	see text
FDA/MARS	6.3	see text
3-NN	3.8	$d = 38$
L-3-NN	3.6 (0.1)	$[d = 38, \alpha = 0.1], \ell = 5750, \#epochs = 7$
LVQ	4.0 (0.1)	see text
CLAFIC	4.3	$[d = 64], D = 29$
ALSM	3.1	$[d = 64, D = 29], \alpha = \beta = 3.1, \#epochs = 9$
Committee	2.5	[LLR,ALSM,L-3-NN]

Table 1: Comparison of classification performance. Shown are testing set classification errors and, in parentheses, estimated standard deviation in ten independent trials for certain classifiers. Where applicable, cross-validated parameters are given. Those parameters given within square brackets were determined without direct cross-validation, e.g., taken from the classifier on the previous line.

distributed realizations of the pair  $(\mathbf{X}, J)$ .

In the following, we first comment on the performance of each classifier and classifier group in turn, and then make some overall conclusions. Of the normality based classifiers, LDA was clearly inferior to QDA at each tested dimension. RDA selected at each dimension the parameter  $\lambda$  as zero so that no shrinkage of the class conditional covariance matrices towards the pooled covariance matrix took place. However, at the selected optimal dimension the parameter  $\gamma$  controlling the shrinkage towards a multiple of the identity matrix was selected nonzero so that shrinkage did happen, and this turned out to be beneficial.

Of the two Kernel Discriminant Analysis methods, KDA2 seems to perform slightly better than KDA1. However, a kernel classifier based on a sample of this large size is slow to evaluate and might be too slow to use in practice. The classification error of Reduced Kernel Discriminant Analysis (RKDA) is higher than that of KDA but as a trade-off there is a dramatic drop in classifier complexity: where KDA uses

894 kernels per class, RKDA needs only 35. Thus, after RKDA has been trained, it is much less costly to use in actual digit classification. At the expense of increasing the complexity and required training time, the discrimination performance of RKDA can be further improved by using more kernels.

The results obtained with the MLP classifier are not particularly good, which is probably due to overfitting. The network selected by cross-validation has 1655 adaptable parameters which have to be determined using a training sample of size 8940. It seems difficult to select a network architecture which is flexible enough for the task and at the same time is parameterized parsimoniously enough so that the parameters can be reliably determined from the training data. Some kind of regularization such as weight decay [40, 3] might help in reducing the overfitting.

Excluding the committee classifier, the best results were obtained with Local Linear Regression (LLR). Unfortunately, this method is impractically slow since the weighted linear least squares problem has to be solved separately at each pattern to be classified. However, a classifier operating on roughly the same principles should achieve good performance. A particularly promising candidate in this respect would be the local linear radial basis function method used in [97].

The performance of the tree classifier was disappointing. The natural approach is to use full 64-component pattern vectors and to let the algorithm itself decide how many variables (pattern components)  $x_i$  an optimal tree needs. The tree obtained with 64-component pattern vectors and the default tree growing stopping parameter values had 54 terminal nodes and it actually utilized 15 out of the 64 possible variables. The training error was 20.5% while the testing error was 23.6%. The complexity cost (22) decreased monotonically with increasing tree size and it was not possible to find a natural optimal pruning size. The trees obtained with lower dimensional training data were very similar. On the basis of training error, the best pattern vector dimension appeared to be 16. The default parameters were then adjusted to allow for larger trees but due to the heavy memory requirements it became quickly impossible to use cross-validated pruning. It was feasible to grow trees with about 150 terminal nodes and prune them but the testing errors remained over 20%. The best results were obtained by growing huge trees with no training error and using them as such. The tree grown with 16-dimensional data had 849 terminal nodes and testing set error 16.8% which is the result reported in Table 1.

The natural approach for FDA/MARS is also to use the full 64-component pattern vectors and to let the algorithm pick the optimal model architecture. The default component models  $\mathbf{r}^{(j)}$  are of additive type where products of variables are not allowed in the expansion (23). The algorithm selected models with 116 terms and the testing error was 8.5%. Experimenting with different designs suggested that the training error

could be reduced by allowing higher order interactions and more terms in the models. The required computation times and memory constraints however forced us to limit experimentation to 32-component pattern vectors and second order interactions. The best computationally feasible classifier used 195-term MARS expansions, three times the default maximum allowed model size. The training error was 4.9% while the testing error was the 6.3% given in Table 1.

The performance of the Learning 3-NN and the LVQ classifiers improved when the codebook size was increased. This is probably due to the fact that the data in a high dimensional pattern space are so sparse that reliable classification using a small number of class prototypes is not possible. The LVQ classifier used pattern vector dimension  $d = 38$  which had produced the best results with the 3-NN classifier. The size of the codebook  $\ell$  was selected with cross-validation and the codebook was initialized using  $\ell/10$  pattern vectors from each class. The classifier was trained using 10 epochs of LVQ1 and a varying number of LVQ2 epochs (for definitions of LVQ variants, see [55]). During each epoch the number of training vectors presented to the classifier was equal to the number of vectors in the actual training set and the learning coefficient  $\alpha$  was linearly decreased from 0.2 to zero. LVQ2 used relative window width of 0.5. Cross-validation suggested an optimal codebook size  $\ell = 8000$  and the use of one LVQ2 training epoch.

Of the two subspace classifiers, the performance of the Averaged Learning Subspace Method (ALSM) turned out to be very good. ALSM training was in most cases terminated when the training set error decreased to zero. Although the classification error for the independent testing sample also decreased in this process, it still probably remains above the level achievable with a greater amount of training data.

We next tested classification with the reject option. It is well known that if exceptional handling is possible, using the reject option can improve the performance of a classifier. Figure 4 illustrates rejection using equation (5). We show the classification error percentage as the function of the size of the reject class for the LLR method, which was chosen because it had the best error rate of all the single classifiers tested. The results are shown for the testing sample only; the cross-validation curve resembles closely the curve shown here.

Combining several different classifiers in a committee can provide an improvement over the use of a single classifier. As a final experiment we formed a committee from the LLR, ALSM, and L-3-NN classifiers, all of which performed individually very well. A majority voting rule was used and in situations where all three classifiers disagreed the decision of the L-3-NN classifier was enforced. Using this voting strategy the testing set classification error for the committee was 2.5%. Letting either the LLR or the ALSM classifier break the ties would in fact result in better performance but,

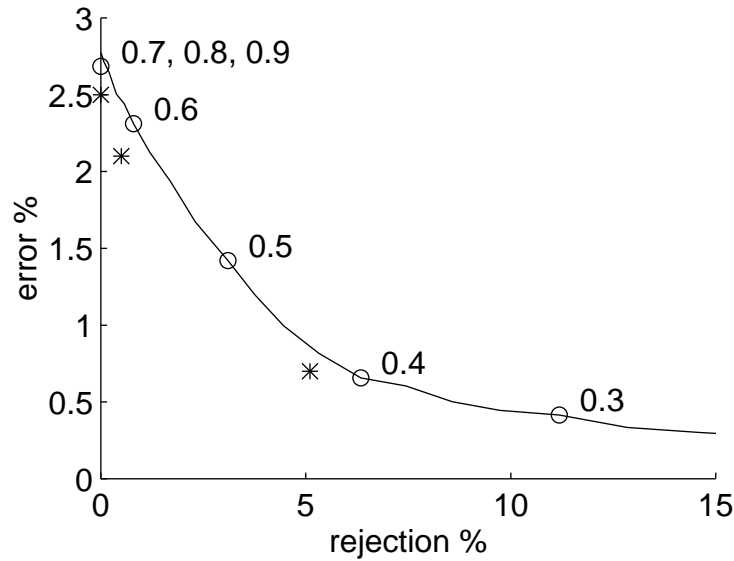


Figure 4: Error reject curve for LLR with the value of the threshold parameter  $\theta$  given at selected points. For comparison, the stars indicate the results obtained with the committee classifier.

in accordance with the single classifier design principles, we wanted to avoid using testing set performance to select an optimal voting strategy. Note however that this ideal is still partially compromised as the three committee members were selected on the basis of their good testing set performance. Rejecting ties between the three classes led to the rejection of 0.5% of the patterns and 2.1% classification error for the remaining 99.5% of the patterns. When any disagreement between the committee members was a cause for rejection, 5.1% of the vectors were rejected and the rest were classified with an error of 0.7%. As can be seen in Figure 4, these results are only slightly better than those achieved with the LLR classifier alone.

To summarize the results of our case study for single classifiers, the Local Linear Regression (LLR) method was the best classifier from the point of view of classification accuracy, with the Averaged Subspace Method (ALSM) following close behind. However, the amount of computation required for these two classifiers is quite different: evaluating the output of the LLR would make the method impossible to use in a practical application. As for the ALSM, the classification rule is based on a number of inner products and it is therefore relatively fast and simple. Furthermore, within the statistical error, several of the other classifiers have similar performance. The QDA and RDA methods which are based on the normality assumption did surprisingly well in our study. This is probably due to the fact that, for our data, the KL

transformed vectors appear to be approximately normally distributed. The Learning 3-NN method and ALSM were better in our study than their nonadaptive counterparts, 3-NN and CLAFIC, respectively. Although Learning  $k$ -NN and ALSM (or the basic Learning Subspace Method) are not directly neural network classifiers, they have been strongly motivated by neural algorithms and seem to be able to improve the classification performance by decision-directed learning using misclassified sample vectors.

## 6 Summary

Neural and statistical classifiers were discussed both from a theoretical point of view and in a case study of handwritten digit recognition. The starting point of neural network algorithms is quite different from that of statistical classifiers. Neural networks are on-line learning systems, intrinsically non-parametric and model-free, with the learning algorithms typically of the error correction type. For some neural learning algorithms, not even an explicit cost function exists. Thus the computational aspect is central: at least in principle, the computations should be on-line, simple and local, and avoid heavy numerical operations that use a fixed training set in batch mode. Most statistical classifiers, on the other hand, follow the Bayesian classification principle and either explicitly try to estimate the class densities and a priori probabilities, or at least the optimal discriminant functions by regression. Computational limits are set by current conventional hardware.

Despite these theoretical differences which may be important in the long range goals of artificial neural systems, in practice it is possible to view also present day neural classifiers in a statistical framework. Then the power of “neural” algorithms is increased flexibility of architecture in the sense of the richness of the discriminant function family and the possibility for incremental learning.

The performance of a number of most typical neural classifiers, including the Multi-Layer Perceptron (MLP) and Learning Vector Quantization (LVQ), was compared to various types of statistical classifiers. These included Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Regularized Discriminant Analysis (QDA), Kernel Discriminant Analysis (KDA), a tree classifier, and Flexible Discriminant Analysis (FDA) with MARS, the  $k$ -Nearest Neighbor ( $k$ -NN) classifier, and CLAFIC. Also, four methods of our own were used in the comparisons: Reduced Kernel Discriminant Analysis (RKDA), the Learning  $k$ -NN Classifier, the Averaged Learning Subspace Method (ALSM), and a modified version of KDA. Also, committee classifiers and classification with rejection were considered. The test case was

handwritten digit recognition with standard preprocessing and Karhunen-Loève feature extraction. The classifier design was done using cross-validation with a training set, and the classification errors were estimated with an independent testing set.

In these experiments, the Local Linear Regression (LLR) method, although computationally prohibitively heavy, was the best classifier from the point of view of classification accuracy, with the Averaged Learning Subspace Method (ALSM) following close behind. For methods having both a learning and a non-learning version, error correcting learning seemed to give an advantage.

## 7 Acknowledgments

We wish to thank Ari Hämmäläinen for computing the RKDA results and Paul Tikkanen for helping to run the initial tests for certain classifiers. This work was supported by grant 4172/95 from Tekes in the program “Adaptive and Intelligent Systems Applications”.

## References

- [1] H. S. Baird. Recognition technology frontiers. *Pattern Recognition Letters*, 14(4):327–334, Apr. 1993.
- [2] R. Becker, J. Chambers, and A. Wilks. *The NEW S Language*. Chapman and Hall, New York, 1988.
- [3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [4] J. L. Blue, G. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson. Evaluation of pattern classifiers for fingerprint and OCR applications. *Pattern Recognition*, 27(4):485–501, 1994.
- [5] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Müller, E. Säckinger, P. Y. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwritten digit recognition. In *Proceedings of 12th International Conference on Pattern Recognition*, volume II, pages 77–82. IAPR, IEEE Computer Society Press, October 1994.
- [6] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.
- [7] J. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. Touretzky,

- editor, *Advances in Neural Information Processing Systems 2*, pages 211–217, San Mateo, CA, 1990. Morgan Kaufmann Publishers.
- [8] J. Chambers and T. Hastie, editors. *Statistical Models in S*. Chapman and Hall, New York, 1992.
  - [9] B. Cheng and D. Titterton. Neural networks: A review from a statistical perspective. *Statistical Science*, 9(1):2–54, 1994.
  - [10] L. Clark and D. Pregibon. Tree-based models. Chapter 9 of [8].
  - [11] W. Cleveland and S. Devlin. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83:596–610, 1988.
  - [12] W. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. Technical report, AT&T Bell Laboratories, 1995. available as 95.3.ps in <http://netlib.att.com/netlib/att/stat/doc/>.
  - [13] P. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerical Mathematics*, 31:317–403, 1979.
  - [14] P. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall International, 1982.
  - [15] H. Drucker, C. Cortes, L. D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, September 1994.
  - [16] H. Drucker, R. Schapire, and P. Simard. Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4):705–719, 1993.
  - [17] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
  - [18] E. Fix and J. Hodges. Discriminatory analysis—nonparametric discrimination: Consistency properties. Technical Report Number 4, Project Number 21-49-004, USAF School of Aviation Medicine, Randolph Field, Texas, 1951. reprinted in [89].
  - [19] T. Flick, L. Jones, R. Priest, and C. Herman. Pattern classification using projection pursuit. *Pattern Recognition*, 23(12):1367–1376, 1990.
  - [20] J. Franke and E. Mandler. A comparison of two approaches for combining the votes of cooperating classifiers. In *Proceedings of the 11th International Conference on Pattern Recognition*, volume II, pages 611–614, Hague, August 1992. IAPR.
  - [21] J. Friedman. Regularized discriminant analysis. *Journal of the American Statistical Association*, 84(405):165–175, 1989.
  - [22] J. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19:1–141, 1991. with discussion.

- [23] J. Friedman and W. Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, 1981.
- [24] K. S. Fu. *Syntactic pattern recognition and applications*. Prentice-Hall, 1982.
- [25] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.
- [26] K. Fukunaga and R. R. Hayes. The reduced Parzen classifier. *IEEE Trans. Pattern Anal. and Machine Intell.*, PAMI-11(4):423–425, April 1989.
- [27] K. Fukunaga and J. M. Mantock. Nonparametric data reduction. *IEEE Trans. Pattern Anal. and Machine Intell.*, PAMI-6(1):115–118, January 1984.
- [28] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988.
- [29] M. D. Garris, J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C. L. Wilson. NIST form-based handprint recognition system. Technical report, National Institute of Standards and Technology, 1994.
- [30] M. Gilloux. Research into the new generation of character and mailing address recognition systems at the French post office research center. *Pattern Recognition Letters*, 14(4):267–276, Apr. 1993.
- [31] I. Grabec. Self-organization of neurons described by the maximum-entropy principle. *Biol. Cybern.*, 63:403–409, 1990.
- [32] I. Guyon. Applications of neural networks to character recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 5(1-2):353–382, 1991.
- [33] D. Hand. *Kernel Discriminant Analysis*. Research Studies Press, Cichester, 1982.
- [34] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, October 1990.
- [35] J. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.
- [36] T. Hastie and C. Loader. Local regression: Automatic kernel carpentry. *Statistical Science*, 8:120–143, 1993. with discussion.
- [37] T. Hastie and R. Tibshirani. *Generalized Additive Models*. Chapman & Hall, 1990.
- [38] T. Hastie and R. Tibshirani. Discriminant analysis by Gaussian mixtures. Technical report, AT&T Bell Laboratories, 1994. available as 94.6.ps in <http://netlib.att.com/netlib/att/stat/doc/>.
- [39] T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89:1255–1270, 1994.
- [40] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., New York, 1994.

- [41] W. Highleyman. Linear decision functions with application to pattern recognition. *Proc. IRE*, 50:1501–1514, 1962.
- [42] G. E. Hinton, M. Revow, and P. Dayan. Recognizing handwritten digits using mixtures of linear models. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Neural Information Processing Systems 7*, pages 1015–1022, Cambridge, MA, 1995. MIT Press.
- [43] T. K. Ho, J. J. Hull, and S. N. Srihari. A regression approach to combination of decisions by multiple character recognition algorithms. In D. P. D’Amato, W.-E. Blanz, B. E. Dom, and S. N. Srihari, editors, *Proceedings of SPIE Conference on Machine Vision Applications in Character Recognition and Industrial Inspection*, number 1661 in SPIE, pages 137–145. SPIE, February 1992.
- [44] T. K. Ho, J. J. Hull, and S. N. Srihari. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, January 1994.
- [45] L. Holmström and A. Hämmäläinen. The self-organizing reduced kernel density estimator. In *Proceedings of the 1993 IEEE International Conference on Neural Networks, San Francisco, California, March 28 - April 1*, volume 1, pages 417–421, 1993.
- [46] L. Holmström, S. Sain, and H. Miettinen. A new multivariate technique for top quark search. *Computer Physics Communications*, 88:195–210, 1995.
- [47] Y. S. Huang, K. Liu, and C. Y. Suen. The combination of multiple classifiers by a neural network approach. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(3):579–597, 1995.
- [48] Y. Idan and J.-M. Auger. Pattern recognition by cooperating neural networks. In S.-S. Chen, editor, *Proceedings of SPIE Conference on Neural and Stochastic Methods in Image and Signal Processing*, number 1766 in SPIE, pages 437–443. SPIE, July 1992.
- [49] Y. Idan, J.-M. Auger, N. Darbel, M. Sales, R. Chevallier, B. Dorizzi, and G. Cazuguel. Comparative study of neural networks and non parametric statistical methods for off-line handwritten character recognition. In I. Aleksander and J. Taylor, editors, *Proceedings of the International Conference on Artificial Neural Networks*, volume 2, pages 1607–1610, Brighton, September 1992. ENNS, North-Holland.
- [50] S. Impedovo, L. Ottaviano, and S. Occhinegro. Optical character recognition—a survey. *International Journal of Pattern Recognition and Artificial Intelligence*, 5(1-2):1–24, 1991.
- [51] R. A. Jacobs. Methods for combining experts’ probability assessments. *Neural Computation*, 7(5):867–888, September 1995.

- [52] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, March 1994.
- [53] F. Kimura and M. Shridhar. Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition*, 24(10):969–983, 1991.
- [54] T. Kohonen. The ‘neural’ phonetic typewriter. *Computer*, 21(3):11–22, 1988.
- [55] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.
- [56] P. Koistinen and L. Holmström. Kernel regression and backpropagation training with noise. In J. Moody, S. Hanson, and R. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 1033–1039, San Mateo, CA, 1992. Morgan Kaufmann Publishers.
- [57] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Neural Information Processing Systems 7*, pages 231–238, Cambridge, MA, 1995. MIT Press.
- [58] J. Laaksonen and E. Oja. Classification with learning  $k$ -nearest neighbors. In *Proceedings of the International Conference of Neural Networks*, 1996. to be published.
- [59] L. Lam and C. Y. Suen. A theoretical analysis of the application of majority voting to pattern recognition. In *Proceedings of 12th International Conference on Pattern Recognition*, volume II, pages 418–420. IAPR, IEEE Computer Society Press, October 1994.
- [60] L. Lam and C. Y. Suen. Optimal combinations of pattern classifiers. *Pattern Recognition Letters*, 16:945–954, 1995.
- [61] J. Lampinen and E. Oja. Distortion tolerant pattern recognition based on self-organizing feature extraction. *IEEE Transactions on Neural Networks*, 6(3):539–547, 1995.
- [62] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.
- [63] D.-S. Lee, S. N. Srihari, and R. Gaborski. Bayesian and neural network pattern recognition: a theoretical connection and empirical results with handwritten characters. In I. K. Sethi and A. K. Jain, editors, *Artificial Neural Networks and Statistical Pattern Recognition*, pages 89–108. Elsevier Science Publishers, 1991.
- [64] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proc. Fifth Berkeley Symp. Math. Stat. and Prob.*, pages 281–297. Berkeley, CA: U.C. Berkeley Press, 1967.
- [65] J. Mantas. An overview of character recognition methodologies. *Pattern Recog-*

- niton, 19(6):425–430, 1986.
- [66] P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman & Hall, second edition, 1989.
  - [67] W. S. McCulloch and W. Pitts. A logical calculus of the idea immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
  - [68] G. McLachlan. *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, Inc., 1992.
  - [69] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine learning, neural and statistical classification*. Ellis Horwood Limited, 1994.
  - [70] J. Morgan and J. Sonquist. Problems in the analysis of survey data and a proposal. *Journal of the American Statistical Association*, 58:415–434, 1963.
  - [71] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of OCR research and development. *Proceedings of the IEEE*, 80(7):1029–1058, July 1992.
  - [72] C. Nadal, R. Legault, and C. Y. Suen. Complementary algorithms for the recognition of totally unconstrained handwritten numerals. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 443–449, Atlantic City, NJ, June 1990. IAPR.
  - [73] E. Oja. *Subspace Methods of Pattern Recognition*. Research Studies Press, Letchworth, 1983.
  - [74] E. Oja. Self-organizing maps and computer vision. In H. Wechsler, editor, *Neural Networks for Perception*, volume I, chapter II.9, pages 368–385. Academic Press, 1992.
  - [75] M. P. Perrone. Pulling it all together: Methods for combining neural networks. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Neural Information Processing Systems 6*, pages 1188–1189, Cambridge, MA, 1994. Morgan Kaufman.
  - [76] M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman & Hall, 1993.
  - [77] C. E. Priebe and D. J. Marchette. Adaptive mixtures: Recursive nonparametric pattern recognition. *Pattern Recognition*, 24(12):1197–1209, 1991.
  - [78] C. E. Priebe and D. J. Marchette. Adaptive mixture density estimation. *Pattern Recognition*, 26(5):771–785, 1993.
  - [79] R. Redner and H. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2), April 1984.
  - [80] M. D. Richard and R. P. Lippman. Neural network classifiers estimate bayesian *a posteriori* probabilities. *Neural Computation*, 3(4):461–483, 1991.
  - [81] B. Ripley. Neural networks and related methods for classification. *Journal of the Royal Statistical Society, Series B*, 56(3):409–456, 1994. with discussion.

- [82] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in brain. *Psychological Review*, 65:386–408, 1958.
- [83] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, Washington, DC., 1961.
- [84] R. J. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, Inc., 1992.
- [85] H. Schwenk and M. Milgram. Transformation invariant autoassociation with application to handwritten character recognition. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Neural Information Processing Systems 7*, pages 991–998, Cambridge, MA, 1995. MIT Press.
- [86] D. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, Inc., 1992.
- [87] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce english text. *Journal of Complex Systems*, 1(1):145–168, February 1987.
- [88] D. Shanno and K. Phua. Remark on algorithm 500. *ACM Transactions on Mathematical Software*, 6(4):618–622, 1980.
- [89] B. Silverman and M. Jones. E. Fix and J.L. Hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation—commentary on Fix and Hodges (1951). *International Statistical Review*, 57(3):233–247, 1989.
- [90] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
- [91] F. Śmieja. The Pandemonium system of reflective agents. Technical Report 1994/2, German National Research Center for Computer Science (GMD), 1994. available in <http://borneo.gmd.de/AS/janus/publi/publi.html>.
- [92] P. Smyth and J. Mellstrom. Fault diagnosis of antenna pointing systems using hybrid neural network and signal processing models. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 667–674. Morgan Kaufmann Publishers, 1992.
- [93] P. Sollich and A. Krogh. Learning with ensembles: How over-fitting can be useful. In *Neural Information Processing Systems 8*, 1995. to appear.
- [94] D. Specht. A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576, November 1991.
- [95] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.
- [96] S. N. Srihari. Recognition of handwritten and machineprinted text for postal address interpretation. *Pattern Recognition Letters*, 14(4):291–302, Apr. 1993.
- [97] K. Stokbro, D. Umberger, and J. Hertz. Exploiting neurons with localized

- receptive fields to learn chaos. *Complex Systems*, 4:603–622, 1990.
- [98] C. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36:111–147, 1974.
- [99] C. Y. Suen, M. Berthold, and S. Mori. Automatic recognition of handprinted characters—the state of the art. *Proceedings of the IEEE*, 68(4):469–487, April 1980.
- [100] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet, and L. Lam. Building a new generation of handwriting recognition systems. *Pattern Recognition Letters*, 14(4):303–315, Apr. 1993.
- [101] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE*, 80(7):1162–1180, July 1992.
- [102] C. Therrien. *Decision, Estimation and Classification: An Introduction to Pattern Recognition and Related Topics*. John Wiley & Sons, 1989.
- [103] H. Tråvén. A neural network approach to statistical pattern classification by “semiparametric” estimation of probability density functions. *IEEE Transactions on Neural Networks*, 2(3):366–377, 1991.
- [104] V. Tresp and M. Taniguchi. Combining estimators using non-constant weighting functions. In G. Tesauero, D. S. Touretzky, and T. K. Leen, editors, *Neural Information Processing Systems 7*, pages 419–426, Cambridge, MA, 1995. MIT Press.
- [105] G. Tutz. An alternative choice of smoothing for kernel-based density estimates in discrete discriminant analysis. *Biometrika*, 73:405–411, 1986.
- [106] W. Venables and B. Ripley. *Modern Applied Statistics with S-Plus*. Springer-Verlag New York, Inc., New York, 1994.
- [107] T. Wakahara. Towards robust handwritten character recognition. *Pattern Recognition Letters*, 14(4):345–354, Apr. 1993.
- [108] M. Wand and M. Jones. *Kernel Smoothing*. Chapman & Hall, 1995.
- [109] S. Watanabe, P. F. Lambert, C. A. Kulikowski, J. L. Buxton, and R. Walker. Evaluation and selection of variables in pattern recognition. In J. Tou, editor, *Computer and Information Sciences II*. Academic Press, New York, 1967.
- [110] H. White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464, 1989.
- [111] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.
- [112] J. X. Wu and C. Chan. A three-layer adaptive network for pattern density estimation and classification. *International Journal of Neural Systems*, 2(3):211–220, 1991.
- [113] L. Xu and M. I. Jordan. EM learning on a generalized finite mixture model for

- combining multiple classifiers. In *Proceedings of the World Congress on Neural Networks*, volume IV, pages 227–230, 1993.
- [114] L. Xu, M. I. Jordan, and G. E. Hinton. An alternative model for mixtures of experts. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Neural Information Processing Systems 7*, pages 633–640, Cambridge, MA, 1995. MIT Press.
- [115] L. Xu, A. Krzyzak, and C. Y. Suen. Associative switch for combining multiple classifiers. In *Proceedings of 1991 International Joint Conference on Neural Networks*, volume 1, pages 43–48. IEEE, INNS, July 1991.
- [116] L. Xu, A. Krzyżak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, May 1992.
- [117] T. Y. Young and T. W. Calvert. *Classification, Estimation and Pattern Recognition*. American Elsevier Publishing Co., Inc., New York, 1974.