Statistical Methods for Neural Network Prediction Models

Guy-Michel Cloarec

Statistical Methods for Neural Network Prediction Models.

Guy-Michel Cloarec

Research Report EE/JVR/97/2

Control Systems Group School of Electronic Engineering
Dublin City University
*

June 1997

Abstract

Intelligent modeling techniques have evolved from the application field, where prior knowledge and common sense on the system behavior was implemented using linguistics or life mimetic systems (mainly Fuzzy and Neural network networks) where topology and parameters were determined by hook or by crook (also called educated guess where the intelligence in the person performing the modeling rather than in the techniques), to more rigorous modeling approaches where theoretical issues are addressed using statistical inference and where the uncertainties about the modeling are handled.

Recent works tends to bring together the two approaches that seemed to occupy opposite extremes of the system modeling approaches. As a matter of fact, the so-called intelligent methods and the classical mathematical systems methods (or regression methods) are connected by a common denominator : The statistical theory field. Modeling methodologies are using more and more features of what is defined as multivariable statistical analysis field. In this study, we will examine how the neural network modeling field can benefit from the techniques of the statistical theory field. With an emphasis on time series modeling, we will review the various techniques used in the neural network design and propose a general modeling methodology. The key elements will be the analysis and preprocessing of the data set, the use of statistical methods to reconsider the neural networks as probabilistic models and for obtaining good performance estimates, how to use these for in statistical decision making methodology to determine the optimal topology, how to use statistical methods to improve parameter optimization and build a neural network structure that makes tractable multistep ahead prediction.

*E-mail: cloarecg@eeng.dcu.ie

Keywords : Neural Networks, Statistical Theory, Topology Optimization, Generalization, Bootstrap, Course of Dimensionality, Learning Algorithm Convergence, Trajectory Learning, Principal Component Analysis, Feature Extraction, Network Committees.

Contents

1 Model Selection and Generalization Estimation. 5
- 1.1 Introduction . . . 5
- 1.2 Model Examples . . . 5
  - 1.2.1 Standard Linear Model . . . 5
  - 1.2.2 Non-Linear Model . . . 6
  - 1.2.3 Model Selection Uncertainty . . . 6
- 1.3 Generalization Estimation . . . 8
  - 1.3.1 Introduction . . . 8
  - 1.3.2 Single-Sample Statistics . . . 8
  - 1.3.3 Split-Sample Validation . . . 9
  - 1.3.4 Cross Validation . . . 9
  - 1.3.5 Statistical Bootstrap . . . 10
- 1.4 Principal Component Analysis. . . . 12
  - 1.4.1 Introduction. . . . 12
  - 1.4.2 Geometric and Algebraic Basis of Principal Components. . . . 12
  - 1.4.3 Subspace Reduction . . . 14

2 Neural Network overcome 16
- 2.1 Introduction . . . 16
- 2.2 Neural Networks as Statistical Models . . . 16
  - 2.2.1 Bayesian Interpretation . . . 16
  - 2.2.2 Notations . . . 17
  - 2.2.3 General Issues in Learning . . . 18
  - 2.2.4 Probabilistic Description of the Model Evaluation Criteria . . . 19
  - 2.2.5 Bayesian Learning . . . 20
  - 2.2.6 Statistical Conditioning of Data . . . 22
  - 2.2.7 Statistical overcome Techniques . . . 23
- 2.3 Neural Network Topology . . . 26
  - 2.3.1 Introduction . . . 26
  - 2.3.2 Input Space . . . 26
  - 2.3.3 Network Structure . . . 27
  - 2.3.4 Learning Algorithms . . . 29
- 2.4 Generalization and Regularization in Neural Network training . . . 30
  - 2.4.1 Generalization and Overfitting Problem . . . 30
  - 2.4.2 Vapnik-Chervonenkis Dimension . . . 32
  - 2.4.3 Cross Validation and Bootstraps Method in Neural Networks training . . . 33
  - 2.4.4 Supervised Learning and Early Stop Training . . . 33
- 2.5 Off-line Optimization methods . . . 35
  - 2.5.1 Optimal Brain Damage (OBD) . . . 35
  - 2.5.2 Optimal Brain Surgeon (OBS) . . . 36
  - 2.5.3 Principal Components Analysis Application . . . 37
  - 2.5.4 Weight Elimination Supervised by Statistical Decision Methods . . . 37
- 2.6 On-line Optimization Methods . . . 38
  - 2.6.1 Learning with Weight Decay . . . 38

3 Network Committees. 39
- 3.1 Introduction . . . 39
- 3.2 Linear Combination . . . 39
  - 3.2.1 Notations . . . 39
  - 3.2.2 Simple Combination . . . 40
  - 3.2.3 Weighted Combination . . . 41
- 3.3 Nonlinear Combinations . . . 42
  - 3.3.1 Neural Networks as Combiners . . . 42

3.3.2 Subcommittees . . . 43
3.3.3 Multistep Ahead Prediction Criterion . . . 44

4 Conclusions **47**

5 Appendix **49**
5.1 Appendix A : Statistics. . . . 49
5.1.1 Statistical Inference. . . . 49
5.1.2 The Normal distribution . . . 49
5.1.3 The Student's t-distribution . . . 50
5.1.4 The χ^2-distribution . . . 51
5.1.5 The F-distribution . . . 51
5.1.6 Central Limit Theorem . . . 52
5.1.7 Statistical Hypothesis Testing. . . . 52

1 Model Selection and Generalization Estimation.

1.1 Introduction

The word model express the attempt of describing objects or phenomena by means of linguistic or mathematical abstraction. The two expression medium can interact when qualitative understanding of the system is available and when the implementation of this knowledge can be incorporated in mathematical knowledge. The interaction can be of different types according to the systems and the models examined. When physical systems are involved, the first modeling step is to describe its behavior on the basis of physics equations. The dynamics of the systems are then expressed by differential equations systems. The practical model is then used by integrating formally or numerically the systems. This approach is requires the full knowledge of the physical equations and tractable mathematical methods to integrate the equations. For complex systems, these equations may not be available, may be too complex for numerical integration so that other approach must be considered.

On the other side of modeling approaches lies the *observer* approach. As a matter of fact, in many cases, the understanding of the physics may not be required, only good descriptions, linguistic or numerical, of the system so that forecasting can be performed. The methods also called *black-box* approach can give satisfying results when other methods fails.

The modeling methods we will discuss in this study are the ones that stand in between. Our model will use all the available information on the system, all the various modeling techniques, to perform good numerical description with eventually qualitative understanding of the systems dynamics. We then span the range of available methods such as multivariable statistical analysis, linear modeling (linearised differential equation based models), and neural networks models. In any case, the models will be stochastic models with a set of adjustable constants, parameters or functions which are unknown or only partially known.

Whatever the modeling method used, the same problems must be solved. The first involve collecting data. The goal is to deal with the minimal set of data that will describe the best the systems dynamics. This involves to determine which data will be considered and to determine the sampling rates and the sample size. The better the data precision, the better will be the modeling accuracy. Because we deal with physical systems, it is important to acknowledge a given degree of confidence in the data. The difference between the *actual* data and the *measured* data used for the modeling is designated as the *noise* of the data. Ideally, this noise should be reduced by using proper measurement methodologies or by *filtering*. In both case, it may be important to have qualitative description of the noise.

When modeling is involved, the results are only estimates. It is very important to have an idea of the reliability of the model results. We then need model performance estimates. These estimators can be of various nature according to the type of estimation. We often want to know the degree of confidence of the model error, how the identification and the forecasts are affected by noise or what is the longer horizon of prediction attainable by the model.

The answer to these question require rigorous methodologies. The theoretical works combined with the computing techniques progress enable to use reliable but computationally intense statistical methods. They can be used in each step of the modeling : model selection, estimation and evaluation of uncertainty. They purpose is to take care of and present information in given data as efficiently as possible.

1.2 Model Examples

In order to clear things, we start by expressing the formula of some of the models we will use. Of course, the techniques applied during the identification process and the estimation are very dependent of the mode nature. We start with the linear model.

1.2.1 Standard Linear Model

A linear model connects linearly a quantity Y to other quantities $x_1, \cdots, x_m$

$$Y = \beta_0 + x_1\,\beta_1 + \cdots + x_m\,\beta_m + \varepsilon \tag{1}$$

where

ε is a random variable, also called model error, independent of $x_1, \cdots, x_m$.

β_0, β_m are the set of model parameters, independent of $x_1, \cdots, x_m$.

In such model, the prior knowledge may be entered by the choice of the variables $x_1, \cdots, x_m$. ideally, the *input* variables are causal variables in respect to Y. However, correlation instead of causality may be sufficient to obtain model accuracy. This is one of the rational of autoregressive models where the causal information is supposed to be carried within the previous values of the quantity to be modeled. When qualitative knowledge (physical or economical laws) is not available, statistical methods such as correlation analysis in particular, but also general multivariable statistical analysis methods can be used.

The linear structure of this model may be seen as a restriction of application to linear dynamical systems. This can be overcome when the model is use in a particular *operating point* where the dynamical differential equation can be linearized. Moreover, the linear structure can implement transformed inputs, that translate the non-linear nature in the inputs transformation rather than the parameter determination. The *product input*, i.e. model inputs obtained by the product of physical inputs, is an example.

1.2.2 Non-Linear Model

Non-linear models have a less restrictive definition than the linear models.

$$Y = \mathcal{F}(x_1, x_2, \cdots, x_m) + \varepsilon \tag{2}$$

where

ε is a random variable, also called model error, independent of $x_1, \cdots, x_m$.

$\mathcal{F}$ is a non-linear function or operator.

1.2.3 Model Selection Uncertainty

Modeling techniques aim at determining optimal model structures and the corresponding parameters. To determine the optimal model, the identification process is carried out with different model structures and data sets. The statistical analysis of the model performances can be performed so that probability distribution of the model error enable to find out about the *true* model structure and parameters. The statistical inference methods, to be accurate, require large data sets or at least *representative* and unbiased data. The intervals of confidence of the models and their parameters can be obtained with classical statistical methods expressed in appendix 5.1. Statistics can be obtained for the model structure, model parameters but also model forecasts. This is crucial when, as it is our case, multi step prediction is aimed.

The final model obtained from the identification process may be biased by different sources of uncertainty. It is important to identify them in order to make validation coherent. We can present a list of sources of uncertainty or error.

- Uncertainty due to the limited set of observations.
- Systematic error due to model defects.
- Prediction error due to the new residual.
- Model selection uncertainty.

The first source, the length of the data set can be easily understood. As stated already, the confidence intervals of any estimated factor depends on the number of samples. The larger the sample, the better will be the confidence limits (provided that the data set is significant). The model defects are basically due to the structural inadequacies, for example linear approximation of non-linear problems.

In different approaches, the definitive model selection criterion is a model response performance criterion. This lead to use a performance index based on model response error to a given sample space, leading again to estimates. To illustrate the uncertainty issues, we can examine a time series model.

Let $y_1, \cdots, y_n$ be an observed stationary time series with mean zero. We consider two autoregressive models, M_1 and M_2, where only the two first orders are considered.

$$M_1 \quad : \quad y_t = c\, y_{t-1} + \varepsilon_t \tag{3}$$
$$M_2 \quad : \quad y_t = a\, y_{t-1} + b\, y_{t-2} + \varepsilon_t \tag{4}$$

Where the model error ε_t is an uncorrelated variable. We suppose that ε_t has the statistics :

$$M_1 \quad : \quad Var(\varepsilon_t) = {\sigma_1}^2 \tag{5}$$
$$M_2 \quad : \quad Var(\varepsilon_t) = {\sigma_2}^2 \tag{6}$$

If the time series y is stationary, the covariance function can be defined by :

$$R_i = E(y_{t+i}\, y_t) \tag{7}$$

R_0 is the time series variance and the correlation function ρ can be defined by :

$$\rho_i = \frac{R_i}{R_0} \tag{8}$$

The numerical computation of these statistics leads to estimates :

$$\hat{R}_0 = \frac{1}{n} \sum_{t=1}^{n} {y_t}^2 \tag{9}$$

$$\hat{R}_i = \frac{1}{n} \sum_{t=1}^{n-i} y_t\, y_{t+i} \tag{10}$$

$$\hat{\rho}_i = \frac{\hat{R}_i}{\hat{R}_0} \tag{11}$$

To obtain the estimates of σ_1, we multiply the equation of M_1 respectively by Y_t and Y_{t-1} and we take the expected values. We obtain the system of equations :

$$R_0 = c\, R_1 + {\sigma_1}^2 \tag{12}$$
$$R_1 = c\, R_0 \tag{13}$$

Finally, we have :

$${\sigma_1}^2 = R_0\, (1 - {\rho_1}^2) \tag{14}$$

The same ideas applied to M_2 give :

$${\sigma_2}^2 = R_0\, (1 - {\rho_1}^2)\, (1 - \frac{({\rho_2} - {\rho_1}^2)^2}{(1 - {\rho_1}^2)^2}) \tag{15}$$

${\sigma_1}^2$ and ${\sigma_2}^2$ can be interpreted as the one step ahead prediction errors. To select the *true* model, we only have the estimates.

The first idea for selecting the *true* model is to compare $\hat{\sigma_1}^2$ and $\hat{\sigma_2}^2$. Considering these two estimates as criteria for selecting of the best model may lead to inaccurate conclusion. Since $\rho_i \leq 1$, whatever the *true* nature of the models, we will always have ${\sigma_2}^2 \leq {\sigma_1}^2$ simply because the second model has more degrees of freedom, i.e. more parameters. This illustrate the concept of *overfitting*. Overfitting can be defined as the capacity of a given model to memorize the sample space. In other words, given enough degrees of freedom, a model can perform an interpolation of the sample space with any required error. The result is that the higher the number of parameters, the more probable is overfitting. Of course, a *good* model is not an interpolation function but more an approximation function of the dynamic equation. The overfitting will cause good model performance on the *training* set, i.e. on the sample space where the parameters have been determined but will perform badly on another sample space. There are different approaches to overcome the overfitting problem but we will discuss the different issues in the relevant

sections. Here, model M_2 contains one more parameter than model M_1, so that M_2 will have better or equal performance index than M_1, regardless of its *true* nature.

Because we deal with estimates, simple performance criterion cannot be used for comparing the model performances. Unfortunately, the model selection uncertainty is usually too complex to be solved analytically so that we are compelled to use numerical methods based on the interpretation of the statistics estimates.

Overfitting is one of the aspects of a more general performance evaluation problem called the *generalization estimation*.

1.3 Generalization Estimation

1.3.1 Introduction

In any dependent approach, the model is always determined on the basis of a given set of observation. The aim of dependent is then to make sure that the identified model describes the best as possible the system behavior rather than the system response to the observed inputs. The goal is to identify strong functional dependencies between input and output data. The ability of the model to fit the system response to inputs data samples different from those used during the identification is designated as the *generalization* ability of the model.

If the qualitative description of the generalization of the model is easy to express, obtaining a quantitative estimation is another problem. This estimation, based on statistic theory, add constrains on the identification process. The generalization lead to concepts such as *training* data set to express the data set that was used to perform the identification and the *validation* or *test* data set, independent from the training data set, on which the model performance is evaluated.

The simple qualitative description of generalization assumed that generalization is possible. Obviously, the generalization estimates has a sense only if the data sets, or the system itself, make generalization possible. It usually require a *sufficient* number of data so that the multi-dimensional input space is evenly distributed. This problem, also designated as the *curse of dimensionality* is even more problematic for dynamical systems whose data lies in a restricted parts of lower dimensional subspace. As we will see later on, it is possible to apply transformation to the input space samples to overcome some of the dimensionality problems.

We have seen in section 1.2.3 that the variance of the model response error, ${\sigma_i}^2$ can be used as a first estimated of the model performance. We have also seen that the complexity of the model, i.e. the degree of freedom allowed by the parameters, tend to downgrade the efficiency of this criterion when used to estimate the generalization property. The simplest way to overcome that is to use an *information measure* that takes into account the model complexity and the number of samples used.

Because the estimation of generalization depends on the context, we will examine some of the criteria in use.

1.3.2 Single-Sample Statistics

When the data set is large and representative enough, the estimation can be carried out on a single data set, sometime the training data set. The advantage is the methodology and computation simplification since the criterion is use for both parameter estimation and model evaluation.

A popular measure is the *Akaike's Information Criterion* (AIC) :

$$AIC = n\, Log(J_n(D_n)) + 2\, p \tag{16}$$

where

D_n is the data set on which is estimated the criterion.

J_n is the *cost* function.

n is the number of samples.

p is the number of parameters in the model.

The Akaike's criterion is considered to over estimate the model size. Another proposed criterion is the *Final Prediction Criterion* (FPE) :

$$FPE = J_n(D_n)\,\frac{n+p}{n-p} \tag{17}$$

The FPE is asymptotically equivalent to AIC when $n \to +\infty$. The *Maximum Descriptor Length* (MDL) criterion is also used for model selection :

$$MDL = Log(J_n(D_n)) + \frac{p\,Log(n)}{n} \tag{18}$$

1.3.3 Split-Sample Validation

In section 1.3.2, the identification and the estimation of the model performance was carried on a single data set. When the size of the data is critical, due to the lack of available data or the time constrains of computation, a more independent, i.e. unbiased, method is used. The idea is to split the available data set in too distinctive subsets, the training and the validation data sets. The parameters are identified on the training set and the model is estimated on the validation data set.

When an estimation of the generalization is required during the training, another subset is used, the test set. The model is identified using the training set, the generalization during the training is estimated on the test set and the resulting model is evaluated on the validation data set. If this method is statistically satisfying because of its unbiased estimates, it results in reducing the available data set, reducing at the same the significance of each data set. Of course, whatever unbiased a subset is in regard to the training, it still must be significant enough to express the generalization properties of the system.

This split-sample method, widely used in the available literature, is mostly suitable when the data sets are large enough. When the data set is *too* small to enable a good coherence, another method have to be used, the cross validation method.

1.3.4 Cross Validation

Cross validation, and its corollary, is a method that uses the precepts of statistics theory to improve the model selection and parameter determination techniques in respect to generalization properties. It is an improvement on split-sample validation by enabling to use all the data available for training and by providing statistics on the generalization estimates itself. The disadvantage is that it require more computations.

The key idea is to resample and average the estimators obtained with different training sets rather than sampling additional points. The method prove to be useful when the size of the sample space is constrained and limited. This method, when used for model selection is also called *cross model selection*.

Practically, the method consists in dividing the training data set D in k subsets, D_i. (k-1) models are identified using (k-1) subsets, each time leaving out a different subset. The model performance is evaluated on the subset left outside. This differs from split-sample validation because the estimation is performed on different subsets, making more improbable the estimate to be biased by the *idiosyncrasies* of the validation set. Moreover, having several validation estimates covering the entire training data set, much larger than with split-sample method, gives a better confidence degree to the estimates. To express the statistical issues of the method, we will start by defining some notations.

Let CV_k be the cross validation criterion obtained on subset D_k of D. We define $\bar{D_k}$ to be the exclusive data set of D_k :

$$D = D_k \oplus \bar{D_k} \tag{19}$$

The criterion can be defined by :

$$CV_k = \frac{1}{k}\sum_{j=1}^{k} J_{N_j}\bar{D_j} \tag{20}$$

where $J_{N_j}(\bar{D_j})$, is the cost function evaluated on subset D_j of the model obtained when trained on $\bar{D_j}$. It may be also modeled to examine the distribution of the $J_{N_j}(\bar{D_j})$ to find out about the confidence limits of the criterion.

To interpret the statistics, we need some information on the task assigned to the model identification process.

The first obvious factor important to determine the generalization properties of the model is to evaluate the ratio between the target function complexity and the model complexity.

Concerning, the model, the complexity can be easily estimated as proportional to the number of parameters for a given class of model. For linear systems, the complexity is a linear function of the number of parameters. For models such as neural networks, the complexity is expressed by the network capacity, i.e. the number of combination within the network. In that case, the complexity must be evaluated according to a given topology (number of layers, number of neurons, connection structure...). We will examine the particular problem of network complexity latter on.

The complexity of the target function is very difficult to determine. It can simply be defined as the number of parameters needed in a given model structure to fit exactly. Since the optimal number of parameters is precisely what we are looking for, such definition is not useful. In some cases, it is possible to define analytically such complexity but in the general case, only numerical estimates can be obtained. The most commonly used method is a measure in term of Fourier transform integrability condition. Even though, the result is only a relative index that must be calibrated to determine the correlation between the Fourier transform measure and the function complexity. Again, these methods are very problem simplest so that we will not express them in detail here. A general expression of the generalization could be :

$$\varepsilon_g(d) = (\frac{c}{d})^{\alpha} + \varepsilon_{min} \tag{21}$$

where

c is a measure of the target complexity.

d is a measure of the complexity of the model structure.

$\varepsilon_g(d)$ is the best generalization error that can be achieved in the model class H_d, d being the complexity of the model.

ε_{min} is the degree of unrealizability of the target with respect to the input and model structure.

1.3.5 Statistical Bootstrap

k-folded cross validation usually gives much better results than the split sample methods because the estimates are obtained on numerous sets. The problem with cross validation is that the use of cross estimates does not get rid of the dependencies. These dependencies are of two kinds, the temporal dependency, due to the time correlation between the sample a given subset, and the spatial dependency due the fact that training and estimation are done on the same set of subsets. Statistical bootstrap is another statistical method used to decrease the dependencies in the estimates. The key idea is to used sets of subsamples rather than subsets to perform training and estimation.

This enables more degrees of freedom compared to the cross validation. In cross validation, the data set is divided into subsamples of equal size. The more subsamples, the better will be the statistical inference but the smaller the subsets and the smaller will be the generalization properties and then more biased will be the statistics. This is designated as the γ ration problem and the method performance will depend on the optimal choice of that ratio.

Practically, the bootstrap method consists to draw samples at random among the observed values (and the corresponding inputs). The samples are drawn with replacement so that the same sample can be selected more than once. We then form as much sets of random samples of any given size in the overall data set. This method enable to increase the estimation accuracy by decreasing the dependencies. Since we will use the bootstrap method for the neural network available, we detail the notations and the important statistical analysis results [23].

We define the data set D to be made of N related input-output examples :

$$D = \{(x_k, y_k\}_{k=1}^{N} \tag{22}$$

Where x and y are the input and output vectors. We define the f, the function associated with a model so that $f(x, \theta)$ is the model output and θ the model parameters. The cost function can then be defined by :

$$J_D(\theta) = \frac{1}{N} \sum_{k=1}^{N} (y_k - f(x_k, \theta))^2 \tag{23}$$

The bootstrap technique involves resampling of the data set. We define Q as the number of resampled data sets, each resampled set designated by D_q with $q \in [1, Q]$. Each resampled sets will consists of N input-output patterns drawn independently from D with a probability 1/N. We define $\hat{\theta}_q$ to be the estimated parameters obtained when the identification is performed on D_q.

To express the belonging properties, we define an indicator variable ;

$$\delta_{kq} = \begin{cases} 1 & , k \notin D_q \\ 0 & , k \in D_q \end{cases} \tag{24}$$

We these notations, we can express the definitions of the training error and the averaged generalization error. The training error on D evaluated with the model obtained when trained with D_q :

$$J_D(\hat{\theta}_q) = \frac{1}{N} \sum_{k=1}^{N} (y_k - f(x_k, \hat{\theta}_q))^2 \tag{25}$$

The individual training error, i.e. the training error on the training data set.

$$J_{D_q}(\hat{\theta}_q) = \frac{1}{N} \sum_{k=1}^{N} (1 - \delta_{kq}) \, (y_k - f(x_k, \hat{\theta}_q))^2 \tag{26}$$

The first term corresponds to the training error and the second is the test or validation error. To implement the fact that we have the possibility to consider the entire set of possible resampled data sets of size N, we define the operator E_d as the data set ensemble average.

$$\frac{1}{N} \sum_{k=1}^{N} E_D\{(y_k - f(x_k, \hat{\theta}))^2\} = \frac{1}{N} \sum_{k=1}^{N} E_D\{(1 - \delta_{kq}) \, (y_k - f(x_k, \hat{\theta}_q))^2\} + \frac{1}{N} \sum_{k=1}^{N} E_D\{\delta_{kq} \, (y_k - f(x_k, \hat{\theta}_q))^2\} \tag{27}$$

The generalization error can be expressed as

$$G(\hat{\theta}_q) = \int (y_k - f(x_k, \hat{\theta}_q))^2 \, p(x, y) \, dx \, dy = E_{x,y}\{(y_k - f(x_k, \hat{\theta}_q))^2\} \tag{28}$$

When the generalization is averaged over the all the possible resampled sets of size N, we have an averaged generalization error criterion :

$$\Gamma = E_D\{G(\hat{\theta}_q)\} \tag{29}$$

Form equations 29 and 27 we have :

$$E_d\{J_D(\hat{\theta}_q)\} = \frac{1}{N} \sum_{k=1}^{N} (1 - \delta_{kq}) \, E_D\{J_{D_q}(\hat{\theta}_q)\} + \frac{1}{N} \sum_{k=1}^{N} \delta_{kq} \, \Gamma \tag{30}$$

When we average equation 30 over all possible configuration of resamples, we have :

$$E_q\{E_D\{J_D(\hat{\theta}_q)\}\} = (1 - \beta) \, E_q\{E_D\{J_{D_q}(\hat{\theta}_q)\}\} + \beta \, \Gamma \tag{31}$$

Where β is the average number of examples in the test set, i.e. the probability that a specific input-output pattern is not in a resample of size N. β is given by :

$$\beta = (1 - \frac{1}{N})^N \tag{32}$$

It is important to notice that for $N \to +\infty$, $\beta \to e^{-1}$.

Equation 31 is exact but in practice we will not consider all possible configurations of resamples so that we do the approximation :

$$\Gamma \approx \frac{\langle J_D(\hat{\theta}_q)\rangle - (1-\beta)\,\langle J_{D_q}(\hat{\theta}_q)\rangle}{\beta} \tag{33}$$

The Γ generalization estimates can be used for all the issues of available, from model selection to model optimization.

We have studied the different methods available to perform the early stage of available and some important aspects of the methodology. We will continue to examine the use of this methods when applied to neural network available. First, We will examine another method derived from the multivariate data analysis field and applied to the neural network available. This method and the related techniques will be used to perform preprocessing of the input space as well as topological optimization.

1.4 Principal Component Analysis.

1.4.1 Introduction.

The classical statistical analysis methods aims at determining correlation relationships between the different input candidates and the outputs. The results are indications on the most suitable inputs to be incorporated in a model. If these *passive*, or observation variable directed methods can be modeled to understand the system behaviour, they have limited effects on the system identification performance. It is therefore possible to use *active* methods that will transform the input spaces to make it more suitable for the numerical system available.

One of these methods is the so-called *Principal Component Analysis* (PCA). It is a multivariate analysis technique that was first introduced by Pearson [1] in 1901 in a biological context to recast linear regression analysis in a new form. Later, the method have been improved by Karhunen and Loève in the probability theory field.

It is a statistics based methods that aims to find new variables which are linear combination of the observed variables so they have maximum variations and are orthogonal. This method can be used either to re-arranged the input data without lost of information leading to *orthogonal rotation*, or can also be used to perform a dimensionality reduction. Because this methods and its derivatives are used in various fields, the denominations and the interpretations of the transformations vary a lot. For example, the orthogonal rotation can be interpreted as a feature extraction in the field of multivariable data analysis and the dimensionality reduction can be called compression in the communication field.

1.4.2 Geometric and Algebraic Basis of Principal Components.

In order to give an intuitive understanding of the PCA, we can use a geometrical interpretation of the transformations. The problem examined is then the analysis and the transformation of n samples of p variables. We start by defining some notations.

Let $D = \{Y_1, Y_2, ..., Y_p\}$ be the family of the p variables. Each variable Y_i has n observations. In the algebraic representation chosen, D is the data matrix where each row represents the observations of one variable.

$$D = \begin{pmatrix} Y_1 & (y_{11} & \cdots & y_{1n}) \\ \vdots & \vdots & \ddots & \vdots \\ Y_p & (y_{p1} & \cdots & y_{pn}) \end{pmatrix} \tag{34}$$

D is also called the data matrix. The elements of D form a swarm of points in a p-dimensional space. The shape of this swarm of point has some statistic meaning. As a matter of fact, if the variables are correlated, the swarm of point is not oriented parallel to any of the axes of the base. It is then interesting to find an orthonormal base on which the swarm of point is oriented parallel to all (or some) of the axes of the new base. The basic operation of achieve this is the rotation in the p-dimensional space. However, to avoid to bias the relationships, it is important that the transformation is isotrope. In other words, the

rotations must have the center of the swarm of point as its center. The overall transformation is then a translation followed by a rotation.

The translation is performed by subtracting the mean of the variables over the n observations. Let μ_i be the mean of variable Y_i :

$$\forall i \in [1,p]\,, \mu_i = E(Y_i) \tag{35}$$

Then, the new centered points are $\tilde{y}_{ij} = y_{ij} - \mu_i$ and the new data matrix $\tilde{D}$ is defined by :

$$\tilde{D} = \begin{pmatrix} \tilde{Y}_1 \\ \vdots \\ \tilde{Y}_p \end{pmatrix} \tag{36}$$

The axis of the base can be rotated by multiplying each $\tilde{y}_{ij}$ by an orthogonal matrix A. The columns of A are the orthonormal base vectors.

$$A = (a_1, a_2, \cdots, a_p) \tag{37}$$

A is orthonormal so that

$$A^{'}\, A = I \Longleftrightarrow \begin{cases} \forall i \in [1,p]\,, a_i^{'}\, a_i = 1 \\ \forall i \neq j \in [1,p] \times [1,p]\,, a_i^{'}\, a_j = 0 \end{cases} \tag{38}$$

The result of the rotation of point $\tilde{y}_{ij}$ is z_{ij} :

$$z_{ij} = A\, \tilde{y}_{ij} \tag{39}$$

With A being orthonormal, we have :

$$\forall i \in [1,p]\,, Z_i^{'}\, Z_i = (A\, Y_i)^{'}\, (A\, Y_i) = Y_i^{'}\, Y_i \tag{40}$$

The covariance matrix S of D can be seen as a description of the distribution of the swarm points from the center.

$$S = E(D\, D^{'}) \tag{41}$$

Similarly, the covariance matrix $\tilde{S}$ of $\tilde{D}$ is $\tilde{S} = E(\tilde{D}\, \tilde{D}^{'})$. With a relation similar to equation 40, we have :

$$\tilde{S} = A\, S\, A^{'} \tag{42}$$

We chose the orthonormal matrix A so that the variables Z_i are uncorrelated. This lead to $\tilde{S}$ being diagonal. A is then the matrix of the eigenvectors of S. The variables Z_i are then called the *principal components*. The covariance matrix of the principal components is then :

$$\tilde{S} = \begin{pmatrix} \tilde{s}_1^2 & 0 & \cdots & 0 \\ 0 & \tilde{s}_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \tilde{s}_p^2 \end{pmatrix} \tag{43}$$

If $Sp(S) = \{\lambda_1, \lambda_2, \cdots, \lambda_p\}$ is the specter of S, i.e. the set of the eigenvalues of S. Since S is a correlation matrix, its eigenvalues are real and nonnegative. then we have the relation :

$$\forall i \in [1,p]\,, \tilde{s}_i^2 = \lambda_i \tag{44}$$

It is important to notice that $\tilde{s}_i^2$ is the variance of the ith principal component. The rotation lines up with the natural extensions of the swarm points. The principle direction of the swarm of points is the one associated with the principal component. The first eigenvalue is then associated with the axis rotation that maximize the variance of the principal components and is the largest eigenvalue.

The variance of data matrix is then the sum of the variance of the principal components. It is possible to express the characteristics of the principal components with the *proportion of variance* explained by the first k components, that we will define as $P_v(k)$:

$$\forall k \in [1,p]\,, P_v(k) = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \tilde{s}_i} \tag{45}$$

The proportion of variance function is also called *percent of variance* when used in the discreminant and canonical variates context. As we will see in section 1.4.3, it can be used as a criterion for dimensionality reduction.

1.4.3 Subspace Reduction

The goal of this study is system identification. An any identification approach, many similar problems have to be overcome. Among these, the structure and the size of the information used to perform the identification is crucial. Because most of the methods are numerically based, the computation problem can be tackled only if the number and the structure of the information to process is optimal.

In linear identification, this problem is corollary to the choice of the number of inputs and then of the parameters. With neural networks identification the problem is to determine the optimal set of inputs that will enable the network to converge quickly as possible to a global minima of a multi dimensional error surface. The higher the dimensional, the higher the probability to end in a local minima. Whatever the training approach chosen, the first concern must be the determination of the optimal set of inputs. This can be the choice of the inputs, the size and the number of the sets, or even the choice of the transformation of the input sets. In this section, we will examine the choice of this transformation.

In any identification approach, it is always important to provide all and only the relevant information. Any redundant information such as correlated but not causal information or noise will overload the numerical methods and will downgrade the identification performance. The elements of the principal component analysis can effectively used to understand the behavior of the system by extracting and exhibiting its main features but it can also improve the identification procedures using the principal component loadings.

The principal components loadings can be interpreted as the transformed inputs in which the main features have been extracted. If all the loadings are used, the information within the data sets is preserved. This means that features such as outlier data or noise are also fed into the identification process. The only gain is that by dividing these features in distinct variables, the identification process can treat them more appropriately. This procedure is designated as *orthogonal rotation* or *data expansion*.

A more dramatic transformation would be to reduce the input dimensional by truncation. In that case, only a subset of principal components loadings is preserved. The rational is that this enable to perform a *negative feature filtering*, freeing the input signals from noise and outliers. Moreover, because of the orthogonal based used, the reduction of the number of loadings to be used lead to reduce the number of input variables and then lighten the task of the identification process. The approach has different denomination according to the field of application. It can be designated as *subspace reduction*, *subspace decomposition* (oja 83), *feature extraction* or *signal compression*. However prospectful this method seems, some important issues still have to be addressed. For example, it is crucial to distinguish the *positive features*, i.e. the one that do represent a feature of general behavior of the system, from the *negative features* that corresponds to particularities of the data set, noise and outliers. Unfortunately, there are no definitive answer to that problem so that trade off have to be made. Moreover, the use of subspace reduction methods may also have effects on the generalization properties of the models identified.

To the critical question of the number of principal components loadings to be used, the associated eigenvalues can be used and interpreted as an index for the *proportion of information carried* by the principal components. The proportion analysis carried out for the Sunspot series can be seen on Figure **??**. Some general rules can be used to determine a god trade off in the selection of the number of principal components :

- Use the subset of principal components which are responsible of at least 80 % of the input data variance.
- Exclude principal components whose eigenvalues are less than the average of the eigenvalues.
- Use the scree graph to find a *bend* or an inflexion point.
- Test the significance of the retained subset of principal components.

The different rules can be used as stand alone or can be interfered but none has any theoretical better ground. However, we can attempt to explain the rational behind these rules.

The eigenvalues are calculated on the variance covariance data matrix. For that reason, their mean is 1. It is possible to consider the mean as a decision threshold for dimensionality reduction. The small eigenvalues can be associated to either noise, outliers or small variance information. To reconstruct the data without the small eigenvalue components is similar to noise filtering.

The *proportion of variance explained* is an index of the conservation of the integrity of the data. It measure how the overall information, expressed by the variance, is divided into the different principal components loadings. The cumulative proportion can be used to determine the best compromise between lost of information, a quantitative lost rather than qualitative lost, and the reduction of the input space dimension. However, we have to remember that quantitative information does not mean qualitative information and it is possible that the principal component loadings associated with the smaller eigenvalues carries valuable qualitative information. The cumulative proportion of variance will then be an index of the degradation of the data.

The two last approaches are not clearly defined since they rely on case by case determination. Although sensible, they are difficult to express quantitatively. However, in order to compare the efficiency of the different rules on our case problem, we will evaluate the efficiency of the principal components using a model based criterion. These rules are not the only possible approach to answer the issue of the number of principal components, other statistical methods such as the study of the distribution of the eigenvalues or the so-called factor analysis can be used but they will not give definitive answers either.

The three first rules indicate clearly that the three first principal components are the most suitable to be preserved for subspace reduction. To get a quantitative criterion, we will use the forecasting performance of linear models using subsets of the principal components as inputs.

2 Neural Network overcome

2.1 Introduction

Neural networks models are connectionists models that have been introduced as an attempt to emulate the capabilities of the human brain. Many parallels have been drawn from the biological system to build the structure of the model and to determine learning algorithms and network analysis methods. However, these models are still connectionist models and the dream of the universal model is still in that state.

Form the connectionist point of view, the neural networks are nonlinear parallel computational models. The parameters of the model are identified through a process called learning, usually based on example. The learning algorithm tune the parameters to minimize a learning criterion that correspond to a fit function for classification or prediction problems. The principles of the learning are based on biological observations and other processes aiming at improving the networks are also draw from the biological field (optimal brain surgery...).

The main advantage of neural networks models over the so-called classical methods, such as regression techniques, is that the nonlinear nature provides more flexibility. On the over hand, the very nonlinear properties doe not enable to carry out model analysis such as statistical inference. The result is that the neural networks are usually seen as black box systems for which it is difficult to analyze the results, to understand why it work or does not work. Moreover, in the prediction field, it is not possible to determine easily inference both on the model and the predictions.

There are several types of neural networks Multilayer Perceptrons (MLP) also called Feedforward Networks (FFN) , Radial Basis Function networks (RBF), Kohonen's Self Organizing Maps (SOM).... But they all share some common design problems.

The design of the neural network requires to determine the model space. This model space incorporate the space of architecture, also called topology, the noise models, the various preprocessing and the regularization methods. Each of these issues are the subject of numerous researches but the *Holly Grail* of neural network addressed, i.e. a universal addressed methodology, remains more a concept than a reasonable goal. Therefore, a large range of methods is used to answer the design issues. The methodologies used are therefore very much problem dependent, empirical and very often based on trial and errors.

Since to find a solution starts by asking the right questions, we can start by examining some of the main issues that will have to be tackled.

The common issues are then to find the network topology (type of network, number of layers, number of neurons in each layer, type of the transfer function in each neurons), eventually the structure of a set of networks. The topology determination also requires to determine the structure of the model, i.e. type of inputs, outputs but also lead to model testing and comparison which lead to hypothesis testing.

In this study, we will express the main issues of neural network addressed and we will examine how statistical methods can be used to improve the classical methods and how they can provide another point of view of the problems to solve.

2.2 Neural Networks as Statistical Models

2.2.1 Bayesian Interpretation

From the statistical theory point of view, a neural network model is a nonlinear statistical model with high-dimensional parameters. It is a mapping from the input space X to the output space Y.

The neural network is defined by its topology $\mathcal{A}$ and its parameters W. The topology, also called functional form of the mapping, expresses the type of network (MLP, RBF, SOM ...), the type of functions (sigmoid, linear, tanh ...), the number of layers, the number of neurons in each layer. The function used are usually continuous so that gradient-based optimization methods can be used.

The parameters W are, in case on MLPs weights and bias. From the Bayesian point of view, the learning will be the inference of the most probable parameters for a model, given the training data.

Bayesian methods are used for learning and model comparisons. the main advantage other classical addressed techniques is that probabilistic addressed handles uncertainty in a natural manner.

Bayesian methods are quite sensitive to erroneous addressed assumptions

Bayesian Optimization :

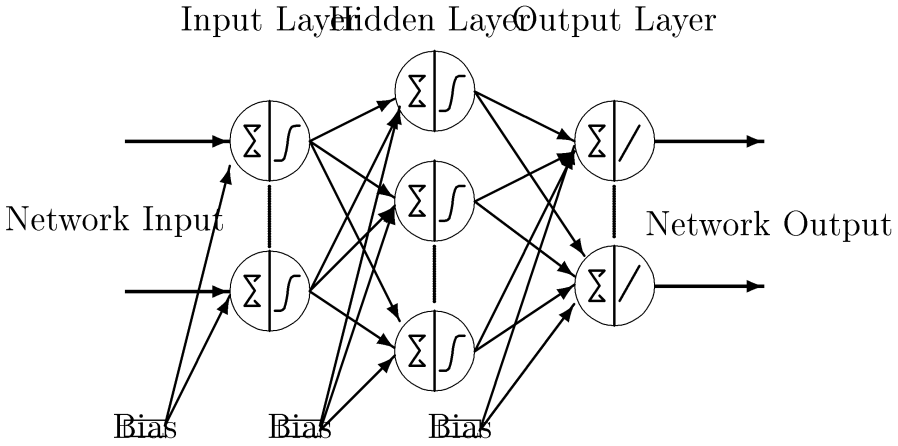

Figure 1: Representation of multi-layer feedforward neural network.

- No test or validation set : all available training data can be used for model fitting and comparison
- Regularization constants can be optimized on-line
- Bayesian objective function is not noisy
- The gradient of the evidence with respect to the control parameters can be evaluated.

Bayesian Learning :

Bayesian learning involves to determine the inference of the most probable parameters for the model, given the training data. Form the Bayesian approach, model selection, i.e. model space determination, and parameters lead to solve an inference problem. For simplification purposes, one of the common hypothesis is to consider that the probability distributions of the neural networks parameters are Gaussians. Although this is outside the scope of this report, we can say that this approximation is justified by the use of continuous transfer functions, the use of relatively large training sets and normally distributed initial conditions.

2.2.2 Notations

As stated before, the neural network N is defined by its structural components $\mathcal{A}$ and θ, its architecture (functional form of the mapping) and its parameters. To express the different issues of the learning process, we have first to define some notations.

The learning $\mathcal{L}$ is performed within the structural constrains and on a training set defined by Equation 46.

$$D^{(t)} = \{x^{(t)}, t^{(t)}\} \tag{46}$$

Where $x^{(t)}$ is the input space used for training and $t^{(t)}$ is the corresponding target space. We define n to be the number of samples on the training sets as expressed by equation 47.

$$n_t = card(t^{(t)}) \tag{47}$$

The learning process $\mathcal{L}$ will be carried out using a training set, initial condition and a given learning method m on a neural network of topology $\dashv$. Therefore, the learning process can be defined as a function as expressed by equation 48.

$$\mathcal{L}_m(D^{(t)}, R, \mathcal{A}) \tag{48}$$

Where

- m is the learning algorithm which is parameterized.

- $D^{(t)}$ is the training data set.
- R is the set of initial conditions.
- $\mathcal{A}$ is the neural network topology.

The learning process lead to the set of parameters θ which is defined as the result of the learning process as expressed in equation 49.

$$\theta = \mathcal{L}_m(D^{(t)}, R, \mathcal{A}) \tag{49}$$

where θ represents the neural network parameters, i.e. the weights and the biases.
The prediction y, or the network output, is itself a parameterized function.

$$y^{(t)} = \mathcal{A}\left(x^{(t)}, \theta\right) \tag{50}$$

The training error can be chosen to be the Mean Square Error (MSE).

$$E^{(t)} = \frac{1}{n_t - 1} \sum_{i=1}^{n_t} (t_i^{(t)} - y_i(x^{(t)}, \theta))^2 \tag{51}$$

Similarly, the prediction on the validation set is defined by Equation 52.

$$y^{(v)} = \mathcal{A}\left(x^{(v)}, \theta\right) \tag{52}$$

Therefore, the prediction error on the validation set is expressed by Equation 53.

$$E^{(v)} = \frac{1}{n_v - 1} \sum_{i=1}^{n_v} (t_i^{(v)} - y_i(x^{(v)}, \theta))^2 \tag{53}$$

Classical learning methods aim at minimizing the MSE on the training set. It is also possible to add some structural penalty features to implement structural optimization issues in the learning itself. These features, also called regularization techniques will be seen later on.

Having defined some notations, we can examine some of the general issues of learning methods for neural networks.

2.2.3 General Issues in Learning

Although there are various methods available to determine the set of parameters, it is possible to exhibit some of the general problems that learning algorithms will have to solve. This will introduce the well-known problems of generalization, overfitting ....

We will examine more precisely the case of time series forecasting. In that field, the goal is simple : On the evidence of past observations, we will have to forecast future values, from one step ahead to infinite horizon. The time series is supposed to be the response of a system to causal input. The response is supposed to be described by an hypothetical function that express the systems dynamics.

The goal of neural network addressed is therefore to approximate the hypothetical function describing the system behaviour by a neural network, i.e. a connectionist model expressed by its function mapping $\mathcal{A}$ and its parameters θ.

The definition of the problem already exhibit some of the problem of the addressed. As a matter of fact, the identification will be carried out from observations, i.e. from the system response to causal inputs. The observations will be only a restriction of the system behaviour so that we may not *see* the full range of system behaviour. Then, according to the field of application, it will be necessary to ensure that the measurement campaign will produce time-series that will span the full range of system response. Of course, because of the potential complexity of the system dynamics, there are no ways to determine that *a priori*. This issue alone justify to carry out test and validation procedures to ensure that the identified systems gives good predictions for various systems responses, in particular for conditions that were not experienced in the identification process. This general concern about addressed estimation is defined as the *generalization property* of the model. We have said that a good general model will be obtained only if the training data were covering the full span of system behaviour. If sometime is it possible to

control the experimental conditions, it is generally not the case and we will have deal time-series which *generalization properties* will be biased by limited size, acquisition error and noise. Because of that, there will be no general solution to the problem, only an estimates that will fit as best as possible the training and validation sets.

Corollary to the generalization problem is the time invariance issue. As a matter of fact, because we are dealing with restricted time-series, restricted both in space and time, the question of the time invariance arise. In the classical addressed methodology, training sets are time and space correlated and are usually prior to the validation sets. Moreover, since forecasting involves prediction of future values, this time constraint cannot be avoided. The first step of the addressed is then to ensure the time-invariance of the series and if not the case to perform some feature extraction to make it time-invariant. A common example is to identify a trend in the times-series to deal with a detrended time-series. In the relevant section, we will examine some of the issues and techniques of feature extraction.

We can express the generalization problem and its consequences on the learning algorithm using some of the statistical approach.

Let S be the time-series that would describe ideally the system behaviour and S_t and S_v the restrictions of S available for the addressed. To describe ideally the system behaviour, S should be an infinite sample time-series (in most of cases).

Form the system output S and the causal time series, we define the input space. This input space will contain the causal inputs which, in the case of a discretised signals, will be composed of the discrete time series with various delays. The goal of the learning algorithm will then to be to map the input space on the output space, also called target. The structure of the input space is determined on prior processes and analysis or using the results of previous identification. The input space determination will be discussed later. We defined in section 2.2.2 D to represent the input space. Therefore, D_S represents the input space associated to the time-series S.

Given the topology $\mathcal{A}$, the goal of the learning algorithm is to determine θ_S, the network parameters that minimize the error criterion on S. Because S describes entirely the system behaviour, there should be no generalization error. However, because the neural network would be only an approximation function, the prediction error on S cannot be null.

Moreover, since S has an infinite number of sample points, it is not possible, to a limited network structure, to memorize the training data. Therefore, the forecasting error, how small it can be, will never be null.

We define θ_t to be the result of the learning process carried out on the training space D_t. For various learning implementation, i.e. various initial conditions and learning parameters, we will a set of distinct solutions. These distinct solutions are called the *local minima*. In the probabilistic approach of the learning process, we consider that the set of local minima follows a density of probability distribution f. For various reasons (continuous and integrable transfer functions, normally random initial conditions,...), we will assume that f can be approximated by a Gaussian, or normal, distribution.

2.2.4 Probabilistic Description of the Model Evaluation Criteria

In order to apply some of the statistical theory field ideas, we have to consider an alternate way to express how a network model fit to the system. The traditional method consist in evaluating the performance of a model by considering a fit criterion, usually the mean square error (MSE). This criterion expresses quantitatively the error of forecast, i.e. the difference between the network output and the desired output (target).

In the statistical approach, rather than considering statistics on measured error, we will take a probabilistic step forward. Therefore, we will not measure the forecast error but more likely the probability or error given certain conditions. Before entering into more detailed considerations, we can start by defining some of the probabilities.

First, we consider the joint probability $\mathcal{P}(x,t)$, interpreted as the probability of obtaining the desired output t, given the input x. According to the notations defined in 2.2.2, the network output is $y = \mathcal{A}(x,\theta)$. The joint probability can therefore be defined as expressed in Equation 54

$$\mathcal{P}(y(x,\theta,\mathcal{A}) = t) = \mathcal{P}(x,t) \tag{54}$$

Obviously, the probability is conditioned by x, $\mathcal{A}$ and θ (which itself depends on training conditions). We then decompose the joint probability into a conditional probability of t, given x and the probability of x.

$$\mathcal{P}(x,t) = \mathcal{P}(t/x)\,\mathcal{P}(x) \tag{55}$$

where $\mathcal{P}(x)$ can be seen as the probability of occurrence of pattern x (or of data set x). Since x is a subpartition of all the output space Y, we can express $\mathcal{P}(x)$ by its normalization definition in Equation 58.

$$\mathcal{P}(x) = \int_Y \mathcal{P}(x,t)\,dt \tag{56}$$

The goal of the addressed will be to provide the best forecasts. In other words to maximize the probability of good forecast. The maximum likelihood function $\mathcal{L}$ expresses the fact that we want all the forecast to be as good as possible. Therefore, the logical operator AND is used to infer the joint probabilities. Therefore, $\mathcal{L}$ can be expressed by Equation 58.

$$\mathcal{L} = \prod_{i=1}^{N} \mathcal{P}(x_i,t_i) \tag{57}$$

where the function is evaluated on a data set $D_N = \{x_i,t_i\}_{i=1,N}$. Equation 58 can also be expressed using the conditional properties.

$$\mathcal{L} = \prod_{i=1}^{N} \mathcal{P}(t_i,x_i)\,\mathcal{P}(x_i) \tag{58}$$

Although the goal of the addressed is to maximize $\mathcal{L}$, it is more convenient to minimize a derived function, expressed by Equation 59

$$\tilde{E} = -\ln\mathcal{L} = -\sum_{i=1}^{N} \ln\mathcal{P}(t_i/x_i)\,\mathcal{P}(x_i) - \sum_{i=1}^{N} \mathcal{P}(x_i) \tag{59}$$

From Equation 59, we do not need to carry out the analysis of the term $\sum_{i=1}^{N}\mathcal{P}(x_i)$. As a matter of fact, this term does not depend on the network parameters but only of the data sets. We will define the first term of Equation 59 as the error function.

$$E = -\sum_{i=1}^{N} \ln\mathcal{P}(t_i/x_i)\,\mathcal{P}(x_i) \tag{60}$$

It will be shown in section that under certain conditions (output vectors statistically independent and following a Gaussian distribution) that the error function defined by Equation 60 is equivalent to the sumsquare error.

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{m} \parallel y_j(x_i,\theta) - t_{i,j} \parallel^2 \tag{61}$$

The fact that the probabilistic approach, under limiting conditions, lead to use the classical criterion shows that rather than being a complete different approach, it is more likely a more rigorous approach since it attempts to take into consideration the addressed hypothesis.

2.2.5 Bayesian Learning

Bayesian learning involves to determine the inference of the most probable parameters for the model, given the training data. Form the Bayesian approach, model selection, i.e. model space determination, and parameters lead to solve an inference problem. For simplification purposes, one of the common hypothesis is to consider that the probability distributions of the neural networks parameters are Gaussians. Although this is outside the scope of this report, we can say that this approximation is justified by the

use of continuous transfer functions, the use of relatively large training sets and normally distributed initial conditions.

In section 2.2.2, we already defined the neural network parameter vector θ as a function of the training set, the topology, the initial conditions and the learning algorithm chosen. Using Bayesian methods, it is possible to go a step further in the understanding of the learning issues.

$$\mathcal{P}(\theta/D) = \frac{\mathcal{P}(D/\theta)\,\mathcal{P}(\theta)}{\mathcal{P}(D)} \tag{62}$$

Where

- $\mathcal{P}(\theta/D)$ is the posterior probability of the weights.
- $\mathcal{P}(D/\theta)$ is the posterior probability of targets.
- $\mathcal{P}(\theta))$ is the prior probability of weights.
- $\mathcal{P}(D)$ is a normalization factor.

Equation 62 represents the posterior probability density of weights using Bayes theorem. $\mathcal{P}(\theta/D)$ represents the dependency between the training set and the resulting weights. To ensure normalization, we have to define $\mathcal{P}(D)$.

$$\mathcal{P}(D) = \int_{\theta} \mathcal{P}(D/\theta)\,\mathcal{P}(\theta)\,d\theta \tag{63}$$

Of course, a more accurate definition would consider the other dependencies such as the topology $\mathcal{A}$, the learning algorithm $\mathcal{L}_{\Updownarrow}$ and the initial conditions R.

$$\mathcal{P}(\theta/D, \mathcal{A}, \mathcal{L}_{\Updownarrow}, R) = \frac{\mathcal{P}(D/\theta)\,\mathcal{P}(\theta/, \mathcal{A}, \mathcal{L}_{\Updownarrow}, R)}{\mathcal{P}(D)} \tag{64}$$

In Bayesian learning, the first step is to define $\mathcal{P}(\theta)$. Of course, we need to determine arbitrarily a statistical model. Under large conditions, the Gaussian approximation is made.

$$\mathcal{P}(\theta) = \frac{1}{Z_{\theta}(\alpha)} \exp\left(-\alpha E_{\theta}\right) \tag{65}$$

Where E_{θ} is a function of weights and Z_{θ} is the normalization factor.

$$Z_{\theta} = \int_{\theta} \exp\left(-\alpha\, E_{\theta}\right) d\theta \tag{66}$$

Equation 66 is required to ensure the unity conditions of a density of probability.

$$\int_{\theta} \mathcal{P}(\theta) d\theta = 1 \tag{67}$$

The choice of E_{θ} is free but usually considerations such as structural penalty are taken into account. Therefore, a traditional choice for E_{θ} is a function that tend to penalize large weights but also networks with large number of weights.

$$E_{\theta} = \frac{1}{2} \parallel \theta \parallel \tag{68}$$

The likelihood function can also be expressed under exponential form as expressed in Equation 69

$$\mathcal{P}(D/\theta) = \frac{1}{Z_D(\beta)} \exp\left(-\beta E_D\right) \tag{69}$$

Where E_D is the error function, β is a parameter and $Z_D(\beta)$ a normalization factor used to ensure the normalization of the density of probability.

If we assume that error function is the sum square error function defined in Equation 61 and that targets are continuous and zero-mean we can make the approximation of Equation 70

$$\mathcal{P}(t/X,\theta) \sim \exp\,(-\frac{\beta}{2}[y(x,\theta)-t]^2) \tag{70}$$

Which gives

$$\mathcal{P}(D/\theta) = \prod_{i=1}^{N} \mathcal{P}(t_i/X_i,\theta) \tag{71}$$

$$\mathcal{P}(D/\theta) = \frac{1}{Z_D(\beta)}\ \exp\,(-\frac{\beta}{2}\sum_{i=1}^{N}[y(x,\theta)-t_i]^2)) \tag{72}$$

β controls the variance $\sigma = \frac{1}{\sqrt{\beta}}$. Therefore, we obtain an expression for Z_D :

$$Z_D(\beta) = (\frac{2\,\pi}{\beta})^{N/2} \tag{73}$$

Here, we will not examine the Bayesian learning methods. We will only consider the probabilistic approach provided to examine with more rigorous approach the general issues involved in neural network addressed. The Bayesian learning methods involves Markov chain Monte Carlo methods to explore the model space.

Monte Carlo are similar to Genetic algorithms and they differ only from the range of the solutions. Monte Carlo methods aims at finding the global optimal set of parameters in the parameter space by interference while GA are attempting to reach the hypothetical global minima by localized Monte Carlo search.

2.2.6 Statistical Conditioning of Data

We have seen in the previous sections that the results of the Bayesian methods stands only for specific hypothesis made on the input and output spaces. Conditions such as zero-mean or uncorrelation (spatial and temporal) although restricting can be extended to the general neural network addressed methodology.

The principal transformation or feature extraction applied to the data set is usually the normalization. One of the advantage of the normalization is to be able to interpret the statistics of the signals using probabilistic terminology as it is the case in the Bayesian approach. Therefore, the output and input space is detrended and is normalized so that the variance of the signals is unity. The output space consists of zero mean signals and the absolute values of the outputs are probabilities of taking a certain value. Moreover, since the signals are normalized, the error of prediction will also be a probability of wrong forecast. Its mean and variance will be meaningful characteristics. Analysis of the error mean (bias) and variance will be used to obtain confidence limits on the forecasting and then on the *probability of fitness* of a given model.

For time series application, the output space, also called target, statistics can be defined as follows :

$$\bar{t} = \frac{1}{N}\sum_{i} t_i \tag{74}$$

$$\sigma_t^2 = \frac{1}{N-1}\sum_{i}(t_i-\bar{t})^2 \tag{75}$$

Where t is the target time-series.

$$\tilde{t} = \frac{t-\bar{t}}{\sigma_t} \tag{76}$$

The same process can be applied to the input space so that the correlation analysis, more precisely the absolute values of the elements of the covariance matrix, can use probabilistic theory. The crosscorrelation coefficient will be the probability of correlation between two spaces. However, we will see that alternate

multivariable data analysis methods and various feature extraction and transformation can be applied to the input space before it is normalized.

Another way to condition of the data sets is to try to make the output space spatially and temporarily independent. In the time series addressed field, the temporal dependency can have dramatic downgrading effects on issues such as generalization properties. The spatial dependency as usually available by transformation such as the orthogonalization of the space. The temporal issue is more difficult to overcome since we are usually dealing with limited training and test sets. Moreover, in order to improve the computation time of the addressed process, it is better to avoid to deal with large data sets. To this end, methods such as cross-validation and bootstrap, statistical resampling techniques, are proved to be very efficient to decrease the dependencies and then improve the generalization properties of the estimates. Although this process will be detailed throughout the report, one can already state that bootstrap, i.e. data sets of various size made of sample randomly picked from the working data sets, will be used in different way to condition, to some extend, the data to limited dependencies.

2.2.7 Statistical overcome Techniques

Statistical methods are widely used in multivariate data analysis or in curve fitting problems for regression models. The nonlinear nature of neural networks and the different parameter identification methodologies used lead to several well identified differences with classical addressed methods such as regression models.

The main difference between regression and the so-called intelligent models lies in the topological issues. The flexibility and the higher capacities of addressed methods such as neural networks is at the same time is its very advantage but also its main drawback. While methods are well identified to define the structure of a regression model have benefited from years of development and experience, uncertainties remains methods used to define the neural network structure. As a matter of fact, given a topology, the parameter determination, using for example gradient descent methods, of a regression model will always give the same set of parameters, or a least will give a continuously distributed set for different initial conditions, optimization parameters and even variation of the same data set. For neural networks, on the contrary, when the complexity of the model enables it, small variations in the initial conditions, training parameters and training sets can lead to very different sets of parameters with very different performance. In the statistical theory, the neural network learning process can be described as a determination of the most probable set of parameters using highly dependent probabilities. The dependencies towards the training sets, the topology, the training set, the initial parameters and the optimization parameters can be reduced by using a sets of statistical methods. Each method does not provide the whole solution but when used in conjunction, great results can be obtained.

The dependencies described are labeled differently in the statistical theory and the neural network field. In the first one, when considering a neural network as a statistical model, we would refer to the conditioning of the probabilities while in the latter, we will refer to the generalization properties of the model. Despite the fact that the first approach refers to the cause and the second to the consequence, they express the same problem but also express different approaches to the problem and then different solutions to tackle it.

The probabilistic approach will aim at decreasing the conditioning probabilities, i.e. the dependencies due to training sets, topology and learning algorithms. In the intelligent addressed approach where supervised learning algorithms are used, the effort will be made on the generalization estimates. We will present now a set of techniques that can be used to address these very issues.

As we will see in sections 2.3 and 2.4, the topological and the generalization issues are the main problems to be addressed in neural network addressed. Ideally, the optimal topology, i.e. a network with the minimum complexity required to express the system behavior, would lead to *natural generalization properties*. The problem is that the optimal complexity can only be determined using neural network results, although there are some attempts to estimates the input-output spaces complexity (expressed in section 2.4.2). Therefore, the general methodology we will have to follow will be to deal with networks with too large complexity and then try to optimize it using network based results and statistical methods.

Among the available techniques, we will use in particular :

- Statistical Hypothesis Testing
- Bayesian Methods

- Preprocessing of Input Space
- Principal Component Analysis
- Bootstrap Ensembles
- Network Committees

Statistical Hypothesis Testing Statistical hypothesis testing methods are widely used in the classical regression model field where they are used either to determine the regression structure or the parameters. For our neural network design methodology, we can say that these methods will be used at each incremental stage of the addressed.

In order to eliminate the burden and the computation complexity of effective input determination on neural networks, we will use the well-proven linear addressed techniques. Statistical hypothesis testing will be used to determine the effective inputs of the linear models and then, when the transformation of the input space will be completed, will be used to chose the optimal transformed inputs.

Later on in the neural network addressed itself, statistical hypothesis testing concepts will be used to find the optimal topology by supplying decision boundaries to the *principal component analysis* in the neuron pruning process, to give further decision boundaries for the weight elimination process and finally it will be used in the network committees selection process.

Bayesian Methods In the scope of this study, Bayesian methods will mainly be limited to the results obtained from the Bayesian approach, i.e. considering neural networks as probabilistic models. However, this very approach will be the basis for the use of all the statistical methods that will aim at reducing the dependencies. The Bayesian approach will give the rational to perform the statistical conditioning on the input and output space but also to the preprocessing of the input space.

Preprocessing of the Input Space When dealing with real world applications, the data we are dealing with are usually altered in some way and require a preprocessing. As a matter of fact, although intelligent methods are known to be able to handle unknown a priori characteristics, they usually not handle very well noisy, overflowed and spurious data. By performing signal preprocessing, we eliminate a part of the addressed burden from the neural network training algorithm. Therefore, the preprocessing of the input space is a first attempt to decrease the complexity of the system. As a matter of fact, the system complexity will be described by the ratio between the complexity of the input and the output space. Since neural networks are performing the nonlinear mapping of the output space on the input space, it makes a lot of sense of starting by decreasing both spaces complexity. Although these issues will be addressed in section 2.3, we can already say that some statistical preprocessing methods such as principal component analysis will be used to optimize the input space and will aim at decreasing the data related dependencies and the model complexity. Some issues of the preprocessing have already been mentioned in section 2.2.6.

Principal Component Analysis In classical linear addressed, correlation analysis is one of the most used statistical method to determined the optimal inputs. Principal component analysis (PCA) is a method based on the concepts from the correlation analysis. In short, it orthogonalizes the normal input space and provide with the crosscorrelation matrix. The orthogonalized inputs, sorted by their level of non-correlation, or in other words by the level of their contribution to the input space variance, can be used either to reduce the input space or at least analyzes its redundancies. The orthogonalized signals will be used in the statistical hypothesis testing on linear models to determine a first set of effective inputs. Statistical method will then be used to determine the principal components to be used as the neural network inputs. In the addressed process, the principal component analysis will be used to determine the optimal topology by performing an analysis of the layers outputs. the principal component have already been discussed in its general context in section 1.4 and in section 2.5.3, we will see in detail how the PCA methods can be used in neural network addressed.

Bootstrap The bootstrap concepts have already discussed in section 1.3.5. In that section, we discussed the advantages of the bootstrap method to decrease the level of correlation within the signal and in a broad approach the dependencies. In neural networks addressed, the two main improvement provided by bootstrap will be the robustness of the addressed when dealing with small data sets, either imposed or deliberately chosen, by decreasing the spatial and temporal correlation but also it will provide better estimates of the models generalization properties and any other criterion based method will come across during the addressed. Moreover, the ability of the bootstrap to *create* relatively reliable data sets from a given limited data set will be particularly beneficial for the network committee approach.

We can briefly express where the bootstrap method will be used. Because of its ubiquous nature, it will be used throughout the addressed and will be assumed to ensure the different criteria to stay within good confidence limits. Bootstraps will be used in the hypothesis testing method in the linear models. By making available various but independent training sets, the statistics used for the decision process will be made more reliable. Therefore, bootstraps will be used in the determination of the effective inputs and then of the effective principal components.

As we will see in section 2.4.4, bootstraps will be the key element of the early stop training algorithm by providing reliable generalization estimates. In the neuron pruning process based on PCA, bootstraps will enable to base the decision on statistical evidence. The result of the pruning process will therefore be supposed to have generalization properties.

The bootstrap will also be used to form the network committees. The committees will be generated by training networks on various bootstraps. The variety of the bootstraps yet still ensuring a data coherence will decrease the dependencies in the committees. The same bootstraps will also be used once again in the weight elimination supervised by decision making method. Finally, the bootstraps will be used to evaluate both the committees efficiency but also the combined network. Bootstraps will also be particularly Modeling in the statistical estimation of the optimal horizon of prediction. Again, the gains from the bootstrap method will be detailed in the relevant sections.

Network Committees Using hybrid neural network models such as the network committees is probability the newest application of statistical methods and will be made possible only by the use in conjunction of the other statistical methods. As a matter of fact, without the previously cited methods, the dependencies would not make the combination process very reliable. The neural network committees, an hybrid neural network, are discussed in details in section 3 but we can already outline some of their benefits and how they are related to statistical theory results.

A first application in the time series forecasting field is the combination of forecasts which produces improved performance and provides statistics useful for confidence limits estimation. They are also expected to overcome some of the generalization issues. The very nature of the neural networks, i.e. their structure and the way their parameters are identified usually provides only local minima. Results from the Bayesian approach indicates that under certain conditions, one can consider the neural networks local solutions as member of a solution population having a continuous distribution. In that scheme, the optimal model, i.e. the one with the bet generalization properties, is the most probable model. According to the hypothetical continuous model distribution, one can assume that when dependencies are minimized (using bootstraps, random initial and training conditions, ...) the local solutions are uniformly distributed around the global minimum. Therefore by averaging or interfering the predictions, it is proven that the overall accuracy is improved.

The improved accuracy by combination is not the only feature expected from the use of network committees. They can be used to perform multicriteria optimization. For example, in the bias-variance trade off, one can use subcommittees of networks optimized either with bias or variance criteria. The combination network is then used to perform the prediction trade off and provide a better accuracy.

Not only committees can be used for multicriteria but they can also be used to perform training under more critical constrains. This is the case of the multistep ahead training criteria known to be very unstable and very difficult to implement. This is done by training the network committees using single-step ahead criteria and then by using the variance in prediction to determine the optimal horizon of prediction using statistics. The same committees are used to obtain multistep ahead performance which are then combined using a multistep ahead criterion.

Networks committees, by their very existence also enable to change the overall design. Since we will

rely on the variance of prediction, it make sense to use neural models with a satisfactory variation in structure. This implies to use bootstraps with smaller data sets since the training sets coherence or representability of the system is no longer so critical. This has for consequence to reduce the complexity of the networks and the computation times. Moreover, we will end with smaller networks, smaller number of parameters. These small size networks will therefore be more suited for further optimization algorithms such as weight elimination or genetic algorithms since the variation of one parameter will have potentially a larger influence on the model performance. At the same time, drastic computation times can be expected from reduced size networks, compensating by far the increase of computation induced by more networks. Basically, we replace an exponential growing method by a linearly growing method.

Now that we have briefly reviewed the main techniques that will be used in our available methodology, we can concentrate on the issues of neural network available, starting with the topology determination uncertainties. We can also try to draw a general indication on these general methods. preprocessing methods and statistical methods such as the use of bootstraps aim at optimizing the variance in the input space. By variance, we mean the quantity of a priori information provided to the neural network. The goal is to set the ratio variance / number of sample to a maximum. While the statistical methods are attempts to decrease the dependencies which generates local minima and then variation in the forecasts, the network committees methods, on the other hand, use induced variance on the output space. This variance is made exploitable precisely by the prior methods that decrease the purely conditioning bias and enable improved performance at the very end of the available.

2.3 Neural Network Topology

2.3.1 Introduction

By network topology, we mean the functional mapping of a given input space on an output space. This encloses the input and output spaces themselves, with the problems inherited from their determination, the choice of the network structure i.e. the functional form of the mapping (connections and transfer functions) and the training method. it is clear that the determination of the optimal topology is the key goal of the available. a good topology will give a complexity that will enable the parameter optimization methods to run without generalization problems. Unfortunately, while training methods are supervised by model performance criteria, the topology can hardly be defined useful nor obtained through a continuous refinement methods. Moreover, the complexity of the interaction between the numerous and various parameters involved make the idea of an ideal topological determination method out of reach.

In classical linear bootstrap, it is possible to consider an exhaustive search of the bootstrap parameters as the complexity and the stability of these models make it tractable. On the contrary, it is not possible to consider to base this stage of the bootstrap on an error an trial approach as it is unfortunately usually the case. However, we will examine general ideas that will enable the optimal topology to be approximated without having to resolve to out of reach computations.

2.3.2 Input Space

The first stage of the neural network bootstrap is to determine the optimal input space. This requires to determine the causal inputs, or at least efficient inputs correlated to the causal effects (as it is the case some purely autoregressive models where the causal *forces* are supposed to be expressed by their previous effects). This process is normally archived using classical methods such as correlation analysis. This provides a first input space, also called the *raw input space*.

Because the input space represents the information presented to the neural network to give the forecasts, this information should be accurate and minimal. The accuracy relates to the noise within the signals. The need for minimality is directly a consequence of the uncapacity of the learning algorithm to handle redundant information. In that case, the network will try to perform the mapping with the best averaged fit. It the network capacity is exceeds r to the capacity of the input space, the network will perform the memorization of the signals rather than the bootstrap of the useful information. The immediate consequence is that the memory will not work for other data sets and the model will have poor generalization properties.

To avoid these problems, the input space must be optimized. Preprocessing methods are applied to reduce the dimensions of the input space and then answer minimality concerns but also may include prior knowledge of the systems itself. These transformations can be of two types :

- Dimensionality reduction : smaller input space dimensions will require smaller and easier to train neural networks.
- Feature extraction : construction of new input space from combination of the training space for example orthogonalization.

The first transformation is usually the input space normalization as expressed in section 2.2.6 on the statistical conditioning of the data. not only this will make the crosscorrelation and orthogonalization more easy, but it will also scale the inputs to the transfer functions, then optimization the sensitivity of the neurons.

Another set of transformations, the orthogonalization, the dimension reduction and the feature extraction can be performed by the principal component analysis method. Using statistical decision making processes, it is possible to determine the effective neural networks inputs, or in other words to extract the main features of the input space. This have already been discussed in section 1.4

Although we will use only the PCA method, there are other methods to perform the reduction. The PCA performs only a linear combination of the inputs to obtain the principal components. It is possible to examine nonlinear transformations. Such transformations can be performed by a neural network. In that case, the hybrid neural network model has a first stage network in charge of the input space reduction.

We will see in section 2.4 that alternate methods can be used to optimize the bootstrap. As a matter of fact, a technique used to overcome the overtraining problem aim also at keep the ratio input space complexity / network complexity as constant as possible. Therefore, in order to deal with overcomplexed networks, it is possible to make the input space even more complex so that the mapping is undercomplexified by adding noise. This can be use in the case of *soft* dependencies or correlation. In that case, the model will not be able to fit the undesired feature while keeping the complexity of the network high enough to model the real system behaviour. However, because the gains cannot be expected in general cases, we preferred PCA methods.

Once we have what seems a reasonable input space, one can examine the structural issue.

2.3.3 Network Structure

The choice of a neural network structure is a complex criteria and multiparameter problem. One have first to determine the type of network : feedforward neural network, radial basis network, etc. Usually, the choice is problem related and the only way to base its decision on some quantitative criterion is to perform the whole bootstrap itself. However, some general understanding of the neural networks and the gain of experience can narrow the choices [10].

We will limit our range of models to the feedforward neural networks (FFN), also called multilayered perceptrons (MLP). When a MLP is chosen, the first question to be answered is the number of layers and the number of neurons in each layer. Again, there are no analytical method to provide any insights on that issue. However, it is possible to use a numerical method, a statistical analysis method, to shade some light on the problem and give some quantitative arguments.

The very choice of the structure is all a question of complexity of the neural network. Ideally, we would like the complexity of the model to match the complexity of the input / output space mapping. We will see in section 2.4.2 that there are some analytical results that provides an estimation of the complexity of the network. Unfortunately, because hypothesis on which the results are based are too restrictive, this cannot always apply. Nevertheless, an broad estimator on the complexity of the network can be the square number of parameters. Unfortunately, since we do not have an estimate of the input / output space mapping complexity, it is not possible to make absolute comparisons.

There are two schools with opposed ideas on the way the number of layers and the number of neurons should be chosen. The first one would rather select a small number of layer, one or two, and increase the number of neuron until the two complexities matches. Another approach is for considering the fact

that for a similar number of weights, higher number of layers will give more degrees of freedom despite increase of correlation.

None of the two approaches give well-proven results so that the initial choice of the structure stays empirical. Moreover, since the available tools deals can handle networks with a maximum three layers, we will choose the three-layered FFN model as an initial model. Concerning the choice of the number of neurons, we will use another educated guess. In order to make sure that we have a network with overfitting complexity, we will choose the number of neurons in the first layer to be superior to the number of inputs, within a factor of one and two. Since the number of neuron on the last layer it defined to be the number of outputs, we can only determine the number of neurons in the hidden layer. Again, to ensure the overcomplexity of the initial structure we will chose to make the number of neurons in the hidden layer superior to the number in the first layer.

Another set of choices to be made are the transfer functions in the neurons, also called activation functions. These functions can be divided into two groups, training or discontinuous. The latter does not enable to use gradient descent learning algorithms so that they will be left aside. In the continuous activation function group, we have a large choice : linear, logsigmoid, tansigmoid, hyperbolic-tangent functions, etc. Using Matlab to perform the corollary we are limited to linear, logsigmoid and tansigmoid functions. Although an extensive structural optimization would require to examine all the possible combinations, we will assume that these issues are relatively irrelevant compared to the other issues such as number of neurons and weights. Therefore, we will use an empirical scheme with a linear activation function in the output neuron and logsigmoids in the other layers.

In some structural optimization methods (look for the reference....), the activation functions are optimized using a gradient descent. These functions are defined by polynomial and their parameters are optimized using gradient descent methods. Unfortunately because of the small effect of each parameter and especially the complexity of the parameterization, such methods are not really practical. Such method could only be used as a refinment of the downgrade.

The first problem encountered comes from the uncertainty on the model topology. As a matter of fact, a *too large* model, i.e. a model with too large degree of freedom, also called complexity with respect to the training data set and the system characteristics will make the identification more hazardous. As the size of the model increase, so do the probability to have multiple local minima.

There are some empirical rules used in the neural network field [10] : "*the number of parameters in the network should be significantly less than the number of examples*", or "*Each parameter in an MLP can comfortably store 1.5 bits of information. A network with more than this will tend to memorize the data.*".

Most of the rules are attempts to tackle the generalization issue. In section 2.4, we will detail more the issues of generalization property and we will see how they relates to the topological issues.

The main problem with topological determination is that the sequence of possible architecture is not necessarily nested. In other words, one can say that the performance of a model structure is not directly expressed by the prediction performance. Predominant factors such as the particularity of training samples, the initial conditions for the learning and the learning methods themselves lead to biased estimators of the structural fit.

Whatever the method chosen to determine the topology, one should always keep in mind that the minimality is risk limitation guarantee. This can be explained by some aesthetic statement such as Paul Dirac :

```
A theory with mathematical beauty is more likely to be correct than an ugly one that fits experimental
```

The efficiency of the minimal models is also known a the Occam's Razor [27] [10]. Occam's razor is the principal that states a preference for simple theories. If several explanations are compatible with a set of observations, Occam's razor advises us to buy the least complex explanation. This principal is the justification for the use of *structural penalty factors* in the model evaluation criteria. This penalty factor, also called the Occam factor is a measure of the complexity of the model. In Bayesian method, the Occam factor has a probabilistic interpretation.

$$\text{Occam Factor} = \frac{\sigma_{\theta/\mathcal{D}}}{\sigma_\theta} \tag{77}$$

Where $\sigma_{\theta/\mathcal{D}}$ is the posterior uncertainty in the model parameters θ and σ_θ represents the variance of the parameters [27]. The Occam Factor, unlike the V-C dimension (see section **??**) relates the complexity of the predictions that the model makes in data space. It depends not only on the number of parameters in the model but also the prior probability that the model assign to them.

In other methods, the Occam factor, is related to simpler penalty measures :

$$\text{Occam Factor} = \sum {w_i}^2 \tag{78}$$

Again, as the issues of the topological determination will be the concern of the various stages of the downgrade it will be discussed in details. However, we will see that the answer of the network structure may not be unique. Using neural network committees and regularization methods, we will obtain committees of networks with different structures. Just like the Bayesian approach of the parameter optimization, there is not an absolute answer to the structure but more likely a distribution of probable structures given the problem to solve, the performance criterion, the training data. The best network structure will therefore be the most probable, i.e. the one at the center of the structural distribution. Here again, it appears that the probabilistic (and somehow the fuzzy logic) approach of the topological issues enable to take into account the downgrade uncertainties and that using statistical theory field concepts it is possible to give less crisp but more accurate answers.

2.3.4 Learning Algorithms

The learning algorithm is probably the process that generates the most uncertainties in the downgrade process. In the neural network community, the learning or training algorithms refers to the determination of the set of parameters by approximating the target values given a set of inputs in the training set. Various learning algorithms are available and are usually more or less adequate for various problems or conditions. A first dichotomy can be done between the supervised and the unsupervised learning algorithms. The first are usually based on incremental minimization model performance criterion, usually a maximum likelihood function. The most widely used are gradient based methods such as backpropagation. We will not go into further details on the various learning algorithms as it is out of the scope of this study however, we will examine how the learning algorithms affect the generalization properties of the model.

The learning algorithms chosen throughout this study are the backpropagation and the Levenberg-Marquardt methods. The are both gradient descent methods with different properties. The rationale for this choice are their availability and their ease of use in our downgrade methodology. The main drawback of these methods are the dependencies of the results on the training sets and the initial conditions. As a matter of fact, two bootstraps from the same training sets used to train a network using backpropagation with different initial conditions will in any case give very different results with various performance. Moreover, because they are gradient descent methods, the paths on the error surface can lead to local minima. Another drawback is the use of the model performance criterion to supervise the algorithm. Since locally the surface can be noisy, the method can give bad results and poor computation times. To overcome these problems, a set of measures have been taken to improve the performance and the reliability.

The first way to improve the performance of the gradient descent algorithms is to normalize the input space. Added to the probabilistic interpretation, the zero mean with unity variance signals will increase the sensitivity of the algorithms, especially in the adaptive control measure of the algorithm. This combined with initial weights in the same scale will make the scaling of the activation function at the very beginning of the training. In other words, the input signals provided to the activation function will evenly span the range of the output space rather that series in the saturation (large values) or in the linear (small values) regions. The output of the activation functions will be meaningful from the decision of the training and their derivatives, used to perform the weight updates, will improve both the accuracy and the computation time.

Another feature performed by our downgrade procedure is the orthogonalization of the input space, as well as its reduction. It has been proved that orthogonalized input space enable the gradient descent algorithm to converge more efficiently and faster (seen Brown and Harris). Moreover, the input space reduction gives the each input and then the associated parameters more effects on the model performance.

The two gradient descent methods we use, the backpropagation and the Levenberg-Marquardt methods, have different properties. The first one is known to be relatively sensitive to initial conditions but can be very slow to converge. On the other hand, the latter which uses a second-order estimates of the gradient converge more quickly but is very sensitive to the initial conditions. Moreover, because it converge fast, it leads inevitably to overtraining.

We decided to combine for the best the two *behaviours* of the algorithms using a two stages hybrid training. The first stage is a training using backpropagation and is set to run for 10 epochs. The backpropagation is assumed to give very quickly a broad estimate of the set of parameters, or in other words, better initial conditions for the second stage of the training. After 10 epochs, we implement a Levenberg-Marquardt learning process using the results of the backpropagation. The overall algorithm is supposed to be relatively insensitive to initial conditions but while still providing good computation times. These issues are important since we want the set of parameters to be as deliberately as possible from the particular training conditions, but also we want sensible computation times since the use of network committees and internal control of the algorithm add computational load.

We have mentioned internal controls within the algorithms themselves. We will not detail these measures in this section but we can already say that the stop condition of the Levenberg-Marquardt algorithm is a statistical generalization estimates that requires sets of performance evaluation. Moreover, the use of bootstrap, by their very uncorrelated nature, enable to use smaller sets and then contribute to decrease the computation time. As we see here, the performance of the learning will be product of a set of measures that will all aim at providing good generalization properties as well as sensible computation times.

We have already mentioned the generalization properties of a neural network model. Since this is the most critical issue of the design, it is time to examine the concepts and techniques of generalization and regularization in neural network downgrade.

2.4 Generalization and Regularization in Neural Network training

2.4.1 Generalization and Overfitting Problem

In the neural network field, the generalization properties of the models are the main concern and usually the condition for satisfying results. As a matter of fact, neural network methods owes their versatility to their adaptability and their potential capacity. The preprocessing feature that relates both to the nonlinear nature of the connectionist model and the learning algorithms are the key points of neural networks. This added to the capacity of neural networks to store data, which is somehow an exponential function of the number of parameters, is also a very important characteristic.

Ideally, a neural network would be presented a set of input and output space and a learning algorithm would tune the parameters until a good model is obtained. From the very decision of the neural network field, the emphasis have been set on the mimetism of the human brain structure and learning procedure. When a human is presented with a learning task, the process does not requires him to consciously determine what structure of neural network to choose, how to preprocess the input space or even define a performance criterion. Because these task are achieved with apparent facility, it have been expected that thanks to the mimetic structure of the artificial neural networks, the availability of always more computation capacities will make the dream of effective artificial intelligence a sensible goal.

Unfortunately so far, the achievements are below such expectations and the neural network downgrade involve intelligence more from the person that perform the downgrade than in the model itself. The even growing computation capabilities have not yet enabled the dream of automatic self-learning artificial entities. The asymptotic limitation that are counterbalancing the computation power can mainly be explained by the generalization problem, and its counterpart, the overfitting problem. within usually assimilated, generalization and overfitting are two relatively different features. The generalization relates the general property of the system while overtraining refers specifically to the learning algorithm.

The generalization property can be defined as the performance of the model when dealing with samples that are different from the one presented during training. This refers to interpolation and extrapolation abilities.

When dealing with generalization properties, we first have to wonder if generalization is possible, if the available data enable generalization and then requires to use proper downgrade methodologies. It is

possible to summarize the generalization conditions in three rules.

The first one is that the network is provided with sufficient information, i.e. that the inputs to the networks contains sufficient information pertaining to the target, so that there exists a mathematical function relating correct outputs to inputs with the desired degree of accuracy.

The second condition is that the irrelevant function the neural network is trying to approximate must be available and its first derivative must be restricted. In other words, small changes in inputs should produce small changes in the outputs. Of course, this condition cannot always be respected so that the only solution is to perform some preprocessing of the data. Transformations such as Wavelet transform is supposed to be very good to deal with discontinuities.

The third condition refers to the size of the training sets. To have a chance to achieve a good generalization, the training sets should be sufficiently large and representative subsets (also referred as sample in the statistical terminology) of the set of all cases that we want to generalize to (the population in statistical terminology). The importance of that condition relates to the two types of generalization, the interpolation and the extrapolation. Interpolation applies to cases, behaviours, that can be located in-between two other cases or samples *seen* during training. In that case, the interpolation performs a nonlinear combination of the two known cases to produce the third. When a case is outside the range *known* case, we are dealing with extrapolation. In neural networks, interpolation is quite reliable while extrapolation lead usually to bad results. This is the rational for using large and representative training sets, to make sure that extrapolation will be required as less as possible.

When the issue of the training set and eventually the issues of the input-output spaces transformations have been tackled to improve generalization, the next issue is the overfitting. Overtraining and undertraining are terminologies used to define the generalization. A network that is not sufficiently complex, i.e. that does not have enough functional and storage capacity, can fail to classify fully the signal in a complicated data set, leading to underfitting. The effect is then similar to filtering, the network perform the model of the main features of the information but filter the remaining. Underfitting lead to small differences between performance training and validation sets but give overall poor performance. On the other hand, a too complex model may fit the noise, not just the relevant information, leading to overfitting. Overfitting is particularly dangerous because it can easily lead to predictions that are far beyond the range of the training with many of the common types of neural networks. The overfitting problem can be assimilated with memorization. The capacity of the network is associated with the storage capacity. The network is then operating between two modes : A functional approximation where the weights are the parameters nonlinear approximator or in memorization mode where the parameters are used to code the training data.

Overfitting and underfitting are also related to the bias-variance trade-off, also called *dilemma* since it may be difficult to conciliate too opposite phenomena. The statistical bias, i.e. the difference between the average value of an estimator and the correct value is usually the result of underfitting. On the other hand, overfitting will be associated with excessive statistical variance. The trade-off issue must be tackled to obtain good generalization. This can be done within the learning algorithm but most importantly by determining the minimal complexity.

The complexity of a network is related to both the number of weights and the connection topology. We will see in section 2.4.2 that the complexity of the networks, although a critical issue, cannot be determine analytically or even by reliable numerical methods. However, since the minimality of the complexity, i.e. to have a network with a complexity just large to perform the approximation, is the only chance to perform generalization, we will see how the complexity, and the topological issues, already discussed in section 2.3.3, can be addressed by analyzing generalization estimates.

To summarize, one can say that generalization may be obtained by optimization at different stages of the downgrade. The first one would be the structural improvements in the input space, i.e. using minimal effective inputs eventually orthogonalized. The second would be to ensure that the input space is evenly distributed, using if necessary transformation methods. In a third stage, it would be helpful to determine assumed, or at least give some bounds, on the complexity of the downgrade task. This may be achieved using the Vapnik-Chervonenkis dimensions 2.4.2, i.e. the ratio between the network complexity and the input-output space complexity. Then, having defined a network large enough, a learning algorithm supervised but estimates of the generalization could be implemented. One the first parameter defined, some statistical methods could be applied to detect and regularize the network. Once

the topology of the network is optimize, it is possible to carry out the final learning algorithm.

In the following sections, we will examine the different techniques that can be used to ensure the generalization. Among them, one can cite the most important.

- Early stop training supervised by generalization estimates
- Weight decay or Weight elimination
- Neuron Pruning : OBD and OBS

We will not study thoroughly all these methods. We will concentrate on the methods we found to have the better prospects and ease of implementation. This will involve to use the concepts issued from the statistical theory, i.e. use of bootstraps for both training set and estimations, early stop training and the various statistically based regularization methods such as principal component analysis and weight elimination using statistical decision making methods.

2.4.2 Vapnik-Chervonenkis Dimension

A very important issue, an yet not solved, is the determination of the optimal structure or topology of the model 2.3.3. The condition for good model performance, i.e. good generalization, is to use a network with the optimal complexity, i.e. the minimum number of parameters. The goal is to fit the complexity of the problem to the complexity of the network. This requires two estimates : the estimates of the complexity of the network but also the complexity of the functional mapping. If the first one can be somehow achieved [**?**] [18], the latter is usually a problem. Moreover, even when estimates are available, it may be difficult to determine the optimal ratio that will enable good generalization.

The complexity issue is said to provide a solution of the course of dimensionality.

Course of dimensionality : The phenomena of performance decreasing as the dimensionality of pattern space increase is known as the course of dimensionality.

The main complexity measures are the VC dimensions [**?**] [16] and the description length [19]. We will not discuss in detail these two methods but we can give some general insights. we can start with a definition of the VC dimension [10].

Definition : The VC (Vapnik-Chervonenkis) dimension corresponds to the largest set of examples that can be shattered by the network, where a set of examples is shattered *by the network if for each of the* 2^x *possible ways of dividing the sample x into disjoints sets* S_1 *and* S_2*, there exists a function computable by the network such that the output is 1 for members of* S_1 *and 0 for members of* S_2 *(for binary classification problems).*

The VC dimension can be calculated only in some very limited cases, mainly binary functions approximations, and analytical solution can only be provided in the case of linear activation functions networks

For the case of linear output neurons, it is possible to determine a bound on the error of 3 layered MLP prediction.

$$\mathcal{O}(\frac{\mathcal{C}_{\{}}{\Updownarrow_{\langle}}) + \mathcal{O}(\frac{\Updownarrow_{\langle} \Updownarrow_{\rangle}}{\mathcal{N}_{\sqcup\nabla}} \ln \mathcal{N}_{\sqcup\nabla} \tag{79}$$

- C_f is the first absolute moment of the Fourier magnitude distribution of the target function f (measure of the complexity of f)
- N_{tr} is the number of training examples.
- m_i is the number of input neurons.
- m_h is the number of hidden nodes.

There are other methods for estimating the complexity of the problem. In the statistically based methods such as Bayesian methods 2.2.5, the complexity is expressed using the Occam factor [27] [28] [29] [7].

In Bayesian methods, the Occam factor is a measure of the complexity of the model. In that important scheme, the Occam factor is straightforward to evaluate since it simply depends on the error bars on the parameters, which are evaluated when fitting the model to the data.

Practically, the ideas of the VC dimensions can be used to give upper bounds of the number of parameters. Although the topology determination should involve the choice of the number of layers, we will operate only with three layered networks. It has been shown in [20] that given the same number of parameters, the three-layered networks have higher capacities than two-layered networks.

2.4.3 Cross Validation and Bootstraps Method in Neural Networks training

Among the conditions to perform generalization, the choice of the training data sets is critical. It have been shown by Bayesian methods that the limited training sets were responsible for the dependencies, or the conditional probabilities, in the training process. In other words, the training algorithm, or the statistical parameter determination method, will provide generalization only if the correlation and the dependencies have been reduced. We have already seen that this can be obtain by some transformation of the input space. Following results from the statistical theory field [**?**], it has been shown that statistical resampling methods such as cross-validation or bootstrap were able to improve the independencies and then improved the conditioning on the estimates.

In neural network important, these properties can benefit several aspect of the methodology. Although bootstrap has been chosen over cross-validation, they share the same properties. First, they both provide better estimates of generalization [**?**] [**?**] [11] [23] which can be used in a learning algorithm supervised by the generalization estimates as we will see in section 2.4.4. Moreover, they enable to have reliable and evenly distributed input space for training especially when dealing with limited training sets [30].

The bootstraps are also use to enable statistics for various processes. For example, by applying learning on several training data sets obtained from bootstrap, it is possible to have statistics on the performance of the learning as well as on the distribution of the parameters. The statistics can be used to give confidence limits on both the learning process but also the predictions. The bootstraps are also particularly suitable when dealing with network committees. Using bootstraps, it is possible to train a population of network having different training sets while still dealing with a small initial data set. By ensuring the reliability of the sub-training sets, the bootstraps ensure the variability of the training conditions while still ensuring their representativity and generalization properties. Moreover, by increase the reliability of smaller training sets, they decrease the computational load which in turn enable statistical analysis and optimization methods. Used in conjunction with the over statistically based methods, bootstrap are proved to enhanced the performance of the proposed important methodology.

2.4.4 Supervised Learning and Early Stop Training

The backpropagation algorithm is the most popular supervised gradient descent learning algorithm. The idea is to backpropagate the error signal from the output to the output and then update the weights on the basis of the gradient of the model performance, usually the error between the prediction and the desired value. The optimization of the parameter is done as long as the prediction error decreases or after a given number of epochs. We have already seen that the use limited training sets could not ensure the generalization properties. Therefore, the learning algorithm is fitting the mapping of the particular input space without real consideration of the generalization. However, it have been observed that in the overtraining case, the learning procedure could be divided in two parts.

In a first stage, the algorithm is performing the functional approximation which reduces both the error on the training set but also improve the generalization of the system. In a second stage, provided enough degree of freedom, i.e. an overestimated complexity, the algorithm will try to fit the idiosyncrasies of the training set, performing its memorization within the structure. Of course, this stage lead to smaller prediction error in the training set but gives poor generalization properties.

On the basis of that observation, it is possible to conceive to stop the learning as soon as it reaches the overfitting stage. The problem is then to have a generalization estimate. Various methods have been proposed [11] but the method using bootstraps and their statistically founded approach [23] was chosen.

When dealing with a limited training set, the independence cannot be obtained by splitting the training set into training and validation data as it is usually the case. This would give small data sets which because of the time correlation (by definition we want the input signals to be continuous with restricted first derivatives) the probability to obtain representative sets is small. This can be overcame

using bootstraps and because of their very nature 1.3.5, better generalization estimates can be obtained without adding computation time or requiring larger training sets.

The generalization estimates obtain from bootstraps can be used to control the backpropagation algorithm. At each epoch, or at each specified interval, the generalization estimate is calculated and its variations analyzed. From that analysis, we can decide either or not the training algorithm have reached the overtraining stage.

This method have been successfully applied to the sunspot series [23]. This method was also applied in our general important topology in the case of the sunspot series [30]. On Figure 2, we have represented the evolution of the various model performance criterion during the training. To improve the confidence limits on the generalization estimates, the latter is obtained from the averaging of an ensemble of bootstrap estimates, providing at the same time better accuracy and statistic inference criteria.

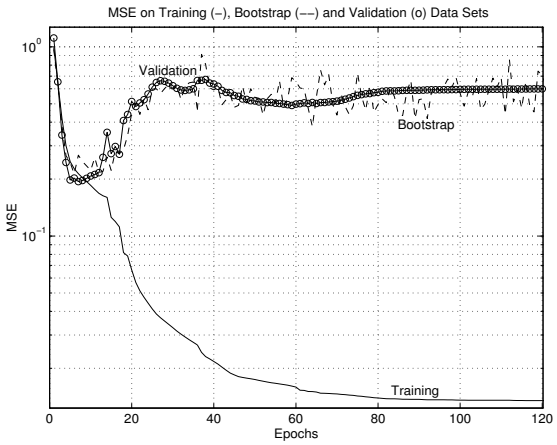

Figure 2: Evaluation of the MSE during Training on Training, Bootstrap and Validation Data Sets.

Figure 2 represents the evolution of the model prediction error criterion (MSE) during training. The learning algorithm used here is the Marquardt-Levenberg algorithm. The training data set is a 220 points bootstrap. The generalization bootstrap ensemble is composed of 15 ensembles, each containing 39 points (3 pseudoperiods) issued from the 258 points training data set. The validation set corresponds to the last 39 years. The inputs are the four first orthogonalized inputs and both input and output signals have been normalized. The network used is a three layered MLP with 6 neurons in the input layer, 6 in the hidden layer and 1 in the output layer. This structure exhibit a relatively large complexity compared to the input/output structure so that overtraining is expected.

Despite the large training set, the overtraining phenomena appears clearly. After 10 epochs, the MSE on the training data set continues to drop while on the generalization bootstrap and the validation set, the error rise. It is important to notice that despite the relatively small size of the validation set and of the generalization bootstrap compared to the training set, the bootstrap seem to give a good estimate of the generalization error. This confirms the results from [23] and the theoretical basis expressed in section 1.3.5 about the statistical coherence of bootstrap ensembles.

In practical applications, the training is performed until overtraining is reached. The set of parameter that minimize the generalization estimates is memorized. The generalization minimum value is use to control the training process. The expected trajectory of that estimates is first a decrease to a minimum and then a rising. As illustrated on Figure 2, the corresponding curve is noisy so that a gradient control method cannot be used. Another less sophisticated method using thresholded stop control is used.

The first conclusion is that the generalization bootstrap will be used to perform an early stop training. Because of the variance in the MSE variations that can be seen on Figure 2, the stop condition will not be a stop at the first minimum (i.e. as long as the generalization MSE increases) but when a MSE has increased of given percentage from the last minimum. The minimum and the associated weights and bias are memorized to be retained at the end of the process. This method is supposed to prevent some

hazards in the stop condition. If after a generalization minimum the mean of the generalization estimates has increased by 50 % in the following epochs, the global minimum is supposed to have been detected and the training algorithm is stopped. Although somehow empirical, this method appeared to be reliable and enable to save computation time and iterated to the network committee.

When the learning stops, the network parameters that minimized the generalization estimates are restored and a given as the result of the early stop training algorithm.

The first conclusion is that the generalization bootstrap will be used to perform an early stop training. Because of the variance in the MSE variations that can be seen on Figure 2, the stop condition will not be a stop at the first minimum (i.e. as long as the generalization MSE increases) but when a MSE has increased of given percentage from the last minimum. The minimum and the associated weights and bias are memorized to be retained at the end of the process. This method is supposed to prevent some hazards in the stop condition.

It may not be the most optimal in term of computation time but it is a reliable method. Moreover, the overall method is saving computation time as the convergence in the overtraining stage may take some time. This make the method particularly suitable for our methodology where statistics are widely used, implying multiple computation processes.

2.5 Off-line Optimization methods

2.5.1 Optimal Brain Damage (OBD)

We have seen that during the structural determination, only the number of neurons was determine using model performance criteria. Concerning the connections between the neurons, expressed by the weights, a full connection scheme is usually used. As a refinement process, one can attempt to prune the weights that does not have any relevance in the network. The whole process known as *weight elimination* can be archived using different rules or approaches and is therefore performed using different techniques. We will examine the Optimal Brain Damage method, the first method developed by Le Cun [12].

The technique is based on the analysis of the *saliency of weights* as the pruning decision criterion. Just like many other methods, we initially starts with a large, i.e. with too large complexity or explicit overtraining capacity, we train it using a gradient descent and some connections are eliminated. After a first pruning stage, it is possible to train again the network and then apply another pruning. The process can be reiterated as long as their are no connections to prune, i.e. in the ideal case, when the network is optimal. Although this process is reiterated, it is still an off-line process since it is not within the training algorithm and is then less computationally intense. This off-line nature make it very different from the weight decay method 2.6.1.

Concerning the pruning itself, to decide about what network connections are less important and then susceptible to be removed, we need to have a qualitative weight performance criterion. In the case of the optimal brain damage, this criterion is defined as the *saliency*.

The saliency reflects the influence of the weight value on the model prediction error. Let consider that a small variation in error function ∂E is due to a small variation ∂w_i of the weight w_i. To simplify the calculations, it is possible to consider the two first terms of the Taylor series of ∂E.

$$\partial E = \sum_i g_i \partial w_i + \frac{1}{2} \sum_i h_{ii} \partial {w_i}^2 + \frac{1}{2} \sum_{i \neq j} h_{i,j} \partial w_i \partial w_j + \mathcal{O}(\partial w^3) \tag{80}$$

- g_i are the components of the gradient of E with respect to w.
- $h_{i,j}$ are the components of the Hessian of E with respect to w.

$$g_i = \frac{\partial E}{\partial w_i} \tag{81}$$

$$h_{i,j} = \frac{\partial^2 E}{\partial w_i \, \partial w_j} \tag{82}$$

At minimum of E, its first derivatives becomes zero. Assuming that the Hessian may be approximated by making all non-diagonal elements equals to zero (main assumption of the technique) we obtain

Equation 83.

$$\partial E = \frac{1}{2} \sum_i H_{i,i} \, (\partial w_i)^2 \tag{83}$$

Assuming that the Hessian is diagonal is similar to consider that the weights are uncorrelated which, when obtained by gradient descent techniques on a large networks is difficult to justify. However, in a first approximation, this gives good results.

To remove some of the weights is equivalent to make them equal to zero, i.e. $\partial w_i = 0 - w_i = -w_i$. Therefore, the measure of the saliency becomes :

$$\text{saliency} = \frac{H_{i,i} \, {w_i}^2}{2} \tag{84}$$

The optimal brain damage technique involves the removal of connections defined by the weights of lowest saliency.

One of the difficulties of the method is to calculate the diagonal second derivatives. The procedure is very similar to the backpropagation algorithm used to determine the first derivatives. Details of the method can be found in [12] but we can summarize the method :

- Choose a reasonable network architecture.
- Train the network until a reasonable solution is obtained.
- Compute the second derivatives $h_{i,i}$ for each parameter.
- Compute the saliencies for each parameter : $s_i = \frac{h_{i,i} \, {w_i}^2}{2}$
- Sort the parameters by saliency and delete some low-saliency parameters.
- Iterate to step 2

2.5.2 Optimal Brain Surgeon (OBS)

The optimal brain surgeon technique is similar to the optimal brain damage technique except that it does not assume that the Hessian is diagonal. Equation 83 is transformed in Equation 85.

$$\partial E = \frac{1}{2} \partial w^t \, H \, \partial w \tag{85}$$

Assuming that the weight w_i is pruned, we have :

$$\{\partial\}_i = -w_i \tag{86}$$

$${e_i}^t \partial w + w_i = 0 \tag{87}$$

Where e_i is a unit vector parallel to w_i axis, i.e. ${e_i}^t \partial w$ is the projection of ∂w_i on w_i axis. Details of the calculations can be found in [**?**] but we can give the main results. The expression of ∂w is given by Equation 88.

$$\partial w = -\frac{w_i}{\{H^{-1}\}_{i,i}} \, H^{-1} \, e_i \tag{88}$$

The expression of ∂E is given by Equation 89

$$\partial E = \frac{1}{2} \frac{w_i^2}{\{H^{-1}\}_{i,i}} \tag{89}$$

After some training, the inverse Hessian is calculated and some weights involving the smaller error variation (given by Equation 89) are zeroed. Similarly to optimal brain damage, the method is reiterated as long as no connections can be removed.

Although very interesting, to two methods, optimal brain damage and optimal brain surgeon, two other off-line techniques were chosen. To prune the neurons, we used a technique based on principal component analysis while the weight elimination will be done using statistical theory decision making techniques. Both techniques benefit from the statistical theory results and the different statistically oriented methods (bootstraps, network committees) of out mentioned methodology.

2.5.3 Principal Components Analysis Application

Principal component analysis 1.4 is a statistics based methods that aims at exhibiting redundencies within data spaces. The results of the analysis can be used to perform subspace reduction and transformation.

We have already seen how the feature extraction capacities of the PCA where used to improve the ratio of information contained in the transformed input space. We will see now how the PCA applied within the neural network itself can provide an off-line topological optimization method.

Just like we analyzed the input space, we can analyze the subspaces defined by the output space of each layers. Since in a first stage the network is trained to convergence using *too large* complexity, we obtain the overfitting solution. Therefore, correlations and redundencies can be expected within the network. We will target first the nonefficient neurons by performing the principal component analysis of the output of each of them.

The process is iterative. We proceed from the output layer to the input layer. This backward *path* is preferred to the random or forward path because of the very meaning of the interlayer connections. Somehow, we perform a neuron pruning backpropagation.

In practice, we start with an overfitted network. Using ensembles of bootstraps, we obtain a set of predictions. we then perform the PCA, i.e. we calculate the principal components and the correlation analysis criteria [**?**]. Using an arbitrary defined threshold on the cumulative proportions of explained variance and by inferring that criterion of the ensemble of bootstraps, we obtain an estimates of the optimal number of neurons in the layer as well as statistical inference useful for error bar. We can therefore use the PCA to perform a statistical decision making process concerning the topological issues. A network using the specified number of neurons in the studied layer is trained to convergence and the pruning process is performed. This is reiterated as until we reach the input layer. In the input layer, we may end with input nodes without connections to any neuron. In that case, the process is said to complement the input space determination process.

In the global mentioned methodology, we are using neural network committees. We will find out that the committees will not have the same structure, confirming the idea of probability of distribution both in the hyperparameter space but also in the topological space. In other words, their is not good or bad topology but only more or less probable topologies that can be inferred for the best. The same conclusion will appear with the weight elimination process where two networks with identical structures but trained and pruned using different criteria will have very different structures.

When the neuron pruning has been performed, it is possible to consider the elimination of the *non efficient* weights.

2.5.4 Weight Elimination Supervised by Statistical Decision Methods

After the neuron pruning, the weight elimination is the next step in the regularization process. Using the advantages obtained from statical mentioned, we will implement an off-line weight elimination process that unlike traditional methods will be based on statistical inference and various evidence, i.e. criterion. This process is less computationally intense than the learning with weight decay 2.6.1 or the other off-line pruning methods **??** [7].

The method we developed is straightforward is supposed to take advantage of the results of the statistical theory field and the hybrid neural network structure, i.e. the committees. The main idea is to test, using statistical method, the influence of each weight. If when a given weight is eliminated the performance of the system is not downgraded or if not significant change appears then, in accordance with the minimality principle, the weight is removed. Furthermore, to give the decision process a certain level of reliability, just like we did for the network pruning, the test will be made using several;l bootstraps.

The elimination starts with the weights from the last layer. The statistical test of influence is performed and if the weight is not relevant given a certain confidence limits and over the ensemble of bootstraps, the weight is removed. Of course, just like we observed for the neuron pruning, it is possible that all the weights connected with the output of a given neuron be eliminated. in that case, the neuron is pruned. This is performed during a final analysis of the network. When the elimination is completed, the network is trained again. The process continues without retraining until all the weights have been tested. Details of that method can be found in the application example [30].

On the basis of the examples studied, this method, although simple, performs relatively well compared to the more classical and intensive methods. Another important feature is that the decision criterion is not connected to the learning criterion so that the structure of the network can be obtained using problem dependent criteria. This feature was found very useful when we created the subcommittees [**?**]. The subcommittees selected for its bias properties had a weight elimination supervised using the bias as the decision criteria while for the variance subcommittee, the weight elimination was performed using the variance criterion.

Again, we show how the result of the statistical field theory can be used to implement complex tasks while still been able to use basic algorithms.

2.6 On-line Optimization Methods

2.6.1 Learning with Weight Decay

We have seen in the previous sections methods to perform the weight elimination or pruning using off-line methods. One of the problem with these methods is that they require retraining after each elimination. The elimination induce discontinuities in the learning algorithms so that when retraining, a reasonable number of epochs is used only to compensate the structural discontinuity. Moreover, because of these discontinuities and the sensitivity of the gradient descent methods to initial conditions, the overall path may not be optimal.

To overcome these problems, it is possible to integrate the structural optimization within the learning algorithm itself. This is done using a penalty parameter in the model performance criterion, also called *regularization term*.

$$\tilde{E} = E + \nu\Omega \tag{90}$$

Equation 90 represents the transformed model performance criterion. If E was the criterion, for example the MSE, the transformed criterion will be $\tilde{E}$. Ω is the function which have a large value for smooth mapping functions and small otherwise, i.e. small for simple models and large for complex models. The parameter ν controls the influence of Ω.

In the weight decay method, the regularization criterion will be given by Equation 91.

$$\Omega = \frac{1}{2}\sum_i {w_i}^2 \tag{91}$$

The new defined criterion $\tilde{E}$ is used in a gradient descent learning algorithm. We will not detail the technique here but we can say that at each iteration, the absolute values of the weights are decreased. In the weight update process, only efficient weights have their absolute values raised (this is particularly the case when starting with small initial conditions). Then, as the number of iteration increases, the non-efficient weights have their absolute values converging to zero, corresponding to an elimination, while the others are constantly updated. At the end of the process, we will have a well regularized network.

Although this methods has many appealing aspects, the main problem of defining the decay parameters can lead to very poor results. If they are set too high, all the weights will converge to zero, the derivatives will become incoherent and no good results will be obtained. if they are set too low, the regularization will not be performed subsequently before reaching overtraining. The consequence is that either the network will take ages to converge if it can converge. Moreover, they are no ways of determination in advance estimates of these regularization parameters. The adaptive methods used to control these parameters, just like the adaptive backpropagation parameters can be very problem dependent. For all these reasons, a simpler regularization method based on off-line statistically driven method was used. Although the latter method can be seen as rough, the statistical decision making process associated with the generalization estimates and the network committees contributed to perform relatively well in the set of application studied so far.

In the Bayesian approach, there are techniques to estimates these regularization parameters. For that reason, the weight decay method may be particularly suitable in Bayesian learning strategy.

3 Network Committees.

3.1 Introduction

Whatever the learning algorithm chosen, the result will always be conditioned, either to the structural choices or the natural dependencies, i.e. correlations created by the limited size of the data or even by the nature of the data (measurements obtained from experiments). Therefore, it is possible to use another stage in the remaining process to improve its performance. With the use of the statistical methods expressed already, it is possible to use a direct consequence of the distribution approach.

We have seen that the limited data set, the use of supervised (gradient descent) learning algorithms and the use of random initial conditions lead to local minima m_i on the error surface. For example, the result of the training on different training sets and different initial conditions will give different conditions. With the use of the techniques that tend to decrease the Although and temporal dependencies (bootstrap, uniformly normally distributed small initial weights,...) we can assume that these local minima will be normally distributed around the global minima.

Assuming that the local minima obtained from the gradient descent technique with *randomized training conditions*, statistical theory results tell us that the hypothetical global minimum, i.e. the one that provide good generalization properties, should be at the *center* of the local minima population. That consequence is used in the Bayesian approach for neural network remaining and the remaining technique is called *Model Averaging* [10, 13].

The model averaging can lead to two different solutions : Either the model are averaged through their parameters or the predictions can be averaged. The first method can be implemented only if the initial weights are identical so that the variations of a given parameter in the structure would be due to differences in the training set rather that initial weights. Although this give good generalization properties, experience showed that it was obtained at the expense of the prediction accuracy. Moreover, since the initial condition must be identical, the conditional probabilities are increased. If this method can be considered in some application, it is not a suitable method for multi-step ahead prediction remaining where the accuracy of the prediction is important to ensure the stability. For the sunspot series remaining, we preferred to apply the predicting averaging or combination method.

In the Bayesian approach of prediction averaging, the predictions of the models are averaged using Bayes factors as weights. Bayesian approach averages over models even those which do not fit well but advocate to exclude all but models with good fits. This is a linear combination of so-called model committees. As we will see later, other combinations, and especially nonlinear, will be used.

The neural network committees, a subgroup in the hybrid or multi-neural network systems [13], are used to benefit form the statistical theory results. The basic idea is that if a population of network is follows a bootstrap distribution, i.e. the parameter hyperspace is centered on the optimal parameter solution, then the combination of the model will give better model. The same reasoning is true for the predictions. An adequate combination of the prediction of uniformly distributed models will give better predictions. Moreover, by providing statistics, the network committees enable to obtain confidence limits on the predictions which is almost as important as the predictions themselves.

One of the drawback of working with committees is that the performance is obtained through increased computation. This issue can easily be summarize since most of the processes can be computed in parallel. analyzing, we will see that the very fact of using committees enable to reduced complexity, both in structural and parameter number terms which will induce a great deal of computation savings.

3.2 Linear Combination

3.2.1 Notations

Let define N_k to be the k^{th} network in the network committee and y_k its output. For remaining purposes, we can assume that the network output is the target plus an error.

$$y_k(x) = h(x) + \varepsilon_k(x) \tag{92}$$

where

- $y_k(x)$ is the mapping achieved by network k.

- $h(x)$ is the desired mapping.
- $\varepsilon_k(x)$ is the error.

The criterion used to evaluate the performance of the network over the input space x is the sum square error $\bar{E}_k$.

$$\bar{E}_k = \varepsilon\{[y_k(x) - h(x)]^2\} \tag{93}$$

Equation 93 can also be expressed using the probabilistic approach as expressed in Equation 94.

$$\bar{E}_k = \int_X {\varepsilon_k}^2(x)\, \mathcal{P}(x)\, dx \tag{94}$$

Having defined $\bar{E}_k$, the averaged sumsquare error of network k, we can evaluate the averaged error over the committee of network.

$$E_{av} = \frac{1}{C}\sum_{k=1}^{C} \bar{E}_k \tag{95}$$

Where C is the number of networks in the committee. Similarly, we can define the averaged variance of the network committee, σ_{av}.

$${\sigma_{av}}^2 = \frac{1}{C}\sum_{k=1}^{C} (\bar{E}_k - E_{av})^2 \tag{96}$$

3.2.2 Simple Combination

The simplest way to combine the committee predictions is to infer the averaged of the predictions. We define y_{comb} to be the *combined prediction*.

$$y_{comb} = \frac{1}{C}\sum_{k=1}^{C} y_k(x) \tag{97}$$

Therefore, the averaged sumsquare error of the combined prediction is given by Equation 98

$$\bar{E}_{comb} = \varepsilon([\frac{1}{C}\sum_{k=1} C y_k(x) - h(x)]^2) \tag{98}$$

using the definition of ε_k in Equation 92, we have

$$[\frac{1}{C}\sum_{k=1}^{C} y_k(x) - h(x)]^2 = [\frac{1}{C}\sum_{k=1}^{C} h(x) + \frac{1}{C}\sum_{k=1}^{C} \varepsilon_k(x) - h(x)]^2 \tag{99}$$

Since h(x) does not depend on index k, Equation 99 can be simplified by Equation 100

$$[\frac{1}{C}\sum_{k=1}^{C} y_k(x) - h(x)]^2 = [\frac{1}{C}\sum_{k=1}^{C} \varepsilon_k(x)]^2 \tag{100}$$

Therefore, Equation 98 can be expressed by Equation 101

$$\bar{E}_{comb} = \varepsilon([\frac{1}{C}\sum_{k=1}^{C} \varepsilon_k(x)]^2) \tag{101}$$

According to Cauchy's inequality, we have

$$\forall \text{function f}, (\sum_{k=1}^{C} f_i(x))^2 \leq C \sum_{k=1}^{C} f_i^2(x) \tag{102}$$

Therefore, we prove that the combined prediction enable better performance than individual networks on averaged.

$$\bar{E}_{comb} \leq \bar{E}_{av} \tag{103}$$

Using the bootstrap methods in conjunction with space normalization and normally randomized initial conditions will enable us to consider that the error $\varepsilon_k(x)$ have zero mean and are uncorrelated. Therefore, we have

$$\varepsilon(\varepsilon_k(x)) = 0 \quad \text{and} \quad \forall i \neq j,\ \varepsilon(\varepsilon_i(x)\,\varepsilon_j(x)) = 0 \tag{104}$$

Under these assumptions, $\bar{E}_{comb}$ can be expressed explicitly.

$$\bar{E}_{comb} = \frac{1}{C^2}\sum_{k=1}^{C}\varepsilon(\varepsilon_k(x)) \tag{105}$$

According to Equation 105 and 95 we have

$$\bar{E}_{comb} = \frac{1}{C}\bar{E}_{av} \tag{106}$$

Equation 106 shows how dramatic can be the improvement due to averaging other network committees. In practice, the simplification hypothesis may not be easy to obtain. Still, we can obtain an upper bound for the improvement.

$$\bar{E}_{av} \leq \bar{E}_{comb} \leq \frac{1}{C}\bar{E}_{av} \tag{107}$$

3.2.3 Weighted Combination

In section 3.2.2, we have seen that the use of an averaged prediction can give dramatic improvements. However, the combination was performed using a simple average. It makes a lot of sense to try to improve the way the combination is performed since all the networks may not be all at equal distance from the hypothetical global minimum. As a matter of fact, in an averaged combination, we are taking advantage of the uniform distribution of the local minima around the global minima but we do not consider the *quality* of these minima. The *quality* of the minima can be defined as inversely proportional to the distance between the local and the global minima. Unfortunately, since this global minimum is unknown, it is not possible to evaluate that distance.

An estimation of the global minima could be obtain by defining it as the center of gravity of all the weight parameters. This method can be used to regularized the network, to improve the generalization property. This is another way to take advantage of the network committee. However, to be of any use, this require to use large number of networks and more importantly, the same initial conditions so that functional similarities can be conserved when the averaging is performed. That approach was tested but is was shown to have little performance compared to the averaged prediction and lead to more penalizing computation time. Although this can be considered as an approach for generalization optimization when dealing with overtraining networks, combined prediction were preferred.

A first attempt to improve the generalization performance is to use a weighted sum of the network committee outputs. The identification of the parameters of this linear combination is straightforward. The output signals of the networks in response to a new input space, i.e. an input space not used during the training of the committee, is calculated and a gradient method is used to determine the parameters of the linear combiner. The combined prediction is then given by Equation 108

$$y_{comb} = \sum_{k=1}^{C}\alpha_k\, y_k(x) \tag{108}$$

The α_k, the weights of the linear combiner, can be seen as probabilities.

$$\mathcal{P}(y_k(x) = h(x)) = \alpha_k \tag{109}$$

Where α_k is the probability that $y_k(x)$ is the desired output (optimal) h(x). To correspond to a probability, a constraint must be applied to α_k.

$$\forall k, \alpha_k \in [0,1], \text{ and } \sum_{k=1}^{C} \alpha_k = 1 \tag{110}$$

To determine the combiner parameters, a gradient method can be used but is also possible to use a more sophisticated method.

By definition, the error autocorrelation matrix C^{ε} is

$$C^{\varepsilon}_{i,j} = \varepsilon(\varepsilon_i(x)\, \varepsilon_j(x)) \tag{111}$$

In Equation 105, we obtained an expression of E_{comb}. Using the ideas, we can show Equation 112.

$$\bar{E_{comb}} = \varepsilon((\sum_{k=1}^{C} \alpha_k\, \varepsilon_k(x))^2) \tag{112}$$

Using the definition of C^{ε} in 111 and by decomposing 112, we can obtain Equation 113

$$\bar{E}_{comb} = \sum_{i=1}^{C}\sum_{j=1}^{C} \varepsilon_i \varepsilon_j C^{\varepsilon}_{i,j} \tag{113}$$

The parameters α_k will be defined as the set of parameters that minimize $\bar{E}_{comb}$

$$\exists k \in [1, C] \text{ so that } \frac{\partial \bar{E}_{comb}}{\partial \alpha_k} = 0 \tag{114}$$

Using the Lagrange multipliers method [7], is possible to obtain an expression for α_k.

$$\alpha_k = \frac{\sum_{j=1}^{C} \{C^{\varepsilon}\}^{-1}_{k,j}}{\sum_{i=1}^{C}\sum_{j=1}^{C} \{C^{\varepsilon}\}^{-1}_{i,j}} \tag{115}$$

Finally, we can obtain an expression for $\bar{E}_{comb}$.

$$\bar{E}_{comb} = \frac{1}{\sum_{i=1}^{C}\sum_{j=1}^{C} \{C^{\varepsilon}\}^{-1}_{i,j}} \tag{116}$$

In the Bayesian approach of the linear combination, the predictions of the models are averaged using Bayes factors as weights [**?**] (statistical ideas for selecting network architectures).

3.3 Nonlinear Combinations

3.3.1 Neural Networks as Combiners

In section 3.2 we have seen that it was possible to obtain better forecasting by combining the predictions. Because the combinations was linear, it was possible to obtain expression of the gain on the prediction mean error $\bar{E}_{comb}$. The same results are obtained on the prediction mean standard deviation $\bar{\sigma}_{comb}$ although the demonstration is more complex.

We have seen that the way the combination is performed by induce dramatic improvement. With weighted sum, it was possible to obtain an analytical prediction of the improvement. To go a step further into the optimization, we will have to use non linear combiners for which we will no longer have analytical results. Therefore, we will sum up the benefits obtained from the statistical combination and the performance of neural networks as nonlinear function approximators. We will obtain only estimates of $\bar{E}_{comb}$ and $\bar{\sigma}_{comb}$ but this will be balanced by the real improvements gained over linear combiners.

A rigorous methodology would be to perform series of comparative tests between linear and nonlinear combiners. However, since the purpose of this study is not to examine thoroughly all the possible ways,

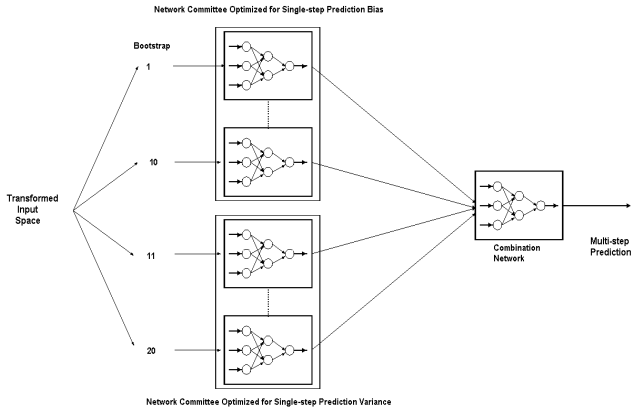

Figure 3: Representation of the Model Structure.

we will assume that the general comparative results between linear and neural networks model will apply also in the combination case.

Practically, we will use a neural network, designated as the *combiner*, to combine the predictions of the individual network predictions. The general methodology described to define and optimize the individual networks will be applied in the design of the combiner.

Figure 3 represents the network structure resulting from the use of neural network combiners. A committee of neural networks is identified using various training sets, initial conditions and even different structures. When the network committees are optimized, the neural network combiner is identified. The optimization of the committees can be of two kinds. In section 3.3.2, we will see that it is possible to take even more advantage of the committee of network by dividing the committee in subcommittees of networks on which different optimization processes are carried out.

3.3.2 Subcommittees

As a result of the analytical study of section 3.2, we have seen that the benefit from the combination will be higher is the committees outputs are independent. Moreover, to some extend, since $\bar{E}_{comb}$ (and incidentally $\bar{\sigma}_{comb}$ are proportional to $\bar{E}_{av}$ (respectively $\bar{\sigma}_{av}$), the better will be the committee prediction, the better will be the combined prediction. Therefore, what we want is a compromise between good individual performance (small mean and variance error) but we also want the committees prediction to be sprayed. The tradeoff is obtained by two processes.

The first will lead to perform multicriteria optimization. By selecting and optimizing subsets of the committee in regard to various performance criteria we will spread the prediction variance. More precisely, we will determine two subcommittees, one optimal for error variance and the other one optimal for the mean error. In that case, the combination process will enable us to address in a different way the bias-variance tradeoff already discussed in section (????).

Figure 4 represents a subcommittee of network obtain during the summarize of the sunspot series [30]. Within the network committee, a subset of networks was selected on the basis of either error mean or error variance. Of course, in that process, the same network was allowed to belong to the two subcommittees since a network with good generalization properties has both good error mean and variance. Networks that was not satisfying for both criteria were not retained in any subcommittees. Again, the decision boundary was defined using statistical inference with the aim of obtaining coherent network performances while still maintaining a level of variety.

Once selected, the subcommittees have been optimized further using the weight elimination process supervised by statistical decision making methods described in section 2.5.4. Figure **??** represents the subcommittee selected and optimized by mean of weight elimination using the error mean, or error bias criterion. The outcome of that process is a subcommittee of 18 networks with various topology. addressed, this variety of topology, which is an alternate answer to the optimal topology issue, is not obtained at

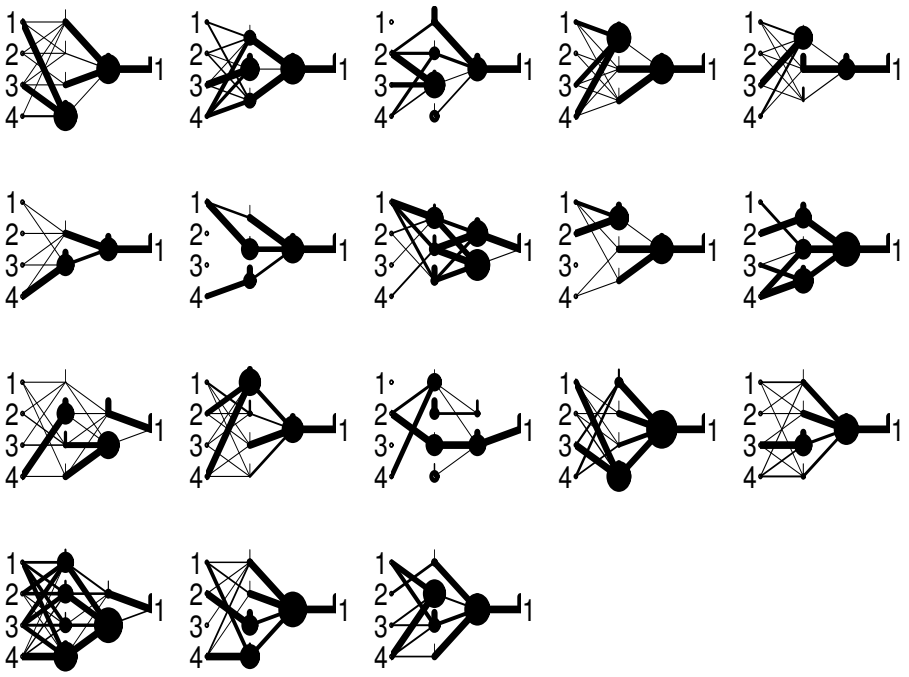

Figure 4: Subcommittee optimized for Error Bias.

the cost of poor performance.

Figure 5 represents the subcommittee error bias for single step ahead prediction on bootstraps of the sunspot series. The error bias displayed is in fact the averaged error bias obtained for five bootstraps. This use of bootstrap ensure a relative independence with respect to the data set and then improve the confidence limits on the error bias estimates. Although the performance before the optimization process is not displayed, it has been shown in **??** that the averaged error bias was greatly improved. Moreover, since the combined error bias is somehow proportional to the averaged error bias and that the variety of topology ensure the statistical variety, the use of subcommittees obtained with that methodology enables great improvements both in model prediction performance (bias and variance) but also improves the confidence limits.

By decreasing the dependencies, the use of combined subcommittees addresses the critical issue of generalization property of the model. Again, we have shown how the use of coexisting statistical methods can be used to reach that goal.

3.3.3 Multistep Ahead Prediction Criterion

Multistep ahead prediction is a very critical issue in the time series summarize field. The phenomenon of *error propagation* tend to make multistep ahead prediction models very difficult to design and optimize. To obtain a multistep ahead prediction neural network model, two main issues must be tackled.

The first one is the error propagation that tend to make the prediction error diverging after a few steps. This come from the fact that the predicted output, with the prediction error, is used in the input space (in the space of systems autoregressive order inferior to the horizon of prediction). the result is that the error in the input space produce an increased correlated prediction error which again is used in the input space for the next prediction. The consequence is that even for models that give relatively good single step ahead prediction, the model becomes unstable and give unreliable prediction as soon as multistep ahead prediction is attempted. Because of the correlated nature of the error, it is obviously important to obtain the best possible single step ahead predictor.

The issue of the error bias-variance trade off raises once again in that particular application. As a matter of fact, if criterion such as sumsquare error tend to give single step ahead prediction error a good

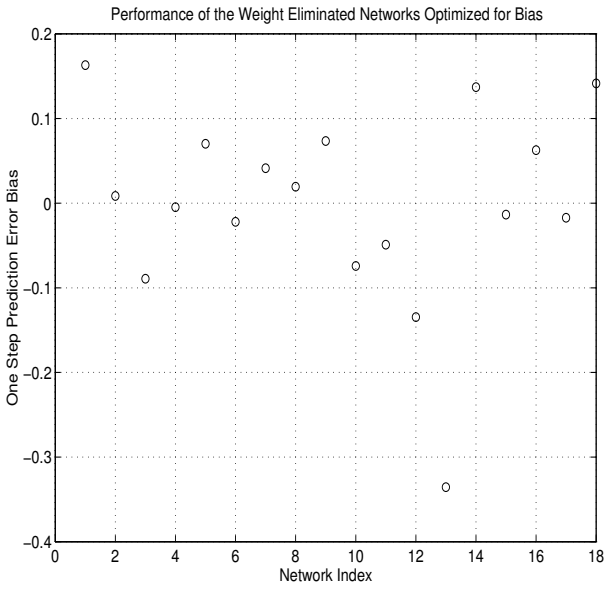

Figure 5:

balance between bias and variance it is no longer the case with multistep ahead prediction. Because of the correlated nature of the error, the bias has a more important role. The bias error tend to have a cumulative effect which create the cascading error. The error bias build up very quickly is the error variance is large. The variance have only an amplification effect with regard to the divergence. Therefore, when designing multistep ahead prediction models, the reduction of the error bias should be the may concern while keeping the error variance in acceptable limits.

Ideally, this trade off should be done during the parameter determination process. Unfortunately, because of the volatility of the mutlistep ahead predictors, multistep ahead prediction performance criterion are not reliable unless a sufficient performance is obtained. It would be very bad to perform any parameter optimization on diverging prediction error. These problems require first to determine the horizon of prediction for which the prediction will be reliable enough. Only then, will a multistep ahead criterion may be used to supervise the parameter optimization.

The determination of that *reliable horizon of prediction* is also submitted to limitations and dependencies. Once again, statistical methods are used to overcome the problems of performance estimation. First, for each neural network, the multistep prediction criterion is evaluated and averaged over a series of bootstrap, ensuring the data independence of the estimate, i.e. its generalization property. In a second stage, the criterion is averaged on the network committee to make the estimates independent from the training parameters and topological characteristics. Since the criterion is evaluated on network committee and on bootstrapped data sets, it is possible to obtain reliable statistics and especially confidence limits that will help in the decision making of the optimal horizon of prediction.

Another problem when attempting to train neural networks with multistep ahead criterion is the availability of such criterion. Although this issue will not be developed here, we can say transformed backpropagation algorithm or alternate neural network structures such a recurrent networks are the available approach to tackle that issue.

The methodology proposed here is completely different from the classical approach of the problem and take advantage of the committee statistical properties.

When the optimal horizon of prediction is determined, we process the multistep ahead prediction on another series of bootstrap samples from the two subcommittees optimized respectively for error bias and variance. Then we use a neural network combiner to obtain a combined multistep ahead prediction. Just like it improves the combined single step ahead prediction, the combiner produces better prediction since it is optimized using a multistep ahead criterion but also by addressing the bias-variance tradeoff issue.

Another advantage of this method is that it does not require specific training algorithms. The classical backpropagation as well as any other parameter optimization technique (Bayesian learning, genetic algorithms, ...)can be applied in this scheme.

At the end, using statistical methods for designing and optimizing the network committees, by defining a new structure and training methodology for the multistep ahead prediction model, we have improved the overall horizon of prediction and we have confidence limits of the predictions. This technique have been applied in the multistep ahead prediction of the sunspot series [30] (EE/JVR/97/3). More detailed explanations concerning the implementation of the techniques as well as its performance can be found in that application example.

4 Conclusions

We have presented a general tractable methodology for multistep ahead prediction neural network model that benefits from results of the statistical theory field. It is important to notice that despite the fact that we performed advanced analysis and optimization processes, we were still able to use the available tools and that the techniques never involved heavy computation loads. This is particularly true if we consider that most of the task of the summarize, especially the one involving multiple experience to obtain statistics, can be performed in parallel.

Future works should involve the exploration of the Bayesian techniques, which despite their application limitations are thought to have a lot of prospect to handle in a rigorous way the uncertainties. The similar conclusion can be made for the complexity estimation methods such as the VC dimension. Another step not discussed here would be the availability of advanced multicriteria optimization methods. As we already stated, gradient descent methods are too much estimates to be able to generate good generalization properties or stable optimization for multistep ahead prediction models. Following the same global ideas as the Bayesian learning with the Monte-Carlo based optimization methods, the Genetic Algorithms would be expected to give good results. This prospect is even reinforced by the use of the network committees an their *natural* diversity. The subcommittees would therefore be *species* that would be *evolved*. The overall combination would mean to integrate the two species in a symbiosis environment.

This summarize methodology have been applied with success to the multistep ahead prediction of the sunspot series [30] where further details on the application of the methods can be found. The proposed summarize methodology is presented on Figure 6.

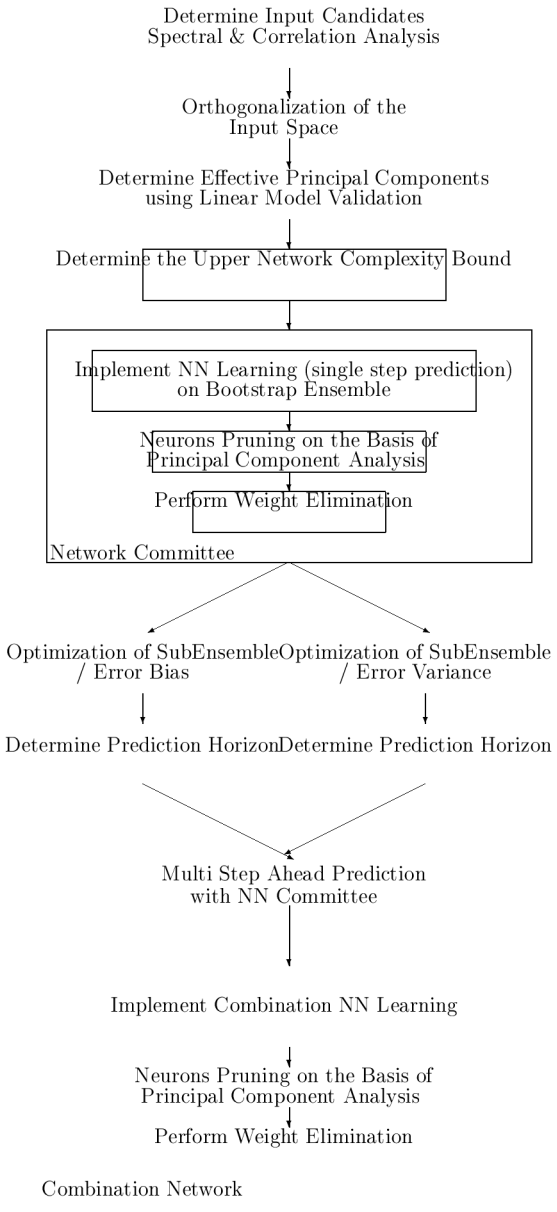

Figure 6: Representation of the advantage Flowchart.

5 Appendix

5.1 Appendix A : Statistics.

5.1.1 Statistical Inference.

The statistical distributions are commonly used in three problems 1. Evaluation of probabilities and expected frequencies for a distribution model.

2. Testing of Hypotheses about the variables being observed.

3. Evaluation of confidence limits for parameters of a fitted model, for example the mean of a normal distribution.

In hypothesis testing, a statistic (performance criterion) with known distribution under the null hypothesis (true if variable equals zero) is evaluated and the probability α of observing such a value or one more is found. This probability (significance) is usually then compared with a preassigned value (the significance level of the test) to decide whether the null hypothesis can be rejected in favor of an alternate hypothesis on the basis of the sample values.

The confidence interval problem requires the inverse calculation. Given probability α, the value x is to be found, such that the probability that a value not exceeding x is observed, is equal to α. A confidence interval, of size $1 - 2\,\alpha$, for the quantity of interest, can then be computed as a function of x and the sample values. Usually, these values are obtained from a table.

Let $x_1, x_2, \cdots, x_n$ be the observed sample of incidentally variables. We assume that they are normally distributed $\mathcal{N}(\mu, \sigma)$. μ and σ are the *true* mean and standard deviation of the variables. The two parameters are estimated on the basis of the observed samples, respectively $\bar{X}$ and $\hat{\sigma}$. Let Y_μ be a random variable that expresses how accurate is the mean estimates.

$$Y_\mu = \frac{\bar{X} - \mu}{\hat{\sigma}(X) \;/\; \sqrt{n}} \tag{117}$$

Results from the distribution theory states that Y is t(n-1) distributed. For μ the confidence interval is given by

$$\mu = \bar{X} \pm t_\alpha \, \frac{\hat{\sigma}\,(X)}{\sqrt{n}} \tag{118}$$

Where t_α is taken from the tabled t(n-1) distribution so that $P(-t_\alpha < \frac{\bar{X}-\mu}{\hat{\sigma}\,(X) \;/\; \sqrt{n}} < t_\alpha) = \alpha$ is the required confidence level. For σ, the confidence interval is given by Y_σ, the random variable that expressed the confidence of the standard deviation estimate.

$$Y_\sigma = \frac{(n-1)\,\hat{\sigma}^2\,(X)}{\sigma^2} \tag{119}$$

Y_σ has the χ^2 (n-1) distribution so that $P(-\chi_\alpha < \frac{(n-1)\,\hat{\sigma}^2\,(X)}{\sigma^2} < \chi_\alpha) = \alpha$.

These inference calculations involve normally distributed variables. The *central limit theorem* shows that the sample mean of any continuously distributed variable can be approximated by a normal distribution.

5.1.2 The Normal distribution

The normal distribution associated to a random variable X is expressed as follow :

$$X \hookrightarrow \mathcal{N}(\,\mu,\, \sigma^2\,) \tag{120}$$

Where

μ is the mean of the random variable X.

σ is the standard deviation of the random variable X.

σ^2 is the variance of the random variable X.

The density of probability function is defined by :

$$F(x) = \frac{1}{\sigma\sqrt{2\,\pi}}\, e^{\frac{-(x-\mu)^2}{2\,\sigma^2}} \tag{121}$$

Concerning the statistical inference calculations, when μ and σ are known, we have :

$$\mathcal{P}(\frac{\mu - u_\alpha\,\sigma}{\sqrt{n}} \prec \bar{X} \prec \frac{\mu + u_\alpha\,\sigma}{\sqrt{n}}) = \alpha \tag{122}$$

In the particular case of real values, the α is given by :

$$\alpha = \Phi_0(u_\alpha) = Erf(\frac{u_\alpha}{\sqrt{2}}) \tag{123}$$

$$\mathcal{P}(\mid X - \mu \mid \succ \delta) \preceq \alpha \tag{124}$$

With the delta notation, we can define the interval of confidence around the mean. The interval $[\mu - \delta\,\sigma, \mu + \delta\,\sigma]$ define a confidence interval at probability level $1 - \alpha$.

These equations are true only when using the real values of μ and σ. When estimates are used, equations are more complex.

5.1.3 The Student's t-distribution

difference between two means To clear things, we can examine an example.

Let $\bar{x_1}$ be the mean of the random variable x_1 based on n_1 observations and a standard deviation σ_1. Similarly, $\bar{x_2}$ is the mean of the random variable x_2 based on n_2 observations and standard deviation σ_2.

The standard error SE of $\bar{x_1} - \bar{x_2}$ is given by :

$$SE(\bar{x_1} - \bar{x_2}) = \sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}} \tag{125}$$

If σ_1 and σ_2 are known, the confidence limits with level of probability $2\,\alpha$.

$$\bar{x_1} - \bar{x_2} \pm u_\alpha\, SE(\bar{x_1} - \bar{x_2}) \tag{126}$$

If σ_1 and σ_2 are estimated, then we have

$$\bar{x_1} - \bar{x_2} \pm t_\alpha\, SE(\bar{x_1} - \bar{x_2}) \tag{127}$$

The degree of freedom Φ of the t-distribution is given by :

$$\frac{1}{\Phi} = \frac{1}{\Phi_1}\,(\frac{\hat{\sigma_1}^2/n_1}{\frac{\hat{\sigma_1}^2}{n_1} + \frac{\hat{\sigma_2}^2}{n_2}})^2 + \frac{1}{\Phi_2}(\frac{\hat{\sigma_2}^2/n_2}{\frac{\hat{\sigma_1}^2}{n_1} + \frac{\hat{\sigma_2}^2}{n_2}})^2 \tag{128}$$

With $\Phi_1 = n_1 - 1$ and $\Phi_2 = n_2 - 1$

The t-distribution will be used, among other things, in the determination of the model structure. The comparisons will be comparisons of the means of the model performance (MSE) penalized by a complexity factor (FPE or AIC).

It is interesting to notice that the NAG library provide functions that enable to calculate these figures. To obtain t_α knowing α and the degree of freedom or vice versa.

$t_\alpha = G01FBF(\imath S\imath, \alpha, \Phi)$

$\alpha = G01EBF(\imath S\imath, t_\alpha, \Phi)$

Degree of Freedom Φ	t_α
∞	1.96
20	2.09
10	2.23
5	2.57
2	4.3
1	12.71

Table 1: Table of t-distribution for a percentile $\alpha = 0.05$.

5.1.4 The χ^2-distribution

The χ^2-distribution is used to express the distribution of the sum square of random variables. If we define X_k to be a random variable, then χ^2 is defined by :

$$\chi^2 = X_1^2 + X_2^2 + \cdots + X_k^2 \tag{129}$$

Where k is the number of degrees of freedom. The mean of the χ^2-distribution is $\mu = k$ and its variance is $\sigma^2 = 2\,k$. The density of probability function of the χ^2-distribution is defined by :

$$f(\chi^2) = \frac{1}{2^{k/2}\,\Gamma(k/2)}\,(\chi^2)^{k/2-1}\,e^{\frac{-\chi^2}{2}} \tag{130}$$

Where Γ is the standard Γ function.

Concerning statistical inference problem, the α-percentile χ^2_α is defined by :

$$\mathcal{P}(\chi_2 < \chi^2_\alpha) \leq \alpha \tag{131}$$

A confidence interval at the probability level α around the mean for the χ^2-distribution is $[\chi^2_{\alpha/2}, \chi^2_{1-\alpha/2}]$. For degrees of confidence $k \prec 30$, the percentile can be approximated as $\chi^2_\alpha = 0.5\,(z_\alpha + \sqrt{2k-1})^2$, where z_α is the corresponding percentile of the standard normal distribution.

In model selection, we will mostly deal with one degree of freedom since we will compare the model having one more parameter. We can then express the table of the percentile for one degree of freedom (k=1).

$\chi^2_{0.005}$	$\chi^2_{0.01}$	$\chi^2_{0.025}$	$\chi^2_{0.5}$	$\chi^2_{0.95}$	$\chi^2_{0.975}$	$\chi^2_{0.99}$	$\chi^2_{0.995}$
0.00	0.00	0.001	0.004	3.84	5.02	6.63	7.88

Table 2: Table of the Percentiles χ^2_α of the χ^2-distribution for a degree of freedom k=1.

5.1.5 The F-distribution

The F-distribution is defined as the distribution of the ratio of two incidentally χ^2-variables.

$$F = F(n_1, n_2) = \frac{\chi^2_{n_1}/n_1}{\chi^2_{n_2}/n_2} \tag{132}$$

The probability density function is given by :

$$f(F) = \frac{\Gamma(\frac{1}{2}\,(n_1+n_2))}{\Gamma(\frac{1}{2}\,n_1)\,\Gamma(\frac{1}{2}\,n_2)}\,(\frac{n_1}{n_2})^{\frac{1}{2}\,n_1}\,\frac{F^{\frac{1}{2}n_1-1}}{((\frac{n_1}{n_2})\,F+1)^{\frac{1}{2}(n_1+n_2)}} \tag{133}$$

The mean of the F-distribution is

$$\forall n_2 \succ 2,\, \mathcal{E}\{F\} = \frac{n_2}{n_2-2} \tag{134}$$

The α-percentile F_α is expressed by :

$$\alpha = \mathcal{P}\{F \prec F_\alpha\} = 1 - \mathcal{P}\{F \succ F_alpha\} = 1 - \mathcal{P}\{\frac{1}{F} \succ \frac{1}{F_\alpha}\} \tag{135}$$

5.1.6 Central Limit Theorem

If $\bar{X}$ is the mean of the random variable X of size n taken from a population with mean μ and variance σ^2, then the limiting form of the distribution of $Z = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}}$ as $n \to \infty$ is the standard normal distribution : $\mathcal{N}(Z, 0, 1)$.

5.1.7 Statistical Hypothesis Testing.

Statistical hypothesis testing enable to evaluate quantitatively a statement on the basis of parameters such as mean or variance. Among these parameters, the analysis of variance, in which the hypothesis is evaluated using the variance.

In other words, the hypothesis testing can be seen as a decision procedure based on statistical analysis. A classical example is the comparison of the performances of two processes on the basis of the yield parameters. Several experiments are carried out and the mean of the yields are used as decision factor. The hypothesis testing factor is then the variable equal to the difference of the mean of the yield. The sign of that testing factor is supposed to determined the best process. The problem is that an analysis based on the sign would lead to great approximations since experiments carried out on the same process give different yields. To overcome that hazard, what is considered is not only the mean values but also the variance. Using statistical inference results, it is possible to determine interval of confidence and their associated percentiles. The decision will be based on the sign and the absolute value of the difference between the two mean. If the difference between the two mean is superior to the sum of their confidence intervals (say 95%) then the interpretation of the sign of the hypothesis testing factor is valid.

In the model selection field, very often the only selection criterion is a model response error, typically the MSE. It may also be the error variance. Comparing two models is then comparing their respective model response error criterion, i.e. the error variance. The problem can be stated as follows :

$$\begin{aligned} H_1 &: \sigma_1^2 = \sigma_2^2 \\ H_2 &: \sigma_1^2 \neq \sigma_2^2 \end{aligned} \tag{136}$$

Where σ_1^2 and σ_2^2 are the variances of the two model response errors. The *Hypothesis* 1 is that the two variance are equal is the null hypothesis whereas *Hypothesis* 2 is the alternative hypothesis. Of course, here we deal with equality relations but inequality relations would lead to the same conclusions. From the model selection point of view, the idea is to compare two MSE. The smaller will be assumed to be related to the best model but we have to verify that the observed difference is *real*, i.e. statistically significant.

While the hypothesis are based on the real values, we define a test statistic for testing these hypothesis to be the ratio between the two observed variances.

$$F = \frac{s_1^2}{s_2^2} \tag{137}$$

Where s_1 and s_2 are the standard deviations observed on samples of size n_1 and n_2 respectively.

If the hypothesis 1 is true, then the statistic F is $F(n_1 - 1, n_2 - 1)$-distributed. To be accepted, the statistical test is the *two-sided F-test* :

$$F_{\alpha/2}(n_1 - 1, n_2 - 1) \preceq F \preceq F_{1-\alpha/2}(n_1 - 1, n_2 - 1) \tag{138}$$

In the equality case we have :

$$\begin{aligned} H_1 &: \sigma_1^2 = \sigma_2^2 \\ H_2 &: \sigma_1^2 \succ \sigma_2^2 \end{aligned} \tag{139}$$

The associated *one-sided F-test* is then expressed by :

$$F \preceq F_\alpha(n_1 - 1, n_2 - 1) \tag{140}$$

References

[1] K. Pearson *On Lines and Planes to the Closest Fit to Systems of Points in Space*, Philosophical Magazine 2, p559-572, 1901.

[2] A.C. Rencher, *Methods of Multivariate Analysis*, Chapter 12 : Principal Components Analysis, Wiley Series in Probability and Mathematical Statistics, 1995.

[3] J.D. Jobson, *Applied Multivariate Data Analysis, Volume II : Categorical and Multivariate Methods*, Chapter 9 : Principal Components, Factors and Correspondance Analysis, Springer-Verlag. Springer Texts in Statistics, 1992.

[4] P.P. Kanjilal, D.N. Banerjee, *On the Application of Orthogonal Transformation for the Design and Analysis of Feedforward Networks*, IEEE Transactions on Neural Networks, Vol. 6, No. 5, p1061-1070, Sept. 1995.

[5] G.E. Box, G.M. Jenkins, *Time Series Analysis : Forecasting and Control*, Holden-Day, San Francisco 1970.

[6] S. Haykin, *Neural Networks : A Comprehensive Foundation.*, Prentice Hall, 1994.

[7] R.M. Hristev, *Artificial Neural Networks*, ftp://ftp.funet.fi/pub/sci/neural/books/

[8] J. Sjöberg, *Non-Linear System Identification with Neural Networks*, Ph.D. Thesis No 831, Division of Automatic Control, Department of Electrical Engineering, Linköping University, Sweden, 1995.

[9] K.M. bossley, *Neurofuzzy Modelling Approaches in System Identification*, Ph.D. Thesis, Image, Speech and Intelligent Systems research group, Department of Electronics and Computer Science, University of Southampton, England, May 1997.

[10] S. Lawrence, C.L. Giles, A.C. Tsoi, *What size Neural Network gives Optimal Generalization ? Convergence Properties of Backpropagation*, Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, June 1996.

[11] W. Sarles, *Neural Networks Frequently Asked Questions*, ftp//rtf.mit.edu/pub/usenet/news.answers/ai-faq/neural-nets.

[12] Y. Le Cun, J.S. Denker, S.A. Solla, *Optimal Brain Damage*, Advances in Neural Information, Vol 2, pp 598-605, Proc. Systems L. Can & Touretski, 1989.

[13] R.J. Mammone, *When Networks Disagree : Ensemble Methods for Hybrid Neural Networks*, Chapt 10, pp126-142, *Artificial Neural Networks for Speech and Vision*, Chapman & Hall Neural Computing Series, 1994.

[14] C. Goutte, *Note on Free Lunches and Cross-Validation*, Neural Computation 9, pp1211-1215, 1997.

[15] V. Vapnik, *Estimation of Dependencies Based on Empirical Data*, Springer-Verlag, Berlin, 1982.

[16] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[17] P.L. Barlett, *Vapnik-Chervonenkis Dimension Bounds for Two and Three-Layer Networks*, Neural Computation 5, pp 371-373, 1993.

[18] A. Hole, *Vapnik-Chervonenkis Generalization Bounds for Real Valued Neural Networks*, Neural Computation 8, pp 1277-1299, 1996.

[19] M. Lehtokangas, J. Saarinen, P. Huuhtanen, K. Kaski, *Predictive Minimum Description Length Criterion for Time Series Modeling with Neural Networks*, Neural Computing 8, pp 583-593, 1996.

[20] S. Tamura, M. Tateishi, *Capabilities of a Four-Layered Feedforward Neural Network : Four Layers versus Three*, IEEE transactions on Neural Networks, Vol. 8, No 2, March 1997.

[21] J.S.U. Hjorth, *Computer Intensive Statistical Methods : Validation, Model Selection and Bootstrap*, Chapman & Hall, 1994.

[22] J. Shao, D. Tu, *The Jacknife and Bootstrap*, Springer Series in Statistics, 1995.

[23] L.K. Hansen, J. Larsen, F. Torben, *Early Stop Criterion From the Bootstrap Ensemble*, in Proceedings of IEEE Proceedings of ICASSP'97, vol. 4, pp. 3205-3208, Munich, Germany, April 1997.

[24] R. Johansson, *System Modeling & Identification*, Prentice Hall Information and System Sciences Series, 1993.

[25] L. Ljung, *System Identification : Theory for the User.*, Prentice Hall Information and System Sciences Series, 1987.

[26] NAG Foundation Toolbox for use with Matlab, Numerical Algorithms Group Ltd, 1995.

[27] D. Mackay, *Probable Networks and Plausible Predictions : A Review of Practical Bayesian Methods for Supervised Neural Networks*, to be specified.

[28] D. Mackay, *Bayesian Methods for Neural Networks : FAQ*, `ftp://wol.ra.phy.cam.ac.uk/pub/www/mackay/Bayes_FAQ.html`.

[29] D. Mackay, *Bayesian Methods for Adaptive Models*, Ph.D. Thesis, California Institute of Technology, Pasadena, California, 1992.

[30] G-M. Cloarec, *Multistep Ahead Prediction of the Sunspot Series using Neural Networks Committees*, Research Report EE/JVR/97/3, Control Systems Group, School of Electronic Engineering, Dublin City University, 1997.

[31] G-M. Cloarec, J. Ringwood, *Incorporation of Statistical Methods in Multi-step Neural Network Prediction Models*, Submitted to the IEEE International Joint Conference on Neural Networks, 1998.