Effects of Noise in Train Data for Neural Classifiers

by
E.W. van der Steen

Supervised by:
Dr. Ir. J.A.G. Nijhuis

Department of Computer Science
Rijksuniversiteit Groningen
Groningen, The Netherlands
May 2003

Abstract

English

When training neural networks with real-life data, there is always some form of noise involved. There are many types of noise, and the effects they have on neural networks vary greatly. Some noise causes patterns to be incorrectly labelled; other forms can garble the input values.

In neural applications, injecting noise into the inputs of training-data can improve performance, for it may enhance the generalization ability of neural networks. In contrast, it became apparent in recent studies, that reducing noise, like removing patterns of occluded images from the training data, actually improved performance.

When new data is obtained, often much pre-processing is done to verify whether the data is suitable for training. Noise is one of the causes that can reduce the learnability of a data set, and time and effort is often spent to deal with it in some way.

In this thesis we will analyse effects of noise in data used for training neural classifiers, with the purpose of estimating whether pre-processing is necessary or not.

Nederlands

Wanneer neurale netwerken getraind worden met real-life data, is er altijd ruis bij betrokken. Er zijn veel verschillende soorten ruis, en de effecten die ze op neurale netwerken hebben variëren sterk. Sommige soorten ruis zorgen ervoor dat patronen verkeerd gelabeld worden, ander vormen verstoren de input waarden ervan.

In neurale toepassingen kan het injecteren van ruis de prestaties verbeteren, omdat het de generalizatie van het netwerk kan versterken. In contrast heeft een recente studie uitgewezen dat het verwijderen van ruis, zoals het verwijderen van patronen van versperde karakters uit de training data, de prestaties ook verbeterde.

Als nieuwe data verkregen is, wordt er vaak veel voorbewerking op gedaan, om te kijken of de data geschikt is voor het trainen. Ruis is een van de oorzaken die de leerbaarheid van de data kan reduceren, en daarom wordt er vaak tijd en moeite in gestopt om het te behandelen.

In dit verslag zullen we de effecten van ruis in data dat gebruikt wordt voor het trainen van neurale netwerken analyzeren, met het doel te verifiëren of voorbewerken nodig is of niet.

Table of Contents

Abstract ... iii

CHAPTER 1 Introduction ... 1

1.1 Pattern Classification and Recognition ... 1
1.2 Neural Networks ... 3
1.2.1 Neuron ... 3
1.2.2 Multi Layer Perceptron ... 3
1.2.3 Learning ... 4
1.3 Neural Classification ... 5
1.4 Problems and Questions ... 5
1.5 Overview Thesis ... 6

CHAPTER 2 Noise in the Targets ... 7

2.1 Mislabelling ... 7
2.2 Effects of Mislabelling on Neural Networks ... 8
2.3 Experiments ... 8
2.3.1 Experiment Structure ... 9
2.3.2 Interpretation of Figures ... 10
2.4 Results ... 11
2.4.1 High Performance Data (Dutch License Plates) ... 11
2.4.2 Difficult Data (Portuguese License Plates) ... 13
2.4.3 Small Data Sets ... 14
2.4.4 Artificial Verification Data Sets ... 15
2.5 Confidence ... 18
2.6 Conclusions ... 19

CHAPTER 3 Noise in the Inputs ... 21

3.1 Noise in Inputs ... 21
3.2 Effects of Noise in Inputs ... 22
3.3 Types of Noise ... 22
3.3.1 Structured Distortions ... 23
3.3.2 Random Distortions ... 23
3.3.3 Outliers ... 23
3.4 Experiments with Filtering ... 24
3.4.1 Filtering Noise by Type ... 24
3.4.2 Filtering on Difference to Average Vector ... 24
3.4.3 Manual Data Selection ... 26
3.4.4 Combinations ... 28
3.5 Experiments with Noise Injection ... 28
3.5.1 Noise Injection ... 29
3.6 Conclusions ... 29

CHAPTER 4 Conclusions ... 31

4.1 Noise at the Output ... 31
4.2 Noise at the Input ... 31
4.3 Further Research ... 31

References ... 33

APPENDIX A Data Sets 35

A.1 License Plates 35
A.1.1 Dutch 35
A.1.2 Portuguese License Plates 35
A.2 Elena Datasets 36
A.2.1 Iris 36
A.2.2 Concentric 36
A.2.3 Clouds 37
A.2.4 Gauss 37

1 Introduction

When training neural networks for practical purposes (explained later in this chapter), there is always some form of noise involved. Sometimes it is bothersome. Sometimes it is beneficial.

In neural applications, injecting noise into the inputs of training-data can improve performance, for it may enhance the generalization ability of neural networks ([15], [11], [4]). In contrast, it became apparent in recent studies, that reducing noise, like removing input patterns of occluded images from the training-data, actually improved performance ([1], [3]).

There are many types of noise, and the effects they have on neural networks vary greatly. When new data is obtained, often much pre-processing is done to verify whether the data is suitable for training, and to remove noise. In this thesis we will analyse effects of noise in data used for training neural classifiers, so an estimate can be made whether pre-processing is necessary or not.

This chapter will give short introductions to Pattern Classification, Neural Networks and Neural Classification. Then we will iterate some of the questions that we have asked ourselves and which of these we want to answer in this thesis. Finally an overview of the thesis is given.

1.1 Pattern Classification and Recognition

An example of the process of classification is the classification of fruit in several types, for instance 'apple', 'banana' and 'orange'. We could weigh each piece of fruit and make a digital photo of it to determine colour and shape. We could then classify fruit by using these features, perhaps after consulting with a grocer for validity.

Pattern Classification in general is the science of applying a label or description to a certain measurement. Usually, these labels are chosen from a fixed number of possibilities (classes).

In the field of classification, a measurement is called a pattern. This is often a vector of numerical values, which correspond to a set of selected properties (features) of an object. Each element of such a vector indicates a degree in which the object has a certain property. Two examples are given in Figure 1-1.

Features can be any kind of property of a subject that is observable and measurable, preferably through an automated process. They can be numerical (e.g. weight or number of wheels), symbolic (e.g. colour or shape) or some complex calculation resulting from a feature extraction algorithm, which extracts features from measurements automatically.

These features have to be carefully selected, as they should make it possible for a classifier to assign a subject to a certain class. Incorrectly chosen features may result in errors or misguided classifiers. Usually experts determine features, or they are determined by analysing how experts make their decisions.

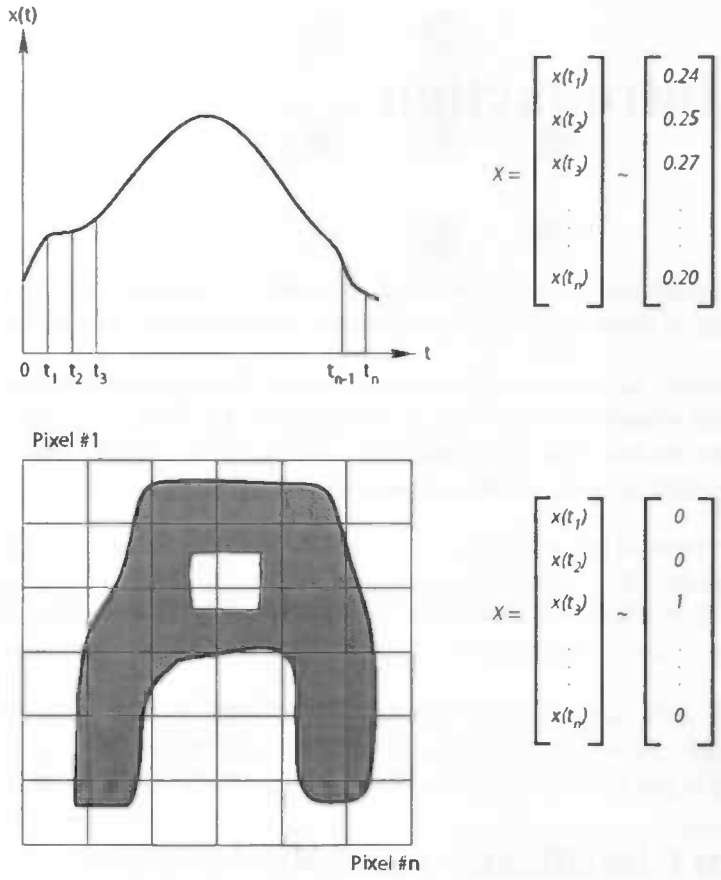

Figure 1-1. Examples of measurements to pattern conversion.

The numerous methods of classification can be categorised as statistical or syntactic approaches. A statistical pattern classifier assumes there is a statistical basis, for instance the proportions between the classes, on which it can make its decisions. Syntactic pattern classification uses relationships between properties (features) of patterns to make a classification, for instance combining shape and colour.

Black box approaches are a relatively new subcategory of statistical methods and emerged with the development of neural network technology. Classifiers in this category are seen as a black box, which means that the exact working cannot be defined by looking at the inner mechanisms, but only by analysing their input-output behaviour. As this thesis is about neural classification, we will explain neural classifiers further in the following two sections.

Neural networks belong to the black box classifier category because they are essentially a complex set of non-linear equations, to which no specific meaning can be assigned after training. It is not possible to give an expected outcome to a selected range of inputs; one can only try an input pattern and see what the network will give as output.

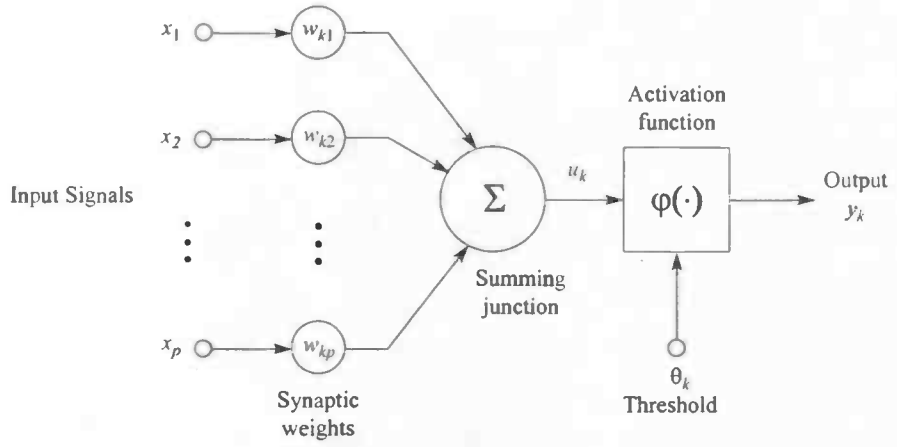

Figure 1-2. A non-linear representation of a neuron, used as a model for most neurons in artificial neural networks.

1.2 Neural Networks

A Neural Network is loosely based on the human brain. There are several types of neural networks, but since we will only use one type, the Multi Layer Perceptron, we will only give a short explanation of this type and leave the rest for further reading ([5]).

1.2.1 Neuron

The basis of a neural network is the neuron. The model of a neuron we use for neural networks is called a perceptron. It is a mathematical, simplified representation of what is known today of neurons in the human brain.

As seen in Figure 1-2, a neuron has a set of one or more input signals. Numerical values of a pattern, or outputs of other neurons, are sent into the neuron here. Each value will be multiplied by a weight value per connection, after which the results of these multiplications will be summed. The output of a neuron is this sum, applied to an activation function that limits the output to a certain interval (usually [0, 1], as with a Sigmoid-function).

The power of neurons lies in the weights of the connections. They can be configured (by training), so that a pattern, applied to the neuron, results in meaningful and usable output. This configuring is further explained in Section 1.2.3 below.

1.2.2 Multi Layer Perceptron

A neural network is a structured group of neurons that are connected to each other, to fulfil a certain task. These neurons are usually grouped in layers, where the outputs of neurons of one layer are used as inputs for the neurons in the next layer.

A Multi Layer Perceptron (MLP) consists of three types of layers. First, there is an input layer. This layer consists of nodes that send the input values of a pattern to appropriate neurons (in the case of a MLP usually to all neurons in the following layer, as seen in Figure 1-3). Sometimes, this layer is not explicitly named, because no computation is done here.

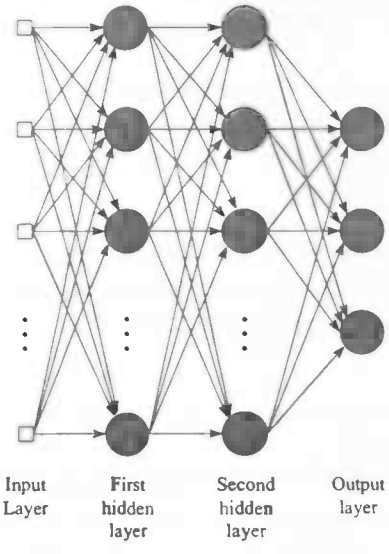

Figure 1-3. A graph representation of a fully interconnected multi layer Perceptron.

The last layer is the output layer. These neurons should give output that can be used. In classification there is usually one neuron per class and the neuron with the highest output determines which class the network thinks the input belongs to. Other layouts are also possible. If a neural network is trained for function approximation, for instance, there usually is one output neuron, which gives the approximation of a function applied to the input.

Between the input and output layer, there are one or more hidden layers. These neurons cannot be seen from outside the network, if viewed as a black box model, hence the name "hidden". The hidden layer adds an extra dimension to the network's capacity for learning, allowing it to generalize significantly better about the learning examples given (see the following section). With generalization, we mean that the neural network can give reasonable output on input patterns it has not been trained with (i.e. has not seen before). This feature is one of the strongest points of neural networks.

1.2.3 Learning

A neural network learns, by adjusting the weights of the connections between neurons with a learning algorithm. Such an algorithm describes how the properties of a network should be adapted, when it is given a data set of input- and output-patterns pairs on which it should train. The output patterns of such data sets have been given the desired output the network should return when given the corresponding input-pattern.

A MLP uses a learning algorithm called Error Back-Propagation, which means, in simplified terms, it sends input patterns through the network and compares the values given by the output layer with the supplied output patterns. The differences between the result and the output pattern, the error, is sent back through the network and used to adjust the weights according to the size of the error, in the hope that the error in the next run will become smaller. This process will be repeated until the error is acceptably small.

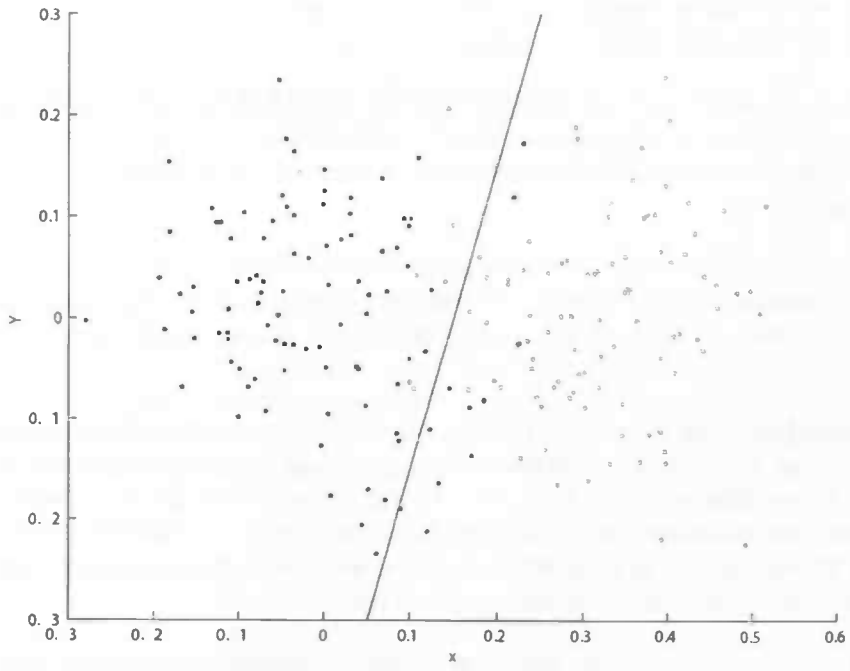

Figure 1-4. An example of a linear decision boundary, creating two decision regions.

1.3 Neural Classification

In concept, the task of classification revolves around finding class-labelled decision regions which partition the domain space in such a way that a maximum number of patterns are classified as belonging to their correct class. Each type of classifier does this in its own way, but just by looking at the input-output behaviour, the effect looks the same. A simple 2D example with a linear decision boundary is given in Figure 1-4.

Neural networks can create multiple non-linear decision boundaries based on probability density, which allows them to classify better, since they can fit the boundaries more precise. As classification can easily be taught by example, neural networks seem very well suited for this task. The generalization property of neural networks confirms this. Previous literature studies have shown that neural networks indeed perform very well on classification tasks ([17], [12]).

One of the drawbacks of neural networks in general is the high level of expertise they require. There are numerous design aspects that have to be chosen well, or the network will not learn a given task. A second disadvantage is themir complexity. Usually, if something goes wrong, the problem at hand cannot be pinpointed to a certain aspect of the network, which makes it difficult to correct such errors. Furthermore, a network that is fully trained cannot easily be described in terms of which element of the network contributes what ability.

1.4 Problems and Questions

There are many design issues when dealing with neural networks. When creating a new neural classifier, the performance is almost never as good as it is required to be, so much tuning is often needed. One type of optimising the performance of a neural network is improving the quality of the

train-data, which can be done in many ways. These optimisations are usually some solution to a problem in the data, which has been detected by data analysis.

For instance, the performance of a neural classifier can degrade severely when the number of patterns per class differs. A class with a relatively small number of patterns will be hard to classify, but will also make it harder to classify other classes. A study of this problem and some countermeasures are given in [8].

In this thesis, we ask ourselves how noise in the data used for training affects a neural classifier, and particularly whether (costly) filtering and cleaning techniques should be applied or whether they are not necessary. Noise is the general term for disturbances and corruption in data, like errors or missing values.

An example of one type of noise are outliers. Outliers are patterns that belong to a certain class, but have values that differ greatly from the average pattern in that class, sometimes the difference is so great, that it even falls outside of the scope of the classifier. There are many causes for outliers; they may be the result of unusual circumstances during sampling, or can be inherent to the subject of the classifier. These outliers can misguide a neural network severely, so it may be beneficial to remove them. More information can be found in [6] and [14].

Another example is missing data. Sometimes, data sets contain patterns that are incomplete, or have missing data. This means there are values missing in a single pattern, or there are patterns missing altogether that are required for proper learning. This could be a result from a sensor, or the even the entire data-source, failing during some time of the sampling period. Methods for dealing with this problem can be found in [10] and [16].

There are many more types of noise, and perhaps even more methods of dealing with them. To answer the question whether it is useful to implement these methods, we need to know in what amount noise affects the performance of neural classifiers.

As mentioned above, we have limited ourselves to optimisations related to noise in data rather than optimising the structure of the networks or learning parameters.

1.5 Overview Thesis

We have divided noise into two types, and analysed how resilient neural classifiers were to these. First, we studied noise at the output of a network. This means that a pattern has been labelled incorrectly, such as labelling an 'A' as a 'B'. We will discuss this further in Chapter 2.

Second, there is noise at the input of a network. This means that there are irregularities in the input patterns of the data. In classification, there are several types of noise at the input, like occlusion and data corruption. This will be further discussed in Chapter 3.

Finally, in Chapter 4, we will give a summary of our findings and suggestions for further research.

2 Noise in the Targets

2.1 Mislabelling

When designing a neural network for classification, the output of such a network consists of several output neurons, one for each class, as explained in Chapter 1. In this chapter, we will empirically study the effects of noise at the output-side of the network, by adding increasing amounts of noise to data sets used for learning and comparing the results of each set.

A data set used for training a neural classifier is in fact a list of patterns. These patterns are pairs of vectors, one with input- and one with desired output-values, also called target-values. The target-vector is filled with low values (typically `0.05`), except at the position corresponding to the class the pattern belongs to, where a high value has been placed by a supervisor (typically `0.95`). Adding noise means that we change the position of the high value, effectively assigning the pattern to a wrong class.

Noise at the output of training-data is in practice not some random form of corruption, but a specific change in the values. Labels with values other than high and low, are rarely used and even then have fixed values, so corruption of these values is easily detected and dealt with. The only possible form of noise at the output of data used for classification, that might not be detected, is that patterns are assigned to a wrong class by mislabelling.

In many applications of classification, a human supervisor assigns the output-labels to patterns, after analysing the input values. In these cases, the supervisor causes noise when he or she makes a mistake (e.g. a typo) and mislabels the pattern.

In our experiments we have assumed this type of error to be uniformly distributed, since there are many causes for them, and these do not follow a structured pattern. Some errors, as when the supervisor mistakes a '1' (one) for a 'l' (lower case L), might be more confusing for a neural classifier then others, but in practice these are just as common as a typo which changes a 'g' into an 'h' (the key next to it). Usually, human supervisors label 1.5% to 2% of the available patterns incorrectly.

Correct:	Error:	Typo:
TL-96-TR	TL-96-IR	TL-86-TR

Figure 2-1. Examples of errors made by supervisors

In other applications labelling has been done automatically, which often results in a larger amount of mislabelled patterns. An example is the Global Land Cover Classification data set, in which experts have found 11732 mislabelled instances of the total of 37340 patterns (31%) [3]. Mislabelling errors created by these systems are not uniformly distributed per se, since these systems are

(crude) classifiers themselves, and may cause strongly structured errors of a more complex nature than just the '1' and 'l's. Therefore, the results of this study may not be relevant to all classifiers trained with automatically labelled data sets.

2.2 Effects of Mislabelling on Neural Networks

The main question we are trying to answer here is whether it is really necessary to put much time and effort into removing mislabelled patterns, because they degrade classification performance, or that they don't misguide neural classifiers at all, and noise can just as easily be ignored.

Looking at the learning algorithm, we see that mislabelled patterns will cause the network to adapt incorrectly. The weights in the network are adapted in the backward pass of the back-propagation algorithm according to the size of the error, which is the difference between the target value and output of the network. The size of this adaptation will potentially be very large, since the output of the network should be low because of training, where the target is set high erroneously.

The back-propagation algorithm has several countermeasures against these abrupt and large changes, like the learning-rate parameter and momentum term parameter, which can inhibit the speed of learning if they are set low. Therefore we expect that a certain amount of errors should be dampened enough so that they do not impair the overall performance of a neural classifier. But when the amount of noise becomes large, and the balance of the classes is heavily changed, performance will degrade surely.

However, this degradation does not have to cause a proportional increase in the error rate of the classifier. Since the outcome of the network for each pattern is determined by a comparison of all the output values (usually in a winner-takes-all algorithm), a decrease of the value of the correct output neuron and an increase of the values of the wrong outputs, does not necessarily result in a higher value of one of the wrong outputs than the right one. In what measure this will happen depends largely on the type and quality of data sets used.

Quinlan [13] showed that introducing noise at the output of training data of decision tree classifiers caused an error of 2.5% to 50% when introducing noise amounts from 5% to 100%, where 100% noise caused the tree to give random answers. Much research has been done, based on his work, on methods to remove mislabelled patterns from data sets to improve classification results ([2], [3], [9]). Since neural networks operate differently from decision trees ([5], [13]), some empirical experiments will show whether this research will benefit neural classifiers as well, or that neural networks are more resistant to noise and do not need filtering and cleansing techniques.

2.3 Experiments

To analyse the effects of noise at the output of a network, we have conducted experiments with several different data sets, comparing results of networks trained with these sets to the results of networks trained with the data sets with increasing amounts of noise applied.

If noise has an profound impact on the performace of the classifier, we expect that the train error will converge slower and more unstable that the error of the network trained with the clean set, and that the test error will be higher. Also, the train error will drop below the amount of noise present in the data set, as shown in Figure 2-2, since mislabelled patterns will be learned as the incorrect class.

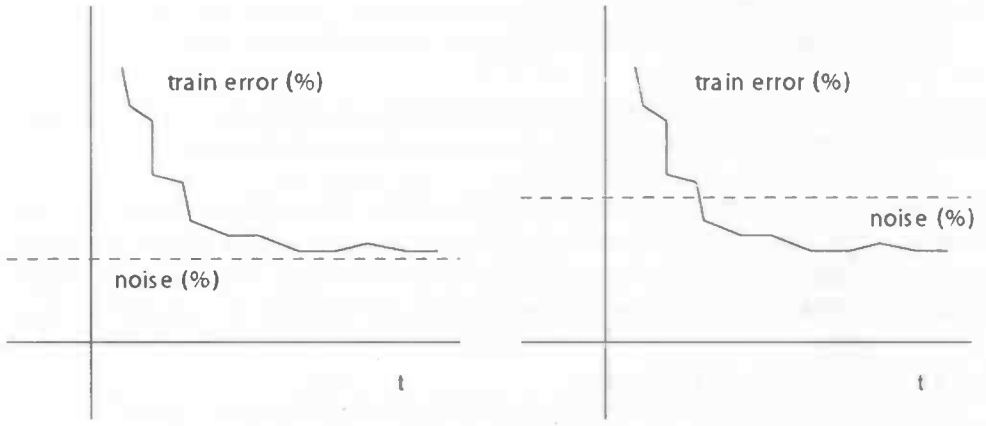

Figure 2-2. When the train error converges to an amount close to the amount of introduced mislabelled patterns (left), this indicates that the network can recognize the mislabelled patterns as errors and does not learn them. When the error falls below this amount (right) the network starts learning mislabelled patterns, degrading performance.

When noise does not interfere with the performance, the error will converge also slower and more irregular than without noise, but to a lesser degree. The final train error rate will converge to the amount of noise present plus the final error of the network trained with the clean data set, since not many mislabelled patterns will be incorrectly learned and most will be errors (they are assigned to their correct class, but deu to the mislabbeling that is an error). The test error will remain close to the test error of the cleanly trained network.

The first experiment we did, the Dutch License Plate data set, made it clear that noise at the output does not have to be a problem at all. Since the performance of neural networks trained with this data set is high, we wanted to test whether weaker or more difficult sets were more susceptible to noise. We tested a difficult data set, the Portuguese data set, a smaller version of the Dutch License Plate set, and several of the benchmark data sets from the ELENA project ([7]).

2.3.1 Experiment Structure

The experiments consisted of the following steps:

1. The data is divided in two sets. Sixty percent of the original data set is used for training the network, and forty percent for testing the network after training. The balance between the classes is kept the same both sets.
2. The train-set is duplicated five times, with increasing amounts of uniformly distributed noise applied (5, 15, 30, 60 and 80 percent). The test-set remains unchanged, since we are only interested in effects of noise on the training process.
3. Each of these train-sets is run through five newly created, randomly initialized networks, with the following learning parameters:

Learning rate:	0.6
Momentum:	0.2
Hidden neurons:	15 (one layer)
Epochs:	100

4. The results of these learning sessions are gathered, averaged and plotted. A detailed description and explanation of these plots is given in the following section.

The primary goal of these experiments is to determine an estimation of the amount of noise that would interfere significantly with the performance of a neural classifier.

One might notice that we used the same settings for the learning parameters for each experiment, even though the data sets used have very different properties. This means that some experiments might attain higher performance with optimisation. Preliminary tests did not show any changes in the effects of noise when changing these parameters, so we have disregarded these in this study.

2.3.2 Interpretation of Figures

All figures used to illustrate the results of the experiments have three plots, each with graphs for each percentage of mislabelling applied. We will give a general description of these plots here. Note that the values shown in the figures are the averaged results of five networks trained per train-set.

Average Train Errors

The first plot is the combination of average train-errors per train-set. Each epoch, the train-set is used to test the neural classifier resulting in a percentage of errors it has made.

The characteristic of the train-error is used to get an indication of the quality of the learning process and the potential left for learning. In Figure 2-3, in the next section, we see an example of steep and steady convergence, indicating a stable learning process. In contrast, Figure 2-4 shows a much more irregular and slow convergence, indicating that the train-data is not easily learnable.

The difference, or gap, between the train-error and the percentage of mislabelled patterns indicates the amount of mislabelled patterns that are erroneously learned by the classifier, see Figure 2-2. When the error falls below the percentage of mislabelled patterns, we know that the train-data was not clear or cohesive enough to allow the classifier to learn the difference between real data and errors. When this happens, performance on the test set will surely decrease.

If, on the other hand, the gap is small, this usually indicates that the network did not suffer from the mislabelled patterns enough to truly misguide it. The last plot, explained later in this section, verifies this.

As a side effect, these train errors give more information about the performance the network has on the data set in question, but this will be discussed further in Section 4.3.

Average Test Errors

The average test errors show the performance of the classifier on data it has not been trained with. This error rate gives a more reliable indication of the actual performance when the classifier is put at work in the field it was designed for, since it tests the generalization feature of neural networks, i.e. its reaction to unknown data.

It is important to remember that the test data has not been corrupted. The mislabelling percentages in the legend are only linked to the train data. The test errors are only caused by faults created in the training process.

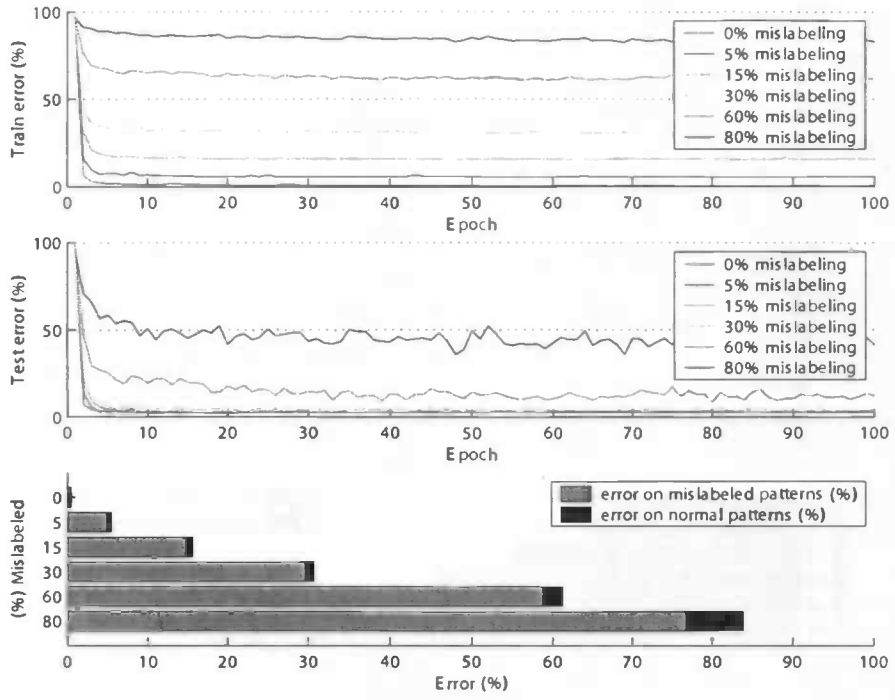

Figure 2-3. Dutch Licence Plates: The train-error increases at an approximately linear rate relative to the percentage of mislabelled patterns, while the test error remains low. The network only learns a small amount of mislabelled patterns, the impact of noise very small.

The average test errors are the most important factor in getting an idea of how much the added noise will influence a neural classifier in a negative way. If the test error increases while increasing the amount of noise, noise clearly degrades performance.

Mislabelled Patterns Learned

The last plot shows the average composition of the train errors at the end of the learning process, separating mislabelled and normal patterns. As mentioned above, when the total train error is close to the percentage of mislabelled patterns, this could indicate that the network has not been misguided by the corrupted data. This assumption can only hold if the train error is indeed mostly caused by the mislabelled patterns. This can be verified with the third plot. The ideal situation is that all mislabelled patterns are detected as errors, and the error on normal patterns is not increased, which would mean that the added noise had no effect whatsoever.

2.4 Results

2.4.1 High Performance Data (Dutch License Plates)

Recognition of license plates is a field of study where much pre-processing is done to prepare data for training. The first data set we used for testing is a set of features per character extracted from images of licence plates. This set has already been cleansed of most noise and has been prepared carefully, so performance is expected to be high. More details of this set can be found in Appendix A.1.

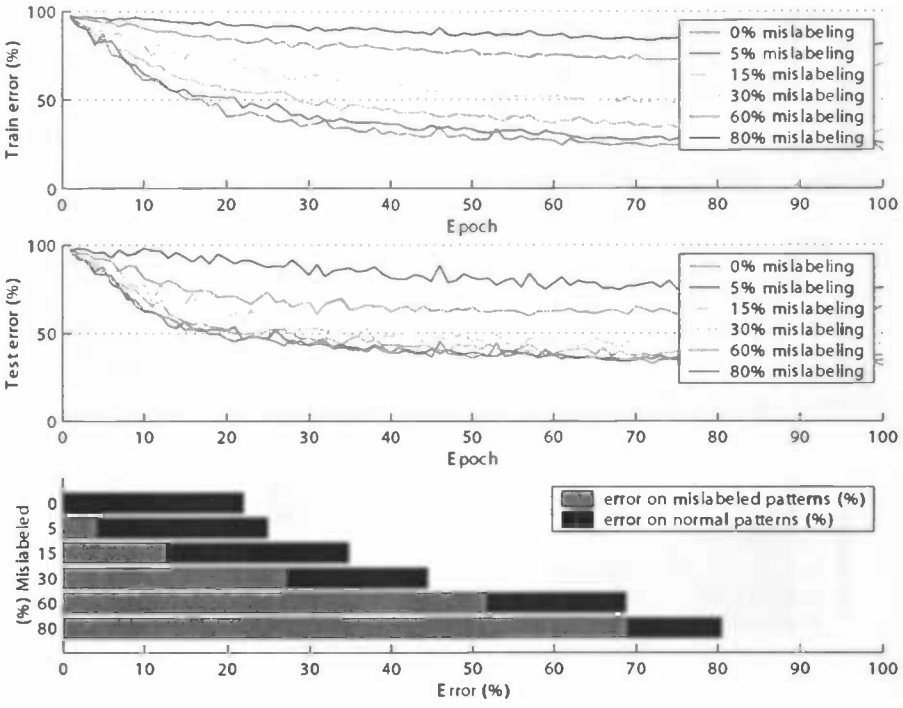

Figure 2-4. Portuguese License Plates: Both error rates remain high and converge slow and in an unstable manner. Furthermore, a large part of the mislabelled patterns is learned erroneously. The network does not learn the data set well enough.

Examining the different train errors, we see that they all converge very fast and remain stable throughout the training process. The distance between the train errors is proportional to the difference in the percentages of mislabelled patterns. Only the train errors on the sets with sixty and eight percent mislabelled patterns are slightly higher in comparison.

The test results show that the network has been trained extraordinary well. Up to fifteen percent mislabelled patterns doesn't really make a difference (0.16% and 0.70%) and thirty percent only in the slightest (1.28%). Sixty and eighty percent noise does have an impact (9.65% and 43.86%), but certainly less than one might expect.

The third chart shows that very few mislabelled patterns are learned, and that the error contains almost all of the mislabelled patterns. There is only a relative small amount corruption in the sets with a high amount of mislabelled patterns.

Combining the separate results, tells us that the neural network easily classifies this data set and a small amount of noise at the output hardly affects the classification performance.

Since we knew beforehand that the Dutch License Plate data set could be classified well by a neural network, the results were not surprising. To gain a perspective on which characteristics show the performance and stability of the classification we tested some sets that are known to be harder to classify for several different reasons, like amount of patterns, number of classes and general difficulty.

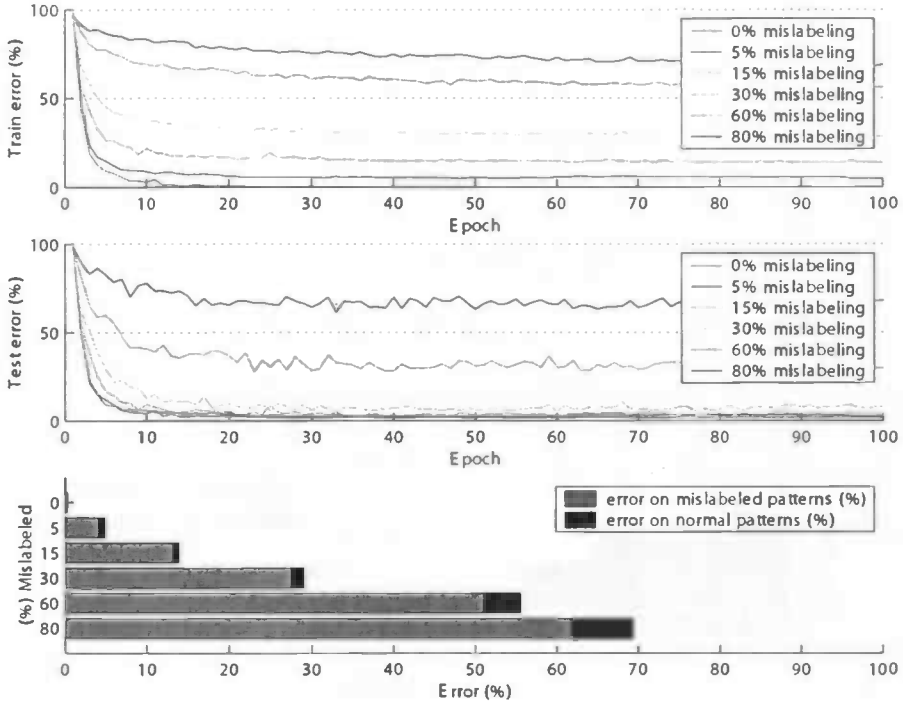

Figure 2-5. Dutch License Plates, less data: With five times less data, convergence takes longer, is less stable and results in more mislabelled patterns learned. However, this does not have a comparatively heavy impact on the final train error on normal patterns and the test error.

2.4.2 Difficult Data (Portuguese License Plates)

The Portuguese License Plate set has the same general characteristics as the Dutch version. However, there are two differences that make training considerably harder. First, the Portuguese set was only a fifth in size compared to the Dutch set, which means the neural network has less examples to learn the given task by. Even worse, the original images of the license plates were compressed to a very low quality, degrading learnability severely. Details of the set can be found in Appendix A.1.2.

The results in Figure 2-4 show, that neural classification of an acceptable form is impossible in this case. The train error decreases too slow and fairly unstable, and does not converge to an acceptable point. The test error shows the same characteristics, but the convergence point is even higher. The effects of noise are more profound here too, already with fifteen percent mislabelled patterns we see an increase in the test error compared to the clean set. From the last plot, we can tell that the neural network has erroneously learned a growing amount of mislabelled patterns with each increment of noise.

Even though the results are truly bad, noise does not make the results that much worse. The effects on performance are not a lot more noticeable than with a data set with a high learnability. Also, these effects might partially be caused by the relative small amount of data available (which is tested in the following section).

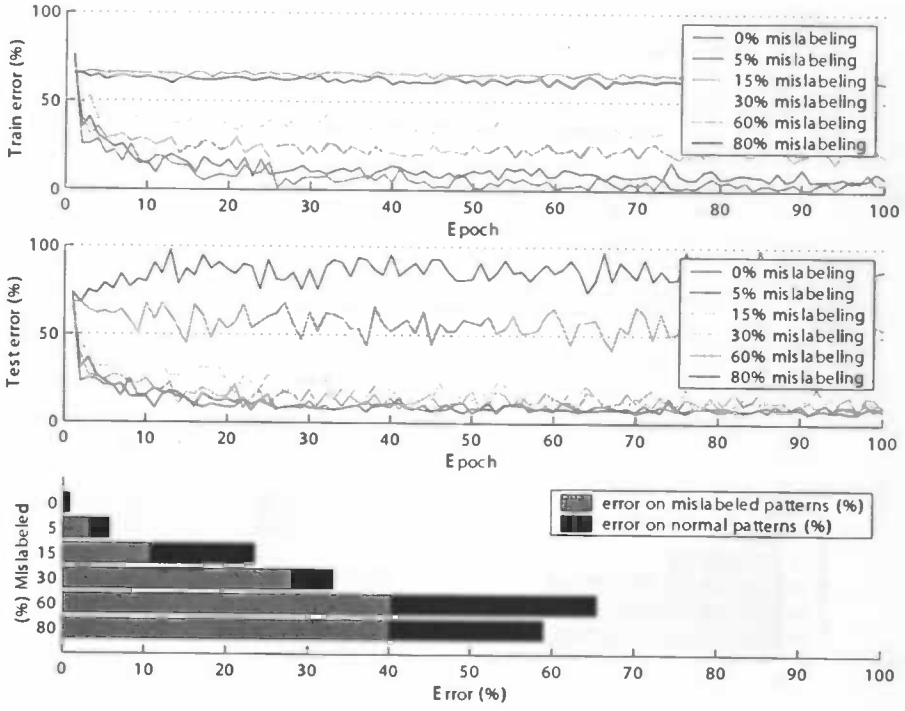

Figure 2-6. Iris data set: Noise is more disruptive with a small data set, but its effects are still relatively small.

2.4.3 Small Data Sets

Dutch License Plates, Small

As the Portuguese set had significantly fewer patterns besides the low quality of source material, a comparative test with the Dutch set could help analyse more precisely whether a small amount of data would make a neural network more susceptible to noise and if so, to what extent.

Figure 2-5 shows results of training a neural network with one fifth of the Dutch License Plate data used. We see that the network has converged later, and that learning was slightly less stable. The third plot also shows that the network has learned noticeably more mislabelled patterns.

Comparing the test errors with those of Figure 2-3 we see that this network is slightly more sensitive to noise than that of the first test. This was to be expected, but the effect is rather small seeing that we only used a fifth of the original data.

Iris Data Set

As a second test, we used the iris data set, described in [7]. Some of the details are repeated in Appendix A.2.1. This set is often used as a benchmark for training with small sets. Our results on the clean set are comparable to those shown in [7] (page 95, Figure 6.4). Introducing noise impacted performance more than with the small Dutch License Plate set, but the effects stay small at the lower mislabelling percentages, as seen in Figure 2-6.

We can conclude, that having a small amount of patterns in a data set makes the neural network more susceptible to noise at the output, probably because there are less patterns to restore the

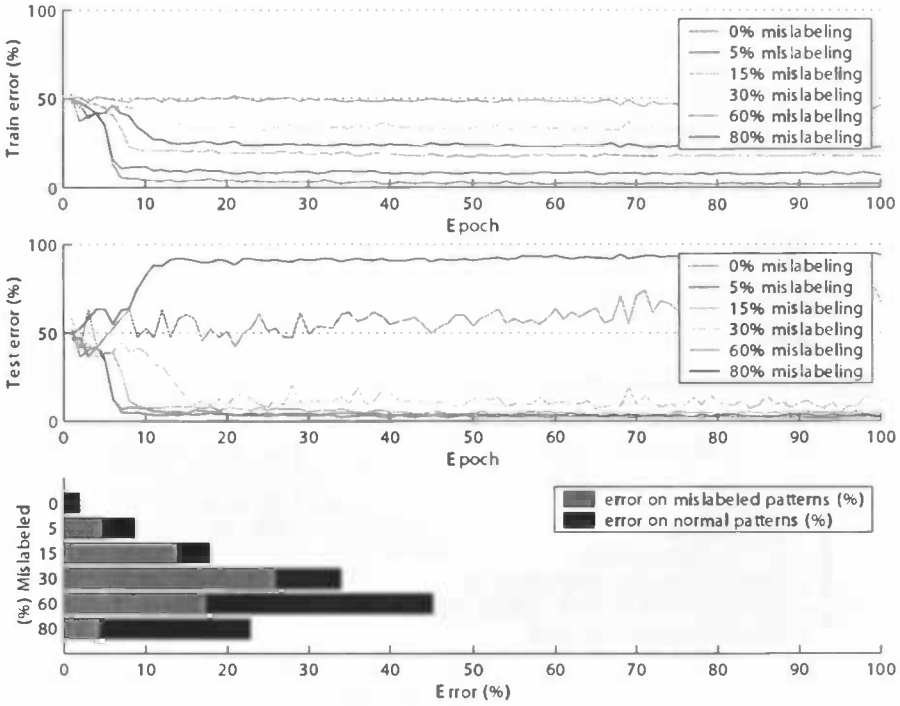

Figure 2-7. Concentric data set: As this is a two-class problem, mislabelling causes more problems, as it inverts the patterns of the data set. Still, the effect is still small up to fifteen percent mislabelling.

weights to their correct values. This decrease in performance is not proportional to the size of the set however, and introduces only a small increase of corruption at the lower noise percentages.

2.4.4 Artificial Verification Data Sets

We selected four more data sets from [7], namely Concentric, Clouds, Gauss 2D and Gauss 8D. Brief descriptions are found in Appendix A.2. All of these sets have two classes, which means that mislabelling will in essence invert patterns. Adding fifty percent of noise or more will result in a lower train error rate because the network will start to learn the inverted set. The test error will of course rise further, since the test set is not changed. The performance of our classifiers trained with the clean train sets was very close to those given in [7].

Concentric

The Concentric data set has two classes. One class consists of patterns in a circle and the other of a ring surrounding the patterns of the first class. There is no overlap between the classes. Again, we see in Figure 2-7 that noise hardly has an impact on performance at the lower percentages of mislabelling. The effects are stronger at the higher percentages though. One might assume that mislabelling in a two-class data set is more confusing than with more classes. Still, it seems that the classifier can absorb some amount of errors and ignore them.

Clouds

The Clouds data set has the same structure as the Concentric set, but the first class is a cloud of patterns, and the second has three separate clouds. There is lot more overlap between the classes, which

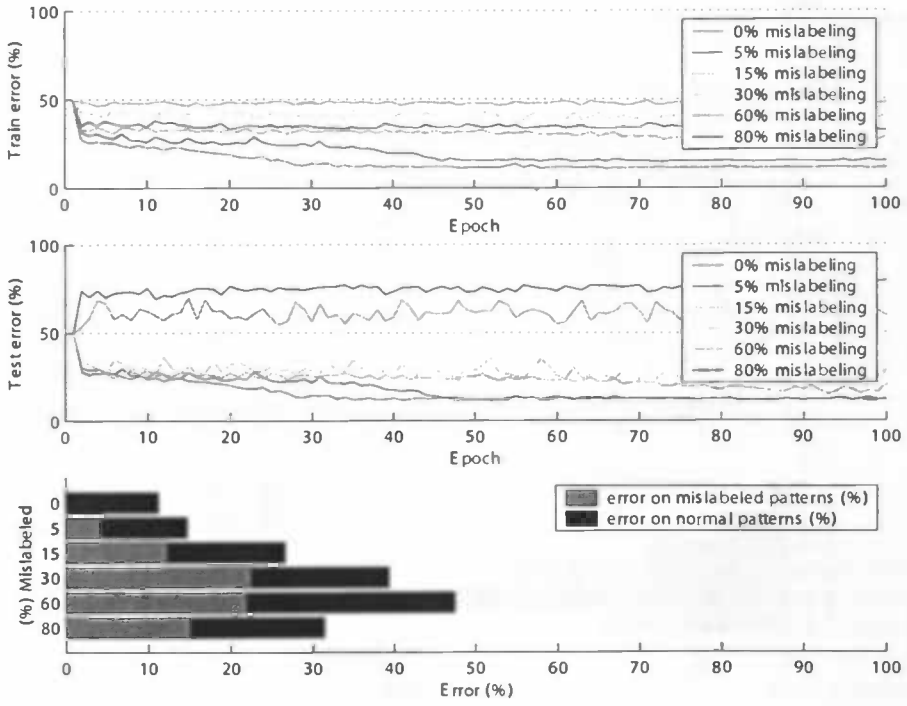

Figure 2-8. Clouds data set: As this is a two-class data set and also hard to learn, effects of noise are stronger. At the lower mislabelling percentages, we can clearly see that training longer reduces part of the error caused by noise.

makes learning certain parts very difficult (if not impossible). From the results, shown in Figure 2-8, we can see very clearly that the effects of noise can be reduced, or even removed altogether, by training longer. The negative impact of noise is also stronger here, compared to the Concentric set, indicating that noise added to a set which is already hard to learn, makes learning even more difficult.

Gauss 2D and 8D

The Gauss data sets are used to test the effect of adding more input dimensions to data sets. We have selected the two most extreme sets of the seven available. They all have two classes, which are fully overlapped. Adding extra input dimensions spreads the patterns over a larger area, making the classes easier to separate, and thus easier to classify. An issue of the test is, that there are not enough patterns, when there are more than five input dimensions.

Comparing the two sets, we see that the 8D set performs best, as expected [7], and also converges earlier in the training process. Furthermore, we see that the fact that there are actually too few patterns available for the 8D data set (as explained in [7]), results in a stronger effect of noise on the 8D set, especially when adding more than fifty percent noise. This is caused by the fact that there is a larger chance that the mislabelled patterns will make the neural network assign relatively empty areas of the domain space to the wrong class. Apparently, this effect is small enough at the lower mislabelling rates that it does not interfere with the increase of performance due to the reduced overlapping. Only at 30% mislabelled data and higher, the effect is strongly noticeable.

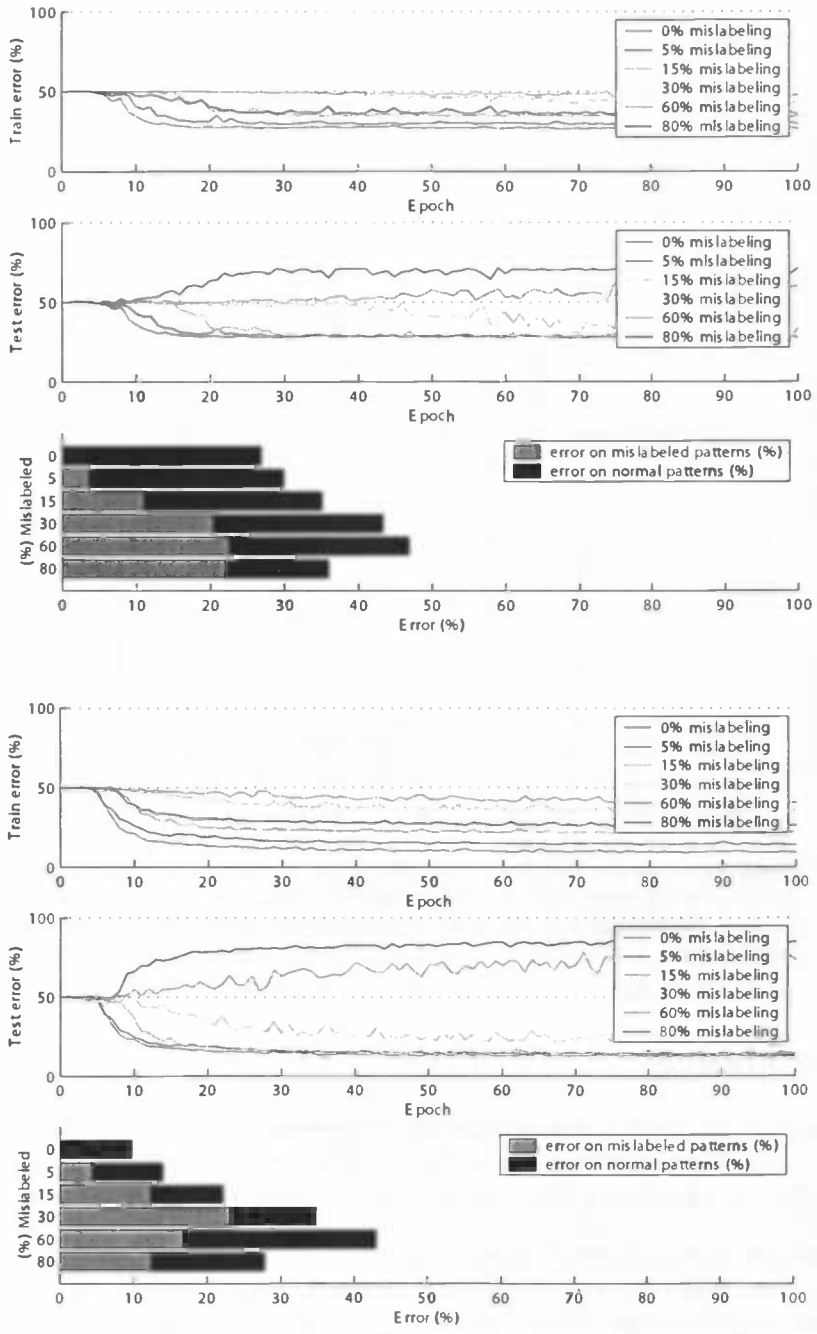

Figure 2-9. Gauss data sets 2D and 8D: Increasing the number of input dimensions results in a lower error rate, since the classes are spread out over a much larger area, making it easier to separate the classes. An effect of insufficient data for the 8D set only occurs at thirty percent mislabelling and higher. There the error is relatively higher than the clean set, than with those of the 2D set.

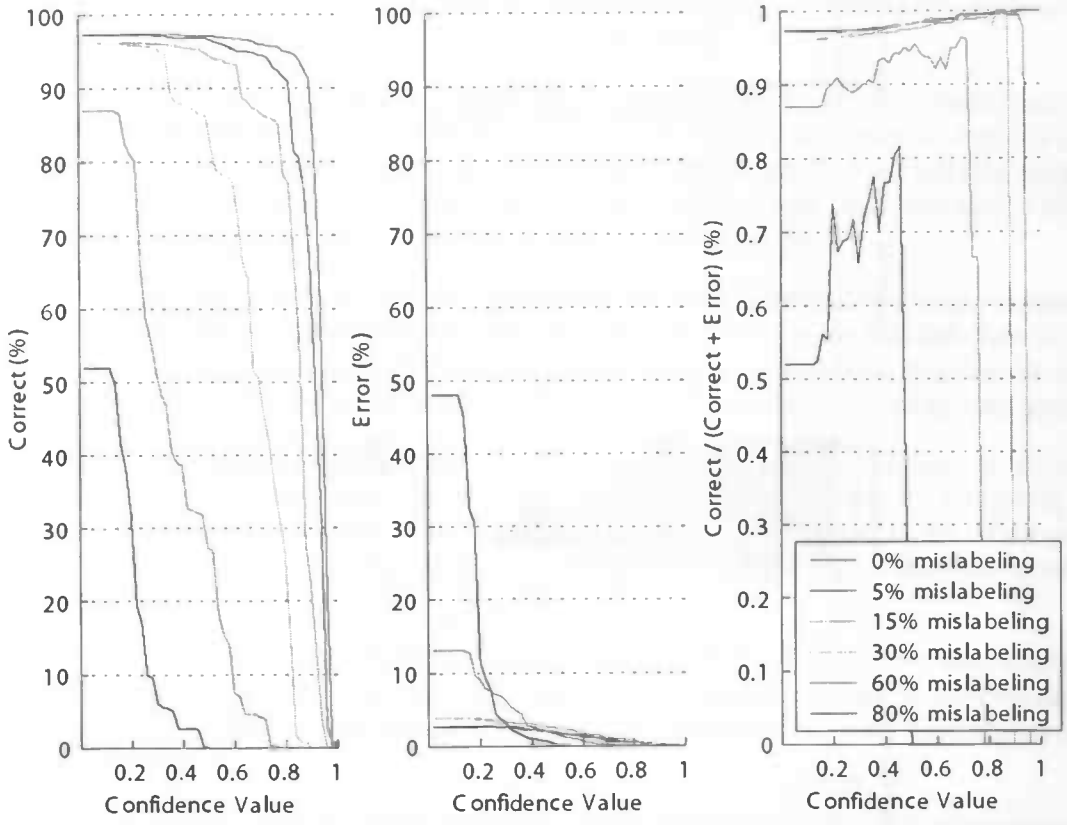

Figure 2-10. Confidence on the Dutch License Plate set: A clear decrease in confidence when increasing the amount of noise is visible here, but the chance on a correct pattern when the confidence level of the classifier is above the set confidence value remains high with most mislabelling percentages.

2.5 Confidence

As mentioned in Section 2.2, the confidence of the neural network will decrease with increasing amounts of noise. The confidence of a neural classifier on its input is the value of the output neuron with the highest score, ranging from zero to one, with zero indicating no confidence at all.

Several measurements and statistics can be generated about the performance of a neural classifier using this value. These all revolve on setting a confidence value to a certain level, requiring the output of the classifier to be higher than this value. When the output of the most activated neuron does not exceed this value, the pattern is rejected as being unknown.

Comparing the amount of rejects, errors, and correctly classified patterns while varying the confidence value, gives a better indication of the quality of the classifier than the train and test error plots alone. The latter ignore the confidence of the network and always accept its answer.

Figure 2-10 is an example of the above-mentioned comparisons, performed on the classifier trained on the Dutch License Plate set (Section 2.4.1). We incremented the confidence value from zero to one in fifty steps, each step counting the number of correctly classified patterns, number of errors and the chance on a correctly classified pattern when the confidence level would be above the set confidence value.

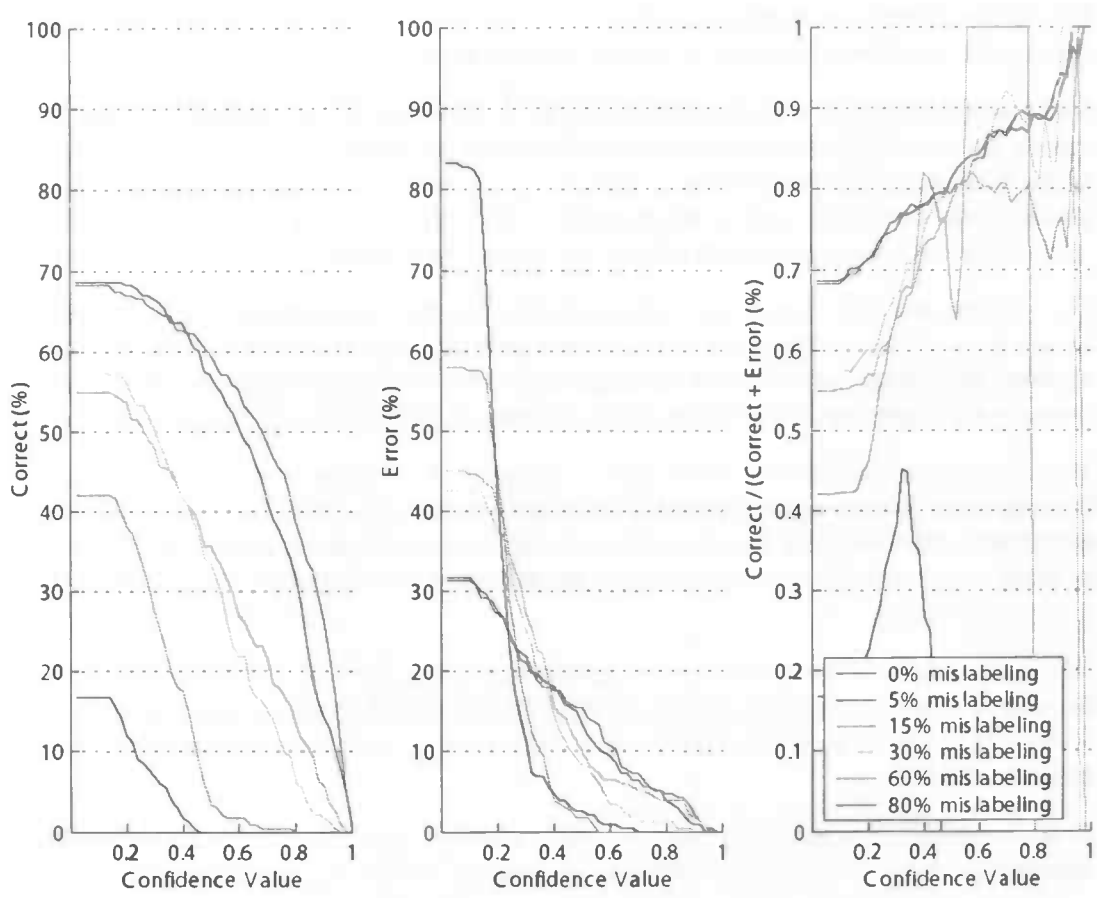

Figure 2-11. Confidence on the Portuguese License Plate set: The plots show about the same trends as those of the Dutch License Plate set in Figure 2-10, but at a much lower performance. The differences in trends are comparable to those of Figure 2-3 and Figure 2-4.

The plots show that the general confidence level of the network deteriorated with each increment of noise, but that the overall correctness of the classifier remained about the same for noise levels up to thirty percent.

Plots created for the other data sets all show about the same characteristics, but with deteriorations similar to those of the results seen in the previous sections. Figure 2-11 is an example to illustrate this. When the conditions for learning are worse, the confidence levels decrease more rapidly and the plot of the chance on a correct output becomes more erratic. No new conclusions could be drawn from them.

2.6 Conclusions

Having done all these experiments, the general conclusion is, that noise does not have as much an impact as might be expected.

On a data set with high learnability and plenty of training samples noise levels of up to 30% only cause a slight increase in the test error (0.17% to 1.28%). When the conditions for learning become worse, the effect of noise will also increase, but not as strong as the change of condition itself.

Neural classifiers have shown themselves to be very resistant to noise. The effects of noise certainly don't occur in a linear form as it does with decision trees ([13]).

Still, noise does decrease performance, and if it is important to obtain absolute maximum performance, the mislabelled patterns need to be filtered out. However, this is only a wise decision if there is plenty of data, since studies done in data cleansing have shown that filtering bad data samples is possible, but always at a cost to the correct data ([2], [9]). Certainly for small data sets, this might mean the there is not enough data left for the network to generalize well.

The experiments have shown, that adding noise causes the training process to converge slower and makes it more erratic. However, the test error rates of noise levels up to 30% all show a steady decrease towards the error rate of the clean set. Perhaps by adjusting the learning parameters, for instance with applying annealing, the effect of noise might altogether be removed in these instances.

When considering whether or not to apply a (expensive) cleansing process to a data set, we believe it is important to have some knowledge about the amount of noise in the data set. Removing noise is probably effective if the amount of noise is known to be large, say more than 5%. If the amount of noise is unknown or noise is not apparent, perhaps it is better to try other methods to increase performance first.

Moreover, if noise is inherent to the data gathering process, probably the same amount of noise will be present when the classifier is put to use. In these cases, it might be better to train the network with noise left in the train set, and then selecting a proper confidence value at which noisy patterns might be rejected.

In all, noise isn't necessarily an insurmountable problem. In most cases, it can be circumvented by using proper countermeasures available in the training process of neural classifiers.

3 Noise in the Inputs

3.1 Noise in Inputs

In the previous chapter we noted that noise at the output layer of a network was only possible in one form, namely mislabelling of patterns. In contrast, noise at the input layer of a network can take many forms.

In the typical system structure of a pattern classifier (Figure 3-1), many parts can be affected by noise, which may result in distortions in the input data. A sensor can malfunction or can be obstructed; there may be flaws in the measurement processing, or the feature extraction algorithm might be used in a wrong way.

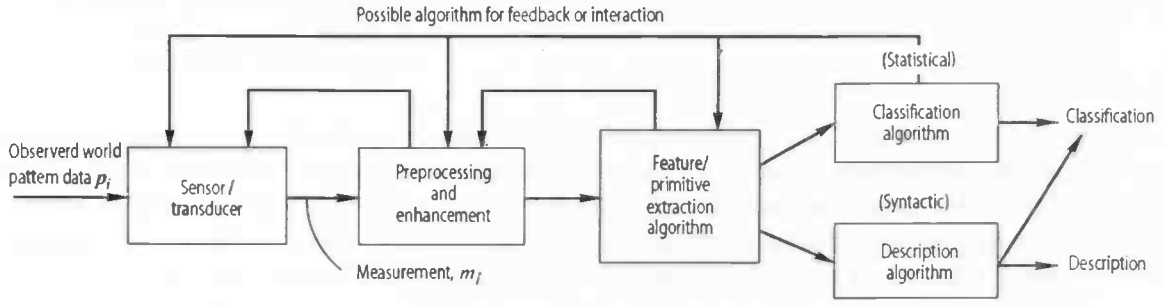

Figure 3-1. A structure diagram of a typical pattern classifier. Each of the structures can be affected by noise, which may corrupt data (see Figure 2. in [17]).

In the system structure, the observed world is the first element where noise can occur. More often then not, the environment is inherently susceptible to noise, so patterns may already contain noise at this point.

The next element, the sensor is usually a third party device, which might be prone to failure, misalignment or other forms of errors, which cannot always be dealt with by the user, thus causing additional noise in the input data.

Noise in the raw data (measurement m_i), should be dealt with by the pre-processing and enhancement structure. This structure will become extremely complex if it is to deal with all forms of noise in such a way that they will not affect the following structures. In practice, this is often too expensive to develop well. If bad measurements can be detected, they are usually either discarded, or they are sent through without change (perhaps with the argument that it might make the classifier more robust).

In contrast, there are many fields in which real data is not available or hard to come by, and data is generated. This replaces the first three system elements with a generation block. Since the real world is often very hard to model, generated data usually contains only a subset of patterns that can

occur in the real world. Noise is often omitted from these models and therefore does not occur in the generated data.

The Feature/primitive extraction algorithm also operates on the input data, and might also produce noise in the form of errors, but as these algorithms are fairly standard, and highly controllable by the user, we will ignore it in this study, since it will hardly ever produce noise in practice. The last parts of the classifier system don't alter the input data so we can disregard them.

We have noted that the amount of noise at the input is very problem dependant. Sometimes noise is abundant, in other cases there is no noise in the data at all. The way noise is handled in practice is very diverse as well. Sometimes it is eliminated, in other cases it is corrected, or it is just left as it is. Elimination and correction are often very time consuming and/or difficult tasks. The question we ask ourselves, is which of these strategies is best for neural classifiers, or rather, whether the amount of work needed for removing or correcting is worth the trouble.

3.2 Effects of Noise in Inputs

Noise at the input, is very different from noise at the output. The effect of noise at the output is just the erroneously large error value being sent through the network in the backward pass. The effect of noise at the input already begins at the start of the algorithm and usually only occurs at one or a few of the input columns, resulting in subtle changes due to the complex architecture of neural networks.

Also, the supervisor causes noise at the output, whereas the world, or the observer of the world, causes noise at the input, usually in a structural manner. It cannot be viewed as a simple error anymore, but rather as an unexpected, different view on the domain world. The network just accepts the corrupted patterns as another example of the given class, thus the neural classifier cannot deal with it the same way it can do with noise at the output.

In a study of classification with decision trees, Quinlan ([13]) noted that it is counter-productive to remove noise from the training data when the same type of noise occurs when the classifier is put to use. This seems logical, since the classifier bases it's decisions on the examples it is given, thus giving it examples of noisy patterns will enable it to learn that form of noise, resulting in better classification. This will probably only work well up to a certain amount of noise, and as long as there is some form of structure in the noise.

3.3 Types of Noise

The following examples of character cutouts of license plates show that noise can take many, very different, forms as shown in Figure 3-2. Each type will probably have a different effect on the network during training.

In past publications ([15], [11], [4]), uniformly distributed noise (pepper and salt noise) has been added to improve generalization and robustness. In Figure 3-2, we see that the distribution and concentration of noise is not uniformly distributed, so adding this type of noise before the feature extraction algorithm is run, see Figure 3-1, will probably only improve the network's performance marginally, and will certainly not enable the network to cope with the types of noise seen in the examples above. Adding this type of noise on the inputs, after the feature extraction algorithm in Figure 3-1, might work, but this is an optimisation strategy for the classifier itself, so we did not study this here.

Figure 3-2. Several possible types of noise on character cut-outs of license plates, with the recognized character and confidence level. We see (a) noise causing a character change, (b) cut-out artefacts looking like characters, (c) missing bottoms, (d) noise causing a wrong cut-out size, (e) missing top, (f) occlusion and (g) noise causing character merges.

Adding noise that is known to occur in the domain to which the classifier is applied, will most certainly increase performance on patterns with these types of noise, but will probably degrade the performance on normal patterns and those with a different types of noise.

In general, we can distinguish three categories of noise that have an impact on classifiers. In the following subsections, we will discuss what kind of impact they have, and what effect they have on performance.

3.3.1 Structured Distortions

Noise of this type has a clearly structured form. Geometric changes, localized dirt blobs and missing or occluded parts are examples (see Figure 3-2 [a, c, d, e, f]). Since there are many possible forms, and they typically alter the data in a strong way, classifiers have trouble learning data that contains structured noise. Most builders remove the noise before training so the network train performance increases. More often, these types of noise are not present in the data in the first place, especially if the data is generated.

If the train data does not contain any distortions, the network will almost certainly not be able to cope with distortions after training, so some other mechanism is needed later. [1] contains an example of how this can be done, which is summarized in Section 3.4.1.

3.3.2 Random Distortions

Random distortions cause a change in data that has a stochastic characteristic. Examples are degraded quality by compression of images, focussing problems with a camera, and static in signals. These distortions are not much of a problem for most neural applications, because of their generalization property. In many cases, this type noise, usually uniformly distributed, is even added to a train set, to help the network generalize, as mentioned earlier in this section.

3.3.3 Outliers

Outliers are data patterns that have been altered by noise in such a way, that it falls outside of the domain the classifier is meant to be trained on. Examples are mislabelled patterns (see the previous chapter), and cutout artefacts (see Figure 3-2 [b, g]).

Since outliers are outside the domain of the classifier, they should be handled by the pre-processing system, both before and after training. Will not study this type of noise in this thesis.

3.4 Experiments with Filtering

Noise at the output of the training data can easily be generated, but noise at the input is much more complex, so generation is difficult. Also, datasets that contain real-world noise are not readily available. For this reason, we have chosen to narrow the experiments to a single data set, the large Dutch License Plate data set (Appendix A).

Inspired by the filtering mechanism described in the following section, we will examine the effect of noise at the input by filtering noise out of the training data, and compare the results of training with and without noise. We will consider an automated filtering mechanisms, and filtering manually.

3.4.1 Filtering Noise by Type

In a recent thesis, written here at the University of Groningen ([1]), the effect of occluded characters in license plate recognition was examined. Occluded characters are characters that have parts of them hidden by an outside element (like dirt, stickers, or other obscuring surfaces).

The conclusion of the thesis, was that bad optical character recognition (the processing and enhancement structure) caused a decrease in performance of twenty to fifty percent in the neural network used for classification. A part of this percentage is caused by occlusion. Adding occluded patterns to the otherwise clean train set did help, but the slight decrease in the performance on normal patterns was unacceptable.

The solution suggested, was to create a decision system with two separate networks, one for normal characters, and one for the occluded characters. The final result was an increase in performance from `81%` to `98%` with `22%` occluded characters (the most common percentage of occlusion in license plate recognition) while the performance on non-occluded characters remained the same.

Clearly, filtering noise makes sense, but filtering alone is not enough. If all bad patterns are filtered out, performance on reasonably normal patterns will improve, but recognition of noisy patterns will decrease greatly. If there is a large amount of noise present during application of the classifier, a separate mechanism for these patterns is required. However, when there are many types of noise present at the same time, this mechanism may become extremely complex.

3.4.2 Filtering on Difference to Average Vector

The first experiment we did was to filter patterns by calculating the differences of its input values to those of the average input vector of its class. When the number of differing pixels of which the pixel value falls above or below the 95% boundaries, is larger than a certain threshold (δ), the pattern is removed from the set *(D)*.

$$D' = \left\{ \vec{x} = (x_0, x_1, ..., x_n) \;\middle|\; (\vec{x} \in D) \wedge (\sharp \{ i \mid x_i \leq min_i^{95\%} \vee x_i \geq max_i^{95\%} \} < \delta) \right\}$$

This is a rather straightforward method of filtering, since no prior knowledge of the data set is used, thus some outlying correct patterns might be removed.

When we applied the filter to the large Dutch License Plate set, 2430 of the 18202 patterns were pruned (13.4%). Examining the removed patterns we see that most contain noise or are shifted away from the average form, but there are also patterns that an expert supervisor would leave in. The fil-

Figure 3-3. Averaged characters of the Dutch License Plates data with intensity in greyscale. Figure 3-3 (a) shows the original train data, (b) the original test data, (c) the automatically filtered train set, and (d) the manually filtered train data. It is clear that the filtered data sets are much more precise.

tered data set still contains a few patterns that might be considered noisy, but overall it is much cleaner.

Since this data set is extracted from images, a good visualisation of the filtering mechanism can be made by showing the actual average vectors for each class, as done in Figure 3-3. We see that the filtered train set is much more precise than the original train and test sets, indicated by the higher overall intensity.

In Figure 3-4 we see that filtering noise has a negative effect on the actual performance, when the same type of noise is present in the test set. The filtered data is trained almost perfectly, but the network cannot learn any form of noise, so the test error is larger than the error of the network trained with noise, since there is still noise in the test set.

In the error vs. correct confidence plot, we plot all combinations of percentages of errors and percentage of correctly classified patterns, for each confidence step measured. The resulting graph gives an indication of the trade-off between the amount of correctly classified patterns and the amount of errors, when varying the confidence value. We see that the original network is better at classifying the test set with all confidence values.

3.4.3 Manual Data Selection

In the second experiment, we filtered bad samples ourselves. We tried to remove patterns that did not seem like real characters, or had extra artefacts in them. In general, we tried to generate a set with only good samples. We filtered 3045 samples of the original 18202 (16.7%), a little more than the automated filtering method. As a result, the set looked a little better, with less dirty patterns.

From Figure 3-4 we can conclude, that this set performed marginally better, but still much worse than the network trained with noise left in.

3.4.4 Combinations

One might expect that filtering noise creates a more precise neural classifier, enabling it to filter out noise by itself, because data samples with noise might generate a lower confidence value. However, this does not seem to be the case, as shown in the confidence plot in Figure 3-4.

To test this further, we combined the original data set with the filtered data sets, using half of each train set for training, and using the other halves of the filtered data sets for testing. What we can conclude from Figure 3-5 is that even when the data samples are filtered when the classifier is put to use, the classifier trained with noise still performs equally well, or better than the filtered train sets. A confidence plot could not be made, because too little data points were generated for a reliable graph.

3.5 Experiments with Noise Injection

As an final experiment, we introduced increasing amounts of uniformly distributed noise to the original and manually filtered data Dutch License Plate sets and compared performance. On one hand, previous studies suggested that this might increase performance; on the other hand, this type of noise was not present in the input images so it might not help at all, especially since the feature extraction algorithm probably doesn't take this change into account.

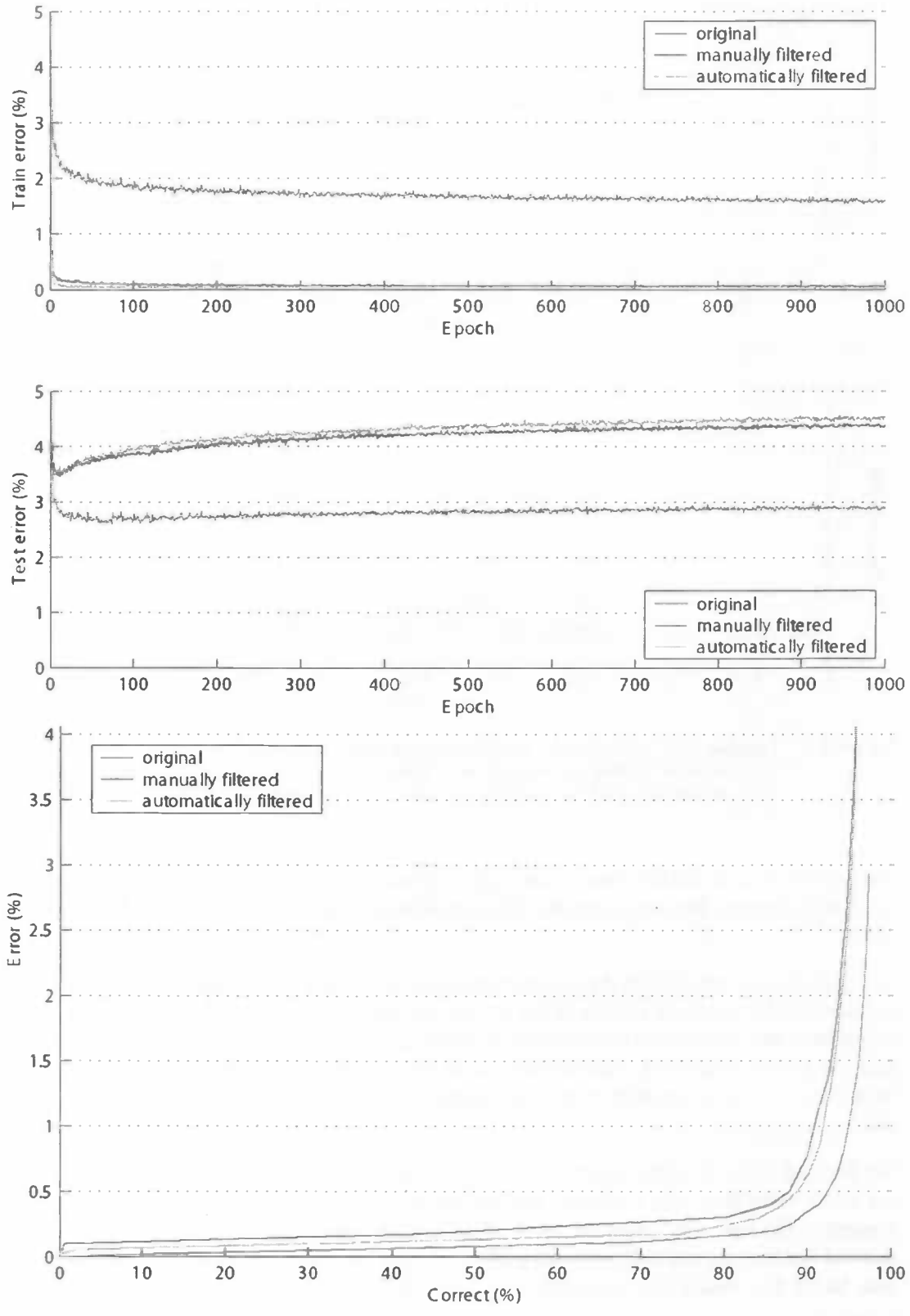

Figure 3-4. Filtering noise reduces the train error to virtually zero, but raises the test error since it cannot generalize about noise. The error vs. correct confidence plot shows that the network trained with noise is the best network on this test set.

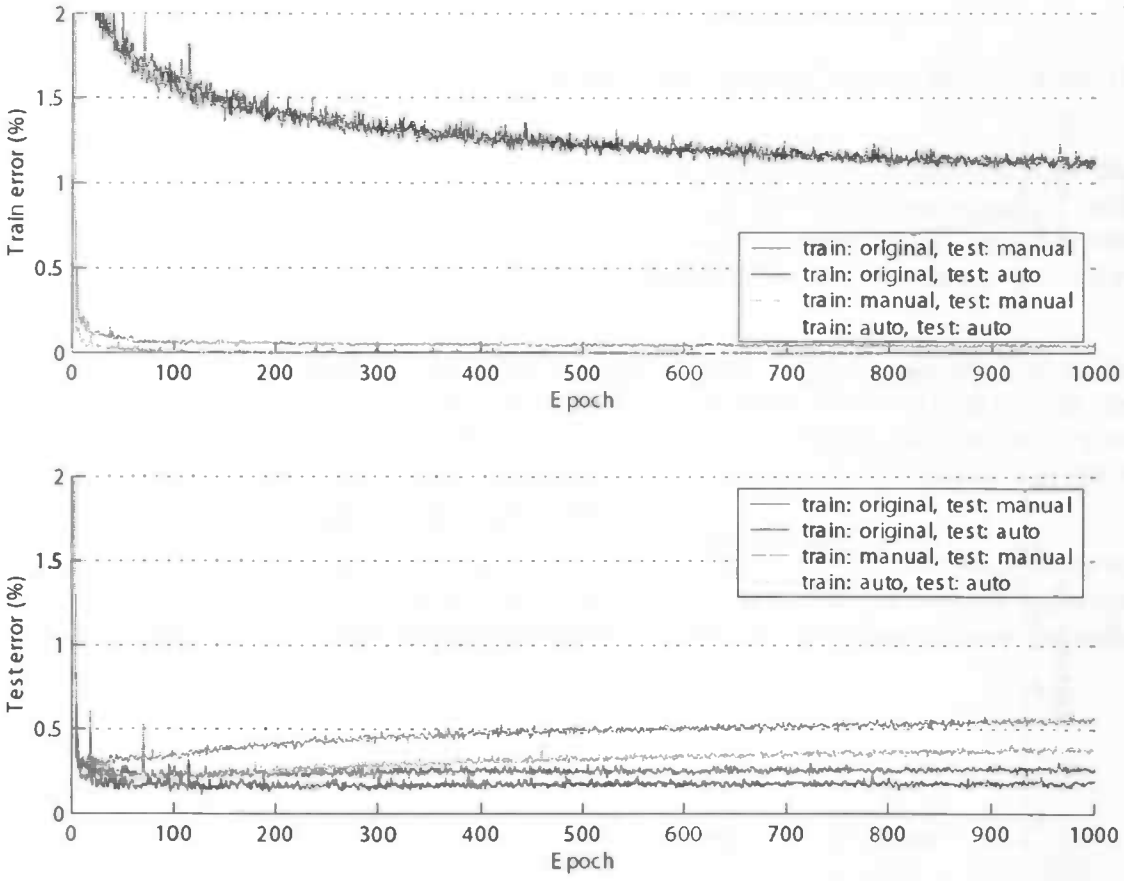

Figure 3-5. Training with noise results in better performance than training with noise filtered out, even when noise is filtered out when noise is filtered out when the classifier is put to use in the field. With optimisation of the training process, the performance might become equal of slightly better.

The experiments we did involved inverting the value of random pixels. This meant that a white pixel became black and a black pixel white. This was done in several quantities, from 5% to 20% in steps of 5%.

In Figure 3-6 we see that in the original (noisy) data set and in the filtered set introducing noise degrades performance. A review of the studies we thought of in the first place, made it clear that they meant that injection of noise should be done on the inputs of the network, not to the data samples, see [15], [11] and [4]. Furthermore, as we thought, this type of noise does not help, probably because it is not of a type that the pre-processing systems and the network itself would find beneficial.

We decided not to do extra experiments to verify the claims of the above-mentioned studies, since not much extra time was available. Further experiments on introducing types of noise that were present in the data sets might be more helpful, as the experiments of the previous section have showed that the original noisy data set performed better in almost all instances than the filtered data sets. Sadly, this would take too much time, so it is left as further research.

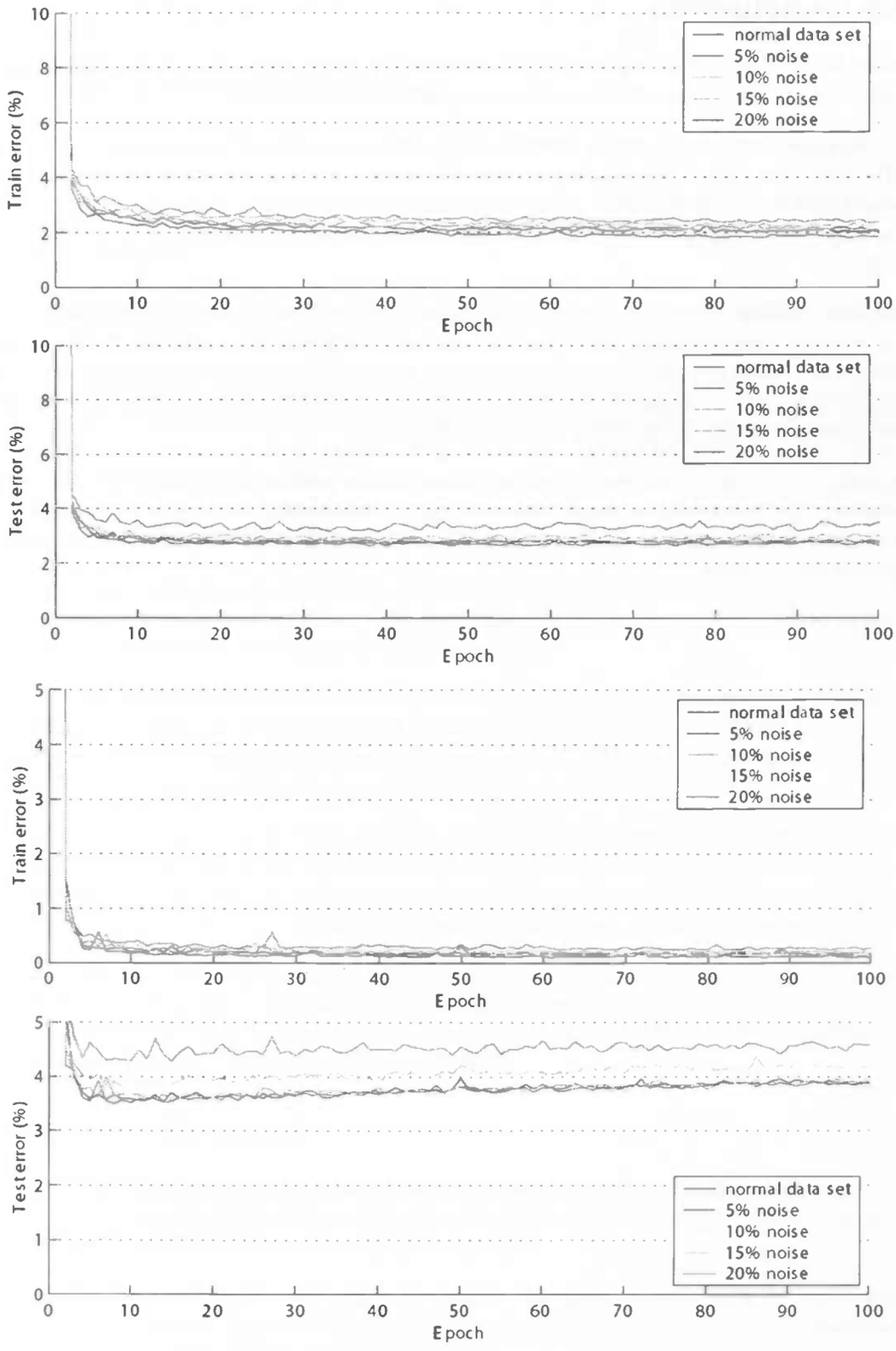

Figure 3-6. Adding noise to the original images of the data set degrades performance of the classifier, both in the original (upper two graphs) and the filtered (lower two graphs) Dutch License Plate data sets.

3.6 Conclusions

Noise in the inputs of the training sets is considerably more complex than that in the targets. The multitude of types makes dealing with it very hard.

As the experiments have shown, filtering data is not a good idea, if, during application of the classifier, the same type of noise is present as in the training process. Quinlan's observation [13] can be generalized to neural classifiers. As he mentioned, leaving noise in seems logical, since the classifier may learn to recognise noise the real world has introduced and learn how to deal with it in some way.

We also saw that filtering noise caused overfitting, probably due to the simplification of the data set. Perhaps with more fine-tuning, a network trained without noise combined with several other networks, which are trained with noise of several types might result in better performance, in a way similar to the improvements found in [1]. Introducing noise in the data, of a type that was not present earlier is of no help, and probably reduces performance.

In general, if noise in the inputs poses a problem, dealing with it is possible, but the solution will most certainly be rather complex. If time is short, just leaving it up to the neural network and accepting the error might be best, since a correctly chosen instantiation of a neural network can generalize about it up to a point.

4 Conclusions

4.1 Summary

In this thesis, we have researched in what amount noise in training data for neural classifiers is a problem, and whether it is justified to invest a lot of time and effort into dealing with it.

We explored the effects of mislabelling, or noise in the targets. The results of experiments where we injected mislabelling into otherwise clean data sets, showed that a data set of high quality makes the neural network very resilient to mislabelling. Up to 30% noise only caused a small increase in the test error. When the quality of a data set is not optimal, the effects of noise become stronger. Even though the test error did not increase much, the confidence of the networks decreased more quickly, even with a high quality data set.

When maximum performance is required, removing noise is necessary. Since filtering is difficult ([2], [9]), and reduces the number of samples, optimising the training process might be a better solution for dealing with noise, but this needs further research.

Noise in the inputs of data is much more complex, and can also cause a decrease in performance of the classifier. There are many types of noise possible, which makes dealing with it much harder. A recent study ([1]) has shown that dealing with noise by type works well, but because of the many possible forms, a solution of this type can become highly complex.

Globally filtering noise in the in inputs caused a decrease in performance and overfitting. The network needs noisy examples of the types that are present when the classifier is put to use, just as decision trees do ([13]). We had hoped that a more precise network, trained with filtered data would give very low confidence scores on noisy patterns, but this did not show in the experiments.

An experiment with injecting uniformly distributed noise in the input data to help the network generalize failed as well, since we applied it before the feature extraction algorithm, which was not prepared for this type of noise. As it turned out, we should have applied the noise right on the inputs of the network, instead of in the data ([15], [11], [4]).

4.2 Further Research

A common issue with training neural networks is the question whether the data set used has enough data to train the neural network. Often, data is hard or expensive to come by, thus it is important to know as soon as possible whether there is sufficient data.

The most common method for testing this, is training the neural network with less data, perhaps decreasing the amount of data in several steps, and comparing the results. When the performance decreases, there probably is not enough data available.

In Chapter 2 we discussed the effect of noise at the output of neural classifiers. The resulting plots showed that noise had a less than expected effect on the performance of the classifier, but they also showed something else.

When we compare the relative gaps between the train errors of the differently mislabelled sets, an indication of the maximum attainable performance of the network can be extrapolated. In most cases, there is a proportional dependence discernable between the percentage of mislabelled patterns and the increase in errors caused by them. This means that we can estimate what the lowest error-rate could be.

When we take a look at the train plot of Figure 2-3 for instance, we see that the train errors converges to a value very close to the percentage of mislabelled patterns, and the mislabelling plot shows that the error almost entirely consists of those mislabelled patterns. The graphs of the data sets of lower quality show a drop beneath the level of noise introduced.

It might be worthwhile to research whether this type of testing can be used as a quality measure for the network, perhaps even removing the need for a test set, which results in more train patterns. In any case, it will give more information that only a single train error.

References

[1] A. Boersma. *Modular Neural Network Classifier for Optical Occluded Character Recognition*. University of Groningen, 2001

[2] C.E. Brodley, M.A. Friedl. "Identifying Mislabelled Training Data", *Journal of Artificial Intelligence Research,* **11**: 131-167, 1999.

[3] J.C. Chan, M.C. Hansen, R.S. Defries. Machine Learning Methods to Identify Mislabelled Training Data and Appropriate Features for Global Land Cover Classification, *21st Asian Conference on Remote Sensing (ACRS 2000),* 2000.

[4] Y. Grandvalet, S. Canu. A comment on noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics,* **25 (4)**: 678-681, 1995.

[5] S. Haykin. *Neural Networks: A Comprehensive Foundation*, second edition. Prentice-Hall International, 1999.

[6] D. Hawkins. *Identification of Outliers*. Chapman & Hall, 1980.

[7] C. Jutten (Project Coordinator). "ELENA (Enhanced Learning for Evolutive Neural Architecture)", *Deliverable R3-B4-P, Task B4: Benchmarks, ESPRIT Basic Research Project Number 6891*, 1995.

[8] S. Lawrence, I. Burns, A. Back, A.C. Tsoi, C.L. Giles. "Neural Network Classification and Prior Class Probabilities", *Tricks of the Trade, Lecture Notes in Computer Science State-of-the-Art Surveys*, 299-314, 1998.

[9] S. Lallich, F. Muhlenbach, D.A. Zighed. "Improving Classification by Removing or Relabelling Mislabelled Instances", *Proceedings of te XIIIth International Symposium on Methodologies for Intelligent Systems (ISMIS 2002),* 2002.

[10] R. Little and D. Rubin. *Statistical Analysis with Missing Data*. New York, Wiley, 1987.

[11] K. Matsuoka. "Noise Injection Into Inputs in Back-Propagation Learning", *IEEE Transactions on Systems, Man and Cybernetics* **22 (3)**: 436-440, 1992.

[12] D. Michie, D.J. Spiegelhalter, C.C. Taylor. *Machine Learning, Nearal and Statistical Classification*. http://www.amsta.leeds.ac.uk/~charles/statlog/ Ellis Horwood, 1994.

[13] J.R. Quinlan. "Induction of Decision Trees", *Machine Learning*, **1**: 81-106, 1986.

[14] V. Rothamsted, V. Barnett, and T. Lewis. *Outliers in Statistical Data*. Wiley, 1994.

[15] J. Sietsma, R.J.F. Dow. "Creating Artificial Neural Networks That Generalize", *Neural Networks*, **4**: 67-79, 1991.

[16] J. Schafer. *Analysis of Incomplete Multivariate Data*. London, Chapman and Hall, 1997.

[17] R. Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, Inc., 1992.

A Data Sets

A.1 License Plates

A.1.1 Dutch

The Dutch License Plate set contains features extracted from characters on license plates, captured by cameras along the highway. The data set has thirty inputs, all results from a principal component analysis, and thirty six outputs, one for each possible character on the license plates. The number of patterns per class varies greatly. The expected error-rate on the sets is about 3.0%. The large data set became available later than the others, and the feature extraction algorithm used had undergone some development resulting in eighteen more input dimensions, and less classes (difficult classes were removed).

Default

Input dimensions:	30
Classes:	36
Total patterns:	3117 (unevenly divided)
Train-test parts:	60% - 40%

Small

Input dimensions:	30
Classes:	36
Total patterns:	583 (unevenly divided)
Train-test parts:	60% - 40%

Large

Input dimensions:	48
Classes:	27
Total patterns:	36386 (unevenly divided)
Train-test parts:	60% - 40%

A.1.2 Portuguese License Plates

The Portugues License Plate set has been gathered at toll booths of the Brisa Corporation. The compression level of the images was set too high, so there is a lot less information available in the images. The layout of this data set is otherwise the same as the Dutch License Plate set.

Input dimensions:	30
Classes:	36
Total patterns:	583 (unevenly divided)
Train-test parts:	60% - 40%

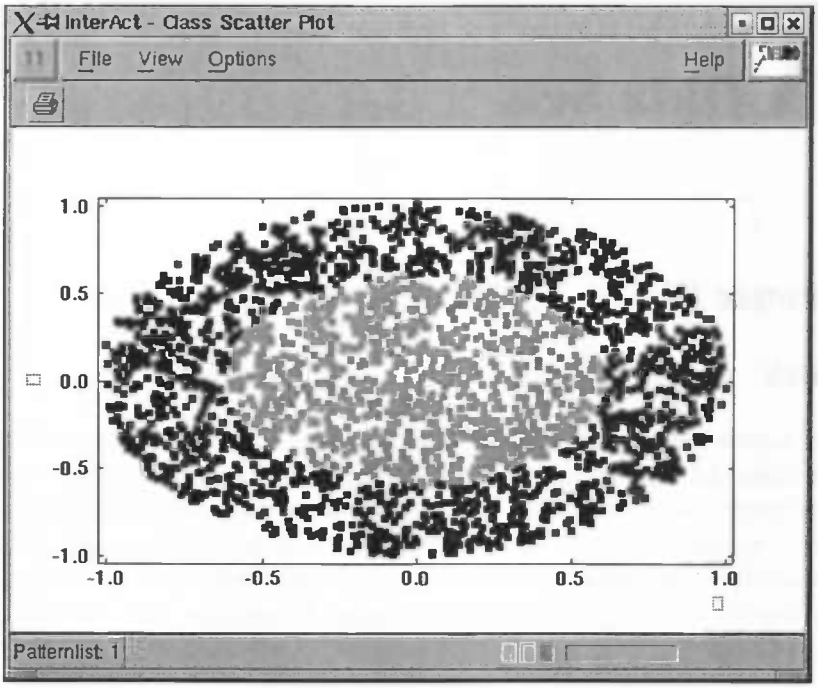

Figure 1-1. A scatterplot of the Concentric data set.

A.2 Elena Datasets

The Elena datasets are extensively described in [7]. To give a general idea about the sets, we will give short descriptions here, with the most important statistics.

A.2.1 Iris

The data set has four input dimensions, and contains three classes of fifty instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other. There are too few patterns for a good classification, certainly after dividing the set in train and test subsets.

Input dimensions:	4
Classes:	3
Total patterns:	150
Train-test parts:	60% - 40%
ELENA MLP error:	about 4.5 ± 5%

A.2.2 Concentric

The Concentric data set has two input dimensions, and two classes. One class a has its patterns in a circle, the other in a ring around the first, as depicted in Figure 1-1. The patterns are unevenly divided between the classes, 1579 in class 1 (63.2%), 921 in class 0 (36.8%). There is no overlap between the classes, so the theoretical error is 0%.

Input dimensions:	2
Classes:	2

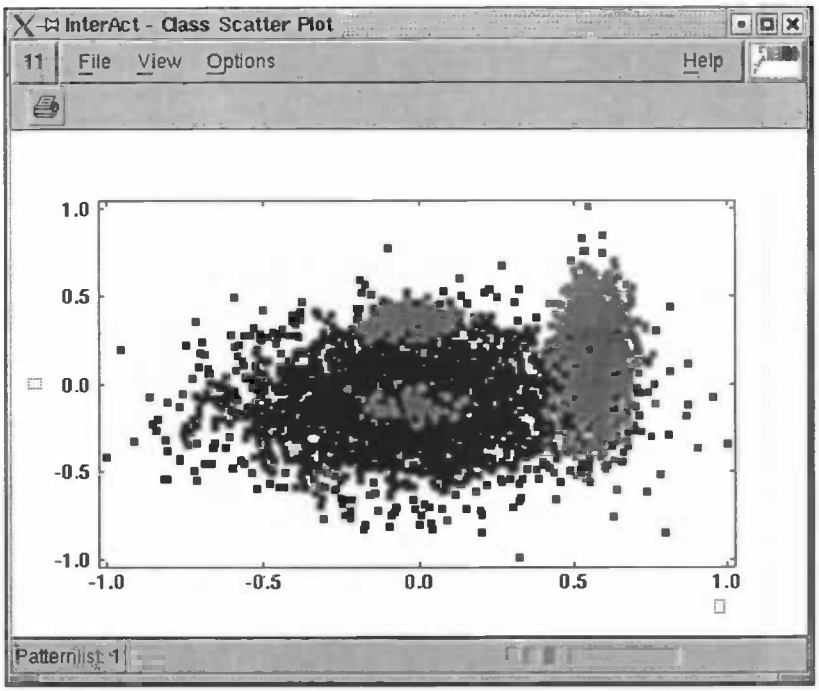

Figure 1-2. A scatterplot of the Clouds data set.

Total patterns:	2500 (unevenly divided)
Train-test parts:	60% - 40%
ELENA MLP error:	about 2.8 ± 1.5%

A.2.3 Clouds

The distribution of the patterns of this data set is gaussian. The first class is the sum of three different gaussian distributions, the second class is a single gaussian distribution. Both classes have 2500 patterns. A scatterplot of the patterns is given in Figure 1-2. The overlapping patterns cause the theoretical error to be 9.66%.

Input dimensions:	2
Classes:	2
Total patterns:	5000
Train-test parts:	60% - 40%
ELENA MLP error:	about 12.5 ± 2%

A.2.4 Gauss

This database is used to study the effect of the input dimensions on the error rates. The distribution of the classes is gaussian, and they overlap since their centers are the same, see Figure 1-3. The input dimensions are increased in six additional sets, but the number of patterns remains the same in each set. This means there are sufficient patterns up to five input dimensions. We have selected the 2D and 8D sets for our experiments.

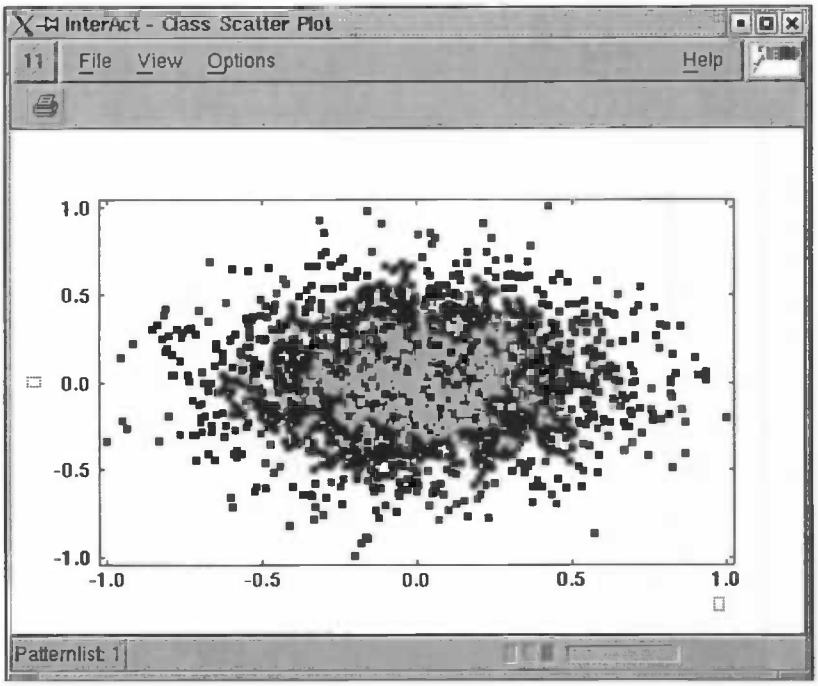

Figure 1-3. A scatterplot of the Gauss 2D data set.

2D

Input dimensions:	2
Classes:	2
Total patterns:	5000
Train-test parts:	60% - 40%
ELENA MLP error:	±27%

8D

Input dimensions:	8
Classes:	2
Total patterns:	5000
Train-test parts:	60% - 40%
ELENA MLP error:	±15%