VALIDATION AND VERIFICATION OF EMBEDDED NEURAL SYSTEMS

Nicolas Fischer

MSc by research in Pattern Analysis and Neural Networks

THE UNIVERSITY OF ASTON IN BIRMINGHAM

September 1997

The University of Aston in Birmingham

Nicolas Fischer

MSc by research in Pattern Analysis and Neural Networks

1. Thesis summary

There is growing interest in neural networks in the industrial world and more and more safety related software involves or will involve them. Therefore a need for assessing the level of safety of a neural network software has become an essential task. The first year of this project gave an overview of neural network technology, which included in particular a description of the main principles on which it is based and an emphasis on the differences between software embedding neural network technology and "classical" software (see [D2]). This lead to a set of good practice rules to develop a successful neural network application, from which guidelines to assess a neural network system had been derived (see [D3]).

The aim of this project is to explore more deeply the different means and techniques which will allow us to assess the ability of a neural network to perform a certain task. Firstly the work concentrates on the data set and the verifications that have to be carried out to ensure its quality. The main problems that have to be addressed in this context are typically the validity of the noise model, the possibly multivalued character of the mapping function, how representative of the real data the data set is and finally the links between the features of the data set and the features of the model (i.e. of the neural network). From this study, the aim is to improve the set of assessment guidelines. Secondly, a shorter part of the thesis shows how to reason about a NN embedded in a safety related environment, that is how a safety case could be obtained for neural network software. Finally a set of issues and suggestions for future research are provided.

This document is necessarily limited in its scope. Indeed, we have chosen to restrict our study to the two main types of models currently used, namely, the radial basis function network and the multi-layer perceptron. We have excluded, for instance, unsupervised techniques and recurrent neural networks that raise notoriously difficult problems. However, whatever the model used, they are related in that they are data-driven. Consequently, issues relative to the data discussed in this thesis are central and generally applicable to each of these models.

Key words:

Safety critical, safety case, feed-forward neural networks, data set, noise, multivalued mappings, model selection, good practice rules.

2. Acknowledgements

I would like to thank Dr Peter Scharbach (Lloyd's Register project manager) for having shared his experience with me and for his kindness for always providing the information I needed for the project. I am also very thankful to Lloyd's Register for the financial support they provided by paying my tuition fees.

I would also like to thank Michael Paven who authorised me to re-use the "Business Context" part of his thesis which gives some general information about LR.

Last but not least, I would like to thank Dr Ian Nabney for having supervised this project and providing a lot of advice and help in both software engineering and neural network technology. I would also like to thank him for his precious help in the rephrasing of some of the sections of this project.

3. Table of Content

4. Table of figures

5. Foreword

- This project has been developed in collaboration with Lloyd's register (LR).

- This document is intended for anyone interested in neural networks (NN) technology, and in particular for assessors of NN applications. To read this document, it would be useful to have read Deliverable 2 ([D2]) of last year's project. No further prior knowledge of NN technology apart from what is in this Deliverable is required. Besides, it is assumed that the reader has studied mathematics to, at least, an A-level standard (some knowledge in statistics might also be particularly useful). Finally, note that experience in conventional software development or in the assessment of conventional software applications may be helpful.

- Appendix A contains two complementary points which fit more in last year's project but are also relevant to this project. These two topics are an introduction to Bayesian learning and a presentation of classification problems using cost matrices and posterior probabilities.

- Appendix B contains the sections of the Guidelines ([D3]) which were modified this year.

6. BUSINESS CONTEXT

6.1 Lloyd's Register

6.1.1 Overview

Lloyd's Register (LR) is an international company, whose work can be summarised by the following statement: LR provides its customers with services intended to help them to comply with safety, quality and reliability standard criteria. LR provides commercial safety and quality assessment, and in the case that compliance requirements are fulfilled LR can award certification. Finally, LR is active in research, which allows LR to play a part in the definition and the improvement of new international safety and quality standards.

Nowadays there is an increasing appreciation of the importance of safety and quality. LR is a financially stable group that has always belonged to the leading group of classification societies. Moreover, LR is one of the pioneers in the domain[1], and hence is now highly renowned.

The range of sectors within which LR operates is considerable, and while its main sector of activity remains the marine, LR is present in numerous others. For instance, LR has clients in the chemical, oil, transport, and software systems industries.

LR is arranged in 2 main divisions and 1 group:

- the ship division: which, in 1994, represented 57% of LR income,
- the industrial division: 21% of LR income in 1994,

6.1.2 Industry division

This division, which is the initiator of the project, has two types of departments:

[1] LR has provided certification services since 1760.

- the plan appraisal departments, which operate in various fields (such as piping, machinery design) and provide expertise in these fields.
- the assessment departments, which provide assessment in the same fields as the Plan appraisal departments.

SIRM (System Integrity and Risk Management) which has commissioned the project, belongs to the latter group of departments, and is itself arranged in 2 groups: "system integrity group" and "risk management group". These have a common theme the safety of software systems.

Most of the staff concerned with the project, belong to the system integrity group.

The system integrity group provides 2 different commercial services: "assurance" and "certification".
Assurance means: ensuring that a system complies with certain criteria in order to answer the question "is this system appropriate for a particular use?". The quality standard criteria are principally concerned with process rather than product: i.e. checking that appropriate methods have been used to develop the software.

One of the main differences between the certification and the assurance processes, is that the certification process may lead to the award of a certificate, whereas the assurance process does not. Nevertheless, assurance and certification can both cover any phase of the software life cycle, and are available to any user or supplier of software. Usually these services are requested when the software forms part of a high-integrity system.

6.2 Validation and verification of embedded neural system

6.2.1 Why neural networks?

Commercial neural network (NN) applications have been in existence for more than 5-10 years, but now the technology is moving from research to products in more safety related areas.

Their power and their efficiency are becoming more widely known, and in a few years, safety related systems including software based on neural network technology will probably start to be widely developed.

For LR, this means a potential new market. Indeed, neural network technology has already been applied in several fields relevant for LR's business[2], and many others will surely be found in the future.

Furthermore, in the case that LR were to be the first to propose the assessment of safety related systems including neural network technology, that would be an important advantage over competitors: that is the advantage of pro-activeness.

NNs are a new way of viewing software. Indeed, the classical view of software development is mainly based on the implementation of an algorithm. This classical approach is not valid with NNs, since using NN consists of building a model that is based on data. For instance, this means an absence of rigorous specification to describe the NN functionality. But above all, this means for LR, that most of the usual software assessment methods no longer cover all aspects of system safety.

Hence, LR would like to have available methods to assess safety related systems including NN technology. As NN technology is currently not mastered by LR, Aston University Neural Computing Research Group is to develop such methods in this project.

This project is important, since its results will be used on the international market. The whole project is planned to last 3 years. The purpose of this MSc project was, first to complete last year's technical context by focusing more on the data and to improve the guidelines. Secondly to analyse whether a safety case could be obtained for NN software and finally to raise the main research issues that will have to be solved to complete the overall project. This phase of the project has lasted one year.

---

[2] for instance, neural networks are used in medical instrumentation, and currently, pilot studies are being carried out in the naval vessel and car motor ignition fields.

6.2.2 Project definition

A safety critical application must be an example of rigour, and "nearly right" is not acceptable, since, by definition, if there is any problem with this kind of application, the consequences may be catastrophic. Hence, in safety critical systems, it is important to be able to show that the probability of dangerous failure is very low.

If a serious problem occurred with a safety critical application certified by LR, and this problem occurred in conditions for which the application had been certified, then LR's credibility would be affected and LR could be liable.

However, as NNs are probabilistic models, a set of questions arises:

- Are there any limitations, or specific context for their use?
- What are the phases of the system development life cycle that cannot be taken into account by using conventional software assessment?
- What are the key safety properties that need to be assessed?

The key aspect that has to be taken into account in neural network software is that it is highly data dependent. This is why the assessment of the data set and the way in which it was used are fundamental. The main task of this MSc project is to identify the problems related to the data set and to provide techniques to tackle them. The guidelines must be updated in order to include the new findings.

The main points that will be examined in the study of the data set are:

- noise and multivalued function mappings.
- "representativity" of the data set, that is if it is appropriate and of good quality for the task we want to carry out.
- the links between data set and choice of the model.

The final part of the project will be to examine neural networks in a safety critical context that is to see if a safety case can be obtained for NN software.

Note: the guidelines must have à very practical aspect[3]. They must enable LR assessors, first, to answer the crucial question "is a system safe for the use for which it has been designed?", and secondly, to understand what are, in terms of safety, the underlying safety requirements. Moreover, they must be relevant for the typical scale

[3] since they are intended for software assessors and not neural network specialists, implementors, nor system designers.

of LR projects. What has also to be firmly emphasised is that the guidelines are only to be used with a good knowledge of the technical documents (typically [D2] and [D8]).

The last stage of the MSc project may consist of testing the guidelines through at least one case studies, in order to correct and refine them; but for scheduling reasons, these cases studies are not included in the thesis.

7. Introduction

We mainly focused on the two main types of neural networks architectures which are Multi Layer Perceptrons (MLP) and Radial Basis Functions networks (RBF). Thus there are no special sections for less frequently used architectures like Kohonen networks (and other unsupervised learning techniques) or recurrent networks. But since all these models rest all on the same principles and are all data driven, all the findings of the project apply to all kinds of neural networks. It can be considered that at the current state of the project, all the problems which are common to all NN have been treated, as well as some which are more particularly related to one kind of architecture or problem (time series data for example). Nevertheless, in the final section we will present some material that future work on the project could cover.

8. Data Analysis

8.1 Introduction

In this section what we seek is a deeper insight in the data in order to improve our guidelines and consequently our methods of NN assessment. Several issues will have to be examined:

Analysis of the quality of the data

Analysing the quality of the data is not an easy task since a great deal of parameters have to be taken into account. Some clear cut rules to assess the quality of our data set will not always be found but we will try to get a standard methodology to assess it, and to study the consequences this data has on the subsequent modelling process.

The issues that we will address are:

- how to model our data density and the importance of mixture models.
- how to check if the data is not too noisy (and hence if the assumptions about the noise are accurate).
- how to check if the functions that maps the inputs to the outputs is not multivalued.
- how to tackle all the difficulties linked with how representative the data is of the problem.
    - * to check if there are enough data points in our data set to carry out a good training.
    - * for classification problems to check if there are enough data points for each class.
    - * to check if the density of the data is homogeneous.
    - * to check if there is some missing data.
    - * to examine how to handle "large" data sets.

Selection of the model

And the other main issue that is related to data analysis is the influence the data will have on the choice of the model (see [D3]).

- How does the data have to be split for time series? Is cross validation the best way of doing what we want? can we use a validation and test set?
- How to choose the best algorithm to train the NN.
- How to find the right complexity for the network and how to reduce it if it is too high?

- How to tackle the problem of random numbers that can occur in the learning process.
- Committees of networks as solution against randomness and generalisation problems.

8.2 Density modelling

Density modelling has already been mentioned in [D2] since it plays a key role in the analysis of the data set and consequently in the training of the NN. At many stages in the process, a good understanding of the data is needed in order to carry out a good training. One of the tasks of the assessors is thus to check if a suitable model has been chosen and if it has been put to an appropriate use. In what follows we will provide a deeper insight into these topics.

8.2.1 Importance of density modelling in the NN context

Density modelling has several applications in NN application development and plays a key role in the following areas:

- in the training process for neural networks.
- in some methods for computing error bars.
- in the modelling of Conditional Density Networks.
- in the estimation of some noise models.
- in the estimation of missing values.

After giving two short examples to explain how parametric and non-parametric methods work we will principally focus on one form of semi-parametric methods: the mixture model.

8.2.2 Parametric models and non parametric models

8.2.2.1 Parametric models

These models assume a specific functional form for the density to be modelled. An easy example to understand how this comes about is to consider a data set S and to model it with a gaussian distribution. Some straightforward computations show that the mean and the variance of the distributions are given by:

$$\mu = \varepsilon[x] \text{ and } \Sigma = \varepsilon\left[(x-\mu)(x-\mu)^T\right] \text{ where } x \text{ is the data set.}$$

This computations are straightforward and computationally not very expensive, but the drawback of these methods is that since they assume a specific functional density, they might be incapable of providing a good representation of the true density.

8.2.2.2 Non-parametric models

In these models, no specific density function is assumed. The simplest non-parametric methods are histograms but they have a lot of disadvantages (non continuous and hard to generalise to higher dimensions because of the dimensionality increase). There are also kernel based methods and a method called K-nearest neighbour algorithm but the main drawback of all these methods is that they all suffer from the "curse" of dimensionality and are very often difficult to use in practice (for more details see [d]). In such an approach, the density function is represented as a linear superposition of kernel functions, each of these being centred on a data point.

8.2.3 Semi-parametric models: mixture models and maximum likelihood

8.2.3.1 Description

In the mixture model, the density is still represented as a linear sum of kernel functions but the number of kernel functions is treated as a parameter of the system and is usually much less than the number of data points. Unlike with non parametric methods, the mixture model grows with the complexity of the problem and not with the size of the data set.

$$p(x) = \sum_{j=1}^{M} p(x|j)P(j)$$

The parameters $P(j)$ are called the mixing parameters. An important property of these mixture models is that they can approximate any continuous density to arbitrary accuracy provided the model is large enough and the parameters chosen carefully enough.

Here we are considering only mixture models where the individual component densities are given by Gaussian distributions.

$$p(x|j) = \frac{1}{\left(2\pi\sigma_j^2\right)^{d/2}} \exp\left\{\frac{-\|x-\mu_j\|^2}{2\sigma_j^2}\right\}$$

Let us now consider the error function that will have to be minimised to find the adjustable parameters. It is a negative log-likelihood error function:

$$E = -\sum_{n=1}^{N} \ln\left\{\sum_{j=1}^{M} p\left(x^n \mid j\right) P(j)\right\}$$

The derivatives of E don't need to be computed (they would anyway lead to highly non linear equations). We can determine the coefficients with an iterative process called the Expectation Maximisation algorithm (EM algorithm) where the error function is going to be decreased at each iteration. It is a simple and practical method to estimate the mixture parameters without having to compute solutions for highly non-linear equations.

8.2.3.2 EM algorithm

We are not going to give a demonstration and a long description of this algorithm (see [d] for that) but only an outline that will show how easy it is to use it.

The input of the algorithm are: data set and initialisation for μ, σ and $P(j)$.

The outputs are: the final values of the mixture model for μ, σ and $P(j)$.

What has to be noticed about this algorithm is that:

- it is iterative
- it tends to converge quite rapidly

Mixture models have many important applications in the NN context, among which are for example:

- the configuration the basis functions in RBF networks.
- the use of mixture density networks.

8.3 Noise models

As we have already mentioned it several times, the data plays a key role in NN training. To obtain an accurate model we have to ensure that the level of noise that corrupts our data is not too high and that the assumptions that we have made to model this noise are accurate.

8.3.1 Rule of thumb ?

In Deliverable D2 we wondered whether a rule of thumb could not be found to estimate the maximum level of noise. This rule of thumb would have been a function of:

- the data set mean value
- the data set standard deviation
- the size of the training set

What is noteworthy here is that the maximum level of noise does almost not depend on the features of the network.

It is straightforward to understand that:

- the smaller the data set mean value is the smaller the maximum level of noise is going to be.
- the larger the size of the training set is, the less influence a given noise level will have and thus the higher the maximum level of this noise will be.
- the influence of the standard deviation might be a harder to explain since it has a direct influence on the quality of the data set.

The literature does not mention the existence of such a rule of thumb because it is difficult to find how these three factors are linked together. Such a rule could probably not work since noise is connected with conditional variance.

That's why we are going to use another method to determine which noise model is the best one.

8.3.2 Different kinds of noise

If f is the function that we try to map then the real noisy function that we will actually map is:

$$y = f(x) + \sigma_o^2(x) + \sigma_w^2(x) + \sigma_i^2(x)$$

where

$\sigma_w^2(x)$ is noise due to the weights of the NN.

$\sigma_o^2(x)$ is noise in the outputs (i.e. targets) of the training set.

$\sigma_i^2(x)$ is noise present in the inputs of our data set.

In our study of the noise we seek to take into account all these different sources of noise in order to be as precise as possible.

8.3.3 Normal quantile plots of the residuals

Most of the time it is assumed that the noise is gaussian and of zero mean. Let us now try to see how such an assumption could be checked.

Once our net has been trained with a given noise model, a good way to test if the noise model was actually gaussian as it had been assumed could have been to plot histograms of the residuals of the net (the residuals are the differences between the real outputs of the net obtained with the training set inputs and the targets used to carry out the training; the residuals are expected to behave like the noise). But as we will see in the examples below, this method is not always very effective. A far more effective method is to draw a normal quantile plot of the residuals. If another kind of noise is assumed there are similar tests which work in the same way. Here we are using a gaussian noise in our example since it is the one most commonly used.

For each quantile we count the number of points we find in it (the quantity is normalised) and then this quantity is plotted with respect to the expected number of point value if it were a gaussian.

If the results are good, that is if the distribution is actually gaussian the normal quantile plotting plots the residuals on the line $y = x$, otherwise the further away from the line the points are, the less normal the distribution is.

We now show some examples that will illustrate how to interpret a normal quantile plotting. Every normal plot of the residuals will be presented with a histogram of the residuals to show that it is more powerful than them. In the examples the data to be modelled is a sine wave with a gaussian noise of zero mean. The network used is an MLP and there are 100 points in the training set (see [o]).

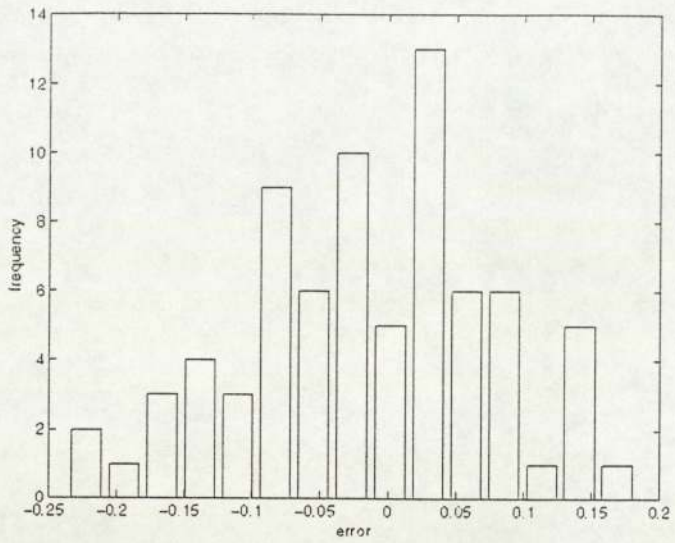

Figure 1: Histogram of the residuals for a well trained NN.

Here the network was trained with input points within [0.1,0.4]. The histogram of the errors does not really look like a gaussian distribution. We can however see that the points are numerous between -0.05 and 0.05.

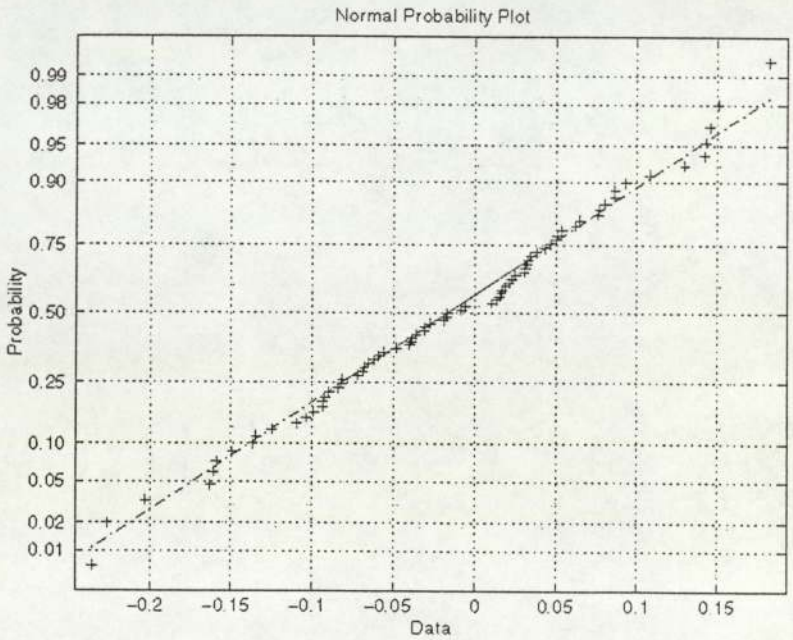

Figure 2: Normal quantile plot for a well trained network.

This is the normal quantile plotting of the network. We can observe that all the points are on the line $y=x$ and thus the residuals are gaussian, that means that the noise is gaussian.

When the network has to extrapolate, the normal quantile plots make it possible to detect it.

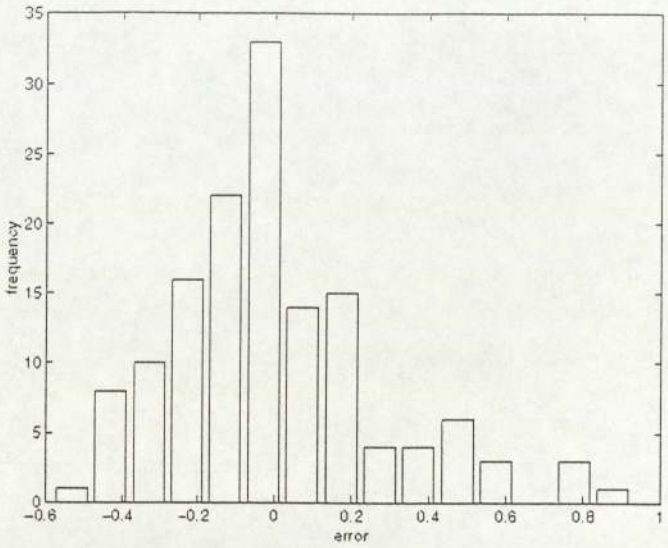

Figure 3: Histogram of the residuals for an extrapolating NN.

Here the training was carried with the same points than previously but the testing set was a set of points within [0, 0.7]. This means that the network had to extrapolate for the values between 0.4 and 0.7. This histogram looks like a skew gaussian distribution with a long tail.

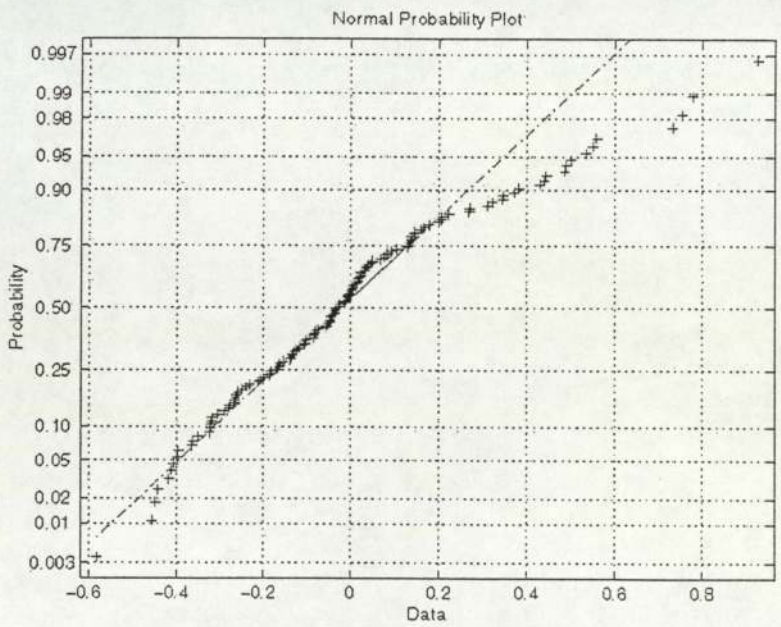

Figure 4: Normal quantile plot for an extrapolating NN.

This is the normal quantile plot of the example tested on [0,0.7]. Here it is obvious that the NN extrapolates for the high values since the points move away from the line at the top. The assumptions were thus poor for these high values.
Let us now consider an undertrained network.

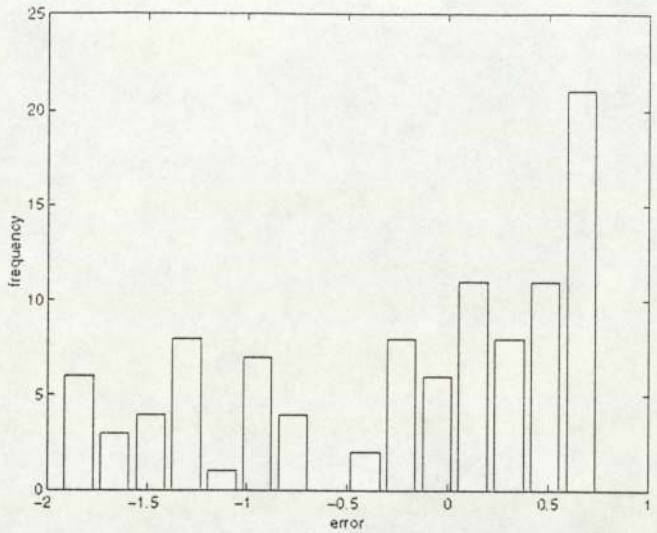

Figure 5: Histogram of the residuals for an undertrained NN.

On the histogram it can also be observed that the distribution of the residual is not at all gaussian and that the noise model assumptions break down.

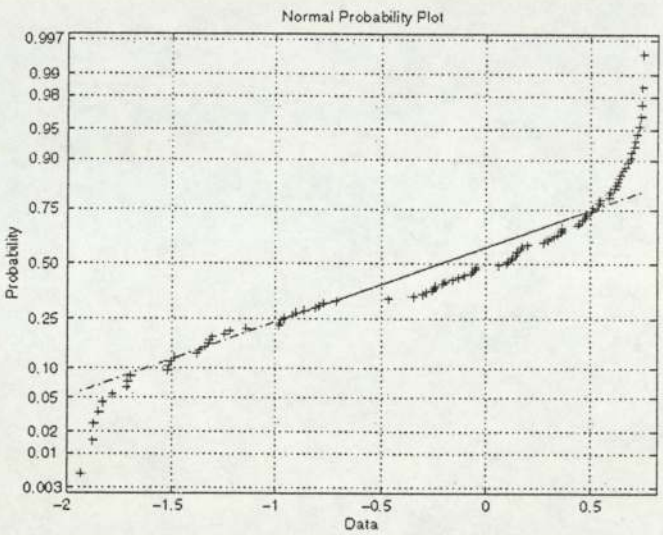

Figure 6: Normal quantile plot for an undertrained NN.

It is obvious that this noise distribution is not gaussian.

8.3.4 Statistical tests to check a distribution

8.3.4.1 The chi-squared goodness-of-fit test

This test is easy to use and is valid over a very wide range of situations. Indeed, all that the test demands is that n independent observations are capable of being classified into a number, say k non overlapping categories and that the probabilities of observations falling into these categories can be calculated when the appropriate null hypothesis is assumed true. The procedure has the name "chi-squared" because the null distribution of the test statistic is approximately a chi-squared distribution. This test can easily be used for our problem of normality checking.

8.3.4.2 The Kolmogorov-Smirnov test

This test is specifically designed for analysing goodness of fit situations where the underlying distribution is continuous. One immediate consequence is that we are saved the awkwardness of creating arbitrary categories as needed in the chi-squared test. Another useful feature of the test is that it can be carried out using either an arithmetical method or an equivalent graphical approach.

The Kolmogorov-Smirnov test statistic is defined as the absolute value of the largest difference between two relative cumulative frequency distributions. This test is used to test the deviations of observed continuous frequency distributions from expected ones and for setting confidence limits to a cumulative frequency distribution.

For a sample of $n=42$ items the *5%* critical values of D is found to be *0.20517.* Observed values of D greater than this critical value would be considered significant. For sample sizes n>100 the two tailed value D_α can be found as $D_\alpha = \sqrt{-\ln(0.5\alpha)/(2n)}$.

8.3.4.3 Significance Testing

This test can be used to check if the value of the mean of the noise distribution that we have assumed is zero. It is a classical student test:

$$t = \frac{x - \mu_0}{s/\sqrt{n}}$$

This test is carried out when we have assumptions that have to be checked about the mean of a distribution. In the formula above s is the square root of the sample variance, that is a value that we can compute easily. If the population is normal then the random variable t has a distribution called the t-distribution. What is more, confidence intervals are easy to compute with this distribution. Let us for example compute this significance test for the same three situations than in the previous paragraph.

	$\bar{x}$	s	t	P
[0.1,0.47]	-0.0160	0.272	-0.2335	0.591
[0,0.7]	-0.0054	0.848	-1.5051	0.934
Undertrained	-0.2321	0.8477	-2.7378	0.997

The conclusion is that the first case is fine since P is near 0.5, the second one is poorer and the last one is very poor.

8.3.5 A new approach: Input noise

8.3.5.1 Presentation of the perturbation model

So far every time we have discussed a noise model, this noise was always on the targets, whereas the inputs were always considered as being noise free. Most of the time it is legitimate to consider the inputs as noise free but it can also be inappropriate. In that case the performance of our NN will be affected by the noise present in the inputs.

The aim of what follows is to try to find out what influence a noise in the inputs can have on the outputs once this noise has been propagated through a NN. The results that we will get are a covariance matrix on the outputs (i.e. an error bar because of the noise present in the inputs).

Since the outputs of one network (or of one layer) are often the inputs of another network or layer, this procedure could be a basis for an error prediction algorithm for NN.

It has to be mentioned that the technique we are dealing with is very new and a lot of research has still to be done to make it widespread practically. This new method is based on a perturbation model, that is to study how an error present in the inputs will affect the reliability of the outputs.

In a perturbation model it is assumed that all the variables of interest can be approximated using an expansion of the type:

$$x = \hat{x} + \delta x$$

where x is the measured value, $\hat{x}$ the real value and δx is the error in the estimate.

It is furthermore assumed that the error δx will have a gaussian distribution, that is:

$$E\left[\delta x \delta x^{T}\right] = X$$

The principal assumptions of this estimate are

- that the previous approximation can be applied not only to the variables but also to the functions.
- that δx is small compared to x.

8.3.5.2 Validity of the Model

It has been said that the high degree of non-linearity present in the component functions prevented the use of perturbation models. However, if we consider that the error is due to the weights rather than to the functions themselves (this is the case provided that we have an appropriate number of hidden units), then the functions can be considered to be independent of the error and act only as propagators of three sources of errors:

- error in the inputs
- error in the weights
- error resulting from an erroneous weight vector acting on an erroneous input vector

If we look at Taylor's expansion of the usual non linear functions used in NN we have that:

$$\hat{y} + \delta y = f(\hat{x} + \delta x) = f(\hat{x}) + \delta x f'(x) + O(\delta x^2)$$

$$\Rightarrow \delta y = \delta x f'(x) + O(\delta x^2)$$

And so we see that the suitability of a perturbation model for estimating the transmission of an error through a non-linear activation function is mainly dependent on the magnitude of the first derivative of the function.

Since the maximum of the derivative of the sigmoid (and the logistic sigmoid) activation function is bounded, there are no problems for this perturbation analysis. But let us now apply this analysis to an MLP.

8.3.5.3 Application of the perturbation model to an MLP

MLPs can be treated as having several independent layers for the purpose of perturbation models analysis because the outputs of any layer in the model is solely dependent of the inputs of that same layer.

This analysis has to be performed once the training has been carried out.

The input vector x is given by:

$$x = \hat{x} + \delta x$$

where x is the actual input value, $\hat{x}$ the expected value and δx the error present in the input.

The output is given by $y = f(x)$

v_j is the weight vector corresponding to the hidden layer node j.

This weight vector is decomposed in the same way than the inputs:

$$v_j = \hat{v}_j + \delta v_j$$

Thus according to Taylor's expansion:

$$\hat{y}_j + \delta y_j = f\left(\hat{v}_j^T \hat{x}\right) + \left(\hat{v}_j^T \delta x + \delta v_j^T \hat{x} + \delta v_j^T \delta x\right) f'\left(\hat{v}_j^T \hat{x}\right) + o\left(\delta x^2\right)$$

$$\Rightarrow \delta y_j \approx \left(\hat{v}_j^T \delta x + \delta v_j^T \hat{x}\right) f'\left(\hat{v}_j^T \hat{x}\right)$$

We now compute the covariance of an output.

Note: m is the index of node y_m and n is the index corresponding to y_n; consequently v_m and v_n are the weight vectors which link the inputs to respectevely y_m and y_n.

$$\begin{aligned} Y|_{mn} &= E\left[\delta y^T \delta y\right] \\ &= E\left[\left(\hat{v}_m^T \delta x + \delta v_m^T \hat{x}\right) f'\left(\hat{v}_m^T \hat{x}\right)\left(\hat{v}_n^T \delta x + \delta v_n^T \hat{x}\right) f'\left(\hat{v}_n^T \hat{x}\right)\right] \\ &= f'\left(\hat{v}_m^T \hat{x}\right) f'\left(\hat{v}_n^T \hat{x}\right) E\left[\left(v_m^T \delta x\right)\left(v_n^T \delta x\right) + \left(\hat{v}_m^T \delta x\right)\left(\delta v_n^T \hat{x}\right) + \left(\delta \hat{v}_m^T \hat{x}\right)\left(\hat{v}_n^T \delta x\right) + \left(\delta v_m^T \hat{x}\right)\left(\delta v_n^T \hat{x}\right)\right] \end{aligned}$$

Note: all the terms are second order terms.

if $m = n$:

$$Y|_{mn} = f'\left(\hat{v}_m^T \hat{x}\right)^2 \left(\hat{v}_m^T X \hat{v}_m + 2\hat{v}_m^T U_m \hat{x} + \hat{x}^T V_m \hat{x}\right)$$

if $m \neq n$:

$$Y|_{mn} = f'\left(\hat{v}_m^T \hat{x}\right) f'\left(\hat{v}_n^T \hat{x}\right)\left(\hat{v}_m^T X \hat{v}_n + \hat{v}_m^T U_n \hat{x} + \hat{v}_n^T U_m \hat{x}\right)$$

Where

- X and Y are the covariance matrices for δx and δy.
- U_n is the covariance matrix of δv_n and δx.
- V_m is the covariance matrix of δv_m.

We can now distinguish the three sources of error as mentioned below:

- the first term in X is due to the propagation of the error present in the input.
- the term with the inverse of the hessian is the error added because of the errors in the weights.
- the terms in U are due to further error due to the imperfect weights applied to imperfect inputs.

Let us know see how to implement this results in order to use them to predict the covariance of our outputs.

Once the training has been carried out, we have to use the above formula.

- First term: $\hat{v}_m^T X \hat{v}_m$

$\hat{v}_m$ is a vector with the weights of the first layer of the Network.

X is the covariance matrix of the noise vector. This means that an assumption has to be made about the noise model present in the inputs. This assumption can be checked with the same methods than the for the noise in the outputs. But very often this noise is known thanks to sensors and no further assumptions have to be made. This term can be computed easily.

- Second term: $\hat{x}^T V_m \hat{x}$

This term can be obtained with Bayesian techniques (see [d] pp 397-398). What is actually done in this context is that V_m is equal to the Hessian of the trained NN. Once the Hessian has been computed (many software packages can compute such a Hessian) this term is also easy to compute.

- Third term: $\hat{v}_m^T U_m \hat{v}_m$

U_m is the covariance matrix of δx and δv_n.

This term can be considered as negligible. It is indeed sensible to consider that the probability distributions of the errors in the weights and in the inputs are independent.

8.3.5.4 Example of formulae use

In this example we are going to illustrate what we described previously. Since the formulae look a bit complicated and the indexes may be confusing, it is probably useful to explain how they work.

Let us consider the following MLP with three inputs and five hidden units.

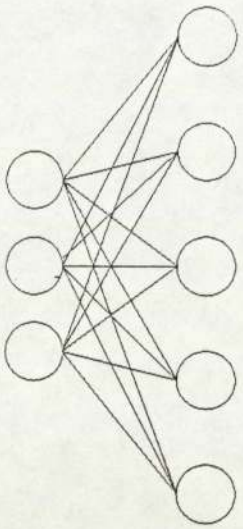

Figure 7: Example MLP for perturbation model computation.

This is the network for which we will compute the covariance matrix of the outputs.

The network is trained with a data set with some noise on the inputs. We will compute the covariance matrix of the outputs. This will be a 5*X*5 square matrix. The expression of a generic term of the covariance matrix is:

$$\text{if } \underline{m=n}: \qquad Y|_{mn} = f'\left(\hat{v}_m^T \hat{x}\right)^2 \left(\hat{v}_m^T X \hat{v}_m + 2\hat{v}_m^T U_m \hat{x} + \hat{x}^T V_m \hat{x}\right)$$

$$\text{if } \underline{m \neq n}: \qquad Y|_{mn} = f'\left(\hat{v}_m^T \hat{x}\right) f'\left(\hat{v}_n^T \hat{x}\right)\left(\hat{v}_m^T X \hat{v}_n + \hat{v}_m^T U_n \hat{x} + \hat{v}_n^T U_m \hat{x}\right)$$

We have already explained why the terms in U_n could be discarded. Two kinds of terms have to be computed: $\hat{v}_m^T X \hat{v}_n$ and $\hat{x}^T V_m \hat{x}$.

- $\underline{\hat{v}_m^T X \hat{v}_n}$

X is the covariance matrix of the noise. It can be computed with any statistical package once the noise has been estimated by sensors or a model has been assumed. The indexes of the two hidden unit output nodes are m and n. The two weight vectors that are used here are those inputting the two output nodes as it is shown on the figure below.

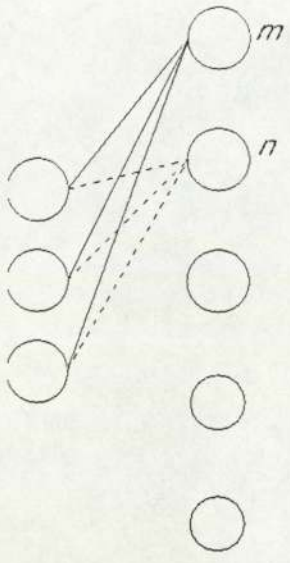

Figure 8: Example NN with only the links relevant for the output covariance computation.

The solid lines are the weights of vector v_m and the dashed line corresponds to v_n.

For every output node all there is to do is to take these two vectors and to compute their product by X like in the formula. The dimensions always fit for MLPs.

- $\hat{x}^T V_m \hat{x}$

This term has to be computed only for values on the diagonal of the covariance matrix. V_m is actually the hessian of the MLP because of the analogies with the bayesian framework (see [d], chapter 9). This hessian can be computed with a statistical package. But attention has to be paid to the fact that in our formula only the values of the hessian corresponding to inputs are needed (and not those of the biases and outputs). Once the hessian contains only input derivatives, the dot product of this hessian by the input vector is rendered possible since the dimensions will fit.

8.3.5.5 Application of a perturbation model to RBF networks

Unlike the layers of an MLP, the two layers of an RBF network perform different functions:

- the first layer performs a non-linear expansion of the inputs
- the second layer performs a weighted sum of the hidden layer output values

In the first layer there are two sources of errors:

- an error present in the inputs: δx
- an error in the center position: $\delta \mu$

The Taylor expansion yields a formula like the one for the MLP. This formula would be a bit more complicated than the one for MLPs but all the terms are however relatively easy to compute (see [u] and [v]).

8.3.5.6 Other methods using input noise

Input noise is not only used in perturbation models. Other methods which are used in NN take it into account. For example the learning method with uncertain data (see paragraph 8.5.5.2) uses input data and in that framework it can be shown that in some cases, input noise can be assimilated to output noise.

Another learning method called clearing (see paragraph 8.6.3) manages to compute the ratio between the input signal and the input noise, and then reduces the complexity of the network.

8.3.6 Noise added to improve generalisation performance

It can sometimes be appropriate to use a noise model in order to improve the generalisation performance of the NN. It consists in adding a noise to the input vectors during the whole training process. For sequential training a different noise vector is added to each input vector before it is presented to the network. For batch training each data point is replicated a number of times and a new random noise vector is added to each copy.

The effect of this noise is very easy to understand heuristically. It will "smear out" each data point and hence make it more difficult for the network to fit too precisely the input points. It is a way to reduce overfitting. The best choice for this noise is surely gaussian noise and its variance should not be too great.

Noise can be added to the activation as well as to the error function, in order to jump out of a local minimum.

Three important points to note are that:

- A different noise value must be added to each value each time it is used by the network. It is not sufficient to simply build a new training set to which noise has been added. Noise must be added dynamically during learning.
- Adding noise to the training data must preserve the mean value of that data. That is to say that the noise has to have zero mean.
- Of course the noise must be turned off in the testing set.

8.4 Multivalued function detection

An important cause of failure in the NN context are multivalued functions. When we indeed try to map a data set with a NN without being aware that this data set was generated by a multivalued function then our NN will not work properly. That is why we have to pay attention to this problem, since once it is known that a data set was generated by a multivalued function, then some appropriate NN solutions exist to map these functions. It is such a method that we will propose in what follows. This method will play a key role in our assessment of the data that is used to train a NN.

In [D5] a visualisation method based on histograms and dimension reduction was proposed but this method is not very effective since it is based only on the data and does not require any training of the network. But for problems involving many inputs and outputs it is very often hard to ascertain whether in some regions the function is multivalued. Visualisation techniques are indeed impossible to use for problems involving more than a few variables.

8.4.1 Forward and inverse problems

For many potential NN applications (control of industrial plants, analysis of spectral data, topographic reconstruction) there exists a well-defined forward problem. This forward problem is characterised by a functional (i.e. single valued) mapping. In practical applications however, we often have to solve the corresponding inverse problem. These inverse problems are often multivalued, that means with input values for which there are several valid output values (i.e. input values for which output distribution is peaked in more than one place)

If a least squares technique is applied to an inverse problem it will approximate the average of the target value and will lead to very poor results.

The example below gives us an illustration of that.

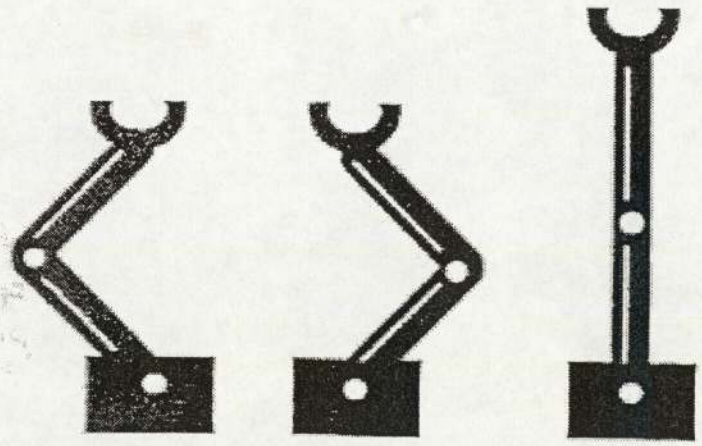

Figure 9: Example of forward and inverse problems.

Here the forward problem is to compare the position of the "hand" (semi-circle on the top) for a given pair of joint angles. The inverse problem however would be to know the value for the pair of joint angles given the position of the hand. As shown on the two drawings on the left there are clearly two different values for the angle for the same position of the "hand". If this problem was tackled with a least-squares method the position of the angle that we would get would be the mean of the two angles that is what is represented on the right hand side drawing. This is completely wrong.

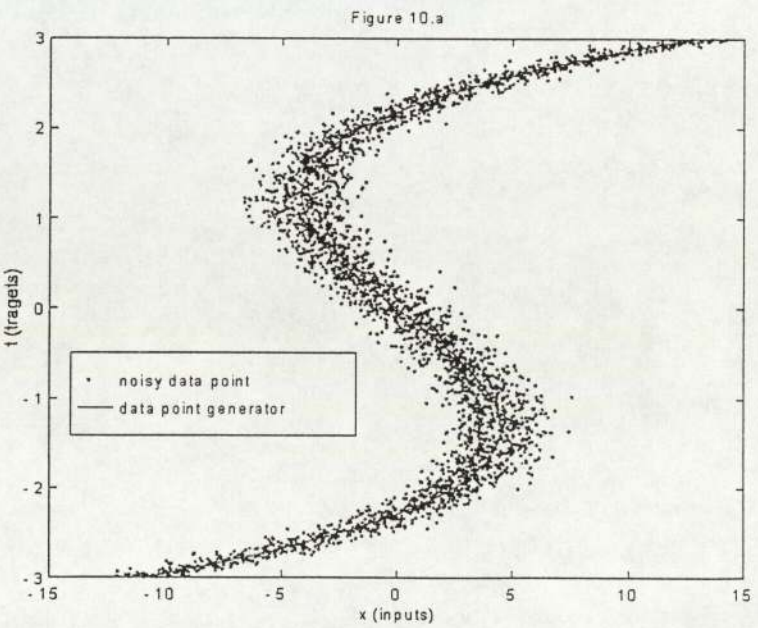

Figure 10: Example of a multivalued mapping function.

This is an example of a multivalued function corresponding to an inverse problem (not the one above though). To spot the forward problem the curve has to be rotated by 90 degrees.

This problem cannot be solved by changing the structure of the network or choosing another algorithm since it is a fundamental consequence of the use of a sum-of - squares error function.

8.4.2 Forward-inverse model

This is a method that can be used when one wishes to model the function only on one of its branches.

To solve our problem we will propagate the error through a forward-inverse model, that is a combination of two networks. This networks will be described in the guidelines below which explain how to carry out a forward-inverse training.

1. Train a network (network 1, see figure 11) the direction in which the mapping function is not multivalued. Once this network has been trained, its weights should no longer change.
2. Build a second network (network 2) which is to learn the inverse problem. This is the network which will finally be used.
3. To train the new network (network 2):

- Present each input and produce an output from the network being trained (network 2).
- Pass the outputs of network 2 through network 1.
- The error is *(n-i)* since the outputs of the double model are equal to its inputs.
- The error is propagated through network 1.
- The error at the end of network 1 is going to be propagated through network 2 (and not the error at the output of network 2).
- Update the weights of network 2. Nothing changes in network 1.

4. Once the training has been completed, network1 can be discarded.
5. Network 2 maps the multivalued function on one of its branches.

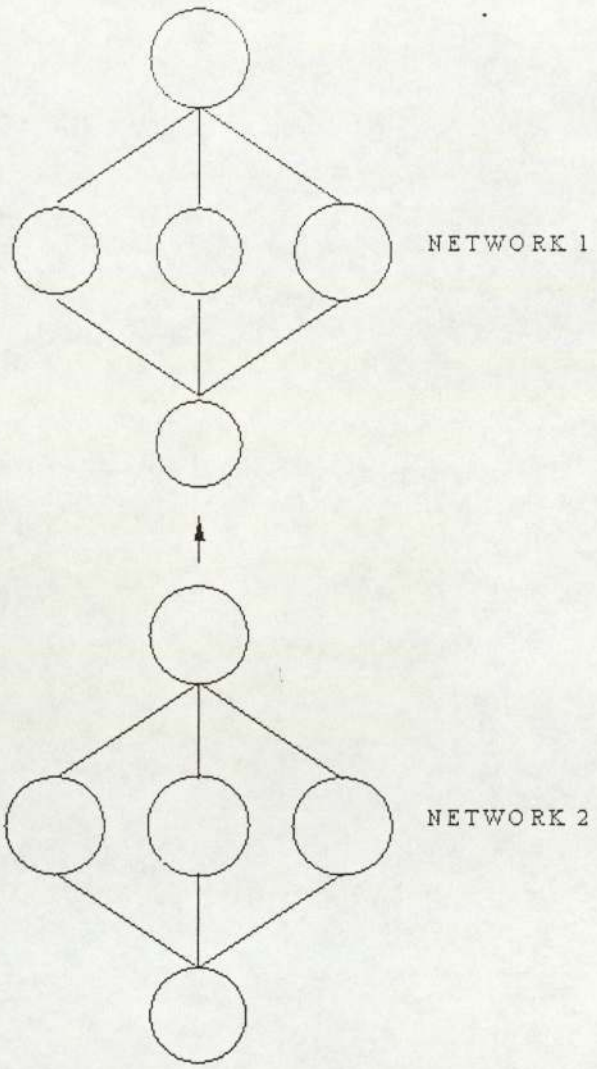

Figure 11: The two NN of the forward-inverse model.

Network 1 is the network which has been trained with the forward problem and network 2 is he network that is being trained.
For details about this method see [t].

Now a more powerful method will be presented. This method will allow to model the whole multivalued function.

8.4.3 Conditional distribution modelling

The basic goal of training a feed forward neural network is to model the statistical properties of the data-generator expressed in terms of class-conditional distribution. function $p(t|x)$.

A sum-of-squares error function as we saw it above (see also [D2]) is a model of the data in terms of a gaussian distribution with a global variance parameter and an x-dependent mean. But this model is inappropriate if the data has a complex structure.

A very powerful means of modelling conditional distributions is based on the use of mixture models (this is one of the applications of the mixture models to which we alluded in the density modelling paragraph).

The Mixture Model represents the output data distribution as a linear distribution of adaptative kernels. In this context the conditional density is

$$p(t|x) = \sum_{j=1}^{M} \alpha_j(x)\phi_j(t|x)$$

where

$$\phi_j(t|x) = \frac{1}{(2\pi)^{c/2}\sigma_j(x)} \exp\left(-\frac{\|t - \mu_j(x)\|^2}{2\sigma_j^2(x)} \right)$$

In practice we are going to use a NN which will give us the parameters (mixing coefficients) $\alpha_j(x)$, $\mu_j(x)$ and $\sigma_j^2(x)$.

This network can be any feed forward network (as it has universal approximation capabilities). We can for example use an MLP with a hidden layer of sigmoidal units and an output layer of linear outputs.

Then finally to obtain our model for the conditional density we will have to minimise a negative log likelihood error function E with respect to the parameters of the NN ($\alpha_j(x)$, $\mu_j(x)$ and $\sigma_j^2(x)$).

The error function is given by:

$$E = -\sum_n \ln\left\{ \sum_{j=1}^{M} \alpha_j(x^n)\phi_j(t^n|x^n) \right\}$$

And to obtain what we are seeking with a NN, we can use a standard backpropagation procedure since we know the expression of the derivatives of the error function with respect to the parameters of the outputs of the nets (see [d] for the expressions of the derivatives).

The minimisation of this model with respect to the parameters of the NN will lead to our model of the conditional density of the target data.

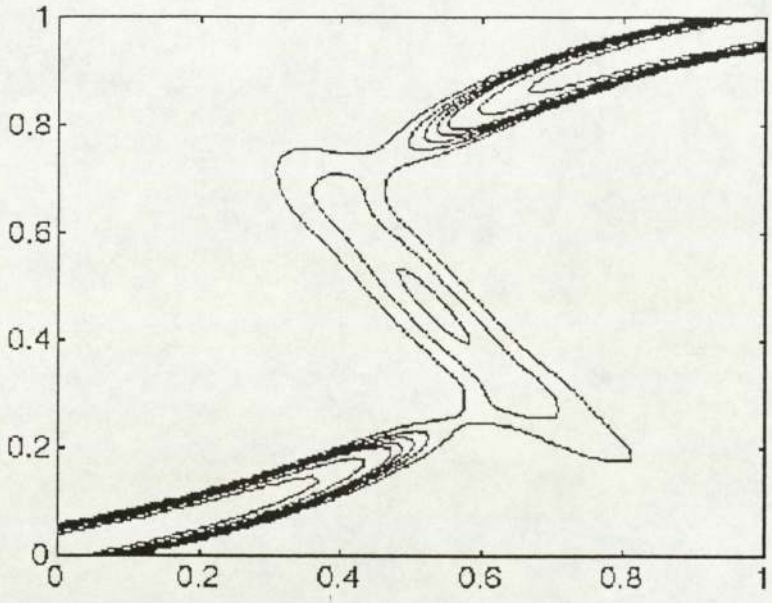

Figure 12: Conditional density obtained with a mixture density network.

This figure shows how the conditional density that we obtained with a Mixture Density Network models the multivalued function of figure 10. These are several gaussians. Three different areas can be observed:

- **from 0 to 0.25: only the first gaussian has an influence that means that only the mixture coefficient for this gaussian is non zero.**
- **from 0.25 to 0.80: all three gaussians of the mixture model play a role. The three mixture coefficients are different from zero.**
- **from 0.80 to 1: only the last mixing coefficient is non zero.**

Actually this methodology allows to model mutivalued distributions but not to detect if a function is multivalued without a training.

Summary

A NN is used to determine the coefficients of the mixture model. These coefficients are the coefficients of the mixture model of the conditional density. Such a network is sometimes called a mixture density network in the literature. The conditional densities that these networks allow us to obtain represent a complete description of the generator of the data.

8.4.4 Mixture density networks and noise: a conclusion over noise

Mixture density networks make it possible to model any data generator; they can thus also be used to model a noisy data set. Neural networks with sum of squares error functions model a data set with constant variance and x dependent mean. With a mixture density network however, for any given value of x the conditional density $p(t|x)$ can be modelled. This can be used to model complex noise.

We have now reached a point were almost all the kinds of noise which corrupt our data can be tackled. A short summary of the techniques we examined is useful.

- When a normal distribution is assumed for the noise, this assumption can be verified with graphical and statistical tools. Similar tools exist for other distributions as well.
- The influence on the outputs of a noise in the inputs can be computed with a perturbation model.
- The mixture density network that we used for multidimensional mapping modelling can also be used to model noise since such a network makes it possible to model any kind of data generator as far as the inputs are concerned, and thus a data generator with noise. This can be used to model complex noise.
- In the Bayesian context that was alluded to in [D2] and in appendix A of this document, the noise is modelled by a conditional density that corresponds to the likelihood. This means that noise actually is modelled in the weights of the NN.

In a word, all the noises that could possibly exist in a training set fall in the scope of one of these domains that we have studied.

8.5 "Representativity" of the data set

In this section we suppose that the study of the noise and of the eventual multivalued character of the function we want to map have been studied (see previous paragraphs). What interests us now is to see if we have enough data points, if these points are representative of our problem and what further checks could be carried out in order to improve the quality of the data as much as possible. We will also discuss how to quantify uncertainty.

8.5.1 Size of the data set

In the [D3] Guidelines we often alluded to a "large" data set or we wrote sentences like "if the data set is large enough" (see [D3]). Nevertheless it is known to be very hard to quantify these ideas. The size of the data set is a relative quantity that can only be compared with the problem that we want to solve using this data. In our validation context we have to distinguish between two situations:

- The data set of the NN software that we have to assess is very small. In such a case we have to check if it has been handled properly.
- The data set is huge in which case we have to ensure that dimension reduction and point discarding have been carried out properly.

In a classification problem, the ideal situation would be that we have an equal number of data points for each class even though we know that there will be a lot more outputs in some classes than in some others.

If this is not the case, we can chose between the alternative with highest performance among these ones:

- Discard some points for the classes where there are too many points so that the training can be carried out with the same number of points in each class; this number being the minimum of all the numbers of points in all the classes. In such a case it is necessary to adjust the posterior probabilities. (see [D2] and Appendix A)
- Another possibility is to train the network with all the available points. If there are enough points in the under represented class the training could nevertheless be good.

The number of points that we need depends also on the level of noise. The noisier the data is, the more points will be needed to carry out a good training.

In a regression problem the main problem is the density of the data in the training set.

8.5.2 Data density

For regression problems particuliar attention has to be paid to whether there are some regions of the input space where there is less data than in others or even no data at all. This can be detected with visualisation techniques for example.

But once such a problem has been detected, most of the time there is no further data available and hence it is impossible to tackle the problem. The only solution that we then have is to take this problem into account in the computation of our error bars (see section 9.2). This will make us aware of the fact that in the regions of lower density, the results are likely to be less accurate than in other ones. The error bars will thus be broader. This will hence make it possible for us to use a network which was trained with data of different densities. In this data density context a distinction can nevertheless be drawn between MLPs and RBF networks.

- MLPs are more efficient in low density regions. They are better in extrapolation tasks than RBFs.
- RBFs are more local oriented NN.

8.5.3 Feature extraction and dimension reduction

This section is relevant for cases where the training set is very large or has a very large dimension so that it has to be reduced to make it computationally less expensive. We seek some standard way of doing that with a range of classical tools.

8.5.3.1 Normalisation and whitening

Normalisation is a very well-known technique. It consists of applying a linear transformation so that all the inputs have similar values: For each variable x_i, its mean and its variance are computed with respect to the training set:

$$\bar{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_i^n \qquad \text{and} \qquad \sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left(x_i^n - \bar{x}_i\right)^2$$

where $n = 1,\ldots,N$ are the pattern labels. We can then define a set of rescaled input patterns.

$$\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i}$$

The transformed variable has zero mean and unit variance. But normalisation has one major drawback: each of the input variables is treated as independent from the other ones. To tackle this drawback we can use a technique called whitening.

Let us group the input variables into a vector $x = (x_1, \ldots, x_d)^T$ which has mean vector and variance matrix given by:

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x^n$$

and

$$\Sigma = \frac{1}{N-1}\sum_{n=1}^{N}(x^n - \bar{x})(x^n - \bar{x})^T .$$

We then consider a vector of linearly transformed input variables given by:

$$\tilde{x}^n = \mathrm{A}^{-1/2} U^T (x^n - \bar{x})$$

where U is the eigenvector matrix of Σ.

In the transformed coordinates, the data set has zero mean and a covariance matrix which is given by the unit matrix. In a word whitening is a more sophisticated linear rescaling which takes into account the correlations between the variables.

Other techniques that can be used are PCA and factor analysis but their description is out of the scope of this document. Information about them can be found in [q].

8.5.3.2 Removing the less useful variables

8.5.3.2.1 Statistical test

It makes sense in a huge data set to discard individual variables that are highly dependent on other variables. When there is no specific knowledge about which variables should be discarded, analytic methods should be employed.

Statistical tests can be used. For classifications tasks for example a *t*-test could be used to compare the distribution of each variable in each of the different classes, using only those who differ significantly.

The independence between the different input variables may be measured by calculating the covariance between each pair of variables. On the covariance matrix the highest values are those of dependent pairs of variables and these may be discarded.

8.5.3.2.2 Neural Network solution

Some less brute force and potentially more effective methods exist and they are based on NN. It consists in building a NN with too many inputs and a small number of hidden units. If the weights in the network are then initialised with small random values then there will be no pressure from the input variables for the weights from the less useful input variables. Consequently the input units in the trained network which have not moved far from their original values may be most safely discarded. A new network can then be trained with only the new points (it can be verified with the new one).

Another even better method to assess the relative importance of different inputs is automatic relevance determination. It is a Bayesian technique based on the use of a separate regularisation coefficient for each input. If a particular coefficient acquires a large value, this indicates that the corresponding input is irrelevant and can be eliminated. To get more information about this technique see [d].

8.5.3.2.3 Information theory: entropy based analysis

Information theory can be used to select a set of input variables for a NN. By measuring the mutual information between each input variable and the set of target outputs in the training set one is able to select the single feature with the most predictive power over the outputs. Further features may then be selected by two criteria. A new feature must predict something about the outputs but it must not predict much about the input variables already in the set. A new variable must share a lot of mutual information with the outputs but not with the inputs. By applying these techniques it is possible to choose a set of variables which are independent and which predict the output data well. This technique is not optimal but it can reach a usable solution in a short time.

Here are the different entropies that can be calculated and their use:

- Entropy between any number of input units:

This provides a measure of independence. A good input set has high unit independence.

- Entropy of an output vector given an input vector:

It measures how well the output data may be predicted.

- Entropy of the input vector given the output vector:

If the entropy of the input vector given the output vector is higher than that of the output given the input, then the problem may be ill posed.

- Entropy of any single output unit given the input vector:

It provides a measure of how well the input data is able to predict the output unit's value.

- Entropy of any single input given the output vector:

It provides a measure of how well it independently predicts the output vector.

8.5.3.2.4 Outlier removal

Some data points are outliers not only due to a given variable value but due to the vector as a whole. In isolation they cannot be detected despite being atypical. These points often show up in a plot of the first or two of the three principal components.
If data is scarce and the act of discarding an example simply because one of its elements is an outlier is to be avoided, outlying values may be replaced with some other value: the mean for that variable is a good choice. But extra care must be paid during the testing phase to ensure that these replacements have not biased the network if such a strategy is adopted.
The input density can also be modelled with the techniques described earlier in this context and low density probability points can be considered as outliers.
[D2] has already dealt with these kind of problems. The techniques described in the previous paragraphs such as entropy computations can be used in this context. It has to be justified in a NN project how a large data set has been handled particularly if some points have been discarded and if so, the reason why some points have been discarded rather than others has to be given.

8.5.4 Time series data

Time series forecasting and analysis is a domain in which NN are often used. The tasks we encounter are regression and sometimes classification. These problems are the usual ones that NN can solve, and in that respect time series problems do not differ from other problems using NN. However, a special section is devoted to time series problems because they present a lot of peculiarities in the pre-processing phase. Several techniques have to be applied to a time series data set before it can be used, and attention has to be paid to this pre-processing in order to get the best performance. In this context, pre-processing plays such a key role that it needs to be assessed. The

assessors will have to check if appropriate manipulations have been carried out on the data set. The main problems that will have to be addressed are:

- to check whether the data distribution is normal if it was assumed so and what transformations can be carried out to render it more normal.
- to check whether the data is stationary or not and if not to find the most effective way to render it stationary.

8.5.4.1 Normality

The data set is often assumed to be normal. Most classical techniques in statistics are derived under the assumption of normality of data; thus it becomes convenient to transform the original data so that transformed data satisfies the assumption of normality.

Normality can be checked with one of the following methods, all of which have already been discussed in this document (see paragraphs 8.3.3 and 8.3.4 of this document for details).

- Histograms: they best describe the skewness in the distribution.
- Normal quantile plotting (see paragraph 8.3.3).
- Statistics: two statistical tests were described in paragraphs 8.3.4.1 and 8.3.4.2 which permit to check the normality of a distribution.

8.5.4.2 Box-Cox transformation to achieve normality

A distribution can be made approximately normal using some transformations. One of the most efficient ones is the Box-Cox transformation. The Box-Cox transformation is a general class of transformations to attain normality; each input point X_t is transformed according to the following formula:

$$\frac{X_t^m - m}{m}$$

where m is chosen by the user and is used to achieve normality. This choice is empirical but it can be estimated by some statistical packages.

The values of $m < 1$ are useful for data skewed to the right, while values of $m > 1$ are used for data skewed to the left. For skewed data, simple transformations such as power or logarithm can be used to remove asymmetry. For more information on the Box-Cox transformation see [f].

8.5.4.3 Stationarity

There is a major difference between usual statistical data and a time series problem. In most statistical problems, random sampling procedures make it possible to obtain replicated observations under identical conditions. In a time series problem, however, at each point we are faced with only one single value for each point in time, and observations which are dependent over time. In order to use some inferential statistics we must be able to recreate some notion of replicability. To do this, the notion of *stationarity* is indispensable (see [k] for more information).

A time series $\{y_t : t = 1,2,...\}$ is said to be stationary if the joint distribution of any collection of k values is invariant with respect to arbitrary shifts on the time axis. This definition is actually not used in practice since it cannot be tested. What is used in practice is the notion of "weak stationarity" (although it is often called stationarity). A time series $\{y_t : t = 1,2,...\}$ is said to be weakly stationary or second order stationary if the mean, the variance and the covariance are time independent. From now on, stationarity will mean weak stationarity.

A stationary time series is characterised by the fact that the statistical (stochastic) portion of the series does not depend on the time but rather depends on the difference between time points. In particular, the series will have the following properties:

- Constant mean.
- The correlation between values at two different time points depends only on how far apart they are.

We may suspect that the series is not stationary if:

1. The mean level is not constant over time or,
2. The extent of deviations from the mean is changing over time.

If such departures from stationarity are evident, it is reasonable to adjust the series by applying a transformation that improves the stationarity without destroying the essential features of the series.

The airline data set of figure 13 shows an example of a non-stationary time series.

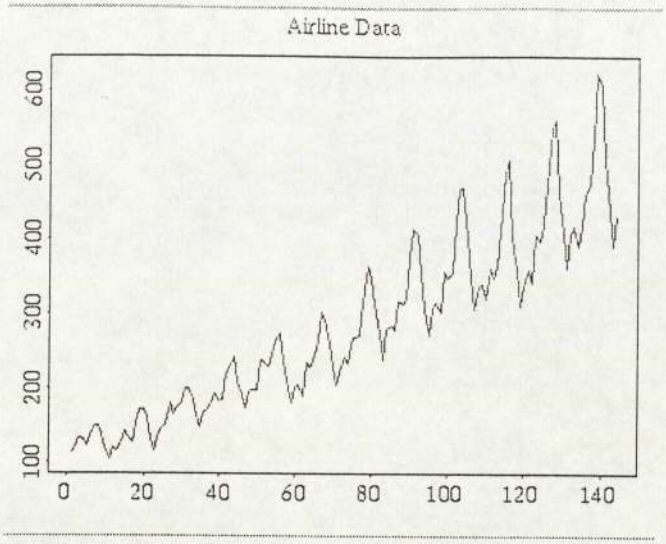

Figure 13: Airline data set.

This airline data set represents the evolution of the total number of millions of miles flown by American aircraft per month (month number 0 is january 1963). This time series exhibits increasing mean and increasing variance: it is not stationary.

A first step towards stationarity could be the logarithmic transformation. It is generally appropriate whenever an analysed series has a variance increasing linearly with mean. Logarithmic transformation helps eliminate the changing variance, as it is shown on the figure below.

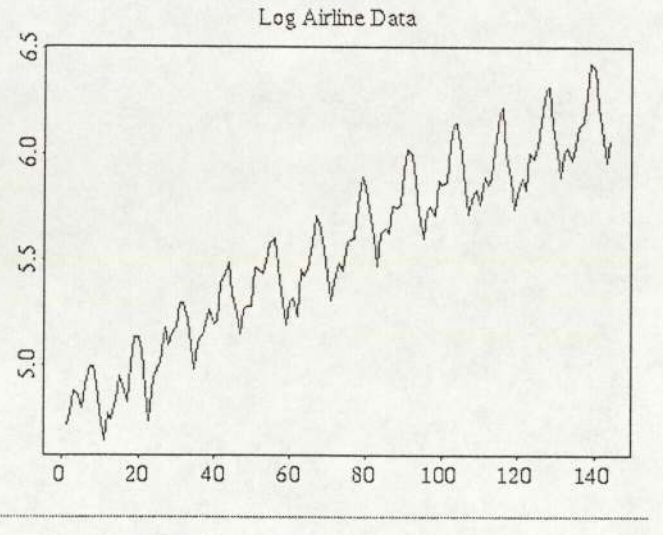

Figure 14: Airline data set after log transformation.

This is the same time series as figure 13 but it has been transformed by a logarithmic transformation. The data in the log scale seems to have a more stable variance, but the assumption of constant mean is still violated by the upward trend. Hence the transformed process is still not stationary.

Another useful means of checking whether a function is stationary or not is to use its Sample Auto Correlation Function (SACF). It is an inferential tool which makes it

possible to describe the behaviour of an observed series. Let us now give the definition of the SACF.

The sample variance is defined as

$$\hat{\gamma}_0 = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2$$

where y_i are the data points and $\bar{y}$ is the mean of these points.

The k^{th} sample covariance is

$$\hat{\gamma}_k = \frac{1}{n}\sum_{i=1}^{n-k}\left(y_i - \bar{y}\right)\left(y_{i+k} - \bar{y}\right)$$

the k^{th} sample correlation is then:

$$r_k = \frac{\hat{\gamma}_k}{\hat{\gamma}_o}$$

This definition corresponds to the maximum likelihood estimator when the data is normally distributed.

If the series is stationary the values of the SACF decrease quickly and are low (see [f] and [m] for more details), thus it is a quite straightforward means for checking stationarity. On the figure below for example, it allows us to draw the conclusion that the series is not stationary.

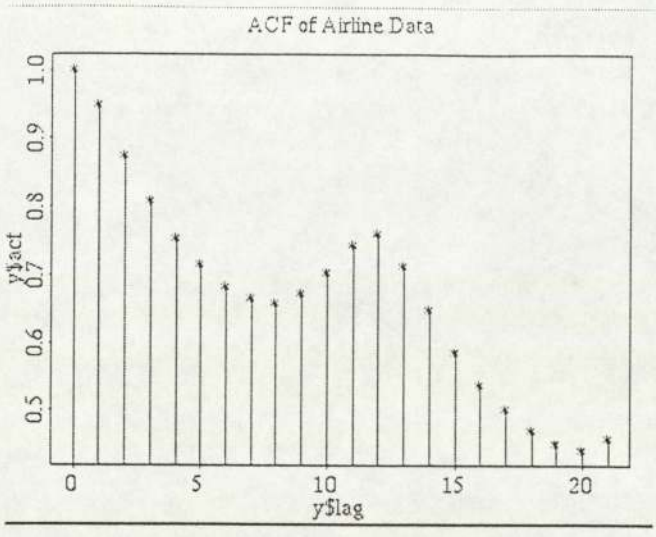

Figure 15: SACF plot of the airline data.

This is a plot of the SACF of the airline data set. The values are high and thus the series is non stationary.

As we have seen in the examples, by using the log transformation, we have stabilised the variance of the series, but the trend in the mean is still violating the stationarity assumption. The following parts of this section will discuss the methods of achieving stationarity, which are usually more effective than simple logarithmic transformations.

8.5.4.4 Detrending

Detrending is a way of removing trends and seasonality, two of the main causes of non-stationarity in a time series. It consists in subtracting an estimated trend and/or an estimated seasonality component from the data. Time series data is detrended because it is usually the local behaviour which is modelled, unencumbered by long term effects such as trends or seasonality, as a stationary stochastic process.

There are two methods of detrending:

1. Regression
2. Moving Average

- Regression as a method of detrending is generally used for removing linear trends over time (this can be carried out by the simple linear regression of data on time), and correcting for seasonality.
- MA detrending is based on the smoothing properties of the moving average model. MA (in a Moving Average Model of order q the measurement of each point is a weighted sum of the q previous white noise. Such a model is by construction stationary) smoothing of the data can successfully extract any linear and non linear trend but may not be able to deal with seasonality.

Choice of the method of correcting for trend is usually chosen by visual examination of the plot of the data.

- If the data exhibits trend close to linear, simple linear regression on time may yield a good result.
- If the trend is obviously non linear, MA smoothing is more effective in estimating the trend.

To illustrate what can be done with regression, let us take a Manufacturing Shipments Series data set: it represents a monthly evolution of the quantity of manufactured shipments. The scales of the axis are not of first relevance. This is a non stationary series with mean increasing over time. So, to attain stationarity, elimination of a non linear trend might be necessary.

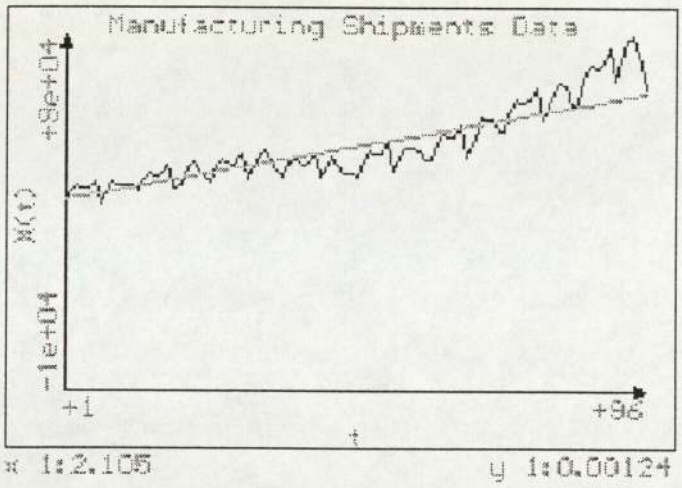

Figure 16: Manufacturing shipment data set with linear regression estimated trend.

The series is clearly not stationary. If we assume that dependence between the mean and time is linear (or close to linear), linear regression can be used to estimate the trend. After subtracting this line from the original data we obtain detrended Manufacturing Shipments series (see below), which is closer to Stationarity but still shows departures from stationarity in mean. This is perhaps due to the non linearity in the trend. The linear trend estimated by linear regression over time is the light line on the plot.

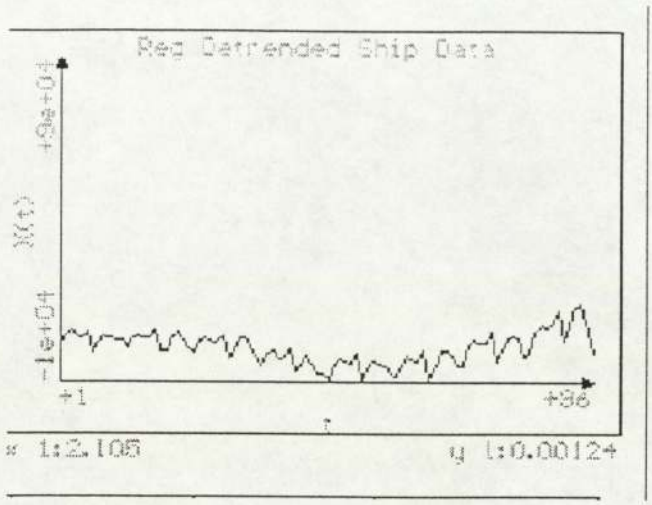

Figure 17: Detrended manufacturing shipment data.

The regression detrended series is obtained as a difference between original series and an estimate of the linear trend. It has to be noted that the detrended series is still non stationary. This is not surprising since the original series exhibits a non linear trend.

Analysis of the SACF can then be a very important part of the data analysis. If a series is non stationary, the SACF may decay very slowly as we have it in the Manufacturing Shipments series example. In this case all lags are high. (see figure 18).

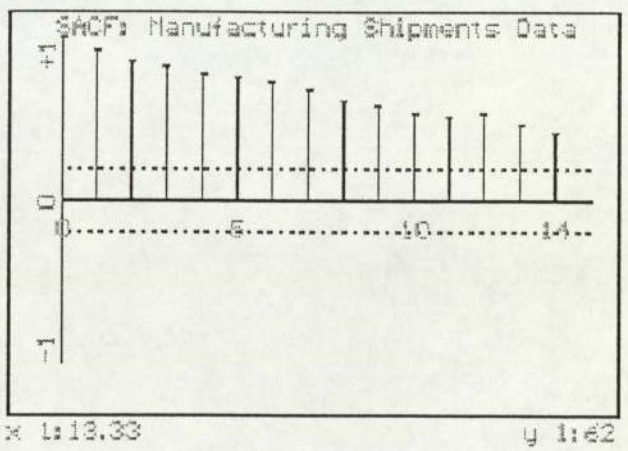

Figure 18: SACF plot of the manufacturing shipment data set.

The SACF of manufacturing shipments series is typical for a non stationary process (i.e. slow decay and high lags). The dotted line is the threshold above which the SACf is significantly different from 0.

Let us now analyse the SACF of the detrended data.

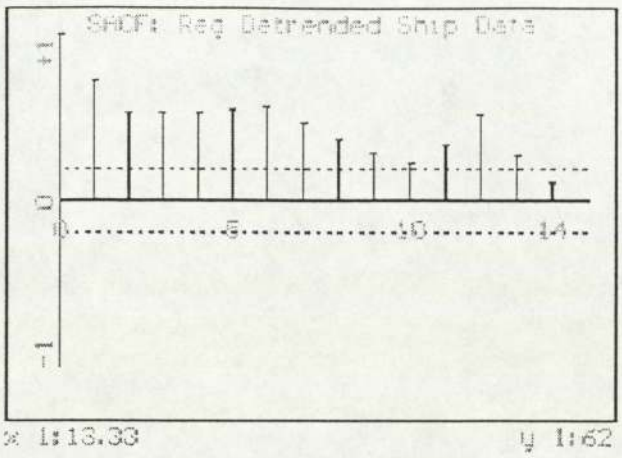

Figure 19: SACF plot of the detrended manufacturing shipment data set.

The SACF of the detrended series is decaying more quickly although it is not completely satisfactory. It does not have as a long tail as the SACF of original series did. SACF at some lags are statistically significant. For example, the significant value at lag 12 tells us that this series exhibits a yearly periodic behaviour. But the number of lags above significance level still indicates the lack of improvement.

Another way of extracting the trend from a series is by using the Moving Average (MA) method. Moving average method of estimating the trend involves averaging successive observations from the series.

We may for example fit a polynomial to the first $(2m+1)^{th}$ points in order to find the trend value at the $(m+1)^{th}$ point (point of the middle of the set; that is why an odd number of points is taken), and the same thing is then done for all the other points of the series. In this case the order of the MA model would have been $2m+1$. Usually this order is determined by an automated procedure on a computer (see for example the Box-Jenkins methodology on paragraph 8.5.4.7)

This method is particularly useful when the data exhibits a non-linear trend, since in a MA model, every part of the time series is represented locally by a polynomial. Let us consider the same shipment data example.

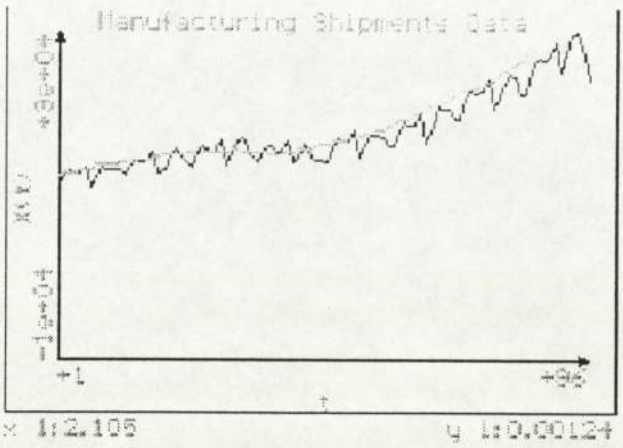

Figure 20: Manufacturing shipment data set with MA estimated trend.

Moving Average provides a smoothed version of the original data. The higher the order of the MA process is, the higher the degree of smoothing. Therefore a high-order MA provides an estimate of the trend in the data. The trend estimated by a MA(5) follows the data structure more closely, providing a more accurate line on the figure.

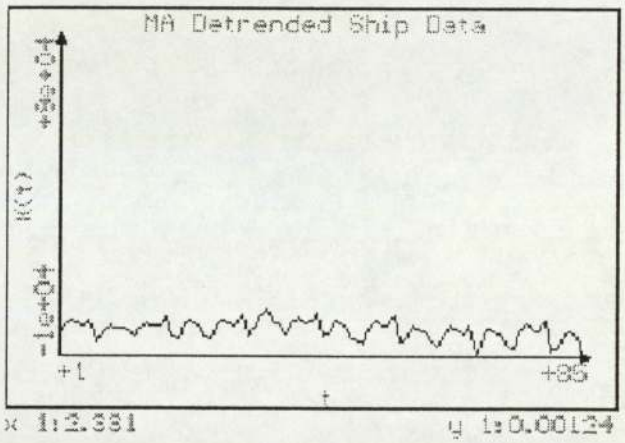

Figure 21: MA detrended manufacturing shipment data set.

Hence, after detrending we obtain the series satisfying stationarity assumptions (the MA detrended series is obtained by subtracting the estimated trend from the original data).

The SACF of the detrended data confirms that the detrended time series is stationary.

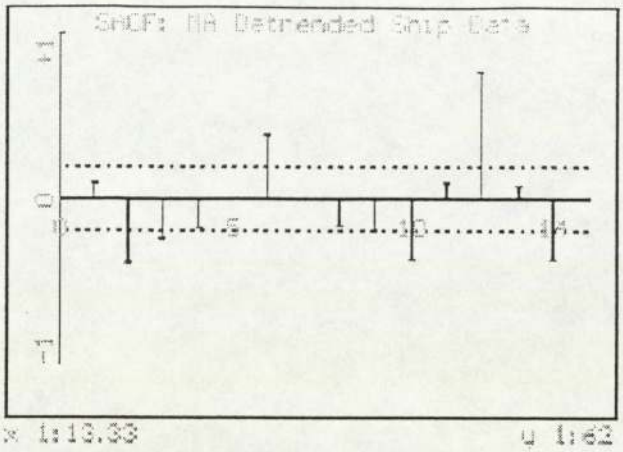

Figure 22: SACF plot of the MA detrended manufacturing shipment data set.

The SACF of the detrended series gives us a small number of significant lags, while the values at all other lags are diminished. The large value at lag 12 indicates that removing of seasonality may also be useful.

Comparison of linear regression and moving average methods of detrending for Manufacturing Shipments data indicates that MA detrending gives more satisfactory results than linear regression. For other series however, the linear regression method can also be used for eliminating trend and seasonality simultaneously (whereas MA cannot eliminate seasonality). Actually the best thing to do is to try both method and to choose the one that gives the best results. A plotting can give a hint of which one it could be.

Let us now say a few words about Correcting for seasonality. Generally elimination of trend and seasonality can be achieved in one step. A linear regression model with trend component is constructed under the assumption that the data depends linearly not only on time but also on the phase of the period. Depending whether the seasonal components are additive or multiplicative, various models can be assumed to get rid of the seasonal component. If m_t is the smooth component of the series, s_t the seasonal component and ε_t the error term, then the model can be

$$y_t = m_t + s_t + \varepsilon_t \text{ or}$$

$$y_t = m_t s_t \varepsilon_t \text{ or}$$

$$y_t = m_t s_t + \varepsilon_t$$

These values can then also be determined with appropriate software (see [k]) and the seasonal effect removed. As it had already been mentioned trend and seasonality can be removed together. A new method called differencing is particularly powerful for that.

8.5.4.5 Differencing

8.5.4.5.1 Definition

Differencing is very effective in removing both trends and seasonality (in the cases of polynomial trend). When a time series exhibits a polynomial trend together with seasonality it proves to be very effective to do so.

The first backward difference of a series is defined as:

$$\nabla X_t = X_t - X_{t-1}$$

∇X_t, the sum obtained by differencing the original sum once, is trend-free if the original series contains only a linear trend.

The second difference of a series is defined as:

$$\nabla^2 X_t = X_t - 2X_{t-1} + X_{t-2}$$

$\nabla^2 X_t$, the sum obtained by differencing the original sum twice, is trend-free if the original series contains at most a quadratic trend. Further differences at higher order lags are defined similarly.

It can also be useful to introduce seasonal differences.

$$\nabla_s X_t = X_t - X_{t-s}$$

In order to use differencing to remove trends and seasonality it ha to be decided how many differences have to be taken. Usually software is used to do that. Otherwise several solutions have to be compared.

8.5.4.5.2 Example

Let us illustrate the effect of differencing on the logarithms of airline data.

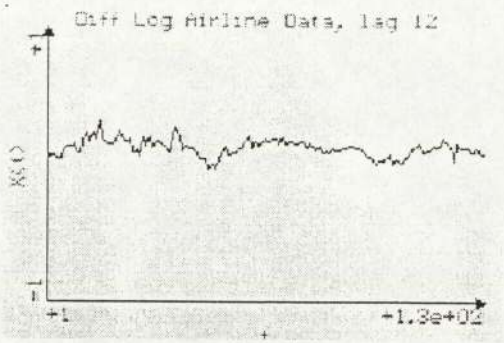

Figure 23: Differenced log airline data set.

To remove seasonality we apply differencing at lag 12. The differenced series has no seasonality but still exhibits some fluctuations.

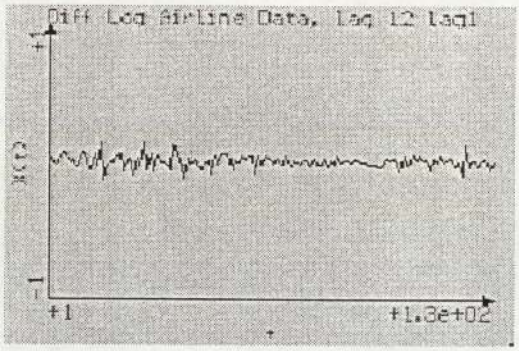

Figure 24: plot of the seasonally differenced airline data set.

The plot of $\nabla\left(\nabla_{12}\left(X_t\right)\right)$ shows that seasonality has almost disappeared. The SACF will have to be plotted to confirm this result.

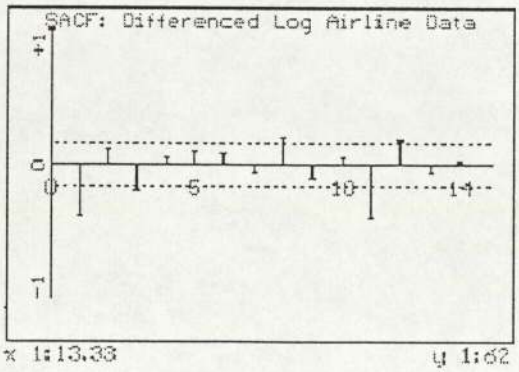

Figure 25: SACF plot of the differenced log airline data set.

The SACF has two significant lags, which may draw our attention, namely lag 1 and lag 11, but the global result is still good.

Differencing is an important tool to transform the data into a stationary series. It can eliminate trends and seasonality. It is often preferred to detrending.

8.5.4.6 Box-Jenkins Method

The Box-Jenkins methodology is one of the main algorithms which are used in time series analysis (see [k]). It consists of several steps, allowing one to fit a model to the

observed data and provides several methods of model testing. B-J modelling consists of three steps:

1. Identification: Using tools such as auto-correlation measures to identify models to explain appropriately transformed data.

2. Estimation: Estimating the parameters of the models identified in the previous step.

3. Diagnostics: Checking the adequacy of the model estimated in the previous step. If the module is not satisfactory, we start with a new model.

8.5.5 Missing data

8.5.5.1 "Usual" methods

What we mean here by missing data is when there are data points where some components are missing. We try to give a standard way of dealing with such problems. Several alternatives can be tried.

- discarding the vector with missing data. But this can lead to poor results because it assumes that the cause of the mechanism responsible for the missing data is independent from the data itself. Nevertheless this procedure can be adequate when there are only a few input vectors with points with missing components.
- another way to address this problem is to seek a maximum likelihood solution (see [d]) and to use the EM algorithm to find the input density. Once the input density has been found the conditional mean of the missing value can be computed. If $x = (x_1, x_2, ..., x_n)$ is the input vector, let us assume that x_1 is missing. Thanks to the density modelling methods $p(x)$ can be modelled. Thus $E[x_1 | x_2, ..., x_n]$ can be computed. The value obtained for this conditional expectation is the value that can be chosen for x_1.
- another technique based on the same input density modelling exists. Instead of using directly the conditional mean as in the method described above, a weighted integration across all possible values for that input is used (see [n]).

Whichever of the methods is used we must attach a measure of confidence to the NN outputs which reflect the influence of such a training point on a given generalisation point. We should manage to take this into account in the computation of error bars (see section 9.2).

Missing data is a field where a lot of research is still going on at present because there's not yet a satisfactory solution for every kind of problem. This is a presentation of a newer method.

8.5.5.2 Input uncertainty

In this section we deal with a new method to tackle these problems like noise in the inputs and missing data. This method can actually be more efficient for missing data problems than the averaging methods of the previous paragraph.

8.5.5.2.1 Framework

We consider a regression model with input points $\{x_1,\ldots,x_k\}$ and outputs $\{y_1,\ldots,y_k\}$. The outputs are given by: $y^k = f(x^k)+\varepsilon^k$, where ε^k is a zero mean random noise. To account for uncertainty it is assumed that no information can be got from x and that we can only obtain samples from z with

$$z^k = x^k + \delta^k$$

where δ^k is a random noise with standard deviation $P_\delta(\delta)^2$. A sum of squares network will approximate the expectation of the network. A network will thus approximate:

$$E(y|z) = \frac{1}{P(z)}\int f(x)P_\delta(z-x)P(x)dx \qquad (1)$$

And thus in general $E(y|z) \neq f(z)$

8.5.5.2.2 Different situations

Let us now assume that ε and δ both have gaussian distributions with respective means *0* and variances σ^y and σ. Several situations can occur.

- If the inputs are certain (that is there is no noise in the inputs) then we have that: $E(z|x) = f(z)$.
- If the inputs are uncertain (i.e. noisy). Equation (1) describes the convolution of *f(x)* with the noise process $P_\delta(z-x)$. The noise will therefore blur or smooth the

original mapping. In some special cases $f(x)$ can be recovered from a network trained on deficient data by deconvolution (but this can be very error sensitive).

- If a linear approximation is valid, then the input noise can be transformed into output noise. This means that if $f(x)$ is approximately linear over the range where $P_\delta(\delta)$ has a significant amplitude, we can substitute the noisy input and the network will still approximate $f(x)$.

8.5.5.2.3 Learning with maximum likelihood:

Now we will give the main steps of the modified maximum likelihood algorithm. More details (particularly about the mathematical background) can be found in [w].

- Train the network using the complete data set. Estimate the covariance of the noise in the targets.
- Estimate the input density $P(x)$ using gaussian mixtures.
- Include the incomplete training patterns in the training.
- For every incomplete training pattern, compute the maximum likelihood and its derivatives (with some integration tricks and approximations described in [w]).

What has to be emphasised is that this method can provide better results for missing data problems than the usual approximations with mean substitution. What can happen with the substitution method is that if the difference between network prediction and target is large, the error is also large and the data point contributes significantly to the gradient although it is very unlikely that the substitute value was the true input.

8.5.6 How to split the data set

8.5.6.1 Small data sets: resampling techniques

With small data sets it can happen that the whole data set has to be used to carry out a training successfully. That means that no data is left for a testing set.

Note: once again we use words like "small data set" and "successfully" which are pretty subjective. But as it has already been said they cannot be quantified since they are too closely related to a specific problem. This means, for example, problems where the data density is low over the whole data space or in a classification problem where we don't have enough points.

In all these situations we cannot afford the luxury of keeping aside part of the data set to test the NN with it.

8.5.6.1.1 Cross validation (CV)

In this case we can use a procedure termed cross-validation. The training set is split in *S* distinct sets. We then train a NN using data from the *S-1* of this small sets and then we test its performance by evaluating the error function with the remaining set. We then start again using another of the sets as testing set. The average error rates over all *S* partitions is the cross-validated error rate. The only drawback of this method is that it requires the training process to be repeated S times. But in our validation context this is not a major drawback. What has also to be emphasised is that the increase in computational speed of computers makes these techniques very practical today for large samples and complex learning systems.

8.5.6.1.2 Bootstrapping

Leave-one-out estimators (i.e. cross-validation) are virtually unbiased. That means that they can be applied to much larger samples than bootstrapping, yielding accurate results.

While leaving one out is nearly unbiased, its variance is high for small samples. Unbiased means that the estimator will over the long run average to the true error rate. This is a bit like a drunk person walking trying to walk a straight line. The person might average right down the center even when wobbling to the right and left.

In small samples variance tends to dominate, thus a low variance estimate that may even be somewhat biased has the potential of being superior to cross-validation on small samples. This is why a bootstrap estimator is interesting (see [h] for more details).

The two main bootstrap estimators are known as e0 and .632 bootstrap. For an e0 bootstrap estimator a training group consists of n cases sampled with replacement from a size n sample. On the average e0 trains on 63.2 % of the cases. The error rate on the test group is the e0 estimator. 200 iterations are considered necessary to get a good result. Thus it is computationally more expensive than CV.

The .632 is another version of this one. This is a technique for small samples. Usually it was considered that a sample was small when it had 30 or fewer cases. The variance effect is most pronounced in quite small samples, 30 or fewer, but the effect continues somewhat up to 100. Both e0 and .632 are low variance estimators. For moderately sized sample sets, e0 is clearly biased pessimistically, because on the average it uses only 63.2 % of the data for the training. It gives however very strong results when the

true error rate is high. As the samples grows in size .632 is overly optimistic but it is very strong on small samples when the true error rate is relatively low.

Under 30 points, bootstrap is better than cross validation. It can be better till 100.

For problem where the number of points is just around one hundred it is not straightforward to choose between a testing set and CV.

8.5.6.2 Testing Sets

When there is enough data to split the set in a training and a testing set another question arises: how are we going to split the data that is what size should our testing set have and what points should it contain ?

8.5.6.2.1 Splitting the data set in two

The Ratio between the size of training set and the size of the testing set (and eventually validation set) usually is two thirds for the training set and one third for the validation set. When there is a validation set one half for the training set and one quarter for each of the others is fine.

8.5.6.2.2 Splitting strategy

See [D2] about how to choose the points of the testing set. The strategy that has to be used to split the data actually depends on the kind of data and of the kind of problem we want to solve.

8.6 Selection of the model

8.6.1 Regularisation

To find the best model, the usual methodology is to train a range of networks and then with a testing set to try to find the one with the smallest error. This method has already been dealt with in [D2]. Here we present another methodology that can be an alternative or a complement. It is based on the control of the effective complexity of the model.

A regularisation term is added to the error function. The larger the weights grow the larger the error becomes, thus introducing a force to choose the simpler of a set of possible solutions and so improve generalisation performance.

The technique of regularisation consists of adding a penalty term Ω to the error function:

$$\tilde{E} = E + \nu\Omega$$

where ν is a parameter that controls the extent to which the penalty term Ω influences the form of the solution. ν should fall between 0 and 1. Several tests can be computed to find the best value.

Ω can have several different values (see [d]) but the most usual one is:

$$\Omega = \frac{1}{2}\sum_i \omega_i^2$$

8.6.2 Complexity evaluation

8.6.2.1 "prediction error"

This complexity criterion will allow us to have an estimation of a "prediction error" (that is something like a "generalisation error") once training has been carried out. This prediction error is minimal for an optimal generalisation performance.

$$PE = \textit{training error} + \textit{complexity term}$$

Where the complexity term is a penalty term which grows with the number of free parameters in the model.

We can see how this works heuristically:

- if the model is too simple, the PE will be large because the training error will be large
- if the model is too complex however then the PE will be large because the complexity term will be large.

So we observe the trade-off between a too simple model and an overfitting model.

For a sum-of-squares error and a linear NN the complexity term is:

$$PE = \frac{2E}{N} + \frac{2W}{N}\sigma^2$$

where:

E is the value of the sum-of-squares error,

N the number of data point in the training set,

W the number of weights,

σ^2 the variance of the noise (a noise model has to be assumed)

Fortunately this criteria can also be generalised to non-linear models and allow the presence of a regularisation term.

The Generalised Prediction Error is then given by:

$$GPE = \frac{2E}{N} + \frac{2\gamma}{N}\sigma^2$$

where E, N and σ^2 are the same parameters than above and where γ is the effective number of parameters in the network, which is given by:

$$\gamma = \sum_{i=1}^{W} \frac{\lambda_i}{\lambda_i + \nu}$$

λ_i are the eigenvalues of the Hessian matrix of the unregularised error evaluated at the error minimum, and ν is the regularisation coefficient.

This method is very interesting since it can be used to find the best NN model in the case where we did not have enough points to split them into training and testing set. After training several models and computing the generalisation error for each of them, finding the best one will be easy. But this "prediction error" can also be used for NN with testing sets. In such cases it can give a hint of the kind of NN that could be tested (see [d] pp 340-341).

8.6.2.2 VC dimension related boundaries

This criteria will be less useful in practice since it provides only some very high boundaries. One result can nevertheless be interesting for classification problems as we will see it.

In [c], Haussler has developed some boundaries for the number of weighting parameters needed to get an optimal generalisation for any case of NN application (regression as well as classification). These boundaries are however so high that it is almost pointless to use them in reality since the network that we are going to use will have less variable parameters than the VC dimension boundaries would have suggested us.

The only useful result for our validation process is the rule of thumb that for classification problems links the number of nodes with the number of patterns needed to carry out a good training.

$$N = W/E$$

where E is the expected error, W the number of weights and N the minimum number of input patterns needed for the training.

This result had already been described in [D2].

In a word these results (and particularly the "prediction error") are a very useful tool for complexity evaluation and thus for finding the NN with best performance. They are very useful for cases where there is not enough data to make a testing set.

8.6.3 Complexity reduction: "Clearning"

Weigend (see[y]) has recently developed a new method which allows us to model conditional densities for the outputs thanks to a very detailed study of the inputs (this study of the inputs is mainly based on the study of the input noise, see 8.3.6). This method also makes it possible to discover properties of the data otherwise not

accessible, and particularly the ratio between the input signal and the input noise. Eventually this method will make it possible to reduce the complexity of a network and to optimize the architecture of this network.

The basic idea of clearning is to simultaneously clean the data and learn the underlying process. That means that on one hand the model will modify the data and on the other hand the data will modify the model. Clearning is usually used in conjunction with standard pruning.

Here we will just present some of the main results of this method, but in no way give a detailed description of it; see [y] for that.

Basically what clearning is supposed to do is to drastically reduce the complexity of a network thus making it more efficient, thanks to a close study of the input data. What is more, interesting results about the structure of the noise in the data will be gained. In the formalism of clearning, the model we are seeking is allowed to modify data if the cost of changing the data is smaller than the benefit associated with the lower output errors.

Practically, learning (use of the data to modify the model) and cleaning (use of the structure to modify the data) come about with the use of a special cost function which is:

$$E = \frac{1}{2}\eta\left(y - y^d\right)^2 + \frac{1}{2}\kappa\left(x - x^d\right)^2$$

where the first part is the usual sum-of-squares error function between the network output y and the data output y^d. The second term is the squared deviation between the cleaned input x and the data input x^d.

The error derivative for the weights is the same as for the standard error function. The cleaned inputs however are updated according to

$$x_i = x^d + \Delta_i$$

$$\text{and } \Delta_{i+1} = (1-\kappa)\Delta_i - \eta\left(y - y^d\right)\frac{\partial y}{\partial x}$$

The update depends thus on:

- proportionality to the output error.
- proportionality to the sensitivity of the output with respect to the input (derivative term).

The two parameters of the error function are the learning rate η and the cleaning rate κ. A statistical interpretation of this function can be given in a maximum likelihood framework. We assume that each pattern was generated by a "true" input (estimated by x) and a true output (estimated by y). We then assume that all input and output components were independently corrupted by additive gaussian noise.

This means that the inputs can be characterised by their noise levels. This method allows to compute the noise level and the signal level variance separately. The deviation $\Delta(t)$ of each input and of each output as functions of the pattern index t (t is the time when each prediction is made, plotted along the horizontal axis) allows us to:

- detect outliers in the output
- characterise input variables by their stochasticity: the mean of the squared error across the time characterises the signal to noise ratio of each input feature.
- estimate the error covariance matrix.: the computation of this matrix can help investigating the validity of the assumption of statistical independence of the noise inputs

In the clearning context, it is the combination of clearning with a pruning algorithm which makes it possible to arrive at very small networks which are more efficient than the huge initial networks. But let us first explain what a pruning algorithm is.

Note: Pruning algorithms

Pruning is a technique aiming at optimising the network architecture as part of the training process itself. A systematic procedure is needed for exploring some space of possible architectures. The usual technique which would consist in training a set of networks and to choose the best one is not very efficient since only a small number of models can be trained (all of the various parameters of the NN cannot be changed; one has to focus on the number of hidden units for example). What is more this technique is computationally very expensive. A good alternative approach is to start with a relatively large network and gradually to remove either connections or complete units (for examples of pruning algorithms see [d] pp 353-363).

In practice, clearning is carried out until $\sum(y - y^d)^2$ increases on the validation set. What has to be removed are the weights that respond most to the noise. In every iteration, a new input is presented and the size of the weights at the end of this iteration are compared to its fluctuations in response to the inputs during that epoch.
A test value which corresponds to the ratio of the weights at the end of the epoch over the fluctuations during the epoch is computed and if this value is large, the weight is kept. If it is small the weight is pruned (for more details see [y]). The computation of this test value is carried out on the cleaned values.

8.6.4 Designing a classification MLP

The best way to find the most appropriate network to a given problem is to test a number of nets and then to select the one with the smallest testing error. What we intend to do in this paragraph is to give a number of rules that have to be respected to find a MLP with a correct degree of complexity. Experimentation will also be required here. Some of the previous results will also be used.
Complexity depends on several causes: the number of inputs and outputs nodes and the number of hidden nodes. Since the number of inputs and outputs depend mainly on the size of the data set we will focus on the hidden layers

8.6.4.1 Hidden layer size

When choosing the number of hidden units h, some elementary rules have to be respected:

- Never choose h to be more than twice the number of input units.
- p patterns of i elements can be loaded into $i \log_2 p$ hidden units. But never more should be used. For good generalisation, less have to be used.
- In a classification problem, it has to be ensured that there are at least 1/e times as many training examples as there are weights in the network.

Feature extraction requires fewer hidden units than inputs.
The number of hidden units required for a classification tasks increases with the number of classes in the task.
But choosing the right number of hidden units does not give the best solution. Some finesse can be introduced with new means: for example a regularisation term (see 8.6.1).

8.6.4.2 Network health measure

Looking at the test error performance is not the only way to gain insight into the health of a network. It can be done thanks to the distribution of the weights values. A NN should have small weights if it is to be able to generalise well. We can construct a very simple picture of the health of a network by drawing a histogram of the weights. It shows a count of the number of weights in the NN which fall in within each range of sizes. They can be plotted as simple bar charts. The range in which the values vary has to be divided y 10 or 20. The vertical axis shows the frequency of weights of each size. A good histogram shows a hump around the values with low magnitude. Indicating that most of the weights are low. A histogram with peaks at the extreme of the horizontal axis has probably overfitted the data.

Note: the variance has also to be taken into account since an untrained net with low initial random weights will produce a healthy histogram. That's why it is also a good idea to monitor the variance in the size of the weights as the network learns. Weights which do not move far from their initial points do not contribute much to the solution (if none of the weights moves, the network has not learned).

8.6.5 Use of random numbers

NN frequently use random numbers in the training process. This can add to the difficulties of assessment since the function that is implemented on the data is not deterministic.

Most training algorithms require a set of random values at the initialisation (weights, etc...) and these random parameters are then incrementally adjusted so as to improve the value of the function to be optimised.

The most obvious test of robustness is to run the algorithm several times with another initialisation and then to check if the state that the NN reaches is the same for every run. But two main issues are raised here:

- how many times does it need to be run?
- how to compare the outcomes of the training algorithm?

Another difficulty arises with the drawing of the different numbers.

Since this problem is central in validation and verification of NN we will provide a method for dealing with it. This method is based on committees of NN.

A general technique for dealing with the random number uncertainty is to repeat the basic training algorithm and then to take the best solution. But the approach with

committees may be better since it is an average of all the networks which is chosen rather than the result of the best network independently from the other ones (and which might depend on the randomly chosen starting point). Members of the committee are created using independently chosen random starting points for the network parameters (see [i] for details).

8.6.6 Committees of Networks

This paragraph about committees of NN has a double purpose: on one side to gain more insight in the advantages of a committee solution in order to allow LR to see how some NN applications that they will have to assess could be improved and on the other hand it will help them to assess NN applications involving committees.

8.6.6.1 Description

A common technique in the NN context is to train a set of different NN with the same training set and then to select the best one. But as we know, the net which gave the best results on the training set is not necessarily the one which will give the best results with the test set. This problem can be overcome by combining the networks in the collection in such a way as to obtain a prediction that is better than the predictions of any individual network.

As a means of dealing with the influence of random numbers used in training, one can see why committees should be effective: we are using a sample of the networks arising from these random numbers, rather than just an individual. The average of a sample should have lower variance.

Committees are also a means to improve generalisation performance of a NN application. Usually in a NN application we train several nets and choose the best one after using a test set. The committee is more powerful since it takes into account all the nets and it is not the same net which gives the best results every time.

8.6.6.2 Simple committees

In the case where the sum-of-squares errors of all the networks present in the committee are not correlated, the average sum-of-squares error of the committee would be divided by L (i.e. the number of networks present in the committee).

In reality however, the improvement is generally much smaller than this because the errors of the different models are highly correlated. From the point of view of validation we want our committee to be sufficiently large so that the observed average is highly likely to approximate the expectation

8.6.6.2.1 Generalised committees

In the simple committee approach discussed above, all we have done is averaging the predictions of the N networks of the committee. However we might know that some of the NN of the committee will make better predictions than others. We would therefore be able to reduce the error still further if we gave a greater weight to some member of the committee.

Thus we consider a generalised committee prediction output given by a weighted combination of the predictions of the members of the form:

$$y_{GEN} = \sum_{i=1}^{L} \alpha_i y_i(x)$$

and the α_i are determined by minimisation of a generalisation error:

$$E_{GEN} = \varepsilon\left[\left(y_{GEN}(x) - h(x)\right)^2\right]$$

8.6.6.3 Conclusion

Committees of NN are actually an example of Multiversion Systems. Such systems are often used in engineering because of their ability to increase the reliability of systems. In the NN context, their reliability will be better than that of any single NN application that is obtainable. Their power comes from their duality:

- combination of a set of appropriate NN
- selection strategy.

As discussed in [D2], if a NN is trained on a given problem this methodology can deliver an equivalent multiversion system with a superior performance and in some cases the increase can be dramatically high.

8.6.7 Mixture of experts

Here we consider the problem of learning a mapping in which the form of the mapping is different for different regions of the input space. Although a single homogeneous network could be applied to this problem the task would be easier if a different expert network was assigned to each different region. A gating network which also sees the input vector decides which of the experts should be used to determine the output.

The decomposition of the problem into different regions might well be known but if it is not, it can be considered as part of the training process; that is that each network should find it. The architecture of such a network is as follows:

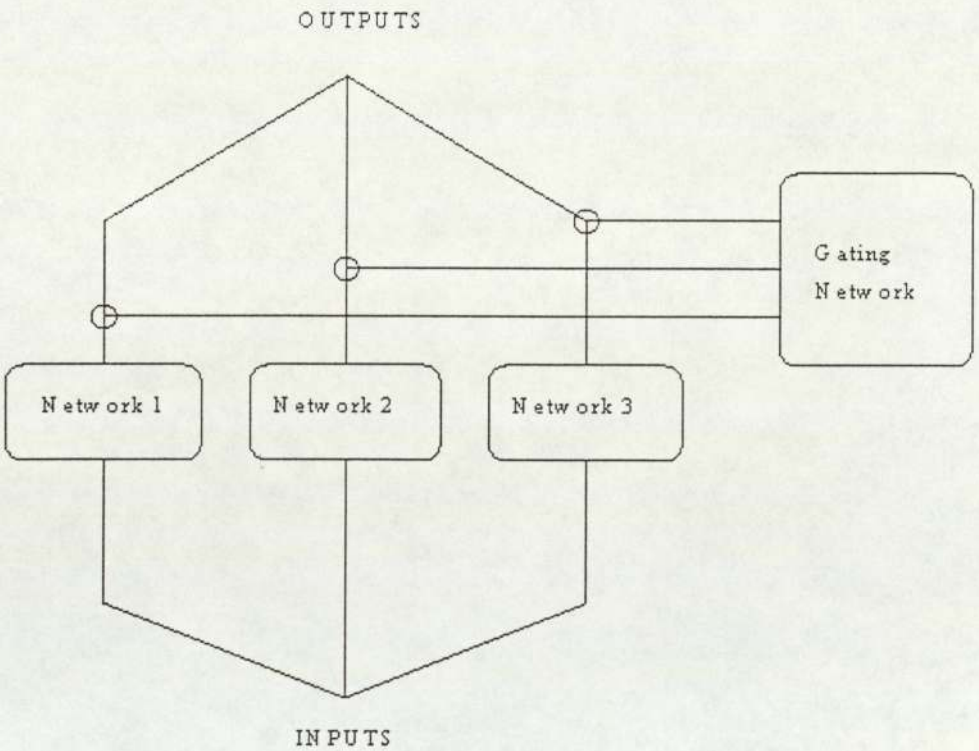

Figure 26: Example of a mixture of experts network.

The error function that is used is the negative logarithm of the likelihood function with respect to the probability density given by a mixture of gaussians.

$$E = -\sum_n \ln\left\{\sum_{j=1}^{M} \alpha_j\left(x^n\right)\phi_j\left(t^n \mid x^n\right)\right\}$$

where the gaussians are functions given by:

$$\phi_j(t|x) = \frac{1}{(2\pi)^{c/2}\sigma_j(x)} \exp\left(-\frac{\left\|t-\mu_j(x)\right\|^2}{2\sigma_j^2(x)}\right)$$

All the experts as well as the gating network are trained together. The mixture of expert networks is trained by minimising the error function simultaneously with respect to the weights in all of the expert networks and in the gating network. The key of this training is the error function. This function is very much the same than the one

of the conditional modelling density network (despite the fact that that network was not a mixture of expert network, but for further information see [d]). Mixture of experts can lead to improvements with highly heterogeneous training sets..

8.6.8 Statistical tests to find the best algorithm in a classification problem

So far we have not talked of the algorithms that are involved in our learning processes. What we want to do here is to find a method which could allow us to test if two algorithms applied to the same data set have the same performance. In what follows, two statistical methods that answer this question are presented. This methodology tackles several problems: randomness and generalisation performance.

8.6.8.1 Presentation of the problem

8.6.8.1.1 The central question

The statistical question that we have to answer is: "given two learning algorithms A and B and a large or a small data set S, which algorithm will produce more accurate classifiers when trained on the same data set S?"

The problem that we have to answer is very closely tied to the amount of data we have available. If there is a large amount of data, then some of it can be put aside and then be used as a test set for evaluating the classifiers. If there is only little data available, all this data will have to be used to train the NN. This means that some resampling will be needed to perform the statistical analysis.

8.6.8.1.2 Sources of variation

To evaluate and to design statistical tests, the first step is to identify the sources of variation that must be controlled by each test. There are four important sources of variation:

- Selection of the test data that is used to evaluate the algorithms. This problem is particularly pressing for small test data sets.
- Selection of the training data set
- Internal randomness (when random points are used within an algorithm).
- Random classification error (mislabelled data points).

A good statistical test should not be fooled by these variations. It should overcome them.

It has to fulfil two main conditions:

- it must take into account the size of the test set and the consequences of change in this set (because of the test data variation and the possibility of random classification error)
- it should execute the learning algorithm several times and measure the variations in accuracy of the different classifiers (because of the training data and internal randomness).

Experiments with real data have shown that the two best tests are Mc Nemar's test and the 5x2cv paired t test. (see [g]).

8.6.8.2 The two methods

8.6.8.2.1 Mc Nemar's test

To apply McNemar's test, we divide our available sample of data S into a training set R and a test set T. We train both algorithms A and B on the training set to obtain classifiers $\hat{f}_A$ and $\hat{f}_B$. We then test these two classifiers on the test set. For each data point of the test set we record how it was classified and construct the following contingency table:

Number of data points misclassified by both $\hat{f}_A$ and $\hat{f}_B$.	Number of data points misclassified by $\hat{f}_A$ but not by $\hat{f}_B$.
Number of data points misclassified by $\hat{f}_B$ .but not by $\hat{f}_A$.	Number of data points misclassified by neither $\hat{f}_A$ nor $\hat{f}_B$.

We will then use the notation

n_{00}	n_{01}
n_{10}	n_{11}

where the sum of this four numbers is the total number of examples in the test set T. If the two algorithms have the same error rate, we term this hypothesis H

The test is then based on a test for goodness-of-fit that compares the distribution of counts expected under hypothesis H to the observed counts.

The expected counts under the null hypothesis are:

n_{00}	$\frac{n_{01}+n_{10}}{2}$
$\frac{n_{01}+n_{10}}{2}$	n_{11}

The statistic is then:

$$\frac{\left(|n_{01}-n_{10}|-1\right)^2}{n_{01}+n_{10}}$$

and is distributed as χ^2 with 1 degree of freedom. It incorporates a continuity correction term (to account for the fact that the statistic is discrete whereas the χ^2 distribution is continuous).

This test involves running the algorithm exactly once and analysing the results. (this means the test does not measure training set variations at all).

8.6.8.2.2 The 5x2cv paired test

This test requires drawing multiple training sets from the data and running the algorithms several times. This permits the test to measure the training set variation directly.

Preliminary: the k-fold cross-validated paired test

Learning algorithm *A* and *B* are trained on training set *R* and the resulting classifier are tested on training set *T*.

We divide the sample S into *k* disjoint sets of equal size $T_1,\ldots,T_k$. We then conduct *k* trials. In each trial, the test set is T_i, and the training set is the union of all of the other $T_j, j \neq i$.

Let $p_A^{(i)}$ and $p_B^{(i)}$ be the observed proportion of test examples misclassified by respectively algorithms *A* and *B* during trial i.

In this test we perform 5 replications of 2 fold cross validation. In each replication, the available data is randomly partitioned into two equal sized sets S_1 and S_2. Each learning algorithm (*A* or *B*) is trained on each set and tested on the other set. This produces four error estimates: $p_A^{(1)}$ and $p_B^{(1)}$ (trained on S_1 and tested on S_2) and $p_A^{(2)}$ and $p_B^{(2)}$ (trained on S_2 and tested on S_1).

Subtracting these corresponding errors estimates gives us two estimated differences:

$p^{(1)} = p_A^{(1)} - p_B^{(1)}$ and $p^{(2)} = p_A^{(2)} - p_B^{(2)}$.

From these two differences we compute the estimated variance is

$$s^2 = \left(p^{(1)} - \overline{p}\right)^2 + \left(p^{(2)} - \overline{p}\right)^2$$

Let s_i^2 be the variance computed from the i-th replication, and let $p_1^{(1)}$ be the $p^{(1)}$ from the very first of the five replications.

It can then be proved that under hypothesis H,

$$t = \frac{p_1^{(1)}}{\sqrt{\frac{1}{5}\sum_{i=1}^{5} s_i^2}}$$

has five degrees of freedom.

8.6.8.3 Results

Experiments have shown that the 5x2cv test is the most powerful statistical tests for the error of type I (detect a difference when no difference exists). This test is also satisfying because it assesses the effect of both the choice of training set (by running the learning algorithms on several different training sets) and the choice of test set (by measuring the performance on several test sets).

According to the experiments it is recommended to use either the 5x2cv t-test, for situations in which the learning algorithms can be run ten times, or Mc Nemar's test, for situations where the learning algorithm can be run only once.

These tests can be useful to develop, understand and improve machine learning algorithms.

Since a classification problem is only one particular case of a regression problem, one can wonder if this method could not be extended to regression problems.

8.6.9 Conclusion

In this data analysis section we first managed to address all the various noise problems that can occur. Mixture densities networks are a very powerful technique that can be used for noise modelling as well as multivalued function mapping. The quality of the data set of the data set was studied in detail and techniques to address the related problems were provided. These problems ranged from missing data and time series data sets to how to handle large data sets. Finally techniques to obtain an optimal

model were presented, including the use of committees of networks which are techniques that are more and more used. Now the updated guidelines (that can be found in appendix B) cover most of the problems and deficiencies that have to be assessed within the current state of the actual NN research.

9. A safety case for NN software

In this section we seek to demonstrate that a safety case can be obtained for NN software. This is an indispensable condition for the use of NN software in a safety critical environment.

9.1 The safety case concept

9.1.1 The safety case as it is seen by LR

Here are some extracts of [s] which explain the purposes of a safety case and highlight the points on which we will have to concentrate.

"The safety of an operation is about more than hardware integrity. It is about more than procedures. It is about the totality of features including equipment and people that have a bearing on 'freedom from harm' or the safety of an operation. The safety case is a document describing those features."

"The purpose of the safety case [...] is to provide a clear, comprehensive, convincing and defensible argument, supported by calculation and procedure that an installation is and will be acceptably safe throughout its life."

"In broad terms the safety case should address the hazards associated with an installation and the management of those hazards."

A safety integrity level is a probability of failure. This a table of some current safety integrity levels as defined in [a].

Safety Integrity Level	Probability of Failure to perform function on demand
4	$10^{-4} - 10^{-5}$
3	$10^{-3} - 10^{-4}$
2	$10^{-2} - 10^{-3}$
1	$10^{-2} - 10^{-1}$

The following activities characterise the development of a safety case and this process is often referred to as formal safety assessment.

1. establish safety requirements (functions and integrity levels); these may be both risk based and deterministic.
2. consider both internal and external hazards, using formal and rigorous hazard identification techniques.
3. estimate the frequency or probability of occurrence of each hazard.
4. estimate the risk and compare with criteria.
5. demonstrate ALARP (As Low As Reasonably Practical).
6. identify remedial measures for design, modification or procedure to avoid the hazard altogether, to reduce the frequency of occurrence or to mitigate the consequences.
7. prepare the detailed description of the installation including information on protective systems and measures in place to control and manage risk.
8. prepare a description of the safety management system and ensure that the procedures within it are appropriate for the risks identified.

9.1.2 Scope of the document:

In the previous list of guidelines, guideline No. 1 is up to the expert of the system in which the NN will be since it is completely dependent on the problem.

No.2, 3 and 4 are typically what we are going to examine in this document:

- to identify the different hazards in our NN software.
- to estimate the probability of occurrence of these hazards.
- to show that they are low enough to get the required safety integrity level:

In this document we will focus only on the NN specific points. The aspects that are common to any software are out of the scope of this document.

9.2 Reliability and Computation of the overall failure rate:

In this safety critical context what we want to do is to compute the probability of failure (i.e. failure to perform its design function on demand) of the NN so that a safety integrity level can be associated with it.

9.2.1 Definitions:

A failure rate is roughly speaking the probability that the Equipment Under Control enters into a failure mode without the NN function having detected this failure mode. In the NN context this means that the error bars lie entirely into the normal mode subspace whereas the true value of the function lies in the failure mode subspace.

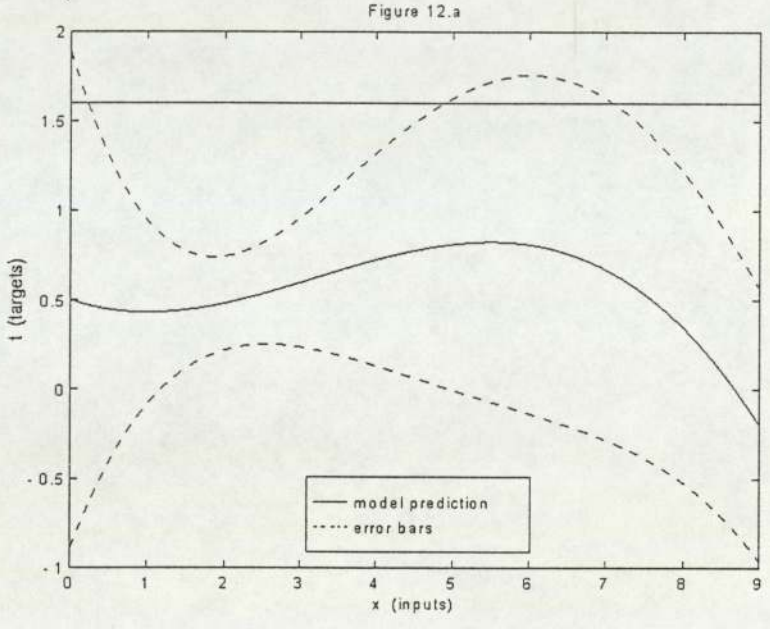

Figure 27: NN prediction with error bars and failure mode boundary.

Here we have the predictions of a NN and the error bars of this predictions. The line shows the failure mode boundary and we can see that the error bars lie within the failure mode space on one spot. This has to be quantified so that the probability of failure can be computed.

Figure 12.e: failure rate determination (undetected failure modes)

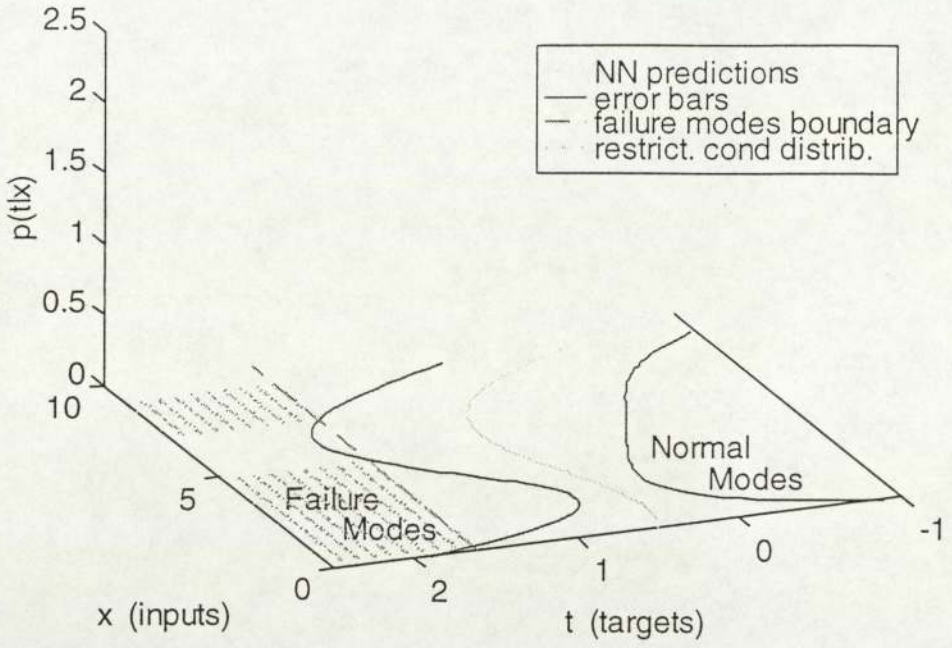

Figure 28: Three dimensional plot of NN with error bars and failure modes.

This is the same figure as the previous one but the third axis has been added to show how the problem looks like in higher dimensions.

9.2.2 Evaluation of the failure rate

What we would like to compute is the value of the probability of falling into the failure zone (i.e. probability of failure) . This probability is given by:

$$\int_V P(t \succ t_{FR} | x) p(x) dV$$

To compute this integral we are going to use a very powerful method based on the Monte-Carlo Theorem. Many classical numerical integration techniques (Trapezium, Romberg, etc.) which are very powerful for small number of variables are totally unsuitable for integrals as the one we are considering, that is which imply huge amounts of variables on complicated spaces. With standard methods, the computational cost would be too high. Let us first remind how the simple MC integration work. Then we will explain how to use them to compute our own example. Let us consider N random points uniformly distributed in a multidimensional volume *V*.

We call these points $x_1, x_2, \ldots, x_N$.

The basic theorem of Monte-Carlo integration estimates the integral of a function over the multidimensional volume V. It is given by:

$$\int_V f dV \approx V < f > \pm V \sqrt{\frac{< f^2 > - < f >^2}{N}}$$

where $< f >$ is the arithmetic mean over N sample points:

$$< f >= \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

What has to be mentioned is that the bound of the integration is not very rigorous and further there's no guarantee that the error is distributed as a gaussian. The indication of the error can thus be approximate (for more details about the Monte-Carlo theorem see [p]).

In the case that we are considering, what we want to do is to evaluate the following integral:

$$\int_V P(t \succ t_{FR} | x) p(x) dV$$

where $P(t \succ t_{FR} | x)$ is the output of the NN which is modelled by a posterior probability and $p(x)$ is a probability density function.

$P(t \succ t_{FR} | x)$ is a gaussian and thus and easy to sample from.

$p(x)$ is the density of the input space and it will be modelled probably by a mixture model.(see [e]).

9.3 Error bars:

The important conclusion of the last paragraph was that the probability of failure of a NN software could be computed. But now what we must question is if we take into account all the causes of uncertainty of our problem. In other words do our error bars take into account all the uncertainties of the problem?

So far in [D2] we only mentioned what error bars were in very general way. Here some precision will be provided since the failure mode analysis we are going to carry out depends on them. They play a key role in the validation of NN applications. This is why the assessors have to assess the quality of the error bars as well as the uses to which they were put. But first let us recall the different error sources that should be taken into account in this failure mode computation.

9.3.1 Different error sources:

In every NN application the aim is to get an output as accurate as possible for a new input point that is for an input that was not in the training set. That is why we seek for an optimal generalisation performance and an optimal training performance.

The quality of the output for this "new point" (i.e. point that was not in the training set) depends mainly on three different causes.

1. the "novelty" of the new points that is whether this new point lies within the domain of validity of the NN and has the same features than the points in the training set.
2. the density of the training data in the region of the new point.
3. the quality of the training data in the region of the new point (if there is missing data, over noisy data, conflicting data).

There are also other causes that are not in the scope of this document:

- choice of the training data: if it fits the problem we want to solve.
- the properties of the NN - network topology, training algorithm, degree of training- in a sense this is what is usually termed choice of the model. But this is not really a problem since in the "Data Analysis" paragraph (see pp14-70) techniques to obtain an optimal NN were exposed. The properties of the network are thus as good as they can be before that we start computing error bars.

The first and second points can be addressed with error bar computations. To take into account the density of the data in the region of the new point we need an error bound

on the outputs which is based either on the distance of the nearest training point to our new point or on a measure of the local training density. This point is obviously linked to the first one (if the new point lies in the validity domain of the network). We will present some of the methods that exist to assign error bars to the NN outputs.

For the last point (i.e. "quality of the training data in the region of the new point") we need to precise what kind of problem there is in the data set because the requirements are different for each of these problems.

- missing data: (see 8.5.5).
- conflicting data (identical inputs but different outputs): in classification problems we can obtain from human experts a probabilistic measure of confidence in each point.
- inaccurate data (mostly noise corrupted): we wish the outputs to include error bars which reflect not only the generalisation properties of the network, but also a measure of the inherent inaccuracies of the training data points.

To take into account the errors present in the inputs of a training set, an error bar computation method was described in 8.3.8 ("perturbation model").

For missing data and conflicting data (generated by a multivalued function) means to tackle these problems were described in 8.4 and so these problems can be solved before the error bars are computed.

9.3.2 Methods for error bar computation:

Now that we have summarised all the sources of error that can affect a NN we will present four methods for computing error bars taking into account error causes No. 1 and No. 2. and we will focus on their advantages and drawbacks.

9.3.2.1 Method 1: Stacked Generalisation (see [b] and [j])

We are going to assign some bounds on the outputs of a NN. Let us consider a standard feed-forward NN (net_1) that is trained on a training set L. Once the training has been carried out one input pattern L_1 is removed from the training set L and the remaining training points are used to train a second network (net_2). L_1 has then to be applied to net_2 and it will deviate from the expected value.

The input pattern L_1, together with the vector from L to its nearest neighbour in the training set now provide an input pattern , and the deviation (obtained from net_2)

provides an output pattern to a third network net_3. The full training set for net_3 is derived by removing each training pattern L_i from L and repeating the process.

When trained, net_3 can be used as a secondary network to net_1. A generalisation point presented to net_1 will produce an output response. The same generalisation point, plus its vector to the nearest training point when presented to net_3 will give as its output the error bounds of the primary network net1.

The main drawback of this methodology is that it requires the storage of the whole training set. This is required to calculate the vector from each generalisation point to its nearest neighbour in the training set. Obviously, the effort required to find the nearest neighbour increases with the dimensionality of the input vector.

9.3.2.2 Method2: Statistical moments (see [r])

We consider i^{th} the pattern of the training set which consists in a set of input variables $\{x\}_i$ and one output variable y_i.

A NN is trained on these patterns and computes the mean of Y in terms of the input variable $Y(\{x\}_i)$. A second training set can be formed from the same input variables y_i and output variables $\left(y_i - Y(\{x\}_i)\right)^2$. A secondary network can now be trained with this set.

This secondary network, when presented with a generalisation point will express the variance of the output of the primary network in terms of the input variables. If gaussian noise is assumed, an estimate of the 95% confidence interval is 2*sqrt(variance) on either side of the mean.

This method can also be used to find higher moments by replacing the squared term with a cube or quartic. The fourth moment can be used to differentiate between stochasticism due to noise and stochasticism due to multiple values.

The following steps will have to be followed when training a second network to predict the errors made by an existing model:

- Creating a input training set with the same input vectors as the original network.
- Creating a target output training sets consisting of one of the following measures, depending on the results one wants to get.
- Training the network on that data.
- If one wants the network to provide confidence levels for new input vectors presented to the original network , remembering to re-scale to the original range if

the values were scaled up to fall between zero and one for the training of the second network.

9.3.2.3 Method 3: (see [d] and [e]).

This method is actually used mainly in the bayesian framework. In Appendix B it is explained that a bayesian training provides some error bars without requiring further computations.

The distribution of the outputs is given by:

$$p(t|x,D) = \frac{1}{\left(2\pi\sigma_t^2\right)^{1/2}} \exp\left(-\frac{(t-y)^2}{2\sigma_t^2}\right)$$

$$\text{with } \sigma_t^2 = \frac{1}{\beta} + g^T H^{-1} g$$

This is the variance of the outputs (i.e. an error bar) that this method provides automatically.

9.3.2.4 Method 4: Input Noise: (see 8.3.5)

This method is actually based on the input noise and it was presented in details on pp 21-26. It allows to compute the covariance of the outputs due to noise in the inputs. This method is very useful to study how an error propagates through the network.

9.3.3 Reliability of error bars

All these methods of attributing accountability to real world networks in the form of novelty detection and error bar generation are necessary and useful, but all of them assume perfect quality training data (that is with no missing points, no overnoisy targets, no conflicting data). So far no error bar computation exists that takes all these sources of error into account. But since these "training data quality" problems can be tackled before the computation of the error bars they are no source of uncertainty anymore when the computations are carried out.

9.4 Conclusion:

In this section we have shown how some of the requirements of a safety case can be fulfilled in a NN application.

- some of the hazards were identified
- each hazard was taken into account in the study:
    - domain of validity
    - data density
    - quality of training data

The novelty of a new point (i.e. its domain of validity) and the data density in its region are taken into account in error bar computations.

In the cases where there is some input noise in the training set, this noise is taken into account as well in the error bar computations.

The other deficiencies in the training set (missing data, multivalued mapping) can be tackled with means described in [D8].

- The probability of failure that was obtained takes into account all the hazard sources of the problem.
- A failure rate can thus be assigned to a NN software.
- The quantitative problem of failure is better founded than the estimates associated with traditional software.

10. Conclusion and Ways ahead

10.1 Conclusion

The main purpose of the project was to improve the work that had been carried out last year thanks to a close study of the properties of data sets.

A precise study of the noise that corrupts most of the data sets had first to be done. We managed to show how the assumptions on the noise can be checked and how the problems due to noise can be tackled. What has to be emphasised is that different kinds of noises were examined and a good analysis of different noise models was obtained. A new aspect of noise problems was studied with the noise in the inputs and the "perturbation model". After problems due to the noise the main difficulty that can arise are multi-valued mappings functions. We showed how to tackle them. Several other deficiencies of the data set were also highlighted and solutions proposed. These include the size of the data set, its density or missing data. In this context it was also shown how "large" data sets should be handled. Particular attention was paid to time series data sets. The final major point that had to be examined in this study of the data was the link between the data set and the NN itself, or how to find the best NN for a given problem with a given data set. A set of techniques ranging from regularisation to the use of committees of NN were proposed to solve this problem.

It can be said that the improved guidelines obtained thanks to this project allow the assessor to point out the major and minor shortcomings that NN software can present. In most of the cases he will be able to propose effective solutions to tackle them and to decide whether or not the assessed software can be used in a safety critical context. In the last part of the project we showed indeed how a safety case can be obtained for NN software. Error bars played a key role in this context.

10.2 Ways ahead

The actual state of the project makes it possible to carry out a high level NN software assessment. Nevertheless a set of points where more work can still be done remain. To complete the technology transfer several case studies should still be carried out.

Here is a set of ideas and topics that could be dealt with in the future. This can be completed according to LR comments, remarks, suggestions and priorities.

Only the outline of what is yet to be done is given here because if a lot of details are provided these might be inaccurate or misleading and eventually be more of an obstacle than an asset for future work.

10.2.1 The perturbation model

So far we have explained the principle and the purpose of the perturbation model (i.e. to compute a covariance matrix of the outputs thanks to the noise in the inputs) but many practical details could still be given apart from how the matrix has to be computed.

There are several interesting topics that could be examined and which could result in some improved guidelines (note: we are in the context of an MLP).

10.2.1.1 Relationship between input noise and features of the mapping function

It has been claimed that in a region of the input space where the function being learned is fairly flat, input noise will have little effect whereas in regions where that function is steep, input noise can degrade generalisation severely. What should be done is to verify this claim and to study to a larger extent the perturbation model with respect to the aspect of the function to map in its region.

10.2.1.2 Error bar computation method

In [D9] we pointed out how important a role the error bars play in the evaluation of the probability of failure. Nevertheless so far no real guideline to choose the most appropriate error bar computation means has yet been provided. The most efficient error bar computation means should be listed and compared more deeply that what has been done yet in [D9]. Above all a real guideline which explains how to compute the most accurate error bars with respect to the features of the data and/or network error should be provided. In this context it should be shown in what situations the perturbation model is the best error bar computation means.

10.2.2 Committees of NN and mixture of experts

So far in this project committees of NN and mixtures of experts have only been presented in a very general way to show to what use they could be put. Since a lot of industrial and real world NN are actually committees, it should maybe be questioned if some guidelines peculiar to them could not improve the assessment of this nets. The same thing is true for mixtures of experts. Among the committee specific matters that could be addressed are:

- can a network which is in a committee be assessed a first time separately or can the committees be assessed only globally. Maybe it could be done in both ways?
- are there some rules about how to group networks in a committee? (number of networks, features of the networks, of the committee, the way in which these networks are put together, ...).
- how to build the best committee?
- how to choose best the gating network in a mixture of experts network.

10.2.3 Complements to the NN safety case

10.2.3.1 Error bar computations

As it has already been mentioned in a previous paragraph, what would have to be done is a complete list and comparison of error bar computation methods.

10.2.3.2 Industrial example

To illustrate the computation of a failure rate it would be interesting to try it on an industrial example to show how to carry out the computations described in D9. What would have to be done is first to compute error bars and then to compute the failure rate integral with a Monte-Carlo method.

10.2.3.3 Eventual improvement of the failure model

Improvement of the model of the failure. Is it not an oversimplification to model the failure rate by a single value? Could another model be found and the failure rate be computed? If yes the computation of the failure rate might have to be slightly changed.

10.2.4 Complement on time series data processing

So far we provided some methods to detect non stationarity and how to tackle it. Methods like detrending or differencing can give very efficient results but in some cases they can also result in a loss of information. Other techniques may exist to tackle such cases. The time series data pre-processing is thus a topic which needs some further investigation.

10.2.5 Unsupervised learning methods

So far we have not really concentrated on unsupervised learning techniques. The definition was given in [D2] but the particular assessment problems that arise with them have not been addressed yet. This is something that should be done.

10.2.6 Control

NN as controllers is a topic that was dealt with in [D10] (deliverable produced by Dr Nabney). Further work in that domain is still to be done.

10.2.7 More cases studies

Since the case studies are the opportunity where the real technology transfer happens, it would be suitable to carry out several of them. Quite a few contacts were made with researchers whose projects would be at a suitable stage for assessment next year. Could there be some industrial projects as well?

11. References

[D2] Neural V Technical Context, version 2.1

[D3] Neural V Guidelines, version 2.1

[D5] Neural V Conclusions and ways Ahead, version 1.0

[D8] Neural V Data Related Problems, version 1.1

[D9] Neural V Safety Integrity Levels, version 1.0

[D10] Neural V NN and control, version 1.0

[a] "Draft IEC 1508: Functional safety: safety related systems", (June 1995).

[b] E B Bartlett, K Kim, "Error bounds on the output of artificial neural networks", Trans American Nuc. Soc. No. 69, pp 197-99 (1993).

[c] E B Baum, D Haussler, "What Size Gives Valid Generalisation ?", Neural Computing 1 pp 152-160, (1989).

[d] C M Bishop, "Neural Networks for Pattern Recognition", Oxford University Press, (1995).

[e] C M Bishop, "Exact Calculation of the Hessian Matrix for the Multi Layer Perceptron", IEE Proc. Vis. Image and Signal Processing 141.4, pp 217-222 (1994).

[f] G E P Box and D R Cox "An analysis of transformations", Roy. Stat Soc. B:26:211-252, (1964).

[g] T G Dietterich, "Statistical Test for Comparing Supervised Classification Learning Algorithms", Department of computer science Oregon State University, (1996).

[h] B Efron, R J Tibshirani, "An Introduction to the Bootstrap", Chapman and Hall (1993).

[i] P Goldberg, "Validation and Verification of Neural Network Systems", Aston Unversity, (1997).

[j] I S Helliwell, M A Turega, R A Cottis, "Accountability of NN trained on real world data", Artificial NN IIE Conference Publication No. 409 pp 218-223 (1995).

[k] M Kendall, J K Ord, "Time Series", Edward Arnold (1990).

[l] D J C Mac Kay, "Bayesian Interpolation", Neural Computation Vol. 4 No. 3 May (1992).

[m] R H. Mayers, "Modern Regression with Applications", PWS-Kent Publishing Company, (1990).

[n] G Morgan, J Austin, "Safety Critical Neural Networks", Artificial NN IIE Conference Publication No. 409 pp 212-218 (1995).

[o] I T Nabney, "Industrial NN course", Aston University (1997).

[p] W H Press, S A Teukolsky, W T Vetterling, B P Flannery, "numerical Recipes in C: the Art of Scientific Computing", Cambridge University Press, (1992).

[q] B D Ripley, "Pattern Recognition and NN", Cambridge University Press (1996).

[r] C J Satchwell, "Finding error bars (the easy way)", Networks Official newsletter of the Neural Computing Application Forum, 5 ,2 (1994).

[s] J T Stansfeld, "The safety case", Paper No. 3 Session 1994-95 of LR's Technical Association.

[t] K Swingler, "Applying NN: A Practical Guide", (1996).

[u] N W Townsend, L Tarassenko, "Estimation of error bounds for RBF networks", Oxford University, (1997).

[v] N W Townsend, "Neural Networks for Mobile Robot Localisation Using Infra-Red Range Sensing", PhD thesis, Oxford University (1995).

[w] V Tresp, S Ahmad, R Neuneier, "Training Neural Networks with Deficient Data", Morgan kaufman (1994).

[x] K Warwick, G W Irwin, K J Hunt, "NN for Control and Systems", (1992).

[y] A S Weigend, P Refenes, Y Abu Mostafa, J Moody, "Neural Networks in Financial Engineering", (proceedings of NNCM 95, London), Singapore World Scientific (1996).

12. APPENDIX A

Introduction

This document deals with two topics (prior probabilities in classification problems and bayesian learning) that were alluded to in [D2] but about which no further details were given. A deeper insight into these topics provides interesting information which makes some parts of [D8] and [D9] easier to understand.

12.1 NN to model the posterior probabilities of class membership

In this classification context the NN models the posterior probabilities of class membership. When the number of classes is at least three, for each possible class there is one output and the activation of the output node corresponds to the posterior probability. These probabilities are then used to make the decision. By considering a NN as a posterior probability generator we will get a number of interesting and powerful results.

12.1.1 Compensation for different prior probabities

If we consider the two class classification problem example of a medical system having to distinguish between normal tissues (C_1) and tumours (C_2) on medical X-Ray images.

From medical statistics we might know that the probability of observing a tumour in such a tissue is of 1% and so the prior probabilities will be $P(C_1) = 0.01$ and $P(C_2) = 0.99$.

If we pay no particular attention to the choice of our training set we will have one tumour for 99 healthy images. If the training is carried out with such a set it will be impossible for the net to learn to distinguish between them unless we have a huge number of screenings. That means that we prefer to take roughly equal numbers of examples from each class to ensure we get a reasonable number of tumours in our training set. If we use such a network directly, there will be as many chances for a given input vector to be classified as healthy as tumour. This is of course a very poor result. But to get the good result, all we have to do is to divide each of the output node

by the corresponding prior probability and the system will give the right results. What is more is that if the prior probabilities change, there will be no need to train the network again. Only a new division by the new prior will have to be done.

In a word we have increased the numbers of tumours in the training set and compensated for the different priors by dividing the outputs by the priors.

Note: the value of the priors has of course to be as accurate as possible and particular attention has to be paid to its collection (see [D3]).

12.1.2 Importance of the risk minimization

There still remains one key issue in our problem: the cost of misclassification. In the medical example above there is no doubt that there are much more serious consequences if we classify an image of a tumour as normal than if we classify a normal image as that of a tumour. What we should do is to affect a different cost to each of these misclassifications. To do that we can use a loss matrix.

In the medical screening, these loss matrix values would probably be chosen by hand, based on the views of experienced medical staff.

For other applications, in finance for example, the coefficients can be chosen in a more systematic way since the risk can be more easily quantified.

12.1.3 Rejection Thresholds

Sometimes when for a given input vector x the largest of the posterior probabilities is relatively low, it might be better not to make an automatic decision in the network. This is called the reject option. In the medical screening for example, the doubtful cases could be classified by an human expert. To detect these doubtful cases, all we have to do is to use a rejection threshold which is actually the value of the posterior probability under which no classification is made.

If $\max_k P(C_k|x)$ is above some threshold θ then x is classified; otherwise x is not classified and an human operator has to do it. the threshold has to be chosen by a human expert. Again no network retraining is needed here.

12.2 Bayesian techniques

So far, bayesian techniques are less widespread than the usual techniques involving MLPs and RBFs. But since they are very likely to be more and more used, it is worthwhile to know their main features and how they work in order to grasp their

peculiarities (this has already be done in the appendix of [D2], but here we try to be a bit more practical). The assessment techniques in the bayesian framework might differ in some respects from MLP and RBF assessment techniques and that is why we need to examine them.

The classical techniques using MLP or RBF are usually based on the minimisation of a likelihood error function and attempt to find a single set of values for the network weights. The bayesian approach however considers a probability distribution function over weight space which represents the relative degree of belief in different values for the weight vector. The outputs are probabilities and we will see how this method brings about an error bar computation method.

We are not going to give here too many details of these techniques (see [d] and [l] for that) but only an outline of how a bayesian learning comes about.

12.2.1 Bayesian weight learning:

In this framework a probability distribution over the weight is considered.

In the absence of any data the prior distribution is described by $p(w)$.

Its expression is:

$$p(w) = \frac{1}{Z_w(\alpha)} \exp(-\alpha E_w)$$

where $Z_w(\alpha)$ is a normalisation factor and E_w an error function.

$w = (w_1, \ldots, w_n)$ are the adaptative weights of the NN and $D = (t_1, \ldots, t_n)$ is the target vector of the data. Once the observed data has been taken into account we can use Bayes' theorem to compute the posterior probabilities of the NN.

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

where :

- $p(D)$ is a normalisation factor.
- $p(D|w)$ represents a noise model for targets of the data and corresponds to the likelihood. Its expression is:

$$p(D|w) = \frac{1}{Z_D(\beta)} \exp(-\beta E_D)$$

But since the data set consists of inputs as well as outputs, these inputs have to be taken into account in the computation of the posterior probability. Bayes' theorem can thus be re-written like this:

$$p(w|D,\chi) = \frac{p(D|w,\chi)p(w|\chi)}{p(D|\chi)}$$

12.2.2 Computation of network outputs:

When a new input vector is presented to the NN then the posterior probability can be computed very easily. This posterior probability is given by:

$$p(t|x,D) = \int p(t|x,w)p(w|D)dw$$

Some computations bring us easily to the final expression which is:

$$p(t|x,D) = \frac{1}{(2\pi\sigma_t^2)^{1/2}} \exp\left(-\frac{(t-y_{MP})^2}{2\sigma_t^2}\right)$$

where σ_t^2 is the standard deviation of the predictive distribution.

It can be interpreted as an error bar on the outputs. The bayesian technique provides error bars without further computations being necessary.

12.2.3 Example of learning algorithm

In this particular case here we assume that the outputs are gaussian. The main steps of the training algorithm are:

- to choose the initial values for the hyperparameters α and β and initialise the weights in the network using values drawn from the prior distribution.
- To train the network using a non linear algorithm to minimise the total error function *S(w)*.

$$S(w) = \frac{\beta}{2}\sum_{n=1}^{N}\left\{y(x^n;w) - t^n\right\}^2 + \frac{\alpha}{2}\sum_{i=1}^{W} w_i^2$$

$$= \beta E_D + \alpha E_w$$

- Every few cycle of training, re-estimate values for α and β using:

$$\alpha^{new} = \frac{\gamma}{2E_W} \text{ and } \beta^{new} = \frac{(N-\gamma)}{2E_D}$$

with $\gamma = \sum \frac{\lambda_i}{\lambda_i + \alpha}$.

The λ_i being the eigenvalues of the hessian matrix. The computation of the hessian matrix and of it eigen value spectrum is thus required

- Repeat steps 1 to 3 for different random initial choices for the network weights in order to find different local minima. (In principle a check should be made in order to ensure that the different solutions are not only related by a symmetry transformation of the network.)

Repeat steps 1-4 for a selection of different network models and compare their evidence using

$$\ln p(D|H_i) = -\alpha E_W - \beta E_D - \frac{1}{2}\ln|H| + \frac{W}{2}\ln\alpha + \frac{N}{2}\ln\beta + \ln M\,!+ 2\ln M + \frac{1}{2}\ln\left(\frac{2}{\gamma}\right) + \frac{1}{2}\ln\left(\frac{2}{N-\gamma}\right)$$

where M is number of hidden units.

The final distribution that is obtained for the targets (different from the weights distribution) is

$$p(t|x,D) = \frac{1}{\left(2\pi\sigma_t^2\right)^{1/2}}\exp\left(-\frac{(t-y)^2}{2\sigma_t^2}\right)$$

$$\text{with } \sigma_t^2 = \frac{1}{\beta} + g^T H^{-1} g$$

This is the variance of the outputs (i.e. error bars) that this method provides automatically.

12.2.3.1 Computational aspects:

In the conventional maximum likelihood approaches involving MLPs and RBFs, the computational efforts are concerned with optimisation in order to find the minimum of an error function.

By contrast in the bayesian approach most of the computational efforts are concerned with integration over multi-dimensional spaces. As long as the outputs (i.e. posterior probabilities) are assumed to be gaussians (or approximated by gaussians). These integrals can be evaluated analytically. But if we wish to avoid the gaussian approximation we need to use efficient numerical integration computation methods. But the usual integration methods which can be used efficiently in many cases involving only a small number of variable are totally unsuitable here since the integrals that are considered often involve integration over spaces of hundreds and thousands of variables.

That is why some random sampling methods have to be used. Such methods are called Monte-Carlo methods. A basic explanation of how these methods work has already been given on pp 78-79.

13. APPENDIX B

This is an extract of the guidelines (the section which were modified). The modified guidelines are those with a +.

4. DATA ISSUES

4.1. Principle

The quality of the data is in general closely related to its origin and the way it has been collected. Once the data has been collected, it is to be analysed in order to evaluate its quality, since this is a central issue with data-driven models. The outcomes of this analysis are to be a key factor in the design phase.
There are 2 broad categories of types of data: the data sampled from the system to be modelled, and the prior knowledge concerning this system *(e.g. for classification problems, the costs associated with misclassifications represent some prior knowledge)*. Typically dedicating 30-40% of the overall development time to the data issues can be considered as normal.

The data collection process should be understood and clearly documented. The data collected should be analysed to determine its qualities and characteristics. Prior knowledge should be tested whenever possible. The results of analysis should be clearly documented.

4.2. Criteria

DATA COLLECTION AND ANALYSIS

4.2.1. The origin of the data sampled is to be clearly identified and stated in the documentation.

4.2.2.+ The data is to be shown homogeneous over space (and time). The homogeneity over space can be checked with visualisation techniques. If some areas of low data density are spotted there are several ways of coping with it.

a) if there is a lot of data available it should be used so that the data set is homogeneous.

b) if there is no other data available than the heterogeneous data, either some points in the regions were the density is the highest can be discarded or the training can be carried out knowing that the data is not homogeneous and that this fact will be taken into account in the error bar computation.

c) using a mixture of experts in the region where there is a problem with density.

4.2.3. The prior knowledge is to be shown to be consistent with the data samples collected.

4.2.4. The degree of control over the data collection is to be stated. For example, data collection from physical plants is often constrained by the need to run

the plant in a stable fashion, and thus data corresponding to a failure mode may be difficult to collect.

4.2.5. If prior knowledge is used to collect the data sampled, it is to be clearly stated in the documentation and easily traceable.

4.2.6 If special groups outside the agency are involved in the collection of the data, the interface between these various groups must be clearly identified and the transmittal of information from one group to another must be done in a controlled way.

DATA ANALYSIS

4.2.7. Appropriate means are to be used to ensure that a data set representative of the problem can be built from the data collected or at least that representative tests of the NN function can be carried out. *For example, techniques such as failure mode analysis, data density modelling, or trace through subsequent analysis can be used.*

4.2.8.+ The model used to estimate the data density has to be justified.

a) If tests have been carried out in order to choose one model instead of another they have to be reported. Mixture models should not be forgotten since they often give better results than a model obtained with a parametric model or a non parametric model.

b) The model used to estimate the data density is to be stated in the documentation. This includes the type of model and the value of the corresponding parameters. *For example, Gaussian mixture model, number of Gaussians in the mixture, prior, mean and variance for each Gaussian.*

4.2.9. The means used to ensure that the data is reliable are to be appropriate. *For example, means to deal with problems related to outliers, corrupted data and missing data.*

4.2.10. The distribution of the values of every variable (input and output variables) are to be studied and documented. *For example, this can be used to detect possible skewed distributions.*

4.2.11.+ It has to be ensured that the noise distribution that has been chosen is accurate.

a) If it was assumed gaussian then at least one of the following methods should be used to check the validity of this assumption by testing the model residuals.

- Normal quantile plotting
- Statistical tests to check normality: Kolmogorov-Smirnov test.

The zero mean has to be checked as well

- t test

b) If it turns out that the chosen model was inaccurate another one has to be chosen and the tests have to be carried out again.

c) If there is some data available about an input noise or if a good model can be found to model it, error bars taking this noise into account should be computed using the perturbation model formulae. (see [D8])

d) If the noise was non gaussian similar means have to be used to check the validity of the assumed distribution.

4.2.12 If prior knowledge is used during the data analysis stage, it is to be clearly stated in the documentation and easily traceable.

4.2.13. For time series problems, some considerations of likely model order are to be provided. *For example, by using correlation analysis.*

4.2.14.+ Detection of multivalued mappings.

a) If the data set comes from a problem where a multivalued mapping would be possible, conditional density modelling should be carried out to check whether or not this was the case. If yes this solution has to be kept. If no, another solution can be sought.

b) If we know that the problem is an inverse problem (see [D8]) it has to be modelled with the class conditional density as explained so that all the branches can be used, or if only one branch is required then a single inverse model can be trained.

4.2.15.+ Missing data problems have to be addressed as well as possible. Points with missing data should not be discarded. Other techniques should be used (see [D8]). Missing data problems are to be solved before the error bar computations.

5. DESIGN, DEVELOPMENT

5.1. Principle

Generally, to develop a solution using NN technology, several models (by model we mean a Neural Network architecture, that is Multi Layer Perceptron or Radial Basis Function NN, with given numbers of layers and nodes) are independently trained using a training set. The models and the associated training set are chosen during the design phase. These models are then trained by using the training sets during the development phase.

The design should propose a selection of models based on the specification and data analysis. Development should use sound methods with particular attention to generalisation and all results should be reproducible.

5.2. Criteria

DESIGN

5.2.1. Design quality plans should describe each of the activities or tasks included in the design process. The responsibility for each of these should be identified.

5.2.2. Any part of the problem that can be tackled easily with conventional programming techniques is to be tackled with conventional programming techniques. *For example, data normalisation.*

5.2.3. The modularity of the NN function is to be appropriate to the problem. *For example, for classification problems, the discriminant function is to be distinct from the NN unit.*

5.2.4. The functions of the pre and post-processing units are to be described unambiguously in the documentation.

5.2.5.+ A range of models is to be considered: from simple models (including linear models), to more complex ones. The right complexity has to be found with at least one of the following techniques:

- with regularisation
- prediction error criterion

5.2.6. An unambiguous description of each model architecture considered is to be given.

5.2.7.+ The data set size is to influence the choice of the models to be tried.

a) for 2-class classification problems, the number of training data points should be at least equal to the ratio between the number of weights in the NN and the maximum expected misclassification rate,

b) For some classification problems, though, there is not always the same proportion of points in each class so the training has to be carried out

- with discarded or added points (if some are available).
- with prior adjustments and a cost matrix. In such a case, these priors have to be chosen with the aid of an expert in the domain of our problem. The same care has to be paid to the choice of an eventual rejection threshold.

5.2.8. Prior knowledge is to be reflected in the model design and its incorporation is to be traceable.

5.2.9. The design is to be traceable to specification and data analysis.

5.2.10.+ Data selection problems are to have been suitably considered:

a) when there are not enough data points to have a test set (i.e. less than 100) cross validation has to be used. If the number of points is smaller than 30, then a bootstrapping method has to be used.

b) when there are enough points to have a training set usually at least one third of the points are in the test set and in order to ensure independence between the different data sets, a suitable randomising method is to be used to split the initial data set, and is to be documented.

c) in order to ensure independence between the different data sets, a suitable randomising method is to be used to split the initial data set, and is to be documented.

d) dimensionality reduction for "big" (i.e. computationally impossible to deal with) data sets: if points have been discarded or if some points have been discarded and not others it has to be justified. Means to justify such behaviours are: information theory, covariance calculations, PCA, FA, Neural nets with less hidden nodes than input nodes.

e) MLPs trained with p patterns of i elements should not have more than $i \log_2 p$ hidden units. For good generalisation, less have to be used.

5.2.11. The different data sets used to develop the solution are to be included in the documentation.

5.2.12. Except if it is shown that a sufficiently large quantity of high quality data is available *(e.g. the PAC learning framework provides a definition of 'large quantity of data')*, suitable means are to be used to reduce the dimensionality of the problem *(e.g. feature extraction, modularity considerations)*. The means used are to be stated in the documentation.

5.2.13.+ Error bars have to be computed, with an appropriate method, suitably justified. The choice of the error bar computation means has to be reported and justified. Failure modes can be mentioned and they can be computed as explained in [D9].

5.2.14. Some means of novelty detection are to be designed. *For example, self-checking of the inputs or the outputs by using data density modelling.*

5.2.15. The noise model used is to be appropriate to the problem.

5.2.16. The design is to be documented and is to be free from ambiguities, contradictions and other internal inconsistencies. *For example, the structure of each NN unit is to be unambiguously described.*

5.2.17. The design is to describe the functions, performance constraints and the interfaces and dependencies between modules, other components or sub-assemblies.

5.2.18. Except if it is shown that the problem to be solved is 'not difficult' (e.g. low dimensionality and large quantity of high quality data available, or satisfying PAC learning criteria) the prior knowledge is to be shown as significantly simplifying the NN function development.

DEVELOPMENT

5.2.19. If MLPs are used, the training algorithms used are to be specified in the documentation.

5.2.20. If RBFs are used, suitable methods are to be used to determine the basis function parameters and the final weight layer.

5.2.21. The suitability for use and the theoretical correctness of every major technique involved in the development of the NN function or incorporated in the NN function are be demonstrated and included in the documentation. Although for well established techniques a formal proof of the correctness is not required, a list of technical references is to be provided. *For example, the convergence property of the training algorithm, techniques used in the pre-processing unit.*

5.2.22. Suitable techniques are to be used to reduce the problem of *bad* local minima. The techniques used are to be reported in the documentation. *For example, multiple random weight initialisations, genetic algorithms.*

5.2.23. Suitable techniques are to be used to reduce the problem of overfitting. The techniques used are to be reported in the documentation. *For example, regularization, early stopping.*

5.2.24.+ Noise to improve generalisation performance.

a) A different noise value must be added to each value each time it is used by the network. It is not sufficient to simply build a new training set to which noise has been added. Noise must be added dynamically during learning.

b) Adding noise to the training data must preserve the mean value of that data. That is to say that the noise must have zero mean. This may be achieved by ensuring that the random number generating function produces a value between $-n/2$ and $n/2$ where n is the magnitude of the range over which the noise must fall

5.2.25. The choice of the error function is to be correctly justified. See [D2] for the choice of an appropriate solution.

5.2.26.+ Every random value necessary to reproduce the results is to be reported in the documentation. *For example, the random initialisations of the weights.* Several algorithms can be compared, all of which contain different random numbers. Committees or mixture of experts can also be a solution that can lead to some improvements with the choices of random initialisation.

5.2.27. The stopping criteria used for the training processes are to be specified in the documentation. *For example, the maximum number of iterations.*

5.2.28.+ The choice of the algorithm has to be justified (it can be chosen with one of the statistical tests presented in [D8] statistical method). The random number generator has to be tested for Markov Chains Monte Carlo types approaches.

6. MODEL SELECTION, INTEGRATION

6.1. Principle

Once the different models have been trained, the best ones are selected by using a validation set. The selected models are then integrated in the overall NN function.

Model selection should be based on specification and test results. The integration of NN and other systems should be tested and the results of the tests documented.

6.2. Criteria

MODEL SELECTION

6.2.1. If a single model is chosen, then it is to be the best model with regard to the specification requirements. Furthermore, among equally good models, the one with the simplest assumptions is to be selected.

6.2.2. If a committee is constituted then the models of this committee are to be selected according to both their individual performance with regard to the requirements specification and their diversity.

6.2.3. The performance of each model considered (even if not selected) is to be reported.

INTEGRATION

6.2.4. All the functions used in the implementation of the NN and in the pre and post processing task must be traceable to ensure that it is always the last version of these functions that is tested. *(e.g. fixed C-code).*

6.2.5. Each unit of the NN function is to be tested separately.

6.2.6. Any design constraint on the NN function that is individually testable, is to be tested. *For example: novelty detection system, validity of the error bars system.*

6.2.7.* Test documents are to define the hardware and software configuration to be used for testing including test tools.

6.2.8.* Test results are to be recorded as defined in the relevant test specification and in a form that permits verification.

6.2.9. All interfaces between the NN function and other software components are to be tested.

7. OVERALL NN FUNCTION TEST

7.1. Principle

During the testing stage, the requirements specification is used as a reference and the NN function is to be shown to comply with this specification.
The aim of the section is to test the solution that we have chosen in section 6 in order to check if it fits the specifications that had been made on the outputs of the NN.

7.2. Criteria

TEST PLANNING AND SPECIFICATION

7.2.1. The test cases are to cover every testable requirement of the specification. Completion criteria (goals) for tests are to be specified and used for evaluating the adequacy of testing. For instance: .

The testing and validation sets are to be representative of the problem, or a suitable technique is to be used to ensure that representative tests can be performed (e.g. by modifying the error function to take into account the discrepancies between the true data density and the density of the validation or testing set).

7.2.2.* The features specified in the user documentation are to be tested.

7.2.3.* Test cases are to be documented, including inputs, expected outputs and pass/fail criteria.

7.2.4. Hypotheses made are to be shown valid or suitably justified. For example, sensitivity analysis can be used to evaluate the importance of the different hypotheses.

7.2.5. If little data is available, specific techniques are to be used to estimate the true performance of NN function. *For example, resampling techniques.*

7.2.6. Tests are to be traceable to the requirements and design.

7.2.7.* Test documents are to define the hardware and software configuration to be used for testing including test tools.

TEST RECORD AND RESULTS

7.2.8.* The NN software tested is to be identical to the NN software under assessment.

7.2.9.* Testing is to be performed in the target operational environment. Where this is not possible, testing is to be done in a test environment that has been shown to be equivalent, supported by a demonstration of successful operation in the target operational environment.

7.2.10.* The overall testing pass/fail criteria are to be shown to be satisfied.