Small Sample Behavior of Multi-Layer Feedforward Network Classifiers: Theoretical and Practical Aspects

PROEFSCHRIFT

Ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie
aangewezen door het College van Dekanen
op dinsdag 30 november 1993 te 16.00 uur.

door

Martin Alfred Kraaijveld

geboren te Rotterdam,
elektrotechnisch ingenieur.

Delft University Press

Dit proefschrift is goedgekeurd door de promotor prof. dr. I.T. Young.

Dr. ir. R.P.W. Duin heeft als toegevoegd promotor in hoge mate bijgedragen aan het tot stand komen van dit proefschrift.

This work was partially sponsored by the Dutch Government as a part of the SPIN/FLAIR-DIAC project, and by the Foundation of Computer Science in the Netherlands (SION) with financial support from the Dutch Organization for Scientific Research (NWO).

ISBN 90-6275-934-3 / CIP

Printed in the Netherlands.

Aan mijn ouders.

Table of Contents

Introduction

In this thesis the small sample behavior of a relatively new type of pattern classifier is studied. These classifiers are called multi-layer feedforward network classifiers, and have become very popular tools for pattern recognition during the last decade. Despite the large amount of attention that these classifiers have attracted, relatively little is known about the consequences of training such classifiers with small training samples. This issue, therefore, serves as the subject of this thesis.

The thesis is split in two parts. In the first part, a number of theoretical issues on this topic are investigated. A review of the literature on small sample problems in pattern recognition is presented, followed by a discussion, in chapter two, of an elegant mathematical framework to study small sample problems. This framework is known as the Vapnik-Chervonenkis theory. The Vapnik-Chervonenkis theory is applied to the class of multi-layer feedforward network classifiers in chapter three, in order to characterize their small sample properties. Moreover, in chapter four, the influence of the (iterative) training procedure on the small sample behavior of the multi-layer network classifier is investigated. Finally, in chapter five, a number of conclusions are presented.

The second part of this thesis is concerned with a number of practical aspects of training a multi-layer feedforward network classifier. In the first place, the problem of the stopping criterion of an iterative learning procedure is studied in chapter six. Second, some issues on the learning speed of a class of iterative learning procedures are investigated in chapter seven.

Part 1

Theoretical Aspects

I

Small Sample Problems in Pattern Recognition

A well-known fact in pattern recognition, is that it is necessary to balance the training sample size with the complexity of the classification function. This is a relevant issue, since a too complex decision function may be sensitive to the noise of the training data, whereas a too simple classifier may not be capable of representing the required decision surface. As a large part of this thesis is dedicated to the small sample behavior of the class of multi-layer feedforward network classifiers, we will first review some general results on the relation between sample size and classifier complexity. Furthermore, we will formulate and motivate the questions that will be addressed in the rest of this thesis.

1.1. Introduction

An important class of tools in data analysis applications are systems that can learn to recognize patterns in data. Such a *pattern recognition system* is able to determine the type or class of an object, given a set of features of the object. These features are obtained by performing a number of measurements on the object, such as its length, its weight, its color, etc. A pattern recognition system is trained with a *training set* that consists of a number of examples of the various classes of objects. In the statistical pattern recognition approach, a training procedure infers a (mathematical) decision function from a set of features of the examples, that assigns a class label to every set of feature values. After the training phase of the system, the class label of a new and possibly unforeseen object is computed by the evaluation of this decision function, given the set of features of the object.

As an illustration, one could build a system that distinguishes between mice and elephants. Prior knowledge of this problem domain would suggest that the "weight" of the object can serve as a good feature to discriminate between the two classes. Thus, from a small set of examples, the system could then easily decide that all objects that are heavier than, say, 10 kg must be elephants, and all objects lighter than 10 kg must be

mice. The inference of this threshold 10 kg is easy, as it is not very critical in this case; a threshold of 1 kg or 100 kg would probably also do the job.

Things become more complicated if the only feature that is available is the color of the animal. As mice and elephants roughly have the same color, it might of course be possible to find a decision function that makes use of subtle differences in the color of mice and elephants. However, it is clear that such a system is likely to make much more errors than a system that is capable of determining the weight of the animal. Another problem is that the actual set of examples that is available in the training set may now become very critical. If the learning set would consist of an unfortunate subset of a-typical elephants and/or mice, with a slightly different color, the decision function that is inferred from the training set may largely be influenced by the a-typical examples, thereby deteriorating the performance of the system in classifying new objects.

Unfortunately, most pattern recognition problems that are encountered in practice do not allow much insight in the quality of the features that can be used, nor is it always clear that a given set of training data is sufficiently representative for the data. For instance, with optical character recognition it is even for humans very difficult to distinguish the handwritten character "O" from the number "0". Furthermore, training sets of handwritten characters may be sensitive to the education, mother-language, and age of the writers. In such problems a relevant question is: *how many*, and *which* features should the pattern recognition system use in inferring a decision function from the training data.

A naive line of thought is that the more features are used in the decision process, the more information is processed in making the decision, and the better the performance of the system will be. However, in most pattern recognition problems, it appears that instead of an increase in performance, the performance may seriously suffer from using many features. This problem is caused by the fact that the more features are used in inferring the decision function, the more parameters have to be determined from the training data. As there is only a finite amount of training data available, determining too many parameters will cause too much uncertainty in their values.

An illustration of this effect is shown in figure 1. For a recognition problem that is not further specified here, the figure depicts the expected classification error as a function of the number of features. The lowest curve in the figure shows the expected error for an infinitely large training set. Whereas the asymptotic error slowly converges to zero for an infinitely large sample size, the expected classification error has a clear optimal number of features for the case of a finite training sample size.

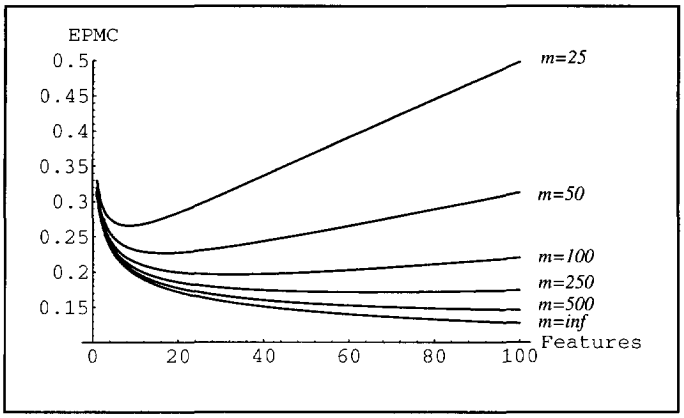


Figure 1: The expected probability of misclassification, or expected classification error, for an analytically tractable pattern recognition problem, as a function of the number of features. The lowest curve is the asymptotic error for an infinitely large sample size, the other curves correspond to a sample size of 500, 250, 100, 50 and 25 respectively. For a finite training sample size, the expected classification error has some optimal number of features. The figure was computed from equation (16) and table 2 in [Raudys 1991b].

The effect that is depicted in figure 1 is known as the *peaking phenomenon* ([Duda 1973], [Duin 1978], [Jain 1982]) or *Hughes-phenomenon* ([Hughes 1968], [Fukunaga 1990]) and can be considered as a fundamental result of the pattern recognition literature. The peaking phenomenon is a generic property in pattern recognition, that emerges in various situations and under various conditions. Another appearance of the peaking phenomenon, in a related problem domain, is depicted in figure 2. The problem is to find an interpolating function for a set of noisy points. In the figure, a least squares fit is used to determine the best polynomials of order 1, 2 and 10 for a given set of points. The figure illustrates that a too simple interpolator, as a linear function, is not flexible enough to serve as a good approximation for the underlying (quadratic) model. On the other hand, the tenth order polynomial is very sensitive to the noise in the points, and suffers from too much flexibility. Thus, although the tenth order polynomial is capable of fitting the data with zero approximation error, it does not yield the best interpolating function. The optimal interpolator, instead, is found in a function that is flexible enough to give a sufficiently good approximation of the set of points, but not so flexible that it totally

adapts to the noise. Clearly, this is yet another appearance of the peaking phenomenon, but now as a function of the "flexibility" of the interpolator that is fitted to the data.

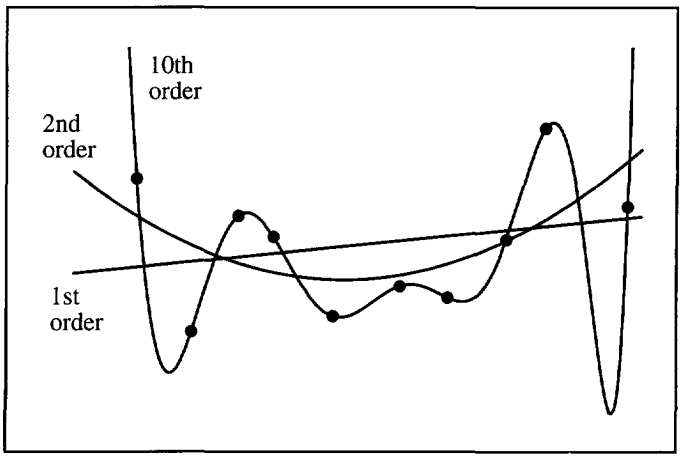


Figure 2: The noisy interpolation problem. For a given set of points, the best polynomial approximators of order 1, 2 and 10 were determined. As the points were generated by a quadratic function, the linear function is not flexible enough to approximate the data, whereas the tenth order polynomial is too flexible.

If we return to the domain of pattern recognition, the previous two examples illustrate that there is a need to balance the amount of training data with either the number of features, or with the "flexibility" of the decision function. One way to formulate this, is to state that there is some optimal *complexity* of a decision function for a given set of training data. As a measure of complexity, one could then use various parameters that characterize the complexity of the decision function, such as the number of features, the order of a polynomial decision function, the total number of free parameters of the decision function, etc. In chapter 2, we will present and discuss another measure of complexity that, in fact, generalizes these complexity measures. However, without making reference to a particular measure of complexity, we first depict the generic behavior of the peaking phenomenon as in figure 3. The figure shows the form of the plots of the true error of a classifier and the error on the training set, as a function of the classifier complexity. Another recognition problem, another training sample size or

another measure of complexity, may affect the height and the location of the optimal complexity, but not the global form of the plots.

In the following section 1.2 we will now review some well-known results on the peaking phenomenon. Then, in section 1.3, we will focus on the subject of this thesis: the small sample behavior of multi-layer feedforward network classifiers. In that section we will formulate the questions that are addressed in this thesis. Finally, an overview of the thesis is presented in section 1.4.

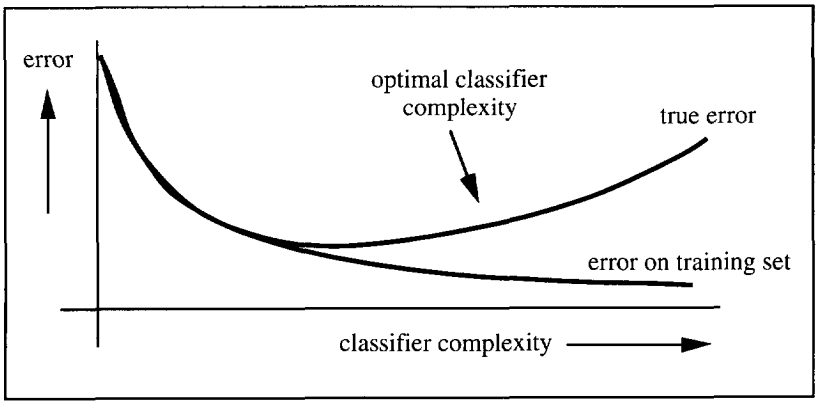


Figure 3: The peaking phenomenon. For a fixed training sample of a finite size, the best classifier is found for some finite value of the classifier complexity. The location of the optimum is a function of the underlying pattern recognition problem, the sample size and the measure of complexity that is used. Other pattern recognition problems, another sample size or another measure of complexity may affect the optimum, but will not affect the global form of the plots. Note that for a large classifier complexity, the error on the training set is a bad predictor for the true error of the classifier.

1.2. A Review of Previous Work

The peaking phenomenon in pattern recognition causes such counterintuitive effects as that a classifier that is based on a wrong, but simple, model of the underlying probability density functions, may outperform a classifier that is based on the right model. For example, when a multi-variate normal distribution is assumed, estimation of

the diagonal elements of the covariance matrix only, causes a large reduction of the number of parameters to be estimated. This may result in an increase of the classification performance of a pattern classifier [Duda 1973].

In the recent history, many authors have studied this problem either from an experimental or a theoretical viewpoint. In the following overview of some of these results, we follow Jain and Chandrasekaran [Jain 1982] in making a distinction between small sample problems in classification performance and small sample problems in error estimation. Before proceeding, however, two important remarks should be made. First, the following overview is written from a pattern recognition viewpoint. In related areas, such as regression, the peaking phenomenon has also been subject of intensive research, e.g. see the overview in [Geman 1992]. Second, the outcome of most of the theoretical work can be summed up by the general heuristic, that a reasonable classifier can be obtained, when the sample size is five to ten times larger than the number of parameters to be determined.

1.2.1. Small Sample Problems in Classification Performance

In statistical pattern recognition it is assumed that the measurement vector of an object is a realization of some underlying stochastic process. This stochastic process is characterized by its probability density function. The best performance that a pattern recognition system can achieve is obviously a function of the intrinsic overlap of the probability density functions of the classes. For example, consider a two-class problem, with equal prior probabilities and class conditional density functions $p_i(\boldsymbol{x})$ of the measurement vector $\mathbf{x}$ from class i, with $i \in \{0,1\}$. The classification error of the best possible classification function, i.e. the Bayes classifier, is given by the following equation:

$$\varepsilon^* = \frac{1}{2}\left(P\left\{ \log\frac{p_0(\boldsymbol{x})}{p_1(\boldsymbol{x})} > 0 \mid \boldsymbol{x} \in class\ 1 \right\} + P\left\{ \log\frac{p_1(\boldsymbol{x})}{p_0(\boldsymbol{x})} > 0 \mid \boldsymbol{x} \in class\ 0 \right\} \right) \quad (1)$$

The error of the Bayes classifier ε^* is equivalent to the intrinsic overlap of the distributions of the two classes. Note that this intrinsic overlap will never become larger by adding a new feature; it will either decrease or remain equal. The Bayes error ε^* is, therefore, a monotonically non-increasing function of the number of features.

A common approach in pattern recognition is to estimate the underlying probability density functions $p_i(\boldsymbol{x})$, and to use the estimated densities $\hat{p}_i(\boldsymbol{x})$ for the computation of

the decision function. As the estimated densities are not exactly equal to the true densities, either through erroneous assumptions on the parametric form of the densities, or through the variance in the estimated parameters, the true error of the classifier will in general be higher than the Bayes error (1). A theoretical result, that bounds the expected error of a classifier without making assumptions to the (parametric) form of the underlying densities is due to Duin ([Duin 1976], [Duin 1978]). Duin bounds the expected error of a classifier in terms of the Bayes error ε^*, and the Kolmogorov distance of the true and estimated densities:

$$E\{\varepsilon\} \leq \varepsilon^* + E\{e_o + e_1\} \tag{2}$$

with:

$$e_i = \frac{1}{2}\int |\hat{p}_i(\boldsymbol{x}) - p_i(\boldsymbol{x})|\, d\boldsymbol{x} = 1 - \int \min(\hat{p}_i(\boldsymbol{x}), p_i(\boldsymbol{x}))\, d\boldsymbol{x} \tag{3}$$

An experimental evaluation of this result, for the case of normal densities, showed that the expected Kolmogorov distance increases with increasing dimensionality of the feature space. This finding essentially characterizes the peaking phenomenon: after a certain point the extra discriminatory power of the added features is overcome by the deterioration in the estimates of the densities [Jain 1982].

Most of the other results in the pattern recognition literature are based on various assumptions on the form of the underlying probability densities in a recognition problem, most notably classes with multi-variate normal distributions. A number of interesting results in this specific context are the following:

- A result that provides considerable insight in the peaking phenomenon, is for the special case that the classes have normal distributions with different mean vectors $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$ and common covariance matrix $\boldsymbol{\Sigma}$. An important parameter that characterizes the distance between the two classes in this case, is the Mahalanobis distance Δ^2:

$$\Delta^2 = (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1) \tag{4}$$

The larger the Mahalanobis distance is, the lower the Bayes error of the problem. As in practical situations, the mean vectors and the covariance matrix are not known, they are

replaced by their unbiased estimates $\hat{\boldsymbol{\mu}}_0$, $\hat{\boldsymbol{\mu}}_1$ and $\hat{\boldsymbol{\Sigma}}$. In [Jain 1978] and [Jain 1982] it is shown that peaking can be prevented if adding a new feature results in a sufficiently large increase of the Mahalanobis distance. In fact, for a recognition problem with d features and sample size m, adding and extra feature should result in an increase $\delta\Delta^2$ of the Mahalanobis distance of at least:

$$\delta\Delta^2 > \frac{\Delta^2}{(m-3-d)} \tag{5}$$

Thus, peaking can be prevented if the newly added feature adds a sufficient amount of discriminatory power. Adding features that are not good enough, i.e. for which condition (5) is not satisfied, will result in a decrease of the expected performance. Two interesting related results, though in a somewhat different context, are presented by Young [Young 1978] and van Otterloo and Young [Otterloo 1978].

• An experimental result, for the case of multi-variate normally distributed classes with different mean vectors and identity covariance matrix, is reported by Raudys ([Raudys 1976], [Jain 1982]). Raudys presents a number of tables that show the relation between training sample size, the dimensionality of the problem, the expected classification error and the complexity of the classifier. The conclusion that follows from the experiments is that the training sample size should scale linearly with the dimensionality, for the case of linear decision functions, whereas for quadratic decision functions the sample size should scale quadratically. In other words, the sample size should scale linearly in the number of free parameters of the decision function.

• In a later paper, Raudys and Jain discuss a number of theoretical results that give rise to the same conclusions [Raudys 1991b]. They show that for linear decision functions and quadratic decision functions that are computed from the estimated mean vectors and covariance matrices, the expected classification error is given by:

$$E\{\hat{\varepsilon}\} \cong \varepsilon + \frac{v}{m} \tag{6}$$

Here, ε is the asymptotic error of the linear or quadratic classifier, and v is a constant that is determined by the problem. For quadratic classifiers, however, v is proportional to d^2, whereas for linear classifiers v is proportional to d. This again confirms the linear relation between the requirements on the sample size and the number of free parameters in

the decision function. Various bounds on ν, for various assumptions on the underlying normal densities, are presented in [Raudys 1991b] and [Fukunaga 1990]. Fukunaga also describes a method to obtain an empirical estimate of ν and ε for problems that are encountered in practice.

The message of the previous results is that, at least in the context of linear and quadratic classifiers that are trained with multi-variate normal data, the requirements on the sample size are proportional to the number of parameters that have to be determined. More information on various aspects of the small sample behavior of the classification performance can be found in all standard text books on pattern recognition, e.g. see [Duda 1973], [Devijver 1982] and [Fukunaga 1990].

1.2.2. Small Sample Problems in Error Estimation

Another domain in pattern recognition where the peaking phenomenon emerges, is in the estimation of the error of a classifier. This is especially relevant for parametric classifiers that are designed by a procedure that optimizes (a function of) the error on the training set. An example of such classifiers is the class of multi-layer feedforward network classifiers, that will be considered in chapter 3 and chapter 4.

In general, the error of a classifier on the training set is binomially distributed. Thus, when the error frequency of a fixed classification function on the training set is equal to $\hat{\varepsilon}$, a confidence interval on the binomial distribution can be used to determine a parameter χ, such that the true error is bounded, in probability, as $\hat{\varepsilon} - \chi \leq \varepsilon \leq \hat{\varepsilon} + \chi$. An elegant way to obtain such a confidence interval is by Hoeffding's inequality that will be discussed in chapter 2.

Things become more complicated, when not a single classifier is considered, but a set of classifiers. For example, a hypothetical learning procedure could estimate the error on the training set for all linear classifiers in $\Re^d$, and then return the classifier with the lowest empirical error. It is very well known that, if the sample size is too small, such a procedure may yield a classifier that has a very large difference between its empirical error and its true error. In fact, it is possible to show that for a training sample size that is equal to $d + 1$, such a procedure will always yield an empirical performance of 100%, even if the two classes have identical underlying probability density functions ([Cover 1965], [Foley 1972], [Duda 1973], [Devijver 1982], [Jain 1982]), see figure 4. Thus, for such a small training sample size, a linear classifier can separate a class from itself, which implies that the estimated performance of 100% can not serve as a good estimate of the true error. As we will be very much concerned with this issue in the rest of this thesis, we

will denote the performance of a classifier on the training set as *significant*, when the deviation to its true error is small. Analogously, the performance on the training set is called insignificant, when the deviation from the true error is large.

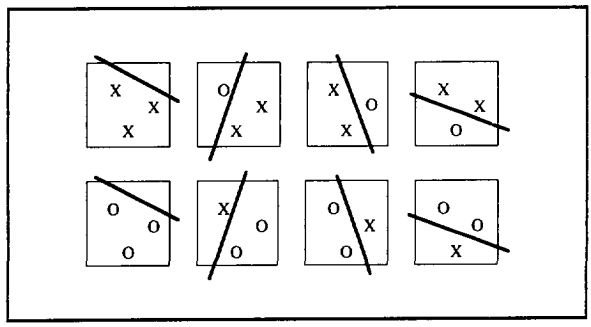


Figure 4: A linear classifier in $\Re^2$ can always yield a zero error on a training set of size 3 or smaller, independently of the class labels of the training data and the underlying probability distributions, except for a set of measure 0. In such case, the error on the training set may not be a good estimate of the true error of the classifier.

In this context, a quantity of considerable interest, is the *maximum deviation* of the true and estimated errors over the set of classifiers under consideration. When the maximum deviation of the true and estimated errors is small, the performance of all classifiers in the set is, according to the foregoing definitions, obviously significant. A formalization of this is the following. Consider a (possibly infinitely large) set of classifiers $F(X,\Theta)$. Every choice of the parameter vector Θ corresponds to a classification function $f(x, \theta) \in F(X,\Theta)$, that has a true error $\varepsilon(\theta)$ and an empirical error on the training set of $\hat{\varepsilon}(\theta)$. As the training data is stochastic, the quantity of interest can be formulated as:

$$P\left\{\sup_{\theta\in\Theta}\left|\varepsilon(\theta)-\hat{\varepsilon}(\theta)\right|>\chi\right\} \qquad (7)$$

The quantity (7) can be interpreted as the probability that a learning set is drawn, for which some classifier $f(x, \theta^*) \in F(X,\Theta)$ can be found, for which the deviation between $\varepsilon(\theta^*)$ and $\hat{\varepsilon}(\theta^*)$ is larger than χ. In chapter 2 we will present and discuss an elegant

mathematical framework, in which this quantity is bounded. This framework is known as the Vapnik-Chervonenkis theory and provides a total characterization of the problem of error estimation with small training samples. Furthermore, it appears that some of the foregoing relations between sample size and classifier complexity, that were either experimentally determined or were derived for some specific classes of underlying distributions, are generic properties in pattern recognition. They are generic in the sense that the Vapnik-Chervonenkis theory shows that they do not depend on the underlying probability density functions, nor on the specific parametric form of the decision function.

We will finish this section with a remark that is relevant for chapter 4. As was shown in figure 4, a linear classifier in $\Re^2$ can always yield a zero error on an arbitrary set of 3 points. As was stated above, an important implication of this, is that the empirical performance of 100% may be insignificant. However, it should be noted that it *may* be insignificant, but that this is not necessarily the case.

An example of such a situation is a very small training sample size in a recognition problem with very well separated classes. The decision functions that most pattern recognition algorithms will find in such problems, have indeed a significant performance. The theoretical prediction that some decision functions exist, for which the performance is not significant is (obviously) correct, but those decision functions are generally not found, see figure 5. The conclusion is that the issue of the significance of the empirical error of a decision function, is especially relevant for problems with overlapping classes. For problems with non-overlapping classes, therefore, the significance of the empirical error is considerably less relevant.

1.3. The Peaking Phenomenon and Multi-Layer Networks

A recent development in pattern recognition, is the use of so-called multi-layer feedforward network classifiers, e.g. see [Rumelhart 1986], [Anderson 1987], [ICNN 1990], [IJCNN 1991], [Sethi 1991], [ICPR 1992], [IEEECNN 1993], etc. As these pattern classifiers are partially inspired by biological models of the neuron, they are commonly referred to as multi-layer feedforward *neural* networks*.

* In this thesis we will not use the suggestive adjective "neural". Instead, we will write "connectionist models" for "neural networks", and we will write "multi-layer (feedforward) network classifiers" where others might use the words "multi-layer neural networks".

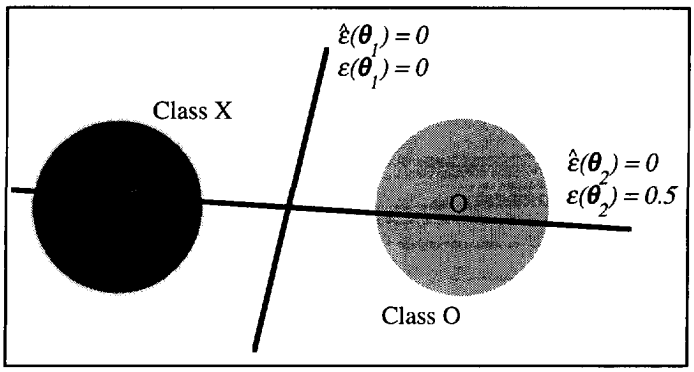


Figure 5: For well separated classes, a very small sample size may still yield a good and significant performance on the training set. For a training sample of size 2, classifier $f(\boldsymbol{x}, \boldsymbol{\theta}_1)$ has a significant performance on the training set, whereas $f(\boldsymbol{x}, \boldsymbol{\theta}_2)$ has a non-significant performance.

Multi-layer network classifiers belong to the class of parametric classifiers; i.e. the parametric form of the decision function is chosen beforehand, and during an iterative learning procedure the parameters are determined such that the network performs a certain desired mapping. The parametric form of the decision function, however, is rather specific and unconventional and may implement highly non-linear decision surfaces. In chapter 3, various of these parametric forms will be discussed, whereas chapter 4 is partly dedicated to some well-known learning procedures for such classifiers, most notably the backpropagation algorithm [Rumelhart 1986].

As for all other pattern classifiers, it is to be expected that multi-layer feedforward network classifiers suffer from the peaking phenomenon. However, as these classifiers are relatively new, this had not been as thoroughly investigated as for many other classifiers. A relevant issue, therefore, that serves as the main subject of this thesis, is the question on *how do multi-layer feedforward network classifiers behave in the context of small training samples.*

A strong motivation for this choice was given by the fact that the apparent linear relation that was found between the number of free parameters of a classifier and the requirements on the training sample size seems to be violated by many experiments in the literature on connectionist models. For example, in many experiments multi-layer network classifiers with tens of thousands of free parameters were trained with a sample

size of several tens or hundreds, e.g. see section 4.1 and table 4.1. The unexpected outcome of these experiments is that these classifiers showed a very good performance, both on the training set, as well as on an independent test set. A more specific formulation of the subject of this thesis is, therefore, the following:

The main question: *How can we characterize the relation between the complexity of a multi-layer feedforward network classifier and the requirements on the training sample size ?*

One of the claims (and outcomes) of this thesis is, that the specific properties of the (iterative) learning procedure for a multi-layer feedforward network classifier, necessitate a reconsideration of the relation between classifier complexity and sample size. For this reason, it makes sense to investigate the small sample behavior of these classifiers, either without making reference to any training procedure at all, or in the context of a specific training procedure. The main question of this thesis is therefore addressed by answering the following underlying and related questions:

a. What are good measures to characterize or quantify the complexity of a classifier?

b. Can we characterize, for such complexity measures, the relation between the complexity of a classifier and the requirements on the training sample size, without making reference to a particular learning procedure?

c. How can we determine such complexity measures for the class of multi-layer feedforward network classifiers, and how do they relate to the number of free parameters, the topology, the architecture and various other properties of such classifiers?

d. What is the influence of the learning procedure on the relation between the complexity of a classifier and the requirements on the training sample size, and is there some way to quantify this influence?

The rest of this thesis is dedicated to the answers to these questions. An overview is presented in the following section.

1.4. Overview

The structure of this thesis is based on the questions that were posed in the previous section. The subsequent chapters are dedicated to the following topics:

• First, an answer to questions a and b is given in chapter 2. This chapter is dedicated to the Vapnik-Chervonenkis theory and provides an elegant mathematical framework to study small sample problems. The issue of the characterization of the complexity of a classifier is resolved by the definition of a parameter that is called *the capacity* or *Vapnik-Chervonenkis dimension*. Within the context of the Vapnik-Chervonenkis theory, this parameter serves as a total characterization of the classifier complexity. Since the Vapnik-Chervonenkis theory is distribution free, and does not make reference to a particular learning algorithm, it serves as an ideal vehicle to study the problem raised by question b. The main contribution of this chapter is a slight improvement of a specific formulation of the Vapnik-Chervonenkis theory. Furthermore, the relations between the Vapnik-Chervonenkis theory and the relatively new approach to learning theory that is called "concept learning" will be discussed.

• Chapter 3 is dedicated to question c. The tools that are developed in chapter 2, will be used in this chapter to study the small sample behavior of number of connectionist models, i.e. multi-layer feedforward network classifiers. First, a thorough overview of the building blocks and parametric forms of these connectionist models will be given. Then for some of these models, the application of the Vapnik-Chervonenkis theory will provide an exact characterization of their small sample properties, independently of the training procedure. The connectionist models that will be considered are basically three variants of multi-layer networks: the general class of multi-layer networks, a class of networks that is related to the Parzen classifier, and a class of networks in which some of the parameters of decision function have been coupled, to lower the number of free parameters of the decision function.

• In chapter 4 question d will be addressed by investigating the influence of the learning procedure. First, a number of examples from the literature on connectionist models will be discussed, that illustrate that the learning procedure may have a considerable influence on the small sample behavior of a classifier. Then, for a well-known learning procedure, called the backpropagation algorithm, this will be verified experimentally. The next step is an effort to characterize the complexity of a classifier, not only as a function of the number of free parameters, but as a combined property of the

learning procedure and the number of free parameters. This characterization allows a better understanding of the influence of the learning procedure on the generalization behavior of a classifier. The last part of chapter 4 is dedicated to a method that provides a good stopping criterion for iterative learning procedures.

• In chapter 5, finally, we will return to the main question. We will review and comment on the results of the preceding chapters and it will be discussed to what extent and under what conditions they provide an answer to the main question. Furthermore, a number of remarks will be made on the usefulness of multi-layer network classifiers in pattern recognition applications.

II A Mathematical Framework for Generalization

In this chapter a mathematical framework will be discussed that quantifies and characterizes the relation between various parameters in the design of a parametric classifier. Among these are the training sample size, the complexity of the classification function and the probability of maximum deviation of the true error from the estimated error. The three theorems that are presented here are a slight improvement of some results of Vapnik [Vapnik 1982], and clearly show a number of generic properties in pattern recognition. Furthermore, a number of applications, implications and limitations of the theoretical framework are discussed.

2.1. The Vapnik-Chervonenkis theory

A standard problem in statistics is to estimate the probability of an event from a finite sample. The variance of the estimate, due to the limited size of the sample, induces a deviation of the estimated probability from the true probability of the event. A standard way to bound this deviation is with the help of Chebyshev's inequality (e.g. see [Mood 1974] or [Wozencraft 1965]), or with the more refined bound of Hoeffding:

Theorem 2.1 *[Hoeffding 1963]: Let* $\Xi^m = \xi_1, \ldots, \xi_m$ *be an i.i.d. sample of size m, with* $0 \le \xi_i \le 1$*, for* $i = 1, \ldots, m$*. Then the probability of deviation of the empirical mean of* Ξ^m *from its mathematical expectation is bounded by:*

$$P\left\{\left|\frac{1}{m}\sum_{i=1}^{m}\xi_i - E(\xi)\right| > \chi\right\} \le 2\exp\left(-2m\chi^2\right) \qquad (1)$$

In pattern recognition this result can be used to obtain an estimate of the probability of error of a classifier. Given a classifier $f(\boldsymbol{x}, \boldsymbol{\theta})$ with some fixed parameter vector $\boldsymbol{\theta}$ and error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ on a set of m samples, theorem 2.1 guarantees that the true error $\varepsilon(\boldsymbol{\theta})$ is bounded by $\hat{\varepsilon}(\boldsymbol{\theta}) - \chi < \varepsilon(\boldsymbol{\theta}) < \hat{\varepsilon}(\boldsymbol{\theta}) + \chi$ with probability at least $1 - 2\exp\left(-2m\chi^2\right)$.

In many applications, however, it is required that the deviation of the empirical error from the true error is not only bounded for a single classifier, but for a (potentially infinitely large) set of classifiers. This is a problem that is frequently encountered in pattern recognition [Devroye 1988].

In pattern recognition, the designer of a recognition system generally chooses a classifier from a larger set of classifiers F. This selection process is essentially the process of choosing a classifier with a sufficiently good performance on the training set, from the classifiers in F. The set F is chosen on the basis of prior knowledge and/or experience of the designer. When the underlying probability density functions of the training data are known, F obviously consists of the Bayes classifier only. In most applications, however, the densities are not known, and the choice for a particular set of classifiers F is a difficult problem. When we limit ourselves to the class of parametric pattern classifiers, the set F could for a practical problem consist of linear classifiers, third order polynomial classifiers and multi-layer network classifiers with at most N units and W weights. When the set is large, i.e. when it has many candidate classifiers to choose from, it is likely that one may find a classifier with good performance in the set. Making the set too large, however, may lead to the problem of over-fitting the training data, which may deteriorate the performance on an independent test set (e.g. see section 1.1 and section 1.2).

A solution to this problem of over-fitting is the requirement that for all classifiers in F simultaneously, the estimated error of a classifier is sufficiently close to its true error. Clearly, when this is the case for all classifiers in F simultaneously one has guaranteed estimates of their performance, and one can safely select the classifier with the *best* performance on the training data. Thus, a relevant subject of study in this context is the *uniform convergence* of the error frequencies to the error probabilities for the classifiers in F. Uniform convergence of frequencies to probabilities over F assures that F is "over-determined" by the training data. In this context, a natural measure of this over determination, is the *maximum deviation* of the true and estimated errors over F.

In the rest of this chapter, a framework will be presented that relates the probability of maximum deviation to various other parameters in the design of a parametric classifier. The framework is based on the work of the Russian mathematician Vapnik [Vapnik 1971a], [Vapnik 1971b], [Vapnik 1982], [Vapnik 1992]. It is closely related to the theorem of Glivenko and Cantelli on the convergence of a cumulative density function to an empirical cumulative density function (e.g. see [Mood 1974] or [Vapnik 1982]), and the Kolmogorov-Smirnov statistic that provides a bound on the rate of this convergence

(e.g. see [Mood 1974]). Furthermore, variations of the framework have been discussed and presented by Vapnik [Vapnik 1982], Devroye ([Devroye 1982] and [Devroye 1988]) and Blumer et al. ([Blumer 1986] and [Blumer 1989]). An excellent survey of applications in pattern recognition is found in [Devroye 1988]. Applications in concept learning in AI are described in [Blumer 1989], [Natarajan 1991] and in the proceedings of the workshops on computational learning theory ([COLT 1988], [COLT 1989] and [COLT 1990]). Applications of Vapnik's theory to the area of connectionist models are described by Baum ([Baum 1989]), Haussler ([Haussler 1991]), Burton ([Burton 1991]) and Gallant ([Gallant 1990a]).

2.1.1. Uniform Convergence over Finite Sets of Classifiers

Uniform convergence of the estimated error to the true error over a set of classifiers is easily derived for the case of *finite* sets of classifiers. This is illustrated by presenting a bound on the probability of *maximum deviation* over the set of classifiers, and a bound on the probability of *maximum one-sided relative deviation.*

2.1.1.1. *A Bound on the Probability of Deviation*

Consider a set $F(\boldsymbol{X},\boldsymbol{\Theta})$ of L classifiers $f(\boldsymbol{x}, \boldsymbol{\theta}_1)$, $f(\boldsymbol{x}, \boldsymbol{\theta}_2)$, ..., $f(\boldsymbol{x}, \boldsymbol{\theta}_L)$ with fixed parameter vector $\boldsymbol{\theta}_i$, for $1 \le i \le L$, and a random independent sample $\Xi^m = \xi_1, ..., \xi_m$ of size m. The error frequency of every $f(\boldsymbol{x}, \boldsymbol{\theta}_i) \in F(\boldsymbol{X},\boldsymbol{\Theta})$ on Ξ^m is denoted $\hat{\varepsilon}(\boldsymbol{\theta}_i)$ and its true error $\varepsilon(\boldsymbol{\theta}_i)$. According to Hoeffding's inequality (1), the probability of deviation of the true and estimated errors can be bounded for each $f(\boldsymbol{x}, \boldsymbol{\theta}_i) \in F(\boldsymbol{X},\boldsymbol{\Theta})$ separately. However, to bound the *maximum deviation* over $F(\boldsymbol{X},\boldsymbol{\Theta})$ we proceed as follows [Vapnik 1982], [Devroye 1988]:

$$P\left\{\sup_i \left|\hat{\varepsilon}(\boldsymbol{\theta}_i) - \varepsilon(\boldsymbol{\theta}_i)\right| > \chi\right\} \le \sum_{i=1}^{L} P\left\{\left|\hat{\varepsilon}(\boldsymbol{\theta}_i) - \varepsilon(\boldsymbol{\theta}_i)\right| > \chi\right\} \le 2L\exp\left(-2m\chi^2\right) \qquad (2)$$

Inequality (2) guarantees that for all $f(\boldsymbol{x}, \boldsymbol{\theta}_i) \in F(\boldsymbol{X},\boldsymbol{\Theta})$ the bound $\hat{\varepsilon}(\boldsymbol{\theta}_i) - \chi < \varepsilon(\boldsymbol{\theta}_i) < \hat{\varepsilon}(\boldsymbol{\theta}_i) + \chi$ is valid with probability at least $1 - 2L\exp\left(-2m\chi^2\right)$. By demanding that the probability of maximum deviation is at most δ, we can solve the parameter χ from:

$$2L\exp\left(-2m\chi^2\right) = \delta \qquad (3)$$

From (3) it can easily be derived that the following inequality is valid with probability at least $1 - \delta$ for all $f(\boldsymbol{x}, \boldsymbol{\theta}_i) \in F(X, \Theta)$ simultaneously:

$$\hat{\varepsilon}(\boldsymbol{\theta}_i) - \sqrt{\frac{\ln L - \ln\left(\frac{\delta}{2}\right)}{2m}} \leq \varepsilon(\boldsymbol{\theta}_i) \leq \hat{\varepsilon}(\boldsymbol{\theta}_i) + \sqrt{\frac{\ln L - \ln\left(\frac{\delta}{2}\right)}{2m}} \tag{4}$$

Also, it follows from (3) that if the maximum deviation should be less than χ with probability at least $1 - \delta$, the sample size m should be:

$$m \geq \frac{\ln L - \ln\left(\frac{\delta}{2}\right)}{2\chi^2} \tag{5}$$

Since the inequalities (4) are valid for all L classifiers simultaneously, they also hold for the classifier in $F(X, \Theta)$ that has the lowest empirical error $\hat{\varepsilon}(\boldsymbol{\theta}_{best})$. This provides a guaranteed bound for the probability of error of the empirically best classifier:

$$\hat{\varepsilon}(\boldsymbol{\theta}_{best}) - \sqrt{\frac{\ln L - \ln\left(\frac{\delta}{2}\right)}{2m}} \leq \varepsilon(\boldsymbol{\theta}_{best}) \leq \hat{\varepsilon}(\boldsymbol{\theta}_{best}) + \sqrt{\frac{\ln L - \ln\left(\frac{\delta}{2}\right)}{2m}} \tag{6}$$

Note that this inequality holds with probability at least $1 - \delta$.

2.1.1.2. *A Bound on the Probability of Relative Deviation*

The Hoeffding inequality (1) is tight for $E(\Xi) \approx 0.5$ [Hoeffding 1963]. For other values of $E(\Xi)$, however, the bound can be rather loose. A tighter formulation can be obtained by bounding the *one sided relative deviation* [Vapnik 1982, p. 148]:

$$P\left\{\frac{E(\Xi) - \frac{1}{m}\sum_{i=1}^{m}\xi_i}{\sqrt{E(\Xi)}} > \chi\right\} \leq \exp\left(-\frac{1}{2}m\chi^2\right) \tag{7}$$

For Bernoulli distributed Ξ with small $E(\Xi)$ the term $\sqrt{E(\Xi)}$ is approximately equal to the variance $\sqrt{E(\Xi)(1 - E(\Xi))}$.

In a pattern recognition context, a bound on the one sided deviation provides an upper bound for the true error, instead of an interval as in (4). Also, bounding the *relative* deviation in a pattern recognition problem makes sense, since one would allow a much smaller deviation for the case of small $E(\Xi)$ than for the case of large $E(\Xi)$. E.g. when $\hat{\varepsilon}(\boldsymbol{\theta}_i) \approx 1\%$, it would be reasonable to bound $\varepsilon(\boldsymbol{\theta}) \leq 2\%$, whereas for $\hat{\varepsilon}(\boldsymbol{\theta}_i) \approx 25\%$ a bound $\varepsilon(\boldsymbol{\theta}) \leq 30\%$ may be appropriate.

With a similar approach as in the previous paragraph, it is possible to guarantee the uniform convergence of the estimated error to the true error for a set of classifiers with a fixed and finite size. From (7) it can easily be derived that the following inequality holds with probability at least $1 - \delta$ for all L classifiers simultaneously [Vapnik 1982]:

$$\varepsilon(\boldsymbol{\theta}_i) \leq \hat{\varepsilon}(\boldsymbol{\theta}_i) + \frac{\ln L - \ln \delta}{m}\left(1 + \sqrt{\frac{2m\hat{\varepsilon}(\boldsymbol{\theta}_i)}{\ln L - \ln \delta}}\right) \tag{8}$$

Note that for small $\hat{\varepsilon}(\boldsymbol{\theta}_i)$ the deviation between $\varepsilon(\boldsymbol{\theta}_i)$ and $\hat{\varepsilon}(\boldsymbol{\theta}_i)$ is proportional to $(\ln L - \ln \delta)/m$, whereas for $\hat{\varepsilon}(\boldsymbol{\theta}_i) \approx 0.5$ the bound behaves like $\sqrt{(\ln L - \ln \delta)/m}$.

2.1.1.3. *An Application: Random Search in the Parameter Space of a Classifier*

An application of the previous theory is the following. Consider the training of a parametric classifier (e.g. a linear threshold function or a multi-layer feedforward network) on a learning set $\boldsymbol{\Xi}^m$ of size m, with a training method that is based on stochastic search. This search technique can be "blind", or based on more advanced techniques, like simulated annealing [Kirkpatrick 1983]. Every realization of the stochastic process in the parameter space $\boldsymbol{\Theta}$ of the classifier, generates a new classifier that is evaluated on $\boldsymbol{\Xi}^m$. L realizations of the stochastic process, therefore, generate L classifiers with empirical error $\hat{\varepsilon}(\boldsymbol{\theta}_i)$, with $1 \leq i \leq L$. The inequalities (2 - 8) show how the various parameters in this training procedure are interrelated; they provide an upper bound for L given m, χ and δ, they provide an upper bound for χ given L, m, and δ, etc.

Due to the fact that L appears in a logarithmic term in inequalities (6) and (8), the requirements for m only increase very slowly with L. A slight increase of m, therefore, allows a large increase of the duration of the search, i.e. a large increase of the number of classifiers to be considered.

Other applications in this context can be found in [Devroye 1988]. It should be noted that all results so far are of a combinatorial nature, and therefore independent of the underlying probability distributions.

2.1.2. Uniform Convergence over Infinite Sets of Classifiers

According to the theory of the previous paragraph, the uniform convergence of (relative) frequencies to probabilities can not be assured if the cardinality L of the set of classifiers F approaches infinity. For example, it is not possible to use inequalities (1) or (7) to bound the probability of maximum deviation for all linear classifiers in $\Re^d$, since the number of classifiers is obviously not bounded. For such cases, it is clear that a different approach is needed. In the following, such an approach will be considered. First, a key parameter in this new framework will be introduced and discussed, and then a number of theorems are presented that clarify the conditions for uniform convergence over infinitely large sets of classifiers.

2.1.2.1. *The Capacity*

As was shown in inequality (2) in section 2.1.1 the probability of maximum deviation of the true and estimated error for the case of a finite set of classifiers is bounded by $2L\exp\left(-2m\chi^2\right)$. In this formula, the exponential term is based on the Hoeffding inequality (1), whereas the cardinality L of the set of classifiers F serves as a characterization of the richness of F.

For L approaching infinity, however, the cardinality of the set can no longer serve as a characterization of the richness of the set of classifiers. Therefore, an alternative measure of "richness" is required in this case. In the following, *the number of dichotomies that the classifiers in F can induce on a sample of m points* will be adopted as such a measure. It is well known that the number of dichotomies is closely related to a parameter that is called *the capacity:*

Definition 2.1 *([Vapnik 1982], [Blumer 1989], [Devroye 1988]): The capacity of a set F of classifiers is defined as the maximum number of points that can be given an arbitrary Boolean label by the functions in F.*

The notion of capacity is closely related to the *capacity* of [Cover 1965], it is equal to the *index* minus one of [Devroye 1988], and it is called the *Vapnik-Chervonenkis dimension* in [Blumer 1989] and [Baum 1989]. The capacity is related to the number of dichotomies by the following two theorems:

Theorem 2.2 *([Vapnik 1982], [Blumer 1989]): The number of dichotomies $S^F(m)$ that can be induced with a set of classifiers F with finite capacity V on a set of m points is at most:*

$$S^F_{\max}(m) \begin{cases} \text{either} \equiv 2^m & \text{for } (m \le V) \\ \text{or} \le \sum_{i=0}^{V} \binom{m}{i} & \text{for } (m > V) \end{cases} \tag{9}$$

The bound for $m > V$ can be reformulated as follows:

Theorem 2.3 *([Sauer 1972], [Blumer 1989]): For $m > V$ the maximum number of dichotomies $S^F_{\max}(m)$ is bounded by:*

$$S^F_{\max}(m) \le \sum_{i=0}^{V} \binom{m}{i} \le \left(\frac{em}{V}\right)^V \tag{10}$$

Here e is the base of the natural logarithm, $e \approx 2.7183$. Note that according to these theorems, the capacity V marks the sample size where the number of dichotomies makes a transition from an exponential in m to a polynomial of order V in m.

The notion of the capacity and its relation to the number of dichotomies can be illustrated with the following simple example. Consider a set of three points in $\Re^2$. It can easily be verified that all three points can be arbitrarily labeled by a linear function; i.e. all $2^3 = 8$ dichotomies can be induced on the three points, see figure 1. A set of four points in $\Re^2$, however, can in general not be labeled arbitrarily by a linear classifier. As can be verified, of the possible $2^4 = 16$ dichotomies, only 14 can be induced, see figure 2. As it is impossible to find any set of four points in $\Re^2$ that can be labeled arbitrarily, the capacity of a linear function in $\Re^2$ is, according to the definition, equal to three.

The example shows an important property of the capacity. Basically it states how many samples it can label in an arbitrary way. This is indirectly related to the number of degrees of freedom of a classifier. If a second order polynomial were used instead of a linear classifier, the remaining two cases could have been labeled by a suitable second order curve. Therefore, the capacity of the set of second order polynomials is higher than the capacity of the set of linear functions.

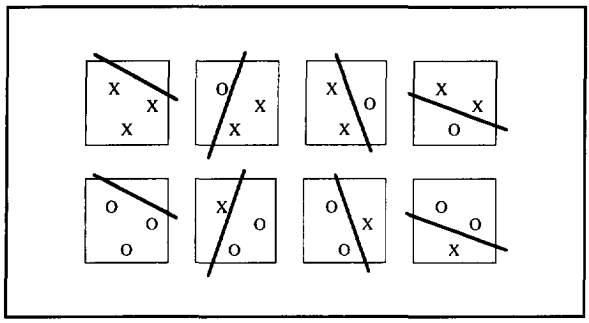


Figure 1: In this figure three (fixed) points in $\Re^2$ are shown. By changing the orientation and position of the linear function, the three points fall on different sides of the linear function and are thereby induce a different label on the points. Note that all $2^3 = 8$ possible labelings of the set of three points can be induced.

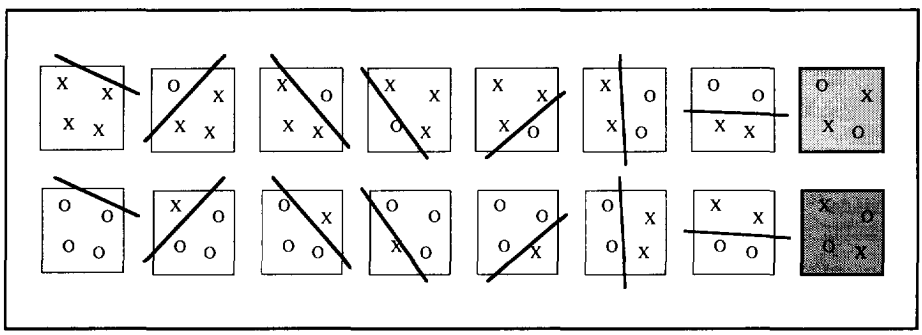


Figure 2: Four (fixed) points in $\Re^2$ can *not* be arbitrarily labeled by a linear function. The two remaining situations on the right are called to be non linearly separable.

According to definition 1 above, it is not necessary that *any* set of points can be arbitrarily labeled. If only one, maybe exceptional, combination of V points can be given an arbitrary label, the capacity is at least V. It is this detail in the definition that makes the derivation of the capacity of a set of functions a non-trivial problem (e.g. see [Wenocur 1981]). For a number of classifiers it is possible to show that the capacity is even larger than the number of degrees of freedom.

The advantage of this approach, however, is that the capacity is defined as a property of the classification function only, thereby being independent of the probability distribution of the points on which the dichotomies are induced. As will be shown in the following sections, this approach allows a distribution free bound on the rate of the uniform convergence of the true and estimated errors.

2.1.2.2. *Bounds on the Probability of Deviation*

In [Vapnik 1982] it is proven that the following requirement is a necessary and sufficient condition for the uniform convergence of frequencies to probabilities:

$$\lim_{m\to\infty} \frac{\ln\left(S^F_{\exp}(m)\right)}{m} = 0 \tag{11}$$

Here $S^F_{\exp}(m)$ is the *expected* number of dichotomies that the classifiers in F can induce on an i.i.d. set of m points. The expected number of dichotomies is obviously lower than the *maximum* number of dichotomies. Therefore, if the following equality is valid:

$$\lim_{m\to\infty} \frac{\ln\left(S^F_{\max}(m)\right)}{m} = 0 \tag{12}$$

equality (11) is certainly satisfied. As was discussed above, the maximum number of dichotomies is a distribution-free quantity that is associated with the classification function instead of the data. This is used in the following theorem, that provides a distribution-free bound on the rate of uniform convergence of frequencies to probabilities:

Theorem 2.4 *[Vapnik 1982]: Let $F(\boldsymbol{X},\boldsymbol{\Theta})$ be a set of classification functions with finite capacity V and let $\boldsymbol{\Xi}^m$ be an i.i.d. sample of size m, drawn according to some distribution D. For $m \geq 2/\chi^2$, the probability that there exists a classifier $f(\boldsymbol{x}, \boldsymbol{\theta}) \in F(\boldsymbol{X},\boldsymbol{\Theta})$ for which the deviation of the error probability $\varepsilon(\boldsymbol{\theta})$ from the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ on $\boldsymbol{\Xi}^m$, is larger than χ, is bounded by:*

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left|\varepsilon(\boldsymbol{\theta}) - \hat{\varepsilon}(\boldsymbol{\theta})\right| > \chi\right\} \leq 6S^F_{\max}(2m)\exp\left(-\frac{1}{4}m\chi^2\right) \tag{13}$$

The maximum number of dichotomies $S^F_{\max}(2m)$ that can be induced on a set of $2m$ points, here serves as a measure of the richness of the class F. The exponential function

is related to the Hoeffding inequality (1), although its rate of convergence is significantly slower. Since the exponential function, for a large enough m, eventually dominates the polynomial $S^F_{\max}(2m)$, it indeed follows from (13) that the error frequencies converge to the error probabilities for all functions in F simultaneously:

$$\lim_{m\to\infty} P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left|\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})\right|>\chi\right\}=0 \tag{14}$$

The proof of (13) is based on a cross validation of two samples Ξ_1^m of size m and $\Xi_2^{\beta m}$ of size βm. The probability of maximum deviation $P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left|\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})\right|>\chi\right\}$ is bounded is terms of the probability of deviation of the empirical errors $\hat{\varepsilon}_1(\boldsymbol{\theta})$ and $\hat{\varepsilon}_2(\boldsymbol{\theta})$ on Ξ_1^m and $\Xi_2^{\beta m}$ respectively, see [Vapnik 1971b], [Vapnik 1982]. By taking $\beta = 1$, the richness of F is then characterized as the number of possible dichotomies $S^F_{\max}(2m)$ on a set of $2m$ points. In [Devroye 1982] a similar result is derived by choosing $\beta = m - 1$, with using $S^F_{\max}\left(m^2\right)$ correspondingly:

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left|\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})\right|>\chi\right\}\le 4\exp\left(4\chi+4\chi^2\right)S^F_{\max}\left(m^2\right)\exp\left(-2m\chi^2\right) \tag{15}$$

Which value of β in (13) or (15) performs better, depends on the actual values of m, V, and χ. When the capacity V is large, the polynomial $S^F_{\max}\left(m^2\right)$ of order V in (15) may become so large that the exponential function $\exp\left(-2m\chi^2\right)$ is eventually dominated, despite the larger constant in the exponential function. For large values of $m\chi^2$, however, formulation (15) will outperform (13) due to the faster convergence of this exponential function. The optimal value of β is obviously a function of m, V, and χ but is difficult to obtain.

With similar methods as presented in section 2.1.1.1 and section 2.1.1.2, it is possible to derive from (13) or (15) an upper bound on χ in terms of V, m and a confidence factor δ, to derive a lower bound on m, when given V, δ, and χ, etc. This will be postponed until the next section, where a bound on the relative deviation is presented.

Related results of theorem 2.4 can be found in [Devroye 1988] and [Vapnik 1992], where also bounds on the *expected* maximum deviation are presented.

2.1.2.3. *A Bound on the Probability of Relative Deviation*

As with Hoeffding's inequality, the formulation of the inequalities (13) and (15) is tight when $\varepsilon(\boldsymbol{\theta}) \approx 0.5$. Similar to the approach in section 2.1.1.2, we can tighten the results of (13) and (15) by bounding the one-sided relative deviation of the true and estimated error instead of the absolute deviation. This is shown in the following theorem:

Theorem 2.5: *Let $F(\boldsymbol{X}, \boldsymbol{\Theta})$ be a set of classification functions with finite capacity V and let Ξ^m be an i.i.d. sample of size m, drawn according to some distribution D. For $m > \max\left(2/\chi^2, 4V/\chi^2\right)$, the probability that there exists a classifier $f(\boldsymbol{x}, \boldsymbol{\theta}) \in F(\boldsymbol{X}, \boldsymbol{\Theta})$ for which the one-sided relative deviation of the error probability $\varepsilon(\boldsymbol{\theta})$ from the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ on Ξ^m, is larger than χ, is bounded by:*

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{\varepsilon(\boldsymbol{\theta}) - \hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} \le 4 S^F_{\max}\left(m(\beta+1)\right) \exp\left(-\frac{1}{2} m\chi^2 \left(\frac{\beta}{\beta+1}\right)^2\right) \tag{16}$$

where β is given by :*

$$\beta = \frac{1}{m}\left\langle \frac{m^2\chi^2}{2V}\left(1 + \sqrt{1 - \frac{4V}{m\chi^2}}\right) - m \right\rangle \tag{17}$$

The proof of this theorem is presented in appendix A. Similar to the proof of theorem 2.4, it is based on a cross validation of two samples Ξ_1^m of size m and $\Xi_2^{\beta m}$ of size βm. Contrary to other formulations and related results, inequality (16) allows the determination of an approximately optimal value for β, as shown in equation (17). In fact, in appendix A it is shown that inequality (16) is valid for any value of $\beta > 0$ such that $\beta m \in \mathbb{N}$, but that (16) is approximately minimized by taking β as in (17). The restriction that $\beta m \in \mathbb{N}$ follows from the fact that βm is the size of the second sample $\Xi_2^{\beta m}$, which is obviously and integer value. Due to the requirement $m > \left(4V/\chi^2\right)$ the parameter β is larger than 1, and converges to $(m\chi^2/V)$ for $m\chi^2 >> V$.

* As was already stated in the list of symbols, the notation $\langle x \rangle$ represents the variable x rounded off to the nearest integer.

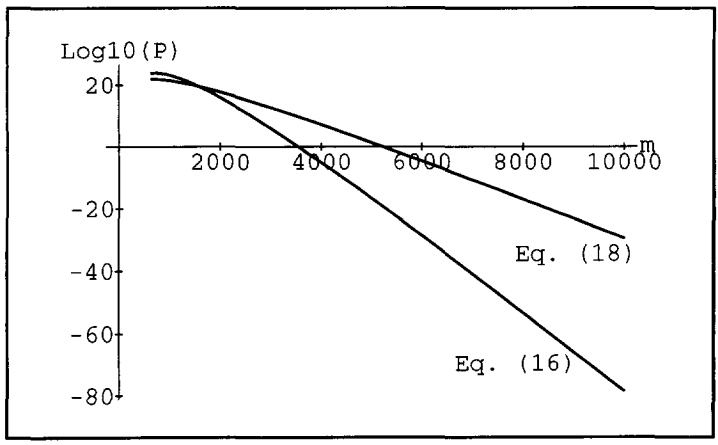


Figure 3: The logarithm base 10 of the probabilities of inequalities (16) (the lower curve on the right side) and (18) (the higher curve on the right side) as a function of m, with $\chi = 0.25$ and $V = 10$. Note that (18) is only smaller than (16) for the case that the bound is trivial; i.e. when the probability of maximum relative deviation is equal to 1.

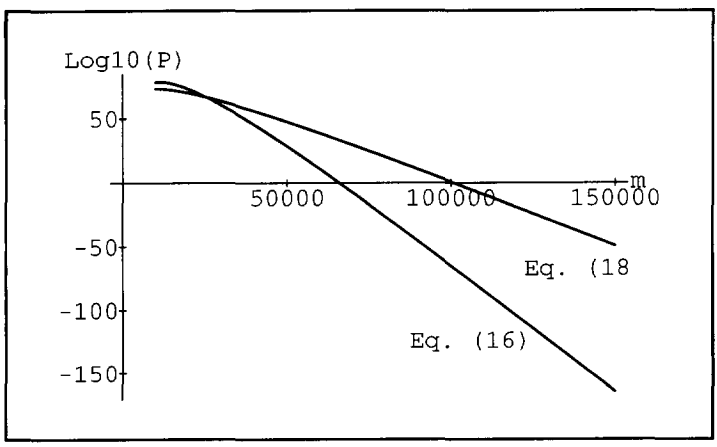


Figure 4: The same plot as in figure 1, now for $\chi = 0.1$ and $V = 25$. The lower curve on the right is computed from (16) and the higher curve from (18).

Theorem 2.5 is a slight improvement of a similar result of Vapnik [Vapnik 1982]. Vapnik's formulation is that for $m \geq 2/\chi^2$:

$$P\left\{\sup_{\theta\in\Theta}\frac{\varepsilon(\theta)-\hat{\varepsilon}(\theta)}{\sqrt{\varepsilon(\theta)}}>\chi\right\}\leq 8S^F_{\max}(2m)\exp\left(-\frac{1}{4}m\chi^2\right) \tag{18}$$

The difference between (16) and (18) is that for large β the exponential function approaches $\exp\left(-m\chi^2/2\right)$, whereas the exponential in (18) is equal to $\exp\left(-m\chi^2/4\right)$. E.g. for $V = 0$, the number of dichotomies $S^F_{\max}\left(m(\beta+1)\right)$ is by definition equal to 1, so theorem 2.5 leads to:

$$P\left\{\sup_{\theta\in\Theta}\frac{\varepsilon(\theta)-\hat{\varepsilon}(\theta)}{\sqrt{\varepsilon(\theta)}}>\chi\right\}\leq 4\exp\left(-\frac{1}{2}m\chi^2\right) \tag{19}$$

Apparently, formulation (19) is only 4 times larger than inequality (7). In figures 3 and 4 the probability of maximum relative deviation is plotted as a function of m for some values of χ and V. The figures show that inequality (18) is only better in the range where the bound is trivial; i.e. where the probability of maximum relative deviation is bounded by 1. For larger values of m, for which the probability of maximum relative deviation is bounded below 1, inequality (16) outperforms (18). For example, from figure 3 it can be seen that for $\chi = .25$, $V = 10$ and $m = 4000$, inequality (16) bounds the probability of maximum relative deviation by 8.1 10^{-6}, whereas inequality (18) provides the trivial bound 1.4 10^{7}. For $m = 6000$, inequalities (16) and (18) bound the probability of maximum relative deviation by 2.1 10^{-29} and 2.1 10^{-5} respectively.

Another notable difference between the formulation of theorem 2.5 and inequality (18) is the restriction $m > \max\left(2/\chi^2, 4V/\chi^2\right)$ in theorem 2.5, whereas (18) only demands $m \geq 2/\chi^2$. The restriction follows from the determination of β in (17). For $m < 4V/\chi^2$ in (17) the argument of the square root becomes negative, and since β is by definition a real and positive number, inequality (17) is not valid anymore. In lemma A.3 in appendix A it is shown that for $m < 4V/\chi^2$ the best bound that can be obtained corresponds to $\beta = 0$. Substitution of $\beta = 0$ in (16) however yields the trivial bound:

$$P\left\{\sup_{\theta\in\Theta}\frac{\varepsilon(\theta)-\hat{\varepsilon}(\theta)}{\sqrt{\varepsilon(\theta)}}>\chi\right\}\leq 4S^F_{\max}(m) \tag{20}$$

In this sense, the restriction $m > \max\left(2/\chi^2, 4V/\chi^2\right)$ seems to illustrate that m has to be significantly larger than V/χ^2 to let (16) become non-trivial. Although a similar restriction is not explicitly mentioned in the formulation of inequality (18) it is certainly implicit, see also figures 3 and 4.

As in section 2.1.1.2, it is possible to reformulate theorem 2.5 such that it is, in probability, an upper bound for the true error, when given the estimated error $\hat{\varepsilon}(\boldsymbol{\theta})$, the capacity V, the sample size m, and the confidence factor δ. Before this result is presented, however, we need the following definition:

Definition 2.2: *The function $z_1(x)$ is defined as the positive root in y of $\ln(y+1)+x-y/2$ as a function of x, with $x \geq 1$:*

$$z_1(x) = \left\{ y > 0 \;\middle|\; \ln(y+1)+x-y/2 = 0;\; x \geq 1 \right\} \tag{21}$$

Since the function $z_1(x)$ is not in closed form, it is plotted and tabulated in appendix B. A good closed form approximation of $z_1(x)$ with an approximation error of less than 3% is given by the function $a_1(x)$:

$$a_1(x) = 2\left(x + \ln(1+x)\right) + 2.5 \tag{22}$$

With the definition of $z_1(x)$ we are now ready for the next result:

Theorem 2.6: *Let $F(\boldsymbol{X}, \boldsymbol{\Theta})$ be a set of classification functions with finite capacity V and let $\boldsymbol{\Xi}^m$ be an i.i.d. sample of size $m \geq V$, drawn according to some distribution D. When the error frequency on $\boldsymbol{\Xi}^m$ for some $f(\boldsymbol{x}, \boldsymbol{\theta}) \in F(\boldsymbol{X}, \boldsymbol{\Theta})$ is denoted $\hat{\varepsilon}(\boldsymbol{\theta})$, and the corresponding probability of error $\varepsilon(\boldsymbol{\theta})$, then the following inequality holds with probability at least 1 - δ for all $f(\boldsymbol{x}, \boldsymbol{\theta}) \in F(\boldsymbol{X}, \boldsymbol{\Theta})$ simultaneously:*

$$\varepsilon(\boldsymbol{\theta}) \leq \hat{\varepsilon}(\boldsymbol{\theta}) + \frac{\chi^2}{2}\left(1 + \sqrt{1 + \frac{4\hat{\varepsilon}(\boldsymbol{\theta})}{\chi^2}}\right) \tag{23}$$

where χ is given by:

$$\chi = \left(\frac{\beta+1}{\beta}\right)\sqrt{\frac{2}{m}\left(\ln\left(S^F_{\max}\left(m(\beta+1)\right)\right) + \ln\left(\frac{4}{\delta}\right)\right)} \tag{24}$$

and β *by:*

$$\beta = \frac{1}{m}\left\langle m\, z_1\left(\ln\left(\frac{em}{V}\right)+\frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)\right\rangle \tag{25}$$

The proof of this theorem is presented in appendix C. The proof shows that equations (23) and (24) are also valid for every $\beta > 0$ such that $\beta m \in \mathbb{N}$, but that (23) is approximately minimized by choosing β as in (24).

Since (23 - 25) are valid for all classifiers $f(\boldsymbol{x}, \boldsymbol{\theta}) \in F(\boldsymbol{X}, \boldsymbol{\Theta})$ simultaneously, they also hold for the classifier with the lowest error frequency $\hat{\varepsilon}(\boldsymbol{\theta}_{best})$. As in section 2.1.1.1 we can state that for the best classifier in $F(\boldsymbol{X}, \boldsymbol{\Theta})$ the following inequality holds with probability at least $1 - \delta$:

$$\varepsilon(\boldsymbol{\theta}_{best}) \le \hat{\varepsilon}(\boldsymbol{\theta}_{best}) + \frac{\chi^2}{2}\left(1+\sqrt{1+\frac{4\hat{\varepsilon}(\boldsymbol{\theta}_{best})}{\chi^2}}\right) \tag{26}$$

with χ and β as in (24) and (25).

In order to connect theory with practice, a number of tables are presented in appendix D, that provide a numerical evaluation of theorem 2.6; a guaranteed upper bound for the true error is given for several values of m, V, $\hat{\varepsilon}(\boldsymbol{\theta})$ and δ. According to this numerical evaluation, theorem 2.6 outperforms Vapnik's reformulation of (18) for all values of m, V, $\hat{\varepsilon}(\boldsymbol{\theta})$ and δ that bound the true error below 1.

A last reformulation of theorem 2.5 is to provide a lower bound for the sample size m, given χ, V and δ. First, however, we will define the function $z_2(x)$:

Definition 2.3: *The function* $z_2(x)$ *is defined as the positive root in* y *of* $\ln\left((y+1)^3 / y^2\right) + x - (y-2)/2$ *as a function of* x, *with* $x \ge 1$:

$$z_2(x) = \left\{ y > 0 \;\middle|\; \ln\left((y+1)^3 / y^2\right) + x - (y-2)/2 = 0; x \ge 1 \right\} \tag{27}$$

The function $z_2(x)$ is plotted and tabulated in appendix B. A good closed form approximation of $z_2(x)$, with an approximation error of less than 3% for x between 1 and 10000, is given by the function $a_2(x)$:

$$a_2(x) = 2x + 0.7\ln(1+x) + 6.5 \tag{28}$$

A lower bound on the sample size is presented in the following theorem:

Theorem 2.7: *Let $F(\boldsymbol{X}, \boldsymbol{\Theta})$ be a set of classification functions with finite capacity V and let $\boldsymbol{\Xi}^m$ be an i.i.d. sample of size m, drawn according to some distribution D. In order to let the probability, that the maximum one-sided relative deviation of the true error $\varepsilon(\boldsymbol{\theta})$ from the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ on $\boldsymbol{\Xi}^m$ is larger than χ, be at most δ:*

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{\varepsilon(\boldsymbol{\theta}) - \hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} \leq \delta \tag{29}$$

the sample size m should at least be:

$$m \geq \frac{1}{\beta}\left\lceil \frac{3}{\chi^2}\frac{(\beta+1)^2}{\beta}\left(V\ln\left(\frac{2e}{\chi^2}\frac{(\beta+1)^3}{\beta^2}\right) + \ln\left(\frac{4}{\delta}\right)\right)\right\rceil \tag{30}$$

where β is given by:

$$\beta = z_2\left(\ln\left(\frac{2e}{\chi^2}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right) \tag{31}$$

The proof of this theorem is given in appendix E. Similar to the previous two theorems, equation (30) is valid for any $\beta > 0$ such that $\beta m \in \boldsymbol{\aleph}$, but it is approximately minimized by taking β as in (31). Theorem 2.7 improves a result of Blumer et al. that is based on Vapnik's equality (18) [Blumer 1989]:

$$m \geq \max\left(\frac{8}{\chi^2}\ln\frac{8}{\delta}, \frac{16V}{\chi^2}\ln\frac{16}{\chi^2}\right) \tag{32}$$

The important aspect of (31) is that, for some fixed β, it clearly shows the linear relation between m and V. Thus, to let two classifiers be equally over-determined (as quantified by χ), with equal probability (as quantified by δ), the sample size of their training sets should be proportional to their capacity V.

An issue that has only slightly been addressed so far, is that χ is generally chosen as a function of the true error $\varepsilon(\boldsymbol{\theta})$. For example, for small $\varepsilon(\boldsymbol{\theta})$, one would allow a much smaller χ than for a large $\varepsilon(\boldsymbol{\theta})$. The key problem that remains here, is that $\varepsilon(\boldsymbol{\theta})$ is generally not known in advance, so it is difficult to choose a value for χ. One way to investigate this fact in more detail, is to write χ as $\chi = \gamma\sqrt{\varepsilon(\boldsymbol{\theta})}$. This leads to the interesting reformulation of the probability of maximum one-sided relative deviation in (16) and (18) as the probability that there is some classifier for which the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ is less than a fraction *(1 - γ)* of the true error $\varepsilon(\boldsymbol{\theta})$:

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{\varepsilon(\boldsymbol{\theta}) - \hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} = P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left((1-\gamma)\varepsilon(\boldsymbol{\theta}) > \hat{\varepsilon}(\boldsymbol{\theta})\right)\right\} \tag{33}$$

Substitution of $\chi = \gamma\sqrt{\varepsilon(\boldsymbol{\theta})}$ in theorem 2.7 leads to the following reformulation of the lower bound on *m*:

$$m \geq \frac{1}{\beta}\left\lceil \frac{3}{\gamma^2\varepsilon(\boldsymbol{\theta})}\frac{(\beta+1)^2}{\beta}\left(V\ln\left(\frac{2e}{\gamma^2\varepsilon(\boldsymbol{\theta})}\frac{(\beta+1)^3}{\beta^2}\right) + \ln\left(\frac{4}{\delta}\right)\right)\right\rceil \tag{34}$$

In this formulation it is clear that m directly depends on $\varepsilon(\boldsymbol{\theta})$. Also, it is not surprising that m goes to infinity as $\varepsilon(\boldsymbol{\theta})$ approaches zero, since the fraction $(1-\gamma)\varepsilon(\boldsymbol{\theta})$ also goes to zero. Without exact knowledge of $\varepsilon(\boldsymbol{\theta})$, however, it is obvious that the formulation as in (34) is not very practical and will therefore not be used in approaching a pattern recognition problem. This is an issue that will be discussed further in section 2.3 and in section 3.2.

A second issue that is illuminated by reformulating χ as $\chi = \gamma\sqrt{\varepsilon(\boldsymbol{\theta})}$ is the difference between the bound on the maximum deviation of the true and estimated errors and the bound on the maximum relative deviation. Substitution of $\chi = \gamma\sqrt{\varepsilon(\boldsymbol{\theta})}$ in theorem 2.5 yields:

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left((1-\gamma)\varepsilon(\boldsymbol{\theta}) > \hat{\varepsilon}(\boldsymbol{\theta})\right)\right\} \leq 4S^F_{\max}\left(m(\beta+1)\right)\exp\left(-\frac{1}{2}m\gamma^2\varepsilon(\boldsymbol{\theta})\left(\frac{\beta}{\beta+1}\right)^2\right) \tag{35}$$

If equation (13) in theorem 2.4 is reformulated in a similar way, we need to substitute $\chi = \gamma\varepsilon(\boldsymbol{\theta})$:

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left((1-\gamma)\varepsilon(\boldsymbol{\theta})>\hat{\varepsilon}(\boldsymbol{\theta})\right)\right\}\leq 3S^{F}_{\max}(2m)\exp\left(-\frac{1}{4}m\gamma^{2}\varepsilon^{2}(\boldsymbol{\theta})\right) \tag{36}$$

For fixed β, a comparison of (35) and (36) shows that the exponential in (35) depends on $\varepsilon(\boldsymbol{\theta})$, whereas the exponential in (36) depends on $\varepsilon^2(\boldsymbol{\theta})$. In this sense, this comparison shows that the bound on the maximum relative deviation (35) is indeed very much tighter than the bound on the maximum deviation (36).

2.2 Applications, Implications and Limitations

In this section some applications, some implications and some limitations of the framework of section 2.1 are discussed.

2.2.1. Applications

In section 2.1 it has become clear that the capacity is a key parameter. If it is possible to determine this parameter for a parametric classifier, the Vapnik-Chervonenkis theory can be used to predict the validity of its generalization. Two examples of applications are the following:

• *Linear classifiers:* The capacity of a linear threshold function in $\Re^d$, and the number of dichotomies that a linear threshold function in $\Re^d$ can induce on a set of m points in $\Re^d$ follow from the following well-known theorem of Cover [Cover 1965], see also [Duda 1973], [Devijver 1982], [Wenocur 1981] and [Devroye 1988]:

Theorem 2.8 *[Cover 1965]: Let F^{ltf} be the class of linear threshold functions in d-dimensional Euclidean space. Then the capacity $V = d + 1$ and for all $m \geq d + 1$:*

$$S^{ltf}_{\max}(m,d)\leq 2\sum_{i=0}^{d}\binom{m-1}{i} \tag{37}$$

Substitution of (37) in theorem 2.5 provides a bound on the probability of maximum one-sided relative deviation for a linear classifier in $\Re^d$, when given a training sample of size m. Theorem 2.6, and its numerical evaluation in appendix D, provide an upper bound on the true error, given a sample size m and with a certain confidence at least $1 - \delta$. Finally, theorem 2.7 provides a lower bound on the sample size m, for given V, χ, and δ. Since theorem 2.8 will play an important role in chapter 3, we will make three more remarks regarding Cover's result:

- Substitution of $V = d + 1$ in (9) indeed shows that (9) upper bounds (37). The resemblance of the general case (9) with the special case (37) of a linear classification function in $\Re^d$, however, shows that there is indeed a connection between the number of free parameters of a classifier and its capacity.

- The value $V = d + 1$ for the capacity of a linear classifier in $\Re^d$ implies with theorem 2.7 that the sample size m should be linear in the dimensionality of the feature space d.

- Cover's result is frequently mentioned in the literature in that it shows that an error estimate of a linear classifier on a small number of points may not have very much significance, since there is a finite probability that the points are linearly separable, e.g. see [Duda 1973] and [Devijver 1982]. However, the question on *how significant* the error estimate is, often remains unanswered. The final answer to this question is given by the Vapnik-Chervonenkis theory, as shown in theorem 2.6.

• *Quadratic classifiers*: The case of quadratic classifiers in $\Re^d$ is easily solved when the quadratic classifier is seen as a *generalized linear classifier*, e.g. see [Duda 1973] and [Devijver 1982]. The dimensionality of this generalized linear classifier can be shown to be equal to: $d^* = 2\,d + d\,(d - 1)\,/\,2$. Therefore, the capacity of a quadratic classifier is equal to: $V = d^* + 1 = 2\,d + d\,(d - 1)\,/\,2 + 1$. Theorem 2.7 here predicts that m should now roughly be proportional to d^2.

In the next chapter more applications in the field of connectionist models will be presented. Other applications in pattern recognition are discussed in [Devroye 1988].

2.2.2. Implications and Limitations

In this section a number of aspects will be discussed that illuminate various implications and limitations of the Vapnik-Chervonenkis theory. These include some of the assumptions that are implicit in the framework, and some thoughts on its practical applicability.

• Probably the most wonderful aspect of the theory is its mathematical elegance. Only one parameter, the capacity, is sufficient to characterize a parametric pattern classifier completely. With this parameter, it is possible to predict how over-determined a classifier

is by a training sample of size m, independently of the distribution of the training data or the type of the classification function (i.e. only the value of the capacity V is important, not whether it originates from a linear classifier in a high-dimensional space or a quadratic classifier in a low-dimensional space). In this sense, the Vapnik-Chervonenkis theory unifies a large number of pattern recognition algorithms within one framework.

• As the Vapnik-Chervonenkis theory is essentially a uniform convergence result, the statements that are made on the classifiers in a set F actually hold for all classifiers in F simultaneously. It was already mentioned that they therefore also hold for the classifier that has the lowest empirical error on the training sample. Since it does not matter how the classifier with lowest error frequency on the training sample is obtained, the theory is independent of the specific learning rule that is used to train a classifier. For example, it does not matter whether a linear classifier is trained with the Widrow-Hoff rule, the Ho-Kashyap rule, a pseudo-inverse method or the perceptron learning rule [Duda 1973], to predict its validity of generalization.

• An important aspect is that the theoretical framework only bounds the *difference* between the true error and the estimated error and that it does not provide any statements on the value of the true error itself. As was discussed in chapter 1, the actual value of the true error, or the lowest true error that can be reached by selecting a classifier from a set of classifiers F is an important issue, that however, can not be solved within the framework of the Vapnik-Chervonenkis theory. The only clue that is available in this context, is that when the class of classifiers F becomes larger (i.e. when its capacity increases), it is more likely that there will be a classifier in F that has low empirical error. On the other hand, making F too large, may result in over-fitting of the data, in which case theorem 2.6 predicts that the empirical error becomes a bad estimate for the true error.

This behavior is essentially equal to the peaking phenomenon that was described in chapter 1. An increase of the number of features of a classifier, or an increase of the complexity of a classification function, will initially lower the true error of a classifier. However, after a certain minimum has been reached, the classifier becomes underdetermined, and a further increase of the number of features or of the classifier complexity will deteriorate the performance. The location of this minimum in the true error is problem dependent, and can not be fit, therefore, within one single framework. The generic behavior of the peaking phenomenon, however, directly follows from the Vapnik-Chervonenkis theory.

• The previous line of thought leads to an approach for the selection of a classification function that is called "structural risk minimization" [Vapnik 1982]. In this approach a *structure* is imposed on a set of classification functions $F(\boldsymbol{X}, \boldsymbol{\Theta})$ such that for the q subsets of F the following relation holds:

$$F_1(\boldsymbol{X}, \boldsymbol{\Theta}) \subset F_2(\boldsymbol{X}, \boldsymbol{\Theta}) \subset \ldots\ldots \subset F_q(\boldsymbol{X}, \boldsymbol{\Theta}) \tag{38}$$

This relation of subsets is such that the following inequalities are valid for the corresponding capacities:

$$V_1 < V_2 < \ldots\ldots < V_q \tag{39}$$

An example of such a structure is a set of polynomial discriminant functions with increasing order. For a given sample of fixed size m, theorem 2.6 can now be used to obtain guaranteed bounds on the true error for each subset $F_i(\boldsymbol{X}, \boldsymbol{\Theta})$ separately:

$$\varepsilon(\boldsymbol{\theta}_i) \leq \hat{\varepsilon}(\boldsymbol{\theta}_i) + \Psi\left(V, m, \hat{\varepsilon}(\boldsymbol{\theta}_i), \delta\right) \tag{40}$$

Here, the function $\Psi()$ is given by equations (23 - 25) in theorem 2.6. The method of structural risk minimization now consists of two steps:

- In the first step, the classifier with lowest error frequency for each subclass $F_i(\boldsymbol{X}, \boldsymbol{\Theta})$ is determined.
- Then in the second step, the classifier from the subclass that has the lowest sum of $\hat{\varepsilon}(\boldsymbol{\theta}_i)$ and $\Psi()$ is selected as the final classification function. Because (40) has been applied q times in the determination of the best classifier, the corresponding upper bound for the true error is valid with probability at least $1 - q\delta$.

By this procedure, the selection of a classifier is based on balancing a low empirical error against the potential increase of the true error, due to the higher complexity of the classification functions. This balancing of the empirical performance and a term that represents the complexity, also appears in the framework of Minimum Description Length model selection, e.g. see [Barron 1991]. This is a Bayesian framework in which models with low description length are considered to have a higher prior probability than models with a large description length. The actual selection of the model is then based on a balance between the empirical error of the model and its prior probability.

• Despite the mathematical elegance of the Vapnik-Chervonenkis theory, its practical applicability is somewhat limited. This is due to the fact that the theory is distribution free, and therefore takes into account the worst possible distribution, see also [Haussler 1992]. Examples of this can be found in the tables in appendix D. For example, for $\hat{\varepsilon}(\boldsymbol{\theta}) = 0\%$ and $\delta = 1\%$, the sample size m should be at least be larger than $20V$ to bound $\varepsilon(\boldsymbol{\theta})$ below 100%, at least be larger than $250V$ to bound $\varepsilon(\boldsymbol{\theta})$ below 10%, and be larger than $4000V$ to bound $\varepsilon(\boldsymbol{\theta})$ below 1%. This quickly leads to requirements on m that are not feasible in many applications.

• As a final remark, an important part of the true value of the Vapnik-Chervonenkis theory is the fact that it provides considerable insight in the scaling behavior of classification problems. The theorems 2.5, 2.6 and 2.7 above, clearly show how the various parameters in the design of a classifier are interrelated and how a change of one parameter will affect the other parameters. Also it is intriguing that the theory predicts a number of the relationships, mentioned in chapter 1, that were found by empirical research or derived in a distribution specific context. In this sense, the Vapnik-Chervonenkis theory shows that these relations are generic properties in pattern recognition, which do not depend on the underlying probability distribution, nor on the actual type of classification function.

2.3 Concept Learning and Computational Learning Theory

An alternative view on the Vapnik-Chervonenkis theory is in use within a relatively new area of applied mathematics, called *concept learning* or *computational learning theory*. Concept learning is a complexity theoretic approach to learning and originates in this sense from an idea of Valiant [Valiant 1984]. Valiant proposed the paradigm of "probably approximately correct" learning (PAC-learning) as a model of natural learning, which has the advantage that it allows a thorough analysis, e.g. see [Valiant 1984], [Blumer 1989] and [Natarajan 1991]. Within this paradigm, it is investigated how difficult it is to learn a certain concept. Here a concept is defined in a technical sense as being a subset of a universe of objects. For example, part of a logical rule for learning the concept of *a cup* could be the following [Natarajan 1991]:

$$(\textit{has handle}) \wedge (\textit{is open}) \wedge (\textit{is container}) \tag{41}$$

A learning machine can learn such a concept from a number of (positive and/or negative) examples. It is supposed that these examples are generated according to some

unknown underlying probability distribution D. Given a set of examples, the machine generates a hypothesis for the target concept, which is consistent with all the examples that are presented (so far). The hypotheses are chosen from a hypothesis space that is called H, and the error of a hypothesis is called ε. Intuitively, ε will decrease as the number of examples increases. However, due to the stochastic nature of the generation of the examples, the concept will be learned in probability. Thus, this leads to the formulation that a hypothesis will have an error less than ε with probability at least $1 - \delta$, when the number of examples is m. This explains the term PAC-learning, since a concept is learned probably (i.e. with probability at least $1 - \delta$) approximately correct (i.e. with error ε). Valiant defined the class of *learnable concepts* as those concepts for which m is polynomial in ε and δ. Concepts for which m is non-polynomial in ε and δ are called *non-learnable*. Obviously, a key issue in this discussion is the complexity of the concept to be learned. In fact, it appears that the complexity of a concept determines whether the concept is learnable or non-learnable. An important topic in the literature, therefore, consists of the characterization of such complexity measures in concept learning, e.g. see [Valiant 1984], [Blumer 1986], [Blumer 1989], [Natarajan 1991] and the proceedings of the workshops on computational learning theory: [COLT 1988], [COLT 1989] and [COLT 1990].

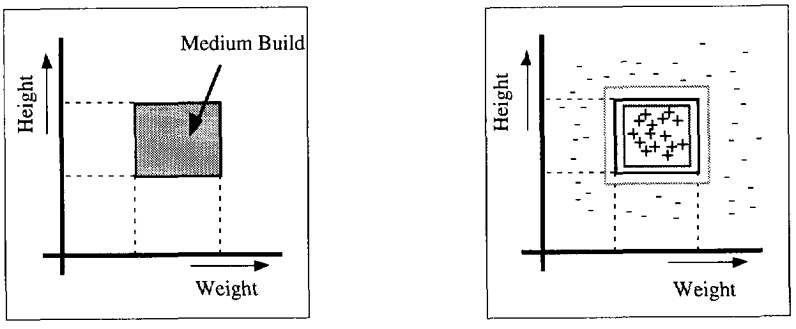


Figure 5: An example from [Blumer 1989]. Left: the rectangular geometric concept "Medium Build" is defined for persons having their weight and height within certain limits. Right: the target concept has to be inferred from a set of examples depicted with + (i.e. positive examples) and - (i.e. negative examples). When the concept has to be learned from the set examples shown here, it appears that there are not enough examples given to unambiguously decide what the concept is.

A class of concepts that we will be concerned within the rest of this section, are geometric concepts. A geometric concept is defined as a subspace of a (possibly) high-dimensional feature space. For example, a geometric concept can be rectangular region of $\Re^2$ (see figure 3), a sphere in $\Re^6$, a half space bounded by a polynomial curve, etc.

Returning to the theory of section 2.1, a natural measure for the complexity of a geometric concept is the capacity. The capacity is consistently called the Vapnik-Chervonenkis dimension in the area of concept learning. As the theoretical framework of concept learning assumes that the environment is noiseless, the learning of geometric concepts can be considered as a special case of the theory developed in section 2.1. Since only hypotheses are considered that are consistent with the examples, we can substitute $\hat{\varepsilon}(\boldsymbol{\theta}) = 0$ in theorem 2.5 to obtain the following expression for the probability of maximum one-sided relative deviation:

$$P\left\{\sup_{\substack{\boldsymbol{\theta}\in\boldsymbol{\Theta}\\ \hat{\varepsilon}(\boldsymbol{\theta})=0}} \frac{\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} = P\left\{\sup_{\substack{\boldsymbol{\theta}\in\boldsymbol{\Theta}\\ \hat{\varepsilon}(\boldsymbol{\theta})=0}} \varepsilon(\boldsymbol{\theta}) > \chi^2\right\} = P\left\{\sup_{\substack{\boldsymbol{\theta}\in\boldsymbol{\Theta}\\ \hat{\varepsilon}(\boldsymbol{\theta})=0}} \varepsilon(\boldsymbol{\theta}) > \varepsilon\right\} \qquad (42)$$

Apparently, for the case that the empirical error is zero, the substitution $\varepsilon = \chi^2$ yields a bound on the probability that the true error is larger than ε. Furthermore, it is obvious that:

$$P\left\{\sup_{\substack{\boldsymbol{\theta}\in\boldsymbol{\Theta}\\ \hat{\varepsilon}(\boldsymbol{\theta})=0}} \frac{\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} \le P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} \qquad (43)$$

as the search for the $\boldsymbol{\theta}$ with the maximum relative deviation is limited to those values of $\boldsymbol{\theta}$ for which the empirical error is zero. In terms of concept learning, this leads to the following interesting corollary of theorem 2.7:

Corollary 2.1: *Let H be a hypothesis space of finite capacity V and let Ξ^m be a set of m i.i.d. positive and negative examples of a target concept c, drawn according to some unknown distribution D. When the number of examples is larger than:*

$$m \ge \frac{1}{\beta}\left\lceil \frac{3}{\varepsilon}\frac{(\beta+1)^2}{\beta}\left(V \ln\left(\frac{2e}{\varepsilon}\frac{(\beta+1)^3}{\beta^2}\right) + \ln\left(\frac{4}{\delta}\right)\right)\right\rceil \qquad (44)$$

then the probability that there exists a hypothesis in H that is consistent with all examples, but has error larger than ε, is less than δ. Here β is given as:

$$\beta = z_2\left(\ln\left(\frac{2e}{\varepsilon}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right) \tag{45}$$

We will conclude this section with three final remarks.

• As theorem 2.7, corollary 2.1 shows for fixed β a linear relation between the capacity V and the sample size m. According to Valiant's definition of a learnable concept, it now follows, that for learnable concepts the capacity can only be polynomially large in its parameters. For example, if the target concept is chosen from the class of half spaces bounded by a polynomial curve, this polynomial curve should be of finite order. Concepts from the space of "all convex polygons" are not learnable within this theoretical framework, since the number of faces of the polygon is not necessarily bounded.

• The prior knowledge that the concept can ultimately be learned with zero error resolves the problem of choosing χ as mentioned in section 2.1.2.3. In fact, the substitution $\varepsilon = \chi^2$ allows a lower bound on the sample size, when a true error of at most ε is allowed. The value of ε is a free parameter whose value will depend on the application.

• Also in corollary 2.1 no assumptions with respect to the underlying probability distribution are made. This has been subject of criticism from a number of authors who argue that this imposes too strict demands on a concept before it is called learnable [Baum 1990]. In fact, an enormous amount of knowledge concerning the shape of the concept is assumed, whereas there is a complete lack of knowledge with respect to the underlying distributions. These might not be realistic assumptions for learning as it appears in nature.

III Applications to some Connectionist Models

In this chapter, the mathematical framework that was presented in chapter 2 will be used to study the generalization behavior of some connectionist models. By using the Vapnik-Chervonenkis theory their generalization capabilities are studied in a worst case sense. The connectionist models that are investigated here, are the general class of multi-layer feedforward networks, a class of kernel-based networks that is related to the Parzen classifier and the class of multi-layer feedforward networks with shared parameters.

3.1. Introduction

During the last decade, a relatively new type of pattern classifiers has attracted considerable attention. These classifiers are called multi-layer feedforward (neural) networks and partially originate from biologically inspired models of the neuron. The recent excitement about neural networks is for a large part motivated by the invention of a training algorithm for such networks, called the backpropagation algorithm [Rumelhart 1986]. As the interest in single-layer networks, like the ADALINE ([Widrow 1960], [Widrow 1985], [Widrow 1988]) and the perceptron ([Rosenblatt 1958], [Duda 1973]), almost completely disappeared after the publication of the thorough study by Minsky and Papert [Minsky 1969], a new wave of interest emerged when it became apparent that multi-layer feedforward networks have considerable more computational capabilities. In fact, the availability of a training algorithm for multi-layer networks, some impressive applications (e.g. [Sejnowski 1987], [Gorman 1988a], [le Cun 1990a]), the proofs that they can approximate any Borel measurable function ([Funahashi 1989] [Hornik 1989], [Hartman 1990], [Park 1991]), and their neurobiological flavor caused an enormous amount of world-wide interest. This has resulted in numerous new books, new conferences, new journals and numerous ways to acquire new funds for research.

Despite of all this enthusiasm, multi-layer feedforward network classifiers have also been subject to criticism, since it became apparent that the backpropagation algorithm has little biological plausibility [Crick 1989], no closed-form solutions of the differential

equations governing the non-linear (stochastic) learning procedure are known to exist ([Maas 1990], [Weiss 1991], [Kolen 1991], [Schmidt 1993a], [Schmidt 1993b]), and finally it did not always seem to offer clear advantages over other pattern classifiers [Kraaijveld 1989] or other approximation methods [Geman 1992].

In some sense, this chapter is part of this critical approach to multi-layer networks. Although the Vapnik-Chervonenkis theory is a worst-case approach to study generalization properties, its application in section 3.2 will show that enormous amounts of data have to be available to train a network of moderate size. First, however, in section 3.1.1, some important basic elements of a number of connectionist models will be presented in a formal way. Since the Vapnik-Chervonenkis theory is independent of a learning rule (cf. section 2.2), reference to learning algorithms is only made for illustrative purposes in this section. Treatment and discussion of the learning rule are postponed to 3.3 and chapter 4. More thoughts on the validity of the excitement for neural networks will be presented in chapter 5.

3.1.1. Network Transfer Functions and Topologies

Many classification schemes and approximation schemes can be mapped onto some kind of network. The network can be regarded as a graphic notation for the classification algorithm or approximation algorithm [Poggio 1989]. Within this formulation, the *network transfer function is a composition of many basis functions.*

In a graphical representation, such a basis function is depicted in figure 1. The basis function can be considered as a processing "unit", since it corresponds to the smallest amount of processing power in the network. The functionality consists of the computation of a so-called activation function followed by an output function. The activation function is computed from the inputs of the unit and the parameters $\boldsymbol{\theta}$ that are associated with each input. An example of a frequently used activation function is the inner product of the input vector and the parameter vector:

$$act^{ip}(\boldsymbol{x}, \boldsymbol{\theta}) = \theta_0 + \sum_{inputs} x_i \theta_i \qquad (1)$$

The parameter θ_0 is an additional parameter that serves as a bias or a threshold. Another commonly used activation function is the Euclidean distance between the input vector and the parameter vector:

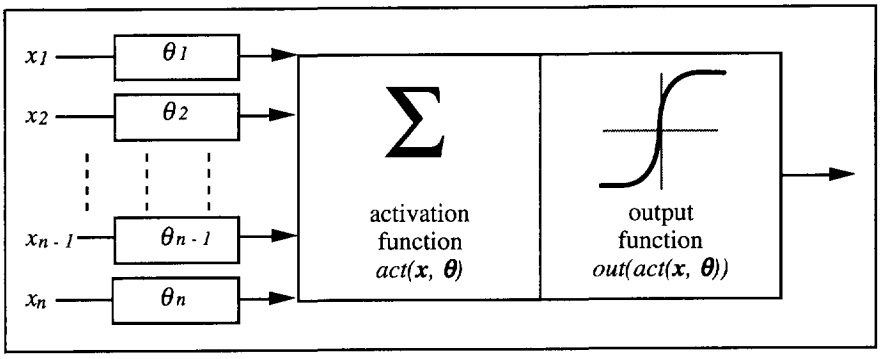


Figure 1: A unit, or basis function, in a multi-layer feedforward network. The unit computes an activation function from the parameter vector $\boldsymbol{\theta}$ and the input vector $\boldsymbol{x}$, followed by an output function that maps the activation to the output.

$$act^{Ed}(\boldsymbol{x},\boldsymbol{\theta}) = \theta_0 - \sqrt{\sum_{inputs}(x_i - \theta_i)^2} \tag{2}$$

Frequently used output functions are the linear output function:

$$out^{lin}(x) = x \tag{3}$$

the threshold function or indicator function:

$$out^{thr}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{4}$$

the sigmoidal output function:

$$out^{sigm}(x) = \frac{1}{1+e^{-x}} \tag{5}$$

and the Gaussian output function:

$$out^{Gaus}(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) \tag{6}$$

The parameters $\boldsymbol{\theta}$ in the activation functions (1) or (2) are determined by a learning procedure, such that the transfer function of the network approximates a certain desired mapping from input to output. The simplest possible network structure obviously consists of one basis function. For instance, if the inner product activation function is chosen with the linear output function, the network performs a weighed summation of the inputs. Such a simple network can be used to approximate a linear function and linear regression can serve as an excellent procedure to determine the parameters [Mood 1974]. If the inner product activation function is chosen in conjunction with the threshold output function, the network behaves as a linear classifier and can be trained with one of the numerous procedures to train such a classifier, e.g. see [Duda 1973].

If the basis functions are nested, the corresponding network structure can be depicted as a multi-layer network, see figure 2. If the basis functions in such a multi-layer network are organized in "layers", such that every basis function is computed with data from the previous layer only, the transfer function of the network can be formulated as*:

$$f^*(\boldsymbol{x},\boldsymbol{\theta}) = out\Big(act\Big(out\Big(act\Big(.....out\Big(act\big(\boldsymbol{x},\boldsymbol{\theta}^1\big)\Big).....,\boldsymbol{\theta}^{l-1}\Big)\Big),\boldsymbol{\theta}^l\Big)\Big) \tag{7}$$

In (7) the basis functions are structured in l layers. The parameters of layer i are indexed as $\boldsymbol{\theta}^i$, with $1 \le i \le l$. A particularly popular structure is the nested sigmoids scheme:

$$\begin{aligned} f^*(\boldsymbol{x},\boldsymbol{\theta}) &= out^{sigm}\Big(act^{ip}\Big(out^{sigm}\Big(act^{ip}\Big(.....out^{sigm}\Big(act^{ip}\big(\boldsymbol{x},\boldsymbol{\theta}^1\big)\Big).....,\boldsymbol{\theta}^{l-1}\Big)\Big),\boldsymbol{\theta}^l\Big)\Big) \\ &= out^{sigm}\Big(\sum_i \theta_i^l out_i^{sigm}\Big(\sum_j \theta_j^{l-1} out_j^{sigm}\Big(.....\sum_k \theta_k^1 x_k\Big).....\Big)\Big) \end{aligned} \tag{8}$$

---

* We make a rather technical distinction here, between the classification function $f()$ that maps onto $\{0, 1\}$, and the continuous function $f^*()$ mapping onto $(0, 1)$. The use of such continuous functions is required for some training procedures. As we are only concerned with pattern classification in this thesis (instead of approximation), the continuous transfer function $f^*()$ is only considered during the training phase. For classification purposes, $f^*()$ is thresholded on a suitable level, to obtain a classification function $f()$.

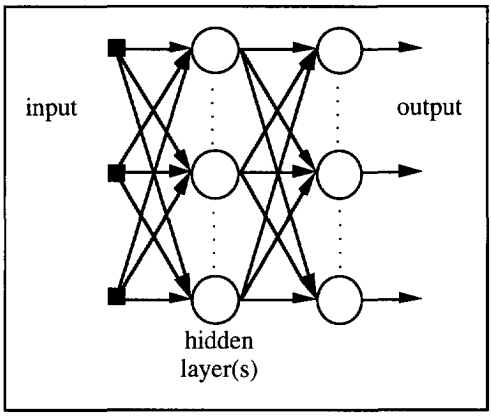


Figure 2: Nested basis functions can be represented as a multi-layer network structure. The network consists of units as depicted in figure 1. In this figure the units are organized in layers, and all units in one layer are connected to the units of the next layer. Layers of units that are not connected to the output are called hidden layers, e.g. in this figure the network has one hidden layer.

Although such a scheme of nested non-linear functions was highly unusual in pattern recognition or in the theory of approximation of continuous functions [Poggio 1989], it has recently become very popular in the literature on connectionist models. This was due to a number of reasons:

• In the first place, an iterative learning algorithm to train multi-layer networks was invented: the backpropagation algorithm [Rumelhart 1986]. This algorithm was believed to demonstrate that Minsky and Papert's pessimism about learning in multi-layer machines was misplaced [Minsky 1969], [Rumelhart 1986]. In fact, one of the first examples of the capabilities of the back-propagation algorithm was that it could learn a multi-layer network to solve the EXOR-problem [Rumelhart 1986]; a problem that in principle can not be solved with a single-layer network. Only a few years later, several authors showed that multi-layer networks with sigmoidal or Gaussian non-linearities and one hidden layer are indeed very powerful, since they can approximate any Borel measurable function [Funahashi 1989] [Hornik 1989], [Hartman 1990], [Park 1991]. The missing links of these results, however, are that they do not predict the correct number of hidden units for a certain application, and that local minima and/or saddle

points in the learning procedure may prevent the back-propagation algorithm from finding the proper values for the parameters θ.

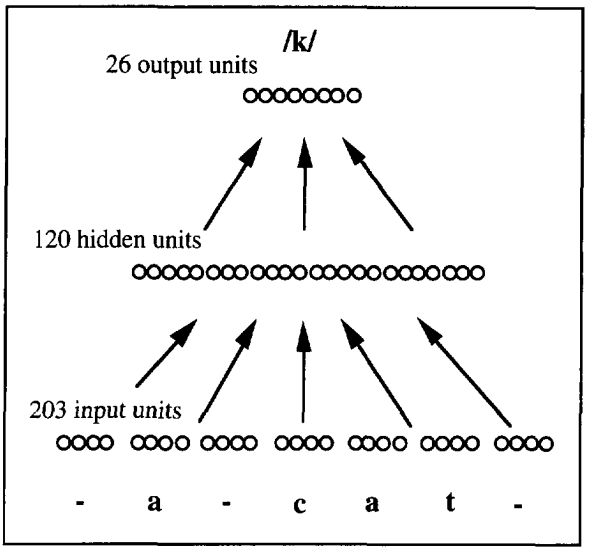


Figure 3: NETtalk [Sejnowski 1987]. The network performs the mapping of text to phonemes. The phoneme corresponding to the central character in a window of seven characters is derived. In this example, the phoneme /k/, corresponding to the *c* of "*a cat* ", is obtained from the context.

• A second reason for the popularity of multi-layer networks, is that a number of impressive demonstrations of their capabilities were published in the literature, e.g. see the survey in [Hertz 1991]. Among these are applications in the mapping of text to phonemes (see figure 3 and [Sejnowski 1987]), handwritten character recognition ([le Cun 1989], [le Cun 1990a]), car navigation [Pomerleau 1989], the world-champion computer program for playing backgammon [Tesauro 1990], and numerous others.

Some of these demonstrations are indeed impressive, since they appear to solve real world problems better than other approaches. The value of other demonstrations, however, was merely that the application was solved for the first time with a statistical pattern recognition approach. A good example is NETtalk [Sejnowski 1987], see figure 3. Previous approaches to map text to phonemes were based on rule-based systems with

hand-coded linguistic rules. Training a multi-layer network with a sufficiently large set of examples appeared to be orders of magnitude faster and easier. The value of NETtalk is therefore partly based on the fact that the approach to the problem is new. Whether the tool that was used in this approach, i.e. the multi-layer network, is better than other tools remains a subject of research. In chapter 4 other successful applications of multi-layer feedforward networks will be discussed and investigated.

3.2. The Capacity of a Multi-Layer Feedforward Network

According to the theoretical framework of chapter 2, the generalization behavior of a parametric classifier is completely characterized by its capacity. Therefore, the problem that we will address in this section, is the determination of the capacity of the class of multi-layer feedforward networks. In order to approach this problem, we will impose some restrictions that will simplify a theoretical solution considerably. In the following, we will limit ourselves to the class of networks that consist of units with inner product activation functions and threshold output functions. Thus, if the network has a layered topology, its transfer function is described by:

$$f(\boldsymbol{x},\boldsymbol{\theta}) = out^{thr}\left(act^{ip}\left(out^{thr}\left(act^{ip}\left(.....out^{thr}\left(act^{ip}\left(\boldsymbol{x},\boldsymbol{\theta}^1 \right)\right).....,\boldsymbol{\theta}^{l-1} \right)\right),\boldsymbol{\theta}^l \right)\right) \quad (9)$$

An upper bound for the capacity of such networks is given by the following theorem of Baum and Haussler, see also [Baum 1988] and [Mitchison 1989]:

Theorem 3.1 *[Baum 1989]: Let $F^{net}(\boldsymbol{X},\boldsymbol{\Theta})$ be the class of all functions computed by multi-layer feedforward networks, with $N \geq 2$ units having inner product activation functions and threshold output functions, and with $\boldsymbol{\Theta}$ being a W-dimensional parameter space. Then the capacity V is bounded by:*

$$V \leq 2W\log(eN) \quad (10)$$

and for $m \geq W$ the number of dichotomies that the functions in $F^{net}(\boldsymbol{X},\boldsymbol{\Theta})$ can induce on a set of m points is bounded by:

$$S_{\max}^{net}(m) \leq \left(\frac{Nem}{W}\right)^W \quad (11)$$

A nice property of theorem 3.1 is that it does not depend on the actual topology of the network. The only parameters that are required to bound the capacity are the number of parameters W and the number of units N. The applicability of theorem 3.1 is therefore broader than the class of layered networks that is described by equation (9). In fact, inequalities (10) and (11) also include networks in which (some of) the connections skip one layer; they connect units that are not in two subsequent layers. Also note that, contrary to linear classifiers and quadratic classifiers, the capacity of a multi-layer network is independent of the dimensionality of the feature space.

From theorem 3.1 Baum and Haussler derive a corollary that is formulated as a lower bound on the sample size, such that the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ is with high probability larger than a fraction $(1 - \gamma)$ of the true error $\varepsilon(\boldsymbol{\theta})$, for some $0 < \gamma \leq 1$, cf. equations (2.33) and (2.35 - 2.36). As it was discussed at the end of section 2.1.2.2, this leads to a formulation in which m is expressed in terms of the unknown parameter $\varepsilon(\boldsymbol{\theta})$. This results in a "chicken and egg" problem in which a sample of size m is needed to determine the error $\varepsilon(\boldsymbol{\theta})$, but m can not be determined since $\varepsilon(\boldsymbol{\theta})$ needs to be determined first. Therefore, a better formulation is the following corollary of theorem 2.7 and theorem 3.1:

Corollary 3.1: *Let $F^{net}(\boldsymbol{X}, \boldsymbol{\Theta})$ be the class of all functions computed by multi-layer feedforward networks, with $N \geq 2$ units, having inner product activation functions and threshold output functions, and with $\boldsymbol{\Theta}$ being a W-dimensional parameter space. Furthermore let $\boldsymbol{\Xi}^m$ be an i.i.d. sample of size m, drawn according to some distribution D. In order to let the probability, that the maximum one-sided relative deviation of the true error $\varepsilon(\boldsymbol{\theta})$ from the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ on $\boldsymbol{\Xi}^m$ is larger than χ, be at most δ:*

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{\varepsilon(\boldsymbol{\theta}) - \hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} \leq \delta \tag{12}$$

the sample size m should at least be:

$$m \geq \frac{1}{\beta}\left\lceil \frac{3}{\chi^2}\frac{(\beta+1)^2}{\beta}\left(W \ln\left(\frac{2Ne}{\chi^2}\frac{(\beta+1)^3}{\beta^2}\right) + \ln\left(\frac{4}{\delta}\right)\right)\right\rceil \tag{13}$$

where β is given by:

$$\beta = z_2\left(\ln\left(\frac{2Ne}{\chi^2}\right) + \frac{1}{W}\ln\left(\frac{4}{\delta}\right)\right) \tag{14}$$

The proof of this corollary is immediate from the proof of theorem 2.7. The advantage of the formulation of corollary 3.1 over Baum and Haussler's formulation is that all parameters are known in advance or can be chosen by the user: N and W are given, and δ and χ are chosen. Apart from the difference in formulations, the lower bound on m that follows from (12 - 14) is also several factors better than can be reached by the approach of Baum and Haussler. This is because the results of chapter 2, that are used in (12 - 14) are better than the results of Vapnik as they were used by Baum and Haussler. As could be expected from theorem 3.1 though, corollary 3.1 shows that the sample size m should be linear in the number of free parameters W.

We will conclude this section with five remarks:

• It is important to realize that the bound $V \leq 2W\log(eN)$ on the capacity of a multi-layer network with W parameters and N units in theorem 3.1 is an *upper* bound. This guarantees that no set of $m = 2W\log(eN)+1$ points exists that can be labeled in all 2^m ways, and that it is possible that *some* set of $m = 2W\log(eN)$ points exists that can be labeled in 2^m ways. For the case of multi-layer networks with one hidden layer, it is also particularly easy to derive a *lower* bound on the capacity, i.e. the number of points that can *certainly* be labeled in an arbitrary way, see [Baum 1988], [Baum 1989]. To introduce the notation, consider a multi-layer network with one hidden layer of h units, d input units and one output unit. As the linear threshold functions of the hidden units represent hyper planes in the d-dimensional feature space, the parameters of a pair of hidden units can be chosen such that they enclose, with an (infinitesimally) small space in-between, exactly d points in general position, see figure 4. So, with h hidden units in this configuration it is possible to enclose an arbitrary set of $\lfloor h/2 \rfloor d$ points. Furthermore, it is possible to choose the parameters of each pair of hidden units in such a way, that the region between the hyper planes is labeled by the hidden units as 1/1, whereas the rest of the space is either labeled as 1/0 or as 0/1. Now, with the appropriate choice of the parameters of the output unit (including the threshold θ_0), it is possible to label the enclosed regions with the output unit by 0 or by 1; choosing the parameters as 1 and thresholding at $\lfloor h/2 \rfloor + 0.5$ labels the enclosed regions as 1 and choosing the parameters as -1 and thresholding at $-(\lfloor h/2 \rfloor + 0.5)$ labels them as 0. Thus, with h hidden units, it is possible to label an arbitrary set of $\lfloor h/2 \rfloor d$ points either as 0 or as 1.

Since for labeling a set of m points at most $m/2$ points have to be separated from the others, we conclude that the lower bound for the capacity is equal to:

$$V \geq 2\lfloor h/2 \rfloor d \tag{15}$$

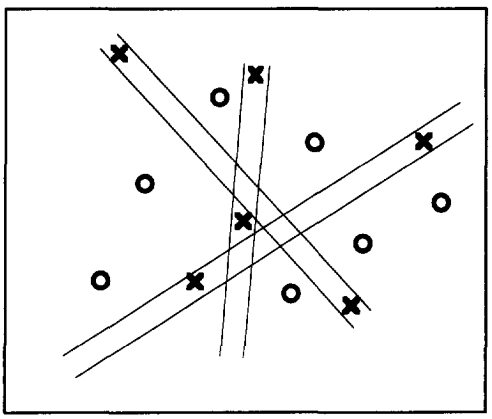


Figure 4: A slicing construction is used to partionate the feature space in areas that are labeled 1 and areas that are labeled 0. Each pair of (hyper)planes in $\Re^d$ allows the enclosing of exactly d points. The hyper-planes corresponding to the 6 hidden units in this figure, provide a lower bound of 12 for the capacity.

It is instructive to compare the lower bound $V \geq 2\lfloor h/2 \rfloor d$ to the upper bound as presented in theorem 3.1. For the class of networks under consideration, the number of free parameters is equal to $W = h\,(d + 1) + h + 1 = h\,(d + 2) + 1$. Substitution in (10) yields:

$$V \leq (2h(d+2)+1)\log(e(h+1)) \tag{16}$$

Comparison of (15) and (16) shows that there is roughly a factor $2\log(e(h+1))$ difference between both bounds. Inequality (15), however, is a clear underestimate of the capacity, since the parameters of the output unit, as well as the parameters of half the number of hidden units are not chosen freely. Inequality (16) is likely to be an overestimate of the capacity, but this is an issue that will be addressed further in section 3.4.

• In chapter 4 it will be discussed that a number of frequently used learning procedures for multi-layer networks aim at minimizing the mean squared mapping error. In such procedures, the parameters of the network are adapted such that for the elements $(\xi_p, \lambda_p) \in \Re^d \times \{0,1\}$, $1 \leq p \leq m$, of the training set Ξ^m, the mean squared difference between the network output $f(\xi_p, \theta)$ and the desired network output λ_p is minimized. A special property of the geometrical construction as depicted in figure 4, that will play an important role in the next chapter, is that it corresponds to a global minimum of the quadratic error criterion; i.e. the parameters are chosen such that for all elements of the training set Ξ^m the relation $f(\xi_p, \theta) = \lambda_p$ holds, $1 \leq p \leq m$.

• The previous upper and lower bound for the capacity imply for a network with 203 input units, 120 hidden units and 1 output unit, i.e. a network that is comparable with the NETtalk architecture [Sejnowski 1987] shown in figure 3, that the capacity is bounded between 24,360 and 411,406. For a training sample size of less than 24,360, therefore, it is not possible to have absolute certainty that the empirical classification error has any significance. If corollary 3.1 is used with an empirical error of 0%, the sample size should at least be 9,680,453 in order to bound the true error below 10% with 99% certainty. To bound the true error below 1% with 99% certainty the sample size should even be 115,001,490. These extreme bounds for the sample size follow both from the minimax approach of the Vapnik-Chervonenkis theory, as well as from the extremely large number of free parameters in the network architecture.

• Theorem 3.1 and corollary 3.1 are both based on the assumption that the network consists of units with linear threshold functions. The case of networks with sigmoidal non-linearities has recently been discussed in the context of continuous function approximation by Haussler [Haussler 1991] and Burton [Burton 1991]. Their analyses show the same linear relation between the sample size and the capacity, although with significantly larger constants.

• A special case of the multi-layer networks that are discussed so far, are networks that consist of parameters with a finite word-length. Whereas the theory that was developed in section 2.1 adopted the number of dichotomies as a measure for the complexity, a natural measure for the case of a discrete parameter space is the *total number of bits representing the free parameters*. If we assume that the W-dimensional parameter space Θ is discrete and all parameters are represented with b bits, the total number of distinct classifiers that can be implemented is equal to 2^{bW}. Since the number

of classification functions is finite in this case, the theory of section 2.1.1 can be used to bound the maximum deviation of the true and estimated errors. From (2.7) it follows that with probability at least $1 - \delta$ the following inequality is valid for all 2^{bW} classifiers simultaneously:

$$\varepsilon(\boldsymbol{\theta}_i) \leq \hat{\varepsilon}(\boldsymbol{\theta}_i) + \frac{bW \ln 2 - \ln \delta}{m}\left(1 + \sqrt{\frac{2m\hat{\varepsilon}(\boldsymbol{\theta}_i)}{bW \ln 2 - \ln \delta}}\right) \tag{17}$$

A proper choice for b can be derived by the method of structural risk minimization [Vapnik 1982], cf. section 2.2. The latter approach is very much related to the framework of Minimum Description Length model selection, see [Barron 1991], [Baum 1988] and section 2.2. Within this framework one chooses the word-length as a function of the amount of data available, such that the parameters are not represented with unnecessary precision. The selection of the best model is based on a balance between the empirical error of the model and its description length. Also from (17) it is immediate that the best bound for the true error is given by the classifier that balances the estimated error $\hat{\varepsilon}(\boldsymbol{\theta})$ and its complexity, as quantified by b and W in the optimal way. A simple model may have a high estimated error $\hat{\varepsilon}(\boldsymbol{\theta})$, but the confidence in that estimate can be high. A more complex model probably has a much lower error $\hat{\varepsilon}(\boldsymbol{\theta})$, but with much smaller confidence. As a final remark, note that this treatment of a discrete parameter space is independent of the topology of the network, and that it is also independent of the actual type of activation and/or output function of the units.

3.3. Minimal Kernel-Based Networks*

In this section we will apply the Vapnik-Chervonenkis theory to study the generalization properties of a class of multi-layer feedforward networks, that are known as minimal kernel-based networks. Contrary to the previous section, we will discuss how the training procedure influences the capacity and thereby the generalization behavior of this type of classifier. A number of training procedures will be discussed, and an upper bound for their capacity will be derived. A comparison with the capacity of the general class of multi-layer feedforward networks, which is independent of the training procedure (cf. section 3.2), will be made, which shows a number of interesting differences between both types of classifiers.

* This section was partially published in [Kraaijveld 1991b]

3.3.1. Introduction

The standard type of multi-layer feedforward networks that are described in the neural networks literature consists of units that compute an inner product of a parameter vector and an input vector, followed by a sigmoidal output function, e.g. see [Rumelhart 1986] and section 3.2. Recently however, a number of authors have studied the classification and approximation properties of networks consisting of units that compute a distance measure between an input vector and a parameter vector, possibly followed by Gaussian shaped output function (e.g. see [Robinson 1988], [Poggio 1989], [Hartman 1990], [Park 1991]). Since the parameter vectors in such a network can be considered as kernels in the feature space, they are called *kernel-based networks*. To classify an unforeseen sample with such a network, the distance between all kernels and the sample is computed, these distances are weighed by the output function of the units, and an output unit computes a label by a (possibly weighted) summation of these weighed distances, see figure 5. A number of authors have emphasized that this type of network has many relations with well-known methods in approximation and classification theory and that their applicability is broader than the nested sigmoids scheme [Poggio 1989], cf. equation (8). Furthermore, several authors have investigated training procedures for such networks, like variants of backpropagation ([Robinson 1988], [Bishop 1991]), or other gradient descent like procedures ([Poggio 1989], [Weymare 1991]).

In this section we will analyze the generalization properties of an alternative approach to design (i.e. to train) a kernel-based network. This approach has relations to several methods from the statistical pattern recognition literature, most notably the *Parzen classifier* ([Parzen 1962], [Duda 1973], [Devijver 1982]), also known as the *Probabilistic Neural Network* ([Specht 1990]), and the *reduced Parzen classifier* [Fukunaga 1989].

A Parzen classifier, or Probabilistic Neural Network, can easily be mapped onto the network representation of section 3.1.1, see figure 5. A problem of the Parzen classifier, however, is that the size of the corresponding network may become prohibitive as the size of the training set increases. This fact distinguishes the Parzen classifier from a multi-layer feedforward network. For a multi-layer feedforward network the number of units and the network topology are fixed parameters that do not depend on the training set, and are chosen prior to the training phase. The choice for a particular number of units is generally based on a-priori knowledge of the problem at hand, or on speed or memory constraints that follow from the application.

In this section we will investigate how the step from a Parzen classifier, having a large number of kernels, to a network with a fixed (and small) number of kernels can be made, and how the Vapnik-Chervonenkis theory of chapter 2 can be applied in this context. The procedure is based on the selection of a good subset of kernels from the Parzen classifier. First, in paragraph 3.3.2 we will define the model that has been adopted for the class of networks under consideration. Then, paragraph 3.3.3 is dedicated to the selection process of the kernels, for which a number of algorithms from the statistical pattern recognition literature are used. In paragraph 3.3.4 we will apply the Vapnik-Chervonenkis theory to study the generalization behavior of this network training procedure, which is followed in paragraph 3.3.5 by a discussion of the results. Finally, the conclusions are presented in section 3.3.6.

3.3.2. A Class of Kernel-Based Networks

As in section 3.2 we will limit ourselves to a sufficiently simple model, in order to investigate the generalization properties in an analytic way. In the following we will consider networks that consist of units with a threshold output function. To be more specific, the transfer function of a unit in a kernel-based network is described by the following equation, cf. equations (2) and (4):

$$f(\boldsymbol{x},\boldsymbol{\theta}) = out^{thr}\left(act^{Ed}\left((x_1,x_2,.....,x_d),\left(\theta_0^1,\theta_1^1,\theta_2^1,.....,\theta_d^1\right)\right)\right) \tag{18}$$

In (18) the parameters $\left(\theta_1^1,\theta_2^1,.....,\theta_d^1\right)$ define the location of a hyper-spheric kernel in the d-dimensional feature space. The parameter θ_0^1 is *common to all kernels in the network* and serves as a threshold; if the distance between the input vector $\boldsymbol{x}$ and the parameter vector $\left(\theta_1^1,\theta_2^1,.....,\theta_d^1\right)$ is less than θ_0^1, the output of the unit is 1 and 0 otherwise. With (18) the transfer function of network consisting of h kernels is described by the following equation:

$$\begin{aligned} f(\boldsymbol{x},\boldsymbol{\theta}) &= out^{thr}\left(act^{ip}\left(out^{thr}\left(act^{Ed}\left(\boldsymbol{x},\boldsymbol{\theta}^1\right)\right),\boldsymbol{\theta}^2\right)\right) \\ &= out^{thr}\left(\sum_{i=1}^{h}\theta_i^2 out_i^{thr}\left(act^{Ed}\left(\boldsymbol{x},\boldsymbol{\theta}^1\right)\right)\right) \end{aligned} \tag{19}$$

The parameter vector $\boldsymbol{\theta}^2$ is an h-dimensional vector with $\theta_0^2 = 0$ and $\theta_i^2 \in \{-1,+1\}$ for $1 \le i \le h$. Equation (19) shows that a kernel based network is a voting scheme in which every kernel has a weight $\theta_i^2\ out^{thr}\left(act^{Ed}\left(\boldsymbol{x},\boldsymbol{\theta}^1\right)\right)$.

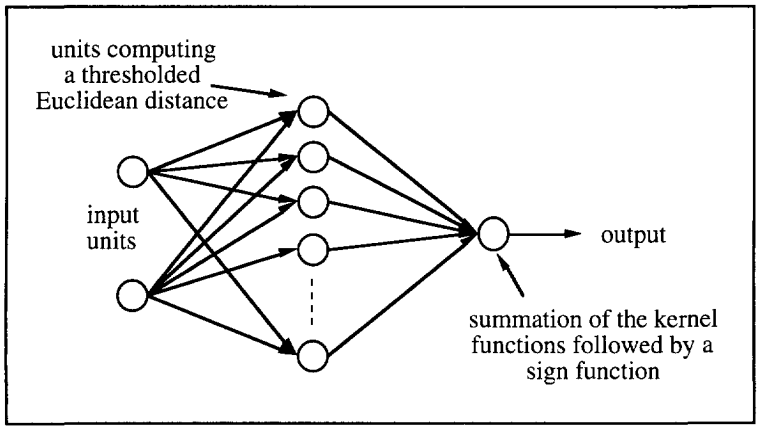


Figure 5: Schematic overview of a kernel-based network.

Equation (19) includes a class of classifiers that are known as the *Parzen classifier* ([Parzen 1962], [Duda 1973], [Devijver 1982]), or *Probabilistic Neural Network* [Specht 1990]. In fact, equation (19) describes a Parzen classifier when $h = m$ kernels are located on the position of the m samples in the training set, i.e. the parameters $\left(\theta_1^1, \theta_2^1, \ldots\ldots, \theta_d^1\right)$ are chosen as $\theta_j^1 = \xi_j$, $1 \le j \le d$, and the parameters θ_i^2 are chosen according to $\theta_i^2 = 2\left(\lambda_i - 0.5\right)$, i.e. $\theta_i^2 = +1$ when the class label is $\lambda_i = 1$ and $\theta_i^2 = -1$ when the class label is $\lambda_i = 0$, $1 \le i \le h$. In this configuration, the activation function of the output unit computes a *non-parametric Parzen estimate* of the density of the underlying class-conditional distributions. As the Bayes rule prescribes, the classifier then selects the class with the highest a-posteriori probability.

One of the attractive aspects of using a Parzen classifier is that it is provably consistent under very mild conditions. Provided that an appropriate value for the common radius of the kernels θ_0^1 is selected, the consistency of a kernel-based network can be guaranteed, since we have not violated any of these conditions for consistency with our choice for the kernel function (18) ([Parzen 1962], [Duda 1973]). Unfortunately there is no theoretical result known that predicts an optimal value for θ_0^1 given a finite set of samples, without exact knowledge of the underlying distributions. This has been subject of many previous studies, e.g. see [Koontz 1972], [Duin 1976], [Devroye 1988] and [Fukunaga 1990]. The approach that we will follow in a subsequent paragraph, is that a

radius that has a good performance on the test set is selected. First, however, we will discuss a number of methods to select a subset of kernels.

3.3.3. Minimization of Kernel-Based Networks

Although the Parzen classifier has the highly desirable feature of consistency, one can worry about the large amounts of data that are involved to store the network, and the large amounts of computation that are required to classify a sample. Especially when the training sample size m gets very large this results in considerable demands for storage and CPU time, even on modern computers. From this point of view, it is much more efficient to fix the network size in advance, since that allows a guaranteed upper bound on the speed of the classifier. However, the price that is paid for this, is that it is necessary to determine the parameters $\boldsymbol{\theta}$ for this fixed number of units, whereas for the Parzen classifier the parameters $\boldsymbol{\theta}$ are given for free. Whereas other authors describe various gradient descent procedures to determine these parameters ([Robinson 1988], [Bishop 1991], [Poggio 1989], [Weymare 1991]), we will study the approach of selecting a good subset of kernels from the Parzen classifier. A good way to perform this operation was published recently by Fukunaga [Fukunaga 1989], and is called the reduced Parzen classifier.

Fukunaga describes a method that aims at selecting a good subset of kernels to approximate the densities. A density that is originally represented by m kernels, will then be represented by a smaller number, say h kernels. The problem to be solved then is: *which subset of h out of m kernels represents the underlying density best.* The similarity function that Fukunaga proposes is the entropy of the densities approximated by m and h kernels:

$$\int \ln\left(\frac{\hat{p}_h(\boldsymbol{x})}{\hat{p}_m(\boldsymbol{x})}\right)\hat{p}_m(\boldsymbol{x})d\boldsymbol{x} \tag{20}$$

Although this problem can not be solved completely, since the number of possible subsets can be prohibitively large, Fukunaga reports very good results with an iterative, but sub-optimal, procedure to select the subsets.

An issue that Fukunaga does not mention in his paper is that for the purpose of classification (instead of density estimation) it is not necessary to represent the density of

the class with the locally lowest density, since the Bayes rule prescribes that a new sample should be assigned to the class with locally the highest density. Therefore, the reduction of the number of kernels with Fukunaga's method, could possibly be improved by first removing those kernels that represent locally the lowest density, and then by approximating the remaining kernels with a method such as the reduced Parzen classifier. We will not give a validation of this proposal, but we will discuss, instead, an algorithm to perform this task.

A well-known algorithm to remove the kernels that belong to the class with the locally lowest density is called the editing algorithm [Devijver 1982], see figure 6. A variant of the editing algorithm is called the *multi-edit algorithm* [Devijver 1982] and consists of 5 steps:

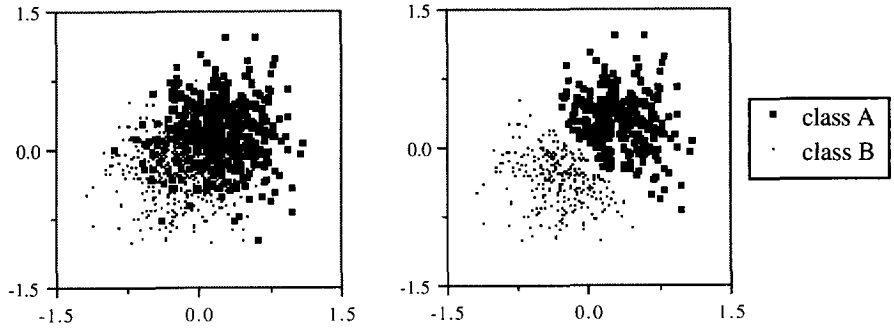


Figure 6: Network minimization by the editing technique. The left figure shows the kernels of the original learning set in the feature space. In the right figure all kernels in the overlapping part of the distributions have been removed by the editing technique.

1. Diffusion: Make a random partition of the learning set Ξ^m into S subsets $\Xi_1,.....,\Xi_S$, $S > 2$.
2. Classification: Classify the samples in Ξ_i using the Parzen classifier (i.e. a kernel-based network) using $\Xi_{(i+1) \, Mod \, S}$ as a training set.
3. Editing: Discard all samples that were misclassified at step 2.
4. Diffusion: Pool all the remaining data to constitute a new set Ξ^m.
5. Termination: If the last I iterations produced no editing then exit with the final set Ξ^m, else go to step 1.

When the density estimates of the Parzen classifier are based on a sufficiently large number of samples the variance in the density estimates approaches zero. Therefore, in step 2, the probability that a sample of the class with the lowest density will be discarded approaches one. In other words; with probability approaching one, all kernels in the overlapping part of the distributions are removed. Although these arguments are relatively straightforward, a similar discussion for editing with the nearest neighbor classifier requires a considerable mathematical treatment to arrive at a comparable statement, e.g. see [Devijver 1982]. Also, it is interesting to note that the consistency of the classifier is preserved, since only those kernels that are not relevant for the classification result are removed.

As a result, a good approach to select a subset of kernels could be: editing of the kernels with the editing algorithm, followed by an approximation of the edited densities with Fukunaga's method. A nice property of this procedure is that it is provably consistent; when the entropy of the edited densities approximated by m and h (out of m) kernels is optimized, it is evident that we can approximate the edited densities arbitrarily well with the best subset of h kernels, when $h \rightarrow \infty$. Therefore we can say that the method is consistent as long as we let $h \rightarrow \infty$, $m \rightarrow \infty$ and $h / m \rightarrow 0$.

It is also clear that the classification performance of the final network is largely based on the quality of the search process for the *best* subset of h kernels. Although we have discussed two reasonable procedures that can guide us through this search, it is clear that numerous other approaches can be adopted to solve this optimization problem.

3.3.4. Generalization Capabilities of Kernel-Based Networks

A notable difference between non-parametric classifiers such as the Parzen classifier or the k-nearest-neighbor classifier, and a parametric classifier such as a linear threshold function, is that the training set is used in the design of the classifier, i.e. for an implicit description of the classification function, and not in the determination of a set of parameters. In fact, for a non-parametric classifier the actual shape of the decision function is implicit in the training data and is never made explicit during the design of the classifier. For this reason it is not possible to apply the Vapnik-Chervonenkis theory at this stage of classifier design, since that framework is concerned with parametric classifiers only.

However, many non-parametric classifiers do have a number of parameters that have to be determined or tuned during the design of a classifier. For the k-nearest-neighbor

classifier this is the number of neighbors k that is taken into account, and for the Parzen-classifier this is the radius θ_0^l of the kernels, cf. equation (18). In the literature numerous other non-parametric classifiers are described, in which much more than one parameter has to be determined, e.g. variable metric nearest-neighbor classifiers [Fukunaga 1990], Parzen-classifiers with elliptical kernels [Fukunaga 1990], histogram rules [Devroye 1988], etc. As the determination of these parameter values is again a parametric problem, it is at this stage of the classifier design process that we can apply the Vapnik-Chervonenkis theory. The procedure is as follows: a training set of m samples is used to design the classifier, and a test set of n samples is used to tune the parameter(s). As the Vapnik-Chervonenkis theory can be used to obtain a lower bound for the size of the test set n, this approach can be used to determine how a finite amount of training data should be divided over the training set of size m and the test set of size n, see also [Devroye 1988]. In fact, the application of the Vapnik-Chervonenkis theory in this context offers an analytically tractable alternative for methods like cross-validation or leave-one-out techniques, e.g. see [Duin 1976], [Cover 1969] or [Devroye 1988].

If we return to the class of minimal kernel-based networks, as described in the previous section, we find that there are two types of parameters that have to be tuned during the design of the network. These are the subset of the kernels to approximate the densities, and the common radius θ_0^l of the kernels. The selection of the subset of kernels is not a true parameter that is chosen from a continuous domain, but merely a parameter that can have a finite number of discrete values. The radius of the kernels θ_0^l, however, is a variable in a continuous domain. In the following we will investigate two separate cases:

- The desired size of the test set n for a network with h kernels with fixed common radius θ_0^l, chosen from m kernels. This case is easy, since only a finite number of classifiers are taken into account. To be more precise, if the search procedure for a good subset of kernels is limited in advance to the investigation of at most L subsets of h kernels, the theory developed in section 2.1.1 serves as an excellent vehicle to obtain a lower bound for n. By substituting m over h for L it is even possible to remove any prior limit on L, since this corresponds to the worst-case of an exhaustive search process.

- The second problem that we investigate is the desired size of the test set n for a network with h kernels with variable common radius θ_0^l, chosen from m kernels. In this case the radius of the kernels is selected that has the best performance on the test set. For this case the following theorem on the capacity of this search procedure is presented:

Theorem 3.2: *Let $5 \leq h \leq m$ and let the set of classification functions $F^{kbn}(\boldsymbol{X}, \boldsymbol{\Theta})$ consist of all kernel-based networks that are designed by selecting a subset of h out of m kernels of which the common radius θ_0^1 can be varied. Then the capacity V is bounded by:*

$$V \leq \frac{3}{2} h \log\left(\frac{em}{h}\right) + \frac{3}{4} \log\left(\frac{h}{4}\right) \tag{21}$$

Proof: Since we can choose from infinitely many values for θ_0^1, the number of classifiers is infinitely large. We therefore proceed as in [Baum 1989] by first deriving an upper bound for the number of dichotomies $S_{\max}^{kbn}(n)$, in order to use this result to upper bound the capacity.

Suppose that a subset of h kernels is chosen. Now consider the effect on one fixed sample of the test set $(\boldsymbol{\xi}_i, \lambda_i)$ of a change in the parameters of the classification function (19), restated here:

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = out^{thr}\left(\sum_{i=1}^{h} \theta_i^2 out_i^{thr}\left(act^{Ed}\left(\boldsymbol{x}, \boldsymbol{\theta}^1\right)\right)\right) \tag{22}$$

At most h sign changes of the classification function (22) can be induced on the sample by varying the common radius θ_0^1 of the h kernels, see also [Devroye 1988]. For n samples in the test set we therefore conclude that the number of dichotomies with h kernels is less than or equal to $(nh + 1)$. Since at most m over h subsets of h kernels can be chosen, the upper bound for the number of dichotomies for the resulting network is:

$$S_{\max}^{kbn}(n) \leq (nh+1)\binom{m}{h} \tag{23}$$

By using Stirling's formula for the factorial (e.g. see [Mood 1974]) it follows that for $m \geq 2, h \geq 2$:

$$S_{\max}^{kbn}(n) \leq (nh+1)\frac{m^h}{h!} \leq \frac{(nh+1)}{\sqrt{2\pi h}}\left(\frac{em}{h}\right)^h \leq \frac{1}{2} n\sqrt{h}\left(\frac{em}{h}\right)^h \tag{24}$$

Inequality (24) can be expressed in the more compact form:

$$S_{\max}^{kbn}(n) \leq an \tag{25}$$

with:

$$a = \frac{1}{2}\sqrt{h}\left(\frac{em}{h}\right)^{h} \tag{26}$$

From (24 - 26), an upper bound for the capacity V can be derived by searching for the sample size n for which the number of dichotomies makes a transition from an exponential in n to a polynomial in n, cf. definition 2.1 and theorem 2.2. Thus, the capacity V can be found by demanding:

$$S_{\max}^{kbn}(n) < 2^{n} \tag{27}$$

We hypothesize that (27) is true for:

$$n = \frac{3}{2}\log(a) \tag{28}$$

Substitution of (25) and (28) in (27) yields:

$$\sqrt{a} > \frac{3}{2}\log(a) \tag{29}$$

Both sides of (29) are monotonic functions of a. Furthermore, it is easily shown that for a larger than 98.8, the left side increases faster than the right side. With $5 \leq h \leq m$ in (26), the minimum value of a is larger than 165, what implies that (29) is true, and that it is also true for larger values of a. Thus, the upper bound for the capacity follows from substitution of (26) in (28):

$$V \leq \frac{3}{2}h\log\left(\frac{em}{h}\right) + \frac{3}{4}\log\left(\frac{h}{4}\right) \tag{30}$$

This concludes the proof of theorem 3.2.

■

3.3.5. Discussion

It is instructive to compare the result of the previous paragraph with the result of Baum and Haussler, as presented in section 3.2. A nice aspect in this comparison is that the bound $V \leq 2W\log(eN)$ (cf. inequality (10)) of Baum and Haussler for a network with N linear threshold units and W parameters is also valid for a network consisting of units with hyper-spheric kernels. This is because the capacity of a linear threshold function in $\Re^d$ is equal to the capacity of a hyper-spheric kernel in $\Re^d$; i.e. d + 1 [Blumer 1989].

If we consider a network with h hyper-spheric kernels and we train it with one of the training procedures of section 3.3.3, the capacity is upper bounded by $V \leq 3/2\ h\log(em/h) + 3/4\ \log(h/4)$, cf. inequality (21). Note that the total number of units in such a network is equal to $h + 1$, i.e. h kernels and one output unit. In the training procedures, the parameters of the output unit are fixed, the locations of the kernels are chosen from a larger set of m prototypes, and the radius of the kernels is the only continuous free parameter to determine. If the training procedure is changed, such that all parameters can be given arbitrary values, the capacity is bounded by $V \leq 2W\log(eN)$. The difference between these bounds is roughly a factor d, since the number of parameters in a network with one hidden layer of N units is roughly equal to $W \approx Nd$. This relatively large factor is not surprising, since the number of free parameters is much larger.

A difference between both types of networks, however, is that for the training procedure of the kernel-based network, first a training set is needed to construct a Parzen classifier and then a separate test set is used to select the best subset of kernels and to tune the radius of the kernels. As was discussed in the previous paragraph, a totally parametric classifier, like a multi-layer feedforward network of section 3.2, does not require a data set to construct the classifier. Only one data set is needed, and is used for the process of training the classifier.

A relevant issue now is whether the training set of a multi-layer feedforward network is larger/smaller than the combination of the training set *plus* the test set for a kernel based network. In any case, theorems 3.1 and 3.2 show that the training set of the multi-layer feedforward network should be much larger than the test set of the kernel based network. The size of the training set of the kernel-based network, however, depends on the complexity of the underlying distributions. If the distributions in the classification problem can easily be approximated with a modest number of kernels, only a small training set is required; e.g. it is not hard to imagine that one can find a reasonable

approximation of a multi-variate normal density with a moderate number of kernels. More complex densities obviously require a larger training set.

An additional advantage of using a kernel-based network, however, is that there are learning procedures available that are provably consistent, cf. section 3.3.3. Clearly, this is a property that is (still) lacking for nested sigmoid networks, that are trained with the backpropagation algorithm.

We finish this section with the remark that the choice for a hyper-spheric kernel in the definition of the kernel function (18) was mainly made for ease of understanding. In [Devroye 1988] it is argued that a kernel function may be the indicator function of any star-shaped set. A set S is star shaped if $x \notin S$ implies that $cx \notin S$ for all $c \geq 1$. The notion of the radius will in this case be replaced by the notion of a scaling factor.

3.3.6. Conclusions

In this section we have studied the generalization properties of the class of minimal kernel based networks, in the context of the Vapnik-Chervonenkis theory. This class of networks has the advantage of low memory and speed requirements and the availability of consistent learning procedures. A number of such training procedures from the statistical pattern recognition literature were discussed and it was shown that their capacity is upper bounded by $V \leq 3/2\ h\log(em/h) + 3/4\ \log(h/4)$. This bound can be used to predict for a minimal kernel based network (and also for the reduced Parzen classifier and the editing algorithm) how a set of training data should be divided over a training set and a test set. Finally, it was shown that the capacity of this class of networks is smaller than the capacity of a regular feedforward network, and it was discussed under which conditions this is advantageous with respect to the requirements on the training data.

3.4. Feedforward Networks with Shared Parameters*

As was discussed in chapter 1, it is a well-known fact that the number of free parameters of a classification function should not be too large, since the parameters have to be estimated from a finite learning set. For the class of multi-layer feedforward network classifiers that was discussed in section 3.2, this implies that the number of parameters and/or units should be limited. However, a fundamentally different approach to decrease the number of free parameters in such networks, suggested by Rumelhart

* This section was partially published in [Kraaijveld 1992].

[Rumelhart 1986] and applied by le Cun ([le Cun 1989], [le Cun 1990a]), is by sharing the same parameters with multiple units.

In this section we will discuss how this parameter sharing technique influences the capacity of the network. First, an upper bound will be derived for the number of dichotomies that can be induced with a layer of units with shared parameters. Then, we will apply this result to bound the capacity of a simple class of parameter-sharing networks. The results show that the capacity of a network with shared parameters is linear in the number of free parameters. Another, somewhat remarkable, outcome is that either that the parameter sharing technique is an effective way of decreasing the capacity of a network, or that the existing bounds for the capacity of multi-layer feedforward networks overestimate the capacity.

3.4.1. Introduction

According to the mathematical framework of chapter 2, a small capacity of a classifier results in low requirements with respect to the training sample size and in high confidence in the classification performance of a classification function. The lesson of this theory for the application of feedforward multi-layer networks, is that the complexity of the network should not be too large, in order to achieve valid generalization. As was discussed in section 3.2, the capacity of a multi-layer feedforward network consisting of units with linear threshold function is upper bounded by $V \leq 2W\log(eN)$ [Baum 1989], with W the number of free parameters in the network, N the number of units and e the base of the natural logarithm, cf. inequality (10). Therefore, the obvious approach to limit the capacity of the network is by decreasing the number of parameters (and/or units).

A fundamentally different approach, proposed by Rumelhart [Rumelhart 1986] and applied by le Cun ([le Cun 1989], [le Cun 1990a]), is to limit the number of free parameters in the network by sharing some of the parameters by multiple basis functions. This approach was motivated by the fact that translation invariance of the classifier could be introduced by sharing the parameters in the lower layers of the network. Although this indeed reduces dramatically the number of free parameters in the network, it was not investigated how this parameter sharing technique affects the capacity of the network. This is a relevant question since the key parameter in the Vapnik-Chervonenkis theory is the capacity, and not the number of free parameters.

In this section, it will be investigated how the parameter sharing technique influences the capacity of a network. This is done in the following way. In section 3.4.2, the implementation of the parameter-sharing technique is discussed. This is followed by the derivation of an explicit upper bound for the capacity of a network with shared parameters

in section 3.4.3. Finally, a discussion of the results and the conclusions follow in sections 3.4.4 and 3.4.5 respectively.

3.4.2. Parameter-Sharing in Multi-Layer Feedforward Networks

A direct conclusion from theorem 2.7 and theorem 3.1 is that the number of free parameters of a multi-layer feedforward network should be sufficiently small, as to let the network be over-determined by the learning set. On the other hand, a too small network will not always be able to approximate a complex decision function. The choice for a specific architecture of a network is therefore a non-trivial, and sometimes critical, problem.

For some specific problems, however, a priori knowledge can serve as a guideline to choose a network architecture. A good example is the character recognition problem as described by le Cun ([le Cun 1989], [le Cun 1990a]). In this application, characters are offered to the network as an array of pixels. The recognition of the characters is based on the extraction of local features in this array of pixels, of which the exact location is not known. Le Cun suggests decreasing the number of parameters in the network by connecting the hidden units of the network with only a small fraction of the inputs, and to share the same parameters with a large number of units, see figure 8. This introduces a functionality that is roughly comparable to a convolution. The backpropagation algorithm [Rumelhart 1986] is then used to learn the convolution coefficients from the learning set. As in section 3.1, the transfer function of such a network is described as, cf. equation (8):

$$f^*(\boldsymbol{x},\boldsymbol{\theta}) = out^{sigm}\left(\sum_i \theta_i^l out_i^{sigm}\left(\sum_j \theta_j^{l-1} out_j^{sigm}\left(\ldots\ldots\sum_k \theta_k^l x_k\right)\ldots\ldots\right)\right) \qquad (31)$$

The difference between (8) and (31) is that for a layer with shared parameters, the parameters θ_{ij} with index j of unit i are equivalent; i.e. the following equality holds for the parameters of all units in that layer:

$$\forall p,q,j:\ \theta_{pj}^k = \theta_{qj}^k \qquad (32)$$

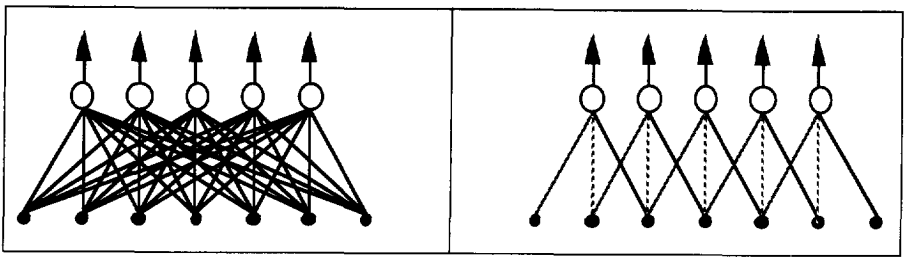


Figure 8: The parameter sharing technique. In the left figure a layer of a network with a fully connected topology is sketched; every hidden unit is connected to all inputs and every parameter of every connection is a free parameter. In the right figure, every hidden unit is connected to only three inputs. To introduce translation invariance, every hidden unit uses the same three parameter values as the other hidden units.

3.4.3. The Capacity of Networks with Shared Parameters

In this paragraph it will first be investigated how the parameter sharing technique influences the number of dichotomies that can be induced in a network layer with shared parameters. The results can be used to estimate the capacity of various network architectures of which (some of) the layers have shared parameters. Then, we will consider one architecture in more detail and we will give an explicit upper bound for its capacity.

As in section 3.2, we will limit ourselves to the case of networks with linear threshold functions. The transfer function of the network is therefore given by:

$$f(\boldsymbol{x},\boldsymbol{\theta}) = out^{thr}\left(\sum_i \theta_i^l out_i^{thr}\left(\sum_j \theta_j^{l-1} out_j^{thr}\left(\ldots\ldots\sum_k \theta_k^1 x_k\right)\ldots\ldots\right)\right) \qquad (33)$$

Here equation (32) holds in a layer with shared parameters. As a starting point, we restate Cover's bound for the number of dichotomies $S_{\max}^{ltf}(m,d)$ that can be induced on m points, with a linear threshold function in d-dimensional space, cf. theorem 2.8 [Cover 1965]:

$$S_{\max}^{ltf}(m,d) \leq 2\sum_{i=0}^{d}\binom{m-1}{i} \tag{34}$$

To bound the number of dichotomies that can be induced by a network layer with shared parameters, we follow an approach related to [Baum 1988]. Consider a (small part of a) network, consisting of two hidden units in a three dimensional space, see figure 9. Both hidden units are connected to two of the three inputs and the parameters for both units are shared. Since, there are two hidden units in this simple case, every point is labeled with a four-valued label; it can be 00, 01, 10, or 11. The problem now is to determine in how many ways m points can be labeled by changing the parameter values w_1 and w_2. The solution is found in the observation that by changing w_1 and w_2, the status of the hidden layer will change, whenever either in the i_1-i_2 plane or in the i_2-i_3 plane a point is crossed. Therefore, the number of different labelings of m points is equal to the number of dichotomies that a linear threshold function can induce on a set of $2m$ points in a 2-dimensional space. According to inequality (34) this is less than $S_{\max}^{ltf}(2m,2)$. This is easily generalized to the case of h hidden units with shared parameters, each having a receptive field of r inputs. We have thus proven the following lemma:

Lemma 3.1: *A layer of h hidden units with shared parameters, each having a receptive field of r inputs, can label a set of m points on at most* $S_{\max}^{ltf}(hm,r)$ *different ways.*

Since it is a reasonable assumption that only a few of the layers in a network have shared parameters, it is not possible to upper bound the number of dichotomies of a complete network with this lemma. However, by using this lemma together with the results of Baum ([Baum 1988], [Baum 1989]), the number of dichotomies of every feedforward network can be bounded, either with or without parameter sharing.

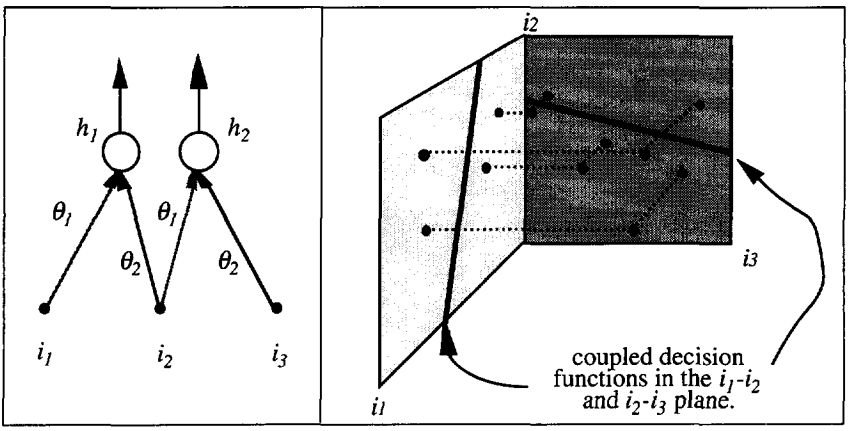


Figure 9: Parameter sharing in a simple network. In the left figure the inputs and the first hidden layer are shown. Instead of one free parameter per connection, there are only two free parameters: θ_1 and θ_2. In the right figure it is shown how points in the three dimensional feature space are projected on the i_1-i_2 plane for the first hidden unit, and projected on the i_2-i_3 plane for the second hidden unit. The decision functions in the i_1-i_2 and i_2-i_3 planes are coupled because the parameters are shared.

To gain a little more insight into the nature of the parameter sharing approach we will work this out for a simple network architecture, consisting of one hidden layer with shared parameters followed by a regular output layer with one output unit, see figure 10.

To bound the number of dichotomies of the complete network we proceed as follows. According to lemma 3.1 a set of m points can be labeled on at most $S_{\max}^{ltf}(hm,r)$ ways by the hidden layer. When the m points are mapped onto the hidden layer, the output layer can label these m points in the h dimensional hidden unit space on $S_{\max}^{ltf}(m,h)$ ways, according to inequality (34). The total number of dichotomies of the network $S_{\max}^{psn}(m)$ is therefore bounded by:

$$S_{\max}^{psn}(m) \leq S_{\max}^{ltf}(hm,r) S_{\max}^{ltf}(m,h) \tag{35}$$

A quick and rather loose way to bound the capacity of this network is the following. Using theorem 2.3, $S_{\max}^{psn}(m)$ is smaller than:

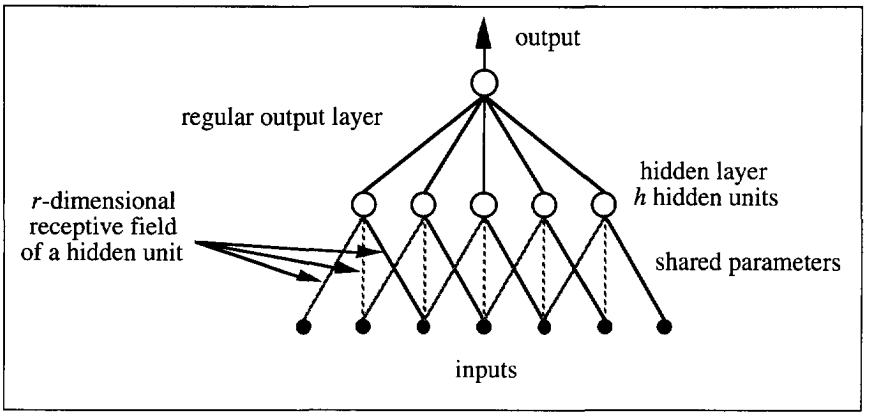


Figure 10: The class of networks for which the capacity is determined

$$S_{\max}^{psn}(m) \leq \left(\frac{ehm}{r+1}\right)^{r+1}\left(\frac{em}{h+1}\right)^{h+1} \leq \left(\frac{ehm}{r+1}\right)^{r+1}\left(\frac{ehm}{h+1}\right)^{h+1} \tag{36}$$

Now, by using a similar approach as in [Baum 1989] this can be bounded by:

$$S_{\max}^{psn}(m) \leq \left(\frac{2ehm}{W}\right)^{W} \tag{37}$$

for all $m \geq W$, where W is equal to the number of free parameters in the network: $W = r + h + 2$. As was discussed in section 3.3, an upper bound for the capacity can be found by searching for the sample size m for which the number of dichotomies makes a transition from an exponential in m to a polynomial in m, cf. inequality (27). Also using the results of Baum [Baum 1989] it follows directly that the sample size for which $S_{\max}^{psn}(m) < 2^m$ is equal to $m \geq 2W\log(2eh)$. Hence, an upper bound for the capacity of this class of networks is: $V \leq 2W\log(2eh)$. This result shows that the capacity of the class of networks under consideration is indeed linear in the number of free parameters.

Since the second inequality in equation (36) is rather loose, it is possible to improve this bound. The price for this better bound, however, is its more limited applicability. Starting with equation (36) we now proceed as follows:

$$S_{\max}^{psn}(m) \leq \left(\frac{ehm}{r+1}\right)^{r+1}\left(\frac{em}{h+1}\right)^{h+1} = h^{r+1}\left(\frac{em}{r+1}\right)^{r+1}\left(\frac{em}{h+1}\right)^{h+1} \leq h^{r+1}\left(\frac{2em}{W}\right)^{W} \tag{38}$$

Again, the capacity is bounded by searching for the value of m for which $S_{\max}^{psn}(m) < 2^m$. We hypothesize that this is true for $m \geq 2W\log(2e) + (r+1)\log(h)$.* Thus, it should be proved that:

$$h^{r+1}\left(\frac{4We\log(2e) + 2e(r+1)\log(h)}{W}\right)^W < 2^{2W\log(2e)+(r+1)\log(h)} \tag{39}$$

Taking logarithms and removing equal terms reduces this to:

$$\log\left(\frac{4We\log(2e) + 2e(r+1)\log(h)}{W}\right) < \log\left(4e^2\right) \tag{40}$$

Therefore:

$$\frac{(r+1)\log(h)}{W} < 2\left(e - \log(2e)\right) \approx 0.551174 \tag{41}$$

Which is certainly the case if:

$$2(r+1)\log(h) < W = r + h + 2 \tag{42}$$

From (42) it follows that:

$$r < \frac{h + 2 - 2\log(h)}{2\log(h) - 1} \tag{43}$$

Which is true whenever:

$$r < \frac{h}{2\log(h)} - 1 \tag{44}$$

Thus, we have proven the following theorem:

---

* We have chosen to use the notation $V \leq 2(r+h+2)\log(2e) + (r+1)\log(h)$ instead of $V \leq 5W + (r+1)\log(h)$ in order to ease the comparison with the bound $V \leq 2W\log(eN)$.

Theorem 3.3: *Let $F^{psn}(X, \Theta)$ be the class of all functions computed by feedforward networks consisting of one layer of h hidden units with shared parameters, each having a receptive field of r inputs, and one regular output unit. Then the capacity V is bounded by:*

$$V \leq 2(r+h+2)\log(2e)+(r+1)\log(h) \tag{45}$$

for all

$$r < \frac{h}{2\log(h)} - 1 \tag{46}$$

The restriction that is imposed by inequality (46) is only problematic for the case of small h. For larger values of h, the size of the receptive field r can increase almost linearly with the number of hidden units. In the case of the character recognition problems described by le Cun, inequality (46) does not seem to impose serious restrictions. For example, in the experiments described in [le Cun 1990a], the first hidden layer contains $h = 576$ units, each with 25 shared parameters. For 576 hidden units, inequality (46) demands that $r \leq 30$, which is indeed the case. In [le Cun 1989] a layer of 64 hidden units, each with 9 shared parameters is used. Theorem 3.3 would therefore not be applicable, and one would have to use the bound $V \leq 2W\log(2eh)$, as derived from inequality (37). The necessity to use this looser bound implies that, for a given sample size m, the confidence in the empirical performance is lower, cf. theorem 2.5 and theorem 2.6. A large hidden layer of 1024 units would allow a receptive field of $r \leq 50$, which roughly corresponds to a window of 7 by 7 pixels.

3.4.4. Discussion

It is illustrative to compare the bounds $V \leq 2(r+h+2)\log(2e)+(r+1)\log(h)$ for a network with shared parameters, with the bound $V \leq 2W\log(eN)$, as derived by Baum [Baum 1989]. However, it should be realized that the comparison of different upper bounds is a non-trivial problem since the bounds might not be equally tight. On the other hand, the two bounds that are compared here can be considered as the tightest bounds available. Also, the techniques and assumptions that were made in the derivation of both bounds were equal, e.g. the steps that were taken in the derivation of inequalities (35), (36) and (37).

Table 3.1 gives some figures that illustrate the behavior of both bounds. The table shows the effect on the number of free parameters W and the capacity V when a regular

feedforward network architecture with one fully connected hidden layer (cf. figure 8) is replaced by a parameter-sharing network with an architecture as depicted in figure 10. The results suggest that the capacity decreases faster than the number of free parameters. This can be explained in two different ways.

• In the first place, the parameter-sharing approach could indeed be an effective way of using the degrees of freedom in the network. Despite a large number of degrees of freedom, the parameter-sharing topology of the network prevents the classification function from labeling too many points in an arbitrary way, which then by definition implies a low capacity.

A relevant remark in this context is that the number of dichotomies of a layer of parameter-sharing units $S_{\max}^{ltf}(hm,r)$ is clearly overestimated in lemma 3.1. This overestimate is due to the fact that it is not taken into account that the hidden units may share a large number of inputs in their receptive fields. This suggests that the upper bound $V \leq 2(r+h+2)\log(2e)+(r+1)\log(h)$ also overestimates the capacity. However, a lower value of the capacity strengthens the idea that the parameter-sharing approach is an effective way of using the degrees of freedom in the network.

• The second explanation of the numbers in table 3.1 could be that the bound $V \leq 2W\log(eN)$, as derived by Baum and Haussler, overestimates the capacity. This explanation would be more important than the previous, since this bound is used for a much wider class of networks. Of course, it is not surprising that Baum's result *is* an overestimate, since its derivation is based on the repeated application of various upper bounds. However, our results might indicate that it is possible to improve the bound with a few factors.

3.4.5. Conclusions

Sharing the parameters in a multi-layer feedforward network is a way to lower the number of degrees of freedom of the network. For recognition problems in which translation invariant data has to be extracted from the input space this is advantageous, since most of the discriminatory information can be extracted, whereas only a relatively small number of free parameters is required. This implies that a good performance can be reached with a relatively small amount of training data.

Table 1: A comparison of the effect of parameter sharing on the number of free parameters and on the capacity. For various values of the dimensionality of the input space d, and number of hidden units h, the decrease of the number of free parameters W and the capacity V is shown, when the regular feedforward network architecture with one hidden layer, is replaced by a parameter sharing network with an r-dimensional receptive field. The value of r is chosen such that inequality (46) is fulfilled.

Network Architecture		Regular Network		Parameter-Sharing Network			Improvement ratio for:	
d	h	W	V	r	W	V	W	V
16	64	1153	17215	2	68	351	16.96	49.16
				4	70	372	16.47	46.28
	256	4609	87095	2	260	1295	17.73	67.30
				4	262	1320	17.59	65.98
				9	267	1385	17.26	62.91
	1024	18433	421899	2	1028	5053	17.93	83.51
				4	1030	5082	17.90	83.02
				9	1035	5157	17.81	81.82
64	64	4225	63080	2	68	351	62.13	180.12
				4	70	372	60.36	169.58
	256	16897	319297	2	260	1295	64.99	246.71
				4	262	1320	64.49	241.90
				9	267	1385	63.28	230.64
	1024	67585	1546900	2	1028	5053	65.74	306.18
				4	1030	5082	65.62	304.39
				9	1035	5157	65.30	300.00
				16	1042	5261	64.86	294.06
				25	1051	5395	64.31	286.75
				49	1075	5752	62.87	268.94
256	64	16513	246542	2	68	351	242.84	703.99
				4	70	372	235.90	662.79
	256	66049	1248105	2	260	1295	254.03	964.38
				4	262	1320	252.10	945.55
				9	267	1385	247.37	901.55
	1024	264193	6046904	2	1028	5053	257.00	1196.89
				4	1030	5082	256.50	1189.88
				9	1035	5157	255.26	1172.70
				16	1042	5261	253.54	1149.48
				25	1051	5395	251.37	1120.93
				49	1075	5752	245.76	1051.31
1024	64	65665	980387	2	68	351	965.66	2799.45
				4	70	372	938.07	2635.61
	256	262657	4963335	2	260	1295	1010.22	3835.06
				4	262	1320	1002.51	3760.18
				9	267	1385	983.73	3585.19
	1024	1050625	24046922	2	1028	5053	1022.01	4759.71
				4	1030	5082	1020.02	4731.83
				9	1035	5157	1015.10	4663.53
				16	1042	5261	1008.28	4571.16
				25	1051	5395	999.64	4457.64
				49	1075	5752	977.33	4180.77

In this section we have presented some methods and results that can be used to bound the capacity and the number of dichotomies of a network with shared parameters. This can be used in the framework of chapter 2 to quantify, in a worst case sense, how over-determined such a network is by a learning set of a certain size.

To gain some more understanding of the parameter sharing technique, the methods have been applied to the class of networks consisting of one layer with shared parameters, followed by a regular output layer. For this simple class of networks it was shown that the capacity is bounded by $V \leq 2(r+h+2)\log(2e)+(r+1)\log(h)$ for $r < h/2\log(h)-1$. The comparison with the bound for the capacity of a regular feedforward network $V \leq 2W\log(eN)$ indicates that there is an additional bonus for using the parameter-sharing technique, since the bound for the capacity decreases faster than the number of free parameters.

IV

Generalization and Iterative Learning

The elegance of the Vapnik-Chervonenkis theory is partly based on the fact that it is independent of the learning procedure that is used to train a classifier. However, in this chapter it will be discussed that the application of a specific learning rule may introduce a completely different view upon the generalization behavior of a classifier. This appears to be especially relevant for multi-layer feedforward network classifiers, that are trained with the backpropagation algorithm. In this chapter we will therefore investigate the influence of the iterative learning procedure on the generalization properties of multi-layer feedforward network classifiers.

4.1. Introduction

As was discussed in the previous chapters, the Vapnik-Chervonenkis theory reveals a number of generic properties in pattern recognition. Among these are the relations between the complexity of a classification function, the training sample size and the probability of maximum (relative) deviation of the true and estimated errors. Furthermore, it was shown that an essential requirement for valid generalization, is that the training sample size m is several (tens of) factors larger than the capacity of the set of classification functions V, in order to let the classifiers be over-determined by the training set. In fact, the Vapnik-Chervonenkis theory predicts that for a training sample size that is smaller than the capacity, the estimated errors may be totally insignificant, since the maximum (relative) deviation of the true and estimated errors can not be bounded below one. This finding is not surprising, as it is in accordance with results of many other authors, cf. chapter 1.

However, despite this large body of well-founded results on the relation between sample size and classifier complexity, a huge number of experiments have been reported in the literature on connectionist models, that seem to violate the fundamental requirement $m > V$. For example, in chapter 3 it was discussed that the sample size that is required to

train the NETtalk network, having 27,627 free parameters, should be in the order of 10^8 to 10^9 to have a reasonable confidence in the performance on the training set. However, in his original paper on NETtalk, Sejnowski describes that the network was trained with the back-propagation algorithm using not more than 5000 samples, and that the performance of the network on an independent test set was in the order of 90% to 95% [Sejnowksi 1987]. Apparently, it was possible to determine 27,627 free parameters by 5000 samples only, in such a way that the performance on the learning set actually did have some significance. In many other publications in the literature similar effects are reported; with a sample size that is a fraction of the number of free parameters, very good results are reported, cf. table 1. A surprising conclusion that we must draw from these results, is that either something is missing or wrong in the mathematical machinery that is currently used to describe the relation between sample size and classifier complexity, or that multi-layer networks and/or the back-propagation algorithm do possess some unexpected "magical" properties. The latter explanation, though apparently attractive for many novice users, does not seem to be a good approach for a Ph.D.-thesis. Instead, we will concentrate on the first explanation, and try to indicate which parts of the theory are missing, and which conditions allow results such as in table 1.

One of the claims of this chapter is that the iterative learning procedure is the missing part in the theory, needed to explain the results of table 1. Therefore, in this chapter, we will investigate a number of issues that are related to the use of such iterative learning procedures. First, in section 4.2, we will derive and discuss some properties of the back-propagation algorithm, as it plays an important role in the rest of this chapter. Furthermore, in section 4.3, some theoretical and experimental facts concerning the influence of an iterative learning procedure on the generalization properties of a classifier will be investigated.

4.2. Training Procedures for Multi-Layer Networks

As was stated in section 3.1, a large part of the world-wide interest in multi-layer feedforward network classifiers is motivated by the development of a learning algorithm for such classifiers. This algorithm is known as the back-propagation algorithm. The back-propagation algorithm has been invented several times in the recent history (e.g. [Werbos 1974], [Parker 1985], [le Cun 1985]), but has become widely known after the publication of the "Parallel Distributed Processing" books by Rumelhart and McClelland [Rumelhart 1986], [McClelland 1986]. In the following sections, we will first present a brief discussion of the algorithm, followed by the description of a number of

experiments, that illustrate some relevant issues on the relation between sample size and classifier complexity.

Table 1: Some experiments in the literature on connectionist models in which the sample size m is considerably smaller than the number of free parameters W. The performance on an independent test set is shown in the last column.

Reference	Application	W	m	Performance
[Sejnowski 1986]	Mapping text to phonemes	> 20,000	≈ 5,000	> 95%
[Golomb 1991]	Discriminate sex of human faces	> 36,000	90	91.9% (better than humans)
[Tesauro 1990], [Hertz 1991]	Back-gammon	> 11,000	3000	world-champion computer program
[Pomerleau 1989]	Control of an autonomous vehicle	>36,000	1200	> 90% (better than any other algorithm)
[Gorman 1988a], [Gorman 1988b]	Sonar target recognition	1,105	192	84.7% (nearest neighbor classifier 82%)
[Sato 1991]	Optical Character Recognition	>10,900	5000	94.5%

4.2.1. The Backpropagation Algorithm

The back-propagation algorithm is an iterative learning procedure for multi-layer feedforward network classifiers, that consist of units with a differentiable output function. The algorithm can be considered as a generalization of the Widrow-Hoff rule for single-layer networks [Widrow 1985], [Rumelhart 1986]. As the Widrow-Hoff rule, the back-propagation algorithm adapts the parameters of the network, such that the mean squared mapping error is minimized. In the context of pattern classification the following error measure is optimized:

$$SE = SE\left(\Xi^m, \boldsymbol{\theta}\right) = \sum_{p=1}^{m} SE_p\left(\boldsymbol{\xi}_p, \boldsymbol{\theta}\right) = \frac{1}{2}\sum_{p=1}^{m}\left(\lambda_p - f^*\left(\boldsymbol{\xi}_p, \boldsymbol{\theta}\right)\right)^2 \tag{1}$$

Thus, the squared differences of the network output $f^*\left(\boldsymbol{\xi}_p, \boldsymbol{\theta}\right)$ and the numerical value that is associated with the class label λ_p are summed up over all samples $\left(\boldsymbol{\xi}_p, \lambda_p\right) \in \Re^d \times \{0,1\}$ of the training set Ξ^m. Note that the network transfer function is denoted $f^*()$, as the network has a continuous transfer function, cf. section 3.1. If we assume, without loss of generality, that the multi-layer network consists of units with a sigmoidal non-linear output function, the transfer function of the network can be written as, cf. equation (3.8):

$$f^*(\boldsymbol{x}, \boldsymbol{\theta}) = out^{sigm}\left(\sum_i \theta_i^l out_i^{sigm}\left(\sum_j \theta_j^{l-1} out_j^{sigm}\left(\ldots\ldots\sum_k \theta_k^1 x_k\right)\ldots\ldots\right)\right) \tag{2}$$

The training phase of a network is initialized by assigning small random values to the parameters $\boldsymbol{\theta}$ of the network transfer function (2). Then, the backpropagation algorithm adapts the parameters, according to an approximation to a first-order gradient descent in the error space defined by (1)*:

$$\Delta_p \theta_{ij}^k = -\eta \frac{\partial SE_p}{\partial \theta_{ij}^k} \tag{3}$$

In (3), θ_{ij}^k represents the parameter associated with input j of unit i in layer k, and η denotes the *learning-rate* or step-size of the adaptations of the parameters. Note that according to (3) the change in the parameters is such that only the error SE_p due to sample p is minimized. When the adaptations of the parameters are small enough, however, the average direction of the adaptations of the parameters is equal to the gradient $-\partial SE / \partial \theta_{ij}^k$. Proceeding from (3), the backpropagation rule is now derived as follows [Rumelhart 1986], cf. section 3.1.1:

$$-\eta \frac{\partial SE_p}{\partial \theta_{ij}^k} = -\eta \frac{\partial SE_p}{\partial act_{pi}^{ip}} \frac{\partial act_{pi}^{ip}}{\partial \theta_{ij}^k} = -\eta \frac{\partial SE_p}{\partial act_{pi}^{ip}} out_{pj}^{sigm} = \eta \delta_{pi} out_{pj}^{sigm} \tag{4}$$

---

* In the following, the arguments of the functions $SE\left(\Xi^m, \boldsymbol{\theta}\right)$, $act^{ip}(\boldsymbol{x}, \boldsymbol{\theta})$ and $out^{sigm}(x)$ are omitted for brevity.

For the step from the second part to the third part of (4), equation (3.1) was used. The term out_{pj}^{sigm} denotes the input coming over the input with index j; i.e. either an element of the input vector ξ_p, or the output of another unit. The term $\delta_{pi} = -\partial SE_p / \partial act_{pi}^{ip}$ that is associated with unit i, is of special interest. For an output unit, equation (1) can be used to derive δ_{pi} as:

$$\delta_{pi} = -\frac{\partial SE_p}{\partial act_{pi}^{ip}} = -\frac{\partial SE_p}{\partial out_{pi}^{sigm}} \frac{\partial out_{pi}^{sigm}}{\partial act_{pi}^{ip}} = \left(\lambda_p - out_{pi}^{sigm}\right) out'^{\,sigm}_{pi} \tag{5}$$

Here, $out'^{\,sigm}_{pi}$ denotes the derivative of the sigmoidal output function (3.5). For a hidden unit δ_{pi} can be derived as:

$$\begin{aligned} \delta_{pi} &= -\frac{\partial SE_p}{\partial act_{pi}^{ip}} = -\frac{\partial SE_p}{\partial out_{pi}^{sigm}} \frac{\partial out_{pi}^{sigm}}{\partial act_{pi}^{ip}} = -\sum_r \left(\frac{\partial SE_p}{\partial act_{pr}^{ip}} \frac{\partial act_{pr}^{ip}}{\partial out_{pi}^{sigm}} \right) \frac{\partial out_{pi}^{sigm}}{\partial act_{pi}^{ip}} \\ &= \sum_r \left(\delta_{pr} \theta_{ri}^{k+1}\right) out'^{\,sigm}_{pi} \end{aligned} \tag{6}$$

For the step from the third part to the fourth part of (6) the chain rule is applied, to express the dependence of δ_{pi} on the activation of those units in a subsequent layer that unit i is connected to. With (5) and (3.1), this is rewritten in the final step as a function of the δ_{pr} of the units in the subsequent layer. Thus, when the δ_{pi} of the output units are known, the δ_{pi} of the units in the preceding layer can be computed, whereafter the δ_{pi} of the units in the next preceding layer can be computed, etc. This backward propagation of the δ_{pi} through the network, explains the name *backpropagation*. Collecting results, the backpropagation learning algorithm can be expressed by the following two equations:

$$\Delta_p \theta_{ij}^k = \begin{cases} \eta\left(\lambda_p - out_{pi}^{sigm}\right) out'^{\,sigm}_{pi}\, out_{pj}^{sigm} & \text{if unit } i \text{ is an output unit} \\ \eta \sum_r \left(\delta_{pr} \theta_{ri}^{k+1}\right) out'^{\,sigm}_{pi}\, out_{pj}^{sigm} & \text{if unit } i \text{ is a hidden unit} \end{cases} \tag{7}$$

The update of the parameters according to (7), is based on an approximation of the gradient of the error landscape. In the following frequently used variant of the backpropagation algorithm, the so-called momentum term α is introduced, that aims at stabilizing the dynamic behavior of the update of the parameters ([Rumelhart 1986], [Tugay 1989]):

$$\Delta_p \theta_{ij}^k = -\eta \frac{\partial SE_p}{\partial \theta_{ij}^k} + \alpha \Delta_{p-1} \theta_{ij}^k \tag{8}$$

As it is well-known that gradient descent is not a very good optimization method [Press 1988], one can wonder if backpropagation deserves any serious attention in the light of four decades of research in numerical methods. The answer to this question is surprisingly "yes". The reason is that more advanced optimization methods generally require the inverse of the Hessian matrix $\partial^2 SE / \partial \theta_{ij}^k \partial \theta_{pq}^r$. As we are frequently concerned with networks that have several hundreds up to tens of thousands free parameters, storage and inversion of the Hessian is clearly non-trivial, and first-order methods are therefore the only remaining alternative.

A related issue is that the backpropagation learning can be considered as the process of solving a system of differential equations. Due to the sigmoidal non-linearities in the network, this system of differential equations is clearly non-linear. Furthermore, since we assume in the context of statistical pattern recognition that the training data is of a stochastic nature, the backpropagation algorithm is actually solving a system of non-linear differential equations of stochastic variables. As there are no closed-form solutions of these equations known to exist, empirical research is the only way to obtain a better understanding of the learning dynamics. As can be expected from the dynamics of such a highly non-linear system, empirical research is bothered by the problems of a critical dependence on the initial conditions [Kolen 1991], or even the possibility of chaotic behavior [Maas 1990].

Despite these shortcomings, backpropagation has proven to be an enormous source of inspiration, as literally hundreds or even thousands of variants have been published in the literature. Among these are generalizations towards other activation and/or output functions ([Robinson 1988], [Poggio 1989], [Weymare 1991]), error-measures that aim at regularization of the network transfer function ([Chauvin 1989], [Weigend 1991], [Bishop 1991], [MacKay 1992]), error-measures that are based on criteria from information theory instead of the squared error criterion [Miller 1991], various adaptations of higher order methods to speed up or to improve the learning process ([Owens 1989], [le Cun 1991a], [Battiti 1992], [Bishop 1992]), procedures to change the network topology or network architecture along with the training of the parameters ([Mozer 1989], [Fahlman 1990], [le Cun 1990b]), recurrent variants that allow learning of temporal sequences of data [Rumelhart 1986], etc.

4.2.2. Some Experiments on Large Networks and Small Samples

As it was stated in the introduction of this chapter, the apparent success of the experiments of table 1 is surprising in the light of the generally accepted view upon the relation between sample size and classifier complexity. The very good performance of these networks in combination with the (very) small training sample size, seem to point at an interesting phenomenon, that deserves considerable attention. However, one should not neglect that a number of simple conditions could facilitate such good results.

One way to explain the successes of table 1 is that the underlying recognition problems are very simple, cf. section 1.2. If the data consist of very compact point-like clusters, good recognition results can be expected, almost independently of the classifier complexity. In such simple problems, probably any classifier can be chosen to obtain good results, and there is no need to think of any special properties of multi-layer networks and/or the back-propagation algorithm.

Second, another explanation could be that, even for difficult underlying recognition problems, the training data could be a very fortunate realization of the underlying stochastic process. For example, if the classes in the data do have a considerable overlapping part of their distributions, a fortunate realization would have almost no data in the overlapping part. A classifier that is trained with such a dataset will perceive a number of well-separated classes, for which it is not too difficult to determine a good decision function. Although this explanation is highly unlikely from a statistical viewpoint, the massive amount of research applications in this area makes it probable that this effect occurs from time to time.

As a conclusion, if we want to learn something about the generalization behavior of multi-layer network classifiers, the effects of fortunate distributions and fortunate realizations need to be excluded. For this reason it was decided to perform a number of experiments with synthetic data based on overlapping classes, in which multiple realizations of the training data were investigated. As we are most interested in the regime $m < W$ (i.e. $m < V$, cf. inequality (3.10)), relatively large networks were trained with relatively small amounts of data.

The goal of these experiments is to illustrate some relevant issues on the generalization behavior of multi-layer networks that are trained with the backpropagation algorithm. They are not intended as a complete coverage of the problem domain; in section 4.3.4 a more exhaustive experimental investigation is performed, aiming at explanation instead of illustration. In the following sections, first the training data is presented, followed by a description of the learning procedures, the experimental results and a discussion of the results.

4.2.2.1. *The Training Data*

To investigate the generalization behavior of multi-layer networks that are trained with the backpropagation algorithm, two datasets were defined. According to the statistics of these datasets, different training sets and test sets with a varying number of samples were generated. The definitions are as follows:

• *Dataset I:* This dataset consists of two equiprobable bivariate normally distributed classes with the following statistics:

Table 2: The statistics of dataset I.

	mean vector	covariance matrix
class A	(0.2, 0.2)	0.333 ***I***
class B	(-0.2, -0.2)	0.333 ***I***

The intrinsic overlap (see section 1.2.1) of the distributions of the classes A and B is 0.2. As the covariance matrices of the two classes are equal, the Bayes decision function is a linear classifier, with a performance of 0.8.

• *Dataset II:* This dataset also consists of two equiprobable bivariate normally distributed classes with the following statistics:

Table 3: The statistics of dataset II.

	mean vector	covariance matrix
class A	(0.0, 0.0)	0.2211 ***I***
class B	(0.0, 0.0)	0.0301 ***I***

The intrinsic overlap of the distributions of the classes A and B is also for this dataset equal to 0.2. The Bayes decision function is a circle centered at the origin.

4.2.2.2. *The Learning Procedure*

As we have a special interest for the case $m < W$, the main experiments were performed with multi-layer networks consisting of 2 inputs, one hidden layer with 100 hidden units and one output unit. For comparison, all experiments were also performed

with networks having 10 and 1 hidden units. The networks were trained with the backpropagation training algorithm, as presented in section 4.2.1. The parameters of the algorithm were as follows. Initially all parameters were set to small values, generated according to a uniform distribution in the range [-0.01, 0.01]. Then, with a given training set, the network parameters were updated 100,000 times, with a learning rate η of 0.1 and no momentum term (i.e. $\alpha = 0$). After each parameter update, the recognition performance on the training set was computed and if the current parameter values performed equally or better than previous parameter values, this set of parameters was stored separately [Gallant 1990b]. The output of the training phase was a network with this best set of parameters. The performance of the resulting network was computed with an independent test set of 2000 samples.

The approach of using the set of parameters that had the best performance on the training set provided the parameter values that were maximally adapted to the training set. In this sense the risk of overtraining (the adaptation of the parameters to the noise in the training data) was maximized, and the procedure must therefore be considered as a worst-case approach. Another advantage is the fact that this approach is relatively insensitive to the total number of parameter updates.

4.2.2.3. *The Experiments*

In the main experiment, the influence of the training sample size on the performance of the network was investigated. For both datasets, a network with one hidden layer of 100 hidden units was trained with randomly drawn datasets consisting of 2, 4, 10, 20, 40, 100 and 200 samples; i.e. 1, 2, 5, 10, 20, 50 or 100 samples per class. This experiment was repeated 100 times, and during each repetition different initial parameter values, a different training set and a different test set were generated. As a reference, the same experiment was also performed for the 1-nearest neighbor classifier. In figures 1 and 2, the average performance of the networks is depicted as a function of the training sample size.

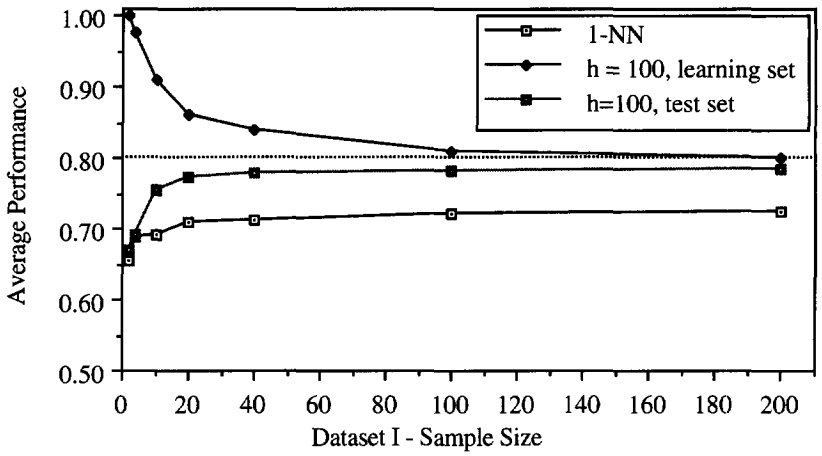


Figure 1: The average performance of a multi-layer network with 100 hidden units on the training set and on an independent test set of dataset I, as a function of the training sample size. The average is taken over 100 parameter initializations and training sets. The average performance of the 1-nearest neighbor classifier is shown as a reference. Note that the Bayes error is equal to 0.2.

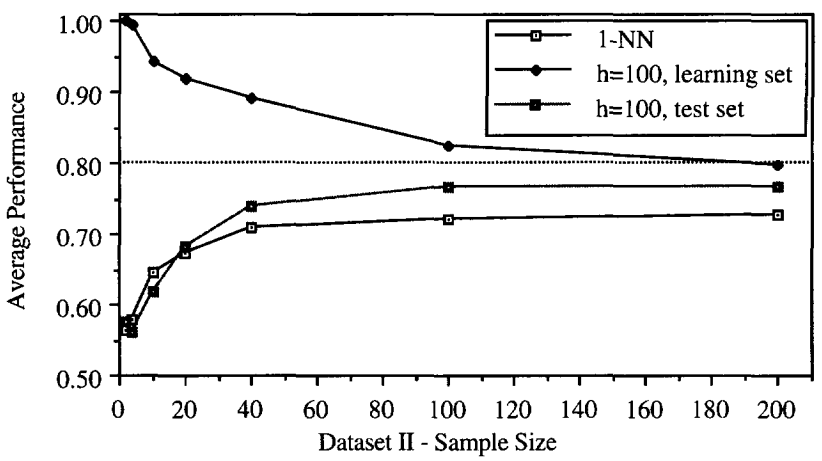


Figure 2: The average performance of a multi-layer network with 100 hidden units on the training set and on an independent test set of dataset II, under similar conditions as in figure 1.

Table 4: The average performance and the standard deviation of the performance estimates of a multi-layer feedforward network with 1, 10 and 100 hidden units and the nearest neighbor classifier, on the training set and an independent test set of dataset I, as a function of the training sample size. The average is taken over 100 parameter initializations and training sets. For each network, the number of free parameters W and a lower and an upper bound for the capacity V are computed from equations (3.10)* and (3.15).

Dataset I	**sample size**	**performance on learning set**		**performance on test set**		**difference**
classifier	m	avg. perf.	std. perf.	avg. perf.	std. perf.	avg. diff.
$h = 1$	2	1.000	0.000	0.658	0.151	0.342
	4	0.983	0.064	0.701	0.107	0.282
$W = 3$	10	0.919	0.085	0.749	0.066	0.170
$3 \le V \le 24$	20	0.878	0.071	0.768	0.033	0.110
	40	0.846	0.053	0.786	0.019	0.060
	100	0.929	0.034	0.795	0.013	0.134
	200	0.820	0.027	0.798	0.010	0.022
$h = 10$	2	1.000	0.000	0.672	0.133	0.328
	4	0.985	0.060	0.704	0.104	0.281
$W = 41$	10	0.932	0.071	0.720	0.091	0.212
$20 \le V \le 401$	20	0.867	0.070	0.771	0.033	0.096
	40	0.838	0.049	0.783	0.022	0.055
	100	0.817	0.039	0.790	0.017	0.027
	200	0.816	0.026	0.799	0.011	0.017
$h = 100$	2	1.000	0.000	0.668	0.144	0.332
	4	0.978	0.072	0.690	0.133	0.288
$W = 401$	10	0.910	0.092	0.756	0.046	0.154
$200 \le V \le$	20	0.861	0.081	0.773	0.033	0.088
6496	40	0.840	0.051	0.779	0.029	0.061
	100	0.810	0.052	0.782	0.029	0.028
	200	0.801	0.036	0.787	0.020	0.014
1-nearest	2	1.000	0.000	0.655	0.134	0.345
neighbor	4	1.000	0.000	0.688	0.104	0.312
classifier	10	1.000	0.000	0.691	0.077	0.309
	20	1.000	0.000	0.708	0.049	0.292
	40	1.000	0.000	0.714	0.036	0.286
	100	1.000	0.000	0.722	0.021	0.278
	200	1.000	0.000	0.724	0.017	0.276

* The upper bound $V \le 2W\log(eN)$, cf. equation (3.10), on the capacity of a multi-layer feedforward network with W parameters and N units, was derived for networks consisting of units with a threshold output function [Baum 1989]. Due to the lack of a similar result for networks with sigmoidal units, this bound is also used here. As the capacity of a network with sigmoidal units is higher than the capacity of a network with threshold units, the actual capacity may even be higher than the upper bounds that are shown in the table.

As the networks that were used in these two experiments all have 401 free parameters, it is clear that the training sample size is indeed much smaller than the number of parameters. In some additional experiments, the effect of a reduction of the number of free parameters was investigated. In tables 4 and 5, the results of similar experiments on networks with 10 hidden units and 1 hidden unit are presented. The results of the experiments as depicted in figures 1 and 2 have been incorporated in the tables.

Table 5: The average performance and the standard deviation of the performance estimates of a multi-layer feedforward network with 1, 10 and 100 hidden units and the nearest neighbor classifier, on the training set and an independent test set of dataset II, under similar conditions as in table 4.

Dataset II	**sample size**	**performance on learning set**		**performance on test set**		**difference**
classifier	m	avg. perf.	std. perf.	avg. perf.	std. perf.	avg. diff.
$h = 1$	2	1.000	0.000	0.566	0.049	0.434
	4	0.948	0.102	0.563	0.057	0.385
$W = 3$	10	0.839	0.084	0.578	0.043	0.261
$3 \leq V \leq 24$	20	0.762	0.061	0.593	0.027	0.169
	40	0.722	0.056	0.595	0.023	0.127
	100	0.672	0.038	0.603	0.021	0.069
	200	0.653	0.026	0.605	0.015	0.048
$h = 10$	2	1.000	0.000	0.561	0.057	0.439
	4	0.998	0.025	0.574	0.056	0.424
$W = 41$	10	0.966	0.064	0.624	0.073	0.342
$20 \leq V \leq 401$	20	0.925	0.078	0.690	0.059	0.235
	40	0.894	0.049	0.742	0.039	0.152
	100	0.846	0.037	0.783	0.022	0.063
	200	0.830	0.045	0.795	0.035	0.035
$h = 100$	2	1.000	0.049	0.575	0.049	0.425
	4	0.995	0.060	0.561	0.060	0.434
$W = 401$	10	0.942	0.070	0.617	0.067	0.325
$200 \leq V \leq$	20	0.918	0.074	0.682	0.059	0.236
6496	40	0.892	0.058	0.740	0.050	0.152
	100	0.826	0.063	0.766	0.041	0.060
	200	0.797	0.053	0.768	0.045	0.029
1-nearest	2	1.000	0.000	0.564	0.050	0.436
neighbor	4	1.000	0.000	0.578	0.078	0.422
classifier	10	1.000	0.000	0.646	0.063	0.354
	20	1.000	0.000	0.674	0.057	0.326
	40	1.000	0.000	0.710	0.037	0.290
	100	1.000	0.000	0.721	0.025	0.279
	200	1.000	0.000	0.728	0.019	0.272

4.2.2.4. *Discussion*

Before the results of the previous section are interpreted, it is illustrative to discuss what kind of results we could have expected beforehand. If we focus on the experiments as depicted in figures 1 and 2, two extreme expectations are the following:

• *The pessimistic view:* as the backpropagation algorithm is optimizing the mean squared mapping error, the iterative learning procedure may eventually end in a global minimum of the error space, such as sketched in figure 3.4, cf. section 3.2. For convenience, this figure is repeated here as figure 3. In this figure, the parameters of the network are chosen such that the squared mapping error is zero and the performance on the learning set is 100%, but this value is useless as an estimate for the performance on an independent test set.

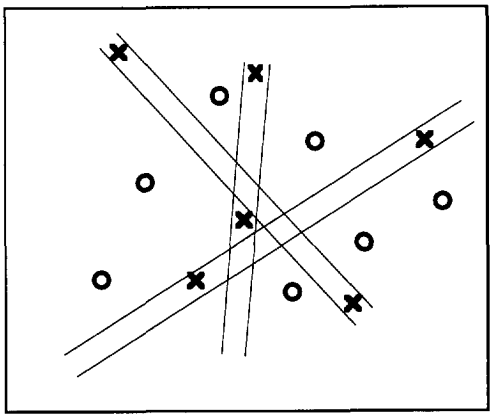


Figure 3: A global minimum of the quadratic error criterion (1) can be constructed by a slicing construction in the feature space; each pair of (hyper)planes in $\Re^d$ allows the enclosing of exactly d points.

• *The optimistic view:* when the ratio of the number of free parameters and the training sample size is very large, a more desirable global minimum of the error space corresponds to a configuration of the network parameters such that it implements the 1-nearest neighbor classifier, cf. section 1.2 and figure 4. Also in this construction the performance on the learning set is 100%, but according to well-known results from the statistical pattern recognition literature this estimate is definitely significant. In fact, for a

training sample size that approaches infinity, the error of the 1-nearest neighbor classifier is bounded by two times the Bayes-error ([Devijver 1982], [Fukunaga 1990]). It should be noted, however, that contrary to the geometrical construction of figure 3, it is not certain that for $m > 2$ a network with 100 hidden units is capable of representing the decision function of the 1-nearest neighbor classifier.

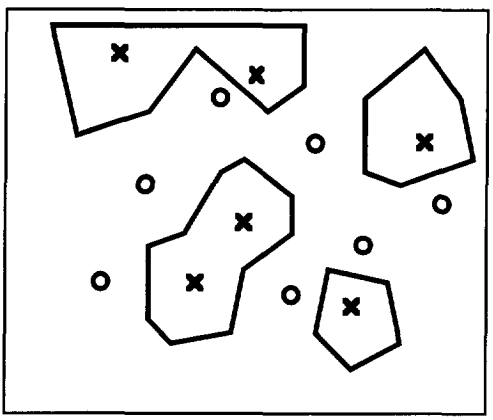


Figure 4: An alternative global minimum of the quadratic error criterion (1) is provided by the 1-nearest neighbor classifier. The performance on the training set of this decision function, however, is much more significant than the function depicted in figure 3.

Now, if we return to the experimental results of figures 1 and 2, there is indeed some reason to be surprised. First, it is immediate that the pessimistic view is unnecessarily pessimistic. A choice of the parameters, such as depicted in figure 3, is simply never found by the backpropagation algorithm, as the performance on the independent test set would in that case be close to 50%. Although there is no doubt that some choice of the parameters exists that implements the construction as depicted in figure 3, this global minimum must be hidden behind local minima and/or saddle points in the error landscape.

With respect to the optimistic view, the performance of the 1-nearest neighbor classifier only serves as a reasonable prediction of the performance of the network, when the sample size is much smaller than the number of free parameters. For such very small sample sizes, a straightforward conclusion from the experimental results is that the backpropagation algorithm generally positions the decision function roughly in the middle of samples of an opposite class. When the decision function would be positioned on any

other location between samples of an opposite class, the performance would have been lower than the performance of the 1-nearest neighbor classifier. Thus, for a very large ratio of the number of free parameters to the number of samples, the backpropagation algorithm is capable of inducing a decision function resembling the 1-nearest neighbor classifier. When the sample size becomes larger, this decision function becomes more and more complex, and would require significantly more degrees of freedom than the network has available and/or the learning procedure can allocate. As the network decision function than necessarily becomes smoother than the 1-nearest neighbor classifier, the performance of the network becomes better than the 1-nearest neighbor classifier.

Another interesting aspect is found by a comparison of the average performance of the networks having 100 hidden units in the hidden layer, and those having 10 or 1 hidden units in the hidden layer. The experiments on dataset I, cf. table 4, convey that the average performance decreases as a function of the number of free parameters. This effect is essentially the peaking phenomenon, cf. chapter 1. As the optimal decision function for dataset I is a linear classifier, the availability of more parameters than is necessary to represent a linear classifier, makes that the parameters will be adapted to the noise of the training set, see also [Kohonen 1988] and [Raudys 1991a]. This effect is clear from table 4, although it is certainly not as dramatic as the prediction of figure 3.

Table 5, finally shows another aspect of the use of multi-layer networks in a classification context. The Bayes decision function of dataset II is a circle. Networks with only 1 hidden unit are not capable of representing an approximation of such a circular decision function. Networks with 10 or 100 hidden units, however, do. The peaking phenomenon again causes the average performance of the networks with 100 hidden units to be slightly worse than the performance of networks with 10 hidden units.

As these experiments illustrate that multi-layer networks are indeed capable of operating in the range $m < W$, it is appealing to investigate what kind of relation exists between the sample size and the deviation of the true and estimated errors. The Vapnik-Chervonenkis theory clearly suggests a linear relation, cf. chapter 2, equation (2.30). However, the Vapnik-Chervonenkis theory is concerned with the maximum deviation of the true and estimated errors over the complete parameter space of a classifier. In our experiments we only consider the classifier that performs best on the training set. The Vapnik-Chervonenkis theory, therefore, is not immediately applicable to the experiments that are described in this section.

Nevertheless, the prediction of the Vapnik-Chervonenkis theory is easily verified by a linear regression on the last column of tables 3 and 4. In figure 5, the training sample

size is plotted versus the reciprocal value of the difference of the performance on the training set and the performance on the test set, for both datasets and for the 100-hidden unit network. The figure indeed shows a very strong linear relation, with correlation coefficient $\rho^2 = 0.999$ for dataset I and $\rho^2 = 0.997$ for dataset II respectively. This behavior is in accordance with the results of various other authors, that were discussed in chapter 1. The most interesting aspect of figure 5, however, is that it clearly shows that the expected deviation of the true and estimated errors is not a function of the number of free parameters of the classifier, as both plots are made for a 100-hidden unit network. Instead, as the only difference between both plots is the underlying recognition problem, the slope of the linear functions must be based on the distributions of the classes.

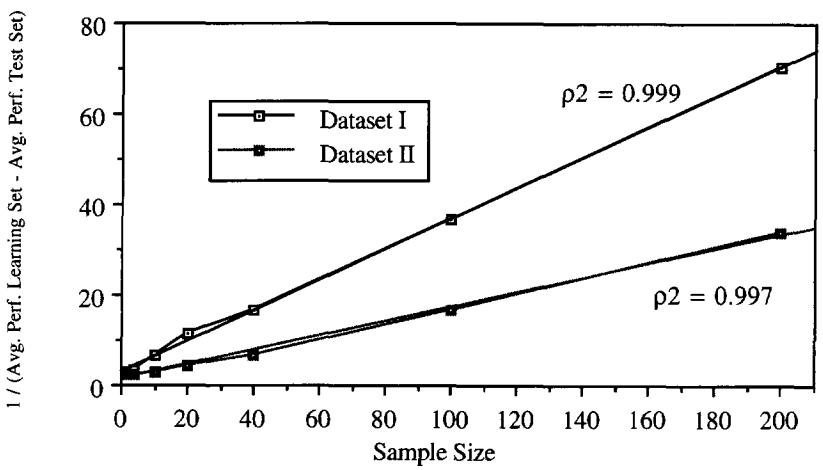


Figure 5: The reciprocal value of the average difference of the performance on the training set and the performance on the test set of a network with 100 hidden units as a function of the training sample size. The high correlation coefficients show the apparent linear relation.

As a conclusion, the experiments that are described in this section lead to a number of interesting observations on the generalization behavior of multi-layer feedforward networks that are trained with the backpropagation algorithm. In the first place, the experiments convincingly illustrate that they can properly operate in the range $m < W$, even for recognition problems with overlapping classes. This finding seems to support the experimental results of table 1, and thus seems to point at some deficiencies in our

knowledge on the relation between sample size and classifier complexity. A second finding is that the peaking phenomenon is not absent, but that it operates far from the worst possible case. The goal of the next section is now to provide a better understanding of these effects.

4.3. Iterative Learning Procedures and Generalization

In this section it is argued that the remarkable properties of multi-layer networks that were reported in the previous section are caused by the iterative learning procedure, i.e. the backpropagation learning rule. However, as the learning dynamics of backpropagation are not very well understood (cf. section 4.2.1), the effects of an iterative learning procedure is first investigated for some very simple cases, that are much more analytically tractable. Then, in section 4.3.2, we will discuss several of the factors that influence the characteristics of the iterative learning procedure. This is followed by the introduction of the notion of the effective capacity in section 4.3.3. In a number of experiments, described in section 4.3.4, some estimates of the effective capacity of a multi-layer network are given and, finally, a discussion of these experiments is presented in section 4.3.5.

4.3.1. Generalization of some Simple Iterative Procedures

As the non-linear dynamics of the backpropagation algorithm does not allow a sufficient insight in its effects on the generalization behavior, we first focus on some considerably simpler learning procedures. Three simple examples are the following:

• A degenerate case of an iterative learning procedure (cf. equation (7)) is given by the following rule:

$$\Delta_p \theta_{ij}^k = 0 \tag{9}$$

From (9) it is clear that such a learning rule is not useful for obtaining a good classification function. However, if we use (9) to "train" a classifier with a training set of moderate size, we can expect that the expected poor performance on the training set can serve as a realistic estimate of the true performance on an independent test set. In fact, as we are dealing with only one value of the parameter vector during the course of the iterative "learning" process, the Hoeffding inequality (2.1) or its variant (2.7) can be used to bound the true error in terms of the estimated error, and there is no need to worry about the capacity or the number of degrees of freedom of the classifier under consideration.

Although (9) is certainly degenerate, it is clear that the foregoing line of thought is approximately valid if (9) is replaced by the rule:

$$\Delta_p \theta_{ij}^k \approx 0 \tag{10}$$

This variant serves as a realistic characterization of the learning process if the learning rate η is taken very small. Another condition in which (10) is relevant, is when an exponential scaling is applied to the error space in which the error-minimization is performed. This is especially relevant for multi-layer feedforward networks with sigmoidal or Gaussian non-linearities, cf. equations (7), (3.5) and (3.6). If the output functions of the units in such a network are "saturated", the change of the parameters of the unit will be negligible, as the adaptation of the parameters is proportional to the derivative of the output function, cf. equation (7). This situation is encountered in practice, when the initial values of the parameters is taken too large or when the training data is not properly scaled. As a result, the training procedure is perfectly characterized by (10) (or even (9) in the discrete domain of a digital computer), and the training phase will most likely end in a poor classifier. We can only take comfort in the thought that with (2.1) or (2.7) we can very accurately estimate how poor the classifier is....

• A second example of an iterative learning procedure is the following:

$$\Delta_p \theta_{ij}^k = c_{ij}^k \tag{11}$$

for some constant c_{ij}^k. This learning rule has the effect that a 1-dimensional sub-space of the parameter space is investigated during the learning phase, independently of the training data. If (11) is applied L times, the theory developed in section 2.1 can be used to make predictions on the validity of the error frequencies of the L classifiers that are encountered during this iterative procedure. As long as L does not become too large (cf. section 2.1.2), there is no need to consider the capacity or the number of degrees of freedom of the classifiers under consideration.

• The last training procedure that we consider here is an exhaustive search of the parameter space of a classifier. Obviously, this is only of a conceptual value as implementations would not be too practical. However, it is for this type of training procedure that we need the Vapnik-Chervonenkis theory to make predictions about the generalization behavior. Only when the parameter space is exhaustively searched, the

probability that we encounter the classifier with the largest deviation of the estimated error to the true error is one. In all other cases, this probability is obviously less than one.

The three procedures above range from no search at all, to an exhaustive search in the parameter space, with large consequences for the generalization behavior of the resulting classifiers. The conclusion of this exercise is that the generalization behavior is largely influenced by the characteristics of the search method, and that, as long as no exhaustive search is performed, the capacity does not serve as the ultimate tool to predict the generalization behavior. Two important questions that now arise are:

- At which position in this list should we place backpropagation?
- Can we still use the notion of the capacity in the context of an iterative learning procedure that does not perform an exhaustive search?

4.3.2. Limitations to the Searching Capabilities of Iterative Learning Procedures

In the previous section, it was suggested that the validity of generalization is a function of the efficiency of the search procedure. Therefore, in this section we will focus on some of the factors that influence this efficiency. Five of such factors are the following:

• If the iterative search procedure is performed in a non-convex error space, local minima and/or saddle points in the error landscape will severely limit the area that is investigated. A consequence of this, is that the learning procedure is not capable of fully exploiting the available degrees of freedom.

A nice illustration of this effect is shown in figure 6. This figure is a representation of the parameter values of the hidden units, in one of the experiments of section 4.2.2.3. In those experiments, a network with one hidden layer of 100 hidden units was trained with a sample of size 200 of dataset I with the backpropagation algorithm. The axes in figure 6 represent the parameter values, after the training phase, of the parameters that are associated with the connections between the inputs and the hidden units. The 100 dots represent the 100 hidden units. The gray value of each dot is weighed by the absolute value of the parameter of the hidden unit to the output unit; a dot with zero parameter value is not visible, but also does not contribute to the network transfer function.

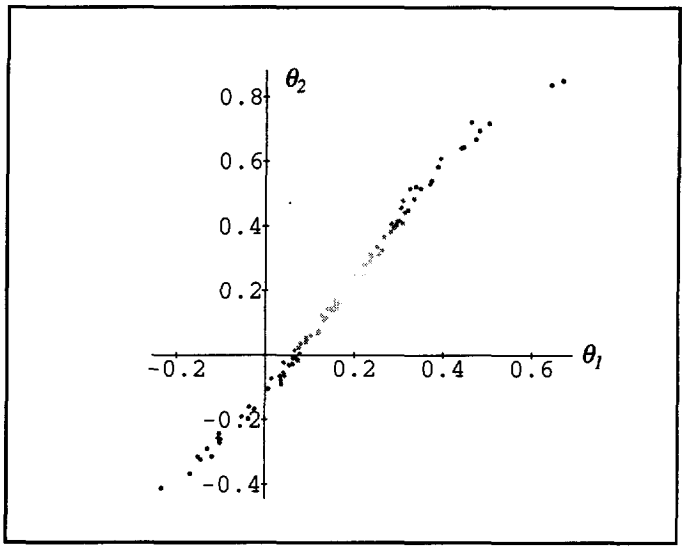


Figure 6: A network with 100 hidden units was trained on a sample of size 200 of dataset I. Every dot represents the two parameters associated with the connections to the inputs, of a hidden unit after the training phase. The gray value of each dot is weighed by the absolute value of the parameter of the hidden unit to the output unit. Note that all dots are approximately on a one-dimensional subspace.

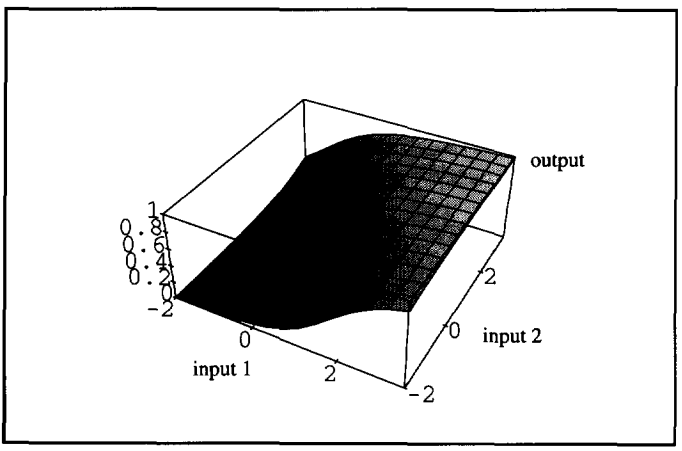


Figure 7: The network transfer function, resulting from the parameters as depicted in figure 6.

From figure 6 it is apparent that the parameter values of all hidden units are approximately found on a one-dimensional subspace of the parameter space. As the ratio of the two parameters to the inputs determines the "direction" of the hidden unit in the input space, it is clear that all hidden units are oriented in the same way. This proves that, instead of exploiting the free parameters such that a global minimum as in figure 3 is reached, the learning rule prefers a local minimum in which all hidden units perform roughly the same mapping. As a result, the transfer function of the network is not the spiky decision function of figure 3, but the very smooth function that is depicted in figure 7.

• Most iterative learning procedures have a large number of implicit or explicit ways to control various aspects of the learning behavior. Among these are the learning rate [Maas 1990], the (statistics of the) initial parameters of the network [Kolen 1991], the number of times a network is trained with different initial parameters ([Schmidt 1993a], [Schmidt 1993b]), the momentum term ([Rumelhart 1986], [Tugay 1989]), the stopping criterion [Kraaijveld 1991a], etc. Especially for learning procedures with highly non-linear dynamics, these factors may have a considerable influence on the area of the parameter space that is investigated by the learning procedure. However, as was stated before, the non-linear dynamics do not allow a good prediction of the consequences of a particular choice.

• A factor that heavily influences the area of the parameter space being investigated, is the amount of time that is used during the search. It is obvious that with a finite amount of training time, it is not possible to perform a search in an infinitely large area with an infinite amount of detail. An interesting result, therefore, is that it was recently shown that limiting the amount of training time is essentially equal to performing some kind of regularization [Sjöberg 1992]. If the search procedure is initialized with small values of the parameters, non-smooth decision functions (far from the origin) can only be reached after a considerable number of iterations.

Various authors have described other applications of regularization theory to the training of multi-layer networks, e.g. see [Chauvin 1989], [Weigend 1991], [Bishop 1991], [MacKay 1992]. In most of these references, the squared error criterion (1) is extended with a term that prefers parameter values close to the origin of the parameter space. As a consequence, the search procedure is likely to be concentrated near the origin.

• The representation of the parameter vector in double floating point numbers in a digital computer, is for most pattern classifiers sufficiently accurate. However, when exponential scaling of the error space is applied, such as with sigmoidal non-linearities, the representation accuracy may become a relevant issue. In fact, it is easily shown that if the argument of a sigmoidal function is smaller than -40, the corresponding function value is of the order of $1 - 10^{-20}$. In a double floating point representation this number can not be distinguished from the value 1.0. This implies that if the product of the input vector and the weights of a unit is smaller than -40, the unit is effectively saturated.

This leads to two conclusions. In the first place, it was already mentioned that the saturation of the output function will prevent further update of the parameters, and that the learning dynamics in that case is essentially characterized by (9). A second consequence is that this example clearly shows that the exponential scaling of the error space may cause a considerable reduction of the total number of dichotomies that a classifier can induce on a dataset. As the search is obviously limited to the dichotomies that the classifier *can* represent, the search is performed over a smaller domain than for the case of continuous parameters.

• Finally, another factor that largely influences the search procedure is not related to the details of the training procedure, but is the training data itself. First, if the data consists of well-separated classes, there is no need for the learning procedure to iterate towards complex decision functions. The search procedure will be focused on the set of classifiers that simply separate the classes, and it is likely that the learning procedure will end in a classifier with good generalization. This principle, together with some of its consequences is worked out in section 4.4. Second, if the data has a low intrinsic dimensionality, the number of distinct classifiers that can be investigated during the search procedure is lower than for the case of a high intrinsic dimensionality. In fact, if the data is hidden on a d^*-dimensional linear subspace of the d-dimensional feature space, and if a linear classifier is trained with such a dataset, the capacity is obviously equal to $d^* + 1$, instead of $d + 1$, cf. theorem 2.8.

4.3.3. Effective Capacity and Actual Capacity

As we now have some insight in the factors that influence the search procedure of an iterative learning rule, the question emerges whether it is possible to adapt the framework of the Vapnik-Chervonenkis to incorporate the notion of the efficiency of the search procedure. The answer to this question is non-trivial and rather difficult. For some of the factors above, it seems to be possible to adapt the theoretical framework to account for

them. For example, in theorem 2.5 there is a factor $S_{\max}^{F}\left(m(\beta+1)\right)$ that is equal to the maximum number of dichotomies that the classifiers in F can induce on a sample of size $m(\beta + 1)$. If the search is limited to a small subset of the parameter space (due to local minima, the finite training time or the finite representation accuracy), the total number of dichotomies that can be induced on the set of $m(\beta + 1)$ points is obviously smaller, possibly giving rise to a sharper version of theorem 2.5. Unfortunately, there is no way in which one can predict beforehand which fraction of the parameter space will be investigated. Conceptually, however, it makes sense to introduce a notion of *effective capacity*, to account for the specific properties of a learning procedure. This is defined as follows, see also [Duin 1993]:

Definition 4.1: the *effective capacity* V^* of a classifier that is trained with an (iterative) learning procedure is defined as the maximum number of points that the learning procedure can give an arbitrary Boolean label.

Furthermore, the Vapnik-Chervonenkis theory can easily be adapted for the case of data with a low intrinsic dimensionality. For such cases we introduce the notion of the *actual capacity* to account for specific properties of the dataset.

Definition 4.2: the *actual capacity* V_A^* of a classifier that is trained with a dataset with an (iterative) learning procedure, is defined as the maximum number of points of the dataset that the learning procedure can give an arbitrary Boolean label.

The notions of effective and actual capacity have recently been proposed by a number of authors. Basically, research has been performed along two lines of thought:

• The first approach originates from regularization theory and leads to the notion of *the effective number of parameters* of a classifier. Its background can be understood by considering the following regularized cost function, cf. equation (1):

$$RSE = SE\left(\Xi^m, \boldsymbol{\theta}\right) + \gamma\|\boldsymbol{\theta}\|^2 = \frac{1}{2}\sum_{p=1}^{m}\left(\lambda_p - f^*\left(\xi_p, \boldsymbol{\theta}\right)\right)^2 + \gamma\|\boldsymbol{\theta}\|^2 \qquad (12)$$

The error criterion (12) consist of a data dependent term $SE\left(\Xi^m, \boldsymbol{\theta}\right)$ and a term $\gamma\|\boldsymbol{\theta}\|^2$ that expresses the preference for parameter values close to the origin of the parameter space. The latter term can be interpreted in a Bayesian context to reflect the a-priori knowledge on the parameter vector $\boldsymbol{\theta}$. The minimization of (12) is very well understood

for the case of a linear classifier. In the linear case, the data dependent term $SE(\Xi^m, \boldsymbol{\theta})$ yields a parabolic error landscape, that is completely characterized by the eigenvalues φ_i, $1 \leq i \leq d$, of the input correlation matrix $E(\boldsymbol{\xi}_p \boldsymbol{\xi}_p^T)$. In fact, the curvatures in the principal directions of the parabolic landscape are equal to the eigenvalues φ_i [Widrow 1985]. The second term $\gamma\|\boldsymbol{\theta}\|^2$ can be interpreted as a string that pulls the parameter values in this parabolic landscape back to the origin. For directions with a high curvature the influence of the spring is negligible, whereas for directions with a low curvature, the spring may be capable of pulling the parameter value over a considerable distance. Thus, some parameters are mainly determined by the regularization term $\gamma\|\boldsymbol{\theta}\|^2$, whereas other parameter values are determined by the parabolic landscape, i.e. by the data dependent term. A mathematical treatment of the foregoing leads to the notion of the effective number of parameters, e.g. see [MacKay 1992], [Moody 1992]:

$$W^* = \sum_{i=1}^{d} \frac{\varphi_i}{\varphi_i + \gamma} \tag{13}$$

According to (13) the effective number of parameters varies between $W^* = d$ (for $\gamma = 0$), and $W^* \to 0$ (for $\gamma \to \infty$). As for a linear classifier the capacity is directly related to the number of free parameters (cf. theorem 2.8), Gyon et al. suggest to use (13) also as a measure for the *effective capacity* V^* of a regularized linear classifier [Gyon 1992a], [Gyon 1992b]. A series of experiments on a character recognition dataset provided considerable support for this idea.

It is important to note that the parameter γ is associated with the learning procedure, as different learning procedures are implemented for different values of γ. According to our definition, the effective capacity is in this context a measure that is related to the characteristics of the learning procedure. A major drawback of the approach, however, is that a generalization towards other classifiers and regularizers is not immediate, if not impossible. For instance, a generalization of (13) to the error surface of a multi-layer network with sigmoidal non-linearities is not known to exist.

• The second experimental approach to the Vapnik-Chervonenkis theory is based on some recent work of Vapnik [Vapnik 1992], [Levin 1992], and has led to the concept of actual capacity. The approach is based on a variant of the theory of chapter 2, to allow for an empirical estimate of the capacity. The underlying idea is based on three steps. First, from the theorem on the probability of maximum (relative) deviation of the estimated error to the true error $P\{\sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} |\varepsilon(\boldsymbol{\theta}) - \hat{\varepsilon}(\boldsymbol{\theta})| > \chi\}$, a bound is derived on $E(\sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} (\hat{\varepsilon}_1(\boldsymbol{\theta}) - \hat{\varepsilon}_2(\boldsymbol{\theta})))$; the *expected maximum deviation* of the error frequencies

$\hat{\varepsilon}_1(\boldsymbol{\theta})$ and $\hat{\varepsilon}_2(\boldsymbol{\theta})$ on two samples Ξ_1^m and Ξ_2^m of size m, cf. equation (2.18). This bound is obviously a function of the variables V and m. In the second step, the iterative learning procedure is used to search for the parameter vector for which the deviation of $\hat{\varepsilon}_1(\boldsymbol{\theta})$ and $\hat{\varepsilon}_2(\boldsymbol{\theta})$ is maximized. This parameter vector can be found by training the classifier with the sample $\Xi_1^m \cup \Xi_2^m$, while the class labels of the second sample Ξ_2^m have been inverted. This process is repeated a number of times to obtain a good estimate of the expected maximum deviation of $\hat{\varepsilon}_1(\boldsymbol{\theta})$ and $\hat{\varepsilon}_2(\boldsymbol{\theta})$, and this is again repeated for various values of m. The third and final step consists of the selection of a value for the capacity V, such that the corresponding theoretical curve that was obtained in step 1 is as close as possible to the empirical results. If the data is hidden on a subspace, such that the classifiers can only implement decision functions with a capacity $V_A^* \leq V$, the notion of an actual capacity V_A^* emerges. Contrary to the definition of effective capacity, this concept is related to the data, instead of the learning procedure.

In [Levin 1992] an empirical verification of this approach is described. For the case of linear classifiers, it is shown convincingly that one can obtain good estimates of the capacity, independently of the underlying distributions of the data. Also for the case of linear classifiers that are trained with data that is hidden on a linear subspace, very accurate estimates of the capacity are provided. However, for other classifiers than linear threshold functions, no results have been reported (yet). It is therefore not clear that this approach is suited to determine the capacity of a multi-layer network. More important, as in this context the concept of the actual capacity is associated with properties of the data, it is not immediate that this approach can also account for an effective capacity due to the characteristics of a particular learning rule.

Despite the limitations of the foregoing approaches, the concepts of effective capacity and actual capacity bear a considerable amount of elegance. As the foregoing approaches of Gyon and Vapnik are either valid for regularized linear classifiers only, or by definition related to the underlying distribution, we therefore now propose an alternative experimental procedure to estimate the effective or actual capacity of a learning procedure and/or dataset. The approach is conceptually very simple and is directly built upon the definition of the capacity. As the capacity of a set of classifiers F is defined as the maximum number of points that the classifiers in F can give an arbitrary Boolean label (cf. definition 2.1), we will try to estimate the effective capacity of a learning procedure and/or dataset by trying to find the *largest subset of the dataset that the learning procedure can label arbitrarily.*

This definition of an effective capacity naturally accounts for special properties of the data, as well as the characteristics of the learning procedure. In the following section, the details of the experimental procedure and a number of experiments will be described.

4.3.4. The Estimated Effective Capacity of Multi-Layer Networks Trained with the Backpropagation Algorithm

In this section a number of experiments are described in which we estimate the effective capacity of a multi-layer network that is trained with the backpropagation algorithm. The conditions of these experiments very much resemble those of the experiments of section 4.2.2, so that the outcome may help in getting a better insight in the results of those experiments. However, as the non-linear dynamics of the backpropagation algorithm may be sensitive to particular choices for the parameters of the algorithm, it is also investigated how the effective capacity is influenced by such factors as the learning rate, the scaling of the training data, etc. In the following sections, we will first discuss the overall procedure that was used in the experiments, followed by some details on the training data and the learning procedure, section 4.3.4.2. Then, an experimental verification of the procedure is described in section 4.3.4.3, followed by an investigation of the influence of the number of hidden units, the initialization of the network parameters, the learning rate, the learning time, the scaling of the training data and the dimensionality of the feature space in sections 4.3.4.4 to 4.3.4.9 respectively.

4.3.4.1. *The Experimental Procedure*

Conceptually, the approach is very simple. The procedure is started with an estimate for the effective capacity V^* that is equal to one. Then, a subset of size V^* is randomly selected from the dataset. If the learning procedure is capable of inducing all 2^{V^*} dichotomies on this subset, the effective capacity is obviously at least equal to V^*. In that case, the estimate of the effective capacity V^* is increased by one and the procedure is repeated. However, if the learning procedure is not capable of inducing all 2^{V^*} dichotomies on the subset, another subset of size V^* is randomly selected from the dataset. If the learning procedure is not capable of inducing the 2^{V^*} dichotomies on 1,000 subsets of size V^*, it is concluded that the effective capacity is equal to $V^* - 1$. A pseudo-code version of the complete procedure is described in appendix F.

Despite the conceptual simplicity of the method, it is clear that the approach results in considerable computational requirements. If at some moment in the procedure V^* is equal to 10, $2^{10} = 1024$ dichotomies have to be investigated. Therefore, the method quickly becomes impractical for larger values of V^*. Also, the choice to investigate at most 1,000

subsets of size V^*, is rather arbitrary, and primarily motivated by computational considerations. Note, however, that the repeated selection of subsets is only necessary to cope with the problem that an unfortunate subset of points could (almost) be hidden in some subspace. Such a set of points could only be labeled arbitrarily with great difficulty. It is reasonable to assume, therefore, that 1,000 different subsets is sufficient to yield at least some subsets for which this is not the case, unless the total dataset is indeed hidden in a subspace.

4.3.4.2. *The Training Data and the Learning Procedure*

In the experiments that are described in the rest of this section, the influence of various parameters of the backpropagation algorithm on the effective capacity is investigated. However, as there many of such parameters, it is not possible to explore the complete space of parameter settings. Instead, we have defined one "standard experiment" and then in subsequent experiments the effect of changing only one parameter was investigated. The settings of this standard experiment were taken as follows:

- According to the statistics of table 2, a dataset of 100 + 100 points was generated as the *standard dataset I*. The dataset was saved to disk, such that in all experiments exactly the same dataset could be used. Note that the feature space of this dataset is two dimensional.
- As the *standard network* of the experiments, a network with two inputs, 100 hidden units in one hidden layer, and one output unit was generated by assigning small random values to the parameters of the network. The initial values of the parameters were generated according to a uniform distribution in the range [-0.01, 0.01]. All units of the network were provided with the standard inner product activation functions and sigmoidal output functions, cf. equations (3.1) and (3.6). The network was also saved to disk, such that all experiments could be performed with the same initial parameters.
- During the training procedure, the parameters of the network were updated for at most 100,000 times with the backpropagation algorithm as described in section 4.2.1. The learning rate η was chosen as 0.1 and no momentum-term was applied ($\alpha = 0$). The simulations were performed in a custom-made environment that was written in C [Kraaijveld 1993].

4.3.4.3. *An Experimental Verification for Linear Classifiers*

To verify the procedure and to gain some insight in its characteristics, some experiments were performed in which the capacity is exactly known. As for linear classifiers the capacity can be determined exactly (i.e. not upper bounded), a number of experiments were performed in which the capacity of a linear classifier was estimated with the experimental procedure described above.

According to the theory of section 2.2.1, the capacity of a linear classifier in d-dimensional space is equal to $d + 1$. For the first experiment, four datasets of 100 + 100 samples with dimensionality 1, 2, 5 and 10 were generated. The statistics of these datasets were comparable to those of table 2; the distance between two normally distributed classes with covariance matrix equal to $\boldsymbol{I}$ was taken such that the overlap of the classes was equal to 0.2. In table 6, the estimated capacity $\hat{V}$, according to the procedure described in section 4.3.4.1 is shown as a function of the dimensionality of the feature space.

Table 6: The estimated capacity of a linear classifier.

dimensionality d	1	2	5	10
true capacity $V = d + 1$	2	3	6	11
estimated capacity $\hat{V}$	2	3	6	11

To investigate the concept of an actual capacity, the experiments on the linear classifier were extended with a series of experiments in which the dataset has a low intrinsic dimensionality. With similar statistics as above, datasets were generated in a 20, 50 and 100-dimensional space, where all data was hidden in a linear subspace of dimensionality 1, 2, 5, or 10. Table 7 presents the estimated actual capacity $\hat{V}_A^*$ as a function of the dimensionality of the feature space and the intrinsic dimensionality of the data.

4.3.4.4. *The Number of Hidden Units*

In the first experiment to determine the effective capacity of a multi-layer network, the influence of the number of hidden units of the network is investigated. Networks with 1, 2, 5, 10, 20, 50, and 100 hidden units were generated and trained with the standard dataset. Note that for the networks with 1, 10 and 100 hidden units the conditions are

equivalent to the experiments described in table 2, cf. section 4.2.2, and can therefore be considered as the determination of the effective capacity in those experiments. Furthermore, the "standard experiment" is the experiment with 100 hidden units.

Table 7: The estimated effective capacity of a linear classifier.

dimensionality d	intrinsic dimensionality d^*	true actual capacity $V = d^* + 1$	estimated actual capacity $\hat{V}_A^*$
20	1	2	2
	2	3	3
	5	6	6
	10	11	10
50	1	2	2
	2	3	3
	5	6	6
	10	11	9
100	1	2	2
	2	3	3
	5	6	6
	10	11	9

Table 8: The estimated effective capacity as a function of the number of hidden units, estimated on the standard dataset I. The result of the "standard experiment" is shown in the shaded cell.

number of hidden units h	1	2	5	10	20	50	100
number of free parameters W	5	9	21	41	81	201	401
lower bound on V, cf. (3.15)	3	4	8	20	40	100	200
upper bound on V, cf. (3.10)	24	54	169	401	945	2860	6496
estimated effective capacity $\hat{V}^*$	3	5	6	7	7	6	**5**

As this experiment may provide some insight in the results of section 4.2.2, it is also repeated for the second dataset that was used in those experiments, cf. table 3 and table 5. The following table shows the effective capacity as a function of the number of hidden units, estimated on dataset II of a size of 100+100 samples.

Table 9: The estimated effective capacity as a function of the number of hidden units, estimated on the dataset II.

number of hidden units h	1	2	5	10	20	50	100
number of free parameters W	5	9	21	41	81	201	401
lower bound on V, cf. (3.15)	3	4	8	20	40	100	200
upper bound on V, cf. (3.10)	24	54	169	401	945	2860	6496
estimated effective capacity $\hat{V}^*$	3	5	6	6	6	6	6

4.3.4.5. *The Initialization of the Network Parameters*

In section 4.3.2 it was discussed that the backpropagation algorithm may be sensitive to the initial values of the parameters of the network. Therefore, it is to be expected that the effective capacity may also be influenced by the initial choice of the parameter vector. In a series of experiments this was verified. In the first place, the statistics according to which the initial parameter values were generated were varied over three domains: $\theta \in$ [-0.01, 0.01], $\theta \in$ [-1.0, 1.0] and $\theta \in$ [-100.0, 100.0]. Second, with each set of statistics 5 networks were generated of which the effective capacity was estimated. The results are presented in table 10.

Table 10: The estimated effective capacity for various realizations of the initial parameter values, and as a function of the statistics that generate the initial parameter values.

domain of uniform distribution for the initial parameters	realization				
	1	2	3	4	5
$\theta \in$ [-0.01, 0.01]	**5**	6	6	6	6
$\theta \in$ [-1.0, 1.0]	6	6	7	7	7
$\theta \in$ [-100.0, 100.0]	1	2	2	1	2

4.3.4.6. *The Learning Rate*

In a subsequent experiment the effect of the learning rate η on the effective capacity was determined. In table 11, it is shown how the effective capacity is influenced by changing the learning rate η between 10^{-6} and 10.

Table 11: The estimated effective capacity as a function of the learning rate η.

learning rate η	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10
estimated effective capacity $\hat{V}^*$	1	1	2	3	4	**5**	5	1

4.3.4.7. *The Learning Time*

The learning time, or equivalently the number of parameter updates during the learning phase, is a parameter that has a considerable influence on the behavior of an iterative learning procedure. Table 12 reports its influence on the effective capacity.

Table 12: The estimated effective capacity as a function of the number of parameter updates.

number of updates	1	10	10^2	10^3	10^4	10^5	10^6	10^7
estimated effective capacity $\hat{V}^*$	1	1	1	3	4	**5**	6	6

4.3.4.8. *Scaling of the Training Data*

In the next experiment the influence of the scale of the training data was investigated. The input vectors of the standard training dataset were multiplied with a constant ranging from 10^{-2} to 10^{+2}. In table 13 the resulting effective capacity is shown.

Table 13: The estimated effective capacity as a function of the scale of the training data.

scaling factor	0.01	0.03	0.10	0.30	1	3	10	30	100
estimated effective capacity $\hat{V}^*$	2	3	3	4	**5**	7	7	7	7

4.3.4.9. *The Dimensionality of the Feature Space*

Finally, another factor that may influence the effective capacity is the dimensionality of the feature space d. In this experiment it was chosen to the number of hidden units fixed at 100, such that the number of free parameters of the network increases as a function of d. The data was generated analogously to the experiments of table 6; the distance between two normally distributed classes with covariance matrix equal to $\boldsymbol{I}$ was taken such that the overlap of the classes was equal to 0.2. The results of these experiments are shown in table 14.

Table 14: The estimated effective capacity as a function of the dimensionality of the feature space d.

dimensionality d	number of free parameters W	lower bound on V cf. (3.15)	upper bound on V cf. (3.10)	estimated effective capacity $\hat{V}^*$
1	301	100	4876	4
2	401	200	6496	5
5	701	500	11357	7
10	1201	1000	19458	11
20	2201	2000	35660	15

4.3.5. Discussion

A striking aspect of the experimental results of the previous section, is that the estimated effective capacity $\hat{V}^*$ of a multi-layer feedforward network classifier, that is trained with the backpropagation algorithm, is so low. For all the variants of the standard experiment, the training procedure has not been able to find some set of 8 points that can be labeled in an arbitrary way, despite the 401 free parameters of the network, a guaranteed lower bound on the capacity of 200, and a huge investment of roughly 10^{15} CPU instructions. Apparently, the application of the backpropagation rule, instead of performing an exhaustive search, reduces the maximum number of points that a classifier can label in an arbitrary way with orders of magnitude. This finding confirms that the backpropagation algorithm is simply not capable of exploiting all the free parameters of the network.

A closer look at the results reveals the following aspects:

• The results of table 6 confirm the correctness of the method of estimating the capacity, as the estimated capacity of a linear classifier is in all cases equal to its true capacity. Table 7 shows that the method also works correctly for estimating the *actual* capacity of a linear classifier, although for some experiments the method provides a slight underestimate of the true actual capacity. This may be caused by an unfortunate choice of the parameter settings of the learning procedure, and revealed that many of such settings should be investigated in order to obtain a good estimate. In tables 8 to 13 this was performed for training multi-layer networks with various parameter settings.

• Tables 8 and 9 present the consequences of changing the number of hidden units on the estimated effective capacity. It is clear that the number of hidden units, or the number of free parameters has a negligible influence on the effective capacity of the network. Changing the number of hidden units from 2 to 100 made the estimated effective capacity vary between 5 and 7. A remarkable fact is that the estimated effective capacity seems to have a maximum for 10 and 20 hidden units in table 8. An explanation for this effect, is that an increase of the number of hidden units generally slows down the learning process. This is caused by the fact that a large number of hidden units results in a high correlation of their outputs, as the number of inputs is constant. It can be shown that, in the beginning of the learning phase, such a high correlation results in a decrease of the learning speed of the output unit [Bakker 1993]. Some support for this explanation is found in table 12, where it is shown that an increase of the total number of parameter updates of the standard network to 1,000,000 or 10,000,000 increases the estimated effective capacity to 6.

• Further influence of the total number of parameter updates is found in table 12. When the number of parameter updates is chosen too small, the learning procedure is not capable of reaching a desired location in parameter space. This causes the effective capacity of the system to be very small. Of course, this is not a property of multi-layer networks classifiers only, as it also holds for linear classifiers. However, the table shows that an increase of the number of parameter updates beyond 1,000,000 has no influence on the estimated effective capacity.

A related parameter of the backpropagation algorithm is the learning rate η. If the learning rate is too small, the learning procedure may be stopped before a suitable (local) minimum or saddle point has been reached. Making the learning rate too large, however, will result in a non-stable, or even chaotic, behavior of the learning procedure. This also prevents the learning procedure of reaching a certain desired location. As table 11 reveals, the optimum is found somewhere in-between: the maximum estimated effective capacity is found for a learning rate of 0.1 or 1.0.

• It is well-known that the behavior of backpropagation critically depends on the initial choice of the parameter vector [Kolen 1991]. The experiments in table 10, however, show that as long as the initial parameters are chosen sufficiently small, the estimated effective capacity is not heavily influenced. Only when the initial parameters are chosen much too large, the dynamic behavior of the learning procedure is affected. This is caused by the fact that the large initial parameter values cause the sigmoidal non-

linearities of the network to saturate, cf. sections 4.3.1. and 4.3.2. For properly chosen initial parameter values, however, the estimated effective capacity again varies between 5 and 7.

• A related parameter of the backpropagation algorithm is the scaling of the training data. A too small scale of the training data will make it difficult to label data in an arbitrary way, as a decision function has to be positioned accurately in a very small area. This is not always possible since the learning rate, i.e. the step-size of the adaptations of the parameters, is fixed. On the other hand, if the scale of the training data is too large, the sigmoidal non-linearities of the network will saturate, which would imply a small effective capacity. However, this is an effect that is not visible in table 13. This can be explained by the fact that the training data is of a stochastic nature; the data is spread out over some area and some of the points of the training set are on locations that have the optimal balance between separability and saturation of the sigmoids. By repeated selection of subsets from the training set, some of these subsets will have many points on these ideal locations. Therefore, the fact that the estimated capacity is constant for large scaling factors, is a property of the procedure to estimate the effective capacity.

• Finally, the last table 14 shows the effect of an increase of the dimensionality of the feature space. Again, it is clear that the estimated effective capacity is orders of magnitude smaller than the true capacity or the number of free parameters. However, it should be noted that increasing the dimensionality of the feature space corresponds to other problems than the standard two-dimensional problem, and that it therefore might be necessary to reinvestigate some of the other parameter settings of the backpropagation algorithm.

As a conclusion from all the tables, it is clear that the effective capacity of a multi-layer network that is trained with the backpropagation algorithm depends on the various settings of the parameters of the algorithm. The experimental results also reveal that the settings that were initially chosen for the standard experiment were not too bad. In the standard experiment an effective capacity of 5 was estimated, and in many tables this was close to the maximum. In fact, the maximum estimated effective capacity that was found in all experiments with a two-dimensional feature space was equal to 7. As the experiments did not perform an exhaustive search of the parameter settings of the backpropagation algorithm, this value of 7 is likely to be an underestimate of the true effective capacity. However, the fact that the value 7 is based on 40,000 attempts to use a

100 hidden unit network to label some subsets in an arbitrary way (tables 10 - 13), does provide some confidence in its value.

4.3.6. Conclusions

In this section it was discussed how the specific properties of an iterative learning procedure influence the generalization behavior of a classifier. This was first illustrated for some very simple learning procedures, and then some of the factors that limit the searching capabilities of an iterative learning procedure were discussed. The notion of effective capacity and actual capacity of a learning procedure and dataset were introduced, and an experimental method to estimate the effective capacity was described. Furthermore, some experiments were performed to estimate the effective capacity of a multi-layer feedforward network classifier that was trained with the backpropagation algorithm. The experiments show that the estimated effective capacity may be orders of magnitude smaller than the true capacity of a network. This proves that the searching capabilities of the backpropagation algorithm are very limited. An implication of this finding, however, is that the performance of such a network on a training set is much more significant than the number of free parameters would suggest.

V

Conclusions

In this chapter we will review and comment on the results of the preceding chapters. Furthermore, we will return to the main question of this thesis, that was posed in section 1.3. Finally, a number of recommendations on the applicability of multi-layer feedforward network classifiers for solving pattern recognition problems will be given.

5.1. A Review of Results

The main question that was addressed in this thesis, is how the relation between the complexity of a multi-layer feedforward network classifier and the requirements on the training sample size can be characterized. The starting point in answering this question was the suggestion of the statistical pattern recognition literature, that the training sample size should be proportional to the number of free parameters of a classifier, cf. chapter 1.

The first refinement of this idea was provided in chapter 2. The *capacity* or *Vapnik-Chervonenkis dimension* was introduced as a measure to quantify the complexity of a classifier. This notion of capacity is simply defined as the maximum number of points that a classifier can give an arbitrary Boolean label. If the capacity of a classifier can be determined, the Vapnik-Chervonenkis theory allows to make statements on the maximum (relative) deviation of the true and estimated error of a classifier, in terms of the capacity and the training sample size. For linear and polynomial classifiers, the capacity is equivalent to the number of free parameters of the classification function. For most non-linear classifiers, however, no such equivalence exists. The capacity is, therefore, a more general measure of complexity than the number of free parameters.

Furthermore, in chapter 2 three theorems were presented that provide a slight improvement of a specific formulation of the Vapnik-Chervonenkis theory. These three theorems provide:

- an upper bound for the probability of maximum relative deviation of the true error from the estimated error, in terms of the maximum relative deviation, the capacity and the training sample size (theorem 2.5).

- an upper bound for the true error of a classifier, in terms of the error on the training set, the training sample size, the capacity and a confidence factor (theorem 2.6).

- a lower bound for the training sample size, in terms of the capacity, the maximum relative deviation and the confidence factor (theorem 2.7).

As the Vapnik-Chervonenkis theory is distribution-free and independent of a learning procedure, the bounds of these theorems are rather pessimistic for a given learning procedure and a given dataset. An important part of their value, therefore, is the fact that they provide a considerable amount of insight in the relation between the variables that are involved in the design of a classifier.

In chapter 3, the Vapnik-Chervonenkis theory was used to study the small sample behavior of a number of connectionist models, without making reference to a particular training procedure. A number of issues on the architecture, topology and basis functions of such classifiers were discussed, together with a theorem of Baum and Haussler on the capacity of a multi-layer feedforward network classifier [Baum 1989]. This theorem, in combination with the theory that was developed in chapter 2, gives rise to an improved lower bound on the sample size that is required to train a multi-layer feedforward network classifier (corollary 3.1).

The second contribution of chapter 3 is a study of a number of learning procedures for a specific class of multi-layer feedforward network classifiers: the minimal kernel based networks. This class of networks has the advantage of low memory and speed requirements and the availability of consistent learning procedures. An upper bound on the capacity of some of these learning procedures was provided, allowing an exact characterization of their generalization behavior. Finally, it was shown that the capacity of this class of networks is smaller than the capacity of a regular feedforward network classifier, and it was discussed under which conditions this is advantageous with respect to the requirements on the training data.

The third connectionist model that was investigated in chapter 3 is the class of multi-layer feedforward network classifiers with shared parameters. The coupling of parameters in such networks results in a large reduction of the number of degrees of freedom, and can be a very fruitful approach for solving invariant recognition problems. Some methods

and results were discussed that can be used to bound the capacity of a parameter-sharing network, and this was applied to a simple class of such networks. A comparison with the bound for the capacity of a regular feedforward network indicates that there is an additional bonus for using the parameter-sharing technique, since the bound for the capacity decreases faster than the number of free parameters.

In chapter four the influence of the (iterative) learning procedure on the generalization behavior of a multi-layer feedforward network classifier was investigated. In a number of experiments it was shown, that multi-layer feedforward network classifiers that are trained with the backpropagation algorithm do have an unexpectedly good performance, when the training sample size is only a fraction of the number of free parameters in the network. As an explanation of this effect, it was discussed how the limited capabilities of a learning procedure in searching through the parameter space of a classifier affect the validity of generalization on the training set. For learning procedures that investigate only a small fraction of the parameter space, the performance on the training set is much more significant than for learning procedures that investigate the complete parameter space.

This gives rise to the notion of an *effective capacity*, to account for the specific properties of a learning procedure. Furthermore, the notion of an *actual capacity* was introduced to account for specific properties of the training data. An experimental procedure was introduced to estimate the effective capacity and the actual capacity of a multi-layer feedforward network classifier. The experimental results convincingly show that the effective capacity of a multi-layer feedforward network may be orders of magnitude smaller than the true capacity. This proves that the search of the backpropagation algorithm is limited to a small fraction of the parameter space only. Therefore, the performance on the training set of a multi-layer feedforward network classifier that is trained with the backpropagation algorithm, is much more valid than the number of free parameters in the network would suggest.

5.2. General Conclusions

The general conclusions of this thesis, and the answers to the main question, are the following:

- The capacity can serve as a good measure of complexity of a multi-layer feedforward network classifier. In the framework of the Vapnik-Chervonenkis theory, this facilitates worst-case statements on the applicability of such classifiers, when no reference to a particular learning rule is made.

• The capacity of the connectionist models that are investigated in this thesis is of the order $W \log W$.

• Within the context of a specific learning procedure, the capabilities of this learning procedure in searching through the parameter space should be taken into account, to make statements on the validity of the performance on the training set. The effective capacity of the learning procedure can serve as such a measure.

• The effective capacity of a multi-layer feedforward network classifier may be orders of magnitude smaller than its true capacity.

5.3. The Applicability of Multi-Layer Network Classifiers

A consequence of the results of chapter four, is that the objection that multi-layer feedforward network classifiers possess too many free parameters to do something useful is removed. For multi-layer feedforward network classifiers, the issue is not the number of free parameters, but the properties of the learning procedure. Despite a large number of free parameters, a multi-layer network may be fruitfully applied for solving pattern recognition problems in various applications.

However, this thesis and the experience in working with various connectionist models that has led to this thesis, have made clear that multi-layer feedforward network classifiers do not systematically outperform other classifiers. Fortunately, multi-layer feedforward network classifiers also do not systematically perform worse. The choice for a multi-layer network or some other classifier from the statistical pattern recognition literature should therefore depend on such factors as:

• A-priori information of the pattern recognition problem. If some knowledge of the parametric form of the underlying probability density functions is known, and a fair amount of training data is available, it is highly unlikely that a multi-layer network will outperform a classifier that is based on the estimated density functions, cf. section 1.2.1.

• Memory and speed requirements of an application. For many classifiers that are not based on parametric models of the underlying probability density functions, such as the nearest-neighbor classifier or the Parzen classifier, a multi-layer network may provide a substantial increase in speed and may significantly lower the memory requirements of a classifier. This is based on the fact that after the training phase, a multi-layer network

explicitly represents the decision function (in a relatively compact way), whereas the nearest-neighbor classifier or the Parzen classifier only implicitly represents a decision function. For large amounts of training data it is obviously advantageous to store the decision function only, as that may allow an increase in processing speed and lower the memory requirements of the classifier. It should be noted, however, that also for the nearest-neighbor classifier and the Parzen classifier procedures exist to speed up the classification process, or to lower the memory requirements, such as the editing algorithm and the condensing algorithm, cf. chapter 6 and [Devijver 1982]. However, similar to the training procedure of a multi-layer network, these procedures require a considerable amount of processing time before the decision function is represented in a sufficiently compact way.

• The shape of the decision functions. In a multi-layer feedforward network, a decision function is approximated by a (nested) sum of (sigmoidal) basis functions. For some applications, this type of basis functions may have some advantages over other basis functions as the decision function may have a sigmoidal shape. For such problems, the application of multi-layer feedforward networks may be advantageous.

It is clear that it is difficult to make statements on the advantage of multi-layer networks, without making reference to a particular application. However, as many authors have shown, multi-layer feedforward networks seem to provide a valuable extension of the data-analysis toolkit.

Part 2

Practical Aspects

Introduction to Part 2

The second part of this thesis is dedicated to two practical aspects of training a multi-layer feedforward network classifier. In chapter six the problem of the stopping criterion, that determines when an iterative learning procedure is terminated, is addressed. An optimal stopping criterion is presented, and its influence on the learning speed and the generalization properties of a multi-layer network classifier are discussed.

Moreover, in chapter seven, some issues on the learning speed of a class of iterative learning procedures are investigated. Some theory is developed that predicts the consequences on the learning speed of an increase of the number of inputs, for single-layer or multi-layer network classifiers.

VI

The Stopping Criterion

M.A. Kraaijveld and R.P.W. Duin

An Optimal Stopping Criterion for Backpropagation Learning

Neural Network World
Vol. 1, no. 6, pp. 365-370, 1991.

AN OPTIMAL STOPPING CRITERION FOR BACKPROPAGATION LEARNING

Martin A. Kraaijveld, Robert P.W. Duin[1]

Abstract: A common problem of iterative learning procedures like the backpropagation algorithm is the lack of insight in the learning phase. Therefore, it is difficult to decide at what moment the learning phase should be terminated. For many ad hoc criteria that are used in practice, it can be shown that they suffer from serious defects, especially for recognition problems in which the distributions of the classes have some overlap. By the application of a technique from the statistical pattern recognition literature, the editing algorithm [3], a learning set can be transformed to a data set in which the overlap of the classes is effectively removed. This results in an optimal stopping criterion for iterative learning procedures, and a number of experiments indicate a moderate improvement in learning speed for the backpropagation algorithm. Moreover, because it can be proven that an edited data set yields a performance which is close to Bayes-optimal for the nearest-neighbor classifier, it is very likely that a classifier which is based on an iterative learning procedure and which classifies *all* samples in the edited learning set correctly, is also close to Bayes-optimal.

Keywords: *iterative learning procedure, statistical pattern recognition, neural networks, backpropagation learning, editing algorithm.*

Received: May, 5, 1991
Revised and accepted: December 9, 1991

1. Introduction

In the last few years an enormous interest in learning procedures for neural networks has emerged. An important development in this area was the invention of the backpropagation algorithm by Rumelhart [9]. Practical applications (e.g.[2], [4], [11], etc.) as well as benchmarking studies (e.g.[8] have shown that classifiers based on the backpropagation algorithm have a good performance. Multi-layer networks therefore seem to offer a reasonable alternative for various parametric and non-parametric techniques of the statistical pattern recognition literature. Especially the feature that complex shaped decision functions can be represented by a limited number of units and weights, resulting in a very compact representation of a classifier, facilitates fast operation for real time applications [7].

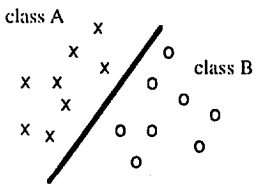
However, from a practical and theoretical viewpoint, there are still some drawbacks:

1. Training a multi-layer network with the backpropagation algorithm is slow. In comparison with other methods (e.g. the nearest-neighbor classifier), multi-layer networks require large amounts of CPU time to obtain a reasonable classifier. This is due to a number of specific properties of the algorithm:
   - The algorithm is iterative in nature.
   - The learning rate η should be small in order to improve convergence. However, a small η results in a long learning phase.
2. The algorithm is essentially based on a gradient descent in an error space which contains local minima. Therefore, the learning phase can get stuck into a local minimum, resulting in a suboptimal performance of the classifier. It is important here to distinguish the cases of recognition problems with overlapping and non-overlapping classes (see Fig.1):

   When there is no overlap between the classes in the learning set, it is desirable that the network, for the learning set as well as for any test set, finally correctly classifies 100% of the samples. Getting stuck in a local minimum implies that the performance is less than 100% , and is therefore suboptimal.

   When the classes in the learning set do overlap, however the situation is completely different. When the learning set has a Bayes error (i.e. an intrinsic overlap) of ε % , no classifier will ever achieve a performance higher than (100 - ε)% . Therefore, when a large multi-layer network has achieved a performance of 100% on the learning set, it must have made many small decision boundaries around samples in the overlapping part of the learning set and will certainly have a suboptimal performance on a test set. As the backpropagation algorithm tries to minimize the

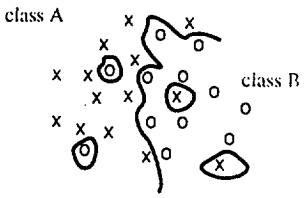
[1] Martin A.Kraaijveld, Robert P.W.Duin
Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, P.O.Box 5046, 2600 GA Delft, The Netherlands



non-overlapping classes:

Mean squared mapping error minimal

Classifier optimal



overlapping classes:

Mean squared mapping error minimal

Classifier SUBoptimal

Fig.1: *For a learning set with non-overlappings classes, a global minimum in the mean squared mapping error corresponds to an optimal classifier. For a learning set with overlapping classes, the global minimum in the mean squared mapping error corresponds to a suboptimal classifier*

mean squared mapping error, it does in fact tend to a situation with decision boundaries around outliers, because this corresponds to a lower point in error space.

From this we conclude that the optimal classifier for a learning set with overlapping classes is surprisingly found in a *local* minimum of the error space. For overlapping classes it is therefore highly desirable that the learning phase gets stuck in a local minimum! (N.B. notice that for a learning set which is not sufficiently representative for the underlying distributions of the classes, the optimal classifier might not even correspond to a local minimum in error space).

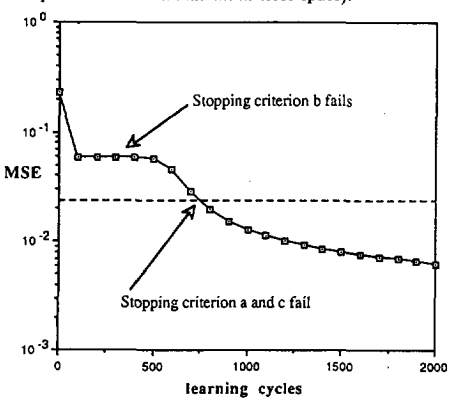


Fig.2: *Ad hoc stopping criteria fail to stop the learning phase at the correct moment.*

3. A third problem is that there is no insight in the learning process. Unless there is some explicit knowledge of the underlying distributions of the classes in the decision problem, it is very difficult to decide in which state the learning phase remains. Is the current state the global error minimum? Is the current state close to a ravine in error space? Is the network currently learning an outlier of the classes in the learning set?, etc.

   Furthermore, it is not clear if learning should continue until all samples are classified correctly (in the case of separable classes), or until another stopping criterion is reached (in the case of intrinsic overlap of the classes). Due to the lack of a stopping criterion that takes into account the state of the network in error space, ad hoc criteria are applied. Examples of these are (see Fig.2):

   (a) continue the learning phase until the the mean squared error is "sufficiently" low.

   (b) continue the learning phase until the time-derivative of the mean squared error is sufficiently low.

   (c) continue the learning phase until a sufficient percentage of the learning set is classified correctly.

Especially the first problem that is sketched above, bears a close parallel to problems that are related to the nearest neighbor classifier. For the nearest neighbor classifier, outliers of a different class will generate regions in which samples will be assigned to the wrong class. This deteriorates the performance of the resulting classifier. For a multi-layer network, the learning procedure will try to

separate these small regions around the outliers, because this decreases the mapping error.

A solution for the problem of the outliers with the nearest-neighbor rule is described by [3], and is called "editing". The editing algorithm effectively removes the overlap in case of overlapping classes, i.e. it effectively separates the classes. This results in a learning set from which all outliers are removed. The main contribution of this paper is to investigate the behavior of the backpropagation algorithm on edited data sets.

2. Learning of Edited Data Sets

The editing technique [3] is an algorithm that is used to improve the performance of the nearest neighbor classifier as well as to reduce the number of samples in the learning set. The algorithm removes the intrinsic overlap (if any) between the classes in the learning set in such a way that it hardly affects the position of the optimal decision boundary. However, erroneously classified samples are removed (see Fig.3). In the context of nearest neighbor pattern classification this largely improves the performance of the resulting nearest neighbor classifier. In fact it can be proven that, provided that the learning set is sufficiently large, the performance of the 1-nearest neighbor rule on an edited data set is very close to Bayes-optimal.

For the experiments described in this paper, a special version of the editing algorithm was used, which is called the *multi-edit algorithm*. It consists of 5 steps:

1. **Diffusion:** Make a random partition of the learning set S into N subsets $S_1, \ldots, S_N$, $N > 2$.
2. **Classification:** Classify the samples in S_i using the 1-nearest neighbor rule with $S_{(i+1) Mod\ N}$ as a training set, $i = 1, \ldots, N$.
3. **Editing:** Discard all samples that were misclassified at step 2.
4. **Diffusion:** Pool all the remaining data to constitute a new set S.
5. **Termination:** If the last I iterations produced no editing then exit with the final set S, else go to step 1.

Using an edited data set instead of non-edited data set for the training of a multi-layer network is expected to have the following consequences:

- By editing the data set, a very deep minimum is in error space is introduced. Because the learning set is now completely separable (although not necessarily linearly separable), this error minimum is also a global minimum and has an error value of zero. The stopping criterion for the learning phase has now become trivial: *as long as not all samples are classified correctly the learning phase should be continued.*
- Because all outliers in the region of the other class are removed from the learning set, there is no danger that the performance of the classifier will deteriorate by learning outliers, even when the network has a (very) large number of hidden units.
- Although it is somewhat difficult to measure, it appears that learning an edited data set is faster than learning a non-edited data set. The difficulties with the measurements are caused by the fact that the learning time of an non-edited data set is based on a certain (ad hoc) stopping criterion. Any figure for the learning time can be produced, however, when the parameters of this stopping criterion are changed. Learning is therefore certainly faster in the sense that we can make an earlier decision to stop the learning phase. Notice that this improvement in speed is derived by pre-processing of the data instead of an adaptation of the learning procedure.

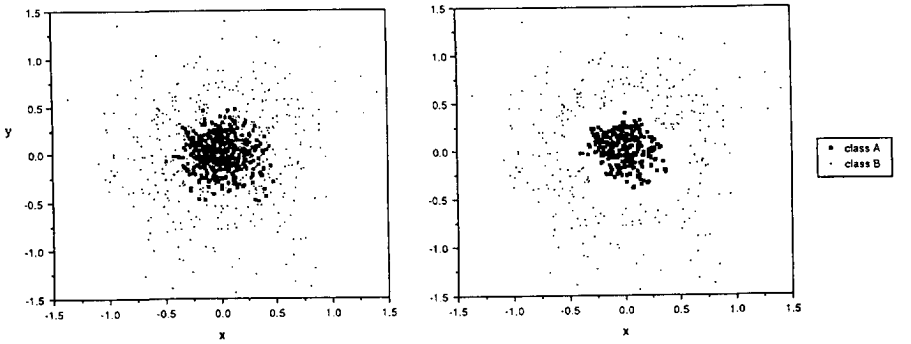


Fig.3: *Data set 1 (left), and the data set after editing*

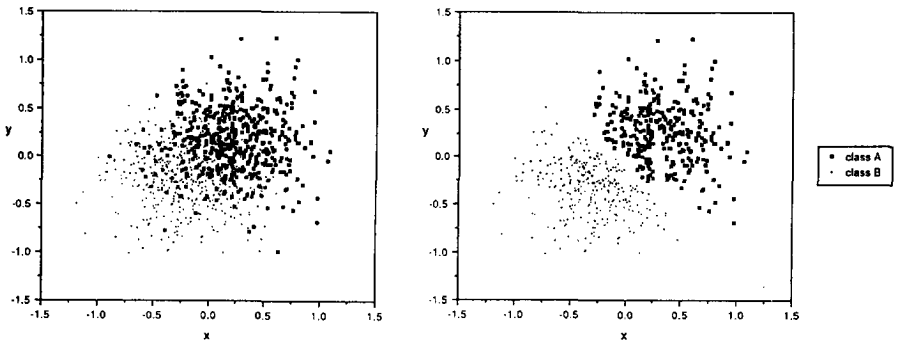


Fig.4: *Data set 2 (left), and the data set after editing*

- When there is no overlap in the classes, the editing algorithm will not affect the learning set. Therefore, the editing algorithm can be applied in any case, overlapping or non-overlapping, to assure that the resulting set is separable.
- The editing algorithm can be considered to perform a transformation of the underlying removing the overlapping part of the densities. It is important to realize that it is this transformation that enables the use of the new stopping criterion. By transforming the original densities to the edited densities we can solve the problem of the stopping criterion in the original domain.
- The editing algorithm, applied to network training as well as nearest-neighbor classification, can be considered as a pre-processing step. However, there are differences in the goals to be reached in both cases. For the nearest neighbor classifier the goal is the decrease of the size of the learning set and the increase of the performance of the nearest neighbor classifier. For network training, however, editing a data set aims at influencing the error space of the learning process such that a good criterion to finish the learning phase can be found.

3. Experiments

To investigate the behavior of the backpropagation algorithm on edited data sets, some experiments were performed on two heavily overlapping circular symmetric Gaussian distributions. Data set 1 (see Fig.3) has both means on the origin (0,0) and standard deviations 0,4702 and 0,1736. Data set 2 (see Fig.4) consists of two classes with means (0,2, 0.2) and (-0.2, -0.2) and equal standard deviations 0.333. The Bayes error for both learning sets is 20% . Each set consists of 1000 samples.

For both learning sets, the original set and its edited version were used as a training set for a number of multilayer feedforward networks. The networks consist of 2 input units, 10,20,50 or 100 hidden units and 1 output unit. The parameters of the backpropagation algorithm were: learning rate η and momentum term α. A sample was considered to be correct when the difference between the actual output and the desired output was smaller than 0,5. The learning phase for the edited data sets was terminated when all samples were classified correctly.

Table 1 shown the perfomance of a trained network on a large test set. Figure 5 shows a plot of the average performance on the learning set as a function of the learning time.

4. Discussion

From the experiments that were performed, it appears that the backpropagation algorithm with an edited data set is a factor three faster when compared to a non-edited data set, *provided that we had a good criterion to stop the learning phase for a non-edited data set.* In practical situations therefore, the optimal stopping criterion results in a considerable improvement in speed. What we also gained with editing is the relative certainty that a reasonable classifier is found, due to the introduction of a large near-Bayes global minimum in the error space. Though, we are still not able to guarantee that the learning process finishes in the desired minimum; in fact for large networks one can still imagine *global* minima which are very undesirable, see [1].

Tab.1: *The average performance of 50 trained networks on a test set of 25,000 samples.*

average performance	data set 1 (figure 3)		date set 2 (figure 4)	
	non-edited (percent)	edited (percent)	non-edited (percent)	edited (percent)
10 hidden units	80.7	79.1	80.1	80.0
20 hidden units	80.3	79.0	80.1	80.1
50 hidden units	80.2	79.0	80.1	80.0
100 hidden units	80.1	78.7	80.0	80.0
nearest neighbor classifer	74.7	78.9	74.0	79.8

Furthermore, it is remarkable that the performance of a multi-layer network is not seriously deteriorated by the overlap in the learning set, as is the case with the nearest neighbor classifier. It is clear that the algorithm does not form small decision boundaries around outliers, as can be expected from a procedure that is based on the minimization of the mapping error. The search for a global minimum of the backpropagation algorithm is apparently constrained by a mechanism that prevents that too small groups of outliers form a separate decision boundary. If this is also the case for the Boltzmann machine [5] is doubtful, since the search for an error minimum of the Boltzmann machine is based on an optimization method which was designed to escape from local minima in the error space [6]. It is therefore to be expected that the Boltzmann machine more seriously suffers from defects as sketched in figure 1. However, it will therefore benefit even more from the editing procedure. Essentially, training a network with an edited data set provides a stopping criterion for *any* iterative learning procedure which optimizes the mean squared mapping error, whether this is backpropagation, some faster variant of backpropagation, the Boltzmann machine or any other iterative procedure. Furthermore, it is interesting to realize that our method changes the behavior of the learning rule, not by changing the learning rule itself, but by changing the data set. It might therefore be interesting to search for other operations on the data set which also influence the learning rule.

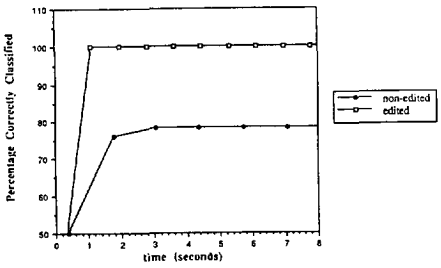


Fig.5: *The average performance of a network with 10 hidden units on the leraning set during the learning phase. Notice that leraning the edited data set stabilises at t=1, whereas leraning the non-edited data set stabilizes at t=9. Every point in this plot was a averaged over 50 instances of a network. The simulations where performed on a SUN 4/280 with a simulator written in C.*

Some additional remarks on the editing algorithm: In the first place, the editing step requires a certain amount of computational effort. This is for the data sets of figure 3 and figure 4 typically 20 seconds of CPU time on a SUN 4/280. Although this implies that the total requirements with respect to CPU time for the presented method are higher, the advantage is the certainty of the new stopping criterion. Without editing, one could train the network for minutes without ever being sure that the performance could not be improved.

In the second place, the editing algorithm requires that the data is sufficiently representative for the underlying distributions of the classes; i.e. when the data set is divided over N subsets, all these subsets must on their own be representative for the distributions.

Returning to the nearest-neighbor classifier, the step which usually follows the editing of a data set is called condensing [3]. The condensing algorithm removes all samples that are not close to the discriminating function, thereby heavily reducing the size of the learning set. For the nearest-neighbor classifier this results in a dramatic improvement in classification speed; instead of computing the distance to all samples in the learning set, only the distance to a few (discriminating) samples has to be determined.

It is interesting to notice a parallel of the condensing algorithm and the multi-layer perceptron: both represent the discriminating surface only (instead of the complete learning set). This leads to a large reduction of data resulting in an improvement of classification speed.

One can wonder if it is possible to speed up the learning phase of the backpropagation algorithm by training the network with a condensed learning set. The small subset containing the discriminating samples would concentrate the learning effort on the discriminating surface only. A little thought learns, however, that its is not always the case; the Euclidean metric which is used by the nearest-neighbor classifier to determine the distance to a sample, is not used by a multi-layer network. This is caused by the

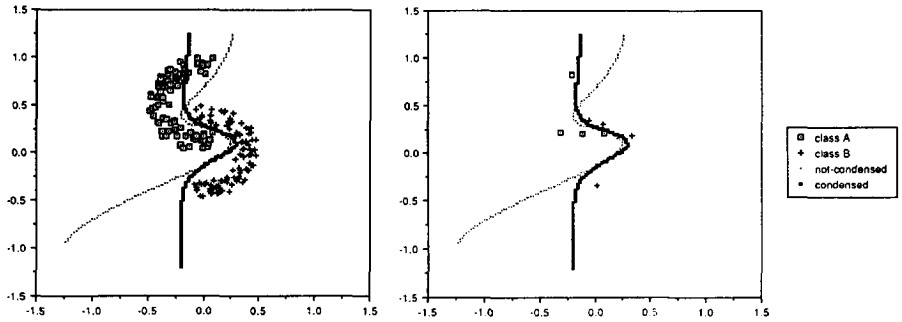


Fig.6: *A data set (left), and the data set after condensing. In both data sets the decision function of a network that was trained with the original data set, and the decision function of a network that was trained with the condensed set are drawn. Notice that the decison function of the condensed set does not correctly classify the original set*

inherent non-linearities of a multi-layer network. Whereas the backpropagation algorithm simply aims at separating classes, the nearest-neighbor classifier aims at separating the classes by drawing a line exactly in the middle between two samples (resulting in a Voronoi-tesselation of the features space). For the backpropagation algorithm it is not essential that the decision surface is exactly between two samples. As figure 6 shows, the differences between these metrics are so large that learning with a condensed learning set can give rise to a suboptimal classifier.

5. Conclusions

The editing algorithm effectively introduces a near-Bayes global minimum in the error space of the backpropagation algorithm. This has the advantages that it is very likely that the learning phase will end in this minimum and that the learning speed is moderately improved. Furthermore this editing technique results in an optimal stopping criterion. Since this stopping criterion is based on an operation on the data, instead of a change of the learning rule, it can be applied for any iterative learning procedure which optimizes the mean squared mapping error.

Experiments indicate that the search for the global minimum in the backpropagation algorithm is essentially constrained by a mechanism that prevents the formation of decision boundaries around small groups of outliers. This results in a very good performance for overlapping classes.

Acknowledgements This work was sponsored by the Dutch Government as a part of the SPIN/FLAIR-DIAC project, and by the Foundation of Computer Science in the Netherlands (SION) with financial support from the Dutch Organization for Scientific Research (NWO).

References

[1] Baum, E.B.: On the Capabilities of Multilayer Perceptrons. Journal of Complexity, Vol.4, 193-215, 1988.

[2] Bounds, *D.G.*, and Lloyd, P.J.: *A Multi-Layer Perceptron Network* for the Diagnosis of Low-Back Pain. Proceedings of the SGAICO conference 1988, University of Zürich, OCt., 1988.

[3] Devijver, P.A., and Kittler, J.: Pattern Recognition, A Statistical Approach. Prentice Hall, 1982.

[4] Gorman, R.P., and Sejnowski, T.J.: Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. Neural Networks, Vol.1, No. 1, 1988.

[5] Hinton, G.E., and Sejnowski, T.J.: Learning and Relearning Boltzmann machines. Chapter 7 in: Rumelhart, D.E., and McClelland, J.L.: Parallel Distributed Processing: Explorations in the miciro structure of cognition. Vol.1, MIT Press, 1986.

[6] Kirkpatrick, S., Gelatt, C.D.Jr., and Vecchi, M.P.: Optimization by Simulated Annealing. Science, Vol.220, 671-680, 1983.

[7] Kraaijveld, *M.S.*: *On the Application of Connectionist Models for* Pattern Recognition, Robotics and Computer Vision: a Technical Report. Delft University Press, Delft, The Netherlands, 1989.

[8] Kohonen, T., Barna, G., and Chrisley, R.: Statistical Pattern Recognition with Neural Networks: Benchmarking Studies. Proceesdins of the Neuro '88 conference, Paris, June 1988.

[9] Rumelhart, D.E., Hinton, G.E., and Williams, R.J.: Learning Internal Representations by Error Propagation. Chapter 8 in: Rumelhart, D.E., and McClelland, J.L.: Parallel Distributed Processing: Explorations in the micro structure of cognition. Vol.1, MIT Press, 1986.

[10] Sejnowski, T.J., and Rosenberg, C.R.: Parallel Networks that learn to Pronounce English Text. Complex Systems, Vol.1, 145-168, 1987.

[11] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K. and Lang, K.: Phoneme Recognition Using Time-Delay neural Networks. IEEE Trans. on ASSP, Vol.37, March 1989.

VII

The Learning Speed

R.R.N. Bakker, M.A. Kraaijveld, R.P.W. Duin and W.F. Schmidt

On the Speed of Training Networks with Correlated Features

Proceedings of the IEEE Conference on Neural Networks
(San Francisco, March 28 - April 1, 1993)
pp. 919 - 922.

On the Speed of Training Networks with Correlated Features

Robert R.N. Bakker, Martin A. Kraaijveld, Robert P.W. Duin and Wouter F. Schmidt
Pattern Recognition Group
Department of Applied Physics
Delft University of Technology
P.O. Box 5046
2600 GA Delft
The Netherlands
e-mail: martin@ph.tn.tudelft.nl

Abstract — The learning speed of the adaptive linear combiner is determined by the condition number of the input correlation matrix of the training data. With known properties of such correlation matrices, it will be shown that increasing the dimensionality of the feature space of an adaptive linear combiner will never increase its learning speed. In fact, the learning speed will at best remain equal, but will deteriorate in most cases. Our result can be applied in adaptive learning problems that are time constrained, and has possibly implications for the training of multi-layer networks.

I. Introduction

An important problem in adaptive signal processing and pattern recognition applications, is the determination of the number of features (i.e. the dimensionality of the input vector) for a learning system. Obviously when the number of features is high much information is being processed, and it is to be expected that a high accuracy can be reached, e.g. in a mean square sense [1], or with respect to the classification performance [2]. A second issue that is especially relevant in time constrained applications, is how the *learning speed* is being influenced by the dimensionality of the input data. To be more specific, in this paper we will investigate how the learning speed of a class of iterative learning procedures is affected by adding an extra feature. This class consists of the steepest descent procedure and the LMS procedure for the Adaptive Linear Combiner, the ALC, see [1]. For these procedures the learning speed is determined by the condition number of the input correlation matrix of the data; i.e. the ratio of the smallest and largest eigenvalues of the input correlation matrix.

A result from linear algebra will be discussed, that proves that the learning speed of the adaptive linear combiner is decreased, when a new correlated feature is added. This is based on the fact that the largest eigenvalue generally becomes larger and the smallest eigenvalue generally becomes smaller, due to adding the new feature. Only when the new feature is not correlated with the old features and it is properly scaled, the learning speed will not be affected.

In the following paragraphs, we will review how the learning speed is influenced by the condition number of the input correlation matrix of the training data. Then, in section 3, it will be proven that the condition number decreases when a new correlated feature is added. Finally, the discussion and conclusions are presented in section 4.

2. The Learning Speed and the Condition Number for the ALC.

An adaptive linear network, like the ALC, is described by the following simple function:

$$y_t = X_t^T W_t = \sum_{i=1}^{n} x_{t,i} w_{t,i} \qquad (1)$$

That is, where t is the time, the scalar output y_t is determined by the inner product of the (n- dimensional) input vector X_t and the weight vector W_t. During an iterative learning procedure, the weight vector W_t is adjusted such that the output y_t approximates a certain desired output as good as possible. For this learning procedure a set of pairs of input vectors X_t with corresponding desired output d_t is used. This learning set is supposed to represent the underlying phenomenon sufficiently well; i.e. the size of the learning set

This work was sponsored by the Dutch Government as a part of the SPIN / FLAIR-DIAC project, and by the Foundation of Computer Science in the Netherlands (SION) with financial support from the Dutch Organization for Scientific Research (NWO).

0-7803-0999-5/93/$03.00 ©1993 IEEE

should be large enough so that the learning set is statistically representative for the underlying problem.

As a measure for the quality of the approximation, a squared error criterion can be taken:

$$MSE \equiv \xi_t = E\left[(y_t - d_t)^2\right] = E\left[\varepsilon_t^2\right] \tag{2}$$

In order to make the necessary definitions, we follow Widrow [1] and express this quadratic error criterion in terms of some statistics of the training data:

$$E\left[\varepsilon_t^2\right] = E\left[d_t^2\right] + W_t^T R W_t - 2P^T W_t \tag{3}$$

Here **R** is the semi-positive definite symmetric input correlation matrix, defined as $E\left[X_t X_t^T\right]$ and P is the vector $E\left[d_t X_t\right]$. From (3) it is clear that the MSE is a quadratic form in W_t. The optimal weight vector W^*, i.e. the location where ξ has its minimal value ξ_{min}, is found by taking the derivative with respect to W. ξ_t can be shown to depend only on the distance between the actual and optimal weight vectors and the input correlation matrix:

$$\xi_t = \xi_{min} + \left(W_t - W^*\right)^T R \left(W_t - W^*\right) \tag{4}$$

Now the coordinate system W can be translated to a system V, such that $\xi_{min} = \xi\big|_{W^*}$ is located at the origin, followed by a rotation to a coordinate system V' in which the V'-axes coincide with the principal axes of the paraboloid ξ-surface:

$$\begin{aligned} \xi_t &= \xi_{min} + \left(W_t - W^*\right)^T R \left(W_t - W^*\right) \\ &= \xi_{min} + V_t^T R V_t \\ &= \xi_{min} + V_t^T \left(Q \Lambda Q^T\right) V_t \\ &= \xi_{min} + V'^T_t \Lambda V'_t \end{aligned} \tag{5}$$

Due to this transformation, **R** is replaced by its eigenvalue matrix Λ. The eigenvectors Q_i $(1 \le i \le n)$ of the input correlation matrix define the principal axes of the error surface and the eigenvalues λ_i $(1 \le i \le n)$ give the second derivatives of the error surface ξ with respect to its principal axes.

In the case of steepest descent minimization of ξ, the weight vector is adjusted as follows:

$$W_{t+1} = W_t - \eta \nabla_t \tag{6}$$

where ∇_t is the gradient and η represents the step-size or learning rate. Due to this iterative learning procedure, the trajectory in weight space can be expressed as*:

$$W_t = W^* + (I - 2\eta R)^t \left(W_0 - W^*\right) \tag{7}$$

Note that (7) gives the weight vector at any time t as a function of the weight vector W_0 at time zero. In terms of V' (7) can be formulated as:

$$V'_t = (I - 2\eta\Lambda)^t V'_0 \tag{8}$$

from which it is apparent that for a stable learning behavior we need to choose:

$$0 < \eta < \frac{1}{\lambda_{max}} \tag{9}$$

Also, from (8) it is clear that the speed of the learning procedure is limited by the smallest eigenvalue λ_{min}, since the term with λ_{min} is the slowest converging term in (8).. Apparently, the condition number of **R**, defined as the ratio of the smallest and the largest eigenvalue, is a good measure for the speed of the iterative learning procedure. In the next section the effect on the learning speed and the condition number caused by adding an extra feature x_{n+1} will be discussed.

3. ADDING A (CORRELATED) FEATURE.

Adding a new feature to the existing set of n features, implies that the matrix **R** is bordered with an n-dimensional vector U and a new diagonal element σ^2.

$$R^{n+1} = \begin{pmatrix} \left[R^n\right] & U \\ U^T & \sigma^2 \end{pmatrix} \tag{10}$$

The question that should be answered now is how the eigenvalues of R are affected by this bordering operation? The answer to this question can be found in a number of textbooks on linear algebra, e.g. see [3 - 5]. It is based on the Courant-Fisher theorem and the Rayleigh theorem and states that bordering the matrix will not decrease the largest eigenvalue, and not increase the smallest eigenvalue of the matrix. The condition number of the matrix will therefore deteriorate, or at best remain equal.

* For the LMS learning procedure, or Widrow-Hoff rule [Duda 1973], the gradient is replaced by an unbiased estimate of the gradient. Equation (7) is then approximately valid.

Here, we will go one step further than the previous (qualitative) proofs, by not only showing that the condition number deteriorates, but we will also provide an implicit equation for the new eigenvalues in terms of the old eigenvalues.

A new eigenvector Q', associated with a λ', obeys the following relation:

$$R^{n+1}Q' = \lambda' Q' \tag{11}$$

Now Q' is written as combination of an n-dimensional vector Ω and a scalar β:

$$\begin{pmatrix} R^n & U \\ U^T & \sigma^2 \end{pmatrix} Q' = \lambda' Q' \Leftrightarrow \begin{pmatrix} R^n & U \\ U^T & \sigma^2 \end{pmatrix} \begin{pmatrix} \Omega \\ \beta \end{pmatrix} = \lambda' \begin{pmatrix} \Omega \\ \beta \end{pmatrix} \tag{12}$$

This equality can be split into two parts. By writing Ω as a vector Ψ that is rotated by the matrix Q, the first part can be expressed as a matrix equation of the form:

$$\left.\begin{aligned} R^n\Omega + \beta U = \lambda'\Omega \\ \Omega = Q\Psi \end{aligned}\right\} R^n Q\Psi + \beta U = \lambda' Q\Psi \Leftrightarrow$$

$$Q\Lambda\Psi + \beta U = \lambda' Q\Psi \tag{13}$$

Note that equation (5) and the property $Q^TQ = I$ were used. The second equation that follows from (12) is the scalar equation:

$$\left.\begin{aligned} U^T\Omega + \sigma^2\beta = \lambda'\beta \\ \Omega = Q\Psi \end{aligned}\right\} U^T Q\Psi = \beta\left(\lambda' - \sigma^2\right) \tag{14}$$

If (13) is multiplied with Q^T on both sides of the equality sign, we find that:

$$\Lambda\Psi + \beta Q^T U = \lambda'\Psi \Leftrightarrow \Psi = \beta(\lambda' I - \Lambda)^{-1} Q^T U \tag{15}$$

Substituting Ψ in (14) yields:

$$U^T Q\left\{\beta(\lambda' I - \Lambda)^{-1}\right\} Q^T U = \beta\left(\lambda' - \sigma^2\right) \tag{16}$$

Which can finally be rewritten as:

$$\sum_{i=1}^{n} \frac{\left(U^T Q_i\right)^2}{\left(\lambda' - \lambda_i\right)} = \left(\lambda' - \sigma^2\right) \tag{17}$$

This is an implicit equation for the new eigenvalues λ' in terms of the old eigenvalues λ_i. It is easily verified that the derivative of the left part of equation (17) with respect to λ' is always negative, with asymptotes for $\lambda' = \lambda_i$. The new eigenvalues are found at the locations where the left part of (17) crosses the linear function $(\lambda' - \sigma^2)$, see figure 1. From figure 1 it is also seen that there is a new eigenvalue left of each old eigenvalue, and one extra eigenvalue being larger than the largest original eigenvalue. This qualitative finding confirms the proofs that are based on the Courant-Fisher and Rayleigh theorems, e.g. see [3 - 5]. Note that the eigenvalues do not change when the new feature is not correlated with the previous features, i.e. when $U = (0, 0, ..., 0)^T$. Also note that the fact that R^{n+1} is a correlation matrix assures that all new eigenvalues are non-negative.

4. DISCUSSION AND CONCLUSIONS

Two direct conclusions of the foregoing theory are, that the smallest eigenvalue will not increase nor will the largest eigenvalue decrease by adding an extra feature. The first conclusion implies that the rate of convergence of equation (8) becomes slower, and the second conclusion necessitates the choice of a smaller step-size η (in equation 9), thereby also constraining the learning speed. It is important to note that especially this second aspect restricts the learning speed in practice. In practical applications the smallest eigenvalues may have neglectable influence on the final error ξ, so the learning phase is generally terminated after some application dependent time interval. Small eigenvalues becoming smaller will probably not influence this time interval. The fact that the largest eigenvalue becomes larger is therefore of much more influence, since a smaller step-size constrains the speed of the total learning process.

It should also be noted that besides these fundamental restrictions on the learning speed, there is also the fact that an increase of the number of features requires more computations per step. This was not taken into account here, but may be a relevant factor in practical applications.

Another issue that needs to be addressed is that our results are in line with previous results that claim that the learning speed is increased by decorrelating and scaling of the data, e.g. see [6] and [7]. From (8) it is easily seen that the learning speed is maximum when all eigenvalues are equal, whereas our results show that only when the new feature is not correlated and properly scaled there is no decrease in learning speed.

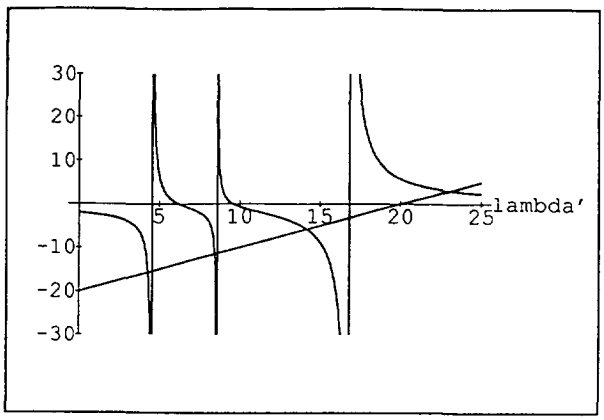


Figure 1: A numerical example of the theory. The matrix ((5, 1, 2), (1, 10, 3), (2, 3, 15)) with eigenvalues 4.58, 8.59 and 16.83 is bordered with the vector $(1, 2, 3, 20)^T$. The left side of equation (17) is the function with asymptotes at the original eigenvalues. The linear function (λ' - 20) corresponds to the right side of (17). The new eigenvalues are given by the locations where the two functions cross, i.e. at 4.57, 8.57, 14.42 and 22.45.

Finally, an important question is how our result generalizes towards more complex adaptive systems like multi-layer feed forward networks, e.g. see [8]. The problem with multi-layer feed forward networks is that the system is not linear, which has the effect of local minima in the error landscape. Obviously, where the error landscape can be locally approximated by a second order approximation, our result holds. Therefore, (as long as the approximation is valid) the learning will locally slow down. When the approximation is not valid anymore, the problem may emerge that the extension of the parameter space induces a new path to a close local minimum. This local minimum may then be reached much faster than some (local) minimum in the original space. One conclusion that does generalize towards non-linear systems, however, is that the learning rate should be taken smaller when the number of features is increased.

REFERENCES

[1] Widrow, B., and Stearns, S.D., *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey 1985.

[2] Jain, A.K., and Chandrasekaran, B., "Dimensionality and Sample Size Considerations in Pattern Recognition Practice", in: P.R. Krishnaiah and L.N. Kanal, editors, *Handbook of Statistics*, Vol. 2, North Holland Publishing Company, pp. 835-855, 1982.

[3] Lancaster, P., and Tismenetsky, M., *The Theory of Matrices*, second edition, Academic Press, 1985.

[4] Horn, R.A., and Johnson, C.R., *Matrix Analysis*, Cambridge University Press, Cambridge, 1988.

[5] Strang, G., *Linear Algebra and its Applications*, Harcourt Brace Jovanovich Publishers, San Diego, 1988.

[6] le Cun, Y., Kanter, I., and Solla, S.A., "Eigenvalues of Covariance Matrices: Applications to Neural-Network Learning", Physical Review Letters, Vol. 66, Nr. 18, May 6, 1991, pp. 2396 - 2399.

[7] Orfanidis, S.J., "Gram-Schmidt Neural Nets", Neural Computation, Vol. 2, pp. 116 - 126, 1990.

[8] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation", in: *"Parallel Distributed Processing: Explorations in the Microstructure of Cognition"*, Vol. 1, Rumelhart, D.E., and McClelland, J.L. (eds.), Cambridge, MA., MIT-Press, pp. 318-362.

A

Proof of theorem 2.5

In this appendix a bound on the uniform convergence of the one-sided relative deviation of the empirical error to the true error will be presented. The bound that is presented here is a slight improvement of a result of Vapnik [Vapnik 1982].

We will prove the following theorem:

Theorem 2.5: *Let $F(\boldsymbol{X},\boldsymbol{\Theta})$ be a set of classification functions with finite capacity V and let Ξ^m be an i.i.d. sample of size m, drawn according to some distribution D. For $m > \max\left(2/\chi^2, 4V/\chi^2\right)$, the probability that there exists a classifier $f(\boldsymbol{x},\boldsymbol{\theta}) \in F(\boldsymbol{X},\boldsymbol{\Theta})$ for which the one-sided relative deviation of the error probability $\varepsilon(\boldsymbol{\theta})$ to the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ on Ξ^m, is larger than χ, is bounded by:*

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}} > \chi\right\} \le 4S^F_{\max}\left(m(\beta+1)\right)\exp\left(-\frac{1}{2}m\chi^2\left(\frac{\beta}{\beta+1}\right)^2\right) \qquad (1)$$

where β is given by:

$$\beta = \frac{1}{m}\left\langle \frac{m^2\chi^2}{2V}\left(1+\sqrt{1-\frac{4V}{m\chi^2}}\right) - m \right\rangle \qquad (2)$$

The proof of this theorem will be presented in a number of steps. First, a number of definitions will be given. Consider two i.i.d. samples in $\Re^d \times [0, 1]$ of size m and size βm, such that $m > 0$, $\beta > 0$ and $\beta m \in \aleph$:

$$\begin{aligned} \Xi_1^m &= (\xi_1,\lambda_1),(\xi_2,\lambda_2),.....,(\xi_m,\lambda_m) \\ \Xi_2^{\beta m} &= (\xi_{m+1},\lambda_{m+1}),(\xi_{m+2},\lambda_{m+2}),.....,\left(\xi_{m(\beta+1)},\lambda_{m(\beta+1)}\right) \end{aligned} \qquad (3)$$

Here, the $\xi_i \in \Re^d$ are the measurements or observations, and $\lambda_i \in \{0, 1\}$ are the class labels, with $i = 1, \ldots\ldots, m(\beta+1)$. The error frequencies of a classifier $f(x, \theta) \in F(X, \Theta)$ on Ξ_1^m and $\Xi_2^{\beta m}$ are denoted as $\hat{\varepsilon}_1(\theta)$ and $\hat{\varepsilon}_2(\theta)$ respectively, whereas the true error of $f(x, \theta)$ is denoted $\varepsilon(\theta)$. We define the event Q_1 in the space of samples of size m:

$$Q_1 = \left\{ \sup_{\theta \in \Theta} \frac{\varepsilon(\theta) - \hat{\varepsilon}(\theta)}{\sqrt{\varepsilon(\theta)}} > \chi \right\} \tag{4}$$

the event Q_2 *in* the space of samples of size $m(\beta+1)$:

$$Q_2 = \left\{ \sup_{\theta \in \Theta} \frac{\hat{\varepsilon}_2(\theta) - \hat{\varepsilon}_1(\theta)}{\sqrt{\frac{\hat{\varepsilon}_1(\theta) + \beta\hat{\varepsilon}_2(\theta)}{\beta+1} + \frac{1}{m(\beta+1)}}} > \chi \right\} \tag{5}$$

and the quantity $R_\theta\left(\Xi^{m(\beta+1)}\right)$:

$$R_\theta\left(\Xi^{m(\beta+1)}\right) = \frac{\hat{\varepsilon}_2(\theta) - \hat{\varepsilon}_1(\theta)}{\sqrt{\frac{\hat{\varepsilon}_1(\theta) + \beta\hat{\varepsilon}_2(\theta)}{\beta+1} + \frac{1}{m(\beta+1)}}} \tag{6}$$

First, it will be shown that the following relation between the probabilities of the events Q_1 and Q_2 holds:

Lemma A.1: *For* $m > \max\left(2/\left(\beta\chi^2\right), 1/\chi^2\right)$, *the following inequality is valid:*

$$P(Q_1) \leq 4\, P(Q_2) \tag{7}$$

The proof of this lemma is a generalized version of a similar result in [Vapnik 1982].

First, denote with I the indicator function:

$$I(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{8}$$

The proof follows from the observation that:

$$P(Q_2) \equiv \int_{\Xi^{m(\beta+1)}} I\left(\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \frac{\hat{\varepsilon}_2(\boldsymbol{\theta}) - \hat{\varepsilon}_1(\boldsymbol{\theta})}{\sqrt{\frac{\hat{\varepsilon}_1(\boldsymbol{\theta}) + \beta\hat{\varepsilon}_2(\boldsymbol{\theta})}{\beta+1} + \frac{1}{m(\beta+1)}}} - \chi \right) dP\left(\Xi^{m(\beta+1)}\right) \tag{9}$$

$$= \int_{\Xi^{m(\beta+1)}} I\left(\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} R_{\boldsymbol{\theta}}\left(\Xi^{m(\beta+1)}\right) - \chi \right) dP\left(\Xi^{m(\beta+1)}\right) \tag{10}$$

$$= \int_{\Xi_1^m} dP\left(\Xi_1^m\right) \int_{\Xi_2^{\beta m}} I\left(\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} R_{\boldsymbol{\theta}}\left(\Xi^{m(\beta+1)}\right) - \chi \right) dP\left(\Xi_2^{\beta m}\right) \tag{11}$$

$$\geq \int_{Q_1} dP\left(\Xi_1^m\right) \int_{\Xi_2^{\beta m}} I\left(\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} R_{\boldsymbol{\theta}}\left(\Xi^{m(\beta+1)}\right) - \chi \right) dP\left(\Xi_2^{\beta m}\right) \tag{12}$$

Now on Q_1, there is an $f(\boldsymbol{x}, \boldsymbol{\theta}^*)$ such that (cf. equation (4)):

$$\varepsilon\left(\boldsymbol{\theta}^*\right) - \hat{\varepsilon}\left(\boldsymbol{\theta}^*\right) > \chi\sqrt{\varepsilon\left(\boldsymbol{\theta}^*\right)} \tag{13}$$

Since $\hat{\varepsilon}\left(\boldsymbol{\theta}^*\right) \geq 0$ this implies that:

$$\varepsilon\left(\boldsymbol{\theta}^*\right) > \chi^2 \tag{14}$$

Returning to (9 - 12) it is clear that for a single classifier $f(\boldsymbol{x}, \boldsymbol{\theta}^*)$ it follows that:

$$P(Q_2) \geq \int_{Q_1} dP\left(\Xi_1^m\right) \int_{\Xi_2^{\beta m}} I\left(\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} R_{\boldsymbol{\theta}}\left(\Xi^{m(\beta+1)}\right) - \chi \right) dP\left(\Xi_2^{\beta m}\right) \tag{15}$$

$$\geq \int_{Q_1} dP\left(\Xi_1^m\right) \int_{\Xi_2^{\beta m}} I\left(R_{\boldsymbol{\theta}^*}\left(\Xi^{m(\beta+1)}\right) - \chi \right) dP\left(\Xi_2^{\beta m}\right) \tag{16}$$

If we assume that:

$$\hat{\varepsilon}_2(\boldsymbol{\theta}^*) > \varepsilon(\boldsymbol{\theta}^*) \tag{17}$$

we can use the same approach as in [Vapnik 1982, pp. 177 - 178], where a similar result is proven for the special case $\beta = 1$, to show that $R_{\boldsymbol{\theta}^*}\left(\Xi^{m(\beta+1)}\right)$ attains its minimum value when:

$$\begin{aligned} \hat{\varepsilon}_1(\boldsymbol{\theta}^*) &= \varepsilon(\boldsymbol{\theta}^*) - \chi\sqrt{\varepsilon(\boldsymbol{\theta}^*)} \\ \hat{\varepsilon}_2(\boldsymbol{\theta}^*) &= \varepsilon(\boldsymbol{\theta}^*) \end{aligned} \tag{18}$$

Substitution of (18) in (6) yields:

$$R_{\boldsymbol{\theta}^*}\left(\Xi^{m(\beta+1)}\right) > \frac{\chi\sqrt{(\beta+1)\varepsilon(\boldsymbol{\theta}^*)}}{\sqrt{(\beta+1)\varepsilon(\boldsymbol{\theta}^*) - \chi\sqrt{\varepsilon(\boldsymbol{\theta}^*)} + \frac{1}{m}}} \tag{19}$$

With (14) and choosing $m > 1/\chi^2$ it follows that $R_{\boldsymbol{\theta}^*}\left(\Xi^{m(\beta+1)}\right) > \chi$. Thus, $R_{\boldsymbol{\theta}^*}\left(\Xi^{m(\beta+1)}\right) > \chi$, whenever $m > 1/\chi^2$ and the assumption $\hat{\varepsilon}_2(\boldsymbol{\theta}^*) > \varepsilon(\boldsymbol{\theta}^*)$ from inequality (17) is true. From this we obtain that:

$$P(Q_2) \geq \int_{Q_1} dP\left(\Xi_1^m\right) \int_{\Xi_2^{\beta m}} I\left(\hat{\varepsilon}_2(\boldsymbol{\theta}^*) - \varepsilon(\boldsymbol{\theta}^*)\right) dP\left(\Xi_2^{\beta m}\right) \tag{20}$$

The second integral in (20) has now become independent of the first integral and is equal to the probability that for a sample of size βm, the error frequency is larger than the true error. With known properties of the binomial distribution it can be shown that $P\left\{\hat{\varepsilon}_2(\boldsymbol{\theta}^*) > \varepsilon(\boldsymbol{\theta}^*)\right\} > 1/4$, if $\beta m > 2/\varepsilon(\boldsymbol{\theta}^*)$ [Vapnik 1982]. Since the two integrals in (20) are independent, this value 1 / 4 can be substituted for the second integral:

$$P(Q_2) \geq \frac{1}{4}\int_{Q_1} dP\left(\Xi_1^m\right) = \frac{1}{4}P(Q_1) \tag{21}$$

Since $\varepsilon(\boldsymbol{\theta}^*) > \chi^2$ implies that $2/\chi^2 > 2/\varepsilon(\boldsymbol{\theta}^*)$ it follows that for $m > \max\left(2/(\beta\chi^2), 1/\chi^2\right)$ the inequality $P(Q_1) \leq 4\,P(Q_2)$ is valid. This concludes the proof of lemma A.1.

■

The next step in the proof of theorem 2.5 is a bound on the probability of event Q_2, since lemma A.1 then allows a bound on the probability of the event Q_1. We will prove the following lemma:

Lemma A.2: *The probability of event Q_2 is bounded by:*

$$P\{Q_2\} \leq S^F_{\max}(m(\beta+1))\exp\left(-\frac{1}{2}m\chi^2\left(\frac{\beta}{\beta+1}\right)^2\right) \tag{22}$$

For the proof of this lemma we proceed as Devroye [Devroye 1982] and denote with T_i one the $m(\beta+1)!$ permutations of the space $\Xi^{m(\beta+1)}$ onto itself. Now, it follows that:

$$P(Q_2) \equiv \int_{\Xi^{m(\beta+1)}} I\left(\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} R_{\boldsymbol{\theta}}\left(\Xi^{m(\beta+1)}\right) - \chi\right) dP\left(\Xi^{m(\beta+1)}\right) \tag{23}$$

$$= \int_{\Xi^{m(\beta+1)}} \frac{\sum_{i=1}^{(m(\beta+1))!} I\left(\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} R_{\boldsymbol{\theta}}\left(T_i\Xi^{m(\beta+1)}\right) - \chi\right)}{(m(\beta+1))!} dP\left(\Xi^{m(\beta+1)}\right) \tag{24}$$

$$= \int_{\Xi^{m(\beta+1)}} \frac{\sum_{i=1}^{(m(\beta+1))!} \sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} I\left(R_{\boldsymbol{\theta}}\left(T_i\Xi^{m(\beta+1)}\right) - \chi\right)}{(m(\beta+1))!} dP\left(\Xi^{m(\beta+1)}\right) \tag{25}$$

If two classifiers $f(\boldsymbol{x}, \boldsymbol{\theta}_1)$ and $f(\boldsymbol{x}, \boldsymbol{\theta}_2) \in F(\boldsymbol{X}, \boldsymbol{\Theta})$ induce the same dichotomy on $\Xi_1^m \cup \Xi_2^{\beta m}$ in (3), then $\hat{\varepsilon}_1(\boldsymbol{\theta}_1) = \hat{\varepsilon}_1(\boldsymbol{\theta}_2)$, $\hat{\varepsilon}_2(\boldsymbol{\theta}_1) = \hat{\varepsilon}_2(\boldsymbol{\theta}_2)$ and therefore also $R_{\boldsymbol{\theta}_1}\left(\Xi^{m(\beta+1)}\right) = R_{\boldsymbol{\theta}_2}\left(\Xi^{m(\beta+1)}\right)$ and $R_{\boldsymbol{\theta}_1}\left(T_i\Xi^{m(\beta+1)}\right) = R_{\boldsymbol{\theta}_2}\left(T_i\Xi^{m(\beta+1)}\right)$, with $i = 1,..,$ $m(\beta+1)!$. Due to this fact, the supremum over $\boldsymbol{\Theta}$ in (24) can be replaced by a supremum

over a *finite* set of classifiers $\boldsymbol{\Theta}^*$, which contains one representative $f(\boldsymbol{x}, \boldsymbol{\theta})$ for each dichotomy on $\Xi_1^m \cup \Xi_2^{\beta m}$ in (3). Proceeding from (25) we derive:

$$P(Q_2) = \int_{\Xi^{m(\beta+1)}} \frac{\sum_{i=1}^{(m(\beta+1))!} \sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}^*} I\left(R_{\boldsymbol{\theta}}\left(T_i \Xi^{m(\beta+1)}\right) - \chi\right)}{(m(\beta+1))!} dP\left(\Xi^{m(\beta+1)}\right) \tag{26}$$

$$\leq \int_{\Xi^{m(\beta+1)}} \frac{\sum_{i=1}^{(m(\beta+1))!} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}^*} I\left(R_{\boldsymbol{\theta}}\left(T_i \Xi^{m(\beta+1)}\right) - \chi\right)}{(m(\beta+1))!} dP\left(\Xi^{m(\beta+1)}\right) \tag{27}$$

$$= \int_{\Xi^{m(\beta+1)}} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}^*} \left\{ \frac{\sum_{i=1}^{(m(\beta+1))!} I\left(R_{\boldsymbol{\theta}}\left(T_i \Xi^{m(\beta+1)}\right) - \chi\right)}{(m(\beta+1))!} \right\} dP\left(\Xi^{m(\beta+1)}\right) \tag{28}$$

The quantity between { } in (28) is equal to the fraction of the permutations of the sample $\Xi_1^m \cup \Xi_2^{\beta m}$ for which $R_{\boldsymbol{\theta}}\left(T_i \Xi^{m(\beta+1)}\right) > \chi$. This fraction is equal to the probability, that if we pick one permutation from the set of $m(\beta+1)!$ permutations, the event $\left\{R_{\boldsymbol{\theta}}\left(T_i \Xi^{m(\beta+1)}\right) > \chi\right\}$ is true.

To determine this fraction, we bound the corresponding probability and denote $\boldsymbol{Y}^{m(\beta+1)} = y_1, y_2, \ldots\ldots, y_{m(\beta+1)}$ as a permutation of $\boldsymbol{Z}^{m(\beta+1)} = z_1, z_2, \ldots\ldots, z_{m(\beta+1)}$, with $z_i = \left(f(\boldsymbol{x}_i, \boldsymbol{\theta}) - \lambda_i\right)^2$ for $i = 1, \ldots\ldots, m(\beta+1)!$; i.e. $z_i = 0$ if $f(\boldsymbol{x}_i, \boldsymbol{\theta}) = \lambda_i$ and $z_i = 1$ otherwise. Furthermore, when μ is defined as:

$$\mu = \frac{1}{m(\beta+1)} \sum_{i=1}^{m(\beta+1)} y_i = \frac{1}{m(\beta+1)} \sum_{i=1}^{m(\beta+1)} z_i \tag{29}$$

we derive:

$$\frac{\sum_{i=1}^{(m(\beta+1))!} I\left(R_{\theta}\left(T_i \Xi^{m(\beta+1)}\right)-\chi\right)}{(m(\beta+1))!} \tag{30}$$

$$= P\left(\frac{\frac{1}{\beta m}\sum_{i=m+1}^{m(\beta+1)} y_i - \frac{1}{m}\sum_{i=1}^{m} y_i}{\sqrt{\frac{\frac{1}{m}\sum_{i=m+1}^{m(\beta+1)} y_i + \frac{1}{m}\sum_{i=1}^{m} y_i}{\beta+1} + \frac{1}{m(\beta+1)}}} > \chi \right) \tag{31}$$

$$= P\left(\frac{\frac{1}{\beta m}\left\{ m(\beta+1)\mu - \sum_{i=1}^{m} y_i \right\} - \frac{1}{m}\sum_{i=1}^{m} y_i}{\sqrt{\mu + \frac{1}{m(\beta+1)}}} > \chi \right) \tag{32}$$

$$= P\left(\frac{\mu - \frac{1}{m}\sum_{i=1}^{m} y_i}{\sqrt{\mu + \frac{1}{m(\beta+1)}}} > \frac{\beta}{(\beta+1)}\chi \right) \tag{33}$$

$$\leq P\left(\frac{\mu - \frac{1}{m}\sum_{i=1}^{m} y_i}{\sqrt{\mu}} > \frac{\beta}{(\beta+1)}\chi \right) \tag{34}$$

$$\leq \exp\left(-\frac{1}{2} m\chi^2 \left(\frac{\beta}{\beta+1} \right)^2 \right) \tag{35}$$

For the step from (34) to (35) inequality 2.6 in chapter 2 was used. Note that the probability (31 - 34) is based on sampling without replacement, whereas inequality 2.7 is a bound for sampling with replacement. With (35) we return to (28) to obtain:

$$P(Q_2) \leq \int_{\Xi^{m(\beta+1)}} \sum_{\theta \in \Theta^*} \exp\left(-\frac{1}{2} m\chi^2 \left(\frac{\beta}{\beta+1}\right)^2\right) dP\left(\Xi^{m(\beta+1)}\right) \tag{36}$$

$$= S^F_{\exp}\left(m(\beta+1)\right) \exp\left(-\frac{1}{2} m\chi^2 \left(\frac{\beta}{\beta+1}\right)^2\right) \tag{37}$$

where $S^F_{\exp}\left(m(\beta+1)\right)$ is the *expected* number of dichotomies that can be induced on an arbitrary set of $m(\beta+1)$ points. Obviously, the expected number of dichotomies is smaller than the *maximum* number of dichotomies, as given in theorem 2.2. Substitution of the maximum number of dichotomies in (37), however, yields a bound that is distribution free:

$$P\{Q_2\} \leq S^F_{\max}\left(m(\beta+1)\right) \exp\left(-\frac{1}{2} m\chi^2 \left(\frac{\beta}{\beta+1}\right)^2\right) \tag{38}$$

This concludes the proof of lemma A.2.

■

In the previous derivations, the parameter β is an arbitrary constant larger than zero. Contrary to previous and related work (e.g. see [Vapnik 1971b], [Vapnik 1982], [Vapnik 1992], [Devroye 1982], [Blumer 1986] and [Blumer 1989]), the formulation of (38) allows the determination of an approximately optimal value for β. This is shown in the following lemma:

Lemma A.3: *For $m \geq 4V/\chi^2$, the bound on the probability of event Q_2 in lemma A.2 is minimized when:*

$$\beta = \frac{1}{m}\left\langle \frac{m^2\chi^2}{2V}\left(1+\sqrt{1-\frac{4V}{m\chi^2}}\right) - m \right\rangle \tag{39}$$

For $m < 4V/\chi^2$ the bound on the probability of Q_2 is trivial with $P(Q_2) = 1$, for the optimal value $\beta = 0$.

In order to quantify the maximum number of dichotomies $S^F_{\max}\left(m(\beta+1)\right)$ in (38) the bound of theorem 2.3 is substituted to obtain that for $m > V$:

$$P\{Q_2\} \leq \left(\frac{em(\beta+1)}{V}\right)^V \exp\left(-\frac{1}{2}m\chi^2\left(\frac{\beta}{\beta+1}\right)^2\right) \tag{40}$$

If we define:

$$g(\beta) = \left(\frac{em(\beta+1)}{V}\right)^V \exp\left(-\frac{1}{2}m\chi^2\left(\frac{\beta}{\beta+1}\right)^2\right) \tag{41}$$

an approximately optimal value for β can be found by differentiation of (41) with respect to β. For convenience we take the derivative of the logarithm of (41). After some algebra it follows that:

$$\frac{\partial \ln(g(\beta))}{\partial \beta} = \frac{V\beta^2 + (2V - m\chi^2)\beta + V}{(\beta+1)^3} \tag{42}$$

Setting the derivative of the logarithm of (41) to zero yields:

$$\beta = \frac{m\chi^2 - 2V \pm \sqrt{m\chi^2(m\chi^2 - 4V)}}{2V} = \frac{m\chi^2}{2V}\left(1 \pm \sqrt{1 - \frac{4V}{m\chi^2}}\right) - 1 \tag{43}$$

It is easily verified that the largest value of β corresponds to a minimum of (41), whereas the smallest value corresponds to a maximum. Since β should necessarily have a value such that $\beta m \in \mathbb{N}$ (cf. equation (3)), its value is slightly adapted such that βm is rounded off to the nearest integer:

$$\beta = \frac{1}{m}\left\langle \frac{m^2\chi^2}{2V}\left(1 + \sqrt{1 - \frac{4V}{m\chi^2}}\right) - m \right\rangle \tag{44}$$

Equation (44) is valid as long as $m > 4V/\chi^2$. For $m < 4V/\chi^2$ the function $\partial \ln(g(\beta)) / \partial \beta$ does not have any real roots. Inspection of (42) learns that the derivative of $g(\beta)$ for $m < 4V/\chi^2$ is always larger than zero, so in order to find the minimum value of $g(\beta)$ the smallest allowable value of β must be substituted. Substituting $\beta = 0$ in (40) however yields:

$$P\{Q_2\} \leq \left(\frac{em}{V}\right)^V \tag{45}$$

which is trivial since $m > V$. This concludes the proof of lemma A.3.

■

For the proof of theorem 2.5 with lemma's A.1, A.2, and A.3 it only needs to be verified that the conditions under which lemma A.1 and lemma A.3 are valid do not contradict. With $m \geq 4V/\chi^2$ it is found that the minimum value of β in (44) is equal to 1. This implies in lemma A.1 that we can choose $m > 2/\chi^2$, since $2/\chi^2 \geq \max\left(2/(\beta\chi^2), 1/\chi^2\right)$. For $V \geq 1$ it is obvious that the condition $m \geq 4V/\chi^2$ in lemma A.3 is more restrictive than the condition $m > 2/\chi^2$. For $V = 0$, however, we need $m > \max\left(2/\chi^2, 4V/\chi^2\right)$. This concludes the proof of theorem 2.5.

■

B

The Functions $z_1(x)$ and $z_2(x)$

In this appendix two functions that will be used in appendices C and E are presented. Since no closed form expression for these functions exists, they are plotted and tabulated in this appendix.

B.1. The function $z_1(x)$

In appendix C we will use the function $z_1(x)$ which is defined as the positive root in y of $\ln(y+1)+x-y/2$ as a function of x, with $x \geq 1$:

$$z_1(x) = \left\{ y > 0 \mid \ln(y+1)+x-y/2 = 0;\ x \geq 1 \right\} \tag{1}$$

Since the domain of interest is $x > 1$, it is easily seen that there is one unique location in y where the functions $\ln(y+1)+x$ and $y/2$ cross. The location of this crossing, as a function of x, is given by the function $z_1(x)$. Although this function can not be formulated in a regular closed form, it is well defined and easily computed with a digital computer. Figure 1 shows the function $z_1(x)$. In table 1, $z_1(x)$ is tabulated for x ranging from 1 to 1000. A good closed form approximation of $z_1(x)$ with a maximum approximation error of well below 3%, for x between 1 and 10,000, is given by the function $a_1(x)$:

$$a_1(x) = 2\left(x+\ln(1+x)\right)+2.5 \tag{2}$$

As on the scale of figure 1 there is no visible difference between $z_1(x)$ and $a_1(x)$, the function $a_1(x)$ is also tabulated in table 1.

B.2. The function $z_2(x)$

Similar to the previous definition of $z_1(x)$, we will define a function $z_2(x)$ as the positive root in y of $\ln\left((y+1)^3/y^2\right)+x-(y-2)/2$ as a function of x, with $x \geq 1$:

$$z_2(x) = \left\{ y > 0 \;\middle|\; \ln\left((y+1)^3 / y^2\right) + x - (y-2)/2 = 0; x \geq 1 \right\} \tag{3}$$

This function will be applied in appendix E. Within the domain of interest, $y > 0$, $x \geq 1$, $z_2(x)$, provides the (unique) location where the functions $\ln\left((y+1)^3 / y^2\right) + x$ and $(y-2)/2$ cross. The function $z_2(x)$ is plotted in figure 2, and tabulated in table 2. A closed form approximation of $z_2(x)$ is given by the function $a_2(x)$:

$$a_2(x) = 2x + 0.7\ln(1+x) + 6.5 \tag{4}$$

which has an approximation error of less than 3% , for x between 1 and 10,000. In table 2 the approximation $a_2(x)$ is also tabulated.

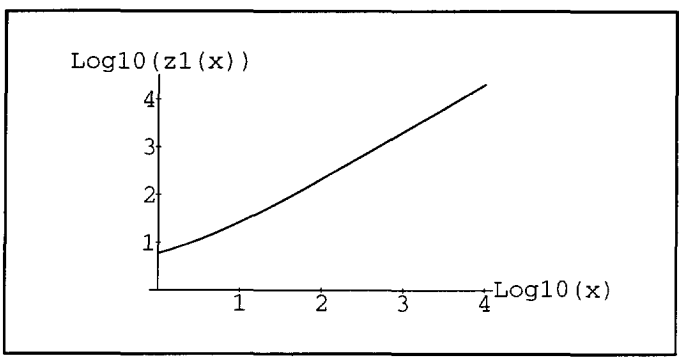


Figure 1: The function $z_1(x)$ in a log-log plot, with logarithm base 10.

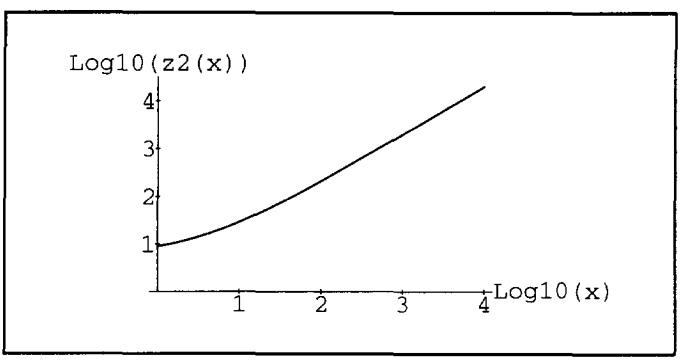


Figure 2: The function $z_2(x)$ in a log-log plot, with logarithm base 10.

Table 1: The function $z_1(x)$ and its approximation $a_1(x)$.

$0 \le x \le 10$			$0 \le x \le 100$			$0 \le x \le 1000$		
x	$z_1(x)$	$a_1(x)$	x	$z_1(x)$	$a_1(x)$	x	$z_1(x)$	$a_1(x)$
			2.5	9.7	10.0	25.	58.	59.
			5.0	15.6	16.1	50.	109.	110.
			7.5	21.2	21.8	75.	160.	161.
1.00	5.85	5.89	10.0	26.6	27.3	100.	211.	212.
1.25	6.54	6.62	12.5	32.0	32.7	125.	261.	262.
1.50	7.21	7.33	15.0	37.3	38.0	150.	311.	313.
1.75	7.86	8.02	17.5	42.5	43.3	175.	362.	363.
2.00	8.50	8.70	20.0	47.8	48.6	200.	412.	413.
2.25	9.13	9.36	22.5	53.0	53.8	225.	462.	463.
2.50	9.75	10.01	25.0	58.2	59.0	250.	512.	514.
2.75	10.36	10.64	27.5	63.3	64.2	275.	563.	564.
3.00	10.96	11.27	30.0	68.5	69.4	300.	613.	614.
3.25	11.56	11.89	32.5	73.6	74.5	325.	663.	664.
3.50	12.15	12.51	35.0	78.8	79.7	350.	713.	714.
3.75	12.74	13.12	37.5	83.9	84.8	375.	763.	764.
4.00	13.32	13.72	40.0	89.0	89.9	400.	813.	814.
4.25	13.90	14.32	42.5	94.1	95.0	425.	864.	865.
4.50	14.48	14.91	45.0	99.2	100.2	450.	914.	915.
4.75	15.05	15.50	47.5	104.3	105.3	475.	964.	965.
5.00	15.62	16.08	50.0	109.4	110.4	500.	1,014.	1,015.
5.25	16.19	16.67	52.5	114.5	115.5	525.	1,064.	1,065.
5.50	16.75	17.24	55.0	119.6	120.6	550.	1,114.	1,115.
5.75	17.32	17.82	57.5	124.7	125.6	575.	1,164.	1,165.
6.00	17.88	18.39	60.0	129.7	130.7	600.	1,214.	1,215.
6.25	18.43	18.96	62.5	134.8	135.8	625.	1,264.	1,265.
6.50	18.99	19.53	65.0	139.9	140.9	650.	1,314.	1,315.
6.75	19.55	20.10	67.5	145.0	146.0	675.	1,364.	1,366.
7.00	20.10	20.66	70.0	150.0	151.0	700.	1,415.	1,416.
7.25	20.65	21.22	72.5	155.1	156.1	725.	1,465.	1,466.
7.50	21.20	21.78	75.0	160.2	161.2	750.	1,515.	1,516.
7.75	21.75	22.34	77.5	165.2	166.2	775.	1,565.	1,566.
8.00	22.30	22.89	80.0	170.3	171.3	800.	1,615.	1,616.
8.25	22.84	23.45	82.5	175.3	176.3	825.	1,665.	1,666.
8.50	23.39	24.00	85.0	180.4	181.4	850.	1,715.	1,716.
8.75	23.93	24.55	87.5	185.5	186.5	875.	1,765.	1,766.
9.00	24.48	25.11	90.0	190.5	191.5	900.	1,815.	1,816.
9.25	25.02	25.65	92.5	195.6	196.6	925.	1,865.	1,866.
9.50	25.56	26.20	95.0	200.6	201.6	950.	1,915.	1,916.
9.75	26.10	26.75	97.5	205.7	206.7	975.	1,965.	1,966.
10.00	26.64	27.30	100.0	210.7	211.7	1,000.	2,015.	2,016.

Table 2: The function $z_2(x)$ and its approximation $a_2(x)$.

$0 \le x \le 10$			$0 \le x \le 100$			$0 \le x \le 1000$		
x	$z_2(x)$	$a_2(x)$	x	$z_2(x)$	$a_2(x)$	x	$z_2(x)$	$a_2(x)$
			2.5	12.5	12.4	25.	60.	59.
			5.0	18.1	17.8	50.	111.	109.
			7.5	23.6	23.0	75.	162.	160.
1.00	9.03	8.99	10.0	28.9	28.2	100.	213.	210.
1.25	9.62	9.57	12.5	34.2	33.3	125.	263.	260.
1.50	10.21	10.14	15.0	39.5	38.4	150.	314.	310.
1.75	10.79	10.71	17.5	44.7	43.5	175.	364.	360.
2.00	11.37	11.27	20.0	49.9	48.6	200.	414.	410.
2.25	11.94	11.83	22.5	55.1	53.7	225.	464.	460.
2.50	12.52	12.38	25.0	60.3	58.8	250.	514.	510.
2.75	13.08	12.93	27.5	65.5	63.8	275.	565.	560.
3.00	13.65	13.47	30.0	70.6	68.9	300.	615.	610.
3.25	14.22	14.01	32.5	75.7	74.0	325.	665.	661.
3.50	14.78	14.55	35.0	80.9	79.0	350.	715.	711.
3.75	15.34	15.09	37.5	86.0	84.1	375.	765.	761.
4.00	15.90	15.63	40.0	91.1	89.1	400.	815.	811.
4.25	16.46	16.16	42.5	96.2	94.1	425.	866.	861.
4.50	17.01	16.69	45.0	101.3	99.2	450.	916.	911.
4.75	17.56	17.22	47.5	106.4	104.2	475.	966.	961.
5.00	18.12	17.75	50.0	111.5	109.3	500.	1,016.	1,011.
5.25	18.67	18.28	52.5	116.6	114.3	525.	1,066.	1,061.
5.50	19.22	18.81	55.0	121.7	119.3	550.	1,116.	1,111.
5.75	19.76	19.34	57.5	126.7	124.3	575.	1,166.	1,161.
6.00	20.31	19.86	60.0	131.8	129.4	600.	1,216.	1,211.
6.25	20.86	20.39	62.5	136.9	134.4	625.	1,266.	1,261.
6.50	21.40	20.91	65.0	142.0	139.4	650.	1,316.	1,311.
6.75	21.94	21.43	67.5	147.0	144.5	675.	1,366.	1,361.
7.00	22.49	21.96	70.0	152.1	149.5	700.	1,417.	1,411.
7.25	23.03	22.48	72.5	157.2	154.5	725.	1,467.	1,461.
7.50	23.57	23.00	75.0	162.2	159.5	750.	1,517.	1,511.
7.75	24.11	23.52	77.5	167.3	164.6	775.	1,567.	1,561.
8.00	24.65	24.04	80.0	172.3	169.6	800.	1,617.	1,611.
8.25	25.19	24.56	82.5	177.4	174.6	825.	1,667.	1,661.
8.50	25.72	25.08	85.0	182.4	179.6	850.	1,717.	1,711.
8.75	26.26	25.59	87.5	187.5	184.6	875.	1,767.	1,761.
9.00	26.80	26.11	90.0	192.6	189.7	900.	1,817.	1,811.
9.25	27.33	26.63	92.5	197.6	194.7	925.	1,867.	1,861.
9.50	27.87	27.15	95.0	202.7	199.7	950.	1,917.	1,911.
9.75	28.40	27.66	97.5	207.7	204.7	975.	1,967.	1,961.
10.00	28.93	28.18	100.0	212.7	209.7	1,000.	2,017.	2,011.

C

Proof of Theorem 2.6

In this appendix the result of theorem 2.5 will be used to bound the maximum deviation of the error frequency to the error probability over a set of classifiers with finite capacity V.

We will prove the following theorem:

Theorem 2.6: *Let $F(X,\Theta)$ be a set of classification functions with finite capacity V and let Ξ^m be an i.i.d. sample of size $m \geq V$, drawn according to some distribution D. When the error frequency on Ξ^m for some $f(x, \theta) \in F(X,\Theta)$ is denoted $\hat{\varepsilon}(\theta)$, and the corresponding probability of error $\varepsilon(\theta)$, then the following inequality holds with probability at least 1 - δ for all $f(x, \theta) \in F(X,\Theta)$ simultaneously:*

$$\varepsilon(\theta) \leq \hat{\varepsilon}(\theta) + \frac{\chi^2}{2}\left(1 + \sqrt{1 + \frac{4\hat{\varepsilon}(\theta)}{\chi^2}}\right) \tag{1}$$

where χ is given by:

$$\chi = \left(\frac{\beta+1}{\beta}\right)\sqrt{\frac{2}{m}\left(\ln\left(S^F_{\max}(m(\beta+1))\right) + \ln\left(\frac{4}{\delta}\right)\right)} \tag{2}$$

and β by:

$$\beta = \frac{1}{m}\left\langle m\, z_1\left(\ln\left(\frac{em}{V}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)\right\rangle \tag{3}$$

For the proof of this theorem we start with theorem 2.5 (cf. inequality A.1) and require that the probability of one-sided maximum relative deviation at most be equal to δ:

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\frac{\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}}>\chi\right\}\leq 4S^{F}_{\max}\left(m(\beta+1)\right)\exp\left(-\frac{1}{2}m\chi^{2}\left(\frac{\beta}{\beta+1}\right)^{2}\right)=\delta \qquad (4)$$

Now, from:

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\frac{\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}}>\chi\right\}\leq\delta \qquad (5)$$

we can solve for $\varepsilon(\boldsymbol{\theta})$ to obtain that

$$\varepsilon(\boldsymbol{\theta})\leq\hat{\varepsilon}(\boldsymbol{\theta})+\frac{\chi^{2}}{2}\left(1+\sqrt{1+\frac{4\hat{\varepsilon}(\boldsymbol{\theta})}{\chi^{2}}}\right) \qquad (6)$$

is valid with probability at least 1 - δ for all classifiers $f(\boldsymbol{x},\ \boldsymbol{\theta})\in F(\boldsymbol{X},\boldsymbol{\Theta})$ simultaneously. From (4) we can also solve χ as:

$$\chi=\sqrt{\frac{2}{m}\left(\frac{\beta+1}{\beta}\right)^{2}\left(\ln\left(S^{F}_{\max}\left(m(\beta+1)\right)\right)+\ln\left(\frac{4}{\delta}\right)\right)} \qquad (7)$$

Substituting the bound on $S^{F}_{\max}\left(m(\beta+1)\right)$ of theorem 2.3 yields:

$$\chi=\sqrt{\frac{2}{m}\left(\frac{\beta+1}{\beta}\right)^{2}\left(\ln\left(\frac{em(\beta+1)}{V}\right)^{V}+\ln\left(\frac{4}{\delta}\right)\right)} \qquad (8)$$

$$=\sqrt{\frac{2V}{m}\left(\frac{\beta+1}{\beta}\right)^{2}\left(\ln(\beta+1)+\ln\left(\frac{em}{V}\right)+\frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)} \qquad (9)$$

$$=\sqrt{b\left(\frac{\beta+1}{\beta}\right)^{2}\left(\ln(\beta+1)+a\right)} \qquad (10)$$

with:

$$b=\frac{2V}{m} \qquad (11)$$

and:

$$a = \ln\left(\frac{em}{V}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right) \tag{12}$$

Note that for $m \geq V$ always $a > 1$. We define:

$$g(\beta) = \left(\frac{\beta+1}{\beta}\right)^2 \left(\ln(\beta+1) + a\right) \tag{13}$$

Clearly, a value of β such that (13) is minimized, also minimizes equations (7 - 10). Differentiation of $g(\beta)$ with respect to β yields:

$$\frac{\partial g(\beta)}{\partial \beta} = \left(\frac{\beta+1}{\beta^3}\right)\left(\beta - 2\left(\ln(\beta+1) + a\right)\right) \tag{14}$$

which has only one root in the admissible domain $a \geq 1$, $\beta > 0$ for $\beta / 2 = \ln(\beta+1) + a$. From (14) it can also be seen that this root corresponds to a minimum of (13). The positive root of $\beta / 2 = \ln(\beta+1) + a$ is given by the function $z_1(x)$ as defined in appendix B. The value of β that minimizes (13) is therefore given by:

$$\beta = z_1(a) = z_1\left(\ln\left(\frac{em}{V}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right) \tag{15}$$

As β necessarily has a value such that $\beta m \in \mathbb{N}$ (cf. equation A.3), its value is slightly adapted such that βm is rounded off to the nearest integer:

$$\beta = \frac{1}{m}\left\langle m\, z_1\left(\ln\left(\frac{em}{V}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)\right\rangle \tag{16}$$

Equality (16), together with (6) and (7), now proves the theorem.

■

A final remark on this proof. The requirement $m > \max\left(2/\chi^2, 4V/\chi^2\right)$ in theorem 2.5 (cf. appendix A) does not restrict the validity of theorem 2.6 as presented here. The first requirement $m > 2/\chi^2$ is always fulfilled, as can be verified by inspection of equation (2). The fact that the second requirement $m > 4V/\chi^2$ is also fulfilled follows

from the observation that the minimum value $a = 1$ gives rise to the (minimum) value $\beta = 5.85$ (see table B.1). Rewriting equation (9) yields:

$$\chi = \sqrt{\frac{2V}{m}\left(\frac{\beta+1}{\beta}\right)^2\left(\ln\left(e(\beta+1)\right)+\ln\left(\frac{m}{V}\right)+\frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)} \tag{17}$$

$$\geq \sqrt{\frac{2V}{m}\left(2.9+\ln\left(\frac{m}{V}\right)+\frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)} > \sqrt{\frac{4V}{m}} \tag{18}$$

This last inequality is always true for $m > V$.

D

A Numerical Evaluation of Theorem 2.6.

In order to provide some idea on the applicability and behavior of the theoretical framework of chapter 2, a numerical evaluation of theorem 2.6 is presented. In a number of tables, the parameter space of the parameters involved is numerically explored.

In this appendix we will evaluate theorem 2.6 (cf. appendix C). The entries in the tables below provide an upper bound for the true error, given the sample size m, the capacity V, the error frequency $\hat{\varepsilon}(\boldsymbol{\theta})$ and the confidence factor δ. Since a thorough numerical evaluation of this four dimensional parameter space in m, V, $\hat{\varepsilon}(\boldsymbol{\theta})$ and δ would be large enough to fill up this thesis by itself, the tables only explore a small subset of this space.

The entries in the tables were computed by using formulas of theorem 2.3, appendix B.1 and theorem 2.6:

$$\varepsilon(\boldsymbol{\theta}) = \hat{\varepsilon}(\boldsymbol{\theta}) + \frac{\chi^2}{2}\left(1 + \sqrt{1 + \frac{4\hat{\varepsilon}(\boldsymbol{\theta})}{\chi^2}}\right) \tag{1}$$

with:

$$\chi = \left(\frac{\beta+1}{\beta}\right)\sqrt{\frac{2}{m}\left(\ln\left(S^F_{\max}\left(m(\beta+1)\right)\right) + \ln\left(\frac{4}{\delta}\right)\right)} \tag{2}$$

$$\beta = \frac{1}{m}\left\langle m\, z_1\left(\ln\left(\frac{em}{V}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)\right\rangle \tag{3}$$

and:

$$z_1(x) = \left\{y > 0 \;\middle|\; \ln(y+1) + x - y/2 = 0;\; x \geq 1\right\} \tag{4}$$

$$S^F_{\max}(m) = \left(\frac{em}{V}\right)^V \tag{5}$$

The 18 tables below are divided over 3 parts: $\delta = 0.1$, 0.01 and 10^{-6}. Every part consists of 6 subparts: $\hat{\varepsilon}(\boldsymbol{\theta}) = 0$, 0.001, 0.01, 0.05, 0.1, and 0.25. Note that all numbers in the table are probabilities. Their format is mantissa±exponent. Empty places in the table correspond to an upper bound larger than or equal to 1: $\varepsilon(\boldsymbol{\theta}) \geq 1$.

A final remark is that, according to the theory developed in appendix A, β has to be chosen such that $\beta m \in \mathbb{N}$. This requirement is also apparent in equation (3). However, if β is not rounded off, but simply chosen as $\beta = z_1\left(\ln(em/V) + (1/V)\ln(4/\delta)\right)$, the data in the subsequent tables will have a negligible difference with the contents of the current tables, being less than 10^{-6}.

Tables 1-3: Guaranteed upper bounds for $\varepsilon(\boldsymbol{\theta})$, when $\delta = 10^{-6}$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0$, $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.001$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.01$.

$\hat{\varepsilon} = 0$ $\delta = 1\text{-}6$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50													
100	.51 +0	.68 +0											
200	.26 +0	.36 +0	.59 +0	.94 +0									
500	.11 +0	.15 +0	.26 +0	.42 +0	.71 +0								
1000	.56 -1	.78 -1	.14 +0	.22 +0	.38 +0	.80 +0							
2000	.29 -1	.40 -1	.72 -1	.12 +0	.21 +0	.44 +0	.79 +0						
5000	.12 -1	.17 -1	.31 -1	.52 -1	.91 -1	.20 +0	.36 +0	.64 +0					
10000	.61 -2	.88 -2	.16 -1	.27 -1	.49 -1	.11 +0	.19 +0	.35 +0	.77 +0				
20000	.31 -2	.45 -2	.84 -2	.14 -1	.26 -1	.57 -1	.10 +0	.19 +0	.43 +0	.77 +0			
50000	.13 -2	.19 -2	.36 -2	.62 -2	.11 -1	.25 -1	.46 -1	.85 -1	.19 +0	.35 +0	.64 +0		
10+5	.66 -3	.97 -3	.19 -2	.32 -2	.59 -2	.13 -1	.24 -1	.46 -1	.10 +0	.19 +0	.35 +0	.77 +0	
10+6	.71 -4	.11 -3	.21 -3	.37 -3	.68 -3	.16 -2	.29 -2	.56 -2	.13 -1	.24 -1	.45 -1	.10 +0	.19 +0
10+7	.75 -5	.12 -4	.23 -4	.42 -4	.78 -4	.18 -3	.34 -3	.65 -3	.15 -2	.29 -2	.55 -2	.13 -1	.24 -1

$\hat{\varepsilon} = 1\text{-}3$ $\delta = 1\text{-}6$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50													
100	.52 +0	.68 +0											
200	.27 +0	.36 +0	.60 +0	.94 +0									
500	.11 +0	.15 +0	.26 +0	.42 +0	.71 +0								
1000	.58 -1	.80 -1	.14 +0	.23 +0	.39 +0	.81 +0							
2000	.31 -1	.42 -1	.74 -1	.12 +0	.21 +0	.44 +0	.79 +0						
5000	.14 -1	.19 -1	.33 -1	.54 -1	.93 -1	.20 +0	.36 +0	.65 +0					
10000	.80 -2	.11 -1	.18 -1	.29 -1	.51 -1	.11 +0	.20 +0	.36 +0	.77 +0				
20000	.49 -2	.64 -2	.10 -1	.16 -1	.28 -1	.59 -1	.11 +0	.19 +0	.43 +0	.77 +0			
50000	.29 -2	.36 -2	.54 -2	.81 -2	.13 -1	.27 -1	.48 -1	.87 -1	.19 +0	.35 +0	.64 +0		
10+5	.22 -2	.26 -2	.36 -2	.50 -2	.77 -2	.15 -1	.26 -1	.48 -1	.11 +0	.19 +0	.35 +0	.77 +0	
10+6	.13 -2	.14 -2	.16 -2	.18 -2	.22 -2	.33 -2	.47 -2	.74 -2	.15 -1	.26 -1	.47 -1	.11 +0	.19 +0
10+7	.11 -2	.11 -2	.12 -2	.12 -2	.13 -2	.15 -2	.18 -2	.22 -2	.32 -2	.47 -2	.74 -2	.15 -1	.26 -1

$\hat{\varepsilon} = 1\text{-}2$ $\delta = 1\text{-}6$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50													
100	.53 +0	.70 +0											
200	.28 +0	.37 +0	.61 +0	.96 +0									
500	.13 +0	.17 +0	.28 +0	.44 +0	.73 +0								
1000	.75 -1	.97 -1	.16 +0	.24 +0	.40 +0	.82 +0							
2000	.47 -1	.59 -1	.91 -1	.14 +0	.23 +0	.46 +0	.81 +0						
5000	.28 -1	.34 -1	.49 -1	.70 -1	.11 +0	.22 +0	.38 +0	.66 +0					
10000	.21 -1	.25 -1	.33 -1	.45 -1	.67 -1	.13 +0	.21 +0	.37 +0	.79 +0				
20000	.17 -1	.19 -1	.24 -1	.31 -1	.43 -1	.76 -1	.12 +0	.21 +0	.45 +0	.79 +0			
50000	.14 -1	.15 -1	.18 -1	.22 -1	.27 -1	.42 -1	.64 -1	.10 +0	.21 +0	.37 +0	.66 +0		
10+5	.13 -1	.14 -1	.15 -1	.18 -1	.21 -1	.30 -1	.42 -1	.64 -1	.12 +0	.21 +0	.37 +0	.79 +0	
10+6	.11 -1	.11 -1	.12 -1	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.29 -1	.42 -1	.64 -1	.12 +0	.21 +0
10+7	.10 -1	.10 -1	.10 -1	.11 -1	.11 -1	.11 -1	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.29 -1	.42 -1

Tables 4-6: Guaranteed upper bounds for $\varepsilon(\boldsymbol{\theta})$, when $\delta = 10^{-6}$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.05$, $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.1$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.25$.

$\hat{\varepsilon}$=5-2 δ= 1-6	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50													
100	.61 +0	.78 +0											
200	.36 +0	.45 +0	.69 +0										
500	.20 +0	.24 +0	.35 +0	.51 +0	.80 +0								
1000	.14 +0	.16 +0	.23 +0	.32 +0	.48 +0	.90 +0							
2000	.11 +0	.12 +0	.16 +0	.21 +0	.30 +0	.54 +0	.88 +0						
5000	.81 -1	.89 -1	.11 +0	.13 +0	.18 +0	.29 +0	.45 +0	.74 +0					
10000	.71 -1	.76 -1	.88 -1	.10 +0	.13 +0	.19 +0	.29 +0	.45 +0	.87 +0				
20000	.64 -1	.67 -1	.75 -1	.85 -1	.10 +0	.14 +0	.19 +0	.28 +0	.52 +0	.87 +0			
50000	.59 -1	.61 -1	.65 -1	.71 -1	.80 -1	.10 +0	.13 +0	.17 +0	.28 +0	.44 +0	.73 +0		
10+5	.56 -1	.57 -1	.61 -1	.64 -1	.70 -1	.83 -1	.99 -1	.13 +0	.19 +0	.28 +0	.44 +0	.87 +0	
10+6	.52 -1	.52 -1	.53 -1	.55 -1	.56 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1	.13 +0	.19 +0	.28 +0
10+7	.51 -1	.51 -1	.51 -1	.51 -1	.52 -1	.53 -1	.54 -1	.56 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1

$\hat{\varepsilon}$=1-1 δ= 1-6	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50													
100	.70 +0	.87 +0											
200	.44 +0	.54 +0	.78 +0										
500	.27 +0	.32 +0	.43 +0	.60 +0	.89 +0								
1000	.21 +0	.24 +0	.30 +0	.40 +0	.57 +0	.99 +0							
2000	.17 +0	.19 +0	.23 +0	.28 +0	.38 +0	.63 +0	.98 +0						
5000	.14 +0	.15 +0	.17 +0	.20 +0	.25 +0	.37 +0	.54 +0	.83 +0					
10000	.13 +0	.13 +0	.15 +0	.17 +0	.20 +0	.27 +0	.37 +0	.53 +0	.96 +0				
20000	.12 +0	.12 +0	.13 +0	.15 +0	.17 +0	.21 +0	.27 +0	.36 +0	.61 +0	.96 +0			
50000	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.24 +0	.36 +0	.53 +0	.82 +0		
10+5	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.27 +0	.36 +0	.53 +0	.96 +0	
10+6	.10 +0	.10 +0	.10 +0	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.27 +0	.36 +0
10+7	.10 +0	.10 +0	.10 +0	.10 +0	.10 +0	.10 +0	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0

$\hat{\varepsilon}$25-2 δ= 1-6	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50													
100	.95 +0												
200	.67 +0	.77 +0											
500	.48 +0	.53 +0	.66 +0	.84 +0									
1000	.40 +0	.43 +0	.51 +0	.62 +0	.81 +0								
2000	.35 +0	.37 +0	.42 +0	.49 +0	.60 +0	.87 +0							
5000	.31 +0	.32 +0	.35 +0	.39 +0	.45 +0	.59 +0	.78 +0						
10000	.29 +0	.30 +0	.32 +0	.35 +0	.39 +0	.47 +0	.59 +0	.77 +0					
20000	.28 +0	.29 +0	.30 +0	.32 +0	.34 +0	.40 +0	.47 +0	.59 +0	.85 +0				
50000	.27 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.44 +0	.58 +0	.77 +0			
10+5	.26 +0	.27 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.47 +0	.58 +0	.77 +0		
10+6	.25 +0	.26 +0	.26 +0	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.47 +0	.58 +0
10+7	.25 +0	.25 +0	.25 +0	.25 +0	.25 +0	.26 +0	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0

Tables 7-9: Guaranteed upper bounds for $\varepsilon(\boldsymbol{\theta})$, when $\delta = 0.01$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0$, $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.001$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.01$.

$\hat{\varepsilon} = 0$ $\delta = 1\text{-}2$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.61 +0	.90 +0											
100	.32 +0	.48 +0	.91 +0										
200	.17 +0	.26 +0	.49 +0	.84 +0									
500	.71 -1	.11 +0	.22 +0	.38 +0	.66 +0								
1000	.37 -1	.58 -1	.12 +0	.20 +0	.36 +0	.78 +0							
2000	.19 -1	.31 -1	.62 -1	.11 +0	.20 +0	.43 +0	.78 +0						
5000	.81 -2	.13 -1	.27 -1	.48 -1	.87 -1	.19 +0	.35 +0	.64 +0					
10000	.42 -2	.68 -2	.14 -1	.25 -1	.47 -1	.10 +0	.19 +0	.35 +0	.77 +0				
20000	.22 -2	.36 -2	.74 -2	.13 -1	.25 -1	.56 -1	.10 +0	.19 +0	.43 +0	.77 +0			
50000	.90 -3	.15 -2	.32 -2	.58 -2	.11 -1	.24 -1	.46 -1	.85 -1	.19 +0	.35 +0	.64 +0		
10+5	.47 -3	.78 -3	.17 -2	.30 -2	.57 -2	.13 -1	.24 -1	.45 -1	.10 +0	.19 +0	.35 +0	.77 +0	
10+6	.51 -4	.88 -4	.19 -3	.35 -3	.66 -3	.15 -2	.29 -2	.55 -2	.13 -1	.24 -1	.45 -1	.10 +0	.19 +0
10+7	.56 -5	.97 -5	.21 -4	.40 -4	.76 -4	.18 -3	.34 -3	.65 -3	.15 -2	.29 -2	.55 -2	.13 -1	.24 -1

$\hat{\varepsilon} = 1\text{-}3$ $\delta = 1\text{-}2$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.61 +0	.91 +0											
100	.32 +0	.48 +0	.91 +0										
200	.17 +0	.26 +0	.49 +0	.84 +0									
500	.73 -1	.11 +0	.22 +0	.38 +0	.67 +0								
1000	.39 -1	.60 -1	.12 +0	.21 +0	.37 +0	.79 +0							
2000	.21 -1	.33 -1	.64 -1	.11 +0	.20 +0	.43 +0	.78 +0						
5000	.10 -1	.15 -1	.29 -1	.50 -1	.89 -1	.20 +0	.35 +0	.64 +0					
10000	.60 -2	.87 -2	.16 -1	.27 -1	.49 -1	.11 +0	.19 +0	.35 +0	.77 +0				
20000	.39 -2	.54 -2	.93 -2	.15 -1	.27 -1	.58 -1	.11 +0	.19 +0	.43 +0	.77 +0			
50000	.25 -2	.32 -2	.50 -2	.77 -2	.13 -1	.26 -1	.47 -1	.87 -1	.19 +0	.35 +0	.64 +0		
10+5	.20 -2	.24 -2	.34 -2	.48 -2	.75 -2	.15 -1	.26 -1	.47 -1	.11 +0	.19 +0	.35 +0	.77 +0	
10+6	.13 -2	.13 -2	.15 -2	.18 -2	.22 -2	.32 -2	.47 -2	.74 -2	.15 -1	.26 -1	.47 -1	.10 +0	.19 +0
10+7	.11 -2	.11 -2	.12 -2	.12 -2	.13 -2	.15 -2	.18 -2	.22 -2	.32 -2	.47 -2	.74 -2	.15 -1	.26 -1

$\hat{\varepsilon} = 1\text{-}2$ $\delta = 1\text{-}2$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.63 +0	.92 +0											
100	.34 +0	.50 +0	.93 +0										
200	.19 +0	.28 +0	.51 +0	.86 +0									
500	.90 -1	.13 +0	.24 +0	.40 +0	.68 +0								
1000	.55 -1	.77 -1	.14 +0	.22 +0	.38 +0	.80 +0							
2000	.36 -1	.49 -1	.81 -1	.13 +0	.22 +0	.45 +0	.80 +0						
5000	.24 -1	.30 -1	.44 -1	.66 -1	.11 +0	.21 +0	.37 +0	.66 +0					
10000	.19 -1	.22 -1	.31 -1	.43 -1	.65 -1	.12 +0	.21 +0	.37 +0	.79 +0				
20000	.16 -1	.18 -1	.23 -1	.30 -1	.42 -1	.75 -1	.12 +0	.21 +0	.44 +0	.79 +0			
50000	.13 -1	.15 -1	.17 -1	.21 -1	.27 -1	.42 -1	.64 -1	.10 +0	.21 +0	.37 +0	.66 +0		
10+5	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.30 -1	.42 -1	.64 -1	.12 +0	.21 +0	.37 +0	.79 +0	
10+6	.11 -1	.11 -1	.11 -1	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.29 -1	.42 -1	.64 -1	.12 +0	.21 +0
10+7	.10 -1	.10 -1	.10 -1	.11 -1	.11 -1	.11 -1	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.29 -1	.42 -1

Tables 10-12: Guaranteed upper bounds for $\varepsilon(\boldsymbol{\theta})$, when $\delta = 0.01$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.05$, $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.1$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.25$.

$\hat{\varepsilon}$=5-2 δ= 1-2	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.71 +0												
100	.41 +0	.58 +0											
200	.26 +0	.35 +0	.59 +0	.94 +0									
500	.15 +0	.20 +0	.31 +0	.47 +0	.76 +0								
1000	.12 +0	.14 +0	.20 +0	.30 +0	.46 +0	.88 +0							
2000	.92 -1	.11 +0	.14 +0	.20 +0	.29 +0	.53 +0	.87 +0						
5000	.75 -1	.83 -1	.10 +0	.13 +0	.17 +0	.28 +0	.45 +0	.74 +0					
10000	.67 -1	.72 -1	.85 -1	.10 +0	.13 +0	.19 +0	.28 +0	.45 +0	.87 +0				
20000	.62 -1	.65 -1	.73 -1	.84 -1	.10 +0	.14 +0	.19 +0	.28 +0	.52 +0	.87 +0			
50000	.57 -1	.59 -1	.64 -1	.70 -1	.79 -1	.99 -1	.13 +0	.17 +0	.28 +0	.44 +0	.73 +0		
10+5	.55 -1	.57 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1	.13 +0	.19 +0	.28 +0	.44 +0	.87 +0	
10+6	.52 -1	.52 -1	.53 -1	.54 -1	.56 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1	.13 +0	.19 +0	.28 +0
10+7	.51 -1	.51 -1	.51 -1	.51 -1	.52 -1	.53 -1	.54 -1	.56 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1

$\hat{\varepsilon}$=1-1 δ= 1-2	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.80 +0												
100	.50 +0	.67 +0											
200	.34 +0	.43 +0	.68 +0										
500	.23 +0	.27 +0	.39 +0	.56 +0	.85 +0								
1000	.18 +0	.21 +0	.28 +0	.38 +0	.54 +0	.97 +0							
2000	.15 +0	.17 +0	.22 +0	.27 +0	.37 +0	.62 +0	.97 +0						
5000	.13 +0	.14 +0	.17 +0	.20 +0	.25 +0	.37 +0	.53 +0	.83 +0					
10000	.12 +0	.13 +0	.15 +0	.16 +0	.20 +0	.27 +0	.36 +0	.53 +0	.96 +0				
20000	.12 +0	.12 +0	.13 +0	.14 +0	.16 +0	.21 +0	.27 +0	.36 +0	.61 +0	.96 +0			
50000	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.24 +0	.36 +0	.53 +0	.82 +0		
10+5	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.27 +0	.36 +0	.53 +0	.96 +0	
10+6	.10 +0	.10 +0	.10 +0	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.27 +0	.36 +0
10+7	.10 +0	.10 +0	.10 +0	.10 +0	.10 +0	.10 +0	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0

$\hat{\varepsilon}$25-2 δ= 1-2	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50													
100	.74 +0	.91 +0											
200	.56 +0	.66 +0	.92 +0										
500	.42 +0	.48 +0	.62 +0	.80 +0									
1000	.37 +0	.40 +0	.49 +0	.60 +0	.78 +0								
2000	.33 +0	.35 +0	.41 +0	.48 +0	.59 +0	.86 +0							
5000	.30 +0	.31 +0	.35 +0	.39 +0	.45 +0	.59 +0	.77 +0						
10000	.28 +0	.29 +0	.32 +0	.34 +0	.38 +0	.47 +0	.58 +0	.77 +0					
20000	.27 +0	.28 +0	.30 +0	.32 +0	.34 +0	.40 +0	.47 +0	.58 +0	.85 +0				
50000	.27 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.44 +0	.58 +0	.77 +0			
10+5	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.47 +0	.58 +0	.77 +0		
10+6	.25 +0	.25 +0	.26 +0	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.47 +0	.58 +0
10+7	.25 +0	.25 +0	.25 +0	.25 +0	.25 +0	.26 +0	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0

Tables 13-15: Guaranteed upper bounds for $\varepsilon(\boldsymbol{\theta})$, when $\delta = 0.1$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0$, $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.001$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.01$.

$\hat{\varepsilon} = 0$ $\delta = 1\text{-}1$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.51 +0	.80 +0											
100	.27 +0	.43 +0	.85 +0										
200	.14 +0	.23 +0	.47 +0	.81 +0									
500	.61 -1	.10 +0	.21 +0	.37 +0	.65 +0								
1000	.32 -1	.53 -1	.11 +0	.20 +0	.36 +0	.78 +0							
2000	.17 -1	.28 -1	.59 -1	.11 +0	.19 +0	.43 +0	.77 +0						
5000	.71 -2	.12 -1	.26 -1	.47 -1	.86 -1	.19 +0	.35 +0	.64 +0					
10000	.37 -2	.63 -2	.14 -1	.25 -1	.46 -1	.10 +0	.19 +0	.35 +0	.77 +0				
20000	.19 -2	.33 -2	.72 -2	.13 -1	.25 -1	.56 -1	.10 +0	.19 +0	.42 +0	.77 +0			
50000	.81 -3	.14 -2	.31 -2	.57 -2	.11 -1	.24 -1	.45 -1	.85 -1	.19 +0	.35 +0	.64 +0		
10+5	.42 -3	.73 -3	.16 -2	.30 -2	.56 -2	.13 -1	.24 -1	.45 -1	.10 +0	.19 +0	.35 +0	.77 +0	
10+6	.47 -4	.83 -4	.19 -3	.35 -3	.66 -3	.15 -2	.29 -2	.55 -2	.13 -1	.24 -1	.45 -1	.10 +0	.19 +0
10+7	.51 -5	.92 -5	.21 -4	.40 -4	.76 -4	.18 -3	.34 -3	.65 -3	.15 -2	.29 -2	.55 -2	.13 -1	.24 -1

$\hat{\varepsilon} = 1\text{-}3$ $\delta = 1\text{-}1$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.52 +0	.80 +0											
100	.27 +0	.43 +0	.86 +0										
200	.15 +0	.23 +0	.47 +0	.81 +0									
500	.63 -1	.10 +0	.21 +0	.37 +0	.66 +0								
1000	.34 -1	.55 -1	.11 +0	.20 +0	.36 +0	.78 +0							
2000	.19 -1	.30 -1	.61 -1	.11 +0	.20 +0	.43 +0	.78 +0						
5000	.90 -2	.14 -1	.28 -1	.49 -1	.88 -1	.19 +0	.35 +0	.64 +0					
10000	.55 -2	.82 -2	.16 -1	.27 -1	.48 -1	.11 +0	.19 +0	.35 +0	.77 +0				
20000	.36 -2	.51 -2	.91 -2	.15 -1	.27 -1	.58 -1	.11 +0	.19 +0	.43 +0	.77 +0			
50000	.24 -2	.31 -2	.49 -2	.76 -2	.13 -1	.26 -1	.47 -1	.87 -1	.19 +0	.35 +0	.64 +0		
10+5	.19 -2	.23 -2	.33 -2	.48 -2	.75 -2	.15 -1	.26 -1	.47 -1	.11 +0	.19 +0	.35 +0	.77 +0	
10+6	.12 -2	.13 -2	.15 -2	.18 -2	.22 -2	.32 -2	.47 -2	.74 -2	.15 -1	.26 -1	.47 -1	.10 +0	.19 +0
10+7	.11 -2	.11 -2	.12 -2	.12 -2	.13 -2	.15 -2	.18 -2	.22 -2	.32 -2	.47 -2	.74 -2	.15 -1	.26 -1

$\hat{\varepsilon} = 1\text{-}2$ $\delta = 1\text{-}1$	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.53 +0	.82 +0											
100	.29 +0	.45 +0	.87 +0										
200	.16 +0	.25 +0	.49 +0	.83 +0									
500	.80 -1	.12 +0	.23 +0	.39 +0	.67 +0								
1000	.50 -1	.72 -1	.13 +0	.22 +0	.38 +0	.80 +0							
2000	.34 -1	.46 -1	.78 -1	.13 +0	.21 +0	.45 +0	.79 +0						
5000	.23 -1	.29 -1	.43 -1	.65 -1	.11 +0	.21 +0	.37 +0	.66 +0					
10000	.18 -1	.22 -1	.30 -1	.43 -1	.65 -1	.12 +0	.21 +0	.37 +0	.79 +0				
20000	.15 -1	.18 -1	.23 -1	.30 -1	.42 -1	.74 -1	.12 +0	.21 +0	.44 +0	.79 +0			
50000	.13 -1	.15 -1	.17 -1	.21 -1	.27 -1	.42 -1	.64 -1	.10 +0	.21 +0	.37 +0	.66 +0		
10+5	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.30 -1	.42 -1	.64 -1	.12 +0	.21 +0	.37 +0	.79 +0	
10+6	.11 -1	.11 -1	.11 -1	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.29 -1	.42 -1	.64 -1	.12 +0	.21 +0
10+7	.10 -1	.10 -1	.10 -1	.11 -1	.11 -1	.11 -1	.12 -1	.13 -1	.15 -1	.17 -1	.21 -1	.29 -1	.42 -1

Tables 16-18: Guaranteed upper bounds for $\varepsilon(\boldsymbol{\theta})$, when $\delta = 0.1$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.05$, $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.1$ and $\hat{\varepsilon}(\boldsymbol{\theta}) = 0.25$.

$\hat{\varepsilon}$=5-2 δ= 1-1	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.61 +0	.90 +0											
100	.36 +0	.53 +0	.95 +0										
200	.23 +0	.32 +0	.56 +0	.91 +0									
500	.14 +0	.19 +0	.30 +0	.46 +0	.75 +0								
1000	.11 +0	.13 +0	.20 +0	.29 +0	.45 +0	.88 +0							
2000	.89 -1	.10 +0	.14 +0	.19 +0	.29 +0	.52 +0	.87 +0						
5000	.73 -1	.81 -1	.10 +0	.13 +0	.17 +0	.28 +0	.45 +0	.73 +0					
10000	.66 -1	.71 -1	.84 -1	.10 +0	.13 +0	.19 +0	.28 +0	.44 +0	.87 +0				
20000	.61 -1	.65 -1	.73 -1	.83 -1	.99 -1	.14 +0	.19 +0	.28 +0	.52 +0	.87 +0			
50000	.57 -1	.59 -1	.64 -1	.70 -1	.79 -1	.99 -1	.13 +0	.17 +0	.28 +0	.44 +0	.73 +0		
10+5	.55 -1	.56 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1	.13 +0	.19 +0	.28 +0	.44 +0	.87 +0	
10+6	.52 -1	.52 -1	.53 -1	.54 -1	.56 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1	.13 +0	.19 +0	.28 +0
10+7	.51 -1	.51 -1	.51 -1	.51 -1	.52 -1	.53 -1	.54 -1	.56 -1	.60 -1	.64 -1	.70 -1	.83 -1	.99 -1

$\hat{\varepsilon}$=1-1 δ= 1-1	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.70 +0	.99 +0											
100	.45 +0	.62 +0											
200	.31 +0	.41 +0	.65 +0										
500	.21 +0	.26 +0	.38 +0	.55 +0	.84 +0								
1000	.17 +0	.20 +0	.27 +0	.37 +0	.54 +0	.97 +0							
2000	.15 +0	.17 +0	.21 +0	.27 +0	.37 +0	.61 +0	.96 +0						
5000	.13 +0	.14 +0	.17 +0	.20 +0	.25 +0	.36 +0	.53 +0	.83 +0					
10000	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.27 +0	.36 +0	.53 +0	.96 +0				
20000	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.21 +0	.27 +0	.36 +0	.61 +0	.96 +0			
50000	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.24 +0	.36 +0	.53 +0	.82 +0		
10+5	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.27 +0	.36 +0	.53 +0	.96 +0	
10+6	.10 +0	.10 +0	.10 +0	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0	.19 +0	.27 +0	.36 +0
10+7	.10 +0	.10 +0	.10 +0	.10 +0	.10 +0	.10 +0	.11 +0	.11 +0	.11 +0	.12 +0	.13 +0	.14 +0	.16 +0

$\hat{\varepsilon}$25-2 δ= 1-1	V=1	2	5	10	20	50	100	200	500	1000	2000	5000	10000
m=1													
2													
5													
10													
20													
50	.95 +0												
100	.68 +0	.86 +0											
200	.52 +0	.63 +0	.90 +0										
500	.41 +0	.47 +0	.60 +0	.79 +0									
1000	.36 +0	.40 +0	.48 +0	.59 +0	.78 +0								
2000	.32 +0	.35 +0	.41 +0	.48 +0	.59 +0	.86 +0							
5000	.30 +0	.31 +0	.34 +0	.38 +0	.45 +0	.59 +0	.77 +0						
10000	.28 +0	.29 +0	.32 +0	.34 +0	.38 +0	.47 +0	.58 +0	.77 +0					
20000	.27 +0	.28 +0	.30 +0	.31 +0	.34 +0	.40 +0	.47 +0	.58 +0	.85 +0				
50000	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.44 +0	.58 +0	.77 +0			
10+5	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.47 +0	.58 +0	.77 +0		
10+6	.25 +0	.25 +0	.26 +0	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0	.38 +0	.47 +0	.58 +0
10+7	.25 +0	.25 +0	.25 +0	.25 +0	.25 +0	.26 +0	.26 +0	.26 +0	.27 +0	.28 +0	.29 +0	.31 +0	.34 +0

E

Proof of Theorem 2.7

In this appendix the result of theorem 2.5 will be used to determine a lower bound on the sample size m, such that the probability of maximum one-sided relative deviation is less than a certain value δ. A special case of this bound is relevant in the context of concept learning, and improves a result of Blumer [Blumer 1986], [Blumer 1989].

We will prove the following theorem:

Theorem 2.7: *Let $F(X,\Theta)$ be a set of classification functions with finite capacity V and let Ξ^m be an i.i.d. sample of size m, drawn according to some distribution D. In order to let the probability, that the maximum one-sided relative deviation of the true error $\varepsilon(\theta)$ to the error frequency $\hat{\varepsilon}(\theta)$ on Ξ^m is larger than χ, be at most δ:*

$$P\left\{\sup_{\theta\in\Theta}\frac{\varepsilon(\theta)-\hat{\varepsilon}(\theta)}{\sqrt{\varepsilon(\theta)}}>\chi\right\}\leq\delta \tag{1}$$

the sample size m should at least be:

$$m\geq\frac{1}{\beta}\left\lceil\frac{3}{\chi^2}\frac{(\beta+1)^2}{\beta}\left(V\ln\left(\frac{2e}{\chi^2}\frac{(\beta+1)^3}{\beta^2}\right)+\ln\left(\frac{4}{\delta}\right)\right)\right\rceil \tag{2}$$

where β is given by:

$$\beta=z_2\left(\ln\left(\frac{2e}{\chi^2}\right)+\frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right) \tag{3}$$

For the proof of this theorem we return to theorem 2.5 (cf. inequality A.1) and demand that the probability of maximum one-sided relative deviation over $F(\boldsymbol{X},\boldsymbol{\Theta})$ is at most δ:

$$P\left\{\sup_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\frac{\varepsilon(\boldsymbol{\theta})-\hat{\varepsilon}(\boldsymbol{\theta})}{\sqrt{\varepsilon(\boldsymbol{\theta})}}>\chi\right\}\le 4S^F_{\max}\left(m(\beta+1)\right)\exp\left(-\frac{1}{2}m\chi^2\left(\frac{\beta}{\beta+1}\right)^2\right)\le\delta \qquad (4)$$

From (4) it follows by taking logarithms:

$$m\ge\frac{2}{\chi^2}\left(\frac{\beta+1}{\beta}\right)^2\ln\left(\frac{4}{\delta}S^F_{\max}\left(m(\beta+1)\right)\right) \qquad (5)$$

Substituting the bound on $S^F_{\max}\left(m(\beta+1)\right)$ of theorem 2.3 yields:

$$m\ge\frac{2}{\chi^2}\left(\frac{\beta+1}{\beta}\right)^2\ln\left(\frac{4}{\delta}\left(\frac{em(\beta+1)}{V}\right)^V\right) \qquad (6)$$

$$=\frac{2V}{\chi^2}\left(\frac{\beta+1}{\beta}\right)^2\ln\left(\left(\frac{4}{\delta}\right)^{\frac{1}{V}}\frac{em(\beta+1)}{V}\right) \qquad (7)$$

This can be expressed in the more compact form:

$$m\ge a\ln(bm) \qquad (8)$$

with:

$$a=\frac{2V}{\chi^2}\left(\frac{\beta+1}{\beta}\right)^2 \qquad (9)$$

and:

$$b=\left(\frac{4}{\delta}\right)^{\frac{1}{V}}\frac{e(\beta+1)}{V} \qquad (10)$$

Before we proceed, we illuminate the following relation between a and b:

$$ab = \frac{2e}{\chi^2}\frac{(\beta+1)^3}{\beta^2}\left(\frac{4}{\delta}\right)^{\frac{1}{V}} \tag{11}$$

Setting the derivative of ab in (11) with respect to β to zero, yields that (11) is minimum for $\beta = 2$. Since $\chi < 1$, and $(4/\delta)^{1/V} > 1$ it then follows:

$$ab \geq 2e\frac{27}{4} \approx 36.6968 \tag{12}$$

Returning to (8), we hypothesize that (8) is fulfilled when:

$$m \geq \frac{3}{2}a\ln(ab) \tag{13}$$

Substituting (13) in (8) yields:

$$(ab)^{\frac{3}{2}} \geq \frac{3}{2}ab\ln(ab) \tag{14}$$

If (14) is true for some value of ab, it is also true for larger values, since the left side of (14) increases faster than the right side. It is easily verified that (14) is fulfilled for the lower bound on ab derived in (12). Substituting (9) and (10) in (13) then yields:

$$m \geq \frac{3}{\chi^2}\left(\frac{\beta+1}{\beta}\right)^2\left(V\ln\left(\frac{2e}{\chi^2}\frac{(\beta+1)^3}{\beta^2}\right)+\ln\left(\frac{4}{\delta}\right)\right) \tag{15}$$

The parameter β is a free parameter, with the restriction that it should be chosen such that $\beta m \in \mathbb{N}$:

$$m \geq \frac{1}{\beta}\left\lceil\frac{3}{\chi^2}\frac{(\beta+1)^2}{\beta}\left(V\ln\left(\frac{2e}{\chi^2}\frac{(\beta+1)^3}{\beta^2}\right)+\ln\left(\frac{4}{\delta}\right)\right)\right\rceil \tag{16}$$

As in appendices A and C, we are now ready to derive an approximately optimal value for β in (16). We write (15) as:

$$m \geq \frac{3V}{\chi^2}\left(\frac{\beta+1}{\beta}\right)^2\left(\ln\left(\frac{(\beta+1)^3}{\beta^2}\right)+\ln\left(\frac{2e}{\chi^2}\right)+\frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right) \tag{17}$$

which can be expressed as:

$$m \geq p\left(\frac{\beta+1}{\beta}\right)^2\left(\ln\left(\frac{(\beta+1)^3}{\beta^2}\right)+q\right) \tag{18}$$

with:

$$p=\frac{3V}{\chi^2} \tag{19}$$

and:

$$q=\ln\left(\frac{2e}{\chi^2}\right)+\frac{1}{V}\ln\left(\frac{4}{\delta}\right) \tag{20}$$

Note that $q > 1$. We define:

$$g(\beta)=\left(\frac{\beta+1}{\beta}\right)^2\left(\ln\left(\frac{(\beta+1)^3}{\beta^2}\right)+q\right) \tag{21}$$

By differentiation of $g(\beta)$ with respect to β it follows after some algebra that:

$$\frac{\partial g(\beta)}{\partial \beta}=\left(\frac{\beta+1}{\beta^3}\right)\left(\beta-2\left(\ln\left(\frac{(\beta+1)^3}{\beta^2}\right)+q\right)-2\right) \tag{22}$$

From (22) it is seen that the only root in the admissible domain $\beta > 0$, $q \geq 1$ is for:

$$\frac{\beta-2}{2}=\ln\left(\frac{(\beta+1)^3}{\beta^2}\right)+q \tag{23}$$

From (21) it is clear that this root corresponds to a minimum of (20). Furthermore, the root in β, as a function of q, corresponds to the function $z_2(x)$ as defined in appendix B. The value of β that minimizes (21) is therefore given by:

$$\beta = z_2(q) = z_2\left(\ln\left(\frac{2e}{\chi^2}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right) \tag{24}$$

This concludes the proof of theorem 2.7.

■

As in appendix C, the requirement in theorem 2.5 (cf. appendix A) does not restrict the validity of theorem 2.7. From (2) it is immediate that $m > 2/\chi^2$ is always fulfilled. The fulfillment of the second requirement, $m > 4V/\chi^2$, follows from substitution of the minimum value of β in equation (2). The minimum of β is attained for $q = 1$: $\beta \geq z_2(1) = 9.03$. This implies for (2):

$$m \geq \frac{1}{\beta}\left\lceil \frac{3V}{\chi^2}\frac{(\beta+1)^2}{\beta}\left(\ln\left(2e\frac{(\beta+1)^3}{\beta^2}\right) + \ln\left(\frac{1}{\chi^2}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right)\right\rceil \tag{25}$$

$$\geq \frac{3V}{\chi^2}\left(4.20 + \ln\left(\frac{1}{\chi^2}\right) + \frac{1}{V}\ln\left(\frac{4}{\delta}\right)\right) > \frac{4V}{\chi^2} \tag{26}$$

F

Pseudo-Code of the Experiments of Section 4.3.4

In this appendix the experimental procedure of section 4.3.4 is described in a pseudo-code version.

The procedure of section 4.3.4 accepts a dataset, and aims at finding the largest subset of the data that can be arbitrarily labeled by the backpropagation algorithm.

```
capacity = 1
capacity_done = FALSE

while (capacity_done == FALSE) {

  repetition = 0
  repetition_done = FALSE

  while (repetition_done == FALSE) {

    take a random subset of the dataset of size "capacity"

    dichotomy = 0
    dichotomy_done = FALSE

    while (dichotomy_done == FALSE) {

      label the subset according to the current dichotomy

      load the network from disk

      update = 0
      update_done = FALSE

      while (update_done == FALSE) {

        perform one learning cycle with the backpropagation algorithm
        (for each element of the subset perform a parameter update)

        compute the performance on the subset

        update = update + capacity

        if (the subset is classified correctly) update_done = TRUE
        if (update > 100,000)                   update_done = TRUE
      }

      dichotomy = dichotomy + 1

      if (the dichotomy could not be induced) dichotomy_done = TRUE
      if (dichotomy == 2^capacity)            dichotomy_done = TRUE
    }

    repetition = repetition + 1

    if (dichotomy == 2^capacity) repetition_done = TRUE

    if (repetition == 1,000) {

      print ("The Estimated Capacity = "capacity - 1")
      exit
    }

    capacity = capacity + 1

    if (capacity > 31) capacity_done = TRUE
  }
}
```

Literature

Anderson 1987

Anderson, J.A., and Rosenfeld, E., (Eds.), *Neurocomputing: A collection of classic papers*. MIT press 1987.

Bakker 1993

Bakker, R.R.N., Kraaijveld, M.A., Duin, R.P.W., and Schmidt, W.F., "On the Speed of Training Networks with Correlated Features", to appear in the proceedings of the IEEE Conference on Neural Networks, (San Francisco, Mar. 28 - April 1, 1993).

Barron 1991

Barron, A.R., and Cover, T.M., "Minimum Complexity Density Estimation", IEEE Transactions on Information Theory, Vol. 37, No. 4, July 1991, pp. 1034 - 1054.

Battiti 1992

Battiti, R., "First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method", Neural Computation, vol. 4, no. 2, pp. 141 - 166, 1992.

Baum 1988

Baum, E.B., "On the Capabilities of Multi-layer Perceptrons", Journal of Complexity 4, pp. 193 - 215, 1988.

Baum 1989

Baum, E.B., and Haussler, D., "What Size Net Gives Valid Generalization?", Neural Computation 1, pp. 151-160, 1989.

Baum 1990

Baum, E.B., "When Are K-NN and Back Propagation Accurate for Feasible Sized Sets of Examples?", proceedings of the EURASIP Workshop on Neural Networks 1990, L.B. Almeida and C.J. Wellekens (eds.), Sesimbra, Portugal, February 15-17, 1990, published as: Lecture Notes in Computer Science, Vol. 412, G. Goos and J. Hartmanis (eds.), Springer Verlag, 1990.

Bishop 1991

Bishop, C., "Improving the Generalization Properties of Radial Basis Function Neural Networks," Neural Computation, vol. 3, no. 4, pp. 579 - 588, 1991.

Bishop 1992

Bishop, C., "Exact Calculation of the Hessian Matrix for the Multi-Layer Perceptron," Neural Computation, vol. 4, no. 4, pp. 494 - 501, 1992.

Blumer 1986

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, K., "Classifying Learnable Geometric Concepts with the Vapnik-Chervonenkis Dimension", Proceedings of the 18th ACM Symposium on the Theory of Computing (Berkeley, California, May 28 - 30, 1986), ACM New York, pp. 273 - 282, 1986.

Blumer 1989

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, K., "Learnability and the Vapnik-Chervonenkis Dimension", Journal of the ACM, Vol. 36, No. 4, Oct. 1989, pp. 929-965.

Burton 1991

Burton, R.M., and Faris, W.G., "Reliable Evaluation of Neural Networks", Neural Networks , vol. 4, no. 3, pp. 411 - 416, 1991.

Chauvin 1989
Chauvin, Y., "A Back Propagation Algorithm with Optimal Use of Hidden Units", Advances in Neural Information Processing I, Morgan Kaufman Publishers, San Mateo, pp. 519 - 526, 1989.

COLT 1988
proceedings of the first workshop on Computational Learning Theory, Morgan Kaufmann, 1988.

COLT 1989
proceedings of the second workshop on Computational Learning Theory, Morgan Kaufmann, 1989.

COLT 1990
proceedings of the third workshop on Computational Learning Theory, Morgan Kaufmann, 1990.

Cover 1965
Cover, T.M., "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition", IEEE Transaction on Electronic Computers, Vol. EC-14, pp. 326-334, 1965.

Cover 1969
Cover, T.M., "Learning in Pattern Recognition", in: *Methodologies of Pattern Recognition*, S. Watanabe, Ed., New York: Academic, 1969, pp. 111 - 132.

Crick 1989
Crick, F., "The recent excitement about neural networks," Nature, vol. 337, 12 Jan. 1989.

Devijver 1982
Devijver, P.A, and Kittler, J., *Pattern Recognition, A Statistical Approach*, Prentice Hall, 1982.

Devroye 1982
Devroye, L., "Bounds for the Uniform Deviation of Empirical Measures", Journal of Multivariate Analysis, Vol. 12, pp. 72 - 79, 1982.

Devroye 1988
Devroye, L., "Automatic Pattern Recognition: A Study of the Probability of Error", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 4, July 1988.

Duda 1973
Duda, R.O., and Hart, P.E., *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, 1973.

Duin 1976
Duin, R.P.W., "On the Choice of Smoothing Parameters for Parzen Estimators of Probability Density Functions", IEEE Transactions on Computers, 1976, pp. 1175-1179.

Duin 1978
Duin, R.P.W., *On the Accuracy of Statistical Pattern Recognizers*, Dutch Efficiency Bureau, Pijnacker, The Netherlands, 1978.

Duin 1990
Duin, R.P.W., "Pattern Recognition and Neural Networks: a Confrontation," in: Proceedings of the Symposium Artificial Intelligence, Fundamentals and Applications (Delft, 3 December 1990), pp. 49-58.

Duin 1993
Duin, R.P.W., "Superlearning Capabilities of Neural Networks?," to appear in: Proceedings of the 8th Scandinavian Conference on Image Analysis", Tromso, Norway, May 1993.

Fahlman 1990
Fahlman, S.E., and Lebiere, C., "The Cascade Correlation Learning Architecture", in: Advances in Neural Information Processing Systems II, (Denver 1989), Lipmann, R.P., Moody, J., and Touretzky, D.S. (eds.), San Mateo, Morgan Kaufmann, pp. 524 - 532, 1990.

Foley 1972
Foley, D.H., "Considerations of Sample and Feature Size", IEEE Transactions on Information Theory, Vol. 18, No. 5, pp. 618 - 626, 1972.

Fukunaga 1988
Fukunaga, K. and Hayes, R.R., "The Reduced Parzen Classifier", IEEE Transactions on PAMI, Vol. 11, No. 4, July 1989.

Fukunaga 1990
Fukunaga, K., *Introduction to Statistical Pattern Recognition*, second edition, Academic Press, 1990.

Funahashi 1989
K. Funahashi, "On the Approximate Realization of Continuous Mappings by Neural Networks.", Neural Networks, Vol. 2 page 183-192, 1989.

Gallant 1990a
Gallant, S.I., "A Connectionist Learning Algorithm with Provable Generalization and Scaling Bounds", Neural Networks, Vol. 3, pp. 191 - 201, 1990.

Gallant 1990b
Gallant, S.I., "Perceptron-Based Learning Algorithms", IEEE Tr. on Neural Networks, Vol. 1, No. 2, 1990.

Geman 1992
Geman, S., Bienenstock, E., and Doursat, R., "Neural Networks and the Bias/Variance Dilemma," Neural Computation, vol. 4, no. 1, pp. 1 - 58, 1992.

Golomb 1991
Golomb, B.A., Lawrence, D.T., and Sejnowski, "SEXNET: A Neural Network Identifies Sex from Human Faces", in: *Neural Information Processing Systems 3*, Lippmann, R.P., Moody, J., and Touretzky, D.S. (eds.), Morgan Kaufmann Publishers, San-Mateo, California, U.S.A., pp. 572 - 577, 1991.

Gorman 1988a
Gorman, R.P., and Sejnowski, T.J., "Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets", Neural Networks, Vol. 1, No. 1, 1988.

Gorman 1988b
Gorman, R.P., and Sejnowski, T.J., "Learned Classification of Sonar Targets Using Massively Parallel Network", IEEE Tr. on ASSP, Vol. 36, No. 7, 1988, pp. 1135 - 1140.

Gyon 1992a
Gyon, I., Vapnik, V., Boser, B., Bottou, L., and Solla, S.A., "Capacity Control in Linear Classifiers for Pattern Recognition", in: Proceedings of the 11th IAPR International Conference on Pattern Recognition, Volume II, Conference B: Pattern Recognition Methodology and Systems (ICPR 11, The Hague, The Netherlands, August 30 - September 3, 1992), IEEE Computer Society Press, Los Alamitos, California, USA, 1992.

Gyon 1992b
Gyon, I., Vapnik, V., Boser, B., Bottou, L., and Solla, S.A., "Structural Risk Minimization for Character Recognition", in: *Advances in Neural Information Processing Systems IV*, Moody, J.E., Hanson, S.J., and Lippman, R.P. (eds.), Morgan Kaufman Publishers, San Mateo, California, USA, 1992.

Hartman 1990
Hartman, E, Keeler, J.D., and Kowalski, J.M., "Layered Neural Networks with Gaussian Hidden Units as Universal Approximations," Neural Computation, vol. 2, no. 2, pp. 210 - 215, 1990.

Haussler 1991
Haussler, D., "Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications", Technical report U.C.S.C.-CRL-91-02, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, 1991.

Haussler 1992
Haussler, D., Kearns, M. and Schapire, R., "Bounds on the Sample Complexity of Bayesian Learning using Information Theory and the VC-Dimension", to appear in the proceedings of Neural Information Processing Systems, Morgan Kaufmann, 1992.

Hertz 1991
Hertz, J., Krogh, A., and Palmer, R.G., *Introduction to the Theory of Neural Computation*, Addison Wesley, 1991.

Hinton 1986
Hinton, G.E., and Sejnowski, T.J., "Learning and Relearning in Boltzmann machines", chapter 7 in: Rumelhart, D.E., and McClelland, J.L., "Parallel Distributed Processing: Explorations in the micro structure of cognition". Vol 1. MIT press 1986.

Hoeffding 1963
Hoeffding, W., "Probability Inequalities for Sums of Bounded Random Variables", American Statistical Association Journal, March 1963, pp. 13 - 30.

Horn 1988
Horn, R.A., and Johnson, C.R., *Matrix Analysis*, Cambridge University Press, Cambridge, 1988.

Hornik 1989
K. Hornik, "Multi-Layer Feedforward Networks are Universal Approximators", Neural Networks, Vol. 2 page 359-366, 1989.

Hughes 1968
Hughes, G.F., "On the Mean Accuracy of Statistical Pattern Recognizers", IEEE Transactions on Information Theory, Vol. 14, pp. 55 - 63, 1968.

ICNN 1990
Proceedings of the International Neural Network Conference (Paris, France, July 9-13 1990).

ICPR 1992
Proceedings of the 11th IAPR International Conference on Pattern Recognition, (ICPR 11, The Hague, The Netherlands, August 30 - September 3, 1992), IEEE Computer Society Press, Los Alamitos, California, USA, 1992.

IJCNN 1991
Proceedings of the International Joint Conference on Neural Networks (Seattle, 8-12 July 1991), IEEE, Piscataway, U.S.A., 1991.

IEEECNN 1993
Proceedings of the IEEE Conference on Neural Networks (San Francisco, March 28 - April 1), IEEE, Piscataway, U.S.A., 1993.

Jain 1978
Jain, A.K., and Waller, W., "On the Optimum Number of Features in the Classification of Multivariate Gaussian Data", Pattern Recognition, Vol. 10, pp. 365 - 374, 1978.

Jain 1982
Jain, A.K., and Chandrasekaran, B., "Dimensionality and Sample Size Considerations in Pattern Recognition Practice", in: P.R. Krishnaiah and L.N. Kanal, editors, *Handbook of Statistics*, Vol. 2, North Holland Publishing Company, pp. 835-855, 1982.

Kirkpatrick 1983
Kirkpatrick, S., Gelatt Jr., C.D, and Vecchi, M.P., "Optimization by Simulated Annealing", Science, N.Y. 220, pp. 671-680, 1983.

Kohonen 1988
Kohonen, T., Barna, G., and Chrisley, R., "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies", proceedings of the Neuro '88 conference, Paris, June 1988.

Kolen 1991
Kolen, J.F., and Pollack, J.B., "Back-Propagation is Sensitive to Initial Conditions", in: *Neural Information Processing Systems*, R.P. Lippmann, J.E. Moody and D.S. Touretzky (eds.), Morgan Kaufmann Publishers, San Mateo, California, USA, 1991, pp. 860 - 867.

Koontz 1972
Koontz, W.L.G., and Fukunaga, K., "Asymptotic Analysis of a Nonparametric Clustering Technique", IEEE Transactions on Computers, Vol. c-21, No. 9, September 1972.

Kraaijveld 1989
Kraaijveld, M.A., "On the Application of Connectionist Models for Pattern Recognition, Robotics and Computer Vision: a Technical Report", Delft University Press, Delft, The Netherlands, 1989.

Kraaijveld 1990
Kraaijveld M.A., and Duin, R.P.W., "On Backpropagation Learning of Edited Data Sets," in: Proceedings of the International Neural Network Conference (Paris, France, July 9-13 1990), pp. 741-744.

Kraaijveld 1991a
Kraaijveld, M.A., and Duin, R.P.W., "An Optimal Stopping Criterion for Backpropagation Learning," Neural Network World, vol. 1, no. 6, pp. 365-370, 1991.

Kraaijveld 1991b
Kraaijveld, M.A., and Duin, R.P.W., "Generalization capabilities of minimal kernel-based networks," in: Proceedings of the International Joint Conference on Neural Networks (Seattle, 8-12 July 1991), pp. I-843 - I-848, IEEE, Piscataway, U.S.A., 1991.

Kraaijveld 1991c
Kraaijveld, M.A., and Duin, R.P.W., "How Much Can We Learn From Examples?", in: Second Quinquennial Review, eds. J.J. Gerbrands, F.C.A. Groen, C. Kamminga, A.W.M. Smeulders, M.A. Viergever, A.M. Vossepoel, 12 pages, Nederlandse Vereniging voor Patroonherkenning en Beeldverwerking, Delft, Oct. 22, 1991.

Kraaijveld 1992
Kraaijveld, M.A., and Duin, R.P.W., "On the Capacity of Feedforward Networks with Shared Weights", invited paper, in: Proceedings of the SPIE conference on the Science of Artificial Neural Networks, D.W. Ruck (editor), (Orlando, April 20-24, 1992), SPIE proceedings series, Vol. 1710, SPIE, Bellingham, Washington, USA, pp. 406 - 414, 1992.

Kraaijveld 1993
Kraaijveld, M.A., and Schmidt, W.F., "ANN-lib - Neural Networks Simulation Library", manual of C-library for developing, training and testing neural network algorithms, Pattern Recognition Group, Department of Applied Physics, Delft University of Technology, Delft, The Netherlands, 29 pages, 1990 / 1991 / 1992 / 1993.

Lancaster 1985
Lancaster, P., and Tismenetsky, M., *The Theory of Matrices*, second edition, Academic Press, 1985.

le Cun 1985
le Cun, Y., "Une Procedure d'Apprentissage pour Réseau à Seuil Assymétrique", in: *Cognitiva 85: A la Frontière de l'Intelligence Artificielle des Sciences de la Connaissance des Neurosciences*, (Paris 1985), pp. 599 - 604, Paris: CESTA, 1985.

le Cun 1989
le Cun, Y., "Generalization and Network Design Strategies", in Pfeifer, R., Schreter, Z., Fogelman, F. and Steels, L., editors, *Connectionism in Perspective*, Zürich, Switzerland, Elsevier 1989.

le Cun 1990a
le Cun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D., "Handwritten Digit Recognition with a Back-Propagation Network", in Touretzky, D., editor, *Neural Information Processing Systems*, Vol. 2, Morgan Kaufmann, 1990.

le Cun 1990b
le Cun, Y., Denker, J.S., and Solla, S., "Optimal Brain Damage", in: Advances in Neural Information Processing Systems II, (Denver 1989), Lipmann, R.P., Moody, J., and Touretzky, D.S. (eds.), San Mateo, Morgan Kaufmann, pp. 598 - 605, 1990.

le Cun 1991a
le Cun, Y., Kanter, I., and Solla, S., "Second Order Properties of Error Surfaces: Learning Time and Generalization", in: Advances in Neural Information Processing Systems III, (Denver 1990), ed. D.S. Touretzky, San Mateo, Morgan Kaufmann, pp. 918 - 924, 1991.

le Cun 1991b
le Cun, Y., Kanter, I., and Solla, S.A., "Eigenvalues of Covariance Matrices: Applications to Neural-Network Learning", Physical Review Letters, Vol. 66, Nr. 18, May 6, 1991, pp. 2396 - 2399.

Levin 1992
Levin, E., le Cun, Y., and Vapnik, V.N., "Measuring the Capacity of a Learning Machine II", AT&T unpublished manuscript, AT&T Bell Laboratories, 1992.

Maas 1990
Maas, H. van der, Verschure, P.F.M.J., and Molenaar, P.C.M, "A Note on Chaotic Behavior in Simple Neural Networks", Neural Networks, Vol. 3, No. 1, 1990.

MacKay 1992
MacKay, D.J.C., "A Practical Bayesian Framework for Backpropagation Networks", Neural Computation, Vol. 4, pp. 448 - 472, 1992.

McClelland 1986
McClelland, J.L. and Rumelhart, D.E., *Parallel Distributed Processing: Explorations in the Micro Structure of Cognition*, Vol 2, MIT-press 1986.

Miller 1991
Miller, J.W., Goodman, R., and Smyth, "Objective Functions for Probability Estimation", in: Proceedings of the International Joint Conference on Neural Networks (Seattle, 8-12 July 1991), pp. I-881 - I-886, IEEE, Piscataway, U.S.A., 1991.

Minsky 1969
Minsky, M., and Papert, S., *Perceptrons: an introduction to computational geometry*, MIT press 1969

Mitchison 1989
Mitchison, G.J. and Durbin, R.M., "Bounds on the Learning Capacity of Some Multi-Layer Networks", Biological Cybernetics, Vol. 60, pp. 345 - 356, 1989.

Mood 1974
Mood, A.M., Graybill, F.A, and Boes, D.C., *Introduction to the Theory of Statistics*, third edition, McGraw-Hill, 1974.

Moody 1992
Moody, J.E., "The Effective Number of Parameters: an Analysis of Generalization and Regularization in Non-Linear Systems", in: *Advances in Neural Information Processing Systems IV*, Moody, J.E., Hanson, S.J., and Lippman, R.P. (eds.), Morgan Kaufman Publishers, San Mateo, California, USA, 1992.

Mozer 1989
Mozer, M.C., and Smolensky, P., "Skeletonization: a Technique for Trimming the Fat from a Network via Relevance Assessment", in: Advances in Neural Information Processing Systems I, (Denver 1988), ed. D.S. Touretzky, San Mateo, Morgan Kaufmann, pp. 107 - 115, 1989.

Natarajan 1991
Natarajan, B.K., *Machine Learning: a Theoretical Approach*, Morgan Kaufmann Publishers Inc., San Mateo, California, 1991.

Orfanidis 1990
Orfanidis, S.J., "Gram-Schmidt Neural Nets", Neural Computation, Vol. 2, pp. 116 - 126, 1990.

Otterloo 1978
Otterloo, P.J. van, and Young, I.T., "A Distribution-Free Geometric Upper Bound for the Probability of Error of a Minimum Distance Classifier", Pattern Recognition, Vol. 10, pp. 281 - 286, 1978

Owens 1989
Owens, A.J., and Filkin, D.L., "Efficient training of the back propagation network by solving a system of stiff ordinary differential equations", in proc. of the IEEE/INNS International Conference on Neural Networks, Washington D.C., June 1989

Park 1991
Park, J, and Sandberg, I.W., "Universal Approximation Using Radial-Basis-Function Networks," Neural Computation, vol. 3, no. 2, pp. 246 - 257, 1991.

Parker 1985
Parker, D.B., "Learning Logic", Technical Report TR-47, Center for Computational Research in Economics and Management Science", Massachusetts Instititute of Technology, Cambridge, Massachusetts, USA, 1985.

Parzen 1962
Parzen, E., "On the Estimation of a Probability Density Function and the Mode", Ann. Math. Statist., Vol. 33, pp. 1065-1076, 1962.

Poggio 1989
Poggio, T., and Girosi, F., "A Theory of Networks for Approximation and Learning", MIT AI-lab Memo 1140, July 1989.

Pomerleau 1989
Pomerleau, D.A., "ALVINN: An Autonomous Land Vehicle in a Neural Network", in: Advances in Neural Information Processing Systems I, (Denver 1988), ed. D.S. Touretzky, San Mateo, Morgan Kaufmann, pp. 305 - 313, 1989.

Press 1988
Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., "Numerical Recipes in C", Cambridge University Press, 1988.

Raudys 1976
Raudys, S.J., "On Dimensionality, Learning Sample Size and Complexity of Classification Algorithms", in: Proceedings of the Third International Joint Conference on Pattern Recognition, Coronado, CA, USA, pp. 166 - 169, 1976.

Raudys 1991a
Raudys, S., and Jain, A.K., "Small Sample Size Problems in Designing Artificial Neural Networks", in: Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections", Sethi and Jain (eds.), Elsevier Science Publishers, pp. 33 - 50, 1991.

Raudys 1991b
Raudys, S.J., and Jain, A.K., "Small sample size effects in statistical pattern recognition: recommendations for practitioners," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-13, no. 3, pp. 252 - 264, 1991.

Robinson 1988
Robinson, A.J., Niranjan, M. and Fallside, F, "Generalizing the Nodes of the Error Propagation Network", Technical Report TR. 25, Cambridge University Engineering Department, Nov. 1988.

Rosenblatt 1958
Rosenblatt, F., "The perceptron: a probabilistic model for information storage and organization in the brain", Psychological Review 65: 386-408. In: Anderson, J.A., and Rosenfeld, E., (Eds.), *Neurocomputing: A Collection of Classic Papers*, MIT press 1987.

Rumelhart 1986
Rumelhart, D.E., Hinton, G.E., and Williams, R.J., *Parallel Distributed Processing: Explorations in the Micro Structure of Cognition*, Vol 1, MIT-press 1986.

Sato 1991
Sato, A., Yamada, K., Tsukumo, J., and Temma, T., Neural Network Models for Incremental Learning", proc. of the International Conference on Neural Networks, Helsinki, 1991.

Sauer 1972
Sauer, N., "On the Density of Families of Sets", Journal of Combinatorial Theory (Series A), Vol. 13, pp. 145 - 147, 1972.

Schmidt 1992
Schmidt, W.F., Kraaijveld, M.A., and Duin, R.P.W., "Feed Forward Neural Networks with Random Weights", in: Proceedings of the Eleventh International Conference on Pattern Recognition, (The Hague, Aug. 30 - Sept. 3, 1992), pp. B-1 - B-4, IEEE Computer Society Press, Los Alamitos, California, U.S.A., 1992.

Schmidt 1993a
Schmidt, W.F., Raudys, S., Kraaijveld, M.A., Skurikhina, M., and Duin, R.P.W., "Initializations, Back-Propagation and Generalization of Feed-Forward Classifiers", to appear in the proceedings of the IEEE Conference on Neural Networks, (San Francisco, Mar. 28 - April 1, 1993).

Schmidt 1993b
Schmidt, W.F., Raudys, S., Kraaijveld, M.A., Skurikhina, M., and Duin, R.P.W., "Initializations, Back-Propagation and Generalization of Feed-Forward Classifiers", Submitted to Neural Networks.

Sejnowski 1987
Sejnowski, T.J., and Rosenberg, C.R., "Parallel Networks that learn to Pronounce English Text", Complex Systems 1 (1987) 145-168.

Sethi 1991
Sethi, I.K., and Jain, A.K., *Artificial Neural Networks and Statistical Pattern Recognition: Old and New Connections*, Elsevier Science Publishers, 1991.

Sjöberg 1992
Sjöberg, J., and Ljung, L., "Overtraining, regularization, and searching for minimum in neural networks", Neuroprose archive, februari 1992.

Specht 1990
Specht, D.F., "Probabilistic Neural Networks", Neural Networks, Vol. 3, pp. 109-118, 1990.

Strang 1988
Strang, G., *Linear Algebra and its Applications*, Harcourt Brace Jovanovich Publishers, San Diego, 1988.

Tesauro 1990
Tesauro, G., Neurogammon Wins Computer Olympiad", Neural Computation, Vol. 1, pp. 321 - 323, 1990.

Tugay 1989
Tugay, M.A., and Tanik, Y., "Properties of the Momentum LMS Algorithm", Signal Processing, Vol. 18, 1989, pp. 117 - 127.

Valiant 1984
Valiant, L.G., "A Theory of the Learnable", Communications of the ACM, Vol. 27, pp. 1134-1142, November 1984.

Vapnik 1971a
Vapnik, V.N., and Chervonenkis, A.Ya., "Theory of Uniform Convergence of Frequencies of Events to their Probabilities and Problems of Search for an Optimal Solution from Empirical Data", Automat. Remote Contr., Vol. 32, pp. 207 - 217, 1971.

Vapnik 1971b
Vapnik, V.N., and Chervonenkis, A.Ya., "On the Uniform Convergence of Relative Frequencies of Events to their Probabilities", Theory of Probability and Applications, Vol. 16, pp. 264 - 280, 1971.

Vapnik 1982
Vapnik, V.N., *Estimation of Dependencies Based on Empirical Data*, Springer Verlag, New York, 1982.

Vapnik 1992
Vapnik, V.N., "Measuring the Capacity of a Learning Machine", AT&T unpublished manuscript, AT&T Bell Laboratories, Holmdel, NJ, April 3, 1992.

Weigend 1991
Weigend, A.S., Rumelhart, D.E., and Huberman, B.A., "Generalization by Weight-Elimination Applied to Currency Rate Prediction", in: Proceedings of the International Joint Conference on Neural Networks (Seattle, 8-12 July 1991), pp. I-837 - I-841, IEEE, Piscataway, U.S.A., 1991.

Weiss 1991
Weiss, S.M., and Kulikowski, C.A., *Computer Systems that Learn*, Morgan Kaufmann Publisher Inc., San Mateo, California, USA, 1991.

Wenocur 1981
Wenocur, R.S, and Dudley, R.M., "Some special Vapnik-Chervonenkis classes", Discrete Mathematics, Vol. 33, pp. 313-318, 1981.

Werbos 1974
Werbos, P.J., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", Ph.D. Thesis, Harvard University, 1974.

Weymare 1991
Weymare, N., and Martens, J.P., "A Fast and Robust Learning Algorithm for Feedforward Neural Networks," Neural Networks , vol. 4, no. 3, pp. 361 - 370, 1991.

Widrow 1960
Widrow, B., and Hoff, M.E., "Adaptive Switching Circuits", 1960 IRE Wescon Convention Record, New York: IRE, pp. 96-104. In: Anderson, J.A., and Rosenfeld, E., (Eds.), Neurocomputing: A collection of classic papers. MIT press 1987.

Widrow 1985
B. Widrow and S.D.Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey 1985.

Widrow 1988
Widrow, B., and Winter, R., "Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition", IEEE Computer, March 1988.

Wozencraft 1965
Wozencraft, J.M., and Jacobs, I.M., *Principles of Communication Engineering*, John Wiley & Sons Inc., New York, 1965.

Young 1978
Young, I.T., "Further Considerations of Sample and Feature Size", IEEE Transactions on Information Theory, Vol. 24, No. 6, pp. 773 - 775, 1978.

List of Symbols

m	sample size of a learning set
n	sample size of a test set
d	the dimensionality of the feature space.
L	the cardinality of a set of classifiers
V	the capacity or Vapnik-Chervonenkis dimension of a set of classifiers
V^*	the effective capacity of a set of classifiers
$\hat{V}^*$	the estimated effective capacity of a set of classifiers
V_A^*	the actual capacity of a a set of classifiers
$\hat{V}_A^*$	the estimated actual capacity of a a set of classifiers
$\boldsymbol{\Theta}$	the parameter space of a classifier
$\boldsymbol{\theta}$	a parameter vector of a classifier
$\boldsymbol{\theta}^i$	the parameters of the units in layer i of a multi-layer network
W	the cardinality of the parameter space $\boldsymbol{\Theta}$
$F(\boldsymbol{X}, \boldsymbol{\Theta})$	the set of classification functions in the space $\boldsymbol{X}$ with parameter vector in $\boldsymbol{\Theta}$, mapping onto $\{0, 1\}$
F	a set of classifiers (abbreviation of $F(\boldsymbol{X}, \boldsymbol{\Theta})$)
$f(\boldsymbol{x}, \boldsymbol{\theta})$	a classification function in the space $\boldsymbol{X}$ with parameter vector $\boldsymbol{\theta}$ mapping onto $\{0, 1\}$
f	a classifier (abbreviation of $f(\boldsymbol{x}, \boldsymbol{\theta})$)
$F^*(\boldsymbol{X}, \boldsymbol{\Theta})$	the set of continuous functions in the space $\boldsymbol{X}$ with parameter vector in $\boldsymbol{\Theta}$, mapping onto $(0, 1)$
F^*	a set of continuous functions (abbreviation of $F^*(\boldsymbol{X}, \boldsymbol{\Theta})$)
$f^*(\boldsymbol{x}, \boldsymbol{\theta})$	a continuous function in the space $\boldsymbol{X}$ with parameter vector $\boldsymbol{\theta}$ mapping onto $(0, 1)$
f^*	a continuous function (abbreviation of $f^*(\boldsymbol{x}, \boldsymbol{\theta})$)
$S_{\exp}^F(m)$	the expected number of dichotomies that the classifiers in F can induce on a sample of m points.
$S_{\max}^F(m)$	the maximum number of dichotomies that the classifiers in F can induce on a sample of m points.
$S_{\max}^{ltf}(m,d)$	the maximum number of dichotomies that a linear threshold function in d-dimensional space can induce on a sample of m points.
$\varepsilon(\boldsymbol{\theta})$	the true error of a classifier with parameter vector $\boldsymbol{\theta}$

$\hat{\varepsilon}(\boldsymbol{\theta})$	the estimated error of a classifier with parameter vector $\boldsymbol{\theta}$
ε^*	the Bayes error
δ	the probability of (maximum) (relative) deviation of the true and estimated error; the confidence factor
χ	the (relative) deviation between the true and estimated error
Ξ^m	a sample of size m
$\xi_1, \ldots, \xi_m$	a sample of size m
ξ_i	element i of a sample
λ	the class label of an element of a sample, $\lambda \in \{0,1\}$
$SE\left(\Xi^m, \boldsymbol{\theta}\right)$	the squared mapping error of a function $f^*(\boldsymbol{x}, \boldsymbol{\theta})$ on a sample Ξ^m
SE	the squared mapping error (abbreviation of $SE\left(\Xi^m, \boldsymbol{\theta}\right)$)
$p()$	the probability density function
$P\{\}$	the probability of an event
$E\{\}$	the expectation operator
N	the number of units of a feedforward multi-layer network.
h	the number of hidden units of a feedforward multi-layer network
l	the number of layers in a feedforward multi-layer network
$act^{ip}(\boldsymbol{x}, \boldsymbol{\theta})$	the inner product activation function of a unit in a multi-layer network.
$act^{Ed}(\boldsymbol{x}, \boldsymbol{\theta})$	the Euclidean distance activation function of a unit in a multi-layer feedforward network.
$out^{lin}(x)$	the linear output function of a unit in a multi-layer feedforward network.
$out^{thr}(x)$	the threshold output function of a unit in a multi-layer feedforward network.
$out^{sigm}(x)$	the sigmoidal output function of a unit in a multi-layer network.
$out^{Gaus}(x)$	the Gaussian output function of a unit in a multi-layer network.
η	the learning rate or the step-size of the adaptation of the parameters in an iterative learning procedure
α	the momentum term of the adaptation of the parameters in an iterative learning procedure
$I(x)$	the indicator function
$\langle x \rangle$	x rounded off to the nearest integer
$\lceil x \rceil$	the smallest integer larger than or equal to x.
$\lfloor x \rfloor$	the largest integer smaller than or equal to x.
$log\ (x)$	the logarithm base 2 of x
$ln\ (x)$	the natural logarithm of x
e	the base of the natural logarithm: $e \approx 2.7182818$

Summary

In this thesis a number of theoretical and practical aspects of the small sample behavior of multi-layer feedforward network classifiers have been investigated. The context in which these investigations have been performed is that of the literature on statistical pattern recognition.

The first part of this thesis is dedicated to a number of theoretical issues on this subject. In chapter one of part one, various results on small sample problems in statistical pattern recognition have been reviewed. It is discussed that, according to the statistical pattern recognition literature, the number of training samples should be proportional to the number of free parameters of the classifier, in order to obtain valid generalization. The number of free parameters is in this context interpreted as a measure of complexity of a classifier.

In chapter two, these results are refined by the introduction of an alternative and more powerful measure of complexity of a classifier, that is called the *capacity* or *Vapnik-Chervonenkis dimension.* Based on the notion of capacity, an elegant mathematical framework is discussed, that provides a complete characterization of small training sample problems in a minimax sense. Three improved theorems are presented that show explicit relations between the capacity of a classification function, the training sample size, and the maximum relative deviation of the true error of the classifier and the error frequency on the training sample. As this theoretical framework is distribution-free and independent of the learning procedure to train the classifier, its predictions are generally very pessimistic. An important part of its value, therefore, is the fact that it provides a considerable amount of insight in the relation between the variables that are involved in the design of a classifier.

In chapter three, the Vapnik-Chervonenkis theory is applied to various types of multi-layer feedforward network classifiers. In the first place, a theorem is discussed that upper bounds the capacity of the general class of multi-layer feedforward network classifiers that have linear threshold units. This theorem, together with the mathematical framework that is discussed in chapter 2, results in an improved corollary that lower

bounds the training sample size, as a function of the number of free parameters in the network.

A second application of the Vapnik-Chervonenkis theory is the class of minimal kernel-based networks. This is a class of networks that is related to the Parzen classifier. A number of training procedures for such networks is discussed, and two theorems are given that allow to make statements on how the amount of training data available, should be split into a learning set and a test set, in order to obtain valid generalization of the network.

Finally, in the last section of chapter three it is investigated how a popular method to lower the number of free parameters of a network affects its capacity. This method is called weight sharing, and aims at coupling several weight values in a network, such that the total number of free parameters in the network becomes smaller. A theorem is proven that upper bounds the capacity of such a network, in terms of the remaining number of free parameters. Moreover, it is shown that the weight-sharing technique is an efficient way of removing free parameters from a network, since the decrease of the capacity of the network is faster than the decrease of the number of free parameters.

In chapter four it was investigated how the (iterative) learning procedure that is used to train a network influences its capacity. First, a number of experiments are presented that convincingly illustrate that the performance of a multi-layer feedforward network, that is trained with the backpropagation algorithm, is consistently better than one could expect from the number of free parameters in the network. As an explanation of this effect, it is discussed that the capabilities of an (iterative) learning procedure in searching through the parameter space, may dramatically influence the validity of generalization of the classifier. In fact, if the iterative learning procedure only investigates a small area of the parameter space, the limited use of the available parameters implies that the performance of the classifier on the training set is much more significant than the *number* of free parameters suggests.

This idea gives rise to the notion of an *effective capacity* of a classifier, to account for the specific properties of a learning procedure. Furthermore, the notion of an *actual capacity* was introduced to account for specific properties of the training data. An experimental procedure was introduced to estimate the effective capacity and the actual capacity of a multi-layer feedforward network classifier. In a number of experiments it is convincingly illustrated that the effective capacity of a multi-layer feedforward network that is trained with the backpropagation algorithm may indeed be orders of magnitude smaller than its true capacity.

In chapter five, finally, the results of the theoretical part of this thesis are reviewed and a number of recommendations on the applicability of multi-layer feedforward network classifiers for solving statistical pattern recognition problems are given.

The second part of this thesis is dedicated to two practical issues of multi-layer feedforward networks. These are the stopping criterion, that determines when the iterative learning procedure should be terminated, and the issue of the speed of the iterative learning procedure.

The issue of the stopping criterion is addressed in chapter six. It is discussed that various ad hoc stopping criteria, that are generally used in practice, may suffer from serious difficulties. This problem is resolved by the application of a technique from the statistical pattern recognition literature; the editing algorithm. The editing algorithm aims at transforming the training data such that the intrinsic overlap of the underlying distributions of the classes is effectively removed. This results in a simple and optimal stopping criterion: continue the iterative learning procedure until all samples in the edited training data are correctly classified. Two additional effects of this procedure are a moderate improvement of the learning speed of the backpropagation algorithm, and an increased confidence that the performance of the network is close to Bayes-optimal.

In the final chapter, the issue of the speed of the iterative learning procedure is addressed. It is shown that for a single-layer network, an increase of the dimensionality of the feature space will never increase the speed of the iterative learning procedure. In fact, the learning speed will at best remain equal, but will deteriorate in most cases. This result can be applied in practical problems that are time constrained and has some implications for the training of multi-layer feedforward network classifiers.

Samenvatting

In dit proefschrift worden de consequenties van een kleine leerverzameling op het generaliserende vermogen van meer-laags-netwerk-classificatoren (multi-layer feedforward network classifiers) bestudeerd. De context waarbinnen dit onderzoek heeft plaats gevonden is die van de statistische patroonherkenning.

Het eerste gedeelte van het proefschrift is gewijd aan een aantal theoretische aspecten van dit onderwerp. In het eerste hoofdstuk van deel één wordt eerst een overzicht gegeven van diverse resultaten uit de statistische patroonherkenningsliteratuur. Zo wordt onder andere het bekende resultaat besproken, dat de grootte van de leerverzameling evenredig moet zijn met het aantal vrije parameters van de classificator. Slechts dan is het mogelijk om te garanderen dat de prestaties van de classificator op de leerverzameling voldoende representatief zijn voor de prestaties op een onafhankelijke testverzameling. Het aantal vrije parameters van een classificator functioneert in deze context als een maat voor de complexiteit van de classificator.

In hoofdstuk twee wordt een verfijning van dit resultaat aangebracht door de introductie van een alternatieve en krachtiger maat voor de complexiteit van een classificator: de *capaciteit* of *Vapnik-Chervonenkis dimensie*. Gebaseerd op het begrip capaciteit wordt een elegante minimax theorie bediscussieerd, die de problemen die samenhangen met een kleine leerverzameling volledig karakteriseert. Zo worden drie verbeterde theorema's gepresenteerd die expliciet de relaties aangeven tussen de capaciteit van een classificator, de grootte van de leerverzameling en de maximale relatieve afwijking van de echte fout van de classificator en de fout op de leerverzameling. Omdat in deze theorie geen aannames omtrent de onderliggende kansdichtheidsfuncties van de klassen gedaan worden, zijn de resulterende voorspelling voor praktische toepassingen vaak erg pessimistisch. Een belangrijk deel van de waarde van dit theoretisch raamwerk schuilt dan ook in het feit dat een goed inzicht verkregen wordt in de wijze waarop de diverse parameters intrinsiek met elkaar samen hangen.

In hoofdstuk drie wordt the Vapnik-Chervonenkis theorie toegepast op diverse klassen van meer-laags-netwerk-classificatoren. Ten eerste wordt een bestaand theorema besproken dat de capaciteit van meer-laags netwerken met lineaire drempel units van

boven begrensd. Dit theorema, tezamen met het mathematisch raamwerk van hoofdstuk twee, resulteert in een verbeterde afgrenzing van de minimale grootte van de leerverzameling als functie van het aantal vrije parameters in het netwerk.

Een tweede toepassing van de Vapnik-Chervonenkis theorie is de klasse van minimale netwerken met radiale basisfuncties. Dit is een klasse van patroonherkenners die gerelateerd is aan de Parzen-classificator. Een aantal leerprocedures voor dergelijke netwerken wordt besproken, en twee theorema's worden gepresenteerd die aangeven hoe de beschikbare hoeveelheid gegevens verdeeld moet worden over een leerverzameling en een testverzameling, om een betrouwbare schatting van de prestaties van het netwerk te verkrijgen.

Tenslotte wordt in het laatste gedeelte van hoofdstuk drie onderzocht hoe een bekende methode om het aantal vrije parameters van een netwerk te verminderen de capaciteit beïnvloedt. Bij deze methode wordt een aantal parameters in het netwerk aan elkaar gekoppeld, zodat het totale aantal vrije parameters van het netwerk afneemt. Een theorema wordt gepresenteerd, dat de capaciteit van een dergelijk netwerk van boven afgrenst als functie van het aantal overgebleven vrije parameters. Verder wordt bewezen dat het koppelen van parameters een efficiënte methode is om het aantal vrije parameters van het netwerk te verminderen, omdat de capaciteit van het netwerk sneller afneemt dan het aantal vrije parameters.

In hoofdstuk vier wordt bekeken hoe de (iteratieve) leerprocedure, die gebruikt wordt in de leerfase van het netwerk, de capaciteit van het netwerk beïnvloedt. Ten eerste worden de resultaten van een experiment besproken, dat op overtuigende wijze illustreert dat de prestaties van een meer-laags-netwerk-classificator, die met behulp van het backpropagation algoritme getraind wordt, structureel beter zijn dan op grond van het aantal vrije parameters in het netwerk verwacht kon worden. Als een verklaring voor dit verschijnsel wordt aangegeven, dat de kracht van de leerprocedure bij het doorzoeken van de parameter-ruimte een dramatische invloed kan hebben op de capaciteit van de classificator. Indien de iteratieve leerprocedure slechts een zeer beperkt gedeelte van de totale parameter-ruimte van het netwerk exploreert, is door het beperkte gebruik van de beschikbare parameters de prestaties van de classificator op de leerverzameling veel significanter dan het *aantal* vrije parameters doet vermoeden.

Deze constatering geeft aanleiding tot de definitie van het begrip *effectieve capaciteit* van een classificator, waarin de eigenschappen van de iteratieve leerprocedure verdisconteerd zijn. Verder wordt het begrip *actuele capaciteit* geïntroduceerd, om ook de specifieke eigenschappen van de trainingsdata op de capaciteit van een classificator te verdisconteren. Om deze effectieve en actuele capaciteit van een netwerk te schatten wordt

een experimentele procedure voorgesteld. In een aantal experimenten wordt overtuigend geïllustreerd dat de effectieve capaciteit van een meer-laags-netwerk-classificator, die met het backpropagation algoritme getraind wordt, ordes van grootte kleiner kan zijn dan de echte capaciteit van het netwerk.

In hoofdstuk vijf, tenslotte, worden de resultaten van het theoretische gedeelte van dit proefschrift samengevat, en wordt een aantal aanbevelingen gedaan voor de toepassing van meer-laags-netwerk-classificatoren bij het oplossen van statistische patroonherkenningsproblemen.

Het tweede gedeelte van dit proefschrift is gewijd aan twee praktische aspecten van meer-laags-netwerk-classificatoren. Dit zijn het stopcriterium, dat bepaalt wanneer de iteratieve leerprocedure beëindigd kan worden, en het probleem van de leersnelheid van de iteratieve leerprocedure.

Het probleem van de keuze van een stopcriterium wordt besproken in hoofdstuk zes. Een analyse van diverse ad-hoc stopcriteria, die in veel praktische toepassingen gebruikt worden, toont aan dat zij kunnen leiden tot suboptimale prestaties van de resulterende classificator. Dit probleem wordt opgelost door de toepassing van een techniek uit de statistische patroonherkenningsliteratuur: het editing-algoritme. Het editing-algoritme transformeert de leerverzameling zodanig dat de intrinsieke overlap van de distributies van de klassen effectief verwijderd wordt. Dit resulteert in een eenvoudig en optimaal stopcriterium: beëindig de leerprocedure indien alle elementen van de leerverzameling goed geclassificeerd worden. Twee bijkomende effecten van deze procedure zijn een bescheiden verhoging van de leersnelheid van het backpropagation algoritme, en een toename van de kans dat de prestaties van het netwerk die van de Bayes-classificator benaderen.

In het laatste hoofdstuk wordt het probleem van de leersnelheid van de iteratieve leerprocedure behandeld. Het wordt mathematisch bewezen dat voor een enkel-laags netwerk een toename van de dimensionaliteit van de kenmerkruimte nooit zal resulteren in een verhoging van de leersnelheid. De leersnelheid zal in feite alleen in het beste geval gelijk blijven. In de meeste gevallen zal een uitbreiding van de dimensionaliteit van het probleem echter resulteren in een lagere leersnelheid. Dit resultaat heeft toepassingen bij problemen waarbij de beschikbare leertijd beperkt is, en heeft een aantal implicaties voor de leerprocedure van meer-laags-netwerk-classificatoren.

Dankwoord

Het verrichten van (promotie) onderzoek is een vreemde bezigheid. Soms gaan er voor je gevoel weken of maanden van noeste arbeid verloren, omdat het begeerde nieuwe inzicht of goede idee uitblijft. Een helder moment op het strand kan een dergelijke periode echter weer geheel goed maken. Wat dat betreft is het jammer dat de vakgroep nooit geïnvesteerd heeft in een zonnebank of iets dergelijks, om het strandgevoel van de medewerkers wat meer te stimuleren.

Verder echter niets dan lof over de omgeving waarbinnen dit onderzoek is uitgevoerd. Het mogen werken in een zo professionele omgeving als de sectie Patroonherkennen van de faculteit der Technische Natuurkunde was zowel een grote eer als een groot genoegen. Vele mensen uit deze omgeving hebben dan ook direct of indirect bijgedragen aan het tot stand komen van dit proefschrift.

Een cruciale rol was wat dat betreft weggelegd voor mijn begeleider en co-promotor *Bob Duin*. Het is twijfelachtig of mijn werk zonder de bezielende en stimulerende begeleiding van Bob ooit tot een proefschrift had geleid. Bob was altijd bereid om te luisteren en mee te denken, om vervolgens precies de juiste vraag te stellen. Als geen ander heeft hij mij geleerd dat het stellen van de juiste vragen tot de kern van het wetenschappelijk onderzoek behoort.

De prettige en vruchtbare samenwerking met *Wouter Schmidt* is eveneens een belangrijk aspect van mijn werk geweest, en heeft in hoge mate aan het plezier in mijn werk bijgedragen. Zonder zijn grote inzet, kennis van zaken en zijn vriendschappelijke manier van samenwerken, was mijn eigen onderzoek nooit zo ver gekomen.

Het hoge nivo en de goede sfeer in de vakgroep worden voor een belangrijk deel veroorzaakt door het aanstekelijke enthousiasme voor het vakgebied van mijn promotor *Ted Young*. *Pieter Jonker* heeft mijn eerste schreden op het wetenschappelijke pad begeleid, en mijn werk in de sectie mogelijk gemaakt. Hoewel het inhoudelijk wat anders is gelopen dan we oorspronkelijk in gedachten hadden, is een groot woord van dank voor hem hiervoor beslist op zijn plaats. Ook *Piet Verbeek* en *Albert Vossepoel* stonden altijd klaar om vanuit een wat andere invalshoek mee te denken. Vooral hun enerverende lunch- en koffiediscussies zal ik missen.

Ad Herweijer heeft een meetbare positieve bijdrage aan dit proefschrift gehad. *Thom Hoeksma*, *Wim van Oel*, *Jan Straver*, *Peter de Jong* en *Rob Ekkers* hebben diverse praktische problemen in de loop der jaren voor me opgelost. *Wouter Smaal* was in diverse omstandigheden de redder in nood. *José de Bruin* en *Annelies Frijters* hebben me ontelbare keren bij administratieve en secretariële problemen terzijde gestaan.

Mijn collega promovendi zijn wellicht het meest direct verantwoordelijk voor de goede sfeer die ik in de vakgroep ervaren heb. Hoewel ik niet van bridge hou, heb ik de samenwerking altijd als zeer prettig, constructief en leerzaam ervaren. *Henri Vrooman*, *Erwin Koomen*, *Ben Verwer*, *Lucas van Vliet*, *Jim Mullikin*, *Hans Netten*, *Hans Buurman*, *Karel Strasters*, *Carol Orange*, *Erik Bouts* en *Rik Janssen* bedankt!

Fons Verbeek en *David Levelt* houden ook niet van bridge, maar gelukkig wel van muziek.

Anil Jain, *Jianchang Mao* and all other members of the Pattern Recognition and Image Processing Laboratory of the Department of Computer Science at Michigan State University are gratefully acknowledged for a most pleasant and fruitful visit at Michigan during the summer of 1991.

De studenten *Robert Bakker*, *Chris Nieuwenhuize*, *Daan van Setten*, en *Bart Venlet* hebben in de loop der jaren diverse goede ideeën en oplossingen aangedragen.

Op iets grotere afstand hebben de *DIAC-AIO's*, mijn *oud-collega's van de faculteit Kunst, Media en Technologie* van de Hogeschool voor de Kunsten in Utrecht, mijn collega *faculteitraadsleden*, mijn *nieuwe collega's* bij SHELL-research, mijn *familie* en al mijn (muzikale) *vrienden en kennissen* een grote directe en indirecte bijdrage aan (het plezier in) mijn werk geleverd.

Tenslotte dient een fatsoenlijk proefschrift natuurlijk met de bijna spreekwoordelijke excuses aan de partner afgesloten te worden. *Wies*, het spijt me dat ik vaak wel fysiek, maar niet mentaal aanwezig was. Bedankt voor alle steun, geduld, en liefde in de afgelopen jaren. Voor *Sebastiaan* een dikke knuffel.