Wegmann, Georg

Working Paper

Feature extraction with hybrid neural networks

Research Notes, No. 00-6

Provided in Cooperation with:
Deutsche Bank Research, Frankfurt am Main

Economics

Quantitative Analysis

Deutsche Bank Research

October 9, 2000

Research Notes in Economics & Statistics

Feature Extraction with Hybrid Neural Networks

Neural networks (NN) and fuzzy logic systems (FLS) are used successfully for financial forecasting, credit rating and portfolio management. In search for more sophisticated modeling techniques a mixture of NN and FLS has proved to be worth consideration. We propose the novel constructive approach by which a neuro fuzzy network is built up with the help of a constrained optimizer. The mathematical motivation for such hybrid networks is presented, using the Kolmogorov theory of metric entropy.

As an application of the proposed approach we build a neuro fuzzy network model which is able to explain the prices of call options written on the S&P 500 stock index. While option pricing theory typically requires a highly complex statistical model to capture the empirical pricing mechanism, our results indicate that this algorithm leads to more parsimonious functional specificationes which have a superior out-of-sample performance.

Georg Wegmann

Editor:
Christof Kreuter
+49 69 910-31748
christof.kreuter@db.com

Deutsche Bank Research
Frankfurt am Main
Germany
E-mail: marketing.dbr@db.com
Fax: +49 69 910-31877

DB Research Management
Axel Siedenberg
Norbert Walter

RESEARCH NOTE RN-00-6

Feature Extraction with Hybrid Neural Networks*

Georg Wegmann**

October 2000

Abstract

Neural networks (NN) and fuzzy logic systems (FLS) are used successfully for financial forecasting, credit rating and portfolio management. In search for more sophisticated modeling techniques a mixture of NN and FLS has proved to be worth consideration. We propose the novel constructive approach by which a neuro fuzzy network is built up with the help of a constrained optimizer. The mathematical motivation for such hybrid networks is presented, using the Kolmogorov theory of metric entropy.

As an application of the proposed approach we build a neuro fuzzy network model which is able to explain the prices of call options written on the S&P 500 stock index. While option pricing theory typically requires a highly complex statistical model to capture the empirical pricing mechanism, our results indicate that this algorithm leads to more parsimonious functional specificationes which have a superior out-of-sample performance.

Keywords: neural networks, fuzzy logic systems, entropy, option pricing

---

* Diploma thesis prepared with the support of Deutsche Bank Research.

** Department of Mathematics, University of Mannheim, wegmann@euklid.math.uni-mannheim.de.

Contents

1 Introduction

Science has evolved from trying to understand and predict the behavior of the universe and the systems within it. Most scientific work is based on finding models which agree with the observations and subsequently analyzing the implications. However, the extraction of a model from observed data using traditional linear methods leads to poor results if the underlying real world system is of an unknown and nonlinear form, which is the case most of the time. Such instances require more sophisticated modeling techniques and this is where modern function approximation methods based upon the parallelism paradigm of neural computation have gained prominence in recent years. A good illustration of this trend can be found in the field of finance where both, neural networks (NN) and fuzzy logic systems (FLS), are used for many complex nonlinear tasks such as financial forecasting, credit rating, portfolio management, event risk analysis and automated trading systems. However, one also finds in finance that the two aforementioned methods are generally considered distinct and hence treated seperately. While there are potentially empirical reasons to do so, since the intuitive content of the two systems varies significantly, this imposed seperation is unnecessary and can be downright inefficient from the standpoint of approximation. For a start both NNs and FLSs are asymptotically equivalent since both of the associated functional families are dense on compacta in many spaces of interest (e. g. the space of continuous functions, of square integrable functions, etc.). Moreover, NNs and FLSs can both be represented in single hidden layer feedforward network forms indicating a strong relationship of their functional families not only asymptotically but also on a small scale. Since in any empirical application of nonlinear modeling the most pertinent question is generally which functional form to use it follows that efficient approximation requires the careful consideration of the aforementioned relationship between neural networks and fuzzy logic systems. This issue of how the functional specification affects the small sample approximation precision is mostly ignored in the literature even though it is the key consideration for any nonlinear model building approach.

The individual problem of approximation efficiency for either a NN or an FLS can be sketched as follows: in any practical application the purpose of nonlinear modeling is to capture some underlying functional relationship between the independent variable(s) (inputs) and the dependent variable(s) (outputs) as precisely as possible given the available data. In general, the more data is available the more exact the result, because more data allows for more complex models with the same number of observation points per parameter. Increased model complexity in turn implies that the resulting function of the model is increasingly close to the underlying target function, if the approximator is asymptotically dense in that space. This relationship has even been shown to be strictly monotonic in many cases of interest.

The comparative problem of approximation efficiency of NNs and FLSs is given by the question, which of these methods — corresponding to different functional representations — to start with since there is no means to know a priori which function is closer to the unknown target function. All we know is that we can capture the target function with increasing precision if we increase the complexity of either individual model (NN or FLS). As an ignorant argument in this context one often receives the suggestion to try out both and choose the better fitting function ex-post. While this belies the often extensive computing efforts for either of these methods, the more critical aspect is the model or data snooping bias introduced through this strategy (see Campbell et. al. 1997).

On the other hand, while casual empiricism and statistical evidence in the literature suggest, that there can indeed be merit in the combination of nonlinear systems, there is neither a known approximation theoretic foundation for these claims nor an efficient strategy for the combination of said systems (NN and FLS).

While the arguments above are not limited solely to these two nonlinear approaches, in this paper we concentrate only on NNs and FLSs and derive a) the theoretical basis reasoning that underlies the aforementioned observed facts and b) an efficient incremental combination strategy for NNs and FLSs.

The remainder of the paper is organized as follows the 2nd Section, we provide a brief review of NNs i.e. their history, their construction and a sketch of a proof of the relevant universal approximation theorem. Common types of algorithms for optimizing values of parameters in a network are also mentioned. Section 3 provides an analogous overview of FLSs.

Section 4 contains the derivation of the proposed new neuro fuzzy network (NFN). Also, the algorithm which allows for an incrementally efficient construction procedure of such a network is given. In Section 5 we derive sufficient conditions under which the hybrid network (NFN) has the desired greater denseness in function space relative to either FLS or NN. Section 6 presents an application to finance, where an NFN is developed to estimate the daily market closing prices of SPX (Standard & Poors 500 Index) options, using transactions data for several years. Finally, Section 7 concludes and points to directions for further research.

2 Neural Networks

2.1 Introduction

NNs are a particular class of functions whose specification was originally inspired by the parallel processing of the human brain. Hence, the processing elements in NNs are called neurons, or hidden nodes. A human brain consists of approximately 10^{11} neurons of many different types [Lin,1995]. This number is close to the number of stars in the Milky Way, and the number of galaxies in the known universe. As many as 10^4 synaptic junctions may about a single neuron, that gives roughly 10^{15} synapses in the human brain. The brain hence represents an asynchronous, nonlinear, massively parallel system of cosmological proportions [Kosko,1992].

Each biological neuron receives input signals which is the information from other nodes or external points. This information is processed locally through an activation or transfer function and produces a transformed output signal to other nodes or external outputs. The receiving neuron *fires* if the electric potential reaches a *threshold,* and a pulse or *action potential* of fixed strength and duration is sent out through the synaptic junctions to other neurons.

A schematic diagram of a typical biological neuron is shown in Figure 2.1:

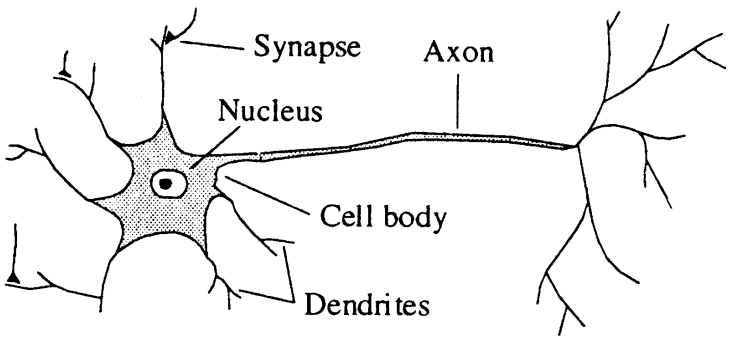


Fig. 2.1: Biological neuron

Figure 2.2 shows a simple mathematical model of the above mentioned biological neuron proposed by McCulloch and Pitts [1943]. In this model, the ith processing element computes a weighted sum of its inputs and outputs $y_i = 1$ (firing) or 0 (not firing) according to whether this weighted input sum is above or below a certain threshold θ_i :

$$y_i = \Psi(g(x)) = \Psi\left(\sum_{j=1}^{r} w_{ij}x_j - \theta_i\right), x = (x_1, \ldots, x_r) \in \Re^r$$

where the activation function $\Psi(\cdot)$ is a unit step function s.t.:

$$\Psi(g(x)) = \begin{cases} 1 \text{ if } g(x) \geq 0 \\ 0 \text{ if } g(x) < 0 \end{cases};$$

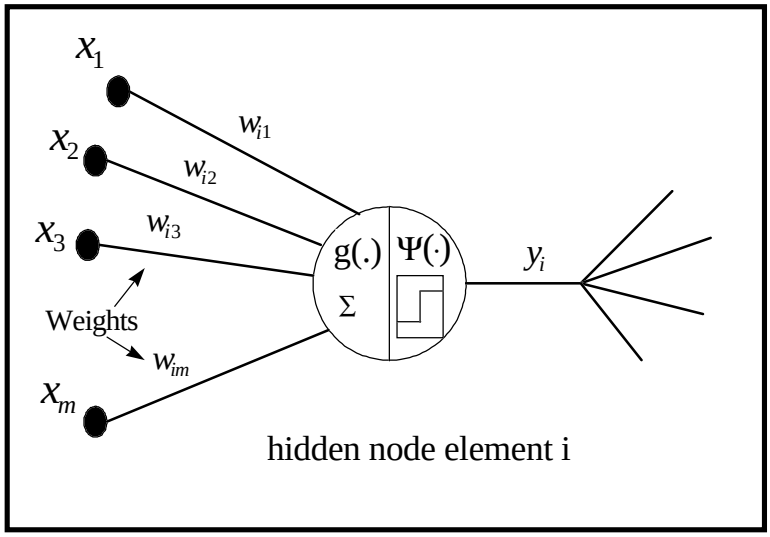


Fig. 2.2: Example for a proessing element

The weight w_{ij} represents the strength of the synapse connecting neuron j to neuron i. A positive weight corresponds to an excitatory synapse, and a negative weight corresponds to an inhibitory synapse. If $w_{ij} = 0$, then there is no connection between the two neurons.

As shown in Fig.2.2 the information processing of a processing element or hidden node can be viewed as consisting of two parts: input and output. Associated with the input of a node i is a function g_i which serves to combine information from an external source or other nodes. This is usually a linear function of the inputs x_j :

$$g_i = \sum_{j=1}^{m} w_{ij} x_j - \theta_i,$$

where θ_i is the threshold of the ith node. More complex functions g_i can also be considered as follows.

Quadratic function:

$$g_i = \sum_{j=1}^{m} w_{ij} x_j^2 - \theta_i,$$

Spherical function:

$$g_i = \rho^{-2} \sum_{j=1}^{m} (x_j - w_{ij})^2 - \theta_i,$$

where ρ and w_{ij} are the radius and the center of the sphere, respectively.

Nonlinear functions (g_i) of the input allow the formation of complex partitions of the feature space called *decision boundaries.* However, the more complex and powerful the functions g_i become, the more complex interactions are required in NNs.

A second action of each node is the output of an activation value as a function of its net inputs through an *activation function* or *transfer function* $\Psi(\cdot)$. The activation function of the node can also vary, but with less effect. What seems to be of most importance is the smoothness of the individual node. Some commonly used activation functions are as follows.

Gaussian function:

$$\Psi(g(x)) = \exp(-\frac{g(x)^2}{2}),$$

Tangents function:

$$\Psi(g(x)) = \tanh(g(x)),$$

Sigmoid function:

$$\Psi(g(x)) = \frac{1}{1 + \exp(-\lambda(g(x))}, \lambda > 0 \tag{2.1}$$

The shape of the above function (eq. 2,1) is shown in Fig. 2.3, where it can be observed that as $\lambda \to \infty$, the sigmoid function reduces to a step function.

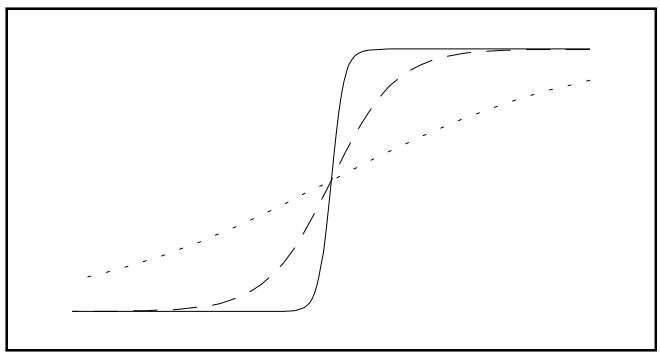
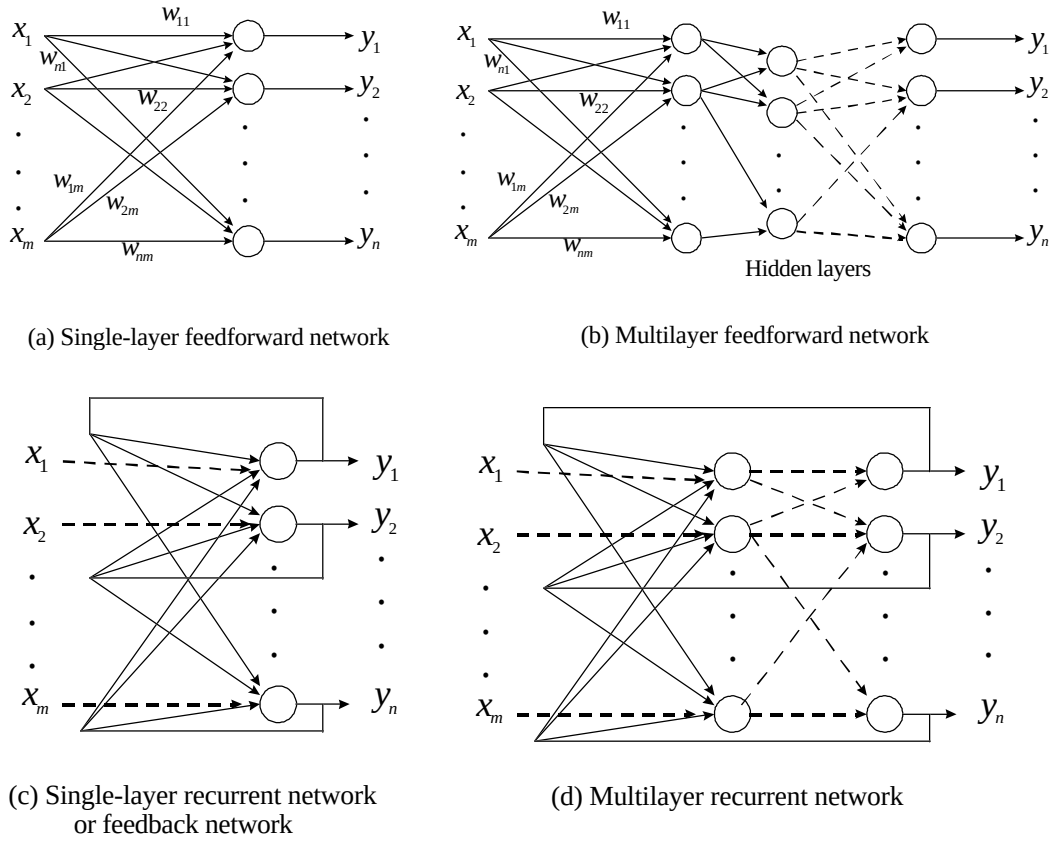


Fig. 2.3 : $\Psi(g(x))$ for $\lambda = 10, 2, 0.5$

An NN consists of a set of interconnected nodes or neurons such that each node's output is connected, through weights, to other nodes or to itself. The structure of these nodes and the connection geometry among them are the defining features of any NN.

Besides the simplest single-node neural network (Fig.2.2), four basic types of connection geometries exist (Fig.2.4).



(a) Single-layer feedforward network

(b) Multilayer feedforward network

(c) Single-layer recurrent network or feedback network

(d) Multilayer recurrent network

Figure 2.4

We can first take a node and combine it with other nodes to make a *layer* of nodes. Inputs can be connected to these nodes with various weights, resulting in a series of outputs, one per node. This results in a single hidden layer feedforward network as shown in Fig. 2.4(a). We can further interconnect several layers to form a multilayer feedforward network as shown in Fig.2.4(b). When outputs can be directed back as inputs to the same layer or to a preceding-layer of nodes, the network is a recurrent network (Fig. 2.4 c,d). More than one of the basic connection geometries summarized in Fig. 2.4 can be used together in an single NN.

However, for the remainder of this paper we will focus on a particular structure of NNs, namely the feedforward networks (see Fig. 2.4(a)), which are the

most prominent design in many applications including finance. Figure 2.5 below shows how feedforward neural networks are capable of spatial partitioning and of constructing complex continuous regions.

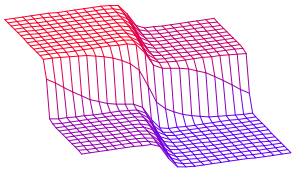


Figure 2.5.a

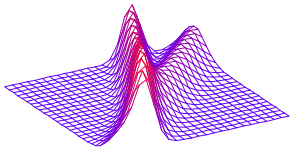


Figure 2.5.b

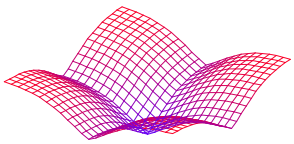


Figure 2.5.c

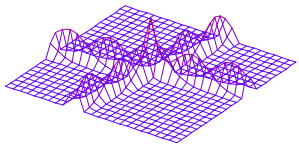


Figure 2.5.d

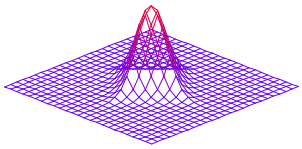


Figure 2.5.e

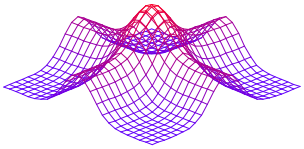


Figure 2.5.f

Note that all the surfaces in Fig.2.5 are constructed by a single-layer feedforward network with two neurons and two input variables, which indicates the scope of potential functional forms inherent in a single, simple, network specification. It is this richness of the single hidden layer feedforward networks, which we focus on in the next chapter.

2.2 Single Hidden Layer Feedforward Networks

The single hidden layer feedforward network is the most commonly applied neural network due to its simplicity and since it is dense in the space of continuous functions and measurable functions on compacta, it can be considered sufficient for most approximation tasks. Due to this eminence, we will review two key aspects, namely their construction and a sketch of the proof of their aforementioned denseness characteristic.

Since there is a bijective correspondence between any network diagram and the underlying mathematical function, we can analyze general network mappings by considering their respective network diagrams (as depicted in Fig. 2.4.(a-d)):

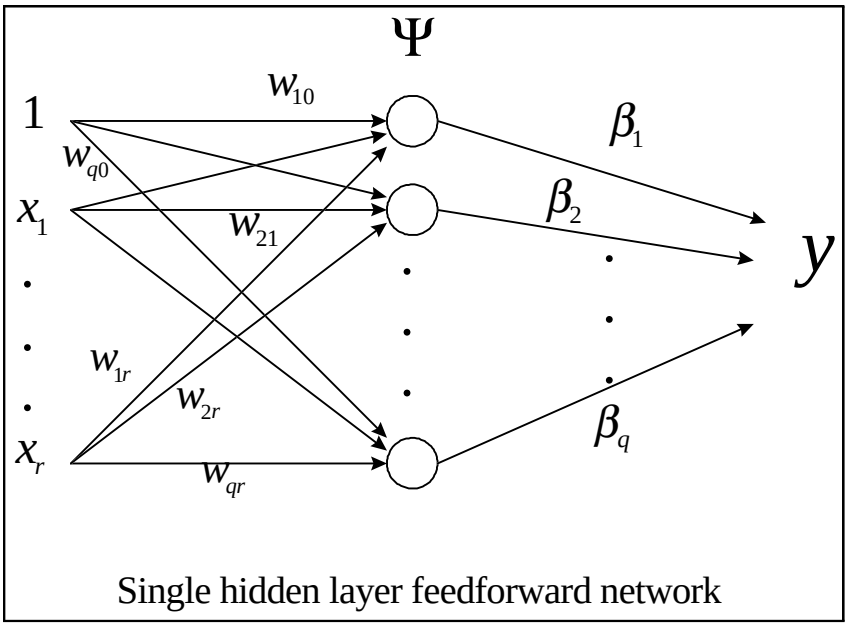


Figure 2.6: NN with q hidden nodes

The network diagram in Fig. 2.6 corresponds to a function of the explanatory (independent) variable vector $X = [1, x_1, \ldots, x_r]$ and the network weights $\delta = (w_{10}, w_{11}, \ldots, w_{qr}, \beta_1, \ldots, \beta_q)$, which are to be estimated in the learning phase. r is the number of input variables and q the number of hidden nodes or neurons. Therefore the NN in Fig. 2.6 has the following functional form:

$$y = f(X, \delta) = \sum_{j=1}^{q} \beta_j \Psi_j \left(\sum_{i=0}^{r} w_{ji} x_i \right),$$

with $\Psi_j(\cdot)$ as the activation function (see equation 2.1).

This functional form shows that an NN consists of a weighted average of shifted superpositions of the bounded nonlinear activation function $\Psi_j(\cdot)$. The

main difference of neural network functions relative to the popular polynomial approximation lies in the full support and asymptotic boundedness of the NN, features which are inherited from the well definedness and boundedness of its components.

These aspects of NNs are attractive, especially in the context of forecasting, where values outside the known model boundaries occur frequently. Additionally for some choices of the activation function, NNs possess all the approximation properties of Fourier series (White, 1992), including a well understood group of mathematical functions as true subsets.

2.3 Universal Approximation

We will now formally establish the denseness property of NNs in the space of continuous functions which we have frequently asserted in the previous paragraphs, since it is pivotal for any nonlinear modeling approach. Recalling figures 2.5, the intuitive richness of the available functional forms inherent in a given neural network family has to be sufficient to have for any given function of interest a neural network, that is approximately equal to the former. This condition is necessary for nonlinear models in the sense that without it one cannot ever put an upper bound on the approximation error. Note that — as emphasized in the introduction — the result is only asymptotic. The proof given here corresponds to the one derived originally by Hornik et al. (1989). Definitions and notation are as follows:

Definition 1 *For any $r \in \{1, 2, \ldots\}$, let A^r be the set of all affine functions from $\Re^r \to \Re$; i.e. $A_i(x) = w_i \cdot x + w_{i0}$ where w_i and x are vectors in $\Re^r$, "$\cdot$" denotes the dot product of vectors, and $w_{i0} \in \Re$ is a scalar.*

In the present context (see Fig. 2.6), x correspond to the network input, $w_i = (w_{i1}, w_{i2}, \ldots, w_{ir})$ and w_{i0} corresponds to network weights from the input to the intermediate layer, $i = (1, \ldots, q)$.

Definition 2 *For any continuous function $\Psi : \Re \to \Re$ and $r \in \{1, 2, \ldots\}$ let the NNs of interest $\sum(\Psi)$ be defined by:*

$$\left\{ f : \Re^r \to \Re \;\middle|\; f(x) = \sum_{j=1}^{q} \beta_j \Psi(A_j(x)), x \in \Re^r, \beta_j \in \Re, A_j \in A^r, q \in \{1, 2, \ldots\} \right\}.$$

Definition 3 *For any continuous function $\Psi : \Re \to \Re$ and $r \in \{1, 2, \ldots\}$ let the following extended set of NNs $\sum \Pi(\Psi)$ be defined by:*

$$\left\{ f : \Re^r \to \Re \;\middle|\; \begin{array}{c} f(x) = \sum_{j=1}^{q} \beta_j \prod_{k=1}^{l_j} \Psi(A_{jk}(x)) \\ = \sum_{j=1}^{q} \beta_j \prod_{k=1}^{l_j} \Psi(\sum_{i=1}^{r} w_{jki} x_i + w_{jk0})) \\ x \in \Re^r, \beta_j w_{jki}, w_{jk0} \in \Re, A_{jk} \in A^r; l_j, q \in \{1, 2, \ldots\} \end{array} \right\}.$$

The strategy of the proof is to establish the denseness result for the more complex $\sum\Pi(\Psi)$ networks and subsequently extend them to simpler $\sum(\Psi)$ networks, since the latter are a special case of $\sum\Pi(\Psi)$ networks ($l_j = 1$ for all j).

Definition 4 *Let C^r be the set of continuous functions from $\Re^r \to \Re$.*

The class C^rcontains many functions relevant in applications (The extension of these results to the class of measurable functions follows easily i.e. Hornik, Ripley, Wang etc.). Closeness of functions f and g belonging to C^r is measured by a metric, ρ. Closeness of one class of functions to another class is described by the concept of denseness.

Definition 5 *A subset S of C^r is said to be uniformly dense on compacta in C^r, if for every compact subset $K \subset \Re^r$, S is ρ_k-dense in C^r, where for $f, g \in C^r, \rho_k(f,g) = \sup_{x\in K} |f(x) - g(x)|$.*

Theorem 6 *Let Ψ be any continuous nonconstant function from $\Re \to \Re$. Then $\sum\Pi(\Psi)$ is uniformly dense on compacta in C^r.*

Proof. *We use the Stone-Weierstrass theorem. Let $K \subset \Re^r$ be any compact set. We need that $\sum\Pi(\Psi)$ is separating on K and $\sum\Pi(\Psi)$ vanishes at no point of K. For any Ψ, $\sum\Pi(\Psi)$ is an algebra on K. If $x, y \in K, x \neq y$, then there is an $A \in A^r$ such that $\Psi(A(x)) \neq \Psi(A(y))$. This ensures that $\sum\Pi(\Psi)$ is separating on K. Second, there are $\Psi(A(\cdot))'s$ that are constant and not equal to zero. To see this, choose $b \in \Re$ such that $\Psi(b) \neq 0$ and set $A(x) = 0 \cdot x + b$. For all $x \in K, \Psi(A(x)) = \Psi(b)$. this ensures that $\sum\Pi(\Psi)$ vanishes at no point of K. The Stone-Weierstrass theorem thus implies that $\sum\Pi(\Psi)$ is ρ_K-dense in C^r on K. Because K is arbitrary, the result follows.* ■

In other words $\sum\Pi(\Psi)$ networks can approximate any continuous function arbitrarily well. In the next step we will show that the simpler $\sum(\Psi)$ networks, despite their less complex structure are sufficient for this task.

Lemma 7 *$\sum(\Psi)$ is uniformly dense on compacta in C^r.*

Proof. *By Theorem 6 the trigonometric polynomials*

$$h(\cdot) = \left\{ \sum_{j=1}^{Q} \beta_i \prod_{k=1}^{l_j} \cos(A_{jk}(\cdot)) : Q, l_j \in \{1, 2, \ldots\}, \beta_j \in \Re, A_{jk} \in A^r \right\} \qquad (1)$$

are uniformly dense on compacta in C^r. Applying the trigonometric identity

(i.e $\cos(a)\cos(b) = \cos(a+b) - \cos(a-b)$*) equation (1) becomes to*

$$h(\cdot) = \left\{ \sum_{j=1}^{T} \alpha_t \cos(A_t(\cdot)) : T \in \{1, 2, \ldots\}, \alpha_t \in \Re, A_t \in A^r \right\}$$

and for any $h(x) = \sum_{j=1}^{T} \alpha_t \cos(A_t(x))$ *and* $x \in K$, $K \subset \Re^r$ *arbitrary compact set and arbitrary* $\varepsilon > 0$ *there exist an* $f \in \sum(\Psi)$ *such that*

$\sup_{x \in K} |h(x) - f(x)| < \varepsilon$ *(see White (1992, Lemma A.4, p. 24). The result follows.* ■

The last lemma proves, that standard feedforward networks with only a single hidden layer can approximate any continuous function uniformly on any compact set. An extension of the above lemma to any Borel-measurable functions, regardless of the dimension of the input space r, is also given in (Horink et al., 1989). This establishes $\sum(\Psi)$ networks as a class of "universal approximators", which implies that any lack of success in application must arise from inadequate learning (see section 2.6), insufficient numbers of hidden nodes, or lack of a deterministic relationship between input and desired output.

2.4 Neural Networks in Statistics

So far we have just considered deterministic nonlinear relationships, which can be approximated by NNs, however, their domain of application also overlaps significantly with statistics. Let Y_t, X_t with $t \in \{1, 2, \ldots\}$ be sequences of identically distributed random variables. Suppose we are interested in the functional relationship between Y_t and X_t. For example in classification or pattern recognition problems, Y_t is a binary or multinomial variable creating class membership and X_t is a set of variables influencing the classification. In forecasting problems, Y_t is the set of variables (or one variable) that we would like to forecast on the help of variables X_t, which may itself have past values of Y_t.

Regardless of whether a deterministic or stochastic relationship exists between Y_t and X_t, the mathematical object of interest in such situations is the conditional expectation of Y_t given X_t, written $E(Y_t|X_t)$. This can be written as a regression function,

$$\theta(X_t) = E(Y_t|X_t).$$

When Y_t can assume a continuum of values, $\theta(x)$ gives the expected value for Y_t given that $X_t = x$. We may also write

$$Y_t = \theta(X_t) + \varepsilon_t,$$

where $\varepsilon_t \equiv Y_t - E(Y_t|X_t)$ is a random error with conditional expectation zero given X_t. When the relationship between Y_t and X_t is deterministic, ε_t is always

zero; else, ε_t is nonzero with positive probability. The regression function θ is assumed to be unknown. Hence problem is to learn (estimate, approximate) the mapping θ from a realization of the sequences Y_t, X_t.

In practice, we observe a realization of only a finite part of the sequences Y_t, X_t, a "training set" or "sample". Since θ is an element of the space of all functions (Θ) it is not possible of learning θ in any complete sense from a sample of finite size. But it is possible to approximate θ to some degree of accuracy using a sample of size T, and to construct increasingly accurate approximations with increasing T. We will refer to such a procedure interchangeably as learning, estimation or approximation. We estimate θ by a network of the form:

$$f(X_t, \delta) = \sum_{j=1}^{q} \beta_j \Psi_j \left(\sum_{i=0}^{r} w_{ji} x_{ti} \right), \tag{2.2}$$

where δ is the parameter vector, $X_t = (1, x_{t1}, \ldots, x_{tr}) \in \Re^{r+1}$ at time t.

The simplest form of error function for regression problems is the method based on the minimization of a sum-of-squares error function. To get an estimation $f(X_t, \delta)$ of θ we have to minimize the quadratic distance between Y_t and the values of $f(X_t, \delta)$ by manipulating the parameter vector δ.

The sum-of-squares error function (SSE) is given by a sum over all patterns in the training set, and over all outputs, of the form:

$$SSE(\delta) = \sum_{t=1}^{T} (Y_t - f(X_t, \delta))^2 \rightarrow \min! \tag{2.3}$$

where $Y = (Y_1, \ldots, Y_T) \in \Re^T$ and $X = (X_1, \ldots, X_i, \ldots, X_T)$ with $X_i = (1, x_{i1}, \ldots, x_{ir}) \in \Re^{r+1}$. T is the sample size.

If we take the mean of the squared errors we get a measure for the regression:

$$MSE = \frac{1}{T} SSE$$

The mean squared error (MSE) of each model $f(X_t, \delta)$ can be divided in two parts, the systematic error (MSE_s) and the non-systematic error (MSE_n) :

$$\begin{aligned} MSE &= E[(Y - f(X, \delta))^2] \\ &= E[(Y - \theta(X) + \theta(X) - f(X, \delta))^2] \\ &= E[(Y - \theta(X))^2] + E[(\theta(X) - f(X, \delta))^2] + \\ &\quad 2E[(Y - \theta(X))(\theta(X) - f(X, \delta)] \\ &= E[(Y - \theta(X))^2] + E[(\theta(X) - f(X, \delta))^2] \\ &= MSE_n + MSE_s, \end{aligned}$$

The last but one row follows from the iterated expectation (i.e. $E[Y] = E_X[E[Y|X]]$) and given that $E[\varepsilon|X] = 0$:

$$\begin{aligned} E[(Y - \theta(X))(\theta(X) - f(X,\delta)] &= E[(\varepsilon)(\theta(X) - f(X,\delta)] \\ &= E_X[E[(\varepsilon)(\theta(X) - f(X,\delta)|X]] \\ &= E_X[E[\varepsilon|X](\theta(X) - f(X,\delta)] \\ &= 0 \end{aligned}$$

The non-systematic error MSE_n is random and no model can approximate this error since it comes from $\varepsilon = Y - \theta(X)$ and $\theta(X)$ is unknown. The systematic error MSE_s is the only error we can use to measure the regression. If we consider each point of the function $f(\cdot)$ as an estimator for the correspondent point of the function $\theta(\cdot)$, we can divide MSE_s, too. The MSE of each estimator can be divided into the bias and the variance of the estimator.

Let f_t be any point of $f(\cdot)$ and θ_t be the correspondent point of $\theta(\cdot)$, then

$$\begin{aligned} MSE_s &= MSE[f_t] = E[(\theta_t - f_t)^2] \\ &= Bias[f_t]^2 + Var[f_t] \end{aligned}$$

with $Bias[f_t] = E[f_t] - \theta_t., Var[f_t] = E[(f_t - E[f_t])^2]$.

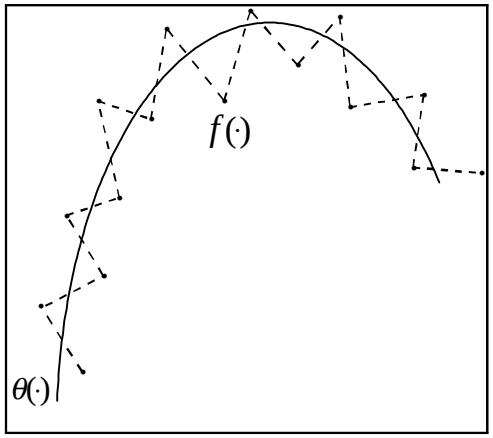
Bias specify the distance from the expected mean of f_t and θ_t. Thus the bias is a systematic distortion of the approximation.

From the variance of f_t we can calculate the area around $E[f_t]$ where the estimated function $f(\cdot)$ can also be. Hence the deviation of any approximation $f(\cdot)$ of $\theta(\cdot)$ is caused by the distortion of the expectation, by the variance of the function or by a combination of bias and variance.

A small model $f(\cdot)$ with less estimable parameter is very inflexible and thus it can not fit the error terms. It is also unable to approximate $\theta(\cdot)$. What we get is a big bias with low variance.

A large model $f(\cdot)$ can fit every observed data point, thus the bias of the estimation vanishes but the variance is very large.

The following pictures illustrate these extreme cases.



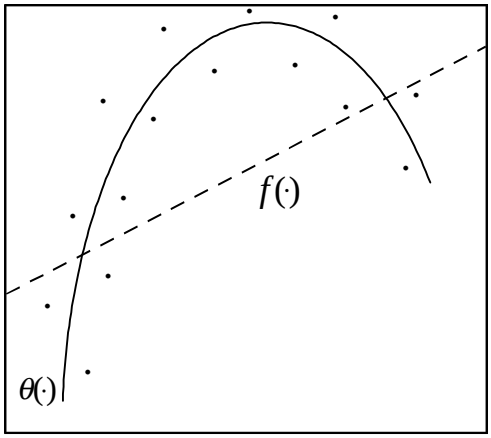


Fig.(2.7.a) Variance of too large model $f(\cdot)$ Fig.(2.7.b) Bias of too small model $f(\cdot)$

In the empirical application of an approximation it is impossible to prevent an error caused by a bias or the variance. This dilemma is termed the "Bias-Variance-Dilemma" (e. g. German/Bienenstock/Doursat (1992)). Bias and variance are complementary quantities, and the best generalization is obtained when we have the best compromise between the conflicting requirements of small bias and small variance.

In order to find the optimum balance between bias and variance we need to have a way of controlling the effective complexity of the model. In the case of neural networks, the complexity can be varied by changing the number of parameters in the network, which is proportional to the number of hidden nodes (see Fig.2.6). Alternatively, one can remove individual weights. However, as we will show, the hidden nodes can be considered as the pivotal and defining elements of both NNs and FLSs, therefore we vary complexity solely by the addition and deletion of hidden units, with the goal of arriving at an optimal network structure.

We next consider how such a network can learn a suitable mapping from a given data set.

2.5 Learning Algorithms

Besides the network structure another major characteristic of an NN application is the form of optimization, learning or training the network undergoes.

Traditional optimization is often described in NN slang by the term "*supervised learning*", because target values for the network outputs are used to minimize the error of the mapping from the input to the output. The remainder of this section is concerned with the networks techniques for training or "learning". The problem of minimizing continuous, differentiable functions of many variables is one which has been studied extensively in applied mathematics, and many of the conventional approaches to this problem are directly applicable to neural networks. Generally one can distinguish among two major types of learning algorithms. On one

side there are the stochastic (global) methods like Simulated Annealing, Simplex Simulated Annealing, Threshold Accepting, Genetic Algorithm and on the other side there are gradient descent (local) methods like Newton-Raphson, Quasi-Newton, Levenberg-Marquardt and Back Propagation. Many standard textbooks cover non-linear optimization techniques including Polak (1971), Gill(1981), Dennis and Schnabel (1984), Luenberger (1984), and Fletcher (1987). For didactic reasons we will contrast one representative of each class of learning algorithms, namely the genetic algorithm for the stochastic class and the backpropagation algorithm for the deterministic class and discuss their domains of application and their limitations.

2.5.1 Genetic Algorithm

Similarly to NNs, Genetic algorithms (GAs) were inspired by processes observed in nature, namely evolution. The basic concept behind this technique is as follows. Any set of network weights is associated by a cost or "fitness" function with its degree of relevance for the output indicating its effectiveness. For example, the fitness can be simply given by $-E$, where E is the value of the cost function or error function for that set of weights. Starting with a random population of such weight vectors, successive generations are constructed using genetic operators to construct new vectors out of old ones such that fitter strings (weights) are more likely to survive and to participate in crossover operations. The crossover operations can in principle bring together good building blocks such as hidden units that compute certain logical functions found by chance in different members of the population. Hence it is natural that GAs have been used to search the weight space of an NN without the use of any gradient information [Montana and Davis, 1989, Whitley and Hanson, 1989, Ichikawa and Sawa, 1992]. Stochastic methods like GAs perform a global searches and are thus on one hand not prone to converge to local minima, on the other hand their speed of convergence is very slow.

2.5.2 Back Propagation

The Back Propagation learning algorithm is one of the most important historical developments in neural networks. It has reaweaken the scientific and engineering community to the modeling and processing of many quantitative phenomena using NNs. This learning algorithm requires the NN to have continuously differentiable activation functions (see figure 2.3). The basis for this weight update algorithm is simply the gradient-descent method. For a given input-output pair $\{(X_k, Y_k)\}, k = 1, 2, \ldots, T$, the back propagation algorithm performs two phases of data flow. First, the input pattern X_k is propagated from the input layer to the output layer and, as a result of this forward flow of data, it produces an actual output $f(X_k, \delta)$. Then the error signals resulting from the difference between $f(X_k, \delta)$ and Y_k are back propagated from the output layer to the previous layers for them to update their weights according to the gradient-descent method. For

more detail description to back propagation we recommend Bishop (1995).

Gradient descent methods like back propagation converge fast, but often become stuck at local minima and are thus often unable to find satisfactory solutions. Repeating the process with random initializations of the weights provides a remedy in this case, but reduces the convergence speed.

Given the pros and cons of these methods it is hardly surprising that we use for the empirical application in section 6 a combination of stochastic optimization technique with local deterministic procedures.

Based on above paragraphs, NNs in conjunction with a learning algorithm can be described as data-driven self-adaptive methods. In other words, they learn from examples and find out functional relationships among the data even if the underlying relationships are unknown or hard to describe. Thus NNs are suited for problems whose solutions require knowledge that is difficult to specify but for which there are enough data or observations. In this sense NNs can be treated as one of the multivariate nonlinear nonparametric statistical methods [White, 1989; Ripley, 1996]. This makes them useful for many complex problems, since it is often easier to get data than to have good theoretical guess about the underlying laws of the systems from which the data are generated. NNs are hence often called *model-free* estimators. However, they are far from unique in that capacity and they do have the drawback that their parameters are quite meaningless for the researchers intention. This is the main difference between NNs and the FLSs which we consider next.

3 Fuzzy Logic

3.1 Introduction

Analogously to NNs, FLSs are inspired by the workings of the human brain. In contrast to NNs, which are modeled after the physical architecture of the brain, FLSs are based upon the way the brain deals with inexact information.

Fuzzy sets (Zadeh, 1965) are a way to quantify the vagueness inherent in linguistics; mathematically they can be considered a generalization of classical set theory. In a classical set, an element of the universe either belongs to or does not belong to the set. That is, the membership of an element is crisp (binary). A fuzzy set is a generalization of an ordinary set in that it allows the degree of membership for each element to range over the unit interval [0,1]. One of the biggest differences between crisp sets and fuzzy sets is that the former have a single unique membership function, whereas for fuzzy sets there exists an infinite number of admissible membership functions that may represent it. Any crisp set can be fuzzified and hence generalized by replacing the binary membership function by the concept of a fuzzy membership function. Such fuzzification yields greater generality through the distinction between degrees of set membership among members of a single set. Due to the mostly unfamiliar nature and terminology, we proceed to introduce the principal concepts and mathematical notions of fuzzy set theory in the following paragraph.

3.2 Basic Concepts of Fuzzy Sets

Definition 8 Fuzzy Set: *Let U be a collection of objects, for example, $U = \Re^n$, and be called the universe of discourse. A fuzzy set F in U is characterized by a membership function $\mu_F : U \rightarrow [0, 1]$, with $\mu_F(u)$ representing the grade of membership of $u \in U$ in the fuzzy set F.*

Example 9 *Let $U = \Re$ and let the crisp set A represent " real numbers greater than or equal to 5 "; where the characteristic function is*

$$1_A(x) = \begin{cases} 0, & x < 5 \\ 1, & x \geq 5 \end{cases} x \in U,$$

which is shown in figure 3.1.(a). Now let fuzzy set B represent " real numbers close to 5 " where the membership function is

$$\mu_B(x) = \frac{1}{1 + 10(x-5)^2}, x \in U,$$

which is shown in figure 3.1.(b).

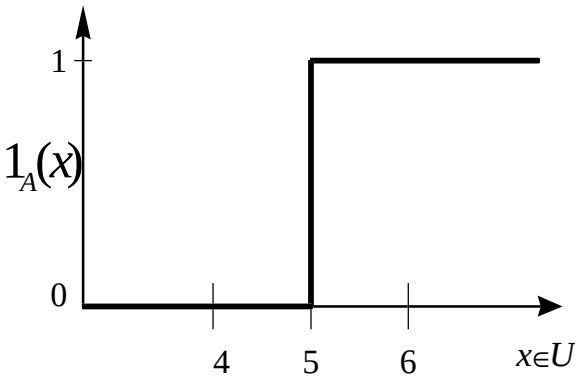


Fig. 3.1.(a)

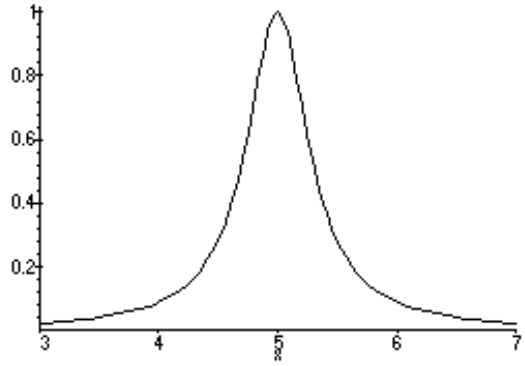


Fig.: 3.1.(b)

Consider another example, the fuzzy variable temperature, which can be described by many different adjectives each with its own fuzzy set. A typical partition of the universe of discourse, 0 - $40^{\circ}C$, is shown in figure 3.2., where the fuzzy sets *cold*, *warm* and *hot* are defined. Here, the crisp temperature $20^{\circ}C$ has a grade of membership of 0.5 for both the *cold* and *warm* fuzzy sets i.e. $\mu_{cold}(20^{\circ}C) = \mu_{warm}(20^{\circ}C) = 0.5$.

It is clear that the definition of fuzzy sets are non-unique and are very context-dependent i.e. these sets may seem wrong to an Eskimo, or even to the reader. This is the nature of language and shows that the actual definition of the sets are both application and user specific.

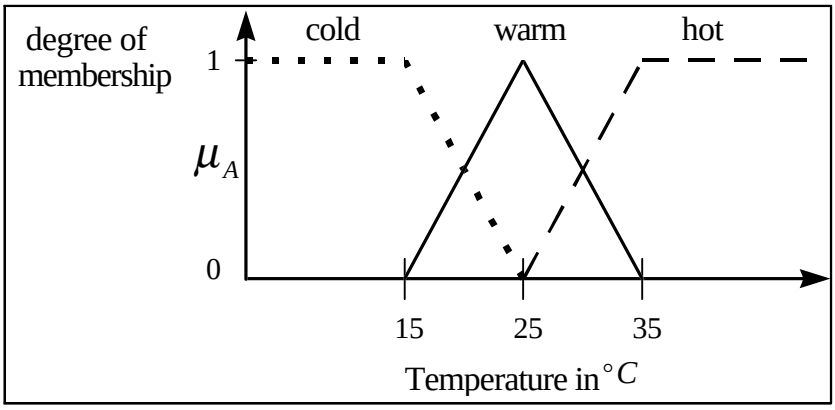


Figure 3.2: Fuzzy sets defined for temperature.

This inaccuracy in language, described by fuzzy sets, can also be thought as uncertainty. This view has lead to the fuzzy versus probability debate, where probabilists question the uses and effectiveness of fuzzy set theory. I do not wish to enter this debate here.

Another way of representing a fuzzy set is through use of *support* or *center* of a fuzzy set.

Definition 10 Support , Center and Fuzzy Singleton: *The support of a fuzzy set F is the crisp set of all points $u \in U$ such that $\mu_F(u) > 0$. The center of a fuzzy set F is the point(s) $u \in U$ at which $\mu_F(u)$ achieves its maximum value. If the support of a fuzzy set F is a single point in U at which $\mu_F = 1$, then F is called a fuzzy singleton.*

Definition 11 Fuzzifier: *A mapping f from a crisp point $u \in U$ to a fuzzy set F is called fuzzifier. There are at least two possible choices of this mapping :*

- *Singleton fuzzifier: F is a fuzzy singleton with support u, that is , $\mu_F(u) = 1$ for $u' = u$ and $\mu_F(u') = 0$ for all other $u' \in U$ with $u' \neq u$.*
- *Nonsingleton fuzzifier: $\mu_F(u') = 1$ and $\mu_F(u')$ decreases from 1 as u' moves away from u , for example, $\mu_F(u') = \exp[-\frac{(u'-u)^T(u'-u)}{\sigma^2}]$, where σ^2 is a parameter characterizing the shape of $\mu_F(u')$.*

We will only consider singleton fuzzifiers.

Definition 12 Defuzzifier: *A mapping f from a fuzzy set A to a crisp point $u \in U$ is called defuzzifier.*

The three basic operations on crisp sets — complement, intersection and union — can be generalized to fuzzy sets in more than one way. However, here are the operations that are usually referred to as standard fuzzy set operations.

Definition 13 Intersection, Union, and Complement: *Let A and B be two fuzzy sets in U. The intersection $A \cap B$ of A and B is a fuzzy set in U with membership function defined for all $u \in U$ by*

$$\mu_{A \cap B}(u) = \min\{\mu_A(u), \mu_B(u)\}.$$

The union of $A \cup B$ of A and B is a fuzzy set in U with the membership defined for all $u \in U$ by

$$\mu_{A \cup B}(u) = \max\{\mu_A(u), \mu_B(u)\}.$$

The complement $\overline{A}$ of A is a fuzzy set in U with the membership function defined for all $u \in U$ by

$$\mu_{\overline{A}}(u) = 1 - \mu_A(u).$$

Applying these standard operations to the fuzzy sets in figure 3.2 we can find for example, that $\overline{cold} \cap \overline{hot} = warm$. The construction of $\overline{cold} \cap \overline{hot}$ is shown in figure 3.3.(c).

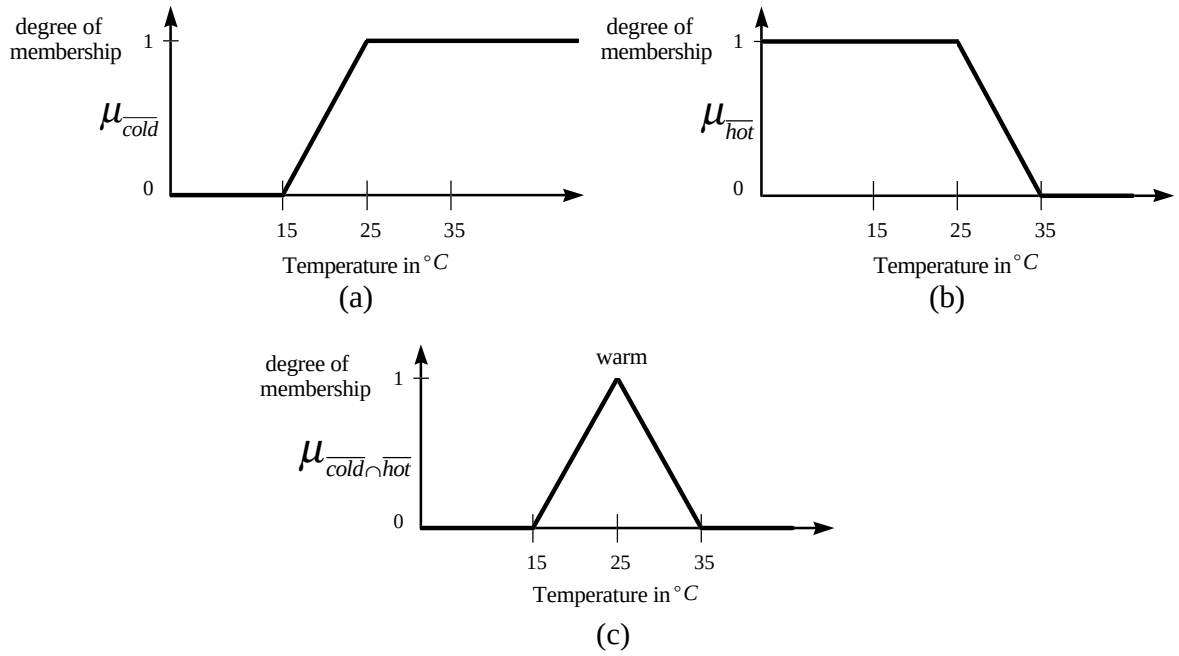


Figure 3.3. (a),(b),(c) : Illustration of standard operations on fuzzy sets

The equation $\overline{cold} \cap \overline{hot} = warm$ makes sense: If the temperature is not cold and not hot then it is warm!

We now introduce the *extension principle* it is one of the most important tools of fuzzy set theory. This principle allows the generalization of crisp mathematical concepts to the fuzzy set framework and extends point-to-point mappings for fuzzy sets. It provides a means for any function f that maps an n-tuple $(x_1, \ldots, x_n)$ in the crisp set U to a point in the crisp set V to be generalized to mapping n fuzzy subsets in U to a fuzzy subset in V. Hence, any mathematical relationship between nonfuzzy elements can be extended to deal with fuzzy entities.

Definition 14 The Extension Principle: *Let U and V be two universes of discourse and be a mapping $f : U \to V$. For a fuzzy set A in U, the extension principle defines a fuzzy set B in V by*

$$\mu_B(v) = \sup \left\{ \mu_A(u) : u \in f^{-1}(v) \right\}.$$

That is, $\mu_B(v)$ is the superium of $\mu_A(u)$ for all $u \in U$ such that $f(u) = v$, where $v \in V$ and we assume that $f^{-1}(v)$ is not empty. If $f^{-1}(v)$ is empty for some $v \in V$, define $\mu_B(v) = 0$.

The notion of relations is basic in science and engineering, which is essentially the discovery of relations between observations and variables. The traditional crisp relation is based on the concept that everything is either related or unrelated. Hence, a crisp relation represents the presence or absence of interactions between the elements of two or more sets. By generalizing this concept to allow for various degrees of interactions between elements, we obtain the fuzzy relation.

Definition 15 Fuzzy Relation: *Let U and V be two universes of discourse. A fuzzy relation R is a fuzzy set in the product space $U \times V$; that is, R has the membership function $\mu_R(u, v)$, where $u \in U$ and $v \in V$.*

A crisp relation among crisp sets $X_1, \ldots, X_r$ is a crisp subset on the cartesian product $X_1 \times \cdots \times X_r$. This relation is denoted by $R(X_1, \ldots, X_r)$. Hence

$$R(X_1, \ldots, X_r) \subset X_1 \times \cdots \times X_r,$$
$$\text{where } X_1 \times \cdots \times X_r = \{(x_1, \ldots, x_r) : x_i \in X_i \text{ for all } i \in \{1, \ldots, r\}\}.$$

It is interpreted that the relation R exists among $\{X_1, \ldots, X_r\}$, if the tuple $(x_1, \ldots, x_r)$ is in the set $R(X_1, \ldots, X_r)$; otherwise the relation R does not exist among $\{X_1, \ldots, X_r\}$. A fuzzy relation is a fuzzy set defined on the cartesian product of a crisp sets $\{X_1, \ldots, X_r\}$, where tuples $(x_1, \ldots, x_r)$ may have varying degrees of membership $\mu_R(x_1, \ldots, x_r)$ within the relation.

In the simplest case, consider two crisp sets X_1, X_2. Then

$$R(X_1, X_2) = \{((x_1, x_2), \mu_R(x_1, x_2)) : (x_1, x_2) \in X_1 \times X_2\},$$

is a fuzzy relation on $X_1 \times X_2$. It is clear that a fuzzy relation is basically a fuzzy set. Since the fuzzy relation is a fuzzy set, the operations for fuzzy sets (see Def.13) can be extended to fuzzy relations. A special kind of fuzzy relation are the fuzzy implications.

Definition 16 Fuzzy Implication: *Let A and B be fuzzy sets in U and V, respectively. A fuzzy implication, denoted by $A \rightarrow B$, is a fuzzy relation in $U \times V$ with the following membership functions:*

- *Fuzzy product rule:*

$$\mu_{A \rightarrow B}(u, v) = \mu_A(u) \cdot \mu_B(v)$$

- *Fuzzy minimum rule:*

$$\mu_{A \rightarrow B}(u, v) = \min\{\mu_A(u), \mu_B(v)\}$$

- *Fuzzy conjunction:*

$$\mu_{A\to B}(u,v) = \max\{0, \mu_A(u) + \mu_B(v) - 1\}$$

- *Generalization of modus ponens:*

$$\mu_{A\to B}(u,v) = \sup\{c \in [0,1] : \max\{0, \mu_A(u) + c - 1\} \leq \mu_B(v)\}$$

- *Generalization of modus tollens:*

$$\mu_{A\to B}(u,v) = \inf\{c \in [0,1] : \min\{1, \mu_B(v) + c\} \leq \mu_A(u)\}$$

A fuzzy implication $A \to B$ can be understood as a fuzzy **IF-THEN** rule:
If u is A **THEN** v is B, where $u \in A$ and $v \in B$ and "u is A" stands for the degree of membership of u in A ; A and B are fuzzy sets. A collection of fuzzy IF-THEN rules is called *Fuzzy Rule Base.*

Definition 17 Fuzzy Rule Base: *A fuzzy rule base consists of a collection of q fuzzy IF-THEN rules in the following form:*

$$R^l : IF\ x_1\ is\ F_1^l\ and\ \cdots\ and\ x_r\ is\ F_r^l,\ \ THEN\ y\ is\ G^l,$$

where F_i^l, G^l *are fuzzy sets and* $F_i^l \in U_i \subset \Re$ *and* $G^l \in V \subset \Re$ *and* $X = (x_1, \ldots, x_r) \in U_1 \times \cdots \times U_r$ *and* $y \in V; l = 1, \ldots, q.$

There are several ways to combine the fuzzy IF-THEN rules into a mapping from fuzzy sets in $U_1 \times \cdots \times U_r$ to fuzzy sets in V as in Def.15 . The rule in which way this IF-THEN rules are combine is called *Fuzzy Inference Rule.* We only be concerned with the product rule of fuzzy implication (Def. 16) i.e. the *product inference rule.* For simplicity we denote $F_1^l \times \cdots \times F_r^l = A$ and $G^l = B$, with the help of the product rule we get out of the IF-THEN rules the following mapping from the fuzzy set A to fuzzy set B :

$$\begin{aligned}\mu_{A\to B}(\overline{x}, y) &= \mu_A(\overline{x}) \cdot \mu_B(y) \\ &= \mu_{F_1^l \times \cdots \times F_r^l}(\overline{x}) \cdot \mu_B(y) \\ &= \mu_{F_1^l}(x_1) \cdots \mu_{F_r^l}(x_r) \cdot \mu_B(y).\end{aligned}$$

As fuzzy sets are extensions of classical crisp sets, fuzzy logic is an extension of classical two-valued logic. Classical bivalence logic deals with propositions that are required to be either true or false, the latter are called the truth values of the propositions. Propositions are sentences expressed in some language and can be expressed, in general, in a canonical form, "u is A" like in Def.16. The truth values of propositions in fuzzy logic are allowed to range over the fuzzy subsets of the unit interval [0,1] or point in the interval.

Fuzzy logic systems (FLS) is the name for systems which have a direct relationship with fuzzy concepts like fuzzy sets, fuzzy relations and fuzzy logic.

3.3 Fuzzy Logic Systems

A fuzzy logic system is a nonlinear model whose behaviour is described by a set of rules such as:

IF temperature is cold **Then** set output of the heater high
or
IF temperature is warm **Then** set output of the heater zero

describing a typical relationship between room temperature and the desired output of a heater. To represent the complete relationship a collection of such rules called *the rule base* is used. Real valued (crisp) inputs are converted to vague fuzzy variables by the *fuzzifier,* these are then presented to the *rule base.* The *rule base* produces a collection of fuzzy output variables from the consequences of the rules. In a *fuzzy inference engine*, fuzzy logic principles are used to combine the fuzzy IF-THEN rules in the fuzzy rule base into a mapping from fuzzy sets in U to fuzzy sets in V. The *defuzzifier* converts these fuzzy sets in V into real-valued outputs. The basic structure of a fuzzy logic system, as described by Wang, is shown in figure 3.4 and a description of these elements are introduced in Section 3.2.

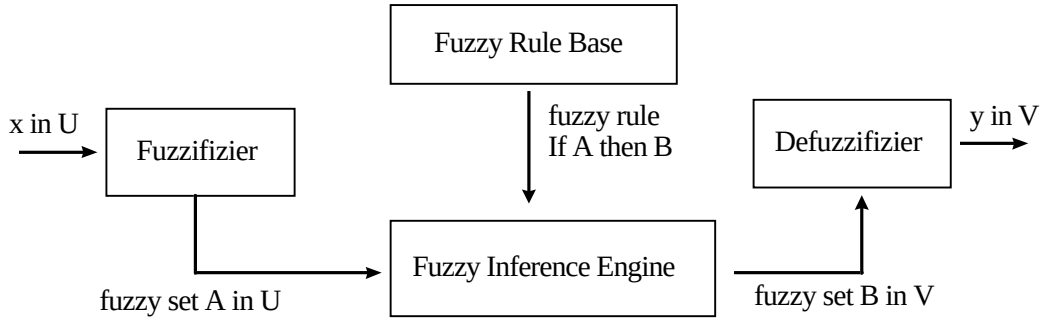


Figure 3.4: The basic components of a fuzzy logic system

Definition 18 Fuzzy Logic System: *A mapping from* $\Re^r \rightarrow \Re$ *consisting of a fuzzifier, a fuzzy rule base with a fuzzy inference engine and a defuzzifier is a fuzzy logic system.*

Example 19 *The class of functions of interest in this paper is* $f : \Re^r \rightarrow \Re$ *s.t.*

$$f(x) = \sum_{j=1}^{q} \beta_j \cdot \mu_{B^j}(\beta_j) \tag{1}$$

with

$$\mu_{B^j}(\beta) = \sup\left\{x \in A : \mu_{U_1^j \times \cdots \times U_n^j \to B^j}(x, \beta) \star \mu_A(x)\right\},$$

where $x = (x_1, \ldots, x_r) \in A \subset U_1 \times \cdots \times U_r \subset \Re^r$, $\beta \in \Re$. *A is a fuzzy set in $\Re^r$ and be the input to the fuzzy inference engine; then each fuzzy IF-THEN rule is interpreted as a fuzzy implication $U_1^j \times \cdots \times U_r^j \to B^j$ in $\Re^r \times \Re$ and determines a fuzzy set B^j in $\Re$. The star $\star$ stands for composition. $\star$ is a twoplace function from $[0,1] \times [0,1]$ to $[0,1]$, which includes fuzzy intersection ($a \star b = \min\{a, b\}$) or algebraic product ($a \star b = ab$), where $a, b \in [0,1]$.*

Lemma 20 *For the product rule as fuzzy implications (Def. 16) and a singleton fuzzyifier (Def. 11) the fuzzy logic system in (1) can be simplified to:*

$$f(x) = \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \mu_{U_k^j}(x_k) \tag{3.1}$$

where $x = (x_1, \ldots, x_r) \in \Re^r$.

Proof. *(See Wang (1994) section 2.6.2).*

Note that there are many more possible fuzzy logic systems (or fuzzy logic approximator) based upon different choices for the four components given above.

3.4 Universal Approximation

Analogously to NNs (see Section 2.3) the denseness property of FLSs in the space of continuous and measurable functions has also been investigated in the literature. Since the FLSs depend not only on the membership functions but also on the implication rule, it should be clear that there are vast numbers of different classes of FLSs resulting from different implication rules, which need to be analyzed individually with regard to their denseness property. Hence there exists a rather diverse literature in the fields of engineering and finance dealing with the respectively relevant fuzzy logic systems. Due to the resulting diversity of the literature and terminology, we will only give a list of the presently known results, relevant to us and not attempt to provide all the individual proofs.

- Fuzzy logic systems such as (3.1) with a center average defuzzifier, product inference rule, a singleton fuzzifier, and a Gaussian membership function μ are dense in C^r and L^2 (Wang 1994) and have the following form:

$$f(x) = \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \mu(x_k) = \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \alpha_{jk} \exp\left(-\left(\frac{x_k - \delta_{jk}}{\sigma_{jk}}\right)^2\right),$$

  with $x = (x_1, \ldots, x_r) \in \Re^r; \alpha_{jk}, \beta_j, \sigma_{jk}, \delta_{jk} \in \Re$.

- An extension of the above to membership functions of positive integer powers of Gaussians is given by Gottschling (1997) such that (3.1) is given by:

$$\begin{aligned} f(x) &= \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \mu(x_k) \\ &= \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \alpha_{jk} \left(\exp\left(- \left(\frac{x_k - \delta_{jk}}{\sigma_{jk}} \right)^2 \right) \right)^n , \end{aligned}$$

where $x = (x_1, \ldots, x_r) \in \Re^r; \alpha_{jk}, \beta_j, \sigma_{jk}, \delta_{jk} \in \Re$ and $n \in \{1, 2, \ldots\}$.

- Minimum inference FLSs with symmetric unimodal and nonnegative membership functions are known as universal approximators on L^P (Gottschling and Kreuter, 1999).

$$\begin{aligned} f(x) &= \sum_{j=1}^{q} \beta_j \cdot \min\{\mu(x_1), \ldots, \mu(x_r)\} \\ &= \sum_{j=1}^{q} \beta_j \cdot \min\left\{ \exp\left(- \left(\frac{x_1 - \delta_{j1}}{\sigma_{j1}} \right)^2 \right), \ldots, \exp\left(- \left(\frac{x_r - \delta_{jr}}{\sigma_{jr}} \right)^2 \right) \right\}, \end{aligned}$$

where $x = (x_1, \ldots, x_r) \in \Re^r; \beta_j, \sigma_{jk}, \delta_{jk} \in \Re$.

- Multiplicative FLSs such as (3.1) with arbitrary continuous, nonconstant and integrable membership functions μ have also been shown to be dense in $C^r$and L^2 (Gottschling and Tatur, 2000).

  Thus, if $\mu : \Re \to [0, 1]$ is a sigmoid functions of the following form:

$$\mu(x_k) = \frac{1}{1 + \exp(-(w_0 + w_1 x_k))}; w_0, w_1, x_k \in \Re$$

  then (3.1) becomes to:

$$\begin{aligned} f(x) &= \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \mu(x_k) \\ &= \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \left(\frac{1}{1 + \exp(-(w_{jk0} + w_{jk1} \cdot x_k))} \right), \end{aligned} \tag{3.2}$$

  and $f(x)$ is also a universal approximator.

3.5 Fuzzy Logic Systems in a Neural Network Context

Sofar we have treated NNs and FLSs as completely distinct nonlinear modeling approaches.However, given the fundamental inspirations for these two fields it should not come as too surprising a result that there are numerous parallels in the resulting functional representations. By observing the functional form of equation (3.2), it follows that a representation of a FLSs as single hidden layer feedforward networks exist (Figure 2.5). It then becomes straightforward to apply all the learning algorithms in section 2.5 to adjust the parameters $\beta_j, w_{jk0}, w_{jk1} \in \Re$. For given input-output pairs $(X_t, Y_t), X_t = (x_{t1}, \ldots, x_{tr}) \in \Re^r, Y_t \in \Re, t = 1, \ldots, T$ and the parameter vector $\delta = (\beta_1, \ldots, \beta_q, w_{110}, \ldots, w_{qr0}, w_{111}, \ldots, w_{qr1})$ our task is to determine a fuzzy logic approximator of the form (eq. 3.2) such that an error function e. g.:

$$SSE(\delta) = \sum_{t=1}^{T} (f(X_t, \delta) - Y_t)^2$$

is minimized. Where T is the number of samples or input-output pairs and $f(X_t, \delta)$ is the FLS from (see eq. 3.2), with the parameter vector δ as the 2nd variable. For any given q this constitutes the number of the IF-THEN rules, equivalent to the number of hidden nodes in neural network language. The corresponding functional representation of $f(X_t, \delta)$ is given by figure (3.5):

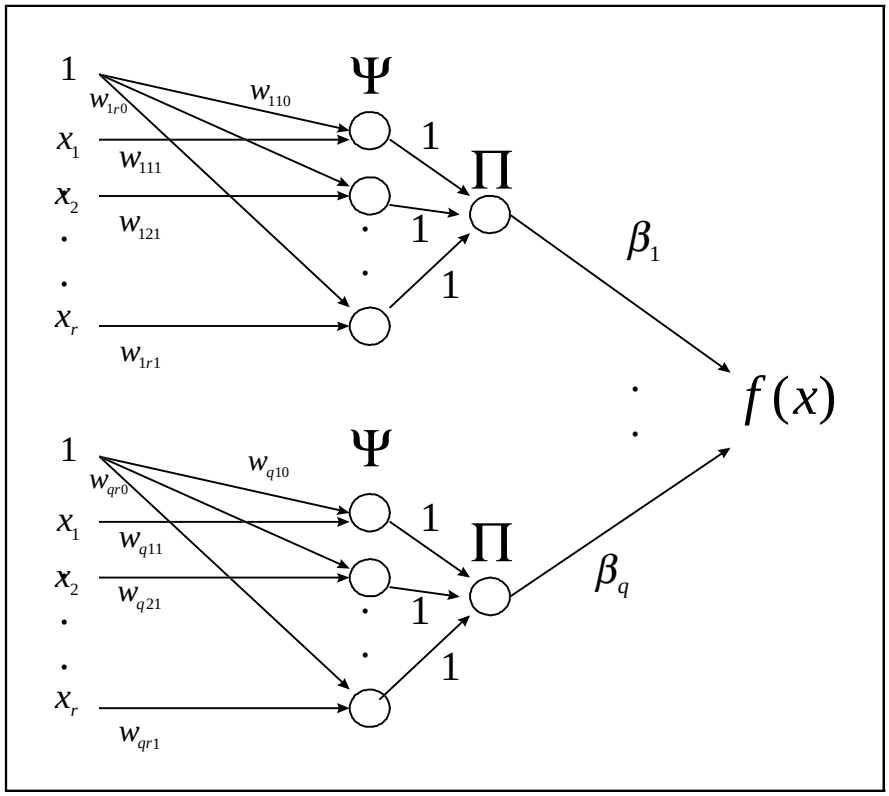


Fig. 3.5: Fuzzy Network with q nodes

Hence we can interpret the fuzzy logic systems of interest e. g. (eq. 3.2) as special cases of networks, which we will refer to as Fuzzy Networks (FN) henceforth. We now proceed to consider the differences and similarities of FNs and NNs.

4 Integrating Neural Networks and Fuzzy Logic Systems

The structural and approximation theoretic similarities point to an area of overlap between fuzzy and neural approximation that can all be analyzed in the context of neural network terminology due to the observations in the last paragraph.

4.1 Differences between Fuzzy Networks and Neural Networks

Recalling the definition of the sigma pi network (Def.3), which was used in the derivation of the universal approximation property for single hidden layer feedforward networks (see section 2.3), one can gain the following insight: a reduced case of the $\sum\Pi$ network, the $\sum\Pi$ is already a sufficiently rich superstructure which contains both the FNs and the NNs (Gottschling, 1997). The original contribution for the nestedness of network structures is recapitulated below.

Definition 21 *For any $r \in \{1, 2, \ldots\}$, let A^r be the set of all functions of the form: $A(X) = wX + w_0; w, X = (x_1, \ldots, x_r) \in \Re^r, w_0 \in \Re$. For any continuous nonconstant functions $\Psi : \Re \to \Re$, let $\sum\Pi$ be the class of functions $f : \Re^r \to \Re$ s.t:*

$$f(X) = \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{l_j} \Psi(A_{jk}(X)) \qquad (4.1)$$

with $\beta_j \in \Re, A_{jk} \in A^r$, $l_j, q \in \{1, 2, \ldots\}$.

Theorem 22 *We can derive a fuzzy logic system (equation 3.1) of the form:*

$$f(X) = \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \mu_{A^k}(x_k)$$

from the above $\sum\Pi$ structural equation (4.1) by imposing 3 restrictions on (4.1):

1) $l_j = r$

2)$\Psi : \Re \to [0, 1]$

3)$A(X) = \left\{ all\ A(X) \;\middle|\; \begin{array}{c} with\ w = (0, \ldots, 0, w_{jk1}, 0, \ldots, 0), \\ w_{jk1} \neq 0, k = 1, 2, \ldots, r; j = 1, \ldots, q; \end{array} \right\}$.

Proof. *By inspection after substitution of 1,2 and 3 .* ■

Theorem 22 gives as a network representation of a fuzzy logic system (see equation 3.2) which is in fact a restricted $\sum\Pi$ network of the form:

$$f(X) = \sum_{j=1}^{q} \beta_j \cdot \prod_{k=1}^{r} \Psi(w_{jk1} \cdot x_k + w_{jk0}); w_{jk1}, w_{jk0} \in \Re; X \in \Re^r \tag{4.2}$$

Implications: From Hornik et al. [1989] we know that a neural network :

$$f(X) = \sum_{j=1}^{q} \beta_j \Psi(A_j(x)) = \sum_{j=1}^{q} \beta_j \Psi(w_{i0} + \sum_{j=1}^{r} w_{ij} x_j) \tag{4.3}$$

are the result of the restriction $l_j = 1$ on the $\sum\prod$-structure in (4.1). Thus we can define the $\sum\prod$-structure (4.1) as the superstructure containing both the neural network (NN) and the fuzzy network (FN).

The following pictures (Fig. 4.1, Fig 4.2) gives the possibility to compare the two network structures:

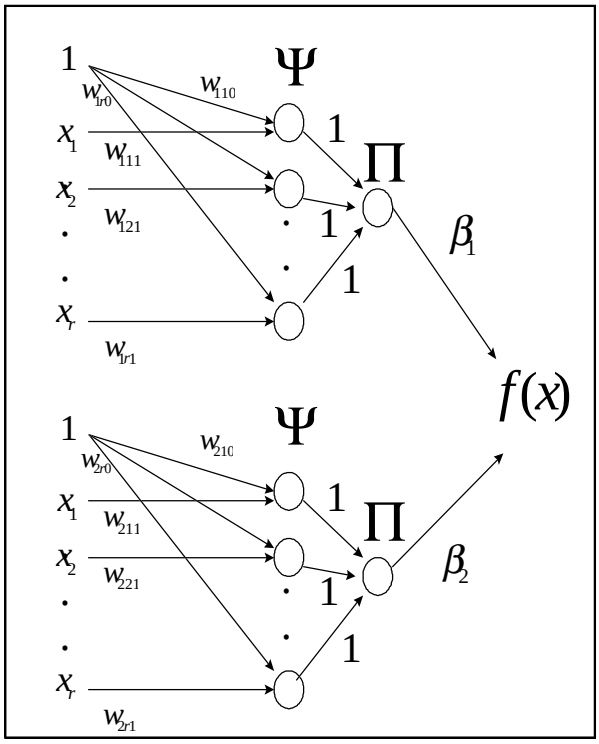


Figure 4.1: Fuzzy network with 2 nodes

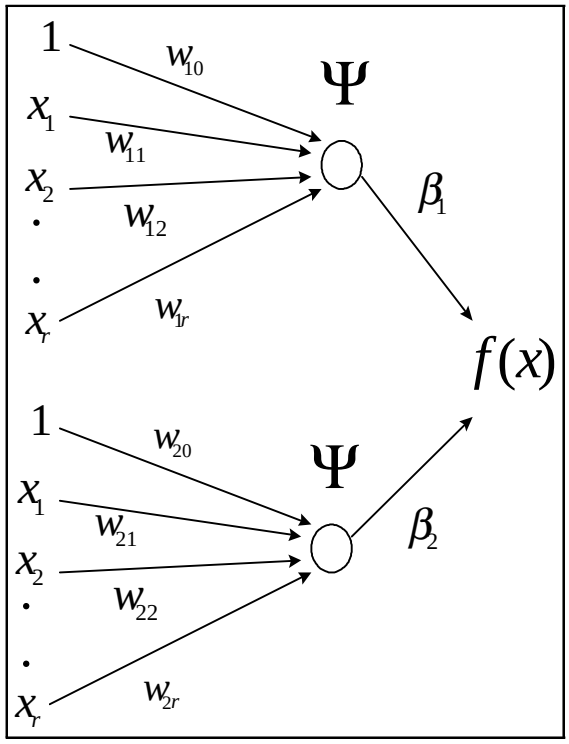


Figure 4.2: Neural network with 2 nodes

The difference of the two network types is the processing of the input variables. In an FN each input variable x_i uses alone one processing element or activation function (Ψ), whereas in an NN an affine image of all input variables uses only

one activation function. Thus in FNs the focus is on the functional relation between each individual input variable (x_i) and the output, whereas in NNs the focus is on the functional relation between the sum of all inputs variables and the output. This provides also the basis for the interpretability of FNs since the direct mapping from variables to output allows for intuitive reasoning with respect to the parameters whilst in an NN the noninvertable subspace projection leads to the "black box" issue.

Since in an FN each input variable has a mean and a scaling parameter associated with it, whilst only one mean parameter is associated with the sum of the inputs in NNs, it follows that the former has more parameters for the same number of hidden nodes. It is this last fact that is partly responsible for the significant difference in practical approximation applications of the NNs and FNs with their asymptotic equivalence notwithstanding. Given the consideration at the outset of the paper, we now have the framework to look into the realm of hybrid (mixed) network structures, which are shown to be highly relevant in small samples for several reasons.

4.2 Motivation of Hybrid Networks

A significant part of statistical literature has considered the problem of robustifying forecasts by combination, both in linear (e. g. Granger) as well as nonlinear contexts (Breimann, Wolpert). While this phenomenon can be explained by variance reduction alone, the question is whether there is also arguments for such a mixture of functions in a purely deterministic context (bias reduction). This has been first addressed by Stinchcombe (1995) by a formalization of the following argument. The universal approximation property or dense span of single hidden layer feedforward network means that the family of functions defined by the network reaches some amount into each and every dimension of the infinite dimensional space of functions. That neural network functions extend into each and every dimension of function space does not imply that they extend either very far into every dimensions or that they extend equally far into all dimensions. However, without both of these criteria, there will exist some functions that can be approximated better than others. Which functions can be approximated well and which ones can not, depends among other criteria on the network structure and activation function. This brings about the issue of precision. Intuitively, functions that are close in structure to some particular neural network representation will be fitted by that representation more precisely than those that are far away. The practical implication is that, there will be functional relations between inputs and outputs that cannot be captured very precisely whilst others can. While it is easy to increase precision in any given network by adding hidden nodes, this strategy is also constrained because the growthrate of number of neurons (q) is limited by the available data points (see section 2.5). It is here that the crucial problem occurs, since growing a network at the rate that does not

increase the variance results in decreasing precision with increasing dimensions. More and more additions to q may be needed to capture the next feature of the functional relation being estimated.

As a suggested remedy it was put forward — however without proof — that a mixture of different types of networks could solve the precision problem. Similar to the variance reduction by combination of forecasts (e. g. Granger), one can potentially gain precision by combining different networks since it means that one combines a function which individually reaches only far into very few dimensions with other functions that achieve the same for dimensions that are not captured well by the first function. As remarked in the introduction, the relevance of this hypothesis has only been substantiated by empirical research and even then the strategy for the combination of networks of different types has always been top-down in the sense that ex-post combinations of networks were analyzed with respect to their joint performance. Neither a theoretical basis nor an efficient strategy has been suggested how to build combined networks bottom-up. That however is necessary to avoid the model selection bias (see Campbell, (1997)).

In the following paragraph we derive a new hybrid network, which is designed around these considerations of precision and subsequently we shall prove that this strategy is both theoretically sound as well as efficient for combining NNs and FNs.

4.3 A Hybrid Construction Approach for Neuro Fuzzy Networks

Starting out from the network diagrams or equivalently equations 4.2 and 4.3 , one can deduce that the idiosyncratic differences of FNs and NNs can be pinpointed at the hidden node level and the input to hidden node weight structure. We can thus first define a hybrid hidden node by constructing the minimal structure that can be used as either a neural or a fuzzy hidden node:

Definition 23 *For* $\delta = \left(w_0, w_1 \ldots, w_r, w_{10} \ldots, w_{(r-1)0}, w_{11}, \ldots, w_{(r-1)1}\right) \in \Re^{3r-1}$ *the parameter vector and any* $r \in \{2, 3, \ldots\}$ *and any continuous nonconstant functions* $\Psi : \Re \rightarrow \Re$ *, let* $h(X)$ *be a hybrid node of the class of functions* $h(X) : \Re^r \rightarrow \Re$ *s.t:*

$$h(X) = \Psi(w_0 + \sum_{j=1}^{r} w_j x_j) \cdot \prod_{k=1}^{r-1} \Psi(w_{k1} x_k + w_{k0}) \qquad (4.4)$$

with $X = (x_1, \ldots, x_r) \in \Re^r$*. A network* $f(X)$ *made out of* q *hybrid nodes has the following form:*

$$f(X) = \sum_{i=1}^{q} \beta_i \cdot h_i(X)$$

$$= \sum_{i=1}^{q} \beta_i \left(\Psi(w_{i0} + \sum_{j=1}^{r} w_{ij} x_j) \cdot \prod_{k=1}^{r-1} \Psi(w_{ik1} x_k + w_{ik0}) \right) \tag{4.5}$$

One hybrid node $h_i(X)$ with r input variables has the following diagram associated with it.

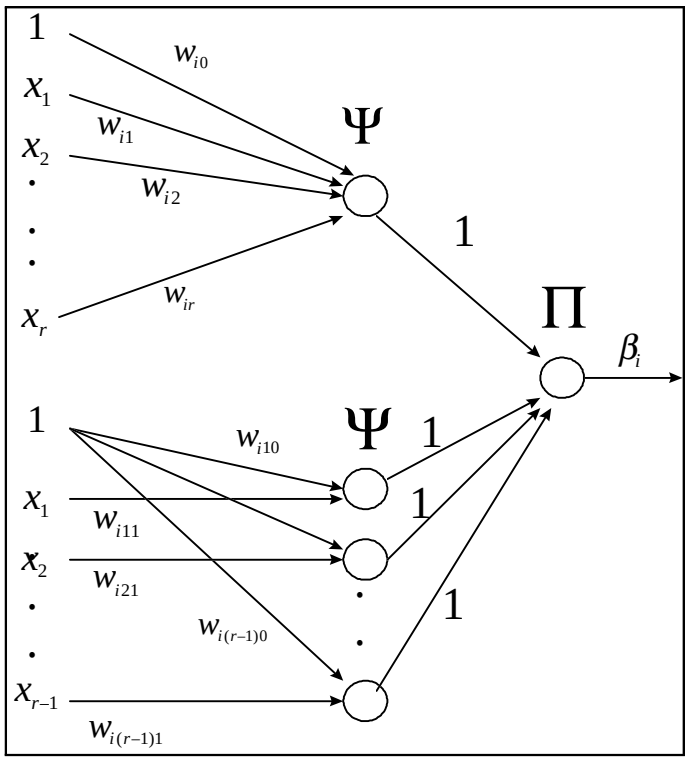


Figure 4.3: Hybrid node $h_i(x)$ with r input variables

Consider that the hybrid unit includes both unit types, the neural hidden node (compare Figure (4.2)) and the fuzzy hidden node (see Figure (4.1)). However, note that it is not possible for the hybrid unit to have a simultaneous realization of both nodes and thus the hybrid unit enforces a choice among the functions during the course of the optimization. For example to get a neural unit out of the hybrid unit we have to set the parameters $w_{ik1} = 0$ for all $k = 1, \ldots, r-1$ in equation 4.5. With this restriction $h_i(X)$ in 4.5 becomes to:

$$h_i(X) = \Psi(w_{i0} + \sum_{j=1}^{r} w_{ij} x_j) \cdot \prod_{k=1}^{r-1} \Psi(w_{ik0}). \tag{4.6}$$

Since no input variable and only parameters are involve in the last term of 4.6, we can replace this term by a single parameter $\beta_i \in \Re$. Thus we get :

$$h_i(X) = \Psi(w_{i0} + \sum_{j=1}^{r} w_{ij} x_j) \cdot \beta_i = \beta_i \Psi(A_i(x)),$$

which is equal to the definition of a neural unit (equation 4.3).

To get a fuzzy unit out of the hybrid unit we have to set the parameters $w_{ij} = 0$ for all $j = 1, \ldots, r-1$ in 4.5, then $h_i(X)$ becomes to:

$$\begin{aligned} h_i(X) &= \Psi(w_{i0} + w_{ir}x_r) \cdot \prod_{k=1}^{r-1} \Psi(w_{ik1}x_k + w_{ik0}) \\ &= \prod_{k=1}^{r} \Psi(w_{ik1}x_k + w_{ik0}). \end{aligned} \tag{4.7}$$

Set $w_{i0} = w_{ir0}$ and $w_{ir} = w_{ir1}$. Equation 4.7 is the definition of a fuzzy unit (see equation 4.2).

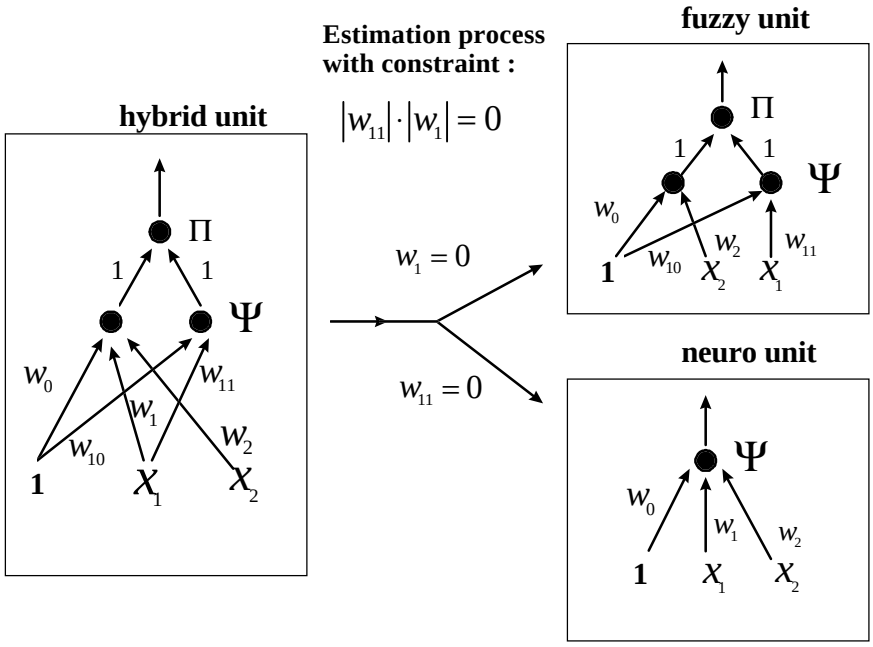


Fig.: 4.4 How a hybrid unit with two variables works

We are now able to get a neural unit or a fuzzy unit out of a hybrid unit (eq. 4.4), just by a simple restriction on the parameters $w_j = 0$ or respectively $w_{j1} = 0$ with $j = 1, \ldots, r-1$; r stand for the number of the input variables. One possible constraint condition to get this restriction on a hybrid unit (see Fig:4.4) is:

$$\left(|w_1| + \cdots + \left|w_{(r-1)}\right|\right) \cdot \left(|w_{11}| + \cdots + \left|w_{(r-1)1}\right|\right) = 0, \forall r. \tag{4.8}$$

This constraint can be use in a process or algorithm to construct a neuro fuzzy network (NFN). All we need is a constraint fit to estimate all parameter of the

NFN simultaneously to fulfilling the constraint (eq. 4.8). Consider the flowchart on the next page to demonstrate this algorithm.

Let T be number of the input-output pairs (X_t, Y_t), $t = 1, \ldots, T$ which describes the wanted function; r be the number of the input variables $X_t = (x_{t1}, \ldots, x_{tr}) \in \Re^r$ and $f(X_t, \delta)$ be the NFN and δ is the parameter vector and let $E(\delta) = \sum_{t=1}^{T} (f(X_t, \delta) - Y_t)^2$ be the error function, which should be minimized with respect to the constraint belonging the hybrid unit. We start with a minimal network which contains just one hybrid unit and then add neural or fuzzy units as the problem requires during the learning process, with the goal of arriving at an optimal network structure.

We estimate the network $f(X_t, \delta)$ by a constraint estimator which allows, beside minimizing $E(\delta)$, also to keep the constraint condition for the parameters of the hybrid unit. We choose all parameter of $f(X_t, \delta)$ randomly and start the estimator. If the estimator after a appropriate time does not converge, we choose other start parameters randomly again until the estimator converges. Then we consider $E(\delta)$, if $E(\delta) \approx 0$ we are finished with our regression, else we have to look at the constraint or respectively at the hybrid unit. If the parameters of the hybrid unit $w_j = 0$ or respectively $w_{j1} = 0$ for all $j = 1, \ldots, r-1$, we take a neural unit or respectively a fuzzy unit (see Fig.:4.4) and continue with the values of the non-zero parameters as our new start parameters of $f(X_t, \delta)$. Our new $f(X_t, \delta)$ has now one neural unit or one fuzzy unit and we start a new estimation with a further hybrid unit and a constraint belonging to this hybrid unit.

We end up adding hybrid units and converting hybrid units into fuzzy or neural units when $E(\delta) \approx 0$. The new neuro fuzzy network has the following functional form.

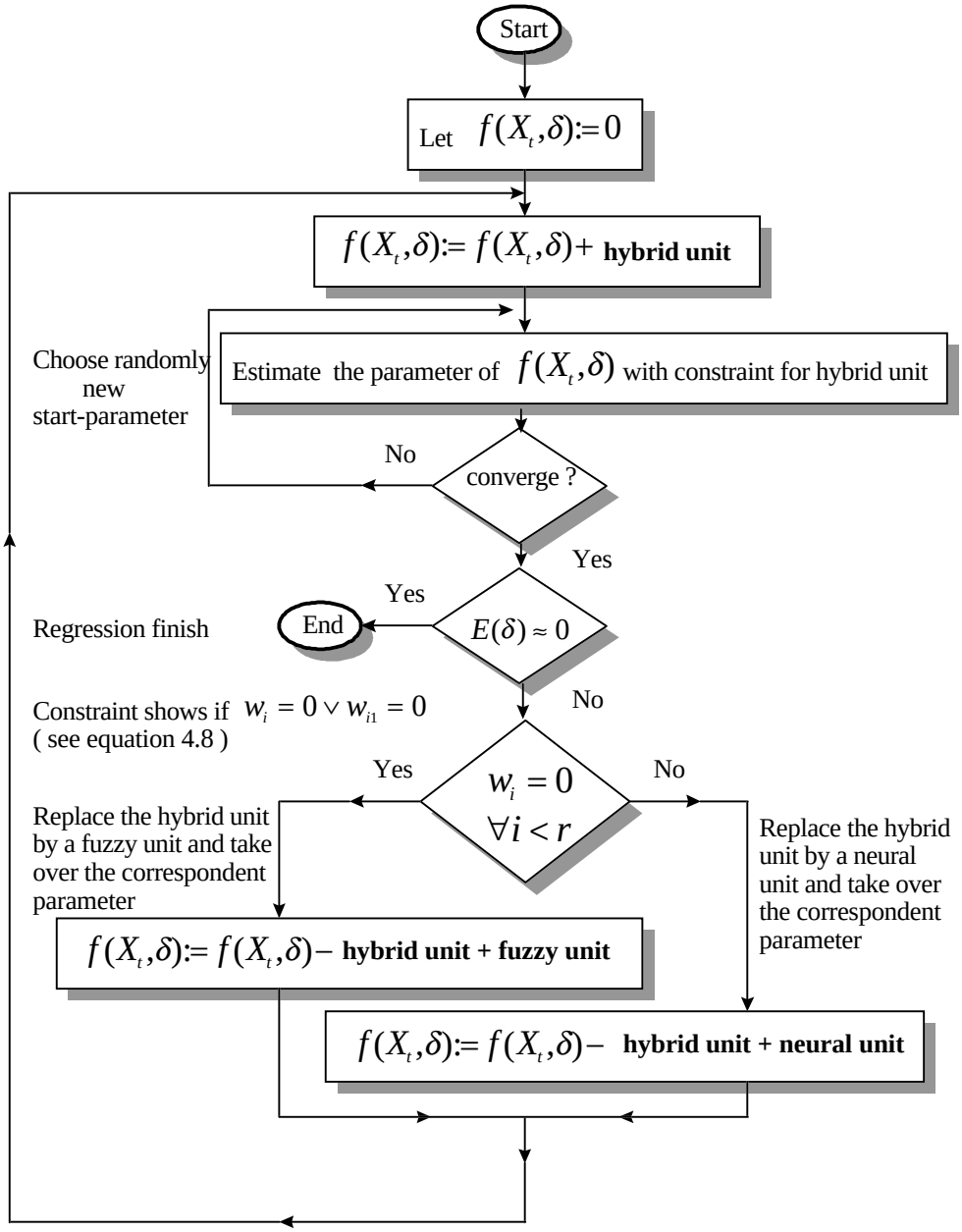
The output of a q hidden unit feedforward NFN with l neural network units and m fuzzy network units given input X_t is defined as

$$f(X_t, \delta) = \beta_0 + \sum_{j=1}^{l} \beta_j \Psi(x_{t1} w_{j1} + \cdots + x_{tr} w_{jr} + w_{j0}) + \sum_{i=1}^{m} \beta_i \left(\prod_{k=1}^{r} \Psi(x_{tk} w_{ik1} + w_{ik0})\right), \tag{4.9}$$

where $l + m = q$,

$\delta = (\beta_0, \ldots, \beta_q, w_{11}, \ldots, w_{lr}, w_{10}, \ldots, w_{l0}, w_{111}, \ldots, w_{mr1}, w_{110}, \ldots, w_{mr0}) \in \Re^n$ is the vector of network parameters with $n = q + 1 + l(r+1) + 2mr$; $l, m, q, r \in \Re$ and Ψ is a sigmoid function (see equation 2.1.).

Algorithm to construct a Neuro-Fuzzy Network:



While the construction is achieved by a well defined algorithm, the properties of the the hybrid network have yet to be established. The next section therefore considers the approximation capability of the NFN.

4.4 Universal Approximation

For completeness we establish that NFNs are also universal approximators (see section 2.3 and 3.4), since they have the same denseness properties as NNs. The required denseness are given by the next result.

Lemma 24 *For any $r, l, m \in \{1, 2, \ldots\}$,the following extended set of NFNs $\sum^{NFN}(\Psi)$ defined by*

$$\sum^{NFN}(\Psi) = \left\{ g : \Re^r \to \Re \;\middle|\; \begin{array}{c} g(X,\delta) = \beta_0 + \sum_{j=1}^{l} \beta_j \Psi(\sum_{n=0}^{r} x_n w_{jn}) \\ + \sum_{i=1}^{m} \beta_i (\prod_{k=1}^{r} \Psi(x_k w_{ik1} + w_{ik0})) \end{array} \right\},$$

is a "universal approximator". Where $g(X, \delta)$ shows the output of NFN with l-neural network units and m-fuzzy network units and given input $X = (1, x_1, \ldots, x_r) \in \Re^{r+1}$.

Proof. *It is clear that an arbitrary neuro-fuzzy network $\sum^{NFN}(\Psi)$ with the same number of neural units as a correspondent neural network has at least the same denseness properties as the $\sum(\Psi)$ (see Def.2). In other words, we will always find for a arbitrary $f \in \sum(\Psi)$ a $g \in \sum^{NFN}(\Psi)$, such that $\rho(f, g) < \varepsilon$ for a given $\varepsilon > 0$, just take over the neural units of f in g and adding redundant fuzzy units. Since $\sum(\Psi)$ is a universal approximator (see section 2.5), it follows that $\sum^{NFN}(\Psi)$ is also a universal approximator.* ■

Given our original interest in constructing efficient networks for small sample problems, we have just established that the NFN is a third type of nonlinear model available for approximation purposes based upon asymptotic arguments. However, in this context all the methods that we consider in the previous sections will work, even the complex $\sum \Pi$ network (see Def.:3) which contains both NNs and FNs simultaneously. In small samples, however, the question of comparative advantage of one network type over another matters. Based upon the problems of the bias-variance dilemma (see section 2.4), it should be clear that the complexity of the network which we consider is proportional to the variance and inversely proportional to the bias. Hence, keeping the number of parameters small is what focuses the attention on the NFN. The key achievement is that the NFN is capable of modeling significant special cases of the much more complex $\sum \Pi$ network (see Def.:3), namely NN and FN without suffering from the same variance. The NFN is also more flexible than either the pure FN or the pure NN, since apart from

reducing to these extreme cases it can end up as a mixture. This is the aspect which makes it relevant in the sense of Stinchcombe discussed above (see section 4.2). In the latter case it is exactly the type of combined network, which is purportedly better, but no formal result has been given. Thus in the next section we will compare the NFN to the NN which was considered by Stinchcombe and formally establish that the NFN provides indeed the improved approximation precision, that was discussed in section (4.2).

5 The Metric Entropy of Neuro Fuzzy Networks

It is not clear how the structure of the different hidden node specifications of a NFN relative to NNs will affect the modeling performance a priori in general. It should be obvious that no single choice of model can be superior to the other for all possible data generating processes, because of the following: if for example the particular functional specification is a member of a NN family then this will give an exact fit up to the limits of the error term. Thus the NFN would be correctly specified, but based on the unnecessary additional parameters, it would be inefficient. The same reasoning applies in reverse if the process is correctly modeled by the neuro fuzzy network. The question that arises then is: can one find conditions under which an NFN performs better than an NN? This is equivalent to analyzing whether the space spanned by an NFN is more extensive in the space of all continuous function than the space spanned by an NN based on the arguments of section 4.2. To answer this question we turn to the theory of metric entropy developed by Kolmogorov and Tihomirov as it allows us to quantify a measure of denseness of different functions in function space.

5.1 Entropy and Capacity

We consider some well-known definitions and describe the idea of entropy and capacity, following the original idea by Kolmogorov and Tihomirov (1959). Let A be a nonvoid set in a metric space R and $\varepsilon > 0$.

Definition 25 *: A family $U_1, \cdots, U_n$ of subsets of R is called an ε-covering of A if the diameter of each U_k does not exceed 2ε and if the sets U_k cover A : $A \subset \bigcup_1^n U_k$.*

Definition 26 *:A finite set of points $x_1, \cdots, x_h$ of R is called an ε-net of A if for each $x \in A$ there is at least one point x_k of the net at a distance from x not exceeding ε : $\rho(x, x_k) \leq \varepsilon$.*

Definition 27 *:The points $y_1, \cdots, y_m \in A$ are called ε-separable if the distance between each two of them exceeds ε : $\rho(y_i, y_k) > \varepsilon$ for all $i \neq k$.*

Corollary 28 *:The following three properties of the set A are equivalent.*

(a) For every ε, there exists a finite ε-covering of the set A.
(b) For every ε, there exists a finite ε-net of the set A.
(c) For every ε, every ε-separated set is finite

The proof is straight forward. If $x_1, \cdots, x_h$ is an ε-net for A, then according to Definition 25 and 26 , there is also an ε-covering of A that consists of h sets; for the U_k we can take the closed balls with centers x_k and radius ε. Each compact set A contains a finite ε-net for each $\varepsilon > 0$. Hence, there is also a finite ε-covering for each $\varepsilon > 0$. Moreover, a compact set A can contains only finitely many ε-separable points. From now on , we suppose that A is compact.

It is clear, that for a given $\varepsilon > 0$, the number of n of sets U_k in a covering family (Def 25) depends on the family, but the minimal value of n, $N_\varepsilon(A) = \min n$ is an invariant of the set A, which depends only upon $\varepsilon > 0$. Also the maximum value of m, $M_\varepsilon(A) = \max m$, the points which are ε-separable (Def. 27) is an invariant of the set A.

We introduce the following two functions, which characterize the "massiveness" of the set A.

Definition 29 *:$N_\varepsilon(A)$ is the minimal number of sets in an ε-covering of A.*
$M_\varepsilon(A)$ is the maximal number of points in an ε-separated subset of A.
We take the logarithms to the base 2 of the functions, because $N_\varepsilon(A)$, $M_\varepsilon(A)$ are often so large that it can not be dealt with conveniently.

$H_\varepsilon(A) = \log_2 N_\varepsilon(A)$ is called the ε-entropy of the set A.
$C_\varepsilon(A) = \log_2 M_\varepsilon(A)$ is called the ε-capacity of the set A.

$H_\varepsilon(A)$, $C_\varepsilon(A)$ depend only on the compact metric space A itself and not on the larger space $R \supset A$. Both functions are monotone-increasing in A and decreasing in ε. We now introduce the main general results about entropies.

Theorem 30 *:For each $\varepsilon > 0$, and each compact set A,*

$$M_{2\varepsilon}(A) \leq N_\varepsilon(A) \leq M_\varepsilon(A)$$

and consequently

$$C_{2\varepsilon}(A) \leq H_\varepsilon(A) \leq C_\varepsilon(A).$$

Proof. *If $y_1, \cdots, y_m$ are 2ε-separable points of A, and $U_1, \cdots, U_n$ is an ε-covering of A, then $m \leq n$, for otherwise two different points y_i will be contained in the same set U_k. Thus $M_{2\varepsilon}(A) \leq N_\varepsilon(A)$; Finally, if $y_1, \cdots, y_m$ are ε-separable points of A, $m = M_\varepsilon(A)$, then they also form an ε-net for A; hence, $N_\varepsilon(A) \leq M_\varepsilon(A)$.* ■

Example 31 *If A is the interval $[a,b]$ with the euclidean metric, then $N_\varepsilon(A)$ is the smallest n with $n \geq \frac{b-a}{\varepsilon}$ and $M_{2\varepsilon}(A)$ is the largest m with $m \leq \frac{b-a}{2\varepsilon}$, hence*

$$N_\varepsilon(A) = \lceil \frac{b-a}{\varepsilon} \rceil, \quad M_{2\varepsilon}(A) = \lfloor \frac{b-a}{2\varepsilon} \rfloor.$$

This gives us $H_\varepsilon(A)$, $C_\varepsilon(A)$.

In most cases we cannot determine the entropy of A exactly, we may have to be content with finding it only up to a strong or weak equivalence. $H_\varepsilon(A) \sim \phi(\varepsilon)$ or $H_\varepsilon(A) \approx \phi(\varepsilon)$, where $\phi(\varepsilon)$ is a known function.

Definition 32 *Let R^s be the s-dimensional space with an arbitrary Banach metric $\rho(x,y) = \|x-y\|_p = \left| \sum_{i=1}^{s} |x_i - y_i|^p \right|^{\frac{1}{p}}$ with $p \in \{1,2,\ldots\}$ and $x, y \in R^s$.*

Note that it is generally known that an arbitrary Banach metric defines one and the same topology in R^s. Also the concept of measure does not essentially depend upon the choice of the metric.
A set $U \subset R$ of diameter 2ε is called *centerable* in R, if U has a center x_0, that is a point $x_0 \in R$ such that $\rho(x_0, x) \leq \varepsilon$ for all $x \in U$. If all sets U_k in Def.25 are centerable, then their centers form an ε-net for A. In what follows, we shall be mainly concerned with the determination of entropies of subsets of spaces of continuous functions $R = C[B]$ where B is a compact metric space. In this connection, the following is of interest: each compact A of $C[B]$ is centerable, this follows by Arzelă's theorem. Hence, if A is a compact subset of $C[B]$, then the minimal number of sets in an ε-covering of A is equal to the minimal number of points in an ε-net of A.

The entropy of a Cartesian product can be estimated if the entropies of all factors are known. Assume for example that $A = \prod_{k=1}^{s} A_k$ is a subset of the s-dimensional euclidean space R^s, and that each A_k is a subset of the kth coordinate axis. Then all possible products $\prod_{k=1}^{s} U_{i_k k}$ with $1 \leq i_k \leq N_\varepsilon(A_k)$ are $\prod_{k=1}^{s} N_\varepsilon(A_k)$ in number form an $\varepsilon-$covering of A. Hence, $N_\varepsilon(A)$ does not exceed the last product, and $H_\varepsilon(A) \leq \sum_{k=1}^{s} H_\varepsilon(A_k)$ and for the capacity we get $C_\varepsilon(A) \geq \sum_{k=1}^{s} C_\varepsilon(A_k)$.

We following propostion helps as to get a closer lower bound and an upper bound for the metric entropy $N_\varepsilon(A)$.

Proposition 33 *For bounded sets $A \subset R^s$ and under some non-essential (mild) conditions, there exist constants λ_s, μ_s such that*

$$N_\varepsilon(A) = \lambda_s |A| \varepsilon^{-s}$$
$$M_{2\varepsilon}(A) = \mu_s |A| \varepsilon^{-s}.$$

Here $|A|$ is the s-dimensional volume or the polycylinder of A, and λ_s, μ_s are some positive constants which depend on s, but not on A.(For the proof see Kolmogorov (1959), Theorem IX).

The precise values of λ_s and μ_s are not known. The determination of λ_s, μ_s is equivalent to the problems of finding the most economical covering in the space R^s by balls of radius 1, and finding the tightest packing of balls into this space. But for an arbitrary s, we can get bounds for λ_s, μ_s;

$$\frac{1}{2^s} \leq \mu_s \leq 1 \leq \lambda_s \leq 2^s, \tag{2}$$

(see Kolmogorov(1959), Theorem X, Chapter 3).

Corollary 34 *From the Theorem 30 and the Proposition 33 and equation (2) it follows that for a compact set $A \subset R^s$, there exist a lower bound and an upper bound for the metric entropy $N_\varepsilon(A)$ such that for all $\varepsilon \leq \min(\varepsilon_1, \varepsilon_2)$ (see Theorem 30), we have*

$$\frac{1}{2^s} |A| \varepsilon^{-s} \leq M_{2\varepsilon}(A) \leq N_\varepsilon(A) \leq 2^s |A| \varepsilon^{-s}. \tag{3}$$

Proof. By Inspection ■

For more details see Kolmogorv and Tihormirov "$\varepsilon-$entropy and $\varepsilon-$capacity of sets in function spaces" (1959).

Since we have (with Corollary 34) at least rough calculable lower and upper bounds for the metric entropy $N_\varepsilon(A)$, we are now able to compare the entropy of the set of functions spanned by arbitrary NNs with those of arbitrary NFNs.

5.2 The Entropy Calculation

Assuming that the data generating process $Z_t = (Y_t, X_t)$ is bounded (without loss of generality by the hypercube $I^{r+1} \equiv \times_{i=1}^{r+1}[0,1]$), we can derive bounds on the metric entropy of the set of functions by the various network types.

For concreteness we write the output of a q-hidden unit NN given input X_t as:

$$f^q(X_t, \delta^q) \equiv \beta_0 + \sum_{j=1}^{q} \beta_j \Psi(X_t w_j) \tag{5.2}$$

where $\delta^q = (\beta^q, w^q)$ is the $p \times 1$ $(p = q(r+2)+1)$ vector of network parameters. Let $\beta^q = (\beta_0, \ldots, \beta_q) \in \Re^{q+1}$, $w^q = (w_1, \ldots, w_q) \in \Re^{q(r+1)}$ with $w_j = (w_{j0}, \ldots, w_{jr})$ for $j = 1, \ldots, q$. Ψ is the sigmoid function and $X_t = (1, x_{t1}, \ldots, x_{tr}) \in I^{r+1}$.

Definition 35 *For any $q \in \{1, 2, \ldots\}$ and $\Delta \in \Re^+$, define $T(\Psi, q, \Delta)$ as*

$$T(\Psi, q, \Delta) \equiv \left\{ \theta \in \Theta : \theta(\cdot) = f^q(\cdot, \delta^q), \sum_{j=1}^{q} |\beta_j| \leq \Delta, \sum_{j=1}^{q} \sum_{i=0}^{r} |w_{ji}| \leq q\Delta \right\},$$

is the set of output functions of all NNs with q hidden units having activation functions Ψ, and with weights satisfying a particular restriction on their sum norm , indexed by Δ. For latter condition , it suffices that $\sum_{i=0}^{r} |w_{ji}| \leq \Delta$.

We write the output of a q-hidden unit feedforward NFN with l-neural network units and m-fuzzy network units given input X_t as

$$g^q(X_t, \delta^q) \equiv \beta_0 + \sum_{j=1}^{l} \beta_j \Psi(X_t w_j) + \sum_{i=1}^{m} \beta_i \left(\prod_{k=1}^{r} \Psi(x_{tk} w_{ik1} + w_{ik0}) \right), \tag{5.3}$$

where $l + m = q, \delta^q = (\beta^q, w^q)$ is the $p \times 1$ $(p = q + l(r+1) + 2rm + 1)$ vector of network parameters. Let $\beta^q = (\beta_0, \ldots, \beta_q) \in \Re^{q+1}$, $w^q = (w^l, w^m) \in \Re^{l(r+1)+2rm}, w^l = (w_1, \ldots, w_l)$ with $w_j = (w_{j0}, \ldots, _{jr})$ for $j = 1, \ldots, l$ $.w^m = (w_1, \ldots, w_m)$ with $w_i = (w_{i10}, w_{i11}, \ldots, w_{ir0}, w_{ir1},)$ for $i = 1, \ldots, m$. Ψ is the sigmoid function and $X_t = (1, x_{t1}, \ldots, x_{tr}) \in I^{r+1}$.

Definition 36 *For any $q \in \{1, 2, \ldots\}$ and $\Delta \in \Re^+$, define a $G(\Psi, q, \Delta)$ as*

$$G(\Psi, q, \Delta) \equiv \left\{ \begin{array}{c} \theta \in \Theta : \theta(\cdot) = g^q(\cdot, \delta^q), \sum_{j=1}^{q} |\beta_j| \leq \Delta, \\ \sum_{j=1}^{l} \sum_{i=0}^{r} |w_{ji}| + \sum_{k=1}^{m} \sum_{t=1}^{r} \sum_{s=1}^{2} |w_{kts}| \leq (l+2m)\Delta \end{array} \right\},$$

is the set of output functions of all NFNs with l-neural network units and m-fuzzy network units having activation functions Ψ , and with weights satisfying a particular restriction on their sum norm, indexed by Δ.

The bounds for the metric entropy of a NFN and a NN is given by the following result, which is a generalization by Gottschling (Theorem 4, 1997) and by White (Lemma 4.3, Chapter 10, 1992).

Theorem 37 *Let $q, l, m, r \in \{1, 2, \ldots\}$ and $\Delta, L, K \in \Re^+$ be given. Let ρ_∞ denote the uniform metric, $\rho_\infty(\theta_1, \theta_2) = \sup_{x \in I^r} |\theta_1(x) - \theta_2(x)|$ and Ψ be the sigmoid function,*

- *α) Let $\theta_1, \theta_2 \in G(\Psi, q, \Delta)$ and let $H_\varepsilon^{NFN}(\Psi, q, \Delta)$ denote the metric entropy of $G(\Psi, q, \Delta)$ with respect to ρ_∞ . Then for all $\varepsilon > 0$ sufficiently small*

$$H_\varepsilon^{NFN}(\Psi, q, \Delta) \leq (q + p + 1)\left(\log_2\left(\frac{1}{\varepsilon}\right) + \log_2(4s\Delta)\right) + p\log_2(l + 2m)$$

$$H_\varepsilon^{NFN}(\Psi, q, \Delta) \geq (q + p + 1)\left(\log_2\left(\frac{1}{\varepsilon}\right) + \log_2\left(\frac{s\Delta}{2}\right)\right) + p\log_2(l + 2m)$$

with $s = (1 + \Delta L(r + 1) + 2\Delta Lr\,(1 + 2K))$, $p = (l(r + 1) + 2rm)$.

- *β) Let $\theta_1, \theta_2 \in T(\Psi, q, \Delta)$ and let $H_\varepsilon^{NN}(\Psi, q, \Delta)$ denote the metric entropy of $T(\Psi, q, \Delta)$ with respect to ρ_∞ . Then for all $\varepsilon > 0$ sufficiently small,*

$$H_\varepsilon^{NN}(\Psi, q, \Delta) \leq p\left(\log_2\left(\frac{1}{\varepsilon}\right) + \log_2(4\Delta(1 + \Delta L(r + 1)))\right) + q(r+1)\log_2(q)$$

$$H_\varepsilon^{NN}(\Psi, q, \Delta) \geq p\left(\log_2\left(\frac{1}{\varepsilon}\right) + \log_2\left(\frac{\Delta + L(r + 1)\Delta^2}{2}\right)\right) + q(r + 1)\log_2(q)$$

with $p = q(r + 2) + 1$.

Proof. α)

We construct an ε-net for the $G(\Psi, q, \Delta)$: (Recall that $G_\varepsilon = \{g_1, \ldots, g_h\}$ is an $\varepsilon-$net for G if for each $\varphi \in G$ there is at least one element $g_k \in G_\varepsilon$ such that $\rho_\infty(g_k, \varphi) \leq \varepsilon$, then the metric entropy of G is at most $\log h$).

Let $\eta > 0$ be given, and let $B_\eta \equiv \{b_k \in B, k = 1, \ldots, s\}$,

$C_\eta \equiv \{c_k \in \Gamma, k = 1, \ldots, n\}$ be η-nets for $B = \{\beta : \|\beta\| \leq \Delta\} \subset \Re^{q+1}$ and

$$\Gamma = \left\{ \begin{array}{c} w : \|w\| \leq (l + 2m)\Delta \,|\forall w, \lambda \in \Gamma \text{ and } \forall i \in \{1, \ldots, n\} \;\exists\; 1 \leq K < \infty \;\; , \\ \text{s.t.} \max_{j \in \{1, \ldots, n\}} |\lambda_i - w_j| \leq K\,|\lambda_i - w_i| \end{array} \right\},$$

$\Gamma \subset \Re^n$ with $n = l(r+1) + 2rm$. (w_j denotes the j-th component of w, while $\|\cdot\|$ here denotes the 1-norm i.e. $\|x\| = \sum_{i=1}^{v} |x_i|$ in $\Re^v$ of whatever dimension), $l + m = q$.

For each Ψ and Δ , $G(\Psi, q, \Delta)$ is a compact set, as it is the continuous image of the compact set $B \times \Gamma$, that is a necessary condition to the existence of an ε-net for the $G(\Psi, q, \Delta)$. Let $D_\eta = B_\eta \times C_\eta$ and $\hat{G}_\eta = \{t \in G(\Psi, q, \Delta) : d \in D_\eta\}$. We choose η so that $\hat{G}_\eta$ is an ε-net for $G(\Psi, q, \Delta)$:

Let $\theta \in G(\Psi, q, \Delta)$ be arbitrary, with corresponding parameters $\delta = (\beta, w) \in B \times \Gamma$. Then there exists $d = (b, c) \in D_\eta$ such that $\|\beta - b\| \leq \eta$ and $\|w - c\| \leq \eta$, since B_η and C_η are η-nets for B and Γ.

Let t be the element of $\hat{G}_\eta$ with corresponding to d such that:

$$\rho_\infty(\theta, t) = \sup_{x \in I^r} |\theta(x) - t(x)| = \sup_{x \in I^r} |g^q(x, \delta) - \hat{g}^q(x, d)| < \varepsilon,$$

We now try to find out how η have to look like, such that $\hat{G}_\eta$ is an ε-net for $G(\Psi, q, \Delta)$, consider:

$$\left| \begin{array}{c} |\theta(x) - t(x)| = |g^q(x, \delta) - \hat{g}^q(x, d)| \\ = \beta_0 + \sum_{j=1}^{l} \beta_j \Psi(x w_j) + \sum_{i=1}^{m} \beta_i (\prod_{k=1}^{r} \Psi(x_k w_{ik1} + w_{ik0})) \\ - b_0 - \sum_{j=1}^{l} b_j \Psi(x c_j) - \sum_{i=1}^{m} b_i (\prod_{k=1}^{r} \Psi(x_k c_{ik1} + c_{ik0})) \end{array} \right|$$

The triangle inequality gives

$$\leq \left| \beta_0 - b_0 + \sum_{j=1}^{l} (\beta_j - b_j) \Psi(x w_j) + \sum_{i=1}^{m} (\beta_i - b_i) (\prod_{k=1}^{r} \Psi(x_k w_{ik1} + w_{ik0})) \right| \quad (*)$$

$$+ \left| \sum_{j=1}^{l} b_j (\Psi(x w_j) - \Psi(x c_j)) \right| \quad (**)$$

$$+ \left| \sum_{i=1}^{m} b_i ((\prod_{k=1}^{r} \Psi(x_k w_{ik1} + w_{ik0})) - (\prod_{k=1}^{r} \Psi(x_k c_{ik1} + c_{ik0}))) \right| \quad (***)$$

Using the fact that Ψ is bounded (set the bound to unity) and the triangle inequality, then the first term (ref:(*)) becomes to:

$$\left| \beta_0 - b_0 + \sum_{j=1}^{l} (\beta_j - b_j) \Psi(x w_j) + \sum_{i=1}^{m} (\beta_i - b_i) (\prod_{k=1}^{r} \Psi(x_k w_{ik1} + w_{ik0})) \right|$$

$$\leq \left| \beta_0 - b_0 + \sum_{j=1}^{l} (\beta_j - b_j) + \sum_{i=1}^{m} (\beta_i - b_i) \right|$$
$$\leq \sum_{i=0}^{q} |\beta_i - b_i| = \|\beta - b\| \leq \eta \text{ (see assumption)} \tag{4}$$

For the second term(ref:(**)) the triangle inequality yields:

$$\left| \sum_{j=1}^{l} b_j (\Psi(xw_j) - \Psi(xc_j)) \right| \leq \sum_{j=1}^{l} |b_j| \, |\Psi(xw_j) - \Psi(xc_j)|$$

$$\leq \left(\sum_{j=1}^{l} |b_j| \right) \left(\sum_{j=1}^{l} |\Psi(xw_j) - \Psi(xc_j)| \right) \leq \Delta \sum_{j=1}^{l} |\Psi(xw_j) - \Psi(xc_j)| .$$

The Lipschitz condition imposed on Ψ (Lipschitz condition is fulfil since $|\Psi(\cdot)| \leq 1$) and the triangle inequality give

$$\begin{aligned}
\Delta \sum_{j=1}^{l} |\Psi(xw_j) - \Psi(xc_j)| &\leq \Delta \sum_{j=1}^{l} L \, |xw_j - xc_j| = \Delta L \sum_{j=1}^{l} \left| \sum_{i=0}^{r} x_i (w_{ij} - c_{ij}) \right| \\
&\leq \Delta L \sum_{j=1}^{l} \sum_{i=0}^{r} |x_i| \, |w_{ij} - c_{ij}| \\
&\leq \Delta L \sum_{j=1}^{l} \left(\sum_{i=0}^{r} |x_i| \right) \left(\sum_{i=0}^{r} |w_{ij} - c_{ij}| \right) \\
&\leq \Delta L \sum_{j=1}^{l} (r+1) \left(\sum_{i=0}^{r} |w_{ij} - c_{ij}| \right) \text{ since } x \in I^{r+1} \\
&\leq \Delta L (r+1) \sum_{j=1}^{l} \left(\sum_{i=0}^{r} |w_{ij} - c_{ij}| \right) \\
&= \Delta L (r+1) \sum_{j=1}^{l} (w_j - c_j) \leq \Delta L (r+1) \, \|w - c\| \\
&\leq \Delta L (r+1) \eta
\end{aligned} \tag{5}$$

For the third term(ref:(***)), the triangle inequality yields:

$$
\begin{aligned}
&\left|\sum_{i=1}^{m} b_i((\prod_{k=1}^{r}\Psi(x_k w_{ik1}+w_{ik0}))-(\prod_{k=1}^{r}\Psi(x_k c_{ik1}+c_{ik0})))\right| \\
\leq &\sum_{i=1}^{m}|b_i|\left|((\prod_{k=1}^{r}\Psi(x_k w_{ik1}+w_{ik0}))-(\prod_{k=1}^{r}\Psi(x_k c_{ik1}+c_{ik0}))\right| \\
\leq &\Delta\sum_{i=1}^{m}\left|((\prod_{k=1}^{r}\Psi(x_k w_{ik1}+w_{ik0}))-(\prod_{k=1}^{r}\Psi(x_k c_{ik1}+c_{ik0}))\right| \\
\leq &\Delta\sum_{i=1}^{m}\left||\Psi(x_u w_{iu1}+w_{iu0})|^r-|\Psi(x_t c_{it1}+c_{it0})|^r\right|, \qquad (6)
\end{aligned}
$$

where

$$
\begin{aligned}
&\left|\prod_{k=1}^{r}\Psi(x_k w_{ik1}+w_{ik0})\right| \leq \left||\Psi(x_u w_{iu1}+w_{iu0})|^r\right| \text{ with} \\
&|\Psi(x_u w_{iu1}+w_{iu0})| = \max_k |\Psi(x_k w_{ik1}+w_{ik0})|
\end{aligned}
$$

and where

$$
\begin{aligned}
&\left|\prod_{k=1}^{r}\Psi(x_k c_{ik1}+c_{ik0})\right| \geq \left||\Psi(x_t c_{it1}+c_{it0})|^r\right| \text{ with} \\
&|\Psi(x_t c_{it1}+c_{it0})| = \min_k |\Psi(x_k c_{ik1}+c_{ik0})|.
\end{aligned}
$$

(6) becomes:

$$
\begin{aligned}
&\Delta\sum_{i=1}^{m}\left||\Psi(x_u w_{iu1}+w_{iu0})|^r-|\Psi(x_t c_{it1}+c_{it0})|^r\right| \\
=\Delta&\sum_{i=1}^{m}\left|\begin{array}{c}(|\Psi(x_u w_{iu1}+w_{iu0})|-|\Psi(x_t c_{it1}+c_{it0})|)\cdot \\ \left(\begin{array}{c}|\Psi(x_u w_{iu1}+w_{iu0})|^{r-1}+|\Psi(x_u w_{iu1}+w_{iu0})|^{r-2}|\Psi(x_t c_{it1}+c_{it0})| \\ +\ldots+|\Psi(x_u w_{iu1}+w_{iu0})|\,|\Psi(x_t c_{it1}+c_{it0})|^{r-2}+|\Psi(x_t c_{it1}+c_{it0})|^{r-1}\end{array}\right)\end{array}\right|.
\end{aligned}
$$

Since all $a,b\in\Re$ fulfil $(a^r-b^r)=(a-b)(a^{r-1}+a^{r-2}b+\ldots+b^{r-1}),\forall r\in\{1,2,\ldots\}$ we get:

$$\leq \Delta \sum_{i=1}^{m} \left| \left(\left| \Psi(x_u w_{iu1} + w_{iu0}) \right| - \left| \Psi(x_t c_{it1} + c_{it0}) \right| \right) r \right| \text{ by the boundedness of } \Psi, |\Psi(\cdot)| \leq 1$$

$$\leq \Delta r \sum_{i=1}^{m} L \left| x_u w_{iu1} + w_{iu0} - x_t c_{it1} - c_{it0} \right| \text{ by the Lipschitz property of } \Psi$$

$$\leq \Delta r L \sum_{i=1}^{m} \left| \sum_{j=0}^{1} x_j (w_{iuj} - c_{itj}) \right| \text{ with } x_0 = 1 \text{ and } x_1 = \max(x_u, x_t)$$

$$\leq \Delta r L \sum_{i=1}^{m} \left| \sum_{j=0}^{1} |x_j| \, |w_{iuj} - c_{itj}| \right| \leq \Delta r L \sum_{i=1}^{m} \left| \left(\sum_{j=0}^{1} |x_j| \right) \left(\sum_{j=0}^{1} |w_{iuj} - c_{itj}| \right) \right|$$

$$\leq \Delta r L 2 \sum_{i=1}^{m} \left| \left(\sum_{j=0}^{1} |w_{iuj} - c_{itj}| \right) \right| \text{ since } |x_j| \in [0, 1]$$

$$= \Delta r L 2 \sum_{i=1}^{m} \left| \left(\sum_{j=0}^{1} |w_{iuj} - c_{ikj} + c_{jki} - w_{ikj} + w_{ikj} - c_{itj}| \right) \right|$$

$$\leq \Delta r L 2 \sum_{i=1}^{m} \left| \left(\sum_{j=0}^{1} |w_{iuj} - c_{ikj}| + \sum_{j=0}^{1} |c_{ikj} - w_{ikj}| + \sum_{j=0}^{1} |w_{ikj} - c_{itj}| \right) \right|$$

The first and the last sum in the brackets can be bounded because of the assumption

$|w_{iuj} - c_{ikj}| \leq \forall k \max_{\forall u} |w_{iuj} - c_{ikj}| \leq K \, |w_{jki} - c_{jki}|$, s.t.:

$$\sum_{i=1}^{m} \sum_{j=0}^{1} |w_{iuj} - c_{ikj}| \leq K \sum_{i=1}^{m} \sum_{j=0}^{1} |w_{ikj} - c_{ikj}|$$

$$\text{and } \sum_{i=1}^{m} \sum_{j=0}^{1} |w_{ikj} - c_{itj}| \leq K \sum_{i=1}^{m} \sum_{j=0}^{1} |w_{ikj} - c_{ikj}| \, .$$

Thus our inequality becomes

$$\leq \Delta r L 2 \, (K + 1 + K) \left(\sum_{i=1}^{m} \sum_{j=0}^{1} |w_{ikj} - c_{ikj}| \right)$$

$$\leq \Delta r L 2 \, (K + 1 + K) \left(\sum_{i=1}^{m} \sum_{k=1}^{r} \sum_{j=0}^{1} |w_{ikj} - c_{ikj}| \right)$$

$$\leq \Delta r L 2 \, (1 + 2K) \, \|w - c\|$$

$$\leq \Delta r L 2 \, (1 + 2K) \, \eta, \tag{7}$$

because $\|w-c\| \leq \eta$.

It follows from (4),(5) and (7) that

$$\rho_\infty(\theta,t) = \sup_{x\in I^r} |\theta(x)-t(x)| \leq \eta + \Delta L(r+1)\eta + \Delta r L 2\,(1+2K)\,\eta = \varepsilon$$

provided that we choose

$$\eta = \frac{\varepsilon}{(1+\Delta L(r+1)+2\Delta Lr\,(1+2K))}.$$

Because θ is arbitrary and $t \in \hat{G}_\eta$, $G_\varepsilon \equiv \hat{G}_{\varepsilon/[1+\Delta L(r+1)+2\Delta Lr(1+2K)]}$ is an ε-net for $G(\Psi,q,\Delta)$.

Let $\#$ be the cardinality operator. Because $\#\hat{G}_\eta = \#D_\eta$ and $D_\eta = B_\eta \times C_\eta$ we have $\#\hat{G}_\eta = (\#B_\eta)(\#C_\eta)$.
Repetition $B_\eta \equiv \{b_k \in B, k = 1,\ldots,s\}$,be η-net for $B = \{\beta : \|\beta\| \leq \Delta\} \subset \Re^{q+1}$ and $C_\eta \equiv \{c_k \in \Gamma, k = 1,\ldots,n\}$, be η-net for $\Gamma = \{w : \|w\| \leq (l+2m)\Delta\} \subset \Re^{l(r+1)+2rm}$. Form the Corollary 34 in Section 5.1 and elementary geometrical arguments it follows that for all ε (hence η) sufficiently small

$$\left(\frac{\Delta}{2\eta}\right)^{q+1} \leq \#B_\eta \leq \left(\frac{4\Delta}{\eta}\right)^{q+1},$$

$$\left(\frac{(l+2m)\Delta}{2\eta}\right)^{l(r+1)+2rm} \leq \#C_\eta \leq \left(\frac{4(l+2m)\Delta}{\eta}\right)^{l(r+1)+2rm}.$$

Note that the volume of the polycylinder B_η is bigger than Δ^{q+1} and smaller than $(2\Delta)^{q+1}$.

Therefore with $s = (1+\Delta L(r+1)+2\Delta Lr\,(1+2K))$, $\eta = \left(\frac{\varepsilon}{s}\right), p = (l(r+1)+2rm)$ we get:

$$\begin{aligned}
\log_2(\#G_\varepsilon) &\leq (q+1)\log_2\left(\frac{4\Delta}{\eta}\right) + (l(r+1)+2rm)\log_2\left(\frac{4(l+2m)\Delta}{\eta}\right) \\
&= (q+p+1)\log_2\left(\frac{4\Delta}{\eta}\right) + p\log_2(l+2m) \\
&= (q+p+1)\log_2\left(\frac{4\Delta(1+\Delta L(r+1)+2\Delta Lr\,(1+2K)}{\varepsilon}\right) + p\log_2(l+2m) \\
&= (q+p+1)\left(\log_2\left(\frac{1}{\varepsilon}\right) + \log_2(4s\Delta)\right) + p\log_2(l+2m)
\end{aligned}$$

$$\log_2(\#G_\varepsilon) \geq (q+1)\log_2\left(\frac{\Delta}{2\eta}\right) + (l(r+1)+2rm)\log_2\left(\frac{(l+2m)\Delta}{2\eta}\right)$$
$$= (q+p+1)\left(\log_2\left(\frac{1}{\varepsilon}\right) + \log_2\left(\frac{s\Delta}{2}\right)\right) + p\log_2(l+2m).$$

Since $H_\varepsilon^{NFN}(\Psi, q, \Delta) = \log_2(\#G_\varepsilon)$ we have the result of α). ■

Proof. β)

The proof is analog to part α) , set $l = q$ and $m = 0$. It follows from the corresponding bounds of (4) and (5) that

$$\rho_\infty(\varphi, h) = \sup_{x \in I^r} |\varphi(x) - h(x)| \leq (1 + \Delta L(r+1))\eta = \varepsilon$$

provided that we choose

$$\eta = \frac{\varepsilon}{(1 + \Delta L(r+1))}.$$

Because $\varphi \in T(\Psi, q, \Delta)$ is arbitrary and $h \in \widehat{T}_\eta$, $T_\varepsilon \equiv \widehat{T}_{\varepsilon/[1+\Delta L(r+1)]}$ is an ε -net for $T(\Psi, q, \Delta)$.
The corresponding $B_\eta \equiv \{b_k \in B, k = 1, \ldots, s\}$, be η-net for $B = \{\beta : \|\beta\| \leq \Delta\} \subset \Re^{q+1}$ and $C_\eta \equiv \{c_k \in \Gamma, k = 1, \ldots, n\}$, be η-net for $\Gamma = \{w : \|w\| \leq q\Delta\} \subset \Re^{q(r+1)}$. Form the Corollary 34 in Section 5.1 and elementary geometrical arguments it follows that for all ε (hence η) sufficiently small we get analogous to part α) the bounds:

$$\left(\frac{\Delta}{2\eta}\right)^{q+1} \leq \#B_\eta \leq \left(\frac{4\Delta}{\eta}\right)^{q+1}$$

$$\left(\frac{q\Delta}{2\eta}\right)^{q(r+1)} \leq \#C_\eta \leq \left(\frac{4q\Delta}{\eta}\right)^{q(r+1)}$$

Therefore with $p = q(r+2)+1$ the result follows:

$$H_\varepsilon^{NN}(\Psi, q, \Delta) \leq (q+1)\log_2\left(\frac{4\Delta}{\eta}\right) + q(r+1)\log_2\left(\frac{4q\Delta}{\eta}\right)$$
$$= p\log_2\left(\frac{4\Delta}{\eta}\right) + q(r+1)\log_2(q)$$
$$= p\log_2\left(\frac{4\Delta(1+\Delta L(r+1))}{\varepsilon}\right) + q(r+1)\log_2(q)$$
$$= p\left(\log_2\left(\frac{1}{\varepsilon}\right) + \log_2(4\Delta(1+\Delta L(r+1)))\right) + q(r+1)\log_2(q)$$

$$H_{\varepsilon}^{NN}(\Psi,q,\Delta) \geq p\left(\log_2\left(\frac{1}{\varepsilon}\right)+\log_2\left(\frac{\Delta+L(r+1)\Delta^2}{2}\right)\right)+q(r+1)\log_2(q).$$

■

As mentioned before, to compare the "size" of a set of functions $G(\Psi,q,\Delta)$ spanned by an NFN to the size of the set of functions $T(\Psi,q,\Delta)$ spanned by an NN, it suffices to compare their respective metric entropies for different functional specifications. Since q (= number of hidden notes), l (= number of neural hidden notes in $G(\Psi,q,\Delta)$), m(=number of fuzzy hidden notes in $G(\Psi,q,\Delta)$) and r(= number of input variables) are all arbitrary non negative integers and bigger than unity, one can compare the metric entropies of all functional specifications by simply comparing the metric entropy of G and T for arbitrary values of q, m, l, and r.

The next Theorem shows that for sufficiently big $\Delta > 0$ the "size" of $G(\Psi,q,\Delta)$ is bigger than the "size" of $T(\Psi,q,\Delta)$.

Theorem 38 *Let $q,l,m,r \in \{1,2,\ldots\}$ and $\Delta,L,K,c \in \Re^+$, then for all $\varepsilon > 0$ and all $\Delta \geq \left(\frac{8}{L(r(4K-5)-7)}\right)$, with $K > \frac{5}{4}$ and $r = \left(\left[\frac{7}{(4K-5)}\right]+1\right)$ ($[\cdot]$ denotes the Gauss brackets) we have*

$$H_{\varepsilon}^{NFN}(\Psi,q,\Delta) > H_{\varepsilon}^{NN}(\Psi,q,\Delta)$$

with $l+m=q$ and $l=cm$.

Proof. *From Theorem 37 follows:*

$$H_{\varepsilon}^{NFN}(\Psi,q,\Delta) \geq \log_2\left[\left(\frac{\Delta s}{2\varepsilon}\right)^{q+1}\left(\frac{(l+2m)\Delta s}{2\varepsilon}\right)^{l(r+1)+2rm}\right],$$

$$\log_2\left[\left(\frac{4\Delta t}{\varepsilon}\right)^{q+1}\left(\frac{4q\Delta t}{\varepsilon}\right)^{q(r+1)}\right] \geq H_{\varepsilon}^{NN}(\Psi,q,\Delta)$$

with $s = 1+\Delta L(r+1)+2\Delta Lr\left(1+2K\right)$, $t = 1+\Delta L(r+1)$.
We have to show that:

$$\log_2\left[\left(\frac{\Delta s}{2\varepsilon}\right)^{q+1}\left(\frac{(l+2m)\Delta s}{2\varepsilon}\right)^{l(r+1)+2m}\right] > \log_2\left[\left(\frac{4\Delta t}{\varepsilon}\right)^{q+1}\left(\frac{4q\Delta t}{\varepsilon}\right)^{q(r+1)}\right] \Leftrightarrow$$
$$\left[\left(\frac{\Delta s}{2\varepsilon}\right)^{q+1}\left(\frac{(l+2m)\Delta s}{2\varepsilon}\right)^{l(r+1)+2rm}\right] > \left[\left(\frac{4\Delta t}{\varepsilon}\right)^{q+1}\left(\frac{4q\Delta t}{\varepsilon}\right)^{q(r+1)}\right].$$

Since $q = l + m$ we get:

$$\left(\frac{\Delta s}{2\varepsilon}\right)^{q+1}\left(\frac{\Delta s}{2\varepsilon}\right)^{l(r+1)+2rm}(l+2m)^{l(r+1)+2rm} > \left(\frac{4\Delta t}{\varepsilon}\right)^{q+1}\left(\frac{4\Delta t}{\varepsilon}\right)^{q(r+1)}(l+m)^{q(r+1)} \Leftrightarrow$$
$$\left(\frac{\Delta s}{2\varepsilon}\right)^{q+1}\left(\frac{\Delta s}{2\varepsilon}\right)^{l(r+1)+2rm}(l+2m)^{l(r+1)+2rm} > \left(\frac{4\Delta t}{\varepsilon}\right)^{q+1}\left(\frac{4\Delta t}{\varepsilon}\right)^{(l+m)(r+1)}(l+m)^{(l+m)(r+1)}.$$

We show the validity of the following three inequations to get our main result:

$$\left(\frac{\Delta s}{2\varepsilon}\right)^{q+1} > \left(\frac{4\Delta t}{\varepsilon}\right)^{q+1} \tag{5.4}$$

$$\left(\frac{\Delta s}{2\varepsilon}\right)^{l(r+1)+2rm} > \left(\frac{4\Delta t}{\varepsilon}\right)^{(l+m)(r+1)} \tag{5.5}$$

$$(l+2m)^{l(r+1)+2rm} > (l+m)^{(l+m)(r+1)} \tag{5.6}$$

For 5.4 we get:

$$\begin{aligned}
\left(\frac{\Delta s}{2\varepsilon}\right)^{q+1} &> \left(\frac{4\Delta t}{\varepsilon}\right)^{q+1} \Leftrightarrow \\
\left(\frac{\Delta s}{2\varepsilon}\right) &> \left(\frac{4\Delta t}{\varepsilon}\right) \Leftrightarrow \\
s &> 8t \Leftrightarrow \\
1 + \Delta L(r+1) + 2\Delta Lr\,(1+2K) &> 8\,(1 + \Delta L(r+1)) \Leftrightarrow \\
\Delta L(r+1) + 2\Delta Lr\,(1+2K) &> 8\Delta L(r+1) + 7 \Leftrightarrow \\
\Delta L\,(2r\,(1+2K) - 7(r+1)) &> 7 \Leftrightarrow \\
\Delta L\,(r\,(4K-5) - 7) &> 7 \Leftrightarrow \\
\left(\frac{8}{L(r(4K-5)-7)}\right) L\,(r\,(4K-5) - 7) &> 7 \Leftrightarrow \\
8 &> 7,
\end{aligned}$$

with $\Delta \geq \left(\frac{8}{L(r(4K-5)-7)}\right)$ (see assumption above).

The inequation 5.5 is true since the exponent $l(r+1) + 2rm \geq (l+m)(r+1)$ for all r, and the base is $\left(\frac{\Delta s}{2\varepsilon}\right) > \left(\frac{4\Delta t}{\varepsilon}\right)$ (see above (5.4)). Analogous the inequation 5.6 is also true since the exponent $l(r+1) + 2rm \geq (l+m)(r+1)$ for all r, and the basis is $(l+2m) > (l+m)$. The result follows. ■

This result gives us the certainty that there are circumstances in which the "size" of the function set, spanned by an NFN is "bigger" than those set of a correspondent NN. Hence, with an NFN we maybe reaching into more dimensions

at given levels of precision for any given number of nonlinear units than if we just use NNs alone. However, the above results which indicate a potential advantage of applying NFNs is based on an average over all possible functional specifications and is therefore only of theoretical value. Since it is difficult to argue for the relevance of one functional specification over another, we analyze the applicability of the NFNs to a popular problem in finance.

6 Option Pricing with Neuro Fuzzy Networks

Two general approaches can be distinguished in the treatment of option pricing in finance. On one side there exists the analytical approach, pioneered in the seminal articles by Black and Scholes (1973) and Merton (1973), in which closed form option pricing formulas were obtained. The basis for these results are numerous assumptions, such as a no-arbitrage condition, normal returns, fixed volatility and interest rates. See for example the models of Merton (1973, 1976) and Cox and Ross (1976). Many extensions and refinements have been proposed and developed, however the lack of empirical substantiation of many of these models has led to a second school of thought. Instead of starting with many assumptions, one considers the option market pricing mechanism to be a complex unknown function of the same inputs as the theory, and then extract an approximate functional form by fitting observed prices via a multivariate nonlinear model. This approach was successfully pioneered in the context of neural networks by Hutchinson et al. (1994). Such a model, once estimated can be used to calculate prices and hedge parameters analogously to its theoretical counterpart. Since option pricing theory is based upon highly nonlinear relations between the option price and its determining variables, a flexible functional form is not only sufficient but also necessary to capture empirical pricing mechanisms since linear models are a failure in this context. Due to the ability of NNs, FNs and NFNs to approximate arbitrary continuous as well as large sets of measurable functions, they seem a priori well suited for this task.

We present a NFN developed to estimate the market prices at closing of S&P 500 options using transactions data for the period January 1, 1986 to December 31, 1989.

In this section we apply the algorithm of section 4 to build a NFN model which is then to fit to the prices of SPX call options. Comparisons with regard to NNs and the Black-Scholes model are made subsequently.

6.1 Option Pricing

In this study we use the Black-Scholes model as a reference point. The derivation of the option price formula relies on the following assumptions: asset prices follow a geometric Brownian motion; mean returns and volatilities are constant over time; interest rates are both constant over time and equal for all maturities; trading occurs continuously on frictionless markets and no arbitrage opportunities exist. From these assumptions one can derive the following formula for the price of a European call option written on a non-dividend paying stock:

$$C_{BS}(t) = SN(d_1) - X\exp(-r(T-t))N(d_2), \qquad (*)$$

where

$$d_1 = \frac{\ln(\frac{S}{X}) + (r + \frac{\delta^2}{2})(T-t)}{\sigma\sqrt{T-t}},$$

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

with

S = Price of the underlying stock

X = Strike price of the option

σ = Volatility of the stock returns

r = Interest rate

$T - t$ = Time to maturity of the option contract

and $N(x)$ is the cumulative distribution function of the standard normal distribution.

The equation (*) shows that the call option price C depends on five variables. Merton (1973, Theorem 9) shows that the option price is linear homogeneous of order one in X and S for every "normal" pricing model, if the return distribution of the underlying stock does not depend on the stock price level. As this condition is valid for the Black-Scholes model, the number of input variables can be reduced to four by treating C/X as a function of $S/X, \sigma, r$ and $(T-t)$, now the pricing formula becomes:

$$\frac{C_{BS}(t)}{X} = \frac{S}{X}N(d_1) - \exp(-r(T-t))N(d_2).$$

6.2 The Data

In our study, we use transactions data on call options (SPX) issued on the Standard & Poor's 500 index (S&P 500). The index is comprised of 500 stocks from a broad range of industries. The component stocks are weighted according to the total market value of their outstanding shares.

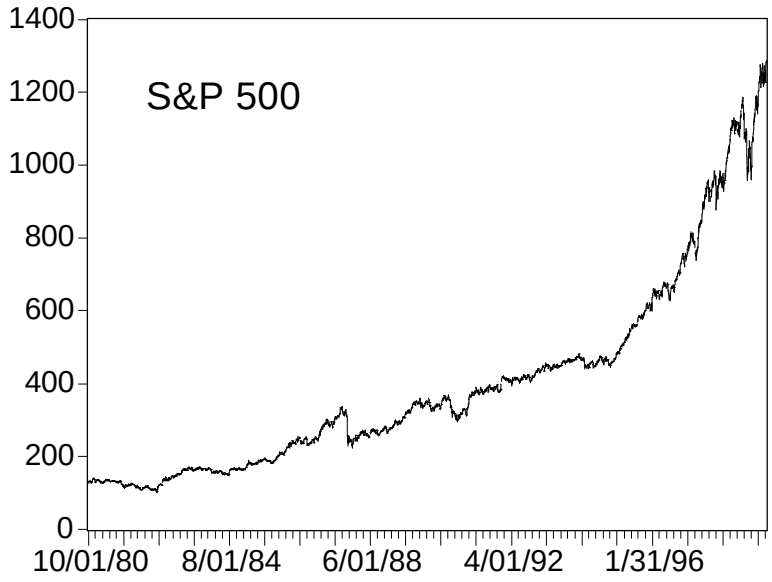


Fig.: 6.1

SPX Option prices are quoted in points. One point equals \$100, the minimum tick size is 1/16 (\$6.25) points and for all other series, 1/8 (\$12.50). The option's exercise prices have fixed increments of five points and 25-point intervals for far months. The expiration date is the Saturday immediately following the third Friday of the expiration month and the expiration months are the three near-term months followed by three additional months from the March quarterly cycle (March, June, September and December). The maximum time to maturity of an option contract is nine months.

Our data set contains data on SPX calls traded on the Chicago Board Options Exchange (CBOE) from January 1986 the end of December 1989. Each transaction record contains the option price C the exercise price X and the time of maturity $(T-t)$.

Since this data set is too large it had to be restricted. For the empirical investigation we remove uninformative and non-representative option records employing exclusion criteria along the lines of Rubinstein (1985) and Xu and Taylor (1994) as follows:

1. Option that are traded at less than \$100 (1 point).
2. Options for which the lower boundary condition is violated:

$$C < S - X\exp(-r(T-t))$$

3. Option which are deep-in or deep-out-of the money: $\frac{S}{X} < 0.8$ or $\frac{S}{X} > 1.2$.

When the option value is very low then it leads to high percentage deviations between observed and theoretical prices. Thus, criterion 1 excludes options with

low prices. The second criterion excludes options whose prices are not consistent with a no-arbitrage pricing model. With criterion 3 options that have almost no informational content are also thrown out, since the trading volume is very low for these. Our resulting data set then consists of 5056 observations.

To obtain the theoretical price according to the Black-Scholes formula as well as for inputs to the networks, we use the underlying S&P 500 (S), the riskfree interest rate (r) and the return volatility (σ). Our interest rate data consists of average daily London interbank offer rates (LIBOR) for a three month maturity.

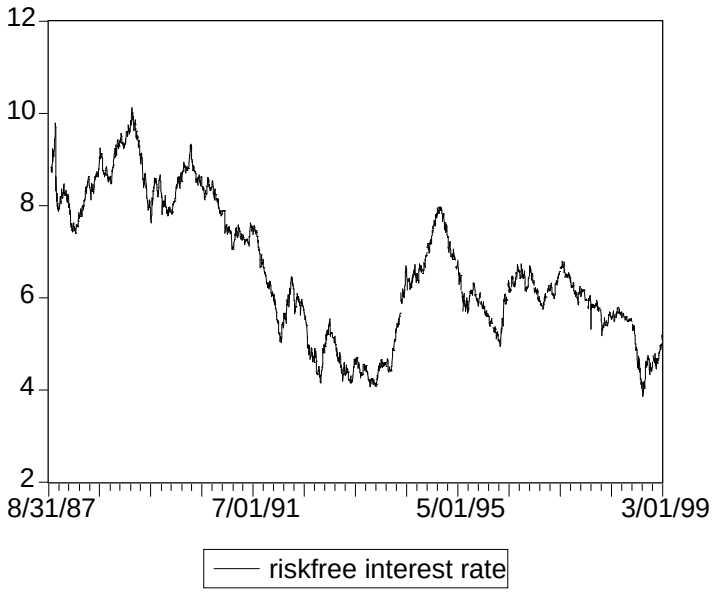


Figure 6.2

As an estimate of the volatility (σ) we calculate the historical 30-day-volatility using $\sigma = s \cdot \sqrt{254}$, where s is the standard deviation of the returns for the close-to-close S&P 500 levels of the most recent 30 days. The factor 254 corresponds to the number of an average trading year. To calculate the volatility we take the returns of the S&P 500 without the crash of the "Black Monday" the 19.10.1987, since that outlier was solely responsible for most of the volatility during that time period (see picture 6.3).

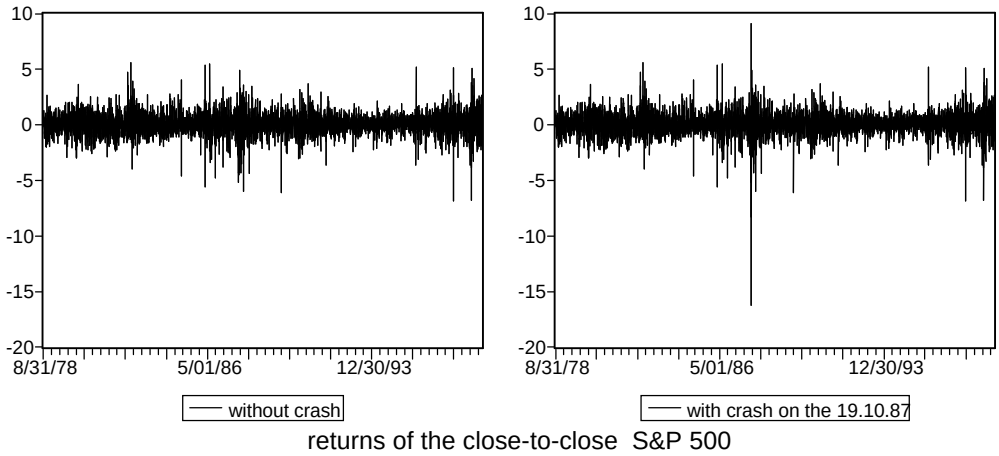


Figure 6.3

The 30-day-volatility which we calculate also seems to match the published volatility index VIX, which is available from the CBOE for more recent time periods than our sample.

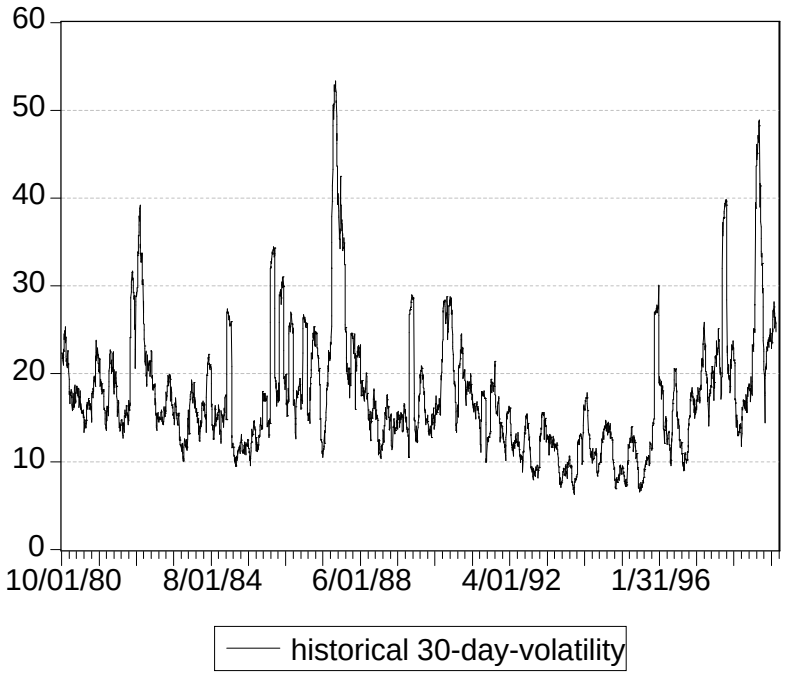


Fig.:6.4

6.3 Optimal Neuro Fuzzy Networks

To gain an insight into the different behaviours of the nonlinear models we use the following three models:

1. Black Scholes
2. Neuro-Fuzzy Network (NFN)
3. Neural Network (NN)

While the first is a case of straightforward arithmetic, the optimization of the networks requires a method for solving constrained nonlinear (minimax) optimization problems (due to the constraint in equation 4.8), which was supplied by the Institute of Systems Research from the University of Maryland, USA. This code is an implementation of two algorithms based on a monotone line search (Armijo type arc search) and a nonmonotone search along a straight line. The merit function used in both searches is the minimum of the error function with respect to the constraint. Partitioning into two random subsamples consisting of 2500 data records and 2555 data records, respectively, the networks were optimized on the former subsample and then tested on the latter. Using this in our specification algorithm (see section 4.3) we get the following "optimal" solution for the NFN:

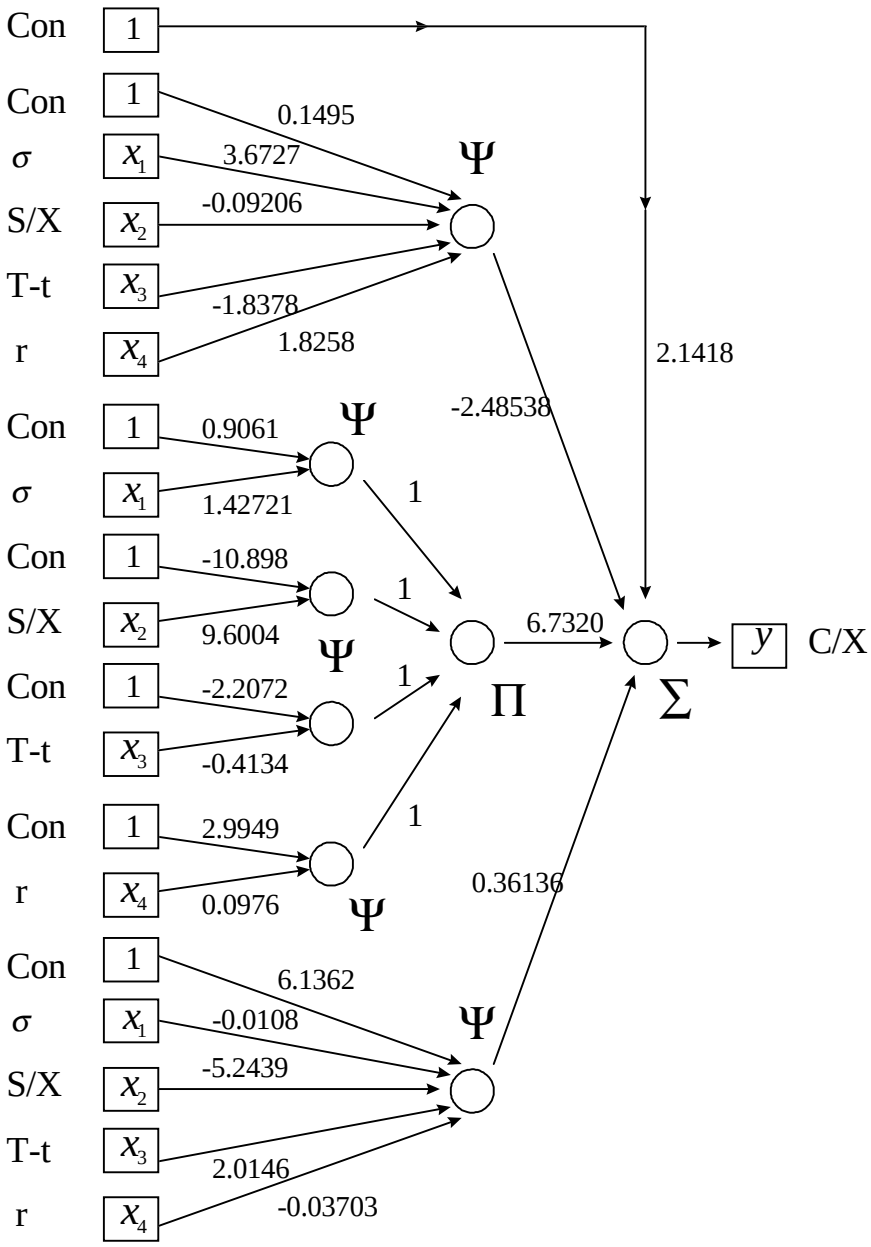


Fig. 6.5:Optimal NFN with 4 input variables. The numbers are estimated weights.

This NFN consists two neural units and one fuzzy unit, where Ψ is a sigmoid function. Figure 6.6 shows systematic differences in the pricing accuracy between the pricing errors of the Black-Scholes model and the NFN.

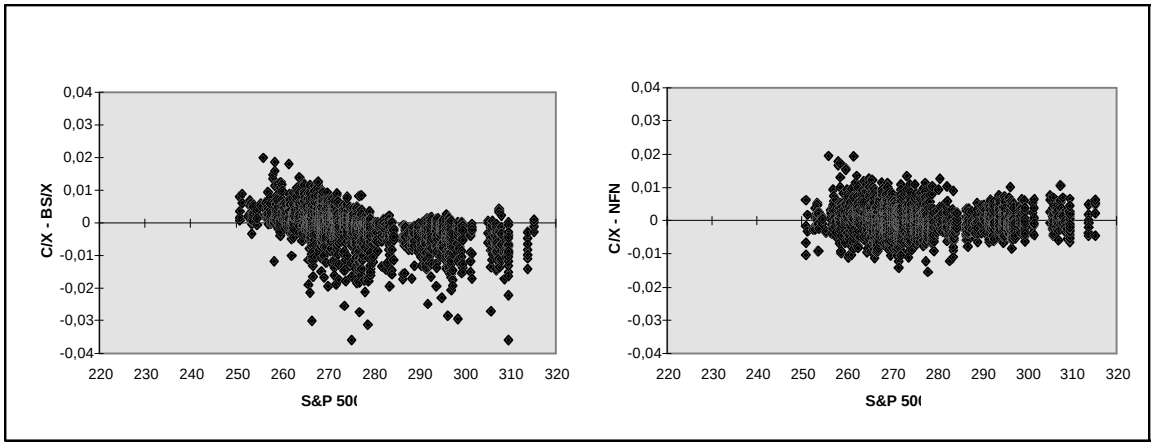


Fig. 6.6:Pricing error of Black-Scholes model and NFN plotted against S&P 500

The NFN errors are spread more evenly than the Black-Scholes errors, indicating a clear bias reduction and there are far less outliers resulting from the NFN than the Black-Scholes model. The results of the three models are now formally compared by the following measures.

6.4 Pricing Accuracy

To compare empirical prices (C/X) with those obtained from the different models, the following measures of fit were computed:

- Root Mean Squared Error statistics depend on the scale of the dependent variable. This is used as a relative measure to compare forecasts for the same series across different models; the smaller the error, the better the forecasting ability of that model according to that criterion.

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}[(\widehat{C/X})_t - (C/X)_t]^2},$$

  where $\widehat{C/X}$ are the fitted option prices and C/X are the observed prices.

- R-squared (R^2) statistic measures the success of the regression in predicting the values of the dependent variable within the sample. R^2 is the fraction of the variance of the dependent variable explained by the independent variables. The statistic will equal one if the regression fits perfectly, and zero if it fits no better than the simple mean of the dependent variable.

$$R^2 = \frac{\sum_{t=1}^{T}[(\widehat{C/X})_t - (\overline{C/X})_t]^2}{\sum_{t=1}^{T}[(C/X)_t - (\overline{C/X})_t]^2},$$

where $\overline{C/X} = \left[\sum_{t=1}^{T}(C/X)_t\right]/T$ is the mean of C/X.

- Adjusted R-squared ($adjR^2$)

  One problem with just using the ordinary R^2 as a measure of goodness of fit is that the R^2 will never decrease as you add more regressors (parameters). In the extreme case, you can always obtain an R^2 of one if you include as many independent regressors as there are sample observations. The $adjR^2$ penalizes the R^2 for the addition of regressors which do not contribute to the explanatory power of the model. The $adjR^2$ is computed as

  $$adjR^2 = 1 - (1 - R^2)\frac{T-1}{T-k},$$

  where k is the number of regressors. $adjR^2$ decrease as you add regressors, and for poorly fitting models, it may be negative.

The measures aim to highlight different aspects of the pricing accuracy. While $adjR^2$ provides a measure of correlation between observed and fitted option prices, the $RMSE$ give absolute measures of price discrepancy. Table 1 shows the performance measures for both the estimation using the first subsample (2500 records) and the out-of-sample evaluation with the second supsample (2555 records).

Some general results can be easily seen now. Firstly, the neuro fuzzy network (NFN) and the neural network (NN) are superior in performance to the Black-Scholes formula, as was to be expected by above mentioned studies. Both, the in-sample as well as out-of-sample measures of error are similarly low, indicating a successful treatment of the bias variance problem (see section 2.5) since excessive complexity would show up in a significant difference between both subsamples. Also the generally small RMSE gives no indication of model bias, which is in line with existing results for networks of similar complexity (e. g. Campbell et al.). Black-Scholes prices on the other hand are in general not only less accurate than the empirically derived pricing functions, given their $RMSE$, they are also highly biased (see picture above) with a lower R^2.

in-sample	RMSE	adjR^2
Black-Scholes Model	0.0059	0.9745
Neural Network	0.0043	0.9873
Neuro Fuzzy Network	0.0040	0.9893

out-of-sample	RMSE	adjR^2
Black-Scholes Model	0.0062	0.9653
Neural Network	0.0045	0.9854
Neuro Fuzzy Network	0.0041	0.9881

Table 1.: Performance measures of competing models

Both network models clearly dominate the Black-Scholes results and the NFN has a small edge even with respect to the NN. It has the highest pricing accuracy with respect to the measures $RMSE$ and $adjR^2$ both in-sample and out-of-sample. While certainly far from all-encompassing, this result is supportive of our theoretical approximation results, because the neuro fuzzy network manages to derive a specification which improves on the fit even of the very accurate neural network without adding complexity. This is achieved by the richer functional relationships that are within the approximation bounds of NFNs but beyond the possibility of NNs. One should note though that given its relative implicity and analytical convenience the fit of the Black Scholes is quite impressive. While there is certainly enough margin for improvement on an economic level, its statistical accuracy issue with most of the underlying assumptions invalidated by empirical financial research it remains study benchmark.

7 Summary and Conclusion

In this paper we achieve a contribution to several key issues in nonlinear modeling with neural networks. Initially, we show that the nonlinear structures of NN and FN are sufficiently similar to construct an efficient hybrid model (NFN) based on a constraint. It is then proved that the combination of different hidden nodes in an NFN can improve generalization through bias reduction without increasing complexity. Also, we propose an algorithm which can achieve an efficient combination of different specifications constructively. Finally we study a common nonlinear problem in finance and find that our theoretically motivated method also offers an empirically viable approach to improve nonlinear approximation. Since the performance comparison based on equivalent parameter space dimensions yields a slight edge of the NFN relative to either NN or FN as well as a distinct advantage over the Black-Scholes option pricing model.

Further research should certainly be comprised of simulation studies of the proposed NFNs' properties in comparison to NNs and FNs on a variety of data generating processes to increase the understanding of optimal domains of application for the new technique.

8 Acknowledgments

I would first and foremost like to thank my advisors. Professor Bartels' interest in applied financial mathematics and Professor Gottschling's patience and support were both essential for this task. I also wish to thank Christof Kreuter from Deutsche Bank Research for competent advice in all matters of programing and Renef Zinsmeister, Dirk Fesser, Nikolaos Avramidis for helpful discussions and good suggestions. Last but not least, I am very grateful to Deutsche Bank Research for allowing me to use their technical equipment as well as their data for this thesis.

9 References

Anders, U. (1997): Statistische neuronale Netze, Franz Vahlen München.

Anders, U., Korn O., Schmitt C. (1996): Improving the Pricing of Options, ZEW Discussion Paper.

Bishop, C.M. (1995): Neural Networks for Pattern Recognition. Clarendon Press Oxford.

Black, F., Scholes M. (1973): The Pricing of Options and Corporate Liabilities. Journal of Political Economy, Vol 81, 654-673.

Campbell, Lo, McKinlay (1997): The econometrics of financial markets. Princeton Univ. Press.

Cox, J.C., Ross S.A. (1976): The Valuation of Options for Alternative Stochastic Processes. Journal of Financial Economics, 3, 145-166.

Dennis, J.E. (1983): Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Englewood Cliffs, NJ: Prentice Hall.

Fletcher, R. (1987): Practical Methods of Optimization. New York: John Wiley.

German, S.; E. Bienenstock, and R. Doursat (1992): Neural Networks and the bias/variance dilemma. Neural computation 4 (1), 1-58.

Gill, P.E (1981): Practical Optimization. London: Academic Press.

Gottschling, A. (1997): Structural Relationships between Nonlinear Approximators, PhD Thesis, University of California, San Diego.

Gottschling, A., Tymon Tatur (1999): Approximation Capabilities of Generalized Feedforward Networks, Mimeo, Deutsche Bank Research.

Halmos, P.R. (1974): Measure Theory. New York: Springer-Verlag.

Hornik, K., Stinchombe M., White H. (1989): Multilayer Feedforward Networks are Universal Approximators. Neural Networks, Vol 2, 359-366.

Hutchinson, J.M., Lo A. W., Poggio T. (1994): A Nonparametric Approach to pricing and Hedging Derivative Securities via Learning Networks. The Journal of Finance, Vol 49, 851-889.

Ichikawa, Y. (1992): Neural network application for direct feedback controllers. IEEE Trans. Neural Networks 3(2):224-231.

Klir, G.J., Yuan Bo (1995): Fuzzy sets and fuzzy logic. Prentice Hall.

Kolmogorov, A. N., and V. M. Tihomirov: "$\varepsilon-$entropy and $\varepsilon-$capacity of sets in functions spaces, Uspehi, 14, No. 2(86) (1959), 3-86.

Kosko, B. (1992): Neural Networks and Fuzzy Systems: Prentice-Hall.

Lin, C.T., Lee C.S.G. (1995): Neural Fuzzy Systems. Prentice Hall.

Lorentz, G. G. (1966): Approximation of Functions, Holt, Rinehart and Winston.

Luenberger, D.G. (1984): Linear and Nonlinear Programming Reading, MA: Addison-Wesly.

Maxwell, T., Giles, G.L., Lee, Y.C. and Chen, H.H. (1986): Nonlinear dynamics in artificial neural systems. New York. American Institute of Physics.

McCulloch, Pitts (1943): A logical calculus of ideas immanent in nervous activity. Bull. Math. Biophys.. 5:115-133.

Merton, R.C. (1976): Option Pricing When Underlying Stock Returns are Discontinuous. Journal of Financial Economics, Vol 3, 125-144.

Montana, D.J., (1989): Training feedforward networks using genetic algorithms.11th Int. Joint Conf. Artif. intell., 762-767, Detroit.

Polak, E. (1971). Computational Methods in Optimization: A Unified Approach. New York: Academic Press.

Ripley, B.D. (1996): Pattern Recognition and Neural Networks. Cambridge University Press.

Rubenstein, M. (1985): Nonparametric Tests of Alternative Option Pricing Models. The Journal of Finance, Vol 40, No 2, 455-480.

Rudin, W. (1974). Real and Complex Analysis. New York: Academic Press.

Stinchcombe, M. (1995): Precision and approximate flatness in artificial neural networks. Neural Computation 5, 1021-1039.

Wang, L.X. (1994): Adaptive fuzzy systems and control. Prentice Hall. 9-47.

White, H. (1992): Artificial Neural Networks. Blackwell.

Whitley, D., and T. Hanson (1989): Optimizing neural networks using fast more accurate genetic search. San Mateo, CA: Morgan Kaufmann.

Xu X., Taylor S.J. (1994): The Term Structure of Volatility Implied by Foreign Exchange Options. Journal of Financial and Quantitative Analysis, Vol 29, No 1, 57-74.

Zadeh, L.A. (1965): Fuzzy Sets. Inf. Cont. 8:338-353.

All Deutsche Bank Research products are also available by e-mail. **Subscribers receive the electronic publication on average four days earlier than the printed version.** If you are interested in receiving this product by e-mail, please get in touch with your Deutsche Bank sales contact or with the DB Research Marketing Team: **marketing.dbr@db.com**

Printed by: HST Offsetdruck GmbH, Dieburg.
Print: ISSN 1615-956X / Elektr.: 1615-9683 / Internet: ISSN 1616-0428