Comparison of Statistical and Neural Classifiers and Their Applications to Optical Character Recognition and Speech Classification

Ethem Alpaydın, Fikret Gürgen
Department of Computer Engineering
Boğaziçi University
TR-80815 İstanbul Turkey
{alpaydin,gurgen}@boun.edu.tr

Neural Network Systems Techniques and Applications (in print)
C. T. Leondes (Ed.), ©ACADEMIC Press

October 24, 1996

Abstract

We give a review of basic statistical and neural techniques for classification. Statistical techniques are based on the idea of estimating class-conditional likelihoods and using Bayes rule to convert these to posterior class probabilities whereas neural techniques estimate directly the posteriors. Statistical techniques include (i) Parametric (Gaussian) Bayes classifiers, (ii) Nonparametric kernel-based density estimators like k-nearest neighbor and Parzen windows, and (iii) mixtures of (Gaussian) densities (a special case of which is the Learning Vector Quantization). As neural classifiers, we include simple perceptrons and multilayer perceptrons with sigmoid and Gaussian hidden units. The neural and statistical techniques are quite similar in many respects and many approaches have been discovered independently twice, once in 1960s by statisticians and once in 1980s by the neural network researchers. One of the aims of this article is to make this link more apparent. We also discuss two, most popular, pattern recognition applications: Optical character recognition and speech recognition. Though they seem different, in many respects, the two applications are similar and in the past, almost the same techniques have been applied for their implementation. We implement the well known statistical and neural classification techniques for two datasets of these applications and compare them in terms of generalization accuracy, memory requirement and learning time. We especially advise to take into account statistics of the sample even if a neural classifier is to be used. The similarity between statistical and neural techniques is greater than generally agreed and simple statistical methods like k-NN perform generally quite well and much of the functionality of neural networks like distributed parallel computation can be obtained by such methods without requiring complicated computation and precise error minimization procedures.

Keywords— Statistical pattern recognition, artificial neural networks, optical character recognition, speech recognition, Bayes decision theory, nonparametric estimation.

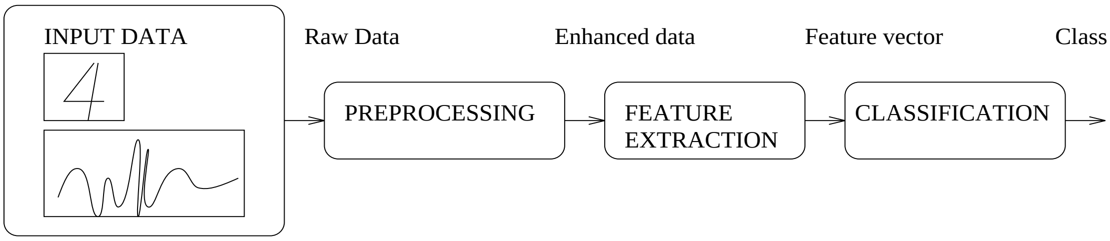

Figure 1: A pattern recognition system where input is an image, like in optical character recognition or a time series, like in speech classification.

1 Introduction

Improving person-machine communication leads to a wider use of advanced information technologies. Towards this aim, character recognition and speech recognition are two applications whose automatization allow easier interaction with a computer. As they are the basic means of person-to-person communication, they are known by everyone and require no special training. Speech in particular is the most natural form of human communication and writing is the tool by which humanity has stored and transfered its knowledge for many millenia.

In a typical pattern recognition system (Fig. 1), the first step is the acquisition of data. This raw data is preprocessed to suppress noise and normalize input. Features are those parts of the signal that carry information salient to its identity and their extraction is an abstraction operation where the important is extracted and the irrelevant is discarded. Classification is the assignment of the input as an element of one of a set of predefined classes.

The rules for classification are generally not known exactly and thus are estimated. A classifier is written as a parametric model whose parameters are computed using a given training sample to optimize a given error criterion. Different approaches for classification differ in their assumptions about the model, in the way parameters are computed or in the error criterion they optimize.

Statistical classifiers model the class-conditional densities and base their decisions on the posteriors which are computed using the class-conditional likelihoods and the priors. Likelihoods are assumed to either come from a given probability density family, e.g., normal, from a mixture of such densities or written in a completely nonparametric way. Bayes decision theory then allows us to choose the class that minimizes the decision risk. The parameters of the densities are estimated to maximize the likelihood of the given sample for that class.

This opposes to approaches where the discriminants are directly estimated. Neural networks are such approaches and their outputs can be converted directly to posteriors, eliminating the need of assuming a statistical model. From a statistical perspective, a multilayer network is a linear sum of nonlinear basis functions. In the neural network terminology, the nonlinear basis functions are called hidden units and the parameters are called connection weights. In a training process, given a training sample, the weights that minimize the difference between network outputs and required outputs is computed.

This article has the aim of comparing these two approaches and extends a previous study [3]. In Section 2, we define the two applications that we are concerned with in this study, namely optical character recognition and speech recognition. We show that these two applications have many common subproblems and quite similar approaches have been used in the past to implement them, both statistical and neural. Section 3 details how in the two applications, data are acquired and preprocessed before they can be fed to the classifier. In Section 4, we define formally the problem from a statistical point of view and explain the three approaches of parametric, nonparametric and semi-parametric estimation. In Section 5, we discuss the neural approaches like simple and multilayer perceptrons and radial basis function networks. A literature survey for the two applications is given in Section 6. In Section 7, we give simulation results on two datasets. We conclude in Section 8.

2 Applications

Character recognition is of two forms: In *printed* character recognition, any character image is one of a predefined number of styles which are called *fonts*. Printed character recognition systems generally work by storing templates of character images for all fonts and matching the given image against these stored images to choose the best match. This opposes to *handwritten* character recognition where there are practically infinite ways of writing a character. It is this latter that we are interested in. In handwritten character recognition, the medium may be two sorts: In *optical* character recognition, the writer writes generally on paper by using a marker of different brightness. The contrast is acquired optically through a scanner or camera and a two dimensional image is formed. Because recognition is done generally long after the writing is done, it is named *off-line*. In *pen-based* character recognition, the writing is done on a touch-sensitive pad using a special pen. While it is moved over the pad, the coordinates of the pen-tip are returned at each sampling, leading to a sequence of pen-tip positions for each written character. Recognition is done while writing takes place and is called *on-line*. A special journal issue on different approaches to character recognition, edited by Pavlidis and Mori [38], recently appeared.

In speech recognition, there is no analogy to printed character recognition as the sound to be recognized is natural, never synthetic. Speech is captured using a microphone and is digitized to get a set of samples in time from the sound waveform. The sound spectogram is a three dimensional representation of the speech intensity, in different frequency bands, over time. Another way of representing speech is by modeling the human vocal tract as a tube whose resonances produce speech; they are called *formants* and they represent the frequencies that pass the most acoustic energy; typically there are three. For a more detailed analysis, refer to the book by Rabiner and Juang [41].

The two tasks of character recognition and speech recognition have a number of common problems. One is the segmentation of characters or phonemes from a stream of text image or speech. This is especially a problem when cursive handwriting or continuous phrases is the case; the system before recognizing individual characters or phonemes, should determine the boundaries. To facilitate recognition, some systems require that inputs be isolated before recognition, by providing separate boxes for different characters or slow, careful articulation of words.

Another problem is that of being independent from writer or speaker; a recognizer that can recognize inputs from a small set of people is rarely useful. But the recognizer should also have the ability to adapt to a particular user if required so as to be able to recognize that user's handwriting/speech with higher accuracy.

Recognition is also dependent on the domain. If only postal codes are to be optically recognized, then in most countries only digits are valid classes or a voice-controlled system may have a small set of commands to be recognized. To minimize errors and rejects, more complicated tasks require also the integration of a vocabulary so as to be able to use lexical information to aid in recognition. This creates the problems of storing a lexicon and accessing it efficiently.

The best existing systems perform well only on artificially constrained tasks. Performance is better if samples are provided for all speakers/writers, when words are spoken/letters are written in isolation, when the vocabulary size is small and when restricted language models are used to constrain allowable sequences [27, 38].

3 Data Acquisition and Preprocessing

3.1 Optical Character Recognition

In optical character recognition, preprocessing should be done before individual character images can be fed to the classifier.

Depending on how small the characters are, a suitable resolution should be chosen first for scanning. Typically a resolution of 300 dpi (dots per inch) is used. With smaller resolutions and smaller character images, dots and accents may be lost or images may get connected.

In most applications, characters are written in special places on pre-printed forms [38, 16]. So the first step after scanning is registration which is that of determining how the original form is translated and rotated to get the final image. This is done by matching a number of points in the input form to the original blank form. Then given the coordinates of the input fields, their coordinates in the input form can be computed and the input field images are extracted. The characters in a field are then segmented and individual character images are obtained. These are then size-normalized to scale all characters to the same height and width and slant-normalized to reduce the slant variations in order to be left only with shape differences.

The so-formed bitmap can be fed as input to the classifier or features may be extracted from the image to get invariances and/or reduce dimensionality. One common approach is low-pass filtering the image, which gives invariance to small translations. Easy to extract features in character images are the ratio of width to height, number of *on* pixels, number of line crossings along certain horizontal and vertical directions, etc. A more expensive preprocessing is to have small local kernels with which all parts of the image are convolved, to detect line segments, corners, etc.

3.2 Speech Recognition

Digitized speech samples are obtained by an antialiasing filter and an analog-to-digital converter (A/D) [33, 41]. A low-pass anti-aliasing filter should be set below the Nyquist frequency (half of the sample rate) so that the Fourier transform of the signal will be band limited. An A/D converter commonly consists of a sampling circuit and a hold circuit. A 10-12 kHz sampling frequency usually includes the first five formants for most talkers, but may not capture all the unvoiced energy such as the /s/ phoneme. An 8 kHz sampling frequency can be selected to be used in a 4 kHz telephone channel.

Once the speech signal has been digitized, the discrete-time representation is usually analyzed within short-time intervals. Depending on the application, an analysis window of 5 – 25 ms is chosen in which it is assumed that the speech signal is time-invariant or quasi-stationary. Time-invariant analysis is essential since parameter estimation in time-varying (non-stationary) system is a much more difficult problem.

The utterances of speakers are recorded in a soundproof room or in certain environmental conditions such as no-noise with a microphone at a certain bandwidth. The required segments of each utterance are manually, or using an accurate segmentation algorithm, endpointed and processed into frames.

A linear predictive coding (LPC) based analysis procedure [33, 41, 34] can be used to obtain desired features such as cepstrum coefficients, fast Fourier transform (FFT) coefficients. Recent feature extraction methods concentrate also on auditory modelling and time-frequency representation of speech signals.

4 Statistical Classifiers

In pattern recognition, we are asked to assign a multidimensional input $\boldsymbol{x} \in \Re^d$ to one of a set of classes $C_j, j = 1, \ldots, m$ [12, 31]. Sometimes the additional action of *reject* is added to choose

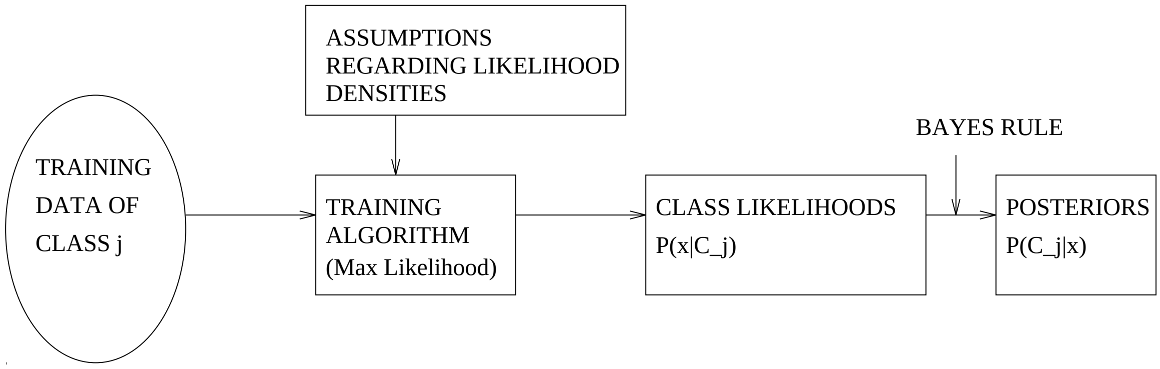

Figure 2: Building a statistical classifier.

when no class or more than one class is probable. A *classifier* is then a mapping from $\Re^d$ to $\{C_1, \dots, C_m, C_{rej}\}$.

In statistical decision theory, actions have costs and the decision having the minimum cost is made [5]. Assuming that correct decisions have no cost, all incorrect decisions have equal cost and no rejects, for minimum expected cost (or risk), the so-called *Bayes' decision rule* states that a given input should be assigned to the class with the highest *posterior* probability [12, 5, 31]:

$$c = \arg\max_j P(C_j|\boldsymbol{x}) \tag{1}$$

The posteriors are almost never exactly known and thus are estimated. We are given a sample $\mathcal{X} = \{\boldsymbol{x}_i, y_i\}, i = 1, \dots, n$ and $y \in \{C_1, \dots, C_m\}$. In statistical pattern recognition theory (Fig. 2), Bayes rule factors the posterior into a *prior* class probability $P(C_j)$ and a class-conditional density or *likelihood* $p(\boldsymbol{x}|C_j)$:

$$P(C_j|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|C_j)P(C_j)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|C_j)P(C_j)}{\sum_k p(\boldsymbol{x}|C_k)P(C_k)} \tag{2}$$

The estimate of the prior is just the fraction of samples belonging to that class. If n_j is the number of samples belonging to class j, $\sum_j n_j = n$ then

$$\hat{P}(C_j) = n_j/n \tag{3}$$

The real problem is that of estimating the class-conditional likelihood densities $p(\boldsymbol{x}|C_j)$. There are three approaches:

1. *Parametric Methods.* These assume that class-conditional densities have a certain parametric form, e.g., normal, whose sufficient statistics are estimated from the data [12, 31]. These methods generally reduce to *distance-based* methods where depending on assumptions made on the data, the good distance metric is chosen.

2. *Non-parametric Methods.* When no such assumptions can be done, the densities need be estimated directly from the data. These are also known as *kernel-based estimators* [12, 45, 46].

3. *Semi-parametric Methods.* The densities are written as a mixture model whose parameters are estimated [12, 40, 50, 36, 48]. In the case of normal mixtures, this approach is equivalent to cluster-based classification strategies like LVQ of Kohonen [24] and is similar to Gaussian radial basis function networks [32].

A decision rule as given in Eq. (1) has the effect of dividing the input space into mutually exclusive regions called the *decision regions* where each region is assigned to one of the m classes. Bounding these decision regions are the decision boundaries or *discriminants* that separate the examples of one class from others. Pattern classification may equally well be thought of as defining appropriate discriminant functions, $g_j(\boldsymbol{x})$, and we assign feature vector $\boldsymbol{x}$ to class c if

$$g_c(\boldsymbol{x}) = \max_j g_j(\boldsymbol{x}) \tag{4}$$

An immediate discriminant function is the posterior probability, or its variants. The following are all equivalent:

$$\begin{aligned} g_j(\boldsymbol{x}) &= P(C_j|\boldsymbol{x}) \\ g_j'(\boldsymbol{x}) &= p(\boldsymbol{x}|C_j)P(C_j) \\ g_j''(\boldsymbol{x}) &= \log p(\boldsymbol{x}|C_j) + \log P(C_j) \end{aligned} \tag{5}$$

4.1 Parametric Bayes classifiers

The shape of decision regions defined by a Bayes classifier depends on the assumed form for $p(\boldsymbol{x}|C_j)$ [12, 31]. Most frequently, it is taken to be multivariate normal which assumes that examples from a class are noisy versions of an ideal class member. The ideal class prototype is given by the class mean $\boldsymbol{\mu}_j$, and the characteristics of the noise appear as the covariance matrix Σ_j. When $p(\boldsymbol{x}|C_j) \sim \mathcal{N}(\boldsymbol{\mu}_j, \Sigma_j)$, it is written as:

$$\hat{p}(\boldsymbol{x}|C_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} \exp\left[-(1/2)(\boldsymbol{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_j)\right] \tag{6}$$

This leads to the following discriminant function (ignoring the common term of $-(d/2)\log 2\pi$):

$$g_j(\boldsymbol{x}) = -(1/2)\log|\Sigma_j| - (1/2)(\boldsymbol{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_j) + \log P(C_j) \tag{7}$$

When $\boldsymbol{x}$ is d dimensional, the free parameters are d for the mean vector and $d(d+1)/2$ for the (symmetric) covariance matrix. This latter is $O(d^2)$ which is disturbing if d is large. A large number of free parameters both makes the system more complex and requires larger training samples for their reliable estimation. Thus assumptions are made to keep the number of parameters small which has the effect of *regularization*.

4.1.1 Independent Features of Equal Variances

When the dimensions of the feature vector $\boldsymbol{x}$ are independent, i.e., $\mathrm{Cov}(x_k, x_l) = 0, k \neq l, \forall k, l = 1, \ldots, d$ and have equal variances $\mathrm{Var}(x_k) = \sigma^2, \forall k = 1, \ldots, d$, then $\Sigma_j = \Sigma = \sigma^2 I$. Because the covariance matrices are equal, the first term of Eq. (7) can be ignored. $\Sigma^{-1} = (1/\sigma^2)I$ and we obtain:

$$g_j(\boldsymbol{x}) = -(1/2\sigma^2)||\boldsymbol{x} - \boldsymbol{\mu}_j||^2 + \log P(C_j) \tag{8}$$

Assuming equal priors, this reduces to assigning input to the class with the nearest mean. If the priors are not equal, the discriminant is moved towards the less likely mean. Geometrically speaking, class densities are hyperspherical with $\boldsymbol{\mu}_j$ as the center and σ^2 defining the radius (Fig. 3). It can easily be shown that the discriminants $\{\boldsymbol{x}|g_i(\boldsymbol{x}) = g_j(\boldsymbol{x}), i \neq j\}$ are linear. The number of parameters for m classes is $m \cdot d$ for the means and 1 for σ^2.

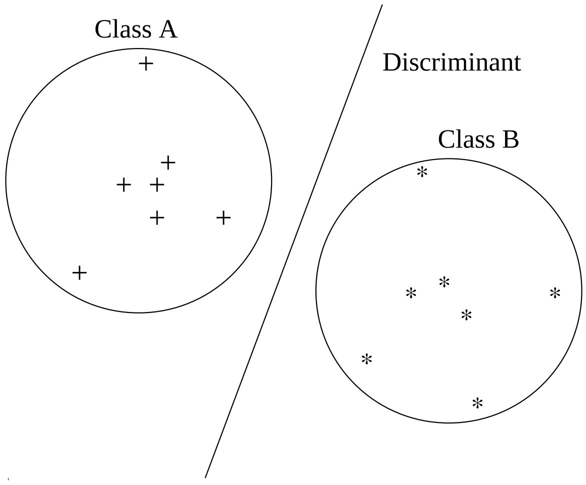

Figure 3: Example two class problem with equal variances and zero covariances and the linear discriminant.

4.1.2 Independent Features of Unequal Variances

If features are independent and the variances along different dimensions vary: $\mathrm{Var}(x_k) = \sigma_k^2, \forall k = 1, \ldots, d$, but are equal for all classes, we obtain:

$$g_j(\boldsymbol{x}) = -\sum_{k=1}^{d} \frac{(x_k - \mu_{jk})^2}{2\sigma_k^2} + \log P(C_j) \tag{9}$$

This also assigns the input to the class of the nearest mean but now Euclidean distance is generalized to *Mahalanobis distance* taking also differences in variances into account. Class densities now are axis-aligned hyperellipsoids and the discriminants they lead to are linear (Fig. 4). The number of parameters is $m \cdot d + d$.

4.1.3 Arbitrary Covariances

We are not interested in estimating full covariance matrices as $O(d^2)$ parameters require quite large training samples for accurate estimation. It is known that when classes have arbitrary but equal covariances, the discriminants are linear and we will be considering linear discriminants in more detail in Section 5.1.

When classes have different and full covariances, discriminants can be quadratic. The total number of parameters to be estimated is $m(d + d(d+1)/2)$ and this can only be feasible with quite small d and/or very large number of samples for not having ill-conditioned covariance matrices. It may thus be preferrable to use a linear discriminant, e.g., Fisher's linear discriminant, when we do not have enough data to estimate it accurately even if we know that the two covariance matrices are different and the discriminant is quadratic. When a common covariance is assumed, this introduces an effect of regularization. Friedman's *regularized discriminant analysis* writes the covariance matrix of a class as a weighted sum of the estimated covariance matrix of that class and the covariance matrix common to all classes, the relative weight of two being estimated using cross-validation. The common covariance matrix can even be forced to be diagonal if there is even less data available providing further regularization [31].

There are also techniques to decrease the dimensionality. One is *subset selection* which means choosing the most important p dimensions from d, ignoring the $d-p$ dimensions. *Principal component analysis* (PCA) chooses the most important p linear combinations of the d dimensions.

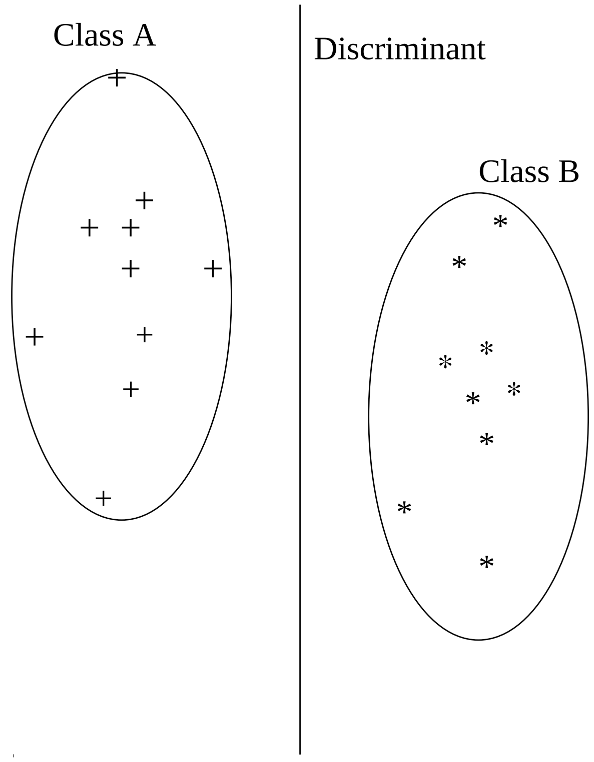

Figure 4: Example two class problem with different variances and zero covariances and the linear discriminant.

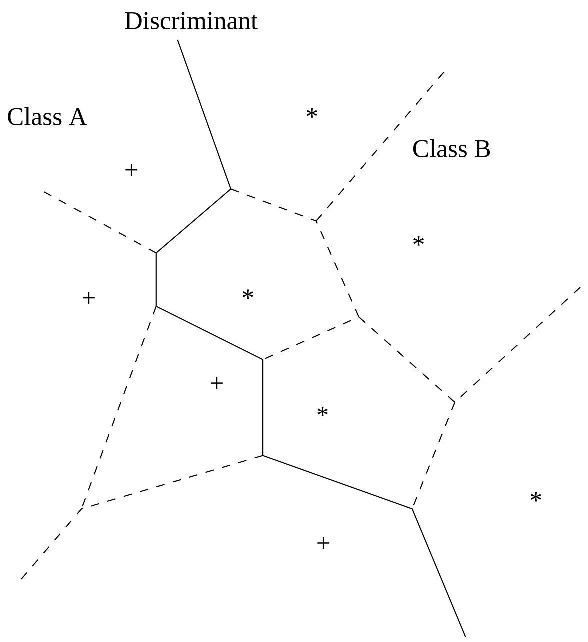

Figure 5: Example two class problem with sample points and the arbitrary discriminant found by one nearest neighbor. The dotted lines show the Voronoi tesselation.

4.2 Nonparametric Kernel-Based Density Estimators

In parametric estimation, we assume the knowledge of a certain form of density family for the likelihood whose parameters are estimated from the data. In the nonparametric case, we directly estimate the entire density function. Then we need a large sample for our estimate not to be biased by the particular sample we use. The *kernel estimator* is given as

$$\hat{p}(\boldsymbol{x}|C_j) = \frac{1}{n_j h^d} \sum_{i=1}^{n_j} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right) \tag{10}$$

h is the *window width* or the *smoothing parameter*. Depending on the shape of K, one can have various estimators [45].

One disadvantage of kernel estimators is the requirement of storing the whole sample. One possibility is to selectively discard patterns that do not convey much information [1]. Another is to cluster data and keep reference vectors that represent clusters of patterns instead of the patterns themselves. The semiparametric mixture models discussed in Section 4.3 correspond to this idea.

4.2.1 K-nearest neighbor (k-NN).

Let us denote the kth nearest sample to $\boldsymbol{x}$ as $\boldsymbol{x}^{[k]}$ and let $V^k(\boldsymbol{x})$ be the volume of the d-dimensional sphere of radius $r^k \equiv ||\boldsymbol{x} - \boldsymbol{x}^{[k]}||$; thus $V^k(\boldsymbol{x}) = r^k c_d$ where c_d is the volume of the unit sphere in d dimensions, e.g., $c_1 = 2$, $c_2 = \pi$, $c_3 = 4\pi/3$, etc. If out of the k neighbors, k^j of them are labelled ω_j, then the k-nearest neighbor estimate is (Fig. 5)

$$\hat{P}(\omega_j|\boldsymbol{x}) = k^j/k. \tag{11}$$

4.2.2 Parzen windows

For $\hat{p}$ to be a legitimate density function, K should be nonnegative and integrate to one. For a smooth approximation, K is generally taken to be the normal density:

$$K(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi}}\right)^d \exp\left[-\frac{\|\mathbf{u}\|^2}{2}\right] \tag{12}$$

Here the kernel estimator is a sum of "bumps" placed at the observations. K determines the shape of the bumps while h determines their widths. When spheric bumps with equal h in all dimensions are used, this corresponds to using Euclidean distance. If this assumption of equal variances (and independent dimensions) is not valid, then different variances (and covariances) can be estimated and Mahalanobis distance can be used instead. This also applies to k-NN.

4.2.3 Choosing h or k

In kernel-based estimation, the correct choice of the spread parameter (k or h) is critical. If it is large, then even distant neighbors affect the density at $\boldsymbol{x}$ leading to a very smooth estimate. If it is small, $\hat{p}$ is the superposition of n sharp pulses centered at the samples and is a "noisy" estimate.

With Parzen windows, when h is small with a small sample, it is possible that no samples fall in the kernel leading to a zero estimate; k-NN guarantees that k samples fall in the kernel. Small or large h leads to a decrease in success. When h is small, there are few samples and when it is large, there are too many. The good h value is to be found using cross-validation on a separate set.

For the k-nearest neighbor, it has been shown [12] that the performance of the 1-nearest neighbor in classification is never worse than twice the Bayesian risk where complete knowledge of the distributions are assumed. It can thus be said that at least half of this knowledge is provided by the nearest neighbor. The performance can be improved by increasing the number of neighbors considered in which case the error asymptotically converges to the Bayes risk.

When the samples are noisy, we expect k-NN with small k not to perform well. Large k takes into account the effect of very distant samples and thus is not good either. When h (or k) is decreased to decrease bias, variance increases and vice versa. This can intuitively be explained as follows [17]. When h (or k) is large, $\hat{p}$ is the weighted average of many samples and thus does not change much from one sample set to another. The variance contribution is small, however response is biased toward the population response (In the extreme case, when $h \to \infty$, $\hat{p}$ is the sample average and is independent of the input $\boldsymbol{x}$). On the other extreme, when h is small, there is small bias but the response is dependent on the particular sample used, therefore the variance contribution is high. The same argument holds with k-NN, with k in place of h. Choosing h or k implies a trade-off between bias (systematic error) and variance (random error).

4.3 Semiparametric Mixture Models

The parametric approach assumes that examples of a class are corrupted versions of an ideal class prototype. This ideal prototype is estimated by the mean of the examples and examples are classified depending on how similar they are to these prototypes. In certain cases, for a class, it is not possible to choose one single prototype; instead there may be several. For example in character recognition while writing '7' one prototype may be a seven with a horizontal middle bar (European version) and one without (American version). A *mixture density* defines the class-conditional density as a sum of a small number of densities [12, 36, 31]:

$$\hat{p}(\boldsymbol{x}|C_j) = \sum_{h=1}^{h_j} p(\boldsymbol{x}|\omega_{jh}, C_j) P(\omega_{jh}) \tag{13}$$

where the conditional densities $p(\boldsymbol{x}|\omega_{jh}, C_j)$ are called the *component densities* and the prior probabilities $P(\omega_{jh})$ are called the *mixing parameters*. Note that here we have one mixture model for each class leading to an overall *mixture of mixtures* [48].

We want to estimate the parameters Φ_j, that includes the sufficient statistics of the component densities and the mixing proportions, that maximizes the likelihood of a given iid sample $\mathcal{X}_j$ of class j:

$$\begin{aligned}\mathcal{L}(\Phi_j|\mathcal{X}_j) &= \sum_{i=1}^{n_j} \log p(\boldsymbol{x}_i|\Phi_j) \\ &= \sum_i \log \sum_h p(\boldsymbol{x}_i|\omega_{jh}, \Phi_j) P(\omega_{jh}) \end{aligned} \tag{14}$$

This does not have an analytical solution but an iterative procedure exists based on the Expectation-Maximization (EM) algorithm [11, 40]. In the Expectation (E) step of the algorithm, using the current set of parameters, we compute the probability that the sample $\boldsymbol{x}_i$ is generated by component h of class j:

$$P(\omega_{jh}|\boldsymbol{x}_i, \Phi_j) = \frac{p(\boldsymbol{x}_i|\omega_{jh}, \Phi_j)P(\omega_{jh})}{\sum_l p(\boldsymbol{x}_i|\omega_{jl}, \Phi_j)P(\omega_{jl})} \equiv \tau_{jhi} \tag{15}$$

Assuming Gaussian components, i.e., $p(\boldsymbol{x}|\omega_{jh}, C_j) \sim \mathcal{N}(\mu_{jh}, \Sigma_{jh})$, we have:

$$\tau_{jhi} = \frac{P(\omega_{jh})|\boldsymbol{\Sigma}_{jh}|^{-1/2} \exp\left[-(1/2)(\boldsymbol{x}_i - \boldsymbol{\mu}_{jh})^T \boldsymbol{\Sigma}_{jh}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_{jh})\right]}{\sum_l P(\omega_{jl})|\boldsymbol{\Sigma}_{jl}|^{-1/2} \exp\left[-(1/2)(\boldsymbol{x}_i - \boldsymbol{\mu}_{jl})^T \boldsymbol{\Sigma}_{jl}^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_{jl})\right]} \tag{16}$$

In the M-step, we update the component parameters Φ_j based on the probabilities computed in the E-step:

$$\begin{aligned}\hat{P}(\omega_{jh}) &= (1/n_j)\sum_i \tau_{jhi} \\ \boldsymbol{\mu}_{jh} &= \frac{\sum_i \tau_{jhi}\boldsymbol{x}_i}{\sum_i \tau_{jhi}} \\ \Sigma_{jh} &= \frac{\sum_i \tau_{jhi}(\boldsymbol{x}_i - \boldsymbol{\mu}_{jh})(\boldsymbol{x}_i - \boldsymbol{\mu}_{jh})^T}{\sum_i \tau_{jhi}} \end{aligned} \tag{17}$$

Like in the parametric approaches, simplifying assumptions can be made on the covariance matrices for regularization. Assuming equal hyperspheric densities, one uses (squared) Euclidean distance in computing the posteriors. If we merely compute the Euclidean distance $||\boldsymbol{x}_i - \boldsymbol{\mu}_{jh}||^2$, find the nearest mean $\boldsymbol{\mu}_{jc}$ nearest to $\boldsymbol{x}$ and set its τ to one and all others to zero, and only update $\boldsymbol{\mu}_{jc}$ for that example, we get the k-means procedure. *Online* version of the same algorithm updates the mean after each pattern. For each pattern $\boldsymbol{x} \in C_j$, we find $\boldsymbol{\mu}_{jc}$ such that

$$||\boldsymbol{x} - \boldsymbol{\mu}_{jc}|| = \min_h ||\boldsymbol{x} - \boldsymbol{\mu}_{jh}|| \tag{18}$$

and then do the update immediately:

$$\Delta\boldsymbol{\mu}_{jc} = \eta(\boldsymbol{x} - \boldsymbol{\mu}_{jc}) \tag{19}$$

where η is a learning factor that is gradually decreased towards zero for convergence. The rationale of this method is that by moving the mean closer to the sample we increase the likelihood of seeing that sample.

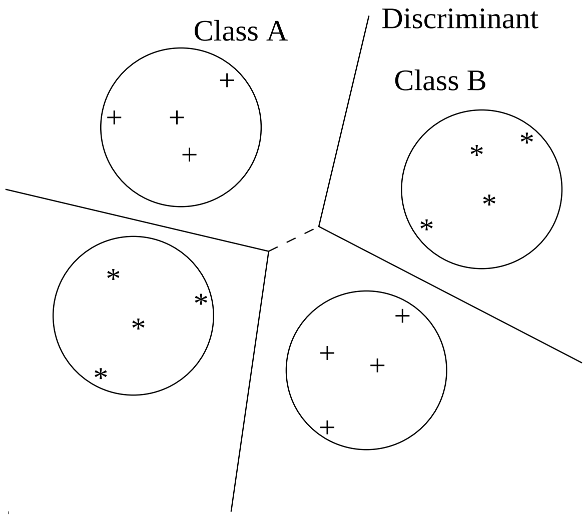

Figure 6: Example two class problem with two reference vectors per class and the arbitrary discriminant found by LVQ. The dotted lines show the Voronoi tesselation.

4.3.1 Learning Vector Quantization

Kohonen [24] proposed Learning Vector Quantization which also moves means of wrong classes away. For a given input $\boldsymbol{x}$, we find the closest mean $\boldsymbol{\mu}_{j\,c}$ among all classes (Fig. 6):

$$||\boldsymbol{x} - \boldsymbol{\mu}_{j\,c}|| = \min_{k,h} ||\boldsymbol{x} - \boldsymbol{\mu}_{k\,h}||$$

and then we move the mean towards the input if the classes of the mean and the input agree and we moved the mean away from the input if they disagree:

$$\Delta \boldsymbol{\mu}_{j\,c} = \begin{cases} +\eta(\boldsymbol{x} - \boldsymbol{\mu}_{j\,c}) & \text{if } \boldsymbol{x} \in C_j \\ -\eta(\boldsymbol{x} - \boldsymbol{\mu}_{j\,c}) & \text{otherwise} \end{cases} \tag{20}$$

5 Neural Classifiers

An artificial neural network is a network of simple processing units that are interconnected through weighted connections [24, 21, 42]. The interconnection topology between the units and the weights of the connections define the operation of the network. We are generally interested in feedforward networks where a set of units are designated as the input units through which input features are fed to the network. There is then a layer of *hidden* units that extract features from the input. This is followed by the layer of output units where in classification each output corresponds to one class.

There are a number of advantages to using neural network type classifiers for pattern recognition [7]:

1. They can learn, i.e., given a sufficiently large labelled training set, the parameters can be computed to optimize a given error criterion.

2. They can generate any kind of nonlinear function of the input.

3. Because they are capable of incorporating multiple constraints and finding optimal combinations of constraints for classification, features do not need to be treated as independent. More

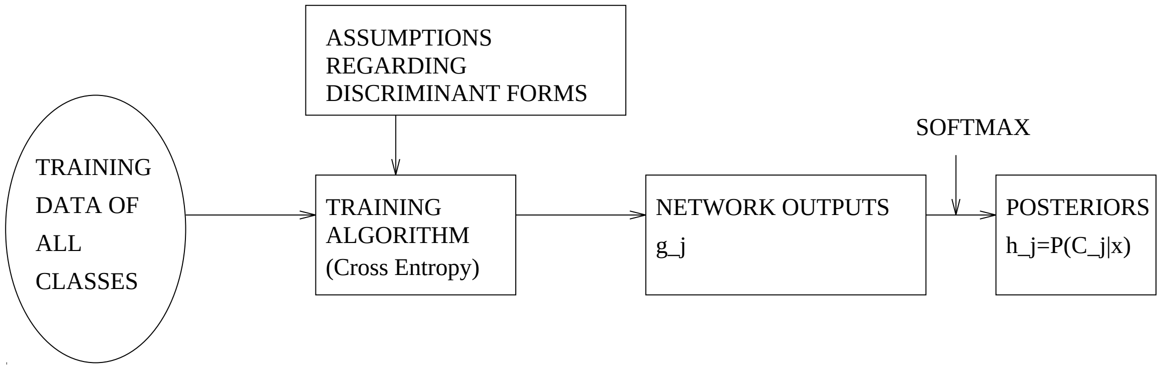

Figure 7: Building a neural classifier.

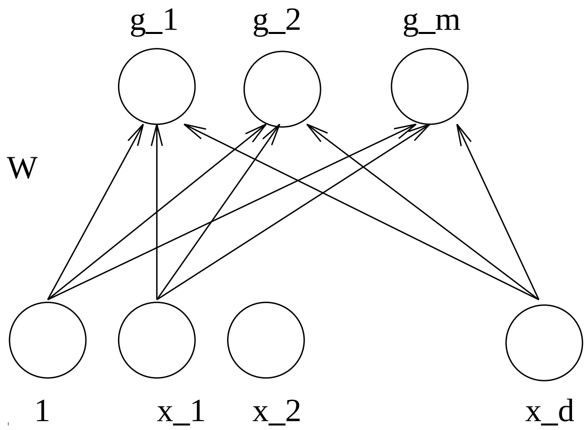

Figure 8: A linear classifier realized as a perceptron neural network.

generally, there is no need for strong assumptions about the statistical distributions of the input features (as is usually required in Bayes classifiers).

4. Artificial neural networks are highly parallel and regular structures which makes them especially amenable to high-performance parallel architectures and hardware implementations.

Statistical pattern recognition assumes a certain model for the densities and using Bayes decision rule, we see what type of discriminant functions they lead to. The neural approach is to assume a certain model for the discriminants (posteriors) directly, as defined by the network operation. For simplicity $g_j(\boldsymbol{x})$ can be assumed to be linear in $\boldsymbol{x}$:

$$g_j(\boldsymbol{x}) = \boldsymbol{W}^T \boldsymbol{x}' = \sum_{k=1}^{d} W_{jk} x_k + W_{j0} \tag{21}$$

where $\boldsymbol{x}'$ is $(\boldsymbol{x}, 1)^T$, augmented to include also an intercept (or bias) term. This is a neural network called a *perceptron* where units in the input layer take the value $\boldsymbol{x}$ and units in the output layer take $g_j(\boldsymbol{x})$. The weights of the connections between are $\boldsymbol{W}$ (Fig. 8).

Discriminants in real life are rarely linear so one way to approximate nonlinear functions is by estimating them as a linear sum of a number of nonlinear basis functions (Fig. 9):

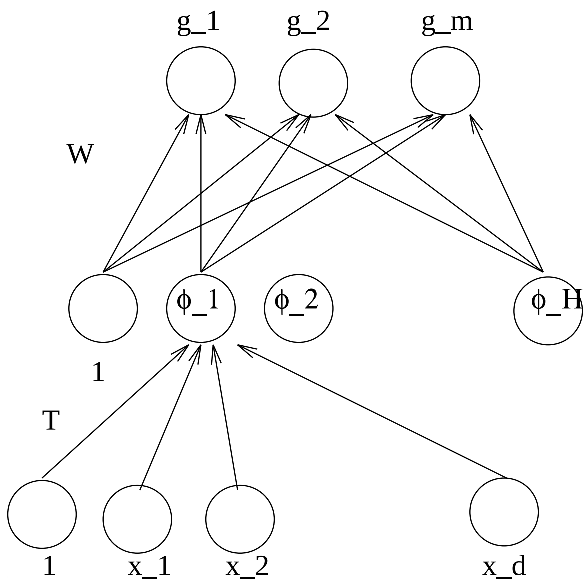

Figure 9: A multilayer neural network.

$$g_j(\boldsymbol{x}) = \sum_{h=1}^{H} W_{jh}\phi_h(\boldsymbol{x}) + W_{j0} \tag{22}$$

In neural network terminology, the basis functions, $\phi_h(\cdot)$, are called the hidden units and if the basis function is Gaussian this approach is called a *radial basis function network* and is called a *multilayer perceptron* if it is a sigmoid. The well known statistical technique of *projection pursuit regression* has the difference that basis functions need not be fixed identical but are estimated in a nonparametric manner.

In classification, we know that outputs are probabilities and that they sum up to one. This can be enforced using the softmax model [42]:

$$h_j = \frac{\exp g_j}{\sum_k \exp g_k} \tag{23}$$

and the error measure to be minimized is the cross-entropy between the two distributions [31, 42]:

$$E = -\sum_i \sum_j r_{ij} \log h_j(\boldsymbol{x}_i) \tag{24}$$

where $r_{ij} = 1$ if $\boldsymbol{x}_i \in C_j$ and 0 otherwise. $\boldsymbol{W}_j, j = 1, \ldots, m$ can be optimized using gradient-descent:

$$\Delta W_{jk} = -\eta \frac{\partial E}{\partial W_{jk}} \tag{25}$$

Internal parameters of the basis functions, i.e., weights from the input layer to the hidden layer, can also be trained similarly if $\phi(\cdot)$ is differentiable. This technique is called *backpropagation* of errors [21].

Note that because of the dependence introduced through softmax, a given sample is used to train the discriminants of all classes. This opposes to the statistical approach where a sample affects the likelihood of one class only.

5.1 Simple Perceptrons

A perceptron as defined in Eq. (21) defines a linear discriminant and works if samples from a class can be separated linearly from samples of all other classes where $\boldsymbol{W}_j$ defines the position and orientation of the separating hyperplane. This model is attractive due to a number of reasons: First, it is optimal when classes are normal and share a common covariance matrix. It has a small number of parameters thus does not require large amounts of memory and it is simple to implement.

5.2 Multilayer Perceptrons

In a multilayer perceptron as defined in Eq. (22), there is also a hidden layer whose units correspond to the basis functions, $\phi_h(\cdot)$. They extract nonlinear input combinations to be able to define nonlinear discriminants [21]. Usually, the hidden units implement perceptrons passed through the sigmoid function:

$$\phi_h(\boldsymbol{x}) = \phi(T_h^T \boldsymbol{x}') = \frac{1}{1 + \exp[-T_h^T \boldsymbol{x}']} \tag{26}$$

Connection weights of both layers, $\boldsymbol{T}$ and $\boldsymbol{W}$, are trained in a supervised manner by gradient descent over a cost function like the cross-entropy.

It has been shown [14, 22] that this type of a neural network is a universal approximator, i.e., can approximate any continuous function with desired accuracy. It has also been shown [43] that in the large sample case, multilayer perceptrons estimate posterior probabilities thus building a link between multilayer networks and statistical classifiers. These theorems do not tell how many hidden units are necessary, so one should test several alternatives on a cross-validation set and choose the best.

5.3 Radial Basis Functions

A radial basis function network (RBF) [32, 39] is another type of feedforward, multilayer network where the basis function is a Gaussian:

$$\phi_h(\boldsymbol{x}) = \phi(||T_h - \boldsymbol{x}||) = \exp\left[-\frac{||T_h - \boldsymbol{x}||^2}{2\sigma_h^2}\right] \tag{27}$$

Sometimes $\phi(\cdot)$ are normalized to sum up to one. RBF is also a universal approximator. Unlike Parzen windows where we have one Gaussian for each sample, in RBF networks we have less. Means of Gaussians may be seen as reference vectors in vector quantization or components in mixture models. The difference being that in the latter cases, a reference vector or a component belongs to one class only whereas here, class discriminants are defined as a linear combination of them.

Training can be done in one of two ways: In the uncoupled version, originally proposed by Moody and Darken [32], the Gaussian centers are trained in an unsupervised manner, e.g., using k-means. σ, the spread of Gaussians, is computed as a factor of the average of inter-center distances. The second layer of $\boldsymbol{W}$ is a single layer perceptron and is trained using gradient-descent rule in a supervised manner. In the coupled version, all parameters are trained in a supervised manner together, using back-propagation.

Because units have local responses, only a small number of Gaussians are active for each input, thus one generally needs a lot more Gaussians than sigmoids, but learning is faster when only a few units need be updated for an input. The generalization ability of RBF can be extended by having the weight of each hidden unit, W_{jh}, not a scalar but a linear function of the input [4]. This corresponds to a piecewise linear approximation of the discriminant instead of a piecewise constant approximation.

6 Literature Survey

6.1 Optical Character Recognition

Optical character recognition is one of the most popular pattern recognition applications and many systems have been proposed in the past towards this aim; see the special journal issue edited by Pavlidis and Mori for a review [38]. This is because it is a significant application of evident economic utility and also because it is a test bed before more complicated visual pattern recognition applications are attempted.

One of the earliest neural network based system for handwritten character recognition is the neocognitron of Fukushima [13]. Significant amount of work on optical recognition of postal ZIP codes is done by a group at AT&T Bell Labs by Le Cun and others [10, 30]. The system uses a multilayered network with local connections and weight sharing trained with back-propagation for classification. This implements a hierarchical cone where simpler local features are extracted in parallel which combine to form higher-level, less local features and which finally define the digits. An extensive study of backpropagation for optical recognition of both handwritten letters and digits is given by Martin and Pittman [28].

Keeler, Martin and others at MCC work on combining segmentation and recognition in one integrated system [23, 29]. This is necessary if characters are touching in such a way that they cannot be segmented by a straightforward segmentation procedure.

Several comparitive studies have also been done; either by fixing the dataset and varying the methods or by also using a number of datasets. Guyon *et al.* [18] is an early reference where simple and multilayer perceptrons are compared with statistical distance-based classifiers like k-NN in recognizing handwritten digits for automatic reading of postal codes. A comparison of k-NN, multilayer perceptron and radial basis functions in recognizing handwritten digits is given by Lee [26]. A review of the task and several neural and conventional approaches is given by Senior [44]. Comparison of distance-based classifiers, single and multilayer perceptrons and radial basis function networks is given in [2].

For the task of optical handwritten character recognition, a significant step was the production of a CDROM (Special Database 3) by the National Institute of Standards and Technology (NIST) [16] which includes a large set of digitized character images and computer subroutines that process them. This allowed many researchers a common testbed of significant size and quality on which to compare their approaches. Many of the above-mentioned work use this dataset or its predecessor. It is available by writing to NIST. A comparison of four statistical and three neural network classifiers is given by Blue *et al.* [6] for optical character recognition and a similar task, fingerprint recognition (for which a similar CDROM was also made available by NIST). Researchers from NIST made several studies using this dataset and technical reports can be accessed over the internet.

Recently with the reduction of cost of computing power and memory, it has been possible to have multiple systems for the same task which are then combined to improve accuracy [47, 49]. One approach is to have parallel models and then take a vote. Another approach is to have models cascaded where simpler models are used to classify simpler images and complex methods are used to classify images of poorer quality.

6.2 Speech Recognition

In speech recognition, the input is dynamic, i.e., changes in time. Classifiers we have considered up to now are static, i.e., assume that the whole input feature vector is available for classification. To use a static classifier for a dynamic task, a *time delay approach* is used [51]. This uses an input layer with tapped delay lines and can be used if the input buffer is large enough to accommodate the longest possible sequence or if a resampling is done to normalize length. This basically is mapping time into space by having multiple copies of the input units.

If the classifier is to accept input vectors sequentially, the classifier should have some kind of internal state that is a function of the current input and the previous internal state. In the neural network terminology, these are named *recurrent* networks which oppose to feedforward networks by having also connections between units in the same layer or connections to units of a preceding layer [21]. For short sequences, a recurrent network can be converted into an equivalent feedforward network by *unfolding* it over time. This is another way of mapping time into spaces, the difference being that now copies of the whole network is done. In some recurrent architectures, a separate set of units is designated as *feedback units* containing the hidden or output values generated by the preceding input. In theory, the current state of the whole network will nonlinearly depend on a combination of the previous network state and the current input [7]. A comparison of different recurrent architectures and learning rules is given in [20].

Furui [15] discusses various methods for speech recognition. Lippmann [27] and Waibel and Hampshire [52] give two reviews on using neural networks for speech recognition. Early work used recurrent neural networks for representation of temporal context but after the introduction of time delay neural networks by Waibel at al. [51], feedforward networks are also used for phoneme recognition. Lee and Lippmann [25], and Ng and Lippmann [35] for the same two artificial and two speech tasks compare a large number of conventional and neural pattern classifiers. Comparison of distance-based classifiers, single and multilayer perceptrons and radial basis function networks for phoneme recognition is given in [19]. The recent book by Bourlard and Morgan [7] discuss in more detail neural approaches to speech classification. Currently the most efficient approach for speech recognition is accepted to be *Hidden Markov Models* (HMMs) [7]. An HMM models speech as a sequence of discrete stationary states with instantaneous transition between states. At any state, there is a stochastic output process that describes the probability of occurrence of some feature vectors and a stochastic state-transition matrix conditioned on the input. It is called "hidden" because the sequence of states is not directly observable but is apparent only from the observed sequence of events. Generally there is one HMM for every word and states correspond to phonemes, syllables or demi-syllables. HMMs are also used to recognize individual phonemes where states correspond to substructures. Bourlard and Morgan [7] give a detailed discussion of HMM models and their use in speech recognition. They also show [34] how HMMs and multilayer networks can be combined for continuous speech recognition where the network estimates the emission probabilities for HMMs. A recent reference on current speech recognition methodologies is [9].

7 Simulation Results

For optical character recognition (OCR) experiments, we used the set of programs recently made available by NIST [16] to generate a database on which to test the algorithms we discuss. 44 people have filled in forms which are scanned and processed to get 32×32 matrices of handwritten digits. These matrices are then low-pass filtered and undersampled to 8 by 8 to decrease dimensionality. Each element is in the range 0 to 16. These 44 forms are divided into two clusters randomly as 30 forms in one side and 14 forms in the other. From the first 30, we generated three sets: training set, cross-validation set, and the writer-dependent test set. The training set has 1,934 digits. The cross-validation set contains 946 digits and is used to choose the best set of parameters during training, e.g. number of basis functions, point to stop training, neighborhood size, etc. The writer-dependent test set is used for testing after training and has 943 digits. The remaining 14 forms have no relationship with those used in training and they constitute the writer-independent test set containing 1,797 digits. We make sure that all sets contain approximately equal number of examples from each class.

For /b,d,g,m,n,N/ speech phoneme recognition (SR) experiments, the database contains 5,240 Japanese isolated words and phrases. Two hundred samples for each class are taken from the even-numbered and odd-numbered words. 600 samples are used for training, 300 for cross-validation and 300 testing. Further 683 phrases are used only for testing after having trained on isolated words.

Table 1: Properties of the datasets used.

	Number of Features	Input Data Type	Number of Classes	Number of Training Examples	Number of Cross-validation Examples	Number of Test Examples	Number of Indep Test Examples
OCR	64	int: 0..16	10	1,934	946	943	1,797
SR	112	real:0..1	6	600	300	300	683

Table 2: Results on the two applications. Values reported are average and standard deviations of ten independent runs (when applicable). Values in parantheses for SR are improved results by allowing different variances for different features.

	OCR		SR	
Method	Test	Independent Test	Test	Phrases
Bayes	90.77	89.43	63.33 (82.00)	36.75 (58.86)
k-NN	97.56	97.61	87.67 (96.33)	62.52 (72.91)
Parzen	97.99	97.44	90.00 (95.67)	67.50 (72.91)
LVQ	96.48, 0.34	96.42, 0.32	83.43, 1.83 (92.73, 2.10)	61.93, 2.71 (67.57, 2.88)
SP	96.06, 0.44	93.85, 0.32	92.10, 0.75	56.41, 2.59
MLP	97.51, 0.41	94.78, 0.41	93.83, 0.82	64.86, 2.44
RBF	98.11, 0.17	95.41, 0.31	91.90, 1.51	57.13, 5.54 (60.38, 5.59)

As is known, the speaking style and speed of phrases differ from the isolated words. Phonemes are extracted from hand-labelled discrete word utterances and phrase utterances which have a sampling frequency of 12 KHz. 7 speech frames (each 10 ms) are used as input. For each 10 ms frame, 16 Mel-scaled FFT coefficients are computed as feature values. The final input fed to the classifier has 112 dimensions. Properties of the datasets are summarized in Table 1.

Results with various algorithms on the two sets are given in Table 2. For each dataset, the first column contains results on the test set that is generated in an identical manner with the training set, i.e., the same writers or the same articulation. The second column contains data that is taken from different writers or continuous speech phrases and thus are more natural than the first and it is actually the success in these columns that matters.

A visual analysis of the results is possible through Figures 10 and 11. The two axes are accuracy on the independent test set and phrases for OCR and SR respectively and memory requirement. We assumed each real valued parameter to require 32 bits of storage. Input features require four bits for OCR (a value in the range 0..16) and 16 bits for SR (32,768 discrete levels). The number of training epochs is also marked for each technique.

In SR, using different variances for different features leads to a big improvement whereas in OCR, it does not. This information is also used in k-NN, Parzen windows and LVQ where it improves accuracy; note the large difference in accuracy between the two percentages in the fourth column of Table 2. Knowing that variances differ with RBF, we allowed Gaussians to have different spreads along different dimensions. A similar method that can be used with any classifier and not only distance-based ones, is *z-normalization* where each feature value is normalized to have zero mean and unit variance. Note that this assumes that all samples for a feature are drawn from one unimodal normal distribution and thus may fail if this assumption does not hold. For example with the multilayer perceptron on SR, though success on the test set increases after z-normalization, success on phrases decreases; this shows that feature values for phrases obey a different distribution.

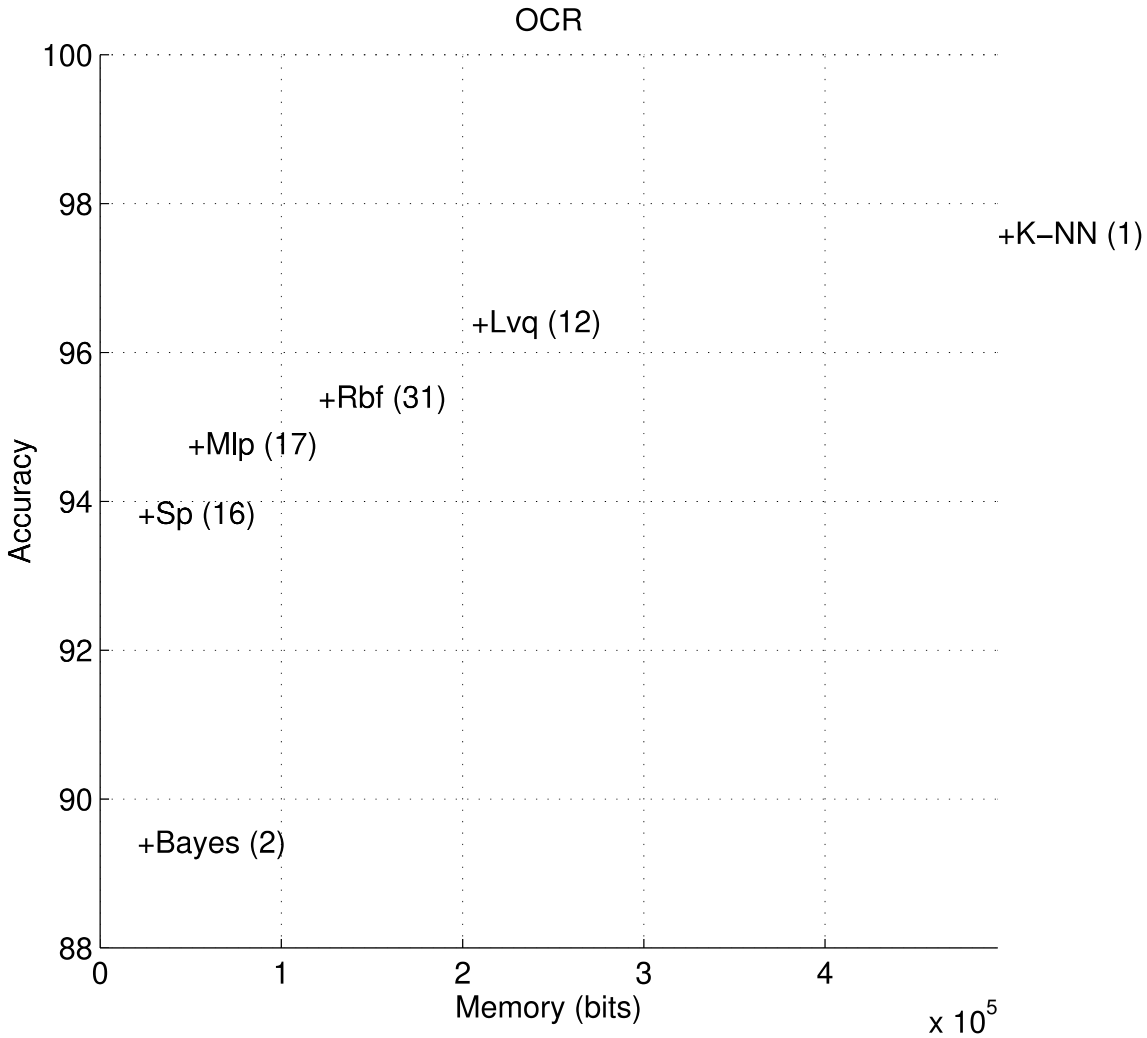

Figure 10: Results on the optical digit recognition dataset. Accuracy is percent correct classification on writer independent test set and memory is the number of bits required to store the free parameters of the system. Each pattern value (in the range 0..16) requires four bits and connection weights are assumed to require 32 bits. Values in parantheses are the number of epochs required for calculating the parameters.

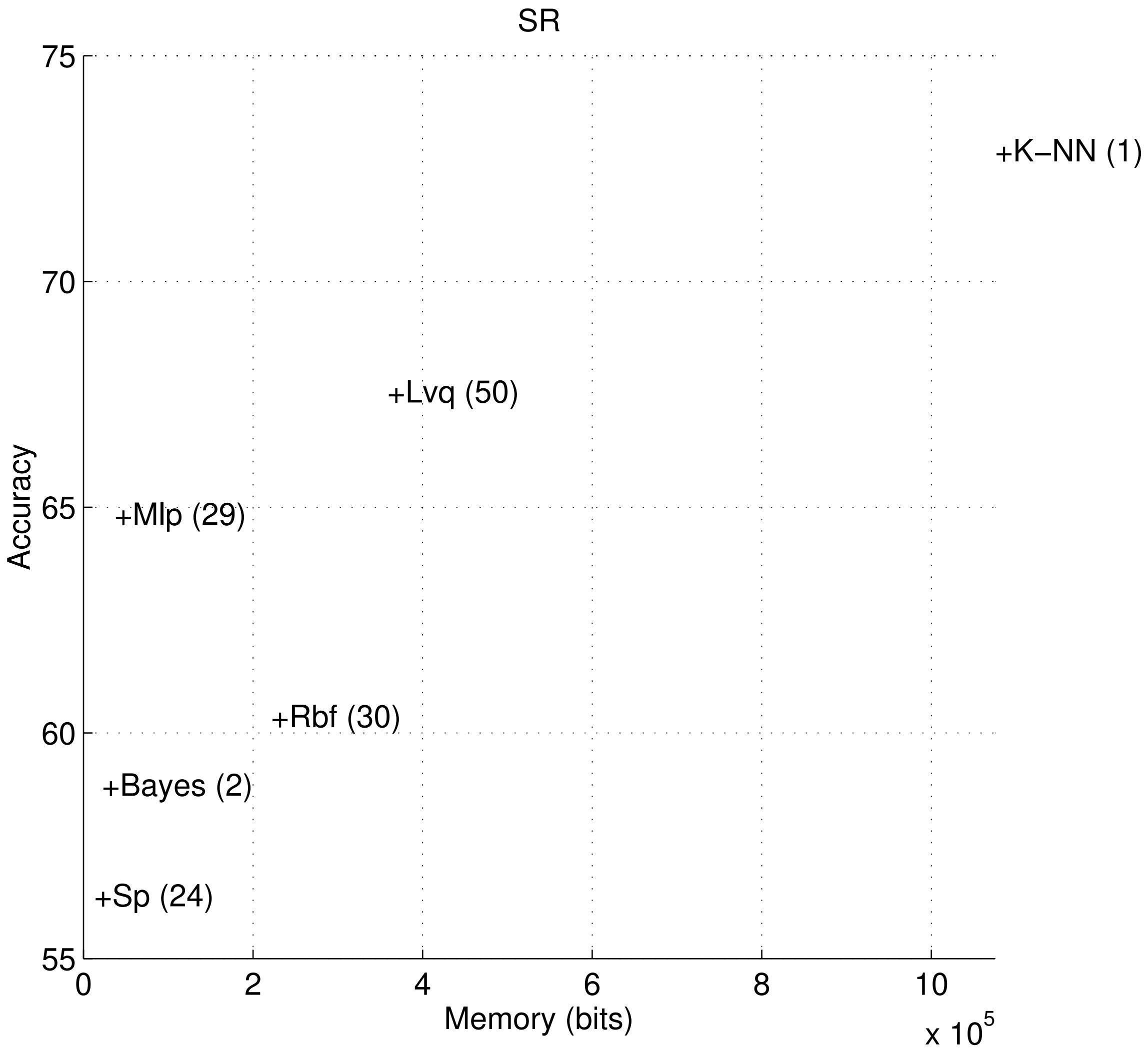

Figure 11: Results on phoneme recognition dataset. Accuracy is percent correct classification on phrase set and memory is the number of bits required to store the free parameters of the system. Each pattern value is assumed to require 16 bits and connection weights are assumed to require 32 bits. Values in parantheses are the number of epochs required for calculating the parameters.

In both applications, k-NN (or Parzen windows) has the highest accuracy. It is also the most expensive technique in terms of computation and memory. This however is no longer a serious drawback as the cost of storage and computation is getting cheaper. LVQ uses less storage by clustering data but has lower accuracy. RBF requires less storage than LVQ by sharing clusters between classes. A multilayer perceptron (MLP) generalizes better than a single layer perceptron (SP) indicating that discriminants are not linear. Parametric Bayes classifiers that assume independent features do not perform well indicating that the input features are highly correlated.

8 Conclusions

The similarity between statistical and neural techniques is greater than generally agreed. Many of the neural techniques are either identical or bear much resemblance to previously proposed statistical techniques. For a good discussion of neural networks from statisticians' point of view and vice versa, see the collection of articles in [8]. The recent interest in neural networks did much to revive interest in the old field of statistical pattern recognition [42].

Omohundro [37] discusses how nonparametric kernel estimators can be implemented as neural networks (by representing each sample with a Gaussian centered at the sample) and also discusses efficient data structures for the purpose. One example is the *probabilistic neural network* of Specht [46] which is a neural network implementation of Parzen windows. This approach is also known as *memory-based* as it can be seen as interpolating from a table of stored samples and is called *lazy* in machine learning literature as there is no learning process but the computation is deferred up until recognition is done.

Neural networks based on mixture models have also been proposed. Nowlan [36] considers them as "soft" variants of competitive approaches when used for vector quantization. Traven [50] proposes to use a mixture of Gaussians and calls this a "a neural network approach" and uses EM to optimize parameters without saying so.

Statistics can also be used to improve the performance of neural techniques. Analysis of variances and using a preprocessing like z-normalization or principal component analysis improve accuracy considerably in practice. The quality of the training sample is perhaps the most important factor as with an unrepresentative sample any statistic would be wrong.

Known statistical techniques like k-NN also provide a benchmark against which more complex approaches like multilayer perceptrons can be compared. Simple methods like k-NN generally perform quite well and much of the functionality of neural networks like parallel distributed processing can be obtained from such distance-based methods without requiring complicated computation, precise weights and lengthy error-minimization techniques.

We have also reached the conclusion that generally there is not one method that is significantly superior to all others in all respects of generalization accuracy, learning time, memory requirements and implementation complexity. The relative importances of these four factors differ from one application to another and thus in choosing one method, all of these should be taken into account and not only generalization accuracy as it has frequently been done in the past.

Acknowledgments

This work is supported by Grant EEEAG–143 from TÜBİTAK, the Turkish Scientific and Technical Research Council and Grant 95HA0108 from Boğaziçi University Research Funds. The OCR dataset has been collected and processed by C. Kaynak using programs made available by NIST. Thanks to F. Masulli and S. Furui for comments on this chapter.

References

[1] Alpaydın, E. (1994) "GAL: Networks that Grow when they Learn and Shrink when they Forget," *International Journal of Pattern Recognition and Artificial Intelligence*, **8**(1), 391–414.

[2] Alpaydın, E., Aratma, S., Yağcı, M. (1994) "Recognition of Handwritten Digits using Neural Networks," *ELEKTRİK, Turkish Journal of Electrical Engineering and Computer Sciences*, **2**(1), 20–31.

[3] Alpaydın, E., Gürgen, F. (1995) "Comparison of Kernel Estimators, Perceptrons and Radial-Basis Functions for OCR and Speech Classification," *Neural Computing and Applications*, **3**, 38–49.

[4] Alpaydın, E., Jordan, M. I. (1996) "Local Linear Perceptrons for Classification," *IEEE Transactions on Neural Networks*, **7**(3), 788–792.

[5] Berger, J. O. (1980) *Statistical Decision Theory and Bayesian Analysis*, 2nd edition, Springer.

[6] Blue, J. L., Candela, G. T., Grother, P. J., Chellappa, R., Wilson, C. L. (1994) "Evaluation of Pattern Classifiers for Fingerprint and OCR Applications," *Pattern Recognition*, **27**(4), 485–501.

[7] Bourlard, H. A., Morgan, N. (1994) *Connectionist Speech Recognition: A Hybrid Approach*, Kluwer Academic Publishers.

[8] Cherkassky, V., Friedman, J. H., Wechsler, H. (1994) *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, NATO ASI Series F, vol. 136, Springer.

[9] Lee, C.-H., Soong, F. K., Paliwal, K. K. (1996) *Automatic Speech and Speaker Recognition: Advanced Topics*, Kluwer Academic Publishers.

[10] le Cun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R. E, Hubbard, W., Jackel, L.D. (1990) "Handwritten Digit Recognition with a Back-Propagation Network," In Touretzky, D. (Ed.) *Advances in Neural Information Processing Systems 2*, 396–404, Morgan Kaufmann.

[11] Dempster, A. P., Laird, N. M., Rubin, D. B. (1977) "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of Royal Statistical Society B*, **39**, 1–38.

[12] Duda, R. O., Hart, P. E. (1973) *Pattern Classification and Scene Analysis*, Wiley.

[13] Fukushima, K. (1988) "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition," *Neural Networks*, **1**, 119–130.

[14] Funahashi, K. (1989) "On the Approximate Realization of Continuous Mapping by Neural Networks," *Neural Networks*, **2**, 183–192.

[15] Furui, S. (1989) *Digital Speech Processing, Synthesis and Recognition*, Marcel Dekker.

[16] Garris, M. D., J. L. Blue, G. T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet and C. L. Wilson (1994). *NIST Form-Based Handprint Recognition System*, NISTIR 5469, National Institute of Standards and Technology, Computer Systems Laboratory, Advanced Systems Division, Gaithersburg, MD, USA.

[17] Geman, S., Bienenstock, E., Doursat, R. (1992) "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, **4**, 1–58.

[18] Guyon, I., Poujoud, I., Personnaz, L., Dreyfus, G., Denker, J., le Cun, Y. (1989) "Comparing Different Neural Architectures for Classifying Handwritten Digits," International Joint Conference on Neural Networks 1989, vol. 2, 127–132, Washington USA.

[19] Gürgen, F., Alpaydın, R., Ünlüakın, U., Alpaydın, E. (1994) "Distributed and Local Neural Classifiers for Phoneme Recognition," *Pattern Recognition Letters*, **15**, 1111–1118.

[20] Gürgen, F., Şıhmanoğlu, M., Alpaydın, E. (1996) "Learning Speech Dynamics by Neural Networks with Delay Elements," *ICT'96, International Conference on Telecommunications*, B. Sankur (Ed.), 156–161.

[21] Hertz, J., Krogh, A., Palmer, R. (1991) *An Introduction to the Theory of Neural Computation*, Addison-Wesley.

[22] Hornik, K., Stinchcombe, M., White, H. (1989) "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, **2**, 359–366.

[23] Keeler, J., Rumelhart, D. E. (1992) "A Self-Organizing Integrated Segmentation and Recognition Neural Net," in J. E. Moody, S. J. Hanson, R. P. Lippmann (Eds.) *Advances in Neural Information Processing Systems 4*, 496–503, Morgan Kaufmann.

[24] Kohonen, T. (1988) *Self-Organization and Associative Memory*, Springer.

[25] Lee Y., Lippmann, R. (1990) "Practical Characteristics of Neural Network and Conventional Pattern Classifiers on Artificial and Speech Problems," in D. Touretzky (Ed.) *Advances in Neural Information Processing Systems 2*, 168–177, Morgan Kaufmann.

[26] Lee, Y. (1991) "Handwritten Digit Recognition Using K-Nearest-Neighbor, Radial-Basis Function, and Backpropagation Neural Networks," *Neural Computation*, **3**, 440–449.

[27] Lippmann, R. P. (1989) "Review of Neural Networks for Speech Recognition," *Neural Computation*, **1**, 1–38.

[28] Martin, G. L, Pittman, J. A. (1990) "Recognizing Hand-Printed Letters and Digits Using Backpropagation Learning," in D. Touretzky (Ed.) *Advances in Neural Information Processing Systems 2*, 405–414, Morgan Kaufmann.

[29] Martin, G. L., Rashid, M. (1992) "Recognizing Overlapping Hand-Printed Characters by Centered-Object Integrated Segmentation and Recognition," in J. E. Moody, S. J. Hanson, R. P. Lippmann (Eds.) *Advances in Neural Information Processing Systems 4*, 504–511.

[30] Matan, O., Baird, H.S., Bromley, J., Burges, C. J. C., Denker, J. S., Jackel, L. D., le Cun, Y., Pednault, E. P. D., Satterfield, W. D., Stenard, C. E., Thompson, T. J. (1992) "Reading Handwritten Digits: A Zip Code Recognition System," *IEEE Computer*, **25**(7), 59–62.

[31] McLachlan, G. J. (1992) *Discriminant Analysis and Statistical Pattern Recognition*, Wiley.

[32] Moody, J., Darken, C. J. (1989) "Fast Learning in Networks of Locally-Tuned Processing Units," *Neural Computation*, **1**, 281–294.

[33] Morgan, D. P., Christopher, L. S. (1991) *Neural Networks and Speech Processing*, Kluwer Academic Publishers.

[34] Morgan, N., Bourlard, H. (1995) "Continuous Speech Recognition: An Introduction to the Hybrid HMM/Connectionist approach," *IEEE Signal Processing Magazine*, 25-42.

[35] Ng, K, Lippmann, R. (1991) "Practical Characteristics of Neural Network and Conventional Pattern Classifiers," in R. Lippmann, J. Moody, D. Touretzky (Eds.), *Advances in Neural Information Processing Systems 3*, 970–976, Morgan Kaufmann.

[36] Nowlan, S.J. (1991) *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures*, PhD Thesis, School of Computer Science, Carnegie Mellon University.

[37] Omohundro, S. M. (1987) "Efficient Algorithms with Neural Network Behavior," *Complex Systems*, **1**, 273–347.

[38] Pavlidis, T., Mori, S. (1992) Special Issue on Optical Character Recognition, *Proceedings of the IEEE*, **80**(7).

[39] Poggio, T, Girosi, F. (1990) "Networks for Approximation and Learning," *Proceedings of the IEEE*, **78**, 1481–1497.

[40] Redner, R. A., Walker, H. F. (1984) "Mixture Densities, Maximum Likelihood and the EM Algorithm," *SIAM Review*, **26**(2), 195–239.

[41] Rabiner, L., Juang, B.-H. (1993) *Fundamentals of Speech Recognition*, Prentice-Hall.

[42] Ripley, B. D. (1994) "Neural networks and related methods for classification," *Journal of Royal Statistical Society B*, **56**, 409–456.

[43] Ruck, D. W., Rogers, S. K, Kabrisky, M., Oxley, M. E, Suter, B. W. (1990) "The Multi-layer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," *IEEE Transactions on Neural Networks*, **1**, 296–298.

[44] Senior, A. W. (1992) *Off-line Handwriting Recognition: A Review and Experiments*, Cambridge University Engineering Department, CUED/F-INFENG/TR 105.

[45] Silverman, B. W. (1986) *Density Estimation for Statistics and Data Analysis*, Chapman & Hall.

[46] Specht, D. F. (1990) "Probabilistic Neural Networks," *Neural Networks*, **3**, 109–118.

[47] Srihari, S. N. (1992). High-performance reading machines. *Proceedings of the IEEE* 80, 1120–1132.

[48] Streit, R. L., Luginbuhl, T. E. (1994) "Maximum Likelihood Training of Probabilistic Neural Networks," *IEEE Transactions on Neural Networks*, **5**(5), 764–783.

[49] Suen, C. Y., C. Nadal, R. Legault, T. A. Mai and L. Lam (1992). Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE* 80, 1162–1180.

[50] Traven, H. G.C. (1991) "A Neural Network Approach to Statistical Pattern Classification by 'Semiparametric' Estimation of Probability Density Functions," *IEEE Transactions on Neural Networks*, **2**(3), 366–377.

[51] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., Lang, K. J. (1989) "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**, 328–339.

[52] Waibel, A., Hampshire II, J. B. (1991) *Neural Network Applications to Speech*, in P. Antognetti & V. Milutinović (Eds). *Neural Networks: Concepts, Applications, and Implementations 1*, 54–76, Prentice Hall.