LEARNING RELIABILITY:

a study on indecisiveness in sample selection

CIP–GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Barakova, Emilia Ivanova

Learning Reliability:
a study on indecisiveness in sample selection – / E.I. Barakova
Proefschrift Rijksuniversiteit Groningen

Keywords: information technology, neural networks, reliability, symmetry, knowledge engineering.

ISBN 90–367–0987–3

PrintPartners Ipskamp B. V.
Capitool 25 Business & Science Park
Postbus 333
7500 AH Enschede

Rijksuniversiteit Groningen

LEARNING RELIABILITY:

a study on indecisiveness in sample selection

Proefschrift

ter verkrijging van het doctoraat in de
Wiskunde en Natuurwetenschappen
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. D.F.J. Bosscher,
in het openbaar te verdedigen op
vrijdag 23 april 1999
om 14.15 uur

door
Emilia Ivanova Barakova
geboren op 30 juli 1965
te Velingrad (Bulgarije)

Promotor:	Prof.dr.ir. L. Spaanenburg
Referent:	Dr.ir. J.A.G. Nijhuis

Beoordelingscommissie: Prof.dr.ir. W.M.G. van Bokhoven
Prof.dr.–ing. A. Kistner
Prof.dr.ir. J. Nerbonne

This thesis is based on the following publications:

1. (Barakova, E.I., and Spaanenburg, L.) *Learning and reproducing*, in: (Spaanenburg, L. et al.) V−Annals II (Shanker Publ., Maastricht), 1999.

2. (Barakova, E.I., and Spaanenburg, L.) *Windowed Active Sampling for Reliable Neural Learning*, Journal of Systems Architecture **44**, No. 8 (Elsevier Scientific Publ., Amsterdam, 1998) pp. 635−650.

3. (Barakova, E.I. and Spaanenburg, L.) *Symmetry: Between Indecision and Equality of Choice*, pp. 903−912, in: (Mira, J., Moreno−Diaz, R., Cabestany, J.) *Biological and Artificial Computation: From Neuroscience to Technology*, Lecture Notes in Computer Science **1240** (Springer Verlag, Berlin) 1997.

4. (Barakova, E.I., and Spaanenburg, L.) *Selective Sampling for High Reliability in Neural Signal Approximation*, Proceedings NICROSP'96 (Venice, August 1996) pp. 183−193.

5. (Barakova, E.I., Spaanenburg, L., and Zaprjanov, J.) *Neural fault diagnosis of a turbogenerator by vibroacoustic data*, Int. Conf. on Signal Processing, Applications & Technology ICSPAT'95 (Boston, MA, USA, 24−26 October 1995) pp. 1454−1458.

Preface.

The design of a product is based on the assumption of how it will be used. Conversely, the product is usually good for only such usage as was assumed during its conception. In a classical sense, the implicit assumption brings an explicit specification from which the design is derived. More often than not, the specification is therefore the starting point of a hopefully structured and well–behaved, but eventually mechanical design effort. Where the customer tends to learn from the design and mandates to change and/or augment the specification during the process, the project planning gets invalidated. Current practice is therefore to fix the specification in advance, for instance by contract.

The interest in Artificial Neural Networks (ANN) is founded on their ability to learn from examples, as derived from the environment in which the product will operate, instead of being designed from an hypothesis about the operation. It is commonly agreed that learning is based on *memorization* (associating or mapping a set of questions to their answers) and *generalization* (the ability to answer new questions about the same problem). As such, ANNs promise a perfect fit to their intended usage. But circumstantial evidence still does not equal a witness observation. Despite its historic fame, an Artificial Neural Network will not learn all, let alone under all circumstances. This is probably the most striking difference with a designed product: there will never be a proof by construction!

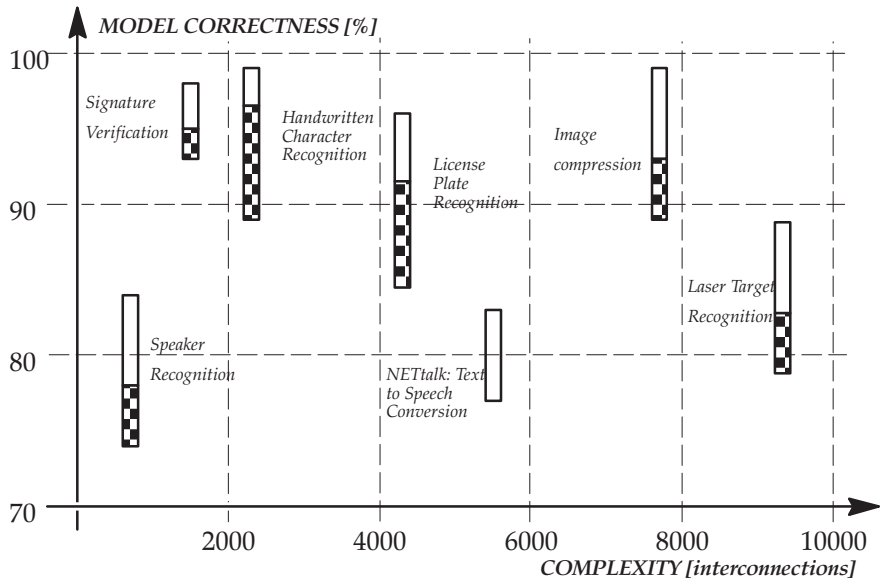


Figure 1: *Overview of real–world neural applications.*

With the coming of age of neural technology, an impressive number of neural products have found their way to the market place [88]. Some popular applications are indicated in figure 1, which position them in the area spanned by computational complexity and

model correctness. The bars indicate the achieved performance: the patterns within the bar indicate the widely achieved results, since the white part stands for the best results in the area. Clearly, none of them achieves a 100% correct functionality. It appears, that for each application a bottom level of functionality can be reached almost without any effort. However, to go beyond requires special attention and has therefore spurred a lot of research to develop new algorithms, to construct alternative architectures, to provide different settings of input parameters or to preprocess input data.

To achieve a product of ultimate performance, two methods can be devised: (a) its function is based on a provably correct algorithm, and (b) an effective redundancy is to be incorporated in the underlying algorithm. As far as ANNs are constructed from analysis of noisy data, they can entirely be considered as systems of the second type. Because statistics is concerned with data analysis as well, there is a considerable overlap between the fields of neural networks and statistics. To analyze learning and generalization of neural networks from noisy/randomized data, statistical inference can also be used.

Performance enhancement can be created by a kind of majority voting. This principle suggests that, instead of providing one neural network solution to a problem, a set of neural networks can be combined to form a neural net system which performs better than any of the networks on its own [116] [138]. The conclusion made in [112] is that mere redundancy does not necessarily increase reliability. Empirically it is common practice to train many different candidate networks to select the winner on basis of predefined criteria. A disadvantage of this method is that training of the losing networks does not help in a further development. Another weak point is that the criterion for choosing the best network is usually the performance on a validation set, which can not guarantee the modeling quality of the underlying data generator. But when the networks are incomplete versions of the same functionality, the combination might raise the functional correctness to a higher level (Figure 2).

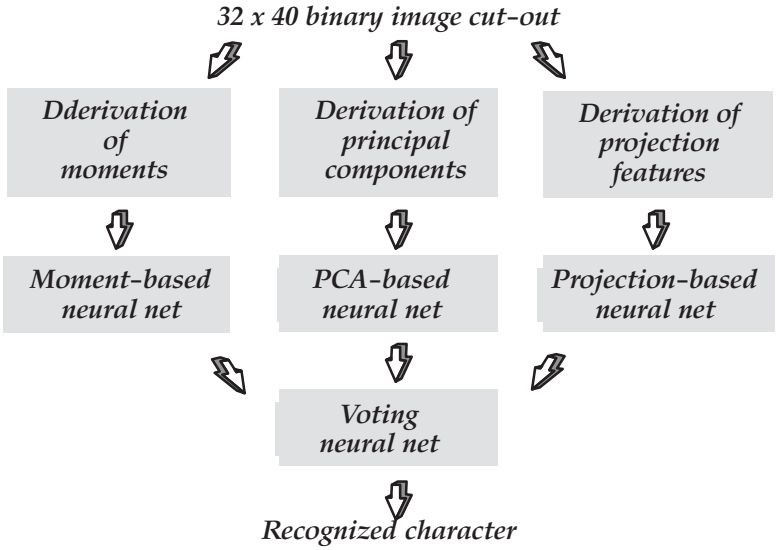


Figure 2: *A typical character recognizer (from [28]).*

The committee arrangement generalizes this idea. It can have significantly better predictions on new data at an acceptable increase of the computational complexity. The performance of the committee can be much better than the performance of each single network in isolation. The committee contains a set of a trained networks diversified in a distinct way. *Diversity* can appear in the number of hidden neurons, in the kind of network model, in the mixture of networks, in the optimization criteria, in the initial weight configuration, training parameters in the training samples, etc. The extent to which reliability can be improved by combining neural net solutions depends on the type of diversity, present in the set of nets.

All such techniques assume that the basic neural network is optimally trained. However, we have noticed that training algorithms are often slow and sometimes unable to converge, even though the underlying techniques often perform very well on other problems. In other words, even though an ANN can be trained to some functionality, there appears to be an underlying problem that causes unreliability in learning. This thesis will therefore be devoted to unravel such circumstances and to contribute ways in which reliable learning can be achieved. By large, the neural paradigm problem is represented as a stream of examples (data) and that guides the learning algorithm to adapt the network parameters until the network is "trained" to give the right answers to the posed questions. Thus the success and the reliability of this training depends to a large extent on the content and composition of this data stream.

Overall unreliable learning can be considered to result from the interaction between three factors: network, problem, and algorithm. In an attempt to answer questions like why and when the learning process will become unreliable and when a systematic failure can appear, backpropagation (still the algorithm with highest practical significance) has been used. The restricted class of architectures it is supposed to be used for and the feedforward architecture allow us to elaborate in more detail on the problem with respect to the chosen architecture and algorithm.

As we found that the conventional focus on network, problem and algorithm leaves much to be desired, we propose here to base the discussion rather on *symmetry*, *randomness* (as basic network design principles), and *knowledge* (the problem to be learned) as the basic ingredients of the universe of discourse. A high degree of symmetry in the initially designed network is historically viewed to favor the learning algorithm in providing an equal chance to move in several directions. However, this has also a drawback: the freedom of choice may lead to indecisiveness. Admittedly, randomness may in turn help the network escape from such a dilemma. But then again, randomness may wipe away the knowledge; hence a working balance should be found.

Symmetry can be dominant in the beginning of, but also at specific moments during, learning. Randomness (for instance as stochastic variable in the learning algorithm or as additional noise at the network input, output or internal parameters) is then required to force the presentation of examples to follow alternative itineraries. When the amount of randomness is not sufficient to counteract symmetry, learning will not be completed: instead of being adapted to ensure the right mapping between input/output data strings, the initial parameters will eventually become zero. If the noise (the randomness) of the system is dominant, learning will also be unsuccessful, because the network will rather learn the noise than the exemplified knowledge. The fundamental issue of learning is

therefore the creation of a functional balance between symmetry and randomness directed by the examples (the knowledge).

To bring this idea into tangible borders, the interaction between learning components is represented in the *error surface* paradigm. The network will be able to extract the necessary information by adapting itself to map the questions posed to the right answers. This adaptation is in fact an optimization procedure and is thus equivalent to finding the minimum energy state on an error landscape. The steps, that the learning algorithm takes on this landscape, are directed by the presented examples and form a *learning trajectory* on this surface. Directing this itinerary properly can help to escape some difficulties to pass surface areas, at which the learning algorithm normally spends a lot of time on or from which it can never escape. For finding an optimal trajectory on the error surface, the so–called regularisation methods have been used. An alternative effect has the introduction of extra noise during training. Our objection here is that the task complexity or the convergence accuracy may be changed in an unwanted direction. The investigation of the statistical long–run effects of example presentation when traveling on the difficult forms of the global error surface brings us to a constructive algorithm which helps in escaping them.

Therefore, the work in this thesis takes an alternative route to ensure reliable learning by focussing on *sample diversity* [116]. On basis of the instantaneous characteristics of the current training set we will conclude on learnability, reorder the set if necessary to establish the best sample sequence and train eventually a single network with success.

In conclusion, this thesis aims to give directions on how learning can be guaranteed so that its duration will be short and stable and its success unquestionable from the outset. In this respect, we aim to contribute to move neural technology from the realm of “Learning by Examples” to “Design by Examples”.

Groningen, march 1999.

Contents.

1 Introduction.

For the benefit of the reader we will shortly provide some of the conventions in neural network technology as will be used in this thesis. From the rich variety of naming and notation conventions, we introduce our choice though in the remainder other naming and notation conventions will regularly be pointed out for the sake of completeness. At the end of this chapter we will shortly mention the problems to be solved in a general setting and provide an overview of the thesis.

The explosive development of *computing science* and technology has made it possible to enhance (or even substitute) many human intellectual activities by computer programs. Traditionally the arithmetical manipulation and logical sequence of the human deduction process is computerized. Over the past decades, *intelligent computing* appears as an alternative approach for mimicking human thinking. In addition to methods, that try to incorporate the logical steps of human thinking, some intelligent techniques imitate the conclusions, that humans make by experience: the answer to a new question is based on the similarity with an already seen example. This property makes them indispensable in unexplored areas, where logical analysis can not go far enough to solve the problem at hand.

From a user's perspective, the decision–making techniques incorporated in a computing device are not of importance, if the required answer can be correctly and reliably obtained. The techniques, which follow the logical development of the events and processes, can be implemented by provably correct algorithms with appropriate safe–guards. Intelligent computations, whose reasoning mechanism is based on similarities with already seen events, can not give a straightforward proof for the correctness of its conclusions, as the underlying algorithms do not include such safe–guards.

The correctness and reliability of intelligent computing techniques can not be explained in general, because of the variety of underlying operational principles and mathematical methods. Our particular interest will be focused on *Artificial Neural Networks* (ANNs), because of their high practical significance. The starting point is the hypothesis that the abnormal learning behavior so often noticed by scientists and practitioners in the field can be readily explained and therefore remedied by a well–structured design methodology.

Neural networks are useful in situations where the logical way of decision–making, as basis for the so–called 'conventional' methods, does not help. The major distinguishing feature of the neural method is that the problem to be solved is not modelled in advance. Instead, the network adapts its parameters to fit the problem, i.e. the network itself becomes a model of the problem. Since the available knowledge on what the problem is represented (encoded) in the set of data pairs, the network models the mapping between these pairs of data. If the data set gives a good representation of the problem, the network will be a model not only of the available data set, but also of all possible data, which can be derived by (measured from) the problem. In this sense it can be said that the network models the internal *data generator.*

Another profitable aspect of neural networks is their highly non–linear behavior. Where mathematical modeling has for a long time preferred a high degree of linearisation, the ANN tackles directly the non–linear reality. This also explains why it has been so difficult to fully explain the ANN operation in mathematical expressions. As further nature tends to provide irreproducible, noisy and incomplete data, it becomes almost self–evident to rely on stochastics rather than analytics.

Neural networks are trained, not programmed, and therefore withstand any invasion by formal proof techniques. In other words, where usually the design creativity is spent on the construction of the product, any structured design technique for ANNs must take the internal structure for granted, being a universal computational template, but focuses on harnassing the interaction with the environment.

To set the scene, we will first introduce the neural network itself. Though still in its infancy as a design discipline in its own right, some universally agreed nomenclature and early successes can be given. Then we go to some elementary design notions and cite a number of seemingly unrelated reliability (trustworthiness) problems of ANNs. Lastly we make our plea for designed–in quality and introduce our contribution.

1.1 Neural networks.

A *neural network* is in its most general form a system of connected elementary processing units (PU), that for historical reasons are called *neurons*. Each neuron is associated with a real numerical value, which is the state of the neuron. Connections between neurons exist so that the state of one neuron can influence the state of others. The system operation can be programmed by weighting each connection to set the degree of mutual influence. Each processing unit performs the following operations: (a) the incoming signals x are collected after weighting, (b) an offset θ is added, and (c) a non–linear function φ is applied to compute the state.

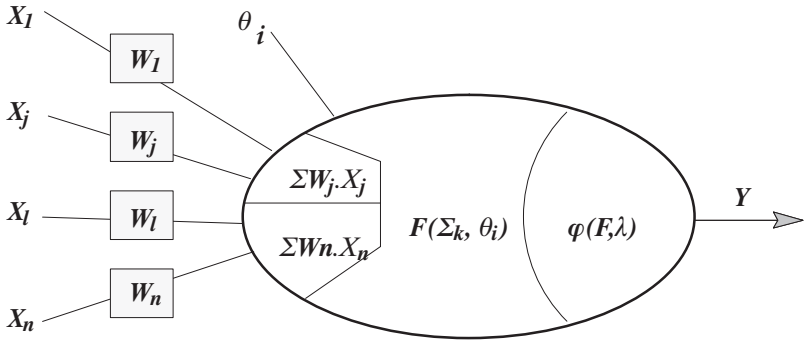


Figure 1–1: *The operation of a single neuron.*

A single neuron is depicted in figure 1–1. It shows the stated series of computations when going from left to right. Creating a network of neurons can be done in numerous ways, each different architecture for different applications. The system gets a specific function from the structure and the weight setting: the structure allows for general func-

tionality, while the weights sets the specific function. In the following these two aspects of a neural system will be coarsely reviewed.

1.1.1 The network structure.

The neural architecture represents globally and graphically the network function. The network function (network transfer function, mapping function, or answer of the network are equivalent terms) accomplishes the main task of the network: adapts to associate the questions, posed to the network, with their answers. The questions and answers are terms, borrowed from the human association process. As widely used alternatives the terms input and output *examples* are used: to the network are shown examples of how the problem to be solved behaves; the input example (the stimulus) is shown together with the adequate reaction to it – the response or the output example. This process is performed in the *forward pass* through the network: the outputs are obtained as a response of the neural system to the input (question) stimulation. Thus the forward pass through the network evaluates the equation that expresses the outputs as a function of the inputs, the network architecture, the nodal transfer, and the parameters so that during the *backward pass* the *learning* algorithm can adapt the connection strengths.

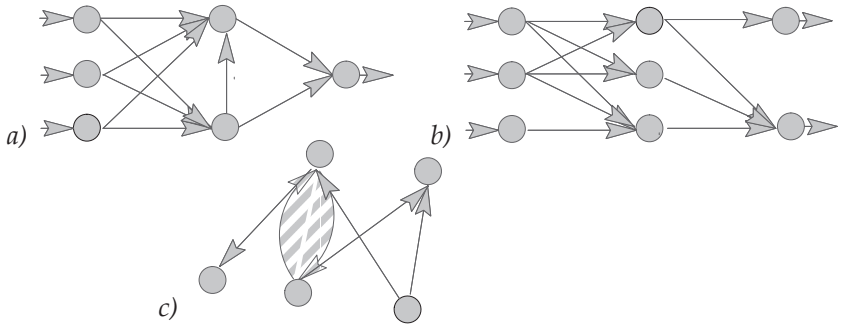


Figure 1–2: *Some ANN structures: a)Layered; b)Feedforward; c)Recurrent.*

In figure 1–2 the most frequently used network structures are given. The structure from figure 1–2a is also called *layered*: its neurons can not influence neurons in previous layers. Figure 1–2b represents the *feedforward* architecture: there are no connections within one layer. The two structures in figure 1–2a and b are nonrecurrent, because there is not a neuron which influences another neuron and is at the same time influenced by the same one. The network from figure 1–2c is called *recurrent*, since two neurons influence each other.

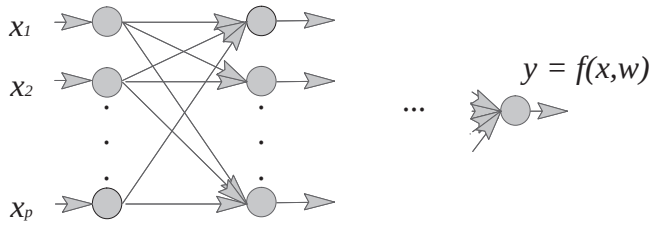


Figure 1–3: *A multilayer feedforward network.*

For the multilayer feedforward neural network (Figure 1–3) the input vector $\boldsymbol{x}$ and the scalar output $\boldsymbol{y}$ are connected via the network function f, as described by equation (1–1)

$$f(\boldsymbol{x}, \boldsymbol{w}) = \varphi\Big(\sum_i w_{ji}\varphi\big(...\varphi\big(\sum_k w_{ik}x_k\big)\big)\Big) \qquad (1\text{–}1)$$

where φ is the nodal transfer and w_{mn} denote the different weight connections within the network architecture with indices according to the direction of information transport. The nested structure of the formula represents the steps that the feedforward track of the learning algorithm has to pass.

The nature of the non–linear neural transfer function has not been specified so far. Basically it could be anything, from a straight threshold to a complicated expression. As formula (1–1) already indicates, it is even possible to wrap an entire neural network to be used in another network as a single neuron with a complex transfer [86]. We will refrain from such and apply only an S–shaped type of transfer, especially a sigmoid function.

More specifically, we will use the *logistic* and the *zero–centered* variety: two sigmoid transfer functions with an exemplary symmetry of which the effects on the overall learning performance are nicely suited to display the behavior as studied in this thesis. Their practical importance relies on the fact that the logistic sigmoid has found major application in digital circuitry, while the zero–centered sigmoid appears foremost in analog circuitry. The specification of these functions is shown in table 1–1.

Table 1–1: *Specification of two sigmoid transfer functions.*

	Logistic	Zero–centered
Function φ	$\frac{1}{1 + e^{-x}}$	$\frac{1 - e^{-x}}{1 + e^{-x}}$
Output range	(0, +1)	(–1, +1)
First–order derivative	$\frac{e^{-x}}{(1 + e^{-x})^2}$	$\frac{2e^{-2x}}{(1 + e^{-x})^2}$
$\varphi' = f(\varphi)$	$\varphi' = \varphi.(1 - \varphi)$	$\varphi' = \frac{1}{2}.(1 - \varphi)^2$

1.1.2 Network operation.

The ANNs adapt themselves to map the event–consequence evidences of behavior of the problem to solve on the known problem areas. In human terms it can be said, that the network *learns* the problem. This learning is an optimization process, accomplished in a lot of the networks by a *gradient optimization* method. The gradient methods can be described as a hill–climbing (descending) procedure, which ends by finding an optimum (extreme) state, in which the network models the problem best. At this state it is expected, that every question from the problem area, presented to the network, will bring the right reaction (answer) at the output of the network (Figure 1–4).

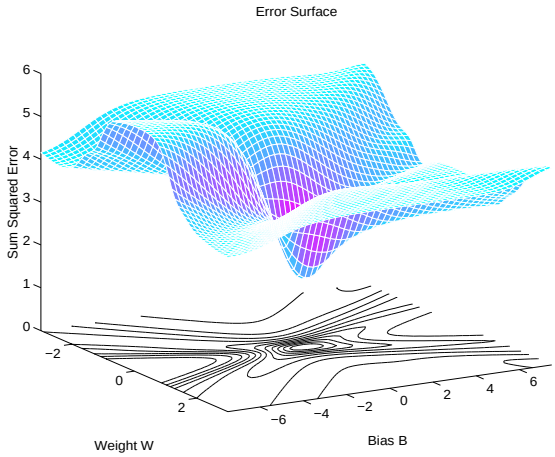


Figure 1–4: *The neural method is a hill–climbing procedure.*

Any optimization problem can be defined by a set of inequalities and a single cost function. In neural networks, this cost function is usually based on a measure of the difference between the actual network response and the desired one. Its values define the landscape in which an extremum must be found. By deliberately adapting the weights in the network, both responses are brought together, which defines the desired extremum. In other words, a measure of the difference in responses for one or more inputs is used in an algorithm to let the network learn the desired weight settings: the *learning algorithm*.

As learning results from the mutual attraction of actual and desired responses, it will be based on the feedback principle. In *supervised learning* this is achieved by an external algorithm in which a new weight setting is computed by the supervisor; in *unsupervised learning* the algorithm is designed into the network structure as an internal competition between adjacent neurons. For this reason the usable network structure for the two approaches is fundamentally different.

The operation of the network is severely influenced by the relation between the number of independent parameters in the problem to be learnt (say, the intrinsic dimensionality of the problem) and the number of independent parameters in the network (say, the dimensionality of the proposed solution). A network is said to be *underparametrized*, when the dimensionality of the proposed solution is too small to contain the problem; conversely, the network is *overparametrized*, when the dimensionality of the problem to be learnt is much smaller than the capacity of the network. While in the former case, the network will clearly never be able to learn, in the latter case, the degree of freedom to learn the problem will usually be counter–productive. It is therefore clear, that a proper parametrization of the network will be somewhat larger than what is required by the problem but then again not too large. It is a fundamental network design issue to determine just the right value [79].

The equi–functionality of neurons in the researched networks, their connectivity and initialization principles, ensures a clear, not predefined start of the optimization process. Beside that these features of network design help the optimization to be guided purely by the problem to solve, they have a negative impact as well, since the gradient

algorithm may be brought to indecisiveness in its attempt to find a descent direction. To help the learning process escape from this indecisiveness, an effective randomization may be added.

This all adds to well–known problemss while reaching out for a global extremum in the presence of many local ones. The path is one with many pitfalls, that may easily draw attention away from the fundamental problem at hand. Nevertheless neural networks have become reasonably successful, notably in those areas where by other means a proper problem representation has already been found. Here, the neural network will provide a reasonable (if not favorable) alternative.

1.1.3 Successful applications.

In the short history of neural networks, a number of impressive applications have been demonstrated. Though most of the really successful industrial applications have not been widely publicized, their occurrence has clearly been spurred by the fore–running toy ones.

The most widely spread story is the *truck–backer–upper* as demonstrated by Widrow [111]. Here a truck is controled by a neural network that is trained by examples on how to back–up a truck. This task is a well–known control problem as it already poses a challenge to the novice human truck–driver. In the course of time, this experiment has been repeated several times and has eventually produced the disclosure of an almost trivial solution [84]. A similarly popular experiment is the *balancing pole* [63]. Here, a pole is placed on top of a car and the car is controled to move such that the pole is well–balanced and will never fall. This experiment is already of a more complex nature because of its dynamics. Even before the first neural experiment has been performed, a closed–form analytic solution has been found from control theoretical analysis. These two experiments have clearly indicated to the community the attractiveness of using neural networks to prototype solutions for non–linear control problems and have thus contributed to further control systems research.

From a different nature is the handling of sensory data on the Challenger engine test–site [68]. From a series of mishaps it has been concluded that, by classical control, accidents can result from wrong sensor readings. A neural network is then introduced to monitor the different readings to diagnose the data quality. When a sensor fails, the trained networks will simply overrule the reading with data that correlates better with the correctly functioning sensors.

All such applications can be viewed as examples of function approximation, where the success of learning becomes heavily dependent on the quality of data preparation. While much theoretical work has been devoted to improvements in neural network structure and learning algorithm, the practical short–cut proves to be the abundant usage of preprocessing such that the requirements for further learning become almost trivial (Figure 1–5). This preparation starts with straight signal shaping, and continues with *feature extraction*, often based on *data clustering*. Thanks to the inspiring work of Kohonen and von der Malsburg, clustering has evolved to a maturity that allows for a range of applications (for instance in automotive motor management [70]). One may be eaily misled to think that proper clustering for feature extraction virtually eliminates the need for intelligent function approximation. However, a closer inspection shows

that also Kohonen maps give no guarantee and a number of problems exist that remain unlearnable. In other words, the step of function approximation remains necessary and it fundamental problems must still be solved.

Figure 1–5: *The neural processing chain.*

1.2 Deriving the neural model.

The ability to learn has been defined in many ways; still there is no agreement to what the best definition is [74]. Commonly, the term learning is used to describe the process of solving the imitation problem as posed to the network. If the network should associate pairs of events, then the learning process completes when the network is able to make this association. If the ability of the network to model the underlying data generator is of higher importance, as in most real–life tasks, the learning process presumes not only the ability of the network to map the asked questions correctly to answers, but also to generalize – to give answers to unseen questions from the same example distribution.

Other differences in the definitions correspond to the training approach. The neural approach identifies learning with optimization of the error function. Drawbacks of this approach are commonly known and will be listed here for the sake of completeness. The alternative in model derivation is the direct optimization of the operational network. Though this solves some older problems, it also introduces some new ones. Hence it pays to inquire on the fundamental nature of learning problems in a new attempt to solve rather than to avoid such causes in a structured way.

1.2.1 Formalization of neural learning.

There are two well–established approaches, which formalize the neural learning process. The first one represents neural learning as a function approximation task; the second takes a Bayesian perspective.

In the first approach, many problems can effectively be modeled as learning an input/output mapping despite insufficient evidence of what this mapping can be. The mapping usually takes the form of some unknown function f between two spaces X and Y and is represented by a set of *examples or data pairs* (x^μ, y^μ), which are consistent with this function. The set of all example pairs that guides the learning algorithm to converge to the real function is called the *example or data set* D_n.

At this point it is of importance to distinguish between three types of data sets. Next to the *learning set* that supplies the examples to actually train the network, there are the test set and the generalization set. The learn set should allow the network to generalize. The *generalization set* will therefore test on the achievement of the generalization. This is different from the *test set*, that has the higher aim of verifying the achieved overall functionality.

During learning, the data set is traversed several times in an arbitrary order. At successive intervals the weights are adapted; in *on–line processing* this happens at every example presentation, in *batch processing* at every set presentation. Learning behavior will therefore be a function of *epoches*: the times at which both presentation and adaptation occur.

An elementary function *f* can be designed into a network using the principle of piecewise linear approximation [84]. Assume a network with 1 input, 1 output and H hidden neurons with sigmoid transfer (Figure 1–6). For such a network the output can be seen as the sum of the individual contributions of the hidden neurons. A constructive solution is then possible by a judicious choice of the bias of such hidden neurons such that for increasing input value x, more hidden neurons become activated. In other words, for a sigmoid transfer function between –1 and 1, the H hidden neurons will divide the function value range into segments at size $2/H$.

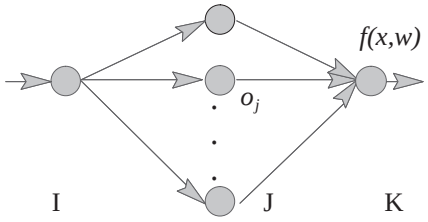


Figure 1–6: *A one–input, one–output feedforward network.*

Lets assume that the input neuron serves merely as a distributor, having connections to the hidden neurons with designable weight values, while the contributions of the hidden neurons are simply summed at the output. The output y reads as

$$y = \sum_{i=0}^{H-1} \varphi(\dot{w}_i x + w_{bi}) \qquad (1–2)$$

For a change in function input x only one hidden neuron will be in the linear region, i.e. for each input segment one hidden neuron must be centered at $x = -w_{bi}/w_i$. Though seemingly elegant, this design method exchanges accuracy for interval. I.e. a long interval with high accuracy will demand a huge amount of hidden neurons. Further it can only handle one signal at a time. In the presence of complex signals, these must first be adequately preprocessed and for each frequency component a separate neural net is required.

These drawbacks have given rise to a need for learning from examples instead of designing by hypothesis. As described before, the neural method consists of modeling a problem, presented as a mapping between input and output data streams. On basis of these examples the learning algorithm tries to converge to the real function. This way learning from examples is equivalent to interpolating or approximating a multivariate function from sparse and noisy data [64] [152]. The potential of a network to generalize from a finite training set to unseen data is one of its most attractive qualities. A 3 – component structure of the neural learning process can be considered, aiming to discover the best approximation to the supervisor's response from a given set of functions:

- Selecting input/output pairs $\{z^\mu \equiv (\boldsymbol{x}^\mu, y^\mu) : \mu = 1, 2, ..., N\}$, according to an unknown random distribution $P(x, y)$, where x^μ and y^μ belong respectively to the input and output spaces X and Y.
- Representing the probability distribution $P(x, y) = P(x)P(y|x)$ of the examples. This representation automatically assumes that there is a functional dependence f between the input and output spaces X and Y (i.e. x and y are dependent pairs of values).
- Inferring the functional dependence $f(x)$ between the input and output spaces X and Y, given the knowledge of the example set z^μ.

Using a probabilistic framework, it has been shown [64] that several different learning problems, such as classification and regression, are equivalent to the function approximation task. When viewing learning as a kind of approximation, the problem of learning a smooth mapping from examples is ill–posed in the sense that the information in the data is not sufficient to reconstruct uniquely the mapping in regions where data are not available. In addition, the data are usually noisy. Some a priori information about the mapping is needed in order to find a unique, physically meaningful, solution. Several techniques have been developed to embed a priori knowledge in the solution of ill–posed problems, and most of them have been unified in a general theory, known as *regularization* theory [64].

Recently the basic ingredients of the design–based approach have re–appeared. Annema has focused on local linearization and elaborated this concept to the multi–dimensional learning problem [4]. Independently Meijer focused on the local linearization and proposed a new neuron function that supplied both the current working–point and the first–order derivative in this working–point [103]. Both approaches present additional insight in the initialization of a neural network but leave the issue of training untouched.

In fact, where Meijer discusses some practical experiments, he touches on some remaining questions that in our view are at the heart of the learning problem. We will present this as a reliability problem as its occurrence is not guaranteed to the naive network developer. He/she may simply be lucky, but literature often states exceptions to a well–behaved learning that are subsequently left out of further analysis. For instance, where a neural network loses knowledge (unlearning), one is simply advised to stop learning beforehand. Such by–passes prevent neural technology from industrial acceptance and must therefore be scrutinized and preferably eliminated.

The other direction is taken in the *Bayesian* approach, or also called *predictive*, approach. Its goal is to find the optimal model of $P(y|x)$. Hereby the approximation of $P(y|x)$ is done by averaging $P(y|x; \boldsymbol{w})$ over the posterior distribution for the weights w. The Bayesian approach requires introduction of so–called hyperparameters α and β. α materializes the idea of integrating out the unwanted variables, and β controls the variance of noise. In practice, Bayesian learning takes the following steps. After the hyperparameters are initialized, a standard nonlinear optimization procedure minimizes the total error function. Every few cycles the hyperparameters α and β are re–estimated after the evaluation of the Hessian matrix and its eigenvector spectrum. This procedure is repeated for different initial choices of initial weights and network models [19].

Despite the mathematical elegance, such algorithms are not yet of practical significance. Implementation efficiency is one reason but the most important one is the lack of performance criteria for industrial (i.e. only vaguely understood) problems. Therefore this thesis returns to the origin of the learning problem and attempts to characterize the internal anomalies. We have found some simple indications on whether such anomalies will be present and we will therefore be able to manipulate the data set on–line to train the network with conflict–free data untill it has mastered the model to such a degree that it can handle also the remaining data by itself. This compromises between simple & straight function approximation and cumbersome off–line data preparation and is therefore closer to industrial reality.

1.2.2 Computational drawbacks of ANNs.

Our brief introduction of the ANN method puts forward the modeling of a certain problem by the network as the major task. Compared to the human thinking process, this model resembles the process of learning, from where its name originates. The underlying design principles and mathematical methods do not allow for a straightforward proof that the learning process will finish successfully: in its search for the best solution the gradient algorithm is helped by incorporating effective randomness in an empirically chosen network architecture. In order to use the ANNs method in practice its proper and reliable functioning should be ensured. This can be done by exploring the known potential drawbacks of ANNs, which we are summarizing to:

1. **Bad replicability of the results.** The stochasticity included in the underlying algorithms can not guarantee that two experiments are exactly the same. In strict terms this means, that an experiment can not be reproduced. From a broader perspective this network property can cause an unreliable operation of the neural system.

2. **Discretization of the gradient procedure.** The mathematical proof of the correctness and convergence of the gradient descent algorithm is based on its continuity. The algorithm will draw a downhill path until the minimum is reached. When implemented, the infinitely small continuous steps shall be replaced with steps of a certain length. The size of the steps in the back–propagation version of the gradient algorithm is determined by the curvature of the hill–like surface: the larger the curvature, the larger the steps. This means, that there is a possibility that the minimum of a valley will be 'jumped' over.

3. **Locality of the gradient methods.** A large group of practical networks apply a gradient–type learning. The locality of gradient methods presumes that the search for solutions is restricted to the area around the initial search position. Thus there is a large chance for a suboptimal solution to be found. In other words the learning algorithm can converge to a suboptimal minimum.

4. **Indecisiveness of the gradient calculations.** Gradient algorithms work on the hill–climbing principle. This implies the risk that the learning algorithm can not select a travel direction when there is not a clear slope in its neighborhood. The neural networks contain simple units with equivalent functionality, which work in parallel. The equivalent structure of every processing element, as well as additional equalities (symmetries) in the neural network or in the problem

create local flatness. This potential is often underestimated in present–day theory and practice. Because it has a crucial influence on learning reliability, this problem will be placed in the focal point of this thesis.

5. **Missing information.** All the methods that should decide on the response in an unknown case while having only examples of how the problem performs in known areas, have the risk to make a wrong prediction. This *generalization* is a major issue and requires that missing information must either (a) be explicitly banned from the network operation, or (b) defaulted to values of common knowledge, or (c) tested after learning as a kind of quality evaluation.

6. **Order of example presentation.** The order in which the examples are presented will have a distinct influence on the function to be learnt. In order to eliminate this dependence, one usually resort to a random example presentation. This technique has long been fruitfully used. The drawback is evident: when the order of presentation has a meaning, this meaning will not be learnt. But also when the presentation order has no meaning, it can occur that certain orderings can not be learnt. In other words, as the order of presentation will always impact the learning of the network, it can not simply be discarded by randomization.

1.2.3 Network optimization.

When learning by error backpropagation is so problematic, one is inclined to search in other directions. One way is to improve the preparation for learning by making sure that the example set is guaranteed to give no problems. In the short history of neuro–engineering learning problems have often been attacked by sieving and re–ordering the data set. This is self–evident in the ANNIE–case, where car driving measurements are used to train a neural network for steering [113]. Clearly training while driving a straight track for a long time holds the risk to forget about the curves. Alternatively [110] shows how the quality of the data set can be enforced by a proper construction of the measurement experiment.

Recently a software tool has been commercially introduced that claims to automate the process of data preparation with guaranteed learnability [105]. It is based on the clustering of data so as to reflect the basic characteristics of the examples. Then by selecting the examples evenly from the clusters, the above discussed imbalance in presentation order is avoided.

Regretfully it is not clear beforehand what the best input features are and next what the optimal cluster sampling strategy is. For the “truck–backer–upper” it took from 1989 till 1993 before the right features became known [84]. Further [133] discusses how the balancing of primitive steering actions can be used to create lawful behavior (e.g. driving on the right side of the street can be enforced by presenting an overmass of “stay away from the wall on the left” examples).

1.3 Creating an intelligent system.

At the root of interest in neural networks lies clearly the need to research biological phenomena in building a plausible model by which experiments can be made that can raise the understanding of the biological counterpart. In the course of time the neural technology matured into an engineering area of its own right. As such, it poses new problems and/or raises the need to solve existing problems. The drawbacks cited above have all found dedicated solutions but in the growth towards an engineering discipline a more generally applicable solution is required.

In the beginning, neural networks were generally viewed as a single computational template. All research at that time looked into a single network in isolation. As applications became en vogue, the neural network was soon viewed as the central element within an overall system. In other words, the neural network became embedded within an environment and was assumed to have an almost autonomous behavior. As a system element, the fact that learning was sometimes not achieved became not acceptable. Instead, the need arose to mature the design and operation of neural networks to the level of the other system components. Consequently, the before named drawbacks of artificial neural networks have become not acceptable and therefore we have endeavored to contribute in this area.

1.3.1 Learning from reactivity.

So far we have seen that a neural network is characterized by two modes of operation: learning and recall. During the *recall phase*, data are presented to the network and it responds with the values to which it has been trained to react with during learning. This is accomplished by a scrutinuous adaption of the weights by which the relative contribution of the paths through the network are balanced. We can therefore built a mental picture of the neural network as a *reactive system* (Figure 1–7).

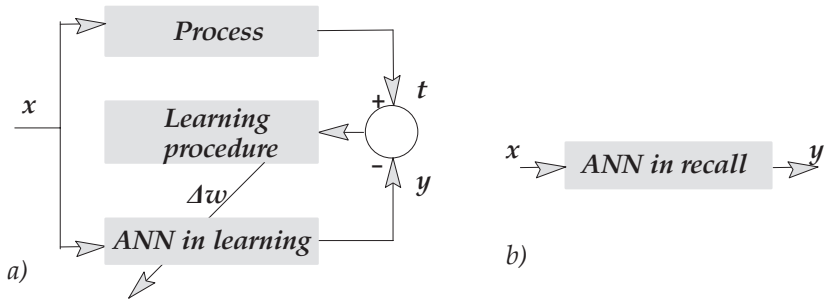


Figure 1–7: *Two neural modes of operation.*

In biological terms, this adaptation to the environment is called *plasticity* [32]. From a control theoretic point of view, the issue is rather *process identification*. The operation of the unknown process is constantly compared with the ANN till both are in agreement. As process and ANN are computationally different, we impose the requirement that any operational process must lead to a working model.

This viewpoint has similarity to what has been researched in the control of such processes. Here a range of techniques have been put forward to ensure the stability of the controled process. It has been found that such stability criteria can be mathematically formulated. Regretfully, the state–of–the–art in neural engineering has not yet advanced to a level that such criteria are available. First we have to find the parameters that influence the stability of the learning process, before such will become feasible.

Still a lot can be learned from control theory. One of the elements is the use of graphical schemes that provide inside into the nature of the stability problem. To this purpose we will use the error surface. In line with classical control theory we will have to distinguish between static and dynamic behavior.

1.3.2 Data or samples?

The original data, i.e. the data as presented to the process, and those, that emanate from the process, result from an experiment that is bounded by some natural laws. Some of them are rare, some are hard to reproduce. Henceforth, such *sample* sets are not by nature fit for learning. As a consequence, a lot of signal processing is required before they have become presentable.

In a first stage, it may be required to filter the data to get rid of spurious elements such as noise, but also of other samples that have no bearing on the task to be learned. In general, the neural network may be targeted to identify only part of the process and therefore the sample set must be cleaned to avoid overparameterization.

In a second stage, it may be required to transform the data to dimensions that are more fit for learning. Sometimes, a notation in the frequency domain instead of in the time domain makes the learning easier. It is only here, that we use the term *data* instead of "samples" as only now have we arrived at information in a representation that is meaningful for the neural network to be learned.

In a third stage, it may be required to reorder the samples to allow for learning. This can be handled off–line, but in the context of this thesis we assume the ability to do this on–line. An input/output pair of samples, presented to the network, gives an example for the problem at this particular point. This is the reason to call a sample, when presented to the network during training, an *example*.

1.3.3 This thesis.

There is a long history in neural network research on developing robust learning algorithms. These are always targeted on one or more of the above stated problems. In this thesis, we take the position that the learning algorithm is not the cause of the problems and will therefore not give a cure. Instead we will study the internal operation of a learning network to disclose the fundamental problems for non–learning in any degree. The rumor that neural networks take a long time to learn can to a large degree be explained by pure ignorance about the internal system operation. Even when a system converges during learning it may be involved with so many problems that it simply takes a long time to finish the assigned job.

As obviously the quality of learning is at stake we will focus in chapter 2 on a universe of discourse that allows to discuss exactly the problem at hand. To this means we will

use the *error surface*: a graphical presentation of the interaction between the network and the problem to be learned. We will discuss this error surface in terms of design quality and product reliability. Here, we can identify the complexity of the problem and the various factors that are involved in getting a neural network that not only learns as desired but also does that every time learning will be attempted.

The use of the error surface is further elaborated in chapter 3, where we introduce indecisiveness in walking around the surface and reproducibility of such walks. It shows a number of effects that have been noted in literature and have usually led to the creation of different topologies and/or algorithms. We will rather conclude that such measures can never provide an overall solution as they do only address part of the problem. Therefore the remaining discussion will be based on the KRS model, which reflects the interaction between knowledge, randomness and symmetry.

In chapter 4 we provide a detailed presentation of the different ways in which symmetry effects play a conflicting role in the learning of neural networks. The basic phenomenon is studied by the simple occurrence of symmetry; then more complex situations are discussed. From this analysis we derive the KRS–ratio: a novel way to monitor the occurrence of learning problems from an external view on the network behavior and prepare for experiments that will illustrate the impact for old and new techniques in the designated problem areas.

Next, in chapter 5, various ways to sample input patterns for presentation of the network will be reviewed. Using the above acquired insight on how problems can be easily identified during learning, we will propose a variation on an existing technique, the *active sampling*, to solve the learning reliability problem. Then, some real–world problems are tackled and it is shown that the mentioned problems do occur in practice, though they generally go unnoticed. By applying the new technique we will train this network better and faster than before. This again illustrates that reliable learning of a neural network involves more than improving the learning algorithm.

2 Learning reliability.

There are many aspects of quality that influence the production process. One of them is the reliability of the product to perform as intended at all times. In this chapter we will review what reliability means for a neural functionality. Reliability shows in fault–tolerance and provides for generalization. It is achieved by the presence of redundancy and can only be marked, once learning is completed. The potential trajectories on the error surface indicate a network sensitivity that underlies fault–tolerance and are therefore better suited for on–line manipulation. An approach for reliability enhancement based on finding a optimal learning trajectory is suggested.

System is a widely used term in everyday's life, as there are biological, ecological, electrical, financial, social etc. systems. In a technical sense, *system* is the cover–all name for an assembly of collaborative parts from a variety of technological origins as a single complex part. This thesis focuses on artificial neural networks (ANN); especially on methods to derive and evaluate the quality of designs for neural systems. Neural systems can be built from either software (SW) or hardware (HW). Though eventually dedicated hardware may be used to increase performance, software will often remain the enabling factor for immediate product innovation. It directly describes the desired functionality on any platform and is therefore a suitable view on system quality. In end effect, a (neural) system will be a judicious mixture of hardware and software; a number of the notions and definitions, developed in this chapter, will therefore relate directly to their origin in hardware and software.

A system has a distinct quality: a fuzzy or hard attribute that quantifies in how far the product lives up to the expectations. In [31], a very coarse definition for quality is given as being based on a compromise between specification and realization:

Def. 2−1: The *quality* of a product is the degree of conformance to applicable specifications and workmanship standards.

This is clearly a simplified rendering of a very complex world, as both specification and realization have many aspects that all come to bear within the single product. As indicated in figure 2–1, quality of design involves capability, usability, performance, installability, documentation, reliability and stability. On the other hand, manufacturers have to deal with four related aspects: time to market, cost, repair and maintenance. And they have to find a balance between all these quality marks with the knowledge that maintenance costs are more than 50% of the total product cost.

Fundamentally we can discern two lines of thought in these lists. Firstly, one has to do with specification. The quality of the product relates then to the way it will be perceived by the potential customer; if it is viewed as a better product than others, it has apparently a higher quality. The second line of thought has to do with the development process: how does one achieve the specification in terms of maturization, robustness and overall stability. Some confusion may arise from the fact that this can all be called reliability. Eventually it will add up to the cost of design and manufacture and we will therefore use the different meanings of reliability irrespectively.

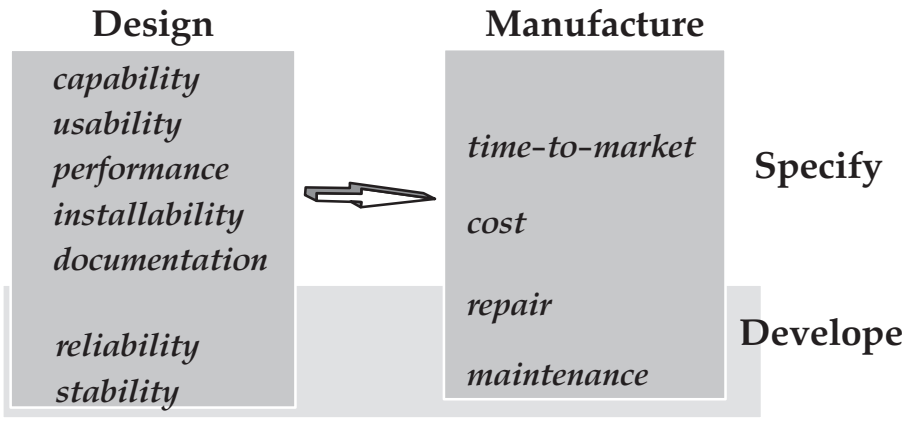


Figure 2–1: *Different aspects of quality (after [87]).*

Reliability has been studied both from a hardware and a software perspective. On first sight, different notions are used stressing different meaning of the word. For discussing the reliability of neural networks, a more generalized view must be taken, while allowing for the ever changing balance between hardware and software within a system.

> ***In RADC–TR–85–37 ("Impact of hardware/software on system reliability", January 85) is written: "The reliability of hardware components in Air Force computer systems has improved to a point where software reliability is becoming the major factor in determining the overall system reliability".***

In this chapter, we will first relate coarsely the notions and definitions, as popularized within hardware and software, to the needs for studying the reliability of artificial neural networks. Then we pursue in more detail the use of reliability measures for neural systems to identify the need for learning reliability. We will elaborate on the use of the error surface for discussing design issues in neuro–engineering and define learning reliability as finding an optimal trajectory on the error surface. Finally we venture some remarks on methods to enhance the reliability in a well–behaved, structured, manner.

2.1 Reliability in neural learning.

The computational drawbacks from Section 1.2.2 imply that, though a network may seem to learn certain problems, this learning may still be of poor quality, i.e. the process never converges or even proves to be off–target. Since learning is the process to change the network from an initial state to a system solution, the quality of the model (the network) will depend on the quality of the modeling process i.e. of learning. Then terms as learning quality and learning reliability also need to be defined.

We will start by shortly reviewing the notions and definitions as have been introduced over time in both the field of hardware and the field of software. Our definitions will largely be based on the commonalities. This enables us to introduce with slight changes in interpretation these same notions and definitions for use with neural networks. Then we will link these areas into a common universe of discourse.

2.1.1 Notions and definitions.

The reliability of a system can be decreased by various factors, such as incorrect operation of system components, influence of noise, design shortcomings, or changes in the environment. An extensive literature discusses such un–anticipated behavior by distinguishing between errors, faults and failures, but these notions require a more precise definition to let the fine differentiation in meaning be fully understood. We cite the following definitions from [109]:

Def. 2−2: **An *error* is a discrepancy between a computed, observed or measured (in other words real) value and the true, specified or theoretically correct (in other words intended) value.**

For instance, a "production error" indicates that the fabrication of the circuit has not been flawless, while a "human error" indicates the maleficent influence of a human designer. Errors are concept–oriented.

Def. 2−3: **A *fault* is a specific manifestation of an error. In this sense, it stipulates an accidental condition that causes a unit to fail its required function.**

For instance, a "stuck–at fault" shows the impact of a production error on the logic behavior of a hardware part, while an "operation fault" reveals an embedded mistake in some software. Faults are developer–oriented.

Def. 2−4: **A *failure* may be caused by several faults and shows the inability of a system (component) to perform a required function within specified limits.**

Failure refers to what happens when one or more faults get triggered to cause the program to operate in another way as intended. Therefore some also specify failure as a fault–effect. For instance, a "system failure" is caused by a concrete fault, even when the error message (or rather failure message) is incomprehensible. Failures are customer–oriented.

In other words, we can relate error, fault and failure in the following way:

> ***An error made in product design / development can cause a fault within the specification / code that produces a failure of the system.***

In reliability engineering, this cause–effect relationship re–appears in the different approaches for qualification: (a) sensitivity, (b) tolerance, and (c) redundancy. In this order, one studies the sensitivity for errors to become faults, the tolerance for faults to become failures and finally the impact of failures to bring down the overall system performance.

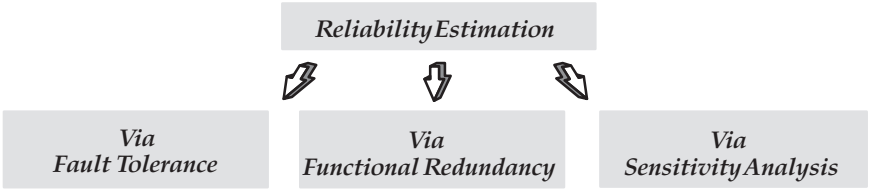


Figure 2–2: *Approaches for reliability estimation.*

In literature, the notions of reliability and fault tolerance are sometimes used interchangeably. Fault tolerance is one possible way to estimate reliability, as reflected in figure 2–2.

2.1.1.1 Sensitivity.

In *sensitivity analysis* it is investigated what the contribution of a specific fault is on system failure. It is based on some metric that enumerates under which input conditions (*controlability*) an effect can be found on the outputs (*observability*). A constructive way to use this metric is called *design centering*, where the system (component) is designed at a central location with respect to the acceptance conditions.

Clearly, a system will not be equally sensitive to all faults under all state conditions, while also not all faults are equally probable. So, in the more general setting, faults can be characterized by an *operational profile*: the collection of system stimuli that may excitate the fault, when present.

Def. 2−5: ***Fault sensitivity*** **quantifies the impact of a fault on at least one system output under well−defined input conditions.**

Sensitivity analysis assumes that controllability and observability will never be zero, as this would mark the untestable case. In practice, not all network nodes are testable, as the information capacity will generally be under–used. An error will only produce a noticeable effect, when it influences that part of the structure that is actually used to implement the intended behavior; otherwise, the error will not lead to a fault.

Degradation is the decrease of network functionality due to an error; not every error will lead to degradation.

2.1.1.2 Tolerance.

Even when an error produces a fault, this does not automatically imply a failure. The influence can also be masked away by the presence of redundant information: information that supports already existing behavior. For instance, the bridging terms in a Boolean expression make the implementation somewhat tolerant to faults. Methods which perform this task are termed *fault tolerance* techniques.

Def. 2−6: ***Fault tolerance*** **is the robustness of the network function in the presence of degradation in the network computational elements.**

Fault tolerance implies techniques for improving reliability. It is possible for faults to exist during system operation, while the reliability of the system is not decreased. This can for instance be due to the potential of the system for self–repair. In coding theory, one then distinguishes between fault–detecting and fault–repairing codes.

On the other hand, reliability can be diminished not only by faults, but also by that the system simply does not meet its specification. Hence fault tolerance characterizes how the system behaves, as faults are introduced in it. A high fault tolerance indicates, that faults will not affect the system performance a lot.

An important property of a system, related to fault tolerance, is known as *graceful degradation*. Graceful degradation is the ability of a system to provide useful service in the presence of faults. Where system quality relates directly to the density of faults, it is only the combination with the operational profile that results in failure intensity, the measure for system reliability.

2.1.1.3 Redundancy.

Degradation can be avoided by judiciously increasing the *redundancy* within the system. *Spatial redundancy* refers to duplicating the functions of groups of physical components, in order to increase the systems computational capacity. An example for performing spatial redundancy is *N–modular redundancy* technique. *Temporal redundancy* involves solving a sub–problem many times, and using the results to construct some form of average final solution, in which an individual failing part may go unnoticed.

2.1.2 Neural system reliability.

Over the years the interest in quality measures for neural networks has largely been confined to the enumeration of the information storage capacity. The neural network is viewed as an *associative map* so that the operational quality restricts the Hamming distance between the stored patterns. From an approximate count of the number of patterns that can be stored follows the maximal affordable Hamming distance. The nature of the pattern set then shows whether such a Hamming distance provides the required characteristics.

For many neural networks such an associative capacity model does not exist; the best one can hope for is a worst–case boundary value, which is normally way beyond a realistic value. Moreover the associative capacity model is not applicable in a number of cases, such as in function approximation, classification and prediction. To estimate the quality and particularly the reliability for these tasks, relevant performance aspects will be determined first. The methodology for estimating and comparing the performance aspects will be categorized in accordance with the "classical" reliability scheme. At last the different methodologies will be compared to motivate the choice of reliability enhancement method, used in this thesis.

2.1.2.1 Performance aspects.

Learning reliability can be defined intuitively as the ability of networks to do always what they are supposed to do. In this sense reliability implies all the quality components from Def. 2–1: the accuracy of mapping, the success of the convergence and the generalization ability, and adds to them the degree of ensurance, that these factors will be satisfied when different experiments are done. The Electronic Industries Association (EIA) defines reliability as [31]:

Def. 2−7: ***Reliability*** **is the probability of a device to perform its purpose adequately for the period of time intended under operating conditions encountered.**

This definition concerns the operational device i.e. within the neural network's paradigm that concerns the learning network. It implies four important subdefinitions, which should be fulfilled simultaneously. First, reliability is a *probability*, i.e. the number of times an event occurs out of a total number of trials. Mapping of this subdefinition on the ANN paradigm is unambiguous – it should calculate the probability that a network converges to the desired solution. A practical way to calculate this probability is to count the number of successful experiments out of all experiments made.

Second, the *adequacy* implies criteria to be established, that clearly specify, describe or define what is considered to be an adequate performance. In the neural network para-

digm there are a number of different criteria, which can define when the performance is satisfactory. These criteria are called *stopping rules*, because they indicate the moment, when the learning process should be stopped. The earliest idea for such a rule is to stop when the error E becomes small. This rule is useful for many applications, but can lead to poor generalization. Later on many ad–hoc stopping rules have been created. The most popular one is to have a validation set and to stop training when the error measure on the validation set starts to raise.

No matter which of the stopping criteria is used, the goal of the neural learning process is the achievement of the required *mapping accuracy*. The potential risk in fulfilling this adequacy criterion is derived from the locality of gradient algorithms: the gradient algorithm can not reach the minimum value which satisfies the value predefined by the stopping rule accuracy.

The *time*, or the measure of the period during which a certain degree of performance can be achieved, is another important reliability factor. Having no information of how long it can take for the network to converge to the satisfying solution, it is not well defined how to assess the probability (the number of trials), that the goal will be achieved. This reliability criterion seems to be inapplicable for neural networks, which are known to have a long convergence time, if they converge at all. In [10] it is shown, that the convergence time and the probability for convergence are closely related. This property is not common for every "device" in the sense of the given reliability definition. Because of that, we suggest for neural networks the term *convergence (learning) success* to notify both the probability for an experiment to converge and the convergence to happen within certain time constraints.

It can be said that the convergence (learning) is successful, when the chosen stopping criterion is satisfied. The success of convergence (learning) refers to one single experiment. Guaranteeing the convergence of the single experiment presumes that the underlying mechanisms are under control and thus every single experiment can be brought to the desired ending. In this sense ensurance of the learning (convergence) success will be the basic step towards investigating the reliability and learning quality in general.

The fourth element in the EIA definition for reliability are the operational conditions, under which a device is expected to function. It implies conditions as humidity, shock, vibration, etc. We find this aspect of the definition inapplicable for neural networks, because it does not imply any risks for their performance in the computational sense. Instead, we will discuss further some specific critical aspects of neural learning.

2.1.2.2 Reliability assessment.

According to [52], there are three well–developed strategies of measuring a neural network's performance – via *fault tolerance*, via *generalization abilities* and via *learning trajectory and speed* (Figure 2–3). Respectively, there are methods developed for reliability enhancement, based on this performance metrics. To be in line with the categorization, summarized in section 2.1.1, we attempt to find the correspondence between the "classical" and the established neural reliability estimation methods. Since fault tolerance has a direct correspondence with the scheme, given in figure 2–2, the relation of the other two reliability estimation methods with the classical framework has to be explained.

Estimation of reliability via network generalization abilities implies evaluating the performance of the trained network on unseen data sets. In [52], a number of techniques

are suggested that improve the generalization performance of the network, among which are: changing the network parameters, hidden units, hidden layers, used stopping criteria etc.. Systemizing all these techniques for reliability enhancement makes obvious their relation to the functional redundancy approach from conventional reliability research. Further we take a more general view on this reliability enhancement method through the prism of the redundancy approach. Conceptually, neural reliability enhancement based on optimized trajectory and speed, is close to sensitivity analysis. Therefore, neural reliability estimation and enhancement methods can be specified as shown in figure 2–3.

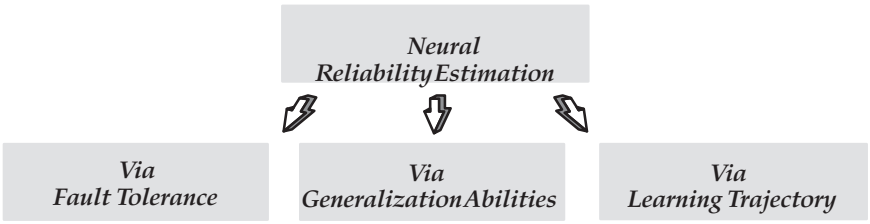


Figure 2–3: *Approaches for neural reliability estimation.*

2.1.3 Fault tolerance in neural networks.

Aspects of fault–tolerance have already been discussed in the previous section to highlight the differences with classical HW/SW; here we continue with the more neural–specific aspects. Though the reliability of neural networks is a widely discussed topic, there is little consensus on how to investigate this area. The three main research lines have been on (a) resilience to faults, (b) robustness to noisy inputs and (c) equality of results when relearning. In early literature, the first two lines have often been mixed. Overall this research aims to qualify reliability by a quantification of the network fault tolerance [52]. Therefore we will shortly review some of its results before progressing towards alternative approaches.

Investigating the reliability of neural system through the analysis of its fault tolerance goes through the following major phases. First, the fault model must be specified, stating the amount and nature of the faults to which the network may be sensitive. Based on this model a reliability measure, suitable for assessing the fault tolerance, is established next. Following the concept of redundancy, which is substantial for developing a fault tolerant system, the empirical data for calculating the reliability measure is gathered. This way, the fault tolerance of the network is determined.

2.1.3.1 Fault models.

On any abstraction level in the design space, *fault models* can be formulated. In coarse subdivision, one distinguishes between *physical faults* for electronic circuits, *logic faults* for the digital circuits and *functional faults* for any higher level. It is debatable whether the latter are not in fact failures. It is also here that models to characterize software behavior have been introduced.

Fault models have largely been derived for hardware realization; in [145] this line of thought is extended for neural networks. Here, we can coarsely discern between neurons, connections and the adaption of weights (Figure 2–4). For connections, the con-

ventional stuck–at assumption can be used: if the connection is either short–circuited or broken, unreliable performance can be expected. Neuron faults can largely be seen as differences in the transfer function: a linear function appears to be a strong limiter. The adaptation fault will largely be concerned with the effectiveness of training the weights.

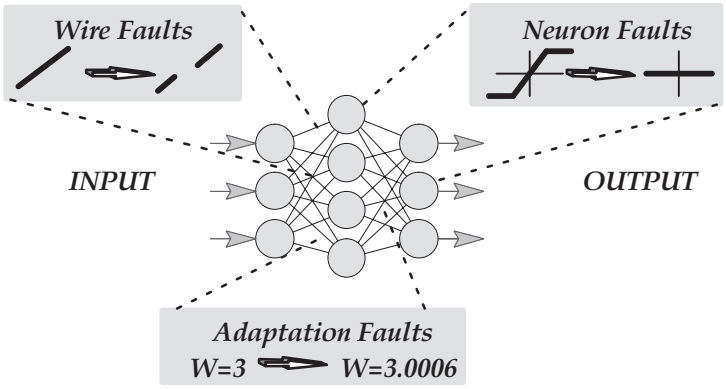


Figure 2–4: *Faults in a neural network.*

Next to these, we have to distinguish between static and dynamic faults. *Static faults* appear independent of the internal or external timing of the network for the handling of the examples. Such faults are usually of the stuck–at or the stuck–on type. However, on a higher level of abstraction static faults can also occur because of value and scope restrictions. As such, the characteristics of the hardware platform creep in on the software level.

Dynamic faults are commonly of the delay–type: different timing characteristics on the hardware nodes may under circumstances influence the neural behavior. Especially in analog hardware with stored weights, such a timing effect can occur. On a more abstract level, timing has also to do with the presentation order of the examples. It is established that dynamic faults can best be analyzed through the use of benchmark tests, as suggested in [140] and [145] (Figure 2–5).

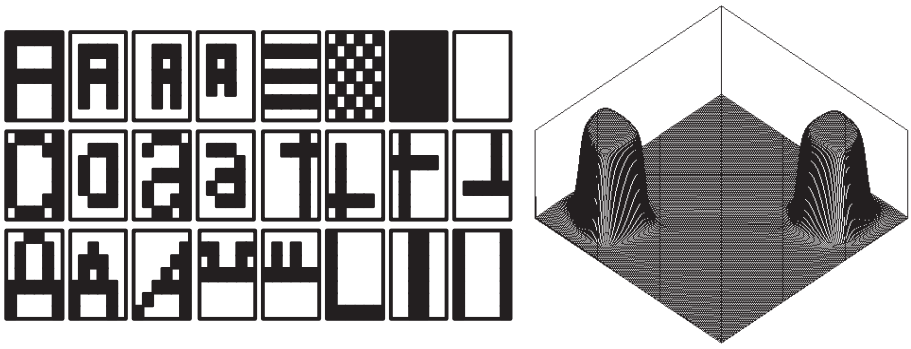


Figure 2–5: *Some benchmark tests (from [140]).*

2.1.3.2 Failure models

In [16] and [153] one distinguishes between the reliability of a neural network (a) when operating as associative memory or classification system, (b) when operating as a function approximator and (c) when the network is used to solve an optimization problem. In the first case the common technique is to evaluate the sample probability that a pattern will be recalled or classified correctly for various fault levels; this leads to a dedicated *failure model*.

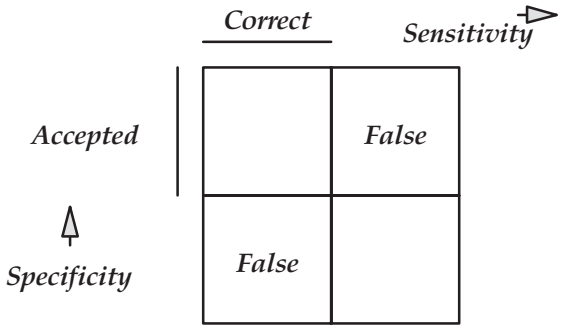


Figure 2–6: *Classification areas.*

In solving classification problems, a continuous threshold function should determine the belonging of a pattern to a class. The output values are giving an approximate (with a certain degree of trust) indication whether an answer belongs to one of the available classes (Figure 2–6). The reliability measure for a classification task should either give the degree of certainty in the network response to answers (*sensitivity*), or the uncertainty in the network classification in the right classes (*specificity*). The difficulty to create such a measure is that the response of the network can be uncertain not because of the fault or failure in the network operation, but because of an input, which does not belong to any of the available classes. Although the measures for reliability and for fault tolerance differ in the sense of section 2.1.1, in practice, both types of measures are used interchangeable [52].

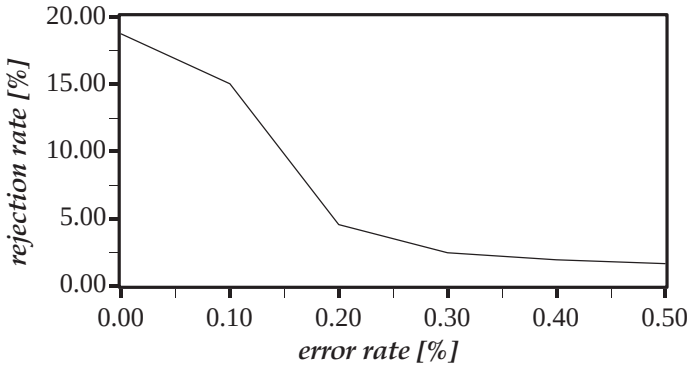


Figure 2–7: *Error–reject curve for a neural network classifier (from [28]).*

Over the wide range of applications, there will be large differences in qualifying such reliability measures. For instance, for the recognition of license plates false acceptance cannot be allowed as such will entice correctly classified persons to start law–suits [28]. On the other hand, in the case of automated cleaning of laundry it is false rejection that will cause problems. Here one rather passes laundry with a last bit of remaining dirt than to re–wash something that is already clean [143]. By setting a threshold on the *error–reject curve* (Figure 2–7), this decision whether to have a low error rate with many rejects or a relatively high error rate with few rejects can be established.

Conversely, a continuous measure of the deviation from the desired function is more appropriate for function approximation tasks [33] [137]. A similar approach is used in [122] to evaluate the outcome of a neural network when solving an optimization problem. The experiments in this thesis concern mainly with function approximation problems. This is the reason that a special place is devoted to the reliability of feedforward neural networks during the learning phase.

The task of assuring reliability in continuous function approximation appears frequently in domains of control systems and function approximation. [137] proposes a measure and compares the reliability of different networks with respect to their fault tolerance when performing a function approximation task. The authors take two aspects into account. The first one is finding a reasonable approximation error measure. The second is to formulate a suitable faulting method. Measuring fault tolerance is then equivalent to applying the faulting method and observing the effect on the measured error.

2.1.3.3 Suitability of reliability analysis.

Because of the massively parallel architecture and the distributed storage of the information between the computational elements, an intrinsic fault tolerance have been claimed for neural networks at its re–birth in the mid 80's. An extensive research in fault tolerance challenges this claim [24] [52] [114] [137] [153]. The main conclusion is, that neural networks have the potential to be fault tolerant, but the learning algorithm used (such as backpropagation) does not always generate the right weight configuration for it. Correspondingly, fault tolerance can not be assumed, but should be posed as a problem to solve.

The impacts of faults is different during the operational phases of learning and recall [33]. During learning, the network can be trained to be admissible to faults: a degree of embedded redundancy may be activated to compensate for the missing functionality. However, once the network is trained, the operation will have a similar vulnerability as in conventional electronic systems. The study of network tolerance during learning can be based on a suitable selection of the potential fault models by sensitivity analysis [145]. We will point out later that this is only part of the reliability problem. The use of fault tolerance as a stepping stone precludes to take other effects on reliability into account.

2.2 Reliability as optimal trajectory.

In section 2.1 several aspects of the reliability of the neural networks have been considered: (a) the ability to learn the underlying mapping with given accuracy; (b) the potential of the learning process to converge and the convergence duration to be kept within

reasonable borders; (c) the probability of a new experiment to have a guaranteed success; (d) the guarantee for network performance on unseen examples (generalization capability).

In search for an unified approach to the posed problems a successful development of the learning process on a single experiment will be thought of. The accuracy required to claim, that the learning process has been successful and that the ability of the network to reach success in every experiment (i. e. the reliability and generalization of learning) will be fulfilled if the mechanisms to bring a single experiment to the desired ending are understood and ensured.

The success of neural learning affirms the ability of a designed neural system to model a certain problem. Correspondingly, the learning success factors can be divided naturally in two groups: factors concerning the neural system (architecture, initialization and algorithm) and factors concerning the problem definition and representation. An useful abstraction, which connects these two groups of learning factors, is the error landscape, which gives the overall difference between the network response on the current example question and the right answer.

If the problem is represented only by question–response coupled data, then the learning process is directed by the data, i.e. neural learning is a *data–driven* task. To represent the steps taken to solve this data–driven task it is enough to draw its trajectory on the error surface. The error surface abstraction is used for both: to understand better the influence of learning factors on the learning success and to explain how learning progresses in time.

2.2.1 The error landscape paradigm.

The error landscape paradigm is attractive for the following analysis, because it naturally combines the network and the problem factors of the learning process. Also, it gives a clear idea of how the different factors can be represented graphically, and thus helps imagining how the different learning factors will influence the learning process – the landscape relief or the itinerary on it. As said before, the features of the network and the problem learning factors determine the learning quality. Correspondingly, the graphical presentability of the error landscape is expected to give a deeper insight of how the learning quality can be achieved.

The error surface is usually thought of as containing hills, plateaus, basins, and valleys. Going downhill should find the lowest point of the error surface, which in the neural literature is known as the global minimum. In the ideal case this is a point on the error surface with zero energy for every example from the training set. In practice, a good enough solution is found – a point with low average error $\mathcal{E}_{av}$. In an attempt to explain how the error surface depends on the different neural factors, the last can be divided into factors, which form directly the relief of the error surface, and factors, which choose the travel trajectory.

The training error for a particular network depends (a) on the inputs x, outputs, y, which form the problem representation and (b) on the network parameters w and architecture f – i. e. on the network. The learning error surface minimum is expected to be also a good minimum of the generalization error – the real measure of the performance of the neural model.

2.2.1.1 Reliable learning trajectory.

The previous approaches are estimating the reliability of a neural system on basis of the observable effects of the learning process. They search for structures and parametrisation for which it is most probable to give good results. A third reliability estimation method analyses how the learning process progresses. As shown in Section 1.2, the objective in training is the minimization of the difference between the given answers by the network and the actual answers to the asked questions. For every training example this difference is represented as a training error. The overall difference forms the error surface.

The development of the training process can be expressed as a path along the surface, called *trajectory*, which will lead to its optimal (minimal) point. Every point of this trajectory is uniquely defined by the network parameters (weights), and the learning paramethers (learning rate and momentum). The reliability analysis method, based on the learning trajectory, analyses the shift weight vector across the error surface. The corresponding reliability enhancement techniques attempt to adapt the weights in an efficient manner.

Since this method gives insights on the learning process itself, it aims to improve the learned solution, i.e. trajectory and the rate at which the result is achieved. In contrast to the other reviewed reliability enhancement methods, the one of assessing the quality of the learning trajectory and speed bases its results on understanding the intrinsic properties of the training process. Correspondingly, only this method can give a good insight on how to improve the network performance on–line, i.e. when the training problem becomes apparent.

There are many known techniques which attempt to ensure an optimal learning trajectory. The simplest method to find a better trajectory is simply to restart the training process and hope for the best. The developed techniques for finding a best trajectory belong to one of the following areas:

- *Local adaptation rates.* The idea behind this group of techniques is that if the weight detects a different error surface area with different gradients around the current error surface position, the new adaptation rate should be applied. This way every weight has a corresponding adaptation rate, depending on the local shape of the error surface. Local adaptation of the weights is often based on heuristics.
- *Second–order methods.* This group of methods use the second–order features of the error surface. The path towards the optimum solution is searched by considering both the curvature as well as the gradient at a certain point. The curvature of the error surface is evaluated by calculating the Hessian matrix and assuming that the surface is locally quadratic. Computationally, this is a very expensive step, even for a small problems. Application of this method in its purest form is practically impossible for real–life applications.
- *Regularization methods*. Globally said, this group of methods aims to smooth a very curvacious error surface areas and to introduce a higher curvature in very flat areas by introducing an extra term in the optimization criteria. This way they change the task of learning. These techniques are founded on the assumption that the new task (the new error surface) has an optimum at the same point as the original one.

- *Conjugate direction methods*. These methods calculate a set of N orthogonal directions in N–dimensional space. The search, made in every of these directions, is linear. If the surface is locally quadratic, N line minimizations will find the minimum point. Although the assumption that the surface is locally quadratic is not always correct, the conjugate gradient techniques are reported to give good results in a number of applications. The difficult point in the conjugate gradient techniques is finding the N orthogonal directions. There is a large variety of algorithms for finding the orthogonal directions; however, they are problem–specific and imply restart heuristics to find a solution.

Globally speaking, reliability estimation based on the learning trajectory has its weak points. The major one is that it is difficult to give definitions about the quality of a learning trajectory and the speed of learning, since the neural optimization process is performed by a data–driven randomized algorithm. The intuitive method for measuring the quality of the learning trajectory is by the time interval in which convergence to an acceptable solution is reached, i.e. by the length of its trajectory. A widely used realization of this method is by counting the number of epochs taken by the network to converge to a given solution.

The reliability estimation approaches as reviewed so far consider different aspects of the neural operation. The fault tolerance approach refers to the resistance of the neural method to malfunction of one or more network components. The functional redundancy approaches study how the diversification in sort of the neural system will influence its performance. Finally the learning trajectory approach analyses the reliability performance in every learning case. There are reliability enhancement techniques, related to every of these approaches.

2.2.1.2 Influences on neural reliability.

Quality of learning is understood as the accuracy, operational time, reliability and generalization ability of a given system when learning a certain problem, if theoretically this system can learn the problem. Thus the learning quality depends as much on the network (the system carrying out the learning process) as on the problem to be learned. Here, the network and the problem components, which influence directly the learning process, will be substantiated and their impact will be shown.

Directly related to the structural aspects of a neural network are the well–known threesome: network architecture, learning algorithm and parameters initialization. These factors provide for the principle outlook of the error surface.

- The *network architecture* includes neurons, activation functions, layers and their interconnectivity. The backpropagation algorithm, considered in most of the further described experiments, is applied most often on the feedforward architecture with hidden layers. A too small number of neurons in a hidden layer restricts the network capacity such that the problem can not be grasped. Too much hidden neurons brings the network to take more, often ill–defined, data relations into account or even to memorization. This not only slows down the operation but also leads to deteriorated generalization. In the next chapter this topic will be discussed in more detail.
- The *learning algorithm*. The gradient algorithms, widely used in the training of neural networks, have the problem of locality of search – they perform

minimization in the local landscape, where they are positioned. In a global view the optimality of their performance can not be guaranteed.

- The *initial network parameters* is a summary term for initial weights and biases configuration, initial learning rate and learning history (momentum) parameters. Since the gradient training algorithms have local action, the state from which they begin is of importance. In cases where prior information is available, it should be used to choose the initial parameters of the network. Lee [97] claims that a wrong choice of initial parameters is the reason for *premature saturation* phenomena. Although in the next chapters a better description of this phenomena will be given, this main conclusion is valid to a large extent.

The influence of the different training sequences on the error surface stems from the training set content and from the presentation procedure. Different training sets and/or different presentation procedures will affect the quality of the eventual neural system.

- The *training set content* contains information on the signal (problem to learn), the way of sampling (uniform or with higher sample density of the difficult areas). The impact of problem representation w.r.t. the improved quality of learning is a widely discussed subject in this thesis.

- The *training set presentation* will be elaborated on in terms of a crucial learning quality factor. The most common observations are that for improving the speed of convergence the training examples should be presented in a random order, and after every epoch a new randomization should be done . The training efficiency can be improved by applying different ways of example presentation [10] [14] [43] [39] [40] [80]. This subject will be widely discussed in this thesis.

2.2.1.3 Correspondence with learning factors.

As explained before, the problem to learn and the system that performs learning intersect in the error surface paradigm. Further on, the correspondence between error surface and learning success will be explained for passive and active learning schemes. Active and passive learning are terms reflecting not so much the learning process in general, but rather the way of pattern selection and its contributions to the learning success.

The factors that condition the learning success are either elements of the neural system or of the problem to learn. The error surface paradigm connects these two sides in a natural way. Even more, it gives a better understanding of their influence on the learning success. Within the framework of error surfaces the learning success factors can be classified in two other groups (see figure 2–8):

- *static factors*, which form the relief of the error surface. To this group belong network architecture, initial state of the network, and training set content.

- *dynamic factors*, which direct the itinerary on it. This group implies the learning algorithm and the training set presentation.

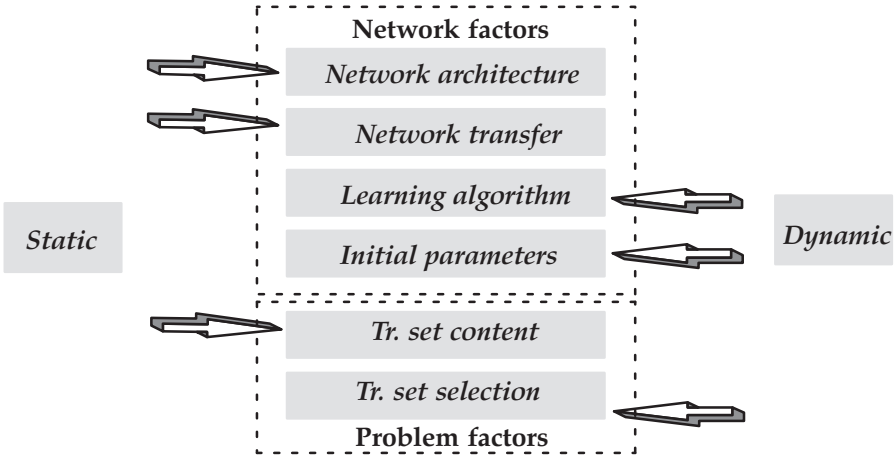


Figure 2–8: *Classification of the neural components w.r.t. network dynamics.*

The passive learning scheme is characterized by a training set with constant content of N input–output pairs $\{z^\mu \equiv (\boldsymbol{x}^\mu, y^\mu) : \mu = 1, 2, ...N\}$. The difference between the network response and the actual knowledge is determined by the error function. For the μ–th training example the error function is:

$$e(z^\mu, \boldsymbol{w}) = y^\mu - f(x^\mu, \boldsymbol{w}) \tag{2–1}$$

The instantaneous squared error in the case of one output neuron is:

$$\mathcal{E}(z^\mu) = 1/2e^2(z^\mu, \boldsymbol{w}) \tag{2–2}$$

The weights of the network are adjusted to minimize the total sum of the squared error between the actual and desired mapping at the example points. The total sum of squared error criterion is obtained by summing $\mathcal{E}(z^\mu)$ over all N training patterns and is denoted by E_{tot}.

$$E_{tot} = \frac{1}{2}\sum_{\mu=1}^{N} \mathcal{E}(z^\mu) = \frac{1}{2}\sum_{\mu=1}^{N} [y^\mu - f(x^\mu, \boldsymbol{w})]^2 \tag{2–3}$$

The normalization of the total training error E_{tot} over the number of patterns makes possible the results of training different signals to be compared and is by far the most common choice of optimization criterion:

$$E = \frac{1}{N}\sum_{\mu=1}^{N} [y^\mu - f(x^\mu)]^2 \tag{2–4}$$

From equation (2–4) it can be concluded, that the error function depends on the *training patterns*, the *network parameters*, and the *network function*. Designing a particular network fixes the *network function*. Defining the problem to learn is done by choosing a training set z^μ. If the training set is fixed, as the passive learning scheme suggests, the training error is a function of the network parameters only. Varying the network parameters with fixed training set (the network function is always fixed) gives all possible values of the network output. Correspondingly the error function defined for all possible

parameters values forms the error surface. The number of parameters defines the dimensionality of this surface.

The error surface is multi–dimensional. It is practically impossible to be plotted for dimensions higher than 3 (i.e. the network with more than two weights). For most practically used networks this is a large restriction. Instead of the error surface itself, a two–dimensional subspace of it can be plotted successfully. Selecting a pair of weights to be varied, all the others are kept fixed.

Every training set defines its own error surface (Figure 2–9). While the training set content defines the relief of the error surface, the way of pattern presentation draws the trajectory of learning process, beginning from one arbitrarily chosen, defined by the initial parameters, position.

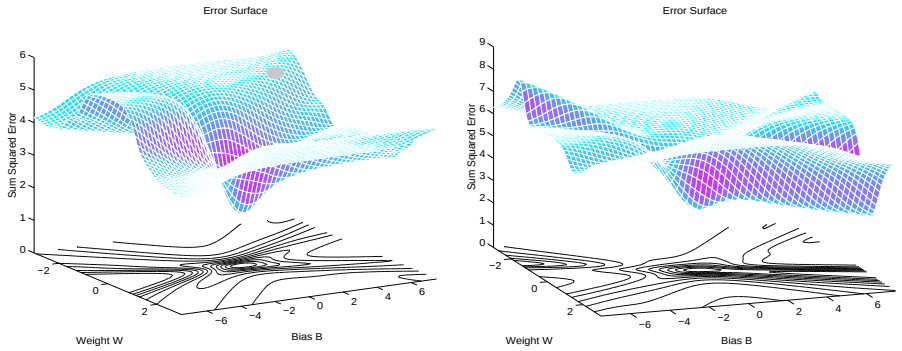


Figure 2–9: *Error surfaces, formed by applying different training sets to the same network.*

It should be noted, that the actual error surface, or also called *generalization surface* is unknown, since the problem to be learned is represented by the number of examples, which presumably give sufficient information about the actual problem by modeling the underlying generator of data. Since the actual data distribution (X, Y) is unknown, the generalization error can not be determined. Instead, it should be estimated from the training set. The defined error function is the simplest adaptation of the generalization error criterion to error criterion for a finite training set of N examples.

Insufficient information on some areas of a problem creates a difference between the learning and the generalization surfaces as well. To make this statement more clear, let us remember which factors form an error relief: the network architecture, the initial parameters and the training set contents. In practice, every problem is partly undefined: there are areas of the problem for which there is no information available. If these areas are defined, new data should be generated. Additional or different data will form a different error relief. This relief will correspond to the generalization error surface. In general, it can be said that the generalization error surface is smoother than the learning error surface. At the same time it may have more relief forms than the learning error surface, if some of the problem features are not represented in the available examples.

This is the reason why every training set representing the same problem will have its own learning error surface. It is assumed, that if the training set has the same unknown distribution as the actual unknown distribution of the problem, the minimum of the

learning error surface will be approximately equal to the minimum of the actual (generalization) error surface. The last elaborations make it easy to understand the difference between active and passive error surfaces.

Active learning concerns not the internal mechanisms of the learning process, but the way the learning problem is presented to the network. There are two groups of methods: the first one defines the strategy of pattern presentation, and the second – the selection of the best training set from all available examples. By both methods the training set is presented in a specific manner and it is not necessarily of a constant content and distributedness.

Mapping these concepts on the error surface paradigm, a substantial difference can be noted. By active sampling the error surface changes continuously, since the training set changes. Consequently, when beginning with an easy subset, the gradient dynamic applied in a first error surface easily and rapidly leads close to a minimum of the surface, which places the network's weights in a good configuration to learn the second subset. This is repeated at each stage, until the end of the learning. In contrast, active selection presumes a non–changing surface, because the content of the training set is constant. Here a lot of attention is paid to the formation of the learning trajectory.

2.2.2 Different views on the error relief.

The error surface relief has been investigated by many researchers to reach a better understanding of neural operation. The work in [71] [81] is of special interest here. The book of Minski and Papert [104] reports that "getting trapped" in a local minimum is rarely a practical problem for backpropagation learning. Investigations on simple problems (XOR problem) [71] [98] also conclude, that local minima are not as common as thought in the beginning of the neural research. [8] has considered the learning in layered feedforward networks with linear neurons trained with backpropagation. The main result of this work is that the error surface has only one minimum, corresponding to the orthogonal projection onto the subspace spanned by the first principal eigen–vectors of a covariance matrix associated with the training patterns. All other critical points of the error surface are saddle points. This result can not be translated directly into networks with non–linear neurons. [81] proves that for the multilayer perceptrons with non–linear neurons the error surface is quite harsh – with a staircase appearance. The surface tends to have a large amount of flatness as well as extreme steepness, but with little variation in between. When the number of training samples is small, there is often a 1–to–1 correspondence between individual training samples and the steps in the surface. If the number of the training samples is increased, the surface becomes smoother.

Each flat spot looks like a minimum, i.e. the gradient in this area is very small. In addition, these flat spots may expand until infinity, which makes application of learning algorithms based on line search erroneous. Therefore alternative optimization techniques such as the *conjugate gradient method* have been proposed for traversing entrapment regions like plateaus and saddles. With this type of surface, a gradient search moves very slowly along these flat parts. Simulated annealing is designed to escape local minima, and to make it easier to avoid entrapments that are not due to a local minimum, since there is no need for annealing in the latter case. Other learning algorithms like stochastic gradient descent [47] can escape the entrapment when not caused by the local minima.

For a general type neural network the error function is a highly non–linear function of the weights. All the minima of this complex error function satisfy $\nabla E = 0$, where ∇E is the gradient of the error function in the weight space.

There are many theoretical models of the formed error surface. A high–order polynomial and a spline model requires a large number of parameters to describe the error surface, when the dimension of the weight space is large. Good insight into the optimization, that takes place in neural learning, is provided by the quadratic model of the error surface; moreover the quadratic model requires a reasonable number of parameters. If the dimension of the weight space is p, then the number of parameters for a local quadratic model description is $p + p(p + 1)/2$.

The quadratic approximation of the error surface is based on the assumption, that the error surface is approximately quadratic in the vicinity of a minimum. Therefore, the locally quadratic Taylor expansion, widely used in numerical analysis, can be applied [58]. In the neighborhood of a local minimum w^* of the training error $E(w)$, the quadratic Taylor expansion can be described in the following way:

$$E(w) = E(w^*) + \frac{1}{2}(w - w^*)^T H(w - w^*) \qquad (2–5)$$

where H is the Hessian matrix. While evaluating at w^* the expanded training error has no linear term, because $\nabla E = 0$. This representation creates a new coordinate system with the eigen–vectors of the Hessian for the axes. In the new coordinate system the contours of constant E are ellipses, centered at the origin. The axes of the ellipses are aligned with the axes of the coordinate system, and the length of these axes are inversely proportional to the square roots of the eigen–vectors. The Hessian H is usually unknown, since the exact location of the minimum w^* is not known. However, there are ways to approximate H, based on the gradients of the errors, calculated during training.

2.2.2.1 Learning process is a trajectory.

Lets summarize once more, that neural networks relate two data streams: the input and the output stream. The goal of neural learning is not to memorize this correspondence (mapping), but to model it: if unseen input patterns are presented, the network output should provide the value of the corresponding output variable. For regression problems the distribution of the output variables, conditioned on the input variables, should be modeled. In classification problems the posterior probability of the class membership should be modeled, conditioned again on the input variables.

The proper neural training must model the underlying generator so that the best possible computation of the target value y corresponds to the newly presented input value x. A problem can be described as a relation between a set of independent (input) variables and one or more dependent (output) variables. If the only definition of a problem is given by measurements of external behavior, its learning is determined by examples. Such a modeling of a problem is called *data–driven*. In other words a data–driven model is any model with parameters, determined by data.

Obviously the operation of neural networks is driven by a stream of data. A logical way to represent the steps taken to solve this data–driven task is by drawing its trajectory on the error surface. Since most often the training examples are represented in a random manner, the arbitrary local steps will be taken on this surface. This is one of the difficulties to ensure a reliable learning trajectory.

The error surface contains hills, plateaus, basins, and valleys. Downhill gradients aim to find the lowest point on the error surface i.e. the global minimum. Theoretically this is a point with zero energy for every example from the training set. In practice, merely an acceptable solution is found – a point with low average error E. The gradient algorithm draws the itinerary from the randomly chosen initial point on the error surface to a reachable minimum in the neighborhood. This itinerary goes through different surface relief forms.

Some of the relief forms are easy to pass; at others the gradient algorithm can't take a downhill step comfortably. The latter group of areas are generally called *stationary areas*, where the learning algorithm is entrapped. They are characterized either by the negligible value or a negative change of the error ∇E. The stationary areas can be of several types. In figure 2–10 all the possibilities are shown.

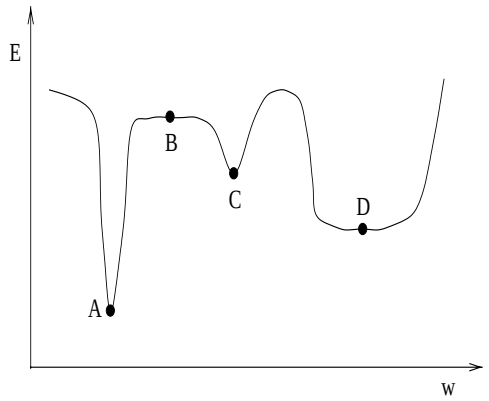


Figure 2–10: *Examples of stationary areas.*

Global and local minima (the areas A and D) are stable stationary points: there does not exist a non–ascending trajectory, which connects these points with a lower energy area [71] and consequently the gradient type algorithm can not escape them without external help. An interesting question is how to distinguish global minimum from local stable minima, if both have negative gradients of the error surfaces in every direction (for every parameter). The answer is that the global minimum contains points with acceptable low energy for every individual pattern, and local stable (or acceptably low) minima for the training set as a whole i.e. for E.

There are unstable stationary (metastable) points as well (the area B), where learning algorithm spends some time but has a large chance to escape from. Examples are valleys, plateaus, flat maxima, or in other words, areas which are flat at least in one direction. The flatness gives negligible or null change of the training error and therefore the learning algorithm does not progress to the global minima.

It is difficult for the gradient algorithm to distinguish between unstable stationary points and flat stable minima (B and D). If ∇E equals zero there is some flatness of the error surface, i.e. an existence of at least two equi–potent points, thus of symmetry of a certain kind [11]. In addition, the instantaneous error $\mathfrak{E}$ and the average error E are both non–zero.

2.2.2.2 Effects of randomness.

Some local minima (representing stable stationary areas) can be escaped due to the randomness of the learning algorithm if the minima are shallow. The unstable stationary areas are not an unsolvable problem for the gradient neural learning. They can entrap the learning algorithm for a time and even hinder its further progress. There are many cases, when learning process never escapes from such a flat area [10]. The gradient algorithm can be trapped indefinitely in such an unstable minimum, because of the indecisiveness of the gradient algorithm on the flat landscape. Additional symmetries, like those of the training examples and the parameter distributions, can make such an algorithm to oscillate endlessly in this flat areas.

Neural optimization is performed by randomized algorithms. This has a two–fold impact on reliability performance. The motivation of including randomness as a design principle is to improve the algorithm performance, thus indirectly to enhance reliability. The negative effect of introduced stochasticity is, that neural experiments can not be reproduced: every single run draws a learning trajectory with different shape and length.

A number of randomizations are made during learning, that aim to improve the neural performance. For instance, the initial values of network parameters are chosen randomly in order not to direct learning beforehand. The inventors of the backpropagation algorithm have suggested that not only the network parameters but also the training examples should be supplied at random [129]. In addition some authors have suggested to add noise to the inputs of to the neural network for better learning performance [78].

Firstly, the random choice of the initial parameters (weights, biases) within a small interval ensures an arbitrary start of the training algorithm. This can have a positive impact on the learning process if the initial start is near an acceptable minimum. In parallel computations (which all the neural networks support) there are certain but pre–determined number of possible solutions. Different solutions correspond to equivalent and repeating parts of the error landscape in the multidimensional space. This causes a repeatedly moving of the network state from one to another subspace of the multidimensional error space.

This phase of the learning process is known as a *symmetrical phase* [131], characterized by indecisiveness of the learning algorithm. A manner for breaking this symmetry in the neural operation should be found by establishing the itinerary to a specific solution. A properly chosen strategy can find a way of rapidly isolating a possible solution in the beginning of the learning process i.e. choosing the subspace on the error landscape in which the fastest solution trajectory can be drawn. This subject will be discussed in later sections together with indecisiveness in the gradient calculations and with permutational symmetry.

Secondly, during learning, random example presentation has a major impact on forming the itinerary, especially when the difficult–to–pass flat areas or shallow local minima of the learning algorithm are met. The example randomness ensures that even if two experiments start from the same place, they never have the same itinerary. In other words, two experiments will never be the same. They differ in the length of the learning phase, of the learning stages they pass etc. This subject will be discussed in detail in section 4 in which different randomizations are analyzed.

2.2.2.3 Distributed representation.

To every network can be associated a measure of its capacity – the amount of information, which can be encoded in it. Correspondingly, the problem to be learned has a certain complexity. If the capacity of a network is able to incorporate the complexity of a problem, it can be said in general terms that the network is able to learn the problem.

Neural networks consist of many equi–functional elements. There is not a direct correspondence between items of information, learned by the network and elements of the network. The information is distributed between storage elements – weights and neurons. There is not a simple measure to predict how many storage elements are necessary to encode an information item. In this sense every separate storage element is unimportant: if removed during training usually networks still succeed in fulfilling their goals. Because of that, neural networks are said to have an inherent fault tolerance.

If too many storage elements are removed during training or due to design faults, there exists the possibility that the capacity of the network is insufficient to incorporate the complexity of the problem to solve. In practice there are conditions which push the network to diminish the distributedness of its representation. This happens, for instance, when the storage elements are brought into a state where they are not functioning properly. In such cases the network looses its distributedness i.e. its effective capacity. Diminishing the distributedness of neural representation thus has a direct impact on its reliability.

2.3 Reliability enhancement.

In this section the two mostly used reliability enhancement approaches will be discussed first. The brief analysis of these approaches is a basis for development of a new reliability enhancement method that overcomes some drawbacks of the existing methods. The ability for *generalization* of neural networks is their most important feature. The performance measure, evaluating this aspect of the neural network operation, is based on counting the degree of achievement of the posed network goal on unseen data.

Improving the generalization performance of the network is underlying idea behind all the reliability enhancement methods. The first basic approach for achieving the ultimate goal of reliable neural learning is by exploring variations of networks and selecting the best of them: the functional redundancy approach.

The second approach takes a close look over the features of the error landscape. The variety of methods, as discussed in 2.2.1.1, attempt to control the learning process development. Mostly used among them are the second order methods and a variety of regularization methods. In the following only the regularization methods will be elaborated on, since they give a fair idea about the possibilities to improve the learning trajectory. In this thesis an alternative approach for reliability enhancement is taken which is developed in the last subsection.

2.3.1 Functional redundancy for enhanced generalization.

As stated before, the learning process can be represented as function approximation. From this point of view, learning of a smooth mapping from examples is an ill–defined

problem. In other words the information in the data is not enough to reconstruct the mapping uniquely in regions with missing data. Generalization in neural networks is based on the assumption, that the basic characteristics (extrema, saddle points, or in other words, the significant curvatures) are uniquely defined from the examples and that therefore the intermittent subfunctions are smooth or at least considerably less curved: small changes in the input parameters determine correspondingly small changes in the output. This makes the test on generalization the ultimate correctness proof on the quality of the example set and the learning reliability.

This metric for neural performance estimation supports a number of techniques for reliability enhancement. As defined in [52], there are three categories of techniques, related to generalization improvement:

- *pre–training parameter adjustment*. This category contains algorithms for improving the generalization ability by changing adaptation rates, number of hidden units, number of hidden layers.
- *in–training complexity constraints*. These techniques attempt to prevent the training data from overfitting (*Growing*).
- *post–training destructive algorithms*. Such measures aim to remove unnecessary units or weights (*Pruning*).

All this methods are finally introducing a variation in the training experiment as a whole and do not search for improvements within the learning process. For taking a more general view on this approach for reliability estimation and systemizing the investigation of all the possible variations, which will improve the generalization performance, the *functional redundancy* approach [116] [138] is suggested.

2.3.1.1 Generalization diversity and committees.

Even small changes in the network initialization, parametrization or structure can result in that the network will generalize very differently. There is an enormous amount of literature which discusses the impact of different diversifications. The first quantitative approach to generalization is made by [49]. The most systematic study of the *diversity* phenomena is made by [117] by using different statistical measures of generalization differences. It considers both generalization diversity within a set of nets, and between sets of nets. The study has an empirical nature: many experiments have been performed with a variety of differences. Later on a statistical measure has been applied to evaluate which kind of diversity gives the best results.

[117] is a good theoretical attempt for systematic study of neural reliability via functional redundancy. The engineering practice has proved useful for practical multiversion programing methods, namely the committee and the ensemble method. Intuitively, there are two possible ways to find the best network for every problem: (a) to train the variety of networks and to choose the best one, or (b) to investigate which kind of variety can bring the better results.

A manner to put the first way into practice is by training many networks and to compare the results. This direct form of the committee approach is computationally very expensive, and the training of the networks, which are not considered as good solutions, is a pure loss of computational effort. The test for goodness of a network is usually its per-

formance on the validation set. This can not be the optimal criterion for evaluation of the network performance, because the validation set represents only a small part of the problem.

The refined version of this approach suggests a combination of networks to be trained in a committee [118] [119]. The *committee* contains a set of networks, which can differ in several ways: the number of hidden units, the kind of network models, the mixture of networks, the optimization criteria, the initial weight configuration, the training parameters, the training samples. The simple committee involves averaging the predictions of the individual networks. More sophisticated methods require an additional *gating network* to decide which of the committee networks should determine the output.

The performance of the committee can be much better than the performance of each single net, taken in isolation. As natural data tend to be noisy, unreproducable and incomplete, a committee network houses a set of data representations, each being likely to occur in actual practice. Instead of recalling from a single compromised representation, a selection from the *ensemble* is made. Once trained, the committee can therefore have a much better performance on unseen data at the expense of a small increase of the computational complexity.

The success of ensemble averaging in neural networks is due to the presence of many local minima, and thus, even with the same training set, different local minima are found when starting from different initial conditions. The idea for the committee approach can be found in statistical models for regression and classification. The different local minima lead to independent predictors, and thus their average reduce the variance.

2.3.1.2 Voting network.

The other approach [116] [138] is based on N–version programming, which aims to improve the reliability of the software programs by using a different kind of redundancy. Developed independently of the committee approach, the N–version programming suggests basically the same: alternative versions of software programs to be developed and executed together. After being executed in parallel, they are either checked for agreement before proceeding, or passed to the voter, which determines the output.

The goal of such a *voting network* is not to smooth away the natural variation but to remove the outliers that may burden each different network. As neural networks are well–known to be almost perfect and each individual network has its own quirks, taking a vote over a diversity of networks is judged to equalize the judgement.

It is therefore of interest to investigate into which kind of diversity gives the best results. In [116] the diversities in the training parameters, initial weight configurations, network architecture, training samples, and contrasting methods is researched. It concludes that the training set diversity has a bigger effect than the diversity in the initial parameters.

In this thesis, during the logical development of the suggested approach, the best way to improve the network performance is found to be the one with reordering or resampling of the training set. From the point of view of reliability estimation by functional redundancy, it improves the network performance by training set variation. The proof, made by [116], that the variability of the training set brings the best results for neural reliability enhancement with respect to the other variation techniques encourages us that the chosen way can bring the optimal solution.

The similarity of the work, done in this thesis, to the described N–version and ensemble methods is, that we use different training sets from the same signal to investigate the

network performance. Instead of training a lot of networks and choosing the best of them, the optimal performance of the single network is obtained by changing the training set content or order of its components during learning. In this way, our approach is a reliability enhancement method, based on the momental condition of the training process. By substantiating the possible difficulties, which the learning process can meet on its way to a successful ending, and analyzing their appearance on the network instantaneous behavior, the change in the example presentation order is performed. Our work combines elements of Active sampling methods, Ensemble methods and Sensitivity analysis.

2.3.2 Regularization methods for reliability enhancement.

An alternative approach towards improved learnability and therefore enhanced reliability of the training process are the methods that analyze the shape of the error surface. As elaborated so far, the reliability of neural networks is not an intrinsic property. The highly symmetrical architecture and initialization will often cause the learning algorithm to be stuck in one or the other static part of the error surface. If this static part is a non–satisfactory minimum, high plateau or saddle, the learning performance and duration can differ significantly from the optimal or the median performance result. To increase the reliability of the network, the optimization strategy should be changed. There are two basic manners of doing that:

- *deterministic*, which includes changes in the learning algorithm, network parametrization, learning strategies, and
- *stochastic*, which in general includes noise injection in the different parts of the network [3].

The most widely used deterministic methods analyze the second–order derivatives of the local error surface. They give an information about the curvature around this local point by calculating the second–order derivative of the error function with respect to all the parameters. This information will undoubtedly help the learning algorithm to make a decision at which direction to go. The calculation of the second derivatives is a costly computation, which is made after the assumption, that the error surface is locally quadratic. In our further work we would like to avoid this two drawbacks. This is the reason not to elaborate further on on the second–order methods.

Widely used techniques for stochastical and deterministic reliability enhancement are the regularization methods. *Regularization* in neural networks [123] [150] has two positive impacts. Firstly, it remedies numerical problems in the training process by smoothing the error surface and by introducing additional curvature in low (possibly zero) curvature regions. Secondly, the regularization is a tool for reducing the variance by introducing extra bias [62].

The basic idea of regularization is to stabilize the solution of some auxiliary non–negative functional that embeds a priori information, and thereby to turn an ill–posed problem to a well–posed one [123]. More specifically, this can be done by adding a penalty term to the objective function of training. This can be done in the following way. Consider again the feedforward neural network as a parametrized non–linear mapping from the a–dimensional input space $x = (x_1, x_2, ...x_a)$ to a b–dimensional output space $y = (y_1, y_2, ...y_b)$. Without loss of generality the output space can be considered as 1–dimensional. The approximation function is represented as $F(x)$.

Tikhonov suggests in his regularization theory that the function F is determined by minimizing a cost functional $\mathcal{E}(F)$. This functional involves two terms:

- the *standard error term*, corresponding to the error function as defined already in equation (2–4) and due to the examples in the training set:

$$E_{stand} = 1/N \sum_{\mu=1}^{N} (y^{\mu} - F(x^{\mu}))^2 \tag{2–6}$$

- the *regularization or penalty term*, depending on the geometric properties of the approximation function $F(x)$ and denoted by:

$$E_{reg}(F) = 1/2\|PF\|^2 \tag{2–7}$$

  where the operator P contains a priori information about the solution.

The success of the regularization approach depends on the form of the penalty term, and on how well the penalty term corresponds to the underlying relation. The weakness of this method is that the type of regularization depends on the problem to be solved. Often it is not clear what kind of regularization can be used for a particular problem.

In the category of deterministic approaches, the simplest intuitive regularization methods are weight decay [77], and weight elimination [156]. In these approaches the penalty term presents an intuitive measure of the size of the network.

Adding a regularization term E_{reg} to the cost function, which can in its simplest way be a weight magnitude, is known as a *weight decay* technique.

$$E_{tot} = E_{stand} + E_{reg} = E_{stand} + \lambda \sum_{i} (w_i^2) \tag{2–8}$$

where λ is a parameter, controlling the weight decay rate. The second term has the effect of reducing large weights. It has been found empirically that a regulizer of this form can lead to significant improvements in network generalization (Hinton [77]).

The idea of increasing the stochasticity in the learning process in order to achieve a better learning performance is not new. Reed [125] and Bishop [20] assume that training with input noise converges to a kind of average of a stochastic error function. On basis of a Taylor expansion of the expected error function, they concluded that training with input noise is equivalent to a form of regularization.

Matsuoka [102] and Holmstrom[78] have contributed to this subject as well. In [3] the objective function is derived, that is minimized by the backpropagation training with input noise. After careful analysis of this objective function he contradicts Reed [125] Matsuoka [102] and Bishop [20]. An [3] concludes that even in a weak noise limit, training with input noise is not equivalent to a regularization method. The difference lies in the noise–induced term in the objective function, that depends on the fitting residuals. This component of the cost function has not been noticed by Reed [125] and Matsuoka [102]. Bishop has concluded, that this term vanishes at the end of training, which is true only for an infinite training set. In this case noise injection is really equivalent to a regularization. An [3] concludes, that the importance of this term is comparable to that of the regularization term. It has the same effect as regularization (inputting of noise has a smoothing effect on the network function), but according to the author [3] this smoothing operates fundamentally different from smoothing by regularization.

The main contributions of noise injection to the neural reliability are shown to be the enhanced generalization ability and fault tolerance. As it will be shown in further chapters, input noise injection can contribute in obtaining a shorter learning trajectory.

2.3.3 Adaptable learning trajectory.

The considerations about the shape of the error surface, elaborated on in the previous section are to be summarized in search for a better trajectory of the learning algorithm. As the described learning drawbacks suggest, the main obstacles for reliable learning are the following tendencies of the gradient algorithm:

- difficulties by escaping the initial symmetrical phase, which was shown to be a shallow valley, symmetrically positioned between the subspaces of the multidimensional vector space;
- getting stuck in a high energy valley or flat plateau after the initial symmetrical phase is broken;
- difficulties to find a satisfactory low point, when the solution is close to the optimal, i.e. in the third (tuning) stage of learning.

Giving an universal prescription of how exactly to overcome all these hurdles on the way of the training algorithm is not an easy task for algorithms with a random nature and for an ill–posed problems, as neural networks are expected to solve. Although it is not possible to give the way of solving every reliability problem in terms of an optimal trajectory, the following reasoning will suggest basic principles, that can be followed.

In the previous section, regularization and noise injection are presented as well–established techniques for optimizing the successful learning of particular groups of problems. Explained in terms of the error surface, the regularization method as well as the noise injection methods improve the behavior of the learning algorithm in areas of the error surface with very low curvature, where the optimization process progresses very slowly and in the areas with very high curvature, where smoothing of the error surface is advisable. Similarly the method, which will be suggested, is also directed to deal with these two difficult types of areas of the error surface. The strong point of this method is that it does not change the definition of the problem, as the noise injection techniques do, nor is it based on adding an extra terms to the cost function, i.e. changing the optimization goal, as all the regularization methods suggest. Both, regularization and noise injection, are thus problem (problem area) specific, which we attempted to avoid in this work.

Problem dependence is a know drawback of all the second–order training algorithms as well. Additionally, the second–order methods have increased level of computational expenses, which can hamper enormously a real–life task. The drawback of increased computational expenses is intrinsic to the functional redundancy methods as well.

Our goal is to develop a reliability enhancement method which insures success of the training process by redirecting the learning trajectory if necessary. This redirection should be based on a criterion, which does not require the heavy computations, since it should have a practical relevance. Development of such a methods requires an understanding of the mechanisms, which guide the learning trajectory as well as the potential

training problems, expressed as a features of the error surface. Furthermore, we would like to keep our reliability improvement method simple to perform and as little dependent on the training task as possible. To find such a solution, unified view on the possible learning difficulties should be build first.

The expected trade–off of such a method is that it will have lower precision and for sure will not solve an arbitrary training problem. It can force to suboptimal solutions as well.

In search for a better trajectory on the global error surface, the following reasoning on the path taken by the learning algorithm will be made. The learning algorithm chooses steps with lengths, determined by the current error signal, and direction, posed by the projection of the weight vector on the multidimensional error surface. For a very simple network with two weights, choosing the gradient direction is illustrated in figure 2–11. From this figure it is obvious, that the gradient direction is a vector sum of the gradients with respect to the two network parameters.

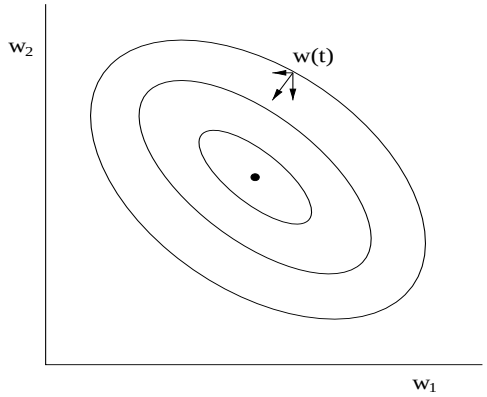


Figure 2–11: *Defining the gradient direction for the two weights network*

The error function can have substantially different curvatures along the different directions [20] (Figure 2–12a). The constant error values form ellipses, whose axes differ a lot, so that the error surface has the shape of a valley. For most points of the error surface the local negative Hessian vector $-\nabla E$ does not point towards the minimum of the error function. Thus, the successive steps of gradient descent oscillate across the valley and the minimum of the valley is reached very slowly. The behavior of the learning algorithm is directed by the way of example presentation, which is most often random.

a) On a highly–curved error relief ***b) On a flatness***

Figure 2–12: *Gradient descent on two peculiar error surface areas.*

2.3.3.1 Criteria for reliability estimation.

The selection of an appropriate error measure depends on the kind of the problem, that the network should solve. For many tasks this involves learning a continuous function, as most of the experiments in this thesis investigate. It is shown by [137] that the *root mean square error* (RMS error) is a good choice for a reliability measure by continuous tasks.

$$\epsilon_{rms} = \sqrt{\frac{1}{N}\sum_{\mu=1}^{N}[y(x^{\mu}) - f(x^{\mu})]^2} \tag{2–9}$$

The authors of [137] consider the problem, inherent for the RMS criteria, that the difference in the scales of the signal to be learned are not encountered by it. Therefore they propose the normalized measure of the approximation quality, which they call *AQ*.

$$AQ = \frac{f_{RMS}}{f_{RMS} + \epsilon_{RMS}} \tag{2–10}$$

where f_{RMS} is the RMS value of the function to be learned. The so–defined *AQ–measure* is dimensionless, bounded between 0 and 1, as 1 corresponds to no error, and 0 – to infinite error. [137] uses a fault tolerance investigation method, based on randomly disconnecting two processing elements at every experiment. When increasing the number of allowable faults, the fault tolerance grows in a combinatorial fashion.

Describing the learning process as an itinerary on the error surface makes it possible to define its quality through the learning trajectory. A simple method of determining the quality of a learning scheme according to its trajectory is to measure the rate, for which it reaches the point of convergence to an acceptable solution. This widely used measure for a learning quality is usually determined in terms of number of epochs until a satisfactory solution is reached. Enhancing the reliability implies that the quality of the learning trajectory should be taken care of. This implies first, that the training duration should be decreased, to insure that the training will be performed in the desired time borders, and second that the improved learning duration can always be achieved. There are several important issues, which need to be taken into account.

Learning trajectories can converge very slow or not at all because they get stuck in a stationary areas. Increasing the learning speed should find a way to shorten the time spent in these areas. Furthermore, the improved learning results should be repeatable. Besides on the learning method the repeatability depends on the random factors, which influence the formation of the learning trajectory. The network parameters are chosen randomly, which define the starting position on the error surface and the presentation order of the training examples, which direct the trajectory further on. To encounter the influence of the random factors, every training experiment will be repeated many times. The conclusions over the quality of the learning trajectory will be made on the basis of statistical characteristics of many experiments.

Let the network output be denoted by $f(x)$. During the learning phase, the network should learn to approximate a dependence $y(x)$ on a set of N examples $\boldsymbol{y}(\boldsymbol{x}^{\mu}) : \mu = 1, 2, ...N$. The distance between the desired dependence $y(x^{\mu})$ and the one actually learned by the network function $f(x^{\mu})$ is given by the cost:

$$C = \sum_{\mu=1}^{N} [y(x^{\mu}) - f(x^{\mu}, \boldsymbol{w})]^2 \qquad (2\text{–}11)$$

It is obvious that this cost function is equivalent to the error function, as defined in the previous chapter. Correspondingly, the learning success and reliability can be defined through the error function.

As discussed in the section 2.1.2.1, the appropriate stopping criteria should judge the learning reliability in terms of adequacy of the solution. If the learning process does not succeed to find a solution in a reasonable time which meets the accuracy criteria posed, it is classified as a fault, although the solution in some cases is close to the satisfactory. The convenient network performance criteria in terms of learning trajectory is to reach the point in the error surface, which has a magnitude below certain value.

Selecting the starting position of the learning process within a small range will give a more stable learning duration. Choosing the initial weights from a larger range will lead to a larger variance of the convergence time, but it can speed up learning, since the larger weight values will prevent the learning algorithm from spending time in the initial symmetrical phase. The choice made for further analysis is for the small initial weight values range. The small range of the initial parameters ensures more stable learning duration and gives a better theoretical foundation for convergence analysis.

The last evaluation point of the learning quality is which of the averaging measures must be used for interpreting the outcome from the subsequent experiments. The representative number of experiments, that allows for a statistically meaningful statement, needs to be determined for an adequate average performance.

A good measure for comparison of the learning reliability in terms of *probability* to converge to a proper solution is just to count the successful trials. The percentage of successfully converging experiments puts grounds for comparison between the different pattern selection strategies.

Defining the learning reliability in terms of *time* – the period during which a certain degree of performance can be achieved, two measures will be given – the average convergence time and its variation. It is well–known from signal processing literature that the median value is far more resistant than the mean, when the inputs are noisy or faulty. Therefore the median training time and its variance will give a good estimate of the quality of the average achieved trajectory with an arbitrary learning method.

To evaluate the quality of a particular experiment, in [122] the average cost–value of a random guess is suggested. This criterion is constructed to imply the answers on the following questions: How good is the solution with respect to the global optimum or the best known answer; how does the performance change with the problem size; what is the performance of the network on two different tasks, i.e. are there easy and difficult problems to solve. The authors define a solution quality q, by mapping the optimal cost value c_{opt}, the average cost value c_{ave}, and the current result of an experiment c_{ave} onto a normalized quality scale.

$$q = \frac{c_{ave} - c}{c_{ave} - c_{opt}} \qquad (2\text{–}12)$$

The solution has value $q = 1$, if $c = c_{opt}$ and quality $q = 0$, if $c = c_{ave}$.

2.3.3.2 Towards an optimal learning trajectory.

In this thesis only neural reliability during the learning phase will be considered. As our interest is directed to find out how a satisfactory convergence can be achieved in a reasonable time, only the reliability during training can be of interest, since only then the neural system is being adapted.

A logical conclusion of this short review of reliability methods is that the performance of a neural operation can be estimated and studied by all of them, but its enhancement needs a deeper understanding of the training process itself. This is the reason to direct our attempts for reliability enhancement towards finding an optimal learning trajectory. As discussed in 2.2.1.1, the main performance enhancement techniques, which the learning trajectory method supports, have a number of major drawbacks. They either require heavy computations, practically impossible to be made for a real–life task, or use heuristics to find a solution, or make assumptions on the shape of the error surface which is not always satisfied, or change the problem definition, or combine few of these drawbacks [47].

In our analysis we take an alternative view on the learning success: the learning depends more on what the network has to learn than on its structure, parametrization and optimization method. We are investigating learnable and not learnable, easy and difficult tasks. Since the same network is able to find the internal structure of one task and fails (or performs unreliably) a task with a similar complexity and structure, we assume that the manner of presentation of this task to the network has a crucial impact on its reliable performance. The error surface paradigm is very useful for understanding the internal mechanisms of problem presentation, since it naturally combines the network and the task (the learning problem).

In our approach heavy computations of the second–order characteristics of the error surface are avoided, as this is unpractical for the current state of computational technology. Instead, the error surface abstraction is considered as containing a static part – the relief obtained when the overall difference between the actual and obtained network output is drawn – and the dynamic part – the steps which the training algorithm takes on it.

The most often encountered reasons for slowing down or getting stuck in the learning process are either flatness of the error relief in at least one direction or areas with a high curvature: local minima and valleys. Since learning trajectories are defined to a large extent by the training examples, we suggest that all the peculiar areas on the error surface can be avoided by finding a proper itinerary of the learning process through the ordering of the training examples in a way that surmount the error surface difficulties. This point of view supports the idea that the simple backpropagation algorithm, by far the most often used in neural computations, can successfully cope with most of the learning reliability problems.

‘

3 Symmetry and indecision.

The computational drawbacks which may prohibit the success of neural learning have been listed in Section 1.2.2. Each drawback can be related to a corresponding error surface. In order to provide for an alternative to the visualisation of the error relief form, one has to analyse in more depth the notion of symmetry. It shows that symmetry has a number of (dis–) advantages. Sometimes symmetry is required to allow for a mathematical closed–form description, but most of the times it simply creates the "Qual der Wahl". This eventually leads us to formulate learning as an evolvable interaction between knowledge, randomness and symmetry.

According to the previous chapter, the major hurdle in achieving high learning reliability lies in the indecisiveness of the learning algorithm. The initial indecisiveness stems from neural design principles: the network is build symmetrically to allow for an arbitrary direction taken by the learning algorithm. Solutions can be found in any direction, since the error surface consists of many repeating parts. We refer to this characteristic of the error landscape as *repeatedness*. Another kind of indecisiveness sets in at later stages of learning, when already captured knowledge makes new information to have hardly any influence. This can occur when the error landscape has a local *flatness*.

Our view on the error relief is that it contains mainly flat and steep areas, as shown theoretically by other researchers [81]. The flatness may be in the shape of a plateau or as a kind of saddle point. In both cases the learning algorithm and the stochasticity, introduced in the learning process, can not traverse these areas easily. The indecisiveness of the learning system corresponds to different kinds of symmetries in it and will be discussed in this chapter.

The terms repeatedness and flatness are often used interchangeably in conjunction with the term *symmetry*. It is necessary to explain what is meant by symmetry. The following definition gives the geometrical sense of the term:

Def. 3−1: Any object which displays some kind of repeating pattern in its structure is said to be symmetric.

This first definition directly relates the repeatedness of the error surface relief in the different subspaces with the symmetry of the error hyperspace. Contemporary physics gives a more general definition of the symmetry [158]:

Def. 3−1a: An entity is symmetrical if there is something that you can do to it so that after you have finished doing it it looks the same as it did before.

This second definition explains why the terms 'repeatedness' and 'symmetry' can be interchanged. The equi–functionality of the neural network in response to different stimuli can be represented as equi–potential points on the error surface. When such equi–potential points come close together, they form a flat area.

To give a more practical sense to this definition, symmetry can be defined in terms of the three linear transformations T in the n–dimensional Euclidian space E^n: reflection,

rotation and translation. A subset S of E^n is symmetric with respect to a linear transformation T if $T(S) = S$.

In the following a number of aspects of the symmetry problem will be treated in order to create a higher–level point of view. The first section focuses on the aspects that are designed into the network at the start of learning. Subsequently, attention moves towards symmetry aspects that occur later in the learning process. Then, we attempt to close the gap by showing the environment influences the occurrence of symmetry problems. Lastly, we provisionally introduce the KRS classification scheme to support further thinking on the nature of learning disabilities.

3.1 Initial symmetry.

Initial symmetry is introduced to the network during the architectural design process. This symmetry gives freedom to the learning algorithm to draw its path in every direction. All the symmetries built prior to the network operation will be named *network symmetries.* The network symmetries result from the choice of network architecture and transfer function as well as from the specific settings of its initial values (weights and biases, scaling intervals of the example strings).

Firstly, the symmetry in the network architecture and transfer function, will be discussed. Since the architecture and transfer of the network determine its structure, these symmetries can be called *structural symmetries.* Later on, the symmetries caused by the initialization of the network will be summarized. As mentioned earlier, here are meant not only the starting values of weights and biases but also the scaling intervals for the examples. Usually all the initialization intervals are symmetrical around the origin, which affects the performance symmetry.

3.1.1 Structural symmetries.

The structural symmetries will be illustrated by using the fully connected multilayer perceptron (MLP). In MLP the structural symmetries are caused by the existence of the so–called *coherent transformations:* transformations that do not change the network functionality. On closer inspection, a coherent transformation can be either a permutation or a sign inversion and therefore induces a large number of functionally equivalent networks.

3.1.1.1 Permutation transformation.

In the standard feedforward topology, neurons from the same layer are prior to learning invariant with respect to their position within the layer (Figure 3–1). This *group invariance* allows to permute all the connections of a neuron (inputs and outputs) with those of another neuron in the same layer without changing the network function.

$$f(\boldsymbol{x}, w) = \varphi\Big(\sum_{k} w_{k,k-1}\varphi\Big(\sum_{k-1} w_{k-1,k-2}\varphi(\ldots\varphi\Big(\sum_{j} w_{ji}x_i\Big)\ldots)\Big)\Big) \qquad (3\text{–}1)$$

Equation (3–1) represents the response of a multilayer perceptron with one output as shown at figure 3–1. If two neurons from the same layer are interchanged, the network output will be the same, because the change affects only the ordering of the elements under the sum sign.

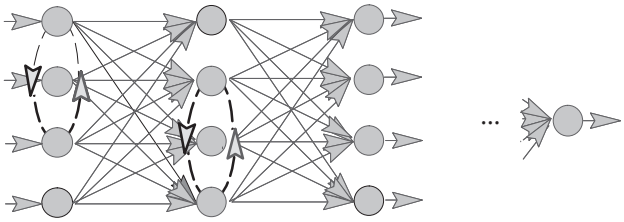


Figure 3–1: *The neurons in a fully–connected topology are group–invariant.*

3.1.1.2 Sign transformation.

Another transformation with coherence properties can be defined on the nature of the individual neural transfer function. These are based on the network function invariance caused by neural symmetry properties: odd and even.

Def. 3−2: A function φ if *odd−symmetric* when $\varphi(x) = -\varphi(-x)$.

The zero–centered sigmoid has odd symmetry (Figure 3–2a). A transformation, that inverses the sign of all input and output connections of a neuron with odd symmetry, does not change the network transfer function. It is proved that this symmetry is valid to a wide range of activation functions [1], i.e. for any infinitely differentiable function σ, satisfying $\sigma(0) = 0$, $\sigma'(0) \neq 0$ and $\sigma''(0) = 0$ and so forth. For the logistic sigmoid holds likewise $\varphi(x) = b - \varphi(-x)$.

Def. 3−3: A function φ if *even−symmetric* when $\varphi(x) = \varphi(-x)$.

The standard Gaussian function has even symmetry (Figure 3–2b). A transformation, that inverses the sign of all input connections of a neuron with even symmetry, does not change the network transfer function. It is proved that this symmetry is valid to a wide range of activation functions [1], i.e. for any infinitely differentiable function σ, satisfying $\sigma(0) \neq 0$, $\sigma'(0) = 0$ and $\sigma''(0) \neq 0$ and so forth. By radial–basis transfer functions holds the the following: $\sigma'(0) \neq 0$, $\sigma''(0) = 0$, $\sigma'''(0) \neq 0$ and so forth.

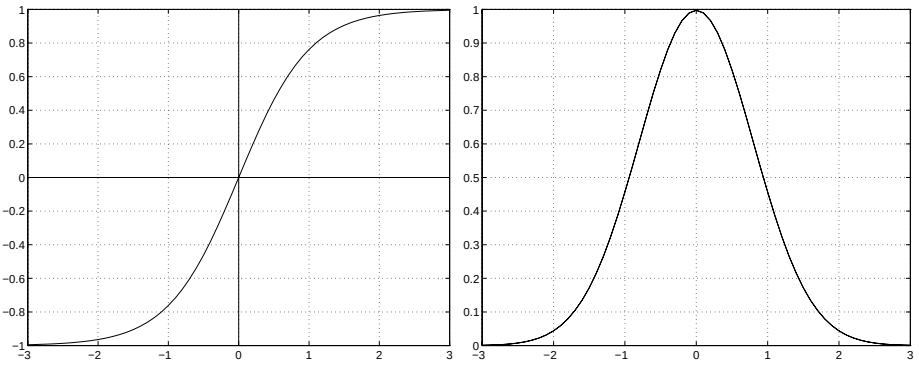


Figure 3–2: *Neuron transfer functions with a) odd symmetry and b) even symmetry do not change the transfer of the complete network.*

The symmetries caused by these types of transformation have different impacts, depending on the functionality of the neurons they are affecting and their local interconnections. If the neurons are directly interacting with the environment (i.e. the input and output layers), the symmetry provides an invariance in pattern presentation: The permutation of the elements of an input string does not alter the output. Because the permutation or sign symmetry at the input makes sense only in the context of reordering the elements of the input string, it will be elaborated together with the spatial symmetries in a problem. Further, each hidden–layer symmetry has a different impact on the learning evolution, as discussed in the next section.

3.1.1.3 Repeatedness.

An invariance of the network function exists for each neuron upon a sign transformation which determines two different weight vectors. For M hidden units, there are M symmetries of this kind, and thus there are 2^M switching possibilities. The state of combinations forms equi–functional weight vectors. In addition, if the values of all the weights and bias of two hidden neurons are interchanged, the resulting network transfer will be unchanged. Only a new network weight vector is obtained.

For M hidden units, there will be $M!$ equivalent weight vectors. These two types of symmetries are combined, and the network has the weight space symmetry factor of $M!2^M$. For feedforward networks with more than one hidden layer the level of the resulting symmetry can be obtained by multiplying these factors for every hidden layer. This result has been summarized by [35]:

The set of all equi–output transformations on the weight space W forms a non–Abelian group G of order #G, with $\#G = \prod_{l=2}^{K-1}(M_l!)(2^{M_l})$

where K is the number of hidden layers, and M is the number of neurons in the hidden layer. The authors of [1], [35], [85], [94], [147], analyzing this group of symmetries, have concluded that each coherent transformation defines a symmetry in the weight space, as consisting of equivalent parts. Consequently, it is possible to restrict the search of solutions to only one of these parts. By analogy, the error surface is also symmetrical. Additionally, if a solution is found during learning there are more equivalent solutions on the symmetrical parts of the error surface. An important conclusion is that the learning trajectories are themselves symmetrical [85]. The general conclusions by the refered authors is that in many cases these symmetries are of little practical importance.

A different interpretation of the importance of network symmetries has been reported in [12] [131]. These papers suggests, that the neural system frequently moves from one subspace of the global error landscape to another subspace in a manner that ensures contradictions in training inputs and delays in the learning process. The system spends a substantial amount of time between regions defined by weight symmetries. It can be confined artificially to one of these subspaces, see [131].

3.1.2 Symmetries in the network parameters.

The initialization of the network is not a thoroughly explored subject. Beside the network structure, also its parametrization can cause prolonged symmetrical learning phases. The most widely accepted approach for choosing the values of the initial param-

eters is given by [129], where an initialization with small random weights is advised. More precise borders of the size of intervals for weight selection are given by [74], where it is suggested that the weights and biases should be chosen from the interval $(-2.4/F_i, +2.4/F_i)$, with F_i for the fan–in of neuron i. The most complete study of the initialization strategies is made by [148]. Appearance of symmetries due to wrong initialization are discussed in [91].

The following conclusions are made in [81] upon investigating a very simple, one node network with two weights. The region near the origin of the error surface (the region where almost all parameters are approximately zero) is a transition region. This fact supports the idea suggested in [129]: the weights should be initialized with small random values. This initialization technique increases the probability of positioning the initial weights near the origin on the steepest part of the surface, where the downhill direction points towards a solution. The author notifies, that this is true for one–node case. In the multilayer case, the steepest part of the surface is not necessarily near to the origin. For the multilayer case, the advantages of selecting the network parameters from a small symmetrical interval are given in [134] [148].

3.1.2.1 Overparametrization.

This kind of initialization can be disadvantageous if the network is overparametrized. As a result the cost function may be almost flat at the minimum point in some directions of the weight space. If the size of the training set is small in comparison to the dimension of the weight space, then a flat cost function may occur. This can be shown by reasoning over the following example: a two–layer neural network with one input and one output neuron (Figure 3–3a). Without loss of generality it is considered that the input–to–hidden weights are fixed to unity. The reasoning is done only for the small random hidden to output weights. The network function in this case has the form:

$$f(x, \boldsymbol{w}) = \varphi \sum_j w_{k,j} \varphi(x) = \varphi \sum_j w_{k,j} o_j \qquad (3\text{–}2)$$

where o_j denotes the output of any hidden neuron and represents a downscaled value of the input sample x. The values of $w_{k,j}$ are drawn from the uniform symmetrical distribution with a small range.

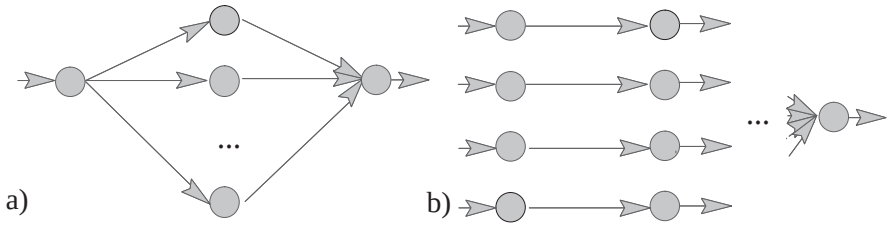


Figure 3–3: *The impact of overparametrising a network: the architectures used.*

It can be concluded that the larger the number of weights, the higher the possibility of the value under the sum sign being zero. In practice, the input–to–hidden weight range is smaller than unity, which implies that the symmetry effect of the large node number

within a layer is stronger than the one described here. This condition increases the initial indecisiveness and escaping from the initial symmetrical phase becomes difficult. This is often observed when training for artificial problems, but rarely in real applications since the diversity of the problem samples may break this symmetry.

$$f(\boldsymbol{x}, w) = \varphi\Big(\sum_{k} w_{k,k-1}\varphi\big(w_{k-1,k-2}\varphi\big(...\varphi\big(w_{ji}x_i\big)...\big)\Big) \qquad (3\text{–}3)$$

3.1.2.2 Range symmetries.

A large number of hidden layers has an equivalent effect on the network performance. Let all the weights in the multilayer structure 3–3b be fixed to unity, and the network inputs to be scaled in the range [–1..1]. If the sigmoid nonlinearity has slope 1, the first hidden layer node outputs will take values between –0.46 and 0.46, as shown in figure 3–4a. The second hidden layer receives inputs in the range [–0.46 .. 0.46] and provides outputs smaller by factor two [–0.227 .. 0.227]. This narrowing increases with increasing number of layers and is strengthened in the fully connected architecture by the fan–in symmetrising impact (see Fig.3–3a). In practice, when the weight values are initialized smaller than unity, the effect of the multilayer structure is more pronounced, as shown in figure 3–4b.

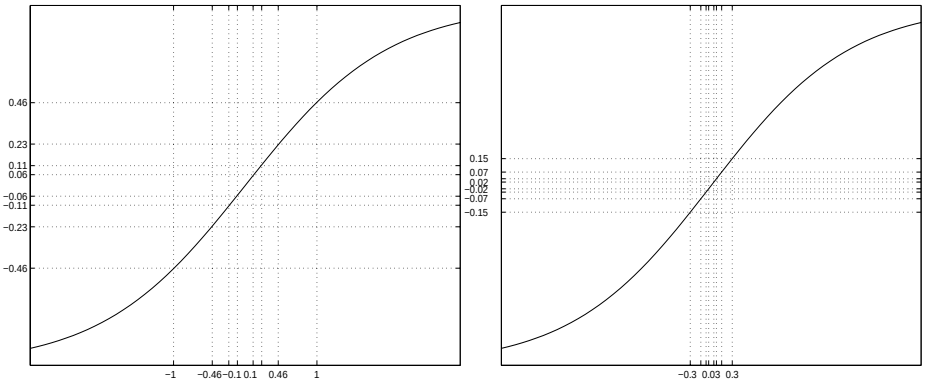


Figure 3–4: *Active zones of the consequent layers of sigmoid function by different range of the input stimuli.*

The network symmetries correspond to the repeating equivalent parts of the error surface. High symmetries in the initial conditions form flat areas on the error landscape, giving the same response of the network to any input stimuli.

In order to improve the generalization performance a number of authors have suggested to add noise in the initialization (as well as in the network parameters or to the problem). For more details one is referred to the review of noise injection in [3]. The effect caused by increasing the task complexity is somewhat similar to that caused by increasing the dominance of the random component. Lower symmetries during the learning evolution are observed in [157] when an non–zero bias factor is introduced in the *committee machine*. An extremely small slope of the nodal transfer [157] as well as a low learning rate [159] can cause equivalent effects.

The initial phase of the neural network learning process has two major features: (a) the parallel initial structure, and (b) the choice of a flat start position for the learning algo-

rithm. The parallel initial structure defines weight space symmetries. From the perspective of the error surface, equivalent landscape parts are formed. The learning algorithm starts from the so–called symmetrical phase. Later on it selects its path in one of the subspaces. From the error surface point of view this initial symmetrical phase is also an error surface relief form, which should be traversed by the algorithm. This form is situated equi–distantly (symmetrically) between the different subspaces (solutions). In section 3.3.2, it will be shown that this surface form is a saddle point; it is flat for small initial parameters.

3.1.3 Statistical mechanics view on symmetry breaking.

Statistical Mechanics explains the symmetry in neural networks and alike problems in a global sense. It relates the microscopic neuron structure to macroscopic neural properties by describing the collective properties of many interacting elements on basis of individual behavior and mutual interaction. The system properties can be more complex than the mere collection of its elements. This feature is a consequence of spontaneous *symmetry breaking*, which means, that a macroscopic system can be in a less symmetrical (more complex) state than the underlying microscopic dynamics.

The error landscape is in the Statistical Mechanics literature usually formulated by an energy function, as introduced by Hopfield (1982). The term “Energy function” comes from the physical analogy to magnetic systems, but the concept is of a much wider applicability. In many research fields there is a state function that always decreases during dynamical evolution, or must be minimized (maximized) to find an optimum state. In the theory of dynamic systems it is called the Lyapunov function. Other names in use are Hamiltonian in Statistical Mechanics, Cost function (Objective function) in Combinatorial Optimization theory, and Fitness function in Evolution theory.

3.1.3.1 Spin–Glass model.

In general, an energy function can be analytically determined if the connection strengths are symmetric, i.e. $\varpi_{ij} = \varpi_{ji}$. Thus, symmetry in the connectivity matrix is a necessary condition for Hopfield networks. Symmetry in the error landscape has its analogy in the temperature charts of the *replica method*, originating from Spin–Glass theory. Spin–Glass theory has been used to explain some network properties among which the *symmetry–breaking* process. This explanation is based on the fact, that some networks like the multilevel Kohonen network are developing *orientation selective neurons*. This orientation selectivity can be modelled by a Spin–Glass model and thus also be an example for the symmetry breaking process. Since Spin– Glass theory is a very specific subject in the field of theoretical physics, its studies on neural dynamics are understood well by a restricted number of scientists. In neural research mainly the results of this studies are used. The Spin–Glass theory is beyond the borders of the research done in this thesis. In the sequel, a simple interpretation of its results will be given, and the heavy mathematical description is entirely skipped.

In [61] techniques are introduced, that enable the application of Statistical Mechanics in feedforward neural networks. The output of the hidden neurons have equal absolute values at the beginning of the learning process, because the small initial values are positioning their outputs in the interval close to the zero point of the transfer function. If during learning the network is not able to extract a rule from the data, the hidden neurons do not specialize to a concrete function. This characterizes a symmetrical phase

in the learning process, corresponding to a metastable state in the error surface. This symmetry should be broken, i.e. the system should reveal its high complexity, which corresponds to a specialization of the hidden layer units, and jump to a lower energy state on the energy landscape. Because initially the symmetry is built into the network, the examples together with the noise are supposed to break it.

3.1.3.2 Soft committee machine.

Statistical Mechanics presumes a straightforward relation between the network, drawn in a symmetrical phase, and the permutation symmetry. In [154], the permutation symmetry is considered in the context of learning evolution: the permutation symmetry makes the learning process to start in the symmetrical phase. Correspondingly, all hidden units have almost equal response. In the language of Statistical Mechanics, every hidden neuron of a student (student is any network under learning) imitates with the same degree a teacher (the network configuration which gives an exact mapping of the input–output dependence). This permutation symmetry should be broken, i.e. every hidden neuron should specialize, or in other words, should differ in response from the others. This way the network will reveal the higher complexity of the problem, i.e. will go in the state of lesser symmetry.

Moreover the Statistical Mechanics model of feedforward neural networks poses a constraint that induces an additional symmetry in the network structure. The most complicated structure, investigated until now with the methods of Statistical Mechanics, is a *soft committee machine* [18]. This is a two–layer neural network with adjustable input–hidden, but fixed hidden–output weights and linear output. The average learning dynamics of this network is studied in the thermodynamic limit of the infinite input dynamics (the number of neurons is $N \to \infty$). The biases of the hidden neurons are fixed to zero. This model is said to be quite similar to real–world networks. The constraint that hidden–output weights are fixed on unity has been removed in [127]. There it is shown that the learning dynamics are usually dominated by the input–hidden weights (true mainly for the initial learning of MLPs) and hence the zero bias constraint is severe enough to introduce sufficient symmetries in the network.

3.1.3.3 Shortcomings.

In off–line (batch) training the symmetry breaking is connected with the effect of *retarded generalization* [154]: only, if the number of examples is large enough, can the hidden units specialize and decrease the generalization error efficiently. Something similar, but not identical happens in on–line learning: initially the hidden units do not specialize which leads to a plateau in the learning curve. If the version space (the space of all weight vectors, consistent with the training set, and not constrained to a specific learning algorithm [60]) of the problem is sufficiently complex, then replica symmetry breaking occurs.

The weak point of the Statistical Mechanics approach is that it can be applied so far only to very simple networks, so–called parity and committee machines [18], [130], [157]. The more complicated networks, as for example 2–layer MLP with learnable analogue output weights are a more difficult task to be investigated by Statistical Mechanics' methods. On the other hand, the two–layer MLP is the simplest network structure in practical use. The results, obtained with simplified structures such as a parity and committee machines, give a good general view on what is happening during the symmetrical phase and how it appears.

It should be pointed out, that structures such as parity and committee machines are introducing additional symmetries in the network, because of their fixed and equal hidden–to–output weights and zero biases. In engineering practice such a long symmetrical learning phase is not observed often.

3.2 Flatness by subsequent learning.

As pointed out before, symmetrical learning phases can also occur later in the learning process. As this is not caused by the design of the network but by the design of the experiment, we will first identify the situations that are related to the structure of the problem, which can cause the network to be drawn into a symmetrical phase.

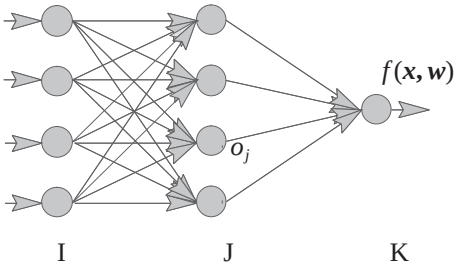


Figure 3–5: *A one–output network.*

3.2.1 Looking at adaptation.

The feedforward neural model shown at figure 3–1 suggests that two equi–potential points on the global error surface have the same cost (equation (3–1)). The network function of a one–output neural network, as represented in figure 3–5, has the following form:

$$f(\boldsymbol{x}, \boldsymbol{w}) = \varphi\Big(\sum_j w_{k,j}.\varphi\Big(\sum_i w_{j,i}x_i\Big)\Big) \tag{3–4}$$

During learning, the weights are successively adapted till the training is completed. One of the ways to model this is by a time series. Abnormal learning will then show by a "longest run" effect as known from classical statistics [57]: changes can be seen to constitute a symmetrical time series, when the effect will eventually become nullified.

The output of a hidden neuron will be needed in the following explanations, thus it can be separately specified. The $j-th$ hidden neuron output equals to:

$$o_j = \varphi\Big(\sum_i w_{ji}x_i\Big) \tag{3–5}$$

and the input of the nonlinearity, associated with the output neuron $\vartheta(\boldsymbol{x}, \boldsymbol{w})$:

$$\vartheta(\boldsymbol{x}, \boldsymbol{w}) = \sum_j w_{k,j}\varphi\Big(\sum_i w_{ji}x_i\Big) \tag{3–6}$$

For the sake of compactness, equation (3–6) can be substituted in (3–4) to give:

$$f(x, w) = \varphi(\vartheta(\boldsymbol{x}, \boldsymbol{w})) \tag{3–7}$$

In order to make conclusions about the equi–potentiality of the points on the global error surface, the error function as specified in (2–4) will be used. Equi–potentiality can be encountered if partial derivatives of the network function to the network parameters are becoming negligibly small, i. e. the error function is flat in some dimensions. Note that $x, \boldsymbol{w}$ are vectors. Consequently, the error surface can be flat in those directions (hyperplanes). The flatness or equi–potentiality of the network function can be in several directions.

3.2.1.1 Weight adaptation in time.

A direct evaluation about different points on the error surface can be made for very simple networks only. Normally, the network weights are used as indication of the error changes, due to the following dependence:

$$\frac{\partial w_{ji}}{\partial t} = -\frac{\partial E}{\partial w_{ji}} \tag{3–8}$$

The weight changes are governed by the *generalization delta rule*, which is:

$$\Delta \boldsymbol{w}_{kj}(t) = -\eta \nabla e(z(t), \boldsymbol{w}) = \eta \delta_k(t) o_j(t) \tag{3–9}$$

Here all the variables determining the weight changes are expressed as being time–dependent. The second multiplicative term in equation (3–9) is the instantaneous error $\delta_k(t)$ – the difference between the desired value f(x,w) and the actual value y of the current example. This term is problem–dependent. Described in more detail, equation (3–9) will change in the following way:

$$\Delta \boldsymbol{w}_{kj}(t) = \eta \cdot (y(t) - f(t)) \cdot \varphi_k'(\vartheta(t)) \cdot o_j(t) \tag{3–10}$$

Substituting equations (3–5) and (3–7) in (3–10) results in:

$$\Delta \boldsymbol{w}_{kj}(t) = \eta(y(t) - \varphi(\vartheta(t))) \cdot \varphi'(\vartheta(t)) \cdot \varphi(\sum_i w_{ji} x_i(t)) \tag{3–11}$$

3.2.1.2 The role of transfer.

The goal of the optimization algorithm, expressed before as minimizing the error function, can be reformulated in terms of weights changes. At the moment that the network reaches a stable point on the error surface, the weight values stabilize. Correspondingly the weight change decreases to zero. The weight change dependence carries information where zeroing of the weights can be expected.

A description in more detail of this dependence shows that the weight change for a hidden–to–output connection contains as a multiplicative term in the weight update of the first derivative of the nodal transfer of neuron i, and the transfer of the neuron j:

$$\Delta \boldsymbol{w}_{kj}(t) \propto \varphi'\big(\sum_j w_{kj} o_j(t)\big) \cdot \varphi\big(\sum_i w_{ji} x_i(t)\big) \tag{3–12}$$

If the network is created from neurons with one and the same transfer function (which is the usual case in neural theory and practice), this multiplicative term has shapes, as shown by the solid lines in figure 3–6 for the most commonly used transfers. The net-

work, composed by logistic sigmoids, has two possibilities to reach a stabilization of the weight change due to this multiplicative term: for both regions where neurons are working in a saturation region. For the zero–centered sigmoid, one more possibility is present: the point where the neurons have zero activation.

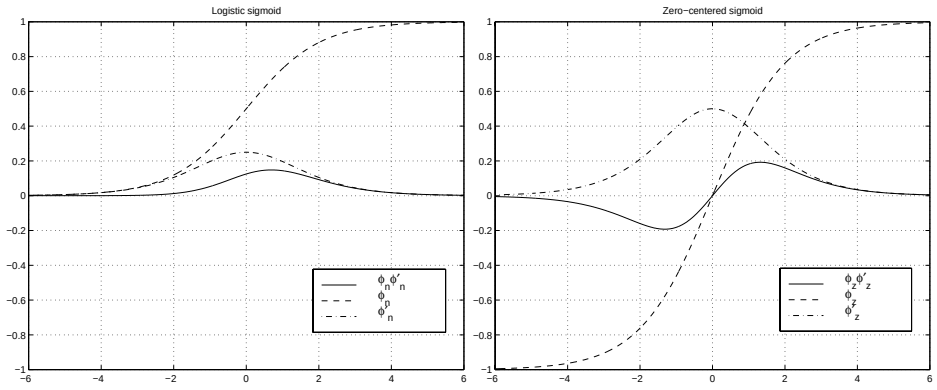


Figure 3–6: *Different sigmoid functions, their first derivative and the adaptation curve.*

We will see later on, that even where saturation is not restricting the learning behavior, learning reliability may be an issue. For the zero–centered sigmoid this can be readily explained from the condition $\varphi'() \cdot \varphi() \simeq 0$. Here also the impact is most directly visible and we will therefore perform most of our experiments with such a sigmoid. However, it seems that also for the logistic sigmoid, some problems can be present, as they tend to display an unreliable learning duration. This is illustrated in figure 3–7, where converging experiments when training perfectly symmetrical functions is performed with differently shifted sigmoids, that show increased learning duration when the transfer functions becomes more zero–centered.

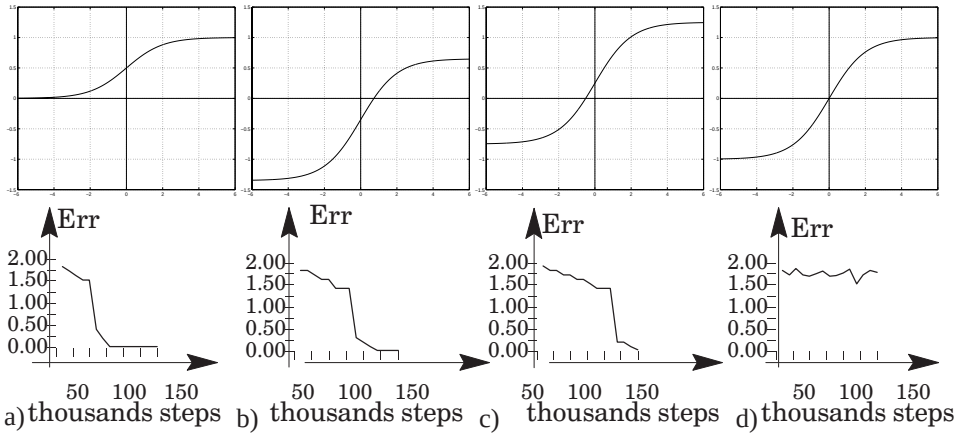


Figure 3–7: *Variable learning duration for some sigmoids.*

We like to point out that where the derivative of $\varphi'() \cdot \varphi()$ may become zero, the change in weight adaptations becomes constant, i.e. when a flat area is entered with a small Δw, the entire passage will be made with this constant but small adaptation. Hence, travel time will depend on the size of the arbitrary adaptation value at the start of the traversion. In other words, the choice of transfer function only does not delimit the learning problem and does not preclude it.

3.2.2 Incomplete adaptation.

Not only can weight changes die out in the course of time, they can also not take place because their backpropagation dies out. In the following we will discuss shortly a number of phenomena that cause such incomplete propagation.

3.2.2.1 Saturation effects.

Since the transfer function is mostly flat (Figure 3–6a,b), its gradient is approximately zero in most of the regions except for the small transitional range. For simplicity, let us assume an one–output network. The network function can be represented by equation (3–4). The output of the network is the value of the output transfer function (on the output sigmoid). If this sigmoid is activated in the saturation area, the operation point will be on the error function's flat region. Correspondingly its first derivative with respect to the weights is almost equal to zero and respectively the resulting weight update is approximately zero, i.e. there are no changes in the network state. This effect is also known as *saturation*, because the transfer function is activated in the saturation areas. Once a node gets into a saturation region, it is (seemingly) trapped due to the extremely small weight updates in the subsequent training cycles.

The above described saturation phenomenon concerns the output layer of the network and can be met by all the known squashed nodal transfers. It should be distinguished from the saturation connected with the hidden layers. The network drawn into saturation is encountered in some classification tasks.

The saturation of the hidden layers is a phenomenon with observable effects, similar to the saturation of the output neurons. This effect is known as *premature saturation*. The premature saturation of the hidden neurons of feedforward neural networks has been investigated intensively in neural learning [36] [38] [61] [97]. The phenomenon is defined as a period of training time, in which the learning error stays almost constant. This definition is not complete, because other phenomena are characterized by the appearance of flat areas [11] not by saturating but by zeroing of the neurons.

Premature saturation is reportedly due to the improperly chosen initial weights with respect to the input parameters [97]. The authors derive the probability of incorrect saturation as a function of the range of initial weights, the number of nodes on the hidden layer and the maximum slope of the activation function. This approach does not give an answer to the question: why tuning the network parameters is crucial in some problems and absolutely not necessary in other problems of a similar complexity? More advanced in an historical and conceptual sense is the work of [61]. The authors find as an undesirable growth of some weights the relationship between the network parameters and the data to be learned.

In [61] is claimed that when a problem is difficult to learn for a network, saturation can be observed. A general definition of 'difficult' to learn task is not given. Instead, an ex-

ample of a difficult task, learning of statistically independent input and target data, is given. The method is derived from the *parity problem* [104] (i.e. XOR and multidimensional parity problem), where the input and target values are statistically independent. Unfortunately, the authors do not give any experimental support of their theory on either real or artificial training data, and from the context it is not becoming clear whether their conclusions are restricted only to the artificial parity problem and can be caused from the small size of the training set (imagine the XOR problem: only 4 examples are repeatedly presented to the network; the generalization issues for the parity problem do not make sense either).

3.2.2.2 Numerical influences.

The tangent hyperbolic can be represented as a Taylor series:

$$\varphi(x) = \tanh(x) = x - \frac{x^3}{3} + \frac{2x^5}{15} - ..., |u| < \frac{\pi}{2} \qquad (3\text{–}13)$$

If the value of x is close to zero, then $tanh(x) \approx x$. Correspondingly, the neuron operates in its linear region – its output is its scaled summary input. This neuron is not working effectively in the nonlinear model. If two or more such a neurons exist, they can be pruned to 1. This case corresponds to approaching the second zero point of the function from figure 3–6. The small weights phenomena is peculiar to the zero–centered sigmoid transfer.

In this case the network parameters are not brought to saturation or nulled. The reason for insufficient update of the network parameters is due to the gradients passed from the upper layer. If these gradients are balanced or if all of the upper layers are saturated, the summary weight update is equal to zero. In this case $\sum_{i=1}^{H} \delta_i = 0$, where H is the number of hidden units.

Another numerical influence becomes apparent when the resolution of the numerical representation is decreased. Again one finds that learning duration increases [96]. Further [51] and [95] note that with decreasing resolution the shape of the error surface changes. It is even possible that *paralysis* (or stand–still) occurs when the errors become smaller than can be represented with the current resolution.

3.2.2.3 Badly balanced training set.

A badly balanced training set with the majority of examples concentrated in a small area finally brings the adaptation of the network state parameters to zero. The network parameters are not claimed to approach infinity or zero. The similarity between the input stimuli and a big difference between the corresponding target makes the network to unify its response. The weight values degrade.

Algorithms of the backpropagation type are not very well suited for applications, where a constant stream of new data are to be learned. As shown by [59], presenting constantly new examples can cause forgetting of the previously learned ones (*unlearning)*. According to the author, the cause of this forgetting is the overlap between the representations of the different patterns, which makes the backpropagation to produce semi–distributed representations. In these representations, only a few hidden units take a value different from zero. The reduced distributedness of the internal representations has the effect of loosing the most interesting neural features – the generalization ability and damage resistance.

One of the reasons for degradation of the weight vector to a few values, which do not adapt further on, can be the symmetry in the examples shown to the network. The interconnection between network and problem symmetries is becoming apparent when the learning process reaches a symmetrical element of the signal under training. To show this clearly, a symmetrical target will be used for training. Firstly, some considerations about the error surface shape in the beginning of the learning process will be given. Further on, the behavior of a symmetrical function in such an area will be revealed. The longest run effect of the symmetries of random distributions will be shown next. Later on the analogous preconditions in latter stages of learning will be proposed. And finally, examples of similar behavior will be shown to appear in later stages of learning.

3.3 Learning scenarios causing degradation.

There is little published research on the locality / distributedness of internal representations in neural networks. The *local learning* concept accepts, that there are links or associations between sets of adjustable parameters and regions of the input space. If the representation of an input pattern on the hidden layer of neurons is optimally distributed between the neurons, there is a considerable interaction among the representations. This interaction has two sides: it contributes to the network generalization abilities, but, on the other hand, it predisposes the network to *catastrophic forgetting*.

Def. 3−4: Catastrophic forgetting is a direct consequence of the overlap of distributed representations and can be reduced by decreasing this overlap (see [59]).

A related term is *interference* of a neural network. The interference characterizes internal representations and appears when learning new information interferes with old information. Networks, that are less sensitive to interference, are referred to as *spatially local*. In [29], this phenomenon is described as *internal conflict* in contrast to the *external conflicts* that are directly distinguishable in the data set itself.

It is self–evident that very local representations will not exhibit catastrophic forgetting, because the interaction among the representations is very small. New information can be taught without interference with the already learned (old) information. The trade–off here is that an entirely local representation excludes the possibility the network to generalize.

A solution of the so–described dilemma is a *semi–distributed representation*. Weaver [155] has proved a useful theorem, which shows that, given an arbitrary large number of weights, a single–hidden–layer Perceptron network with sigmoidal activation function exists that is as local as desired. Formulated in another way, learning at one point will affect another point to a desired small degree which confirms the universal approximation abilities of a single–hidden–layer multilayer perceptron. Also measures of locality of the network representation and interference are suggested in this recent work of Weaver [155]. These measures describe the interference at point x' due to the learning at x. Both measures are based on knowing the gradient of the network output with respect to the weight vector in points x' and x.

We have found one typical display of weight vector degradation when the symmetry of the network is increased by, for instance, learning from a symmetrical training set.

The cause can be explained by interference of the training examples and thus with appearance of interfering internal representations. Here, interference issues will not be elaborated on, because the developed measures and theory is by far insufficient to say more about how to treat the interference problem, or how to predict when a interference will appear. Our practical meaningful criterion, which finally prevents from degradation of the weight vector seem to be very related to this work.

An example will be described in detail, starting with its "pre–history", because it helps to generalize for a larger group of problems. It converges with a weight vector degraded to a different extent, and gives a practical dimension of the abstractly described problem of "catastrophic forgetting", interference, localization and distributedness. Normally, these do not cause severe problems to the learning evolution, because the training set content (the third error surface–forming factor), together with the learning algorithm, overcomes this symmetry.

There are a lot of cases, when this normal development of the learning evolution can be delayed, stopped, or brought to a suboptimal solution. Such difficult problems and their possible solutions have been discussed in literature. Typical learning difficulties, which can cause poor or unreliable learning for problems, whose complexity is surmountable for the network, will be discussed further on. In other words, not the complexity of the problem but its nature is causing the bad learning performance.

3.3.1 Symmetries in the patterns.

For neural network tasks, problems are described by sets of examples (training sets). Symmetries in the patterns are considered as equivalent terms of the symmetries in the problem. The problem symmetries can occur in space and in time domain. The space symmetry may occur within a single pattern in classification. This symmetry will cause that a hidden neuron may be triggered by mutually conflicting stimuli.

In the latter domain, we focus on the presentation order resulting from random sampling and/or from time–sequencing; this appears primarily in approximation and prediction. The time domain symmetry will then cause either poor approximation or unlearning of the previous examples.

In the following we will start by discussing a number of early and constructive applications of symmetry. This illustrates that there are positive sides to what we will further treat as a negative characteristic of neural networks: its sensitivity to symmetry.

3.3.1.1 Spatial symmetries in patterns.

The difficulties of the first–order perceptrons and Hopfield networks to recognize mirrored, rotational and translational symmetries have been addressed in [136]. The authors elaborate on the *order of a problem* which roughly corresponds to the minimum number of input units, that carry any information as relevant to the required output [104]. For example, the Boolean OR function has order 1 because the value of a single input unit contains enough information about the output. The Boolean XOR function has order 2 because each unit of the input array, considered in isolation, contains insufficient information about the output value. The generalized XOR problem, also known as parity function of n binary inputs, has order n. Higher–order problems can be solved by adding hidden units. It is suggested that the XOR is an easily solvable second–order problem if only one hidden unit is added to the network.

[136] has attempted to solve the *mirror symmetry problem*: the network must detect which one of the three possible axes of symmetry is present in an *NxN* binary pixel input pattern. It has been proved that introducing hidden neurons makes symmetry detection an easy task. A network with only two hidden neurons can already establish this property regardless of the size of the input string (Figure 3–8a). For every hidden unit, the weights are chosen with odd–symmetric values. This means, that if a symmetric pattern is applied, both hidden units will receive a null summary input and their output will be also zero, because of their negative bias values. The output unit, with a positive bias in this case, will be equal to one.

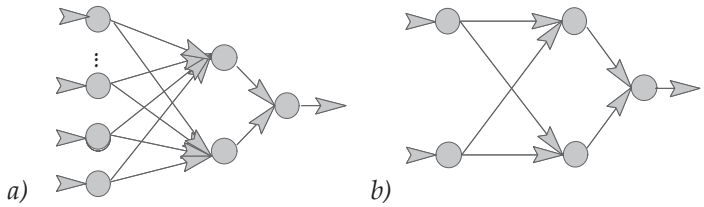


Figure 3–8: *a)Rumelhart's input symmetry detector; b) Blum's XOR network.*

Another observation made by [136] is that the weights on each side of the midpoint of the string are in the ratio 1:2:4. This ensures that each of the eight patterns applied to the network sends an unique activation sum to the hidden unit. Therefore there is no pattern on the left, which will balance a non–mirrored pattern in the right–hand side.

When a symmetry pattern is applied, there is a second–order problem, because single pixels alone can not carry information about the solution of the problem. Sufficient information can be extracted from pixel pairs that are related to the mirror symmetries. The authors demonstrated, that the Boltzmann learning algorithm is capable of solving the mirror symmetry problem.

Later on, [120] made a comparative study of the mirror symmetry problem using the Boltzmann machine, Mean–Field Theory machine and the Multilayer Perceptron. The Mean–Field Theory machine offers superior results to the Boltzman machine and slightly better than the Multilayer Perceptron. The results point to the interesting property that, by using relative entropy, Mean–Field Theory and a Multilayer Perceptron have an approximately similar performance if they both have a single hidden layer and the size of the input layer is much larger than that of the output layer.

In further studies, the effects of introducing symmetries to feedforward neural networks are investigated [139]. The *group–invariance theorem* [104] was referred to by the author. From a practical point of view this approach may provide a method to simplify training in those cases where the target function is known to be invariant under the action of a particular group of inputs. This aspect of the group invariance theorem has been generalized for Multilayer Perceptrons in [139]. The approach aims at simplifying the training task by making the symmetries a priori explicit in the network structure. One example with symmetrical weight assignments is the network, designed to perform the XOR problem, as shown in figure 3–8b. This network is invariant to permutation for components of the input vector and so it could be trained to compute the XOR function with 5 parameters and 3 examples instead of the original 9 parameters and 5 examples [22].

3.3.1.2 Temporal symmetries in patterns.

Since [129] shows that easy and elegant solutions will exist for classifying symmetric input strings, [22] attempts to approximate symmetric target sets. The existence of a manifold of exact neural solutions for 2–variable Boolean functions has been proved. Furthermore there exists a manifold of local minima of the mean–square error E. It is concluded that for linear–separable problems a learning procedure forms stationary points, but these are not local minima and therefore can not cause serious convergence problems.

The described symmetry problem concerns a classification task with neural nets using a nonsymmetrical activation function. The XOR net used by [22] shows that the null weights configuration can also be an attractor during learning. As a consequence they propose to follow Rumelhart's suggestion of setting up the initial weights with small random values. This approach prevents failures in the most cases.

It is shown that the specific structure of the training set of real–life problems may have symmetries induced by the "longest run" potential [10]. The longest run induced symmetries can lead to slow convergence, unpredictable training duration or paralysis.

A popular way to present temporal signals to neural networks is over a tapped delay–line. Design decisions are based on the size of the delay line and the sampling rate. Inadvertedly one may thus introduce temporal symmetries and end up with learning problems. It has been discussed to monitor this circumstance from the curvature of the input/output plot.

The effect of initialization, where symmetry can play a misleading role, is one of the areas scarcely researched in training temporal patterns. Different initializations can easily lead to a different behavior: a phenomenon that bears likeness to chaos in *system dynamics*. Resolving symmetry effects for learning can therefore hardly be based on initialization.

3.3.2 Symmetry in the error surface.

As explained in Section 3.1.1, both sigmoid transfer functions are odd–symmetrical. This implies, that at the origin, all their even derivatives are equal to zero ($\varphi^{2k}(0) = 0$) and their odd derivatives are different from zero ($\varphi^{2k+1}(0) \neq 0$). Because the quadratic error function is the average of a number of squared estimation errors, the local error function $\mathcal{E}(w)$ can be considered without loss of generality to be:

$$\mathcal{E}(w) = [y - f(x, \boldsymbol{w})]^2 \tag{3–14}$$

3.3.2.1 Extrema.

The curvature of the error surface can be estimated by the second–order derivatives of the cost function with respect to different parameters w. In [65], it is shown that a network with a linear output neuron and only one hidden neuron can be used for investigating the shape of the error surface curvature. The one hidden neuron restriction is possible, because the higher–order derivatives, involving parameters associated with two different hidden cells, are zero anyway. The transfer function in this case will look like:

$$f(\boldsymbol{x}, \boldsymbol{w}) = W.\varphi(\boldsymbol{w}.x + w_0) + W_0 \tag{3–15}$$

where x and w are vectors. The first derivative of the local error with respect to any parameter w (weight or bias parameter) is equal to:

$$\frac{\partial \mathcal{E}}{\partial w} = -2[y - f(\boldsymbol{x}, \boldsymbol{w})]\frac{\partial f}{\partial w} \tag{3–16}$$

where $\partial f/\partial w$ represents the partial derivative of the network function with respect to the different weights:

$$\frac{\partial f}{\partial w_i} = W\varphi'(\boldsymbol{w}.x + w_0).\boldsymbol{x} \tag{3–17}$$

or the partial derivative of the network function with respect to the different biases:

$$\frac{\partial f}{\partial w_{0i}} = W\varphi'(w.x + w_0) \tag{3–18}$$

or the partial derivative with respect to the output weight:

$$\frac{\partial f}{\partial W} = \varphi(w.x + w_0) \tag{3–19}$$

or the partial derivative with respect to the output bias which equals to:

$$\frac{\partial f}{\partial W_0} = 1 \tag{3–20}$$

In the inert state (the point where all the weights are zero), all the first derivatives except the derivative to the output bias are 0. In [65], it is proved, that every odd, infinitely differentiable transfer function φ, which has by definition all even derivatives equal to zero, has zero–valued odd derivatives of the quadratic cost function with respect to all parameters except possibly to the output bias. The Hessian is not positive and not definite, which means, that this is not a minimum of the cost function. Because the odd–order derivatives of the cost (error) function are zero, this point is not an inflection point, but rather a saddle point.

3.3.2.2 The saddle point.

It can be concluded, that the zero point is a kind of a N– dimensional saddle point, and the optimization will take place on the direction of the output bias W_0, which provides the only non–zero derivative, i.e. the only direction with some steepness. In this point the equation (3–15) will change to $f(x) = W_0$. The corresponding derivative of the cost function :

$$\frac{\partial E}{\partial W_0} = \frac{1}{N}\sum_{\mu=1}^{N}[y^{\mu} - f(x^{\mu}, \boldsymbol{w})] \tag{3–21}$$

It is obvious that the minimum state of the quadratic cost $E(w)$ in this direction is obtained for $W_0 = \bar{y}$ over the N examples.

This line of reasoning can be continued to define the kind of saddle point. Refering to equation (3–21), the following simple observations can be made. In the neighborhood of the zero point, the value of the error function (equation (3–14)) will be higher than its value in the origin: if the weight $w_{ij} > 0$ with other parameters fixed, $f(x, \boldsymbol{w}) > f(x, \boldsymbol{0})$. Correspondingly, $(y(x) - f(x, \boldsymbol{w})) < (y(x) - f(x, \boldsymbol{0}))$. Since the training patterns (the y–values) and the network output values are scaled to have a smaller than unity absolute value, for the squared difference (and correspondingly for the error function) holds the inequality $(y(x) - f(x, \boldsymbol{w}))^2 > (y(x) - f(x, \boldsymbol{0}))^2$ for all the

parameters, except for the output bias W_0. Represented graphically, the error surface in the neighborhood of the origin has the shape of a valley. One example of a valley is shown at figure 3–9.

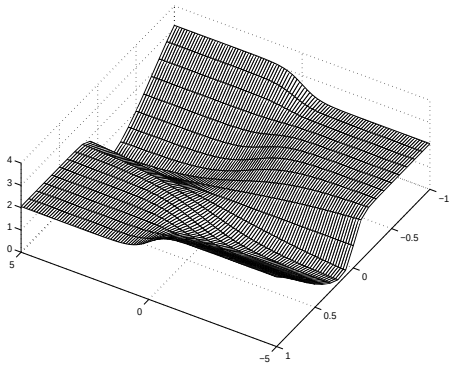


Figure 3–9: *A possible error surface shape near the zero point.*

The saddle points are peculiar to the different stages of learning. The characteristic example of such a case was encountered in the beginning of the learning process. As already discussed before, neural networks are initialized with random values from a small, symmetrical interval around the zero point. If the zero point is a saddle surrounded with steep parts in every direction but the output bias direction, the initialization with small values will position the starting point of the learning process on the slopes of this saddle.

The saddle area is an area with high curvature along the most of the weight dimensions and possibly flatness at one, that defines the bottom of the saddle. As explained in Section 1.2.2 this high curvature causes large gradient vectors. This is the reason that the optimization process does not converge to the zero point – to the saddle bottom. An example of such an area and the behavior of the gradient algorithm in it is shown for the 2–dimensional parameter vector in Fig.3–10a. The direction of the gradient of the parameters in this area changes often.

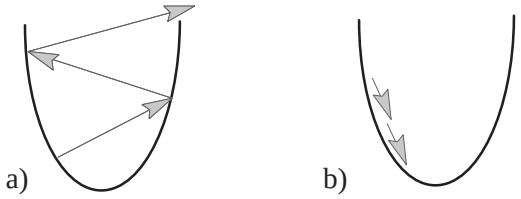


Figure 3–10: *The behavior of the gradient algorithm in the shallow valley. High steepness of the sope allows the network easily to escape the valley.*

The saddle points, as typical at the beginning of the learning process, as in later stages of learning, cause unreliable performance of the network. (The most problematic kind of saddle area is a valley – where the flat area is the one with a minimum energy.) Obviously they are not an obstacle for the learning process in general. When different sym-

metrizing factors exist, the typical behavior of the learning algorithm in this points has changed to a degradation of the weight vector to zero or to an internal representation with a low distributedness [11]. Difficulties can be present for some classes of functions with explicit or hidden symmetry.

3.3.3 Symmetrical signals on problematic regions.

The elaborated difficulties for the learning trajectory are slowing down the learning process. The nature of the signal to be learned can contribute to this slowing down effect and may even lead to paralysis. One example for such a result is the training of a symmetrical set.

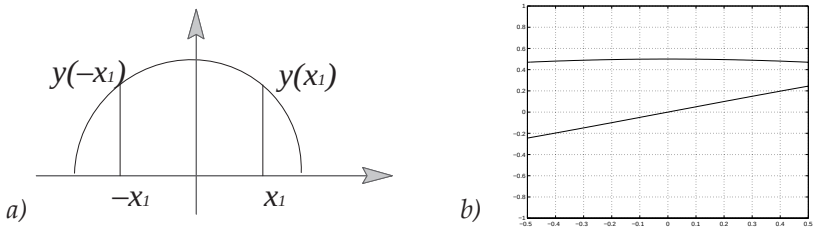


Figure 3–11: *a) Two equidistant points of the symmetrical function; b) The network output and the adaptation slope at the beginning of training.*

If an even–symmetrical function is centered around zero, it has the following characteristic (Figure 3–11a): if x^1 and x^2 are two equidistant from the center of the function points, their values are such that $x^2 = -x^1$ and $y(x^1) = y(-x^1)$. A number of variations exist for the shown function symmetry. In figure 3–12, we give some straightforward examples.

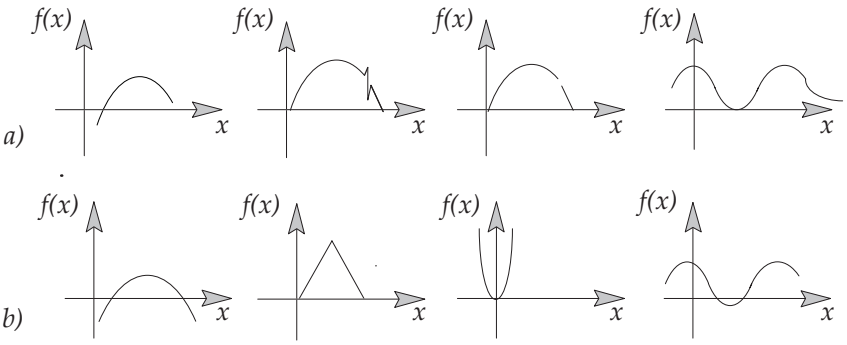


Figure 3–12: *Some examples of a) non–problematic and b) difficult to train (symmetric) signals.*

The uniform symmetrical initialization of the network parameters ensures, that during learning the training signal will be effectively scaled according to the requirements of the nodal transfer, i.e. with the center in the origin of the coordinate system (Figure

3–11a). For better illustration of the forthcoming reasoning, the architecture of the network used for the experiments is shown at figure 3–13.

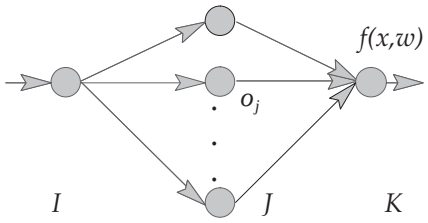


Figure 3–13: *A one–input, one–output feedforward network.*

The corresponding o_j and $\vartheta(x, \boldsymbol{w})$, and the network function, are:

$$o_j = \varphi\left(w_{ji}x\right) \tag{3–22}$$

$$\vartheta(\boldsymbol{x}, \boldsymbol{w}) = \sum_j w_{k,j}\varphi\left(w_{ji}x\right) \tag{3–23}$$

$$f(\boldsymbol{x}, \boldsymbol{w}) = \varphi\left(\sum_j w_{k,j}\varphi\left(w_{ji}x\right)\right) \tag{3–24}$$

For the zero–centered sigmoid and fixed parameter values holds $f(x^1) = -f(-x^1)$ and $\varphi'(\vartheta^1) = \varphi'(\vartheta^2)$. If two symmetrically positioned points are chosen for training examples $z^1 = (x^1, y^1)$ and $z^2 = (x^2, y^2)$ and presented at the network in sequential order, they will cause a change that can be represented on the global error surface as the sum of the changes Δw_1 and Δw_2:

$$\Delta w_1 + \Delta w_2 = \tag{3–25}$$
$$[y(x^1) - f(x^1, \boldsymbol{w})]\varphi'(\vartheta^1)\varphi(w_{ji}x^1) - [y(x^1) + f(x^1, \boldsymbol{w})]\varphi'(\vartheta^1)\varphi(w_{ji}x^1)$$

where the partial derivatives are with respect to the arbitrary parameter w. The terms $\varphi'()\cdot\varphi()$ is the multiplicative term from the equation (3–12). For two symmetrical examples (let's note once more that the signals in neural training are usually scaled to unity) and a zero–centered sigmoid, the multiplicative terms are with equal magnitude and opposite signs (see figure 3–6d). Then the summary change will take the value:

$$\Delta w_1 + \Delta w_2 = -2f(x^1, w)\varphi'(\vartheta^1)\varphi(w_{ji}x^1) \tag{3–26}$$

i.e. it depends only on the network output and its derivative values. In this case the target (teacher) signal does not influence the weight update. As can be concluded from figure 3–6b, the value of $f(x^1, w)$ is very small in the beginning of the learning process. As a result, the sum of the two subsequent weight changes will have the effect of performing the optimization with a smaller than optimal learning rate or is equivalent to smoothing the error surface – the optimization will be directed towards the bottom of the valley (Figure 3–10b), and the further optimization will follow the valley bottom. The initial valley is characterized with a very high error value. If the signal under the consideration is not symmetrical, the points x^1 and $-x^1$ correspond to different $y(x^1)$ and $y(-x^1)$ values, and consequently their summary contributions to the network output change are not zero. The summary output value, multiplied with a large gradient vector will cause

behavior, similar to the one shown at Fig 3–10a. As a result, the network state will swiftly escape the valley.

Of course the possibility for two symmetrically positioned examples to be presented to the network in the subsequent order is very low. However, it can be shown that presentation of the training examples in a random order, a common practice in neural learning, have similar summary effect (see appendix B).

In the simple example of two symmetrical training signals it was assumed, that the current weight value is independent of the history of the learning process. Also, the adaptation of the network parameters between two example presentations was ignored.

To understand the global impact of symmetrical training set D_n on the learning development, let us elaborate on the effect of a subsequent presentation of patterns. The equality of the product $\delta_k(t)o_j(t)$ to $-\partial\mathcal{E}(t)/\partial w_{kj}(t)$ can be seen from the derivation of the backpropagation algorithm. Considering the recursiveness of the weight updating procedure, equation (3–11) can be rewritten as a time series with index t and length n:

$$\Delta w_{kj}(n) = -\eta \sum_{t=0}^{n} \alpha^{n-t} \frac{\partial\mathcal{E}(t)}{\partial w_{kj}(t)} \tag{3–27}$$

Here, $\Delta w_{kj}(n)$ is an exponentially weighted sum. When subsequent partial derivatives $\partial\mathcal{E}(t)/\partial w_{kj}(t)$ have the same sign, $\Delta w_{kj}(n)$ grows in magnitude, thus weights are adjusted by a large amount. When the partial derivatives $\partial\mathcal{E}(t)/\partial w_{kj}(t)$ have opposite signs on subsequent iterations, $\Delta w_{kj}(n)$ shrinks in magnitude, which presumes a small adjustment of the weight values. This description of the weight changes behavior can be made more specific in the following way. Let us assume that n uniformly distributed samples are taken from the input signal. If the samples are taken between –1 and 1, there are $n/2$ samples x^i between –1 and 0, and $n/2$ samples are between 0 and +1, such that for every sample x^i there exist a sample x^j, $i \neq j$, for which $x^i = -x^j$. If the signal to be learned is symmetrical, to every tuple x^i, x^j, $(x^i = -x^j)$ corresponds a y^i, y^j, tuple, such that $(y^i = y^j)$.

The probability, that the network will get two such a samples in subsequent training steps is not very high. On the contrary, the probability for samples presented in a longer run training to contain a large number of tuples which fairly satisfy the made assumption is very high. From equations (3–10) and (3–27), the following equation can easily be derived:

$$\Delta w_{kj}(t) = \eta \sum_{n=0}^{t} \frac{\alpha^{t-n}}{2} y(n)\varphi'(\vartheta(n))o_j(n) - \eta \sum_{n=0}^{t} \frac{\alpha^{t-n}}{2} \varphi(\vartheta(n))\varphi'(\vartheta(n))o_j(n) \tag{3–28}$$

This suggests the following conclusions. In accordance with the assumptions the first additive term of this equation has a negligible influence on the learning process. Since the learning develops practically without teacher, the second term stays also very small, because learning starts with small parameters – correspondingly with small network function values – and it is not stimulated to grow any further. As a result, the optimization in the shallow valley region is done towards its bottom. Such an optimization is equivalent to training with a very small learning rate (see figure 3–10b). The weight changes are becoming negligible. If this phenomenon occurs in the initial phase of the

learning process, the weight values degrade to the zero point. The network capacity is reduced and further optimization is not possible anymore.

If the training set is not symmetrical, the first additive component of equation (3–28) does not tend toward zero and the optimization takes place even if the network is in the stationary area, as for example in the beginning of the learning process. In terms of the reasoning earlier in this section, the dependencies derived here will ensure altering of the sign of the last multiplicative term and of $f(x, w)$ simultaneously and practically at every iteration (Figure 3–6b). (Note that $\varphi'(x, w)$ is always positive.) The shrinkage of the weight change $\Delta w_{kj}(n)$ will be due to $\varphi'()\cdot\varphi()$.

Until now, the bias of the neurons was modelled as a extra node with an unity weight (for instance in equation (3–4). If the bias is represented explicitly, then equation (3–4) will change in the following way:

$$f(x, \boldsymbol{w}) = \varphi\Big(\sum_j (w_{kj}\varphi\Big(\sum_i (w_{ji}x^i + Q_j)\Big) + Q_k)\Big) \qquad (3–29)$$

After long enough training of a symmetrical signal, the network output will become equal to the values, taken by the output bias. Finally the network output value (resp. the bias value) will converge to the average of the target signal. Further optimization is performed only in the output bias direction –the only nonzero term left.

3.3.3.1 Training two identical networks.

The analysis made so far is confirmed in detail by the performed experiments. The appearance of such effects in the beginning of the learning process can be illustrated by training of two functions – one fully symmetrical and another with slightly destroyed symmetry (see figure 3–14a).

Two identical networks (networks with the same structure, transfer function, trained with the same learning algorithm and initialized with equal parameter values) are used. The training examples are endlessly supplied from a uniform random generator. This way the assumption about the existence of antagonistic tuples of examples is fulfilled. The first set contains random samples from a symmetrical function; the second has the same random input string as an input vector and their corresponding target values drawn from the slightly disturbed symmetry of the original target function. The symmetry of the target function can be destroyed in many ways. Here one of them is used, which does not change the input part of the training set: both networks are trained with the same input vector, as visualized partly at the figure 3–14c.

The complexity of both tasks is comparable. The initial state of the network, as the stimuli, which it obtains are also identical. The outcome of the experiments differs enormously (Figure 3–14a, the dashed lines). After practically endless training of a perfectly symmetrical function the network produces a straight line output, while in the case with slightly destroyed symmetry the approximation is precise.

The elements from the first term in the equation (3–28) are canceling each other's impact and the instantaneous error does not grow during learning. The summed difference between the network target and output (equation (3–30)) oscillates around the zero value for the period of one complete presentation of a symmetrical pattern set (Figure 3–14d, the solid line). The repeating period of the random generator is estimated to be around 1000 values. In contrast, when a nonsymmetrical target is trained, the sum of the instant errors grows very quickly (Figure 3–14d, the dashed line).

$$\mathcal{E}(n) = \sum_{t=1}^{t=n} (y(t) - f(x(t), w)) \approx f(x(t), w)). \qquad (3\text{–}30)$$

The summary error for symmetrical function, plotted with a solid line in figure 3–14d, is a typical recording for many performed experiments. As already mentioned, the dashed line shows the sum of errors for a non–symmetrical function, also taken by an average performance of many subsequent tests. Because of the random manner of sampling of a symmetrical function, the example presentation results in almost equal changes of the error values in positive and negative direction, and consequently in canceling to a high degree the corrections, made in the previous learning step. This effect we call *cancelation* [10].

In the simple experiment, displayed in figure 3–14a, cancelation leads to bad approximation – the network gives a straight line output, if a random equidistant sampling is done. This result concerns the beginning of the learning process, when the learning algorithm starts in the initial valley, positioned symmetrically between the different parameter spaces of the error surface. A similar effect can develop in later stages of training as well. In the first two stages of the learning process the valley bottoms and plateaus are very likely to have high energy values and to bring the convergence process to a bad solution. The weight values are not updated anymore and they are oscillating around a suboptimal solution. If this process happens in the beginning of the learning process, the weight values are reduced to the bottom of the initial valley area, which is symmetrically situated between all parameter subspaces. This valley bottom corresponds to the output bias parameter, the parameter, whose derivatives are not nulled. If this process develops in later stages of training, the weight values oscillate endlessly around the reached value.

In training more complex signals, cancellation affects learning time or it is also a reason for poor approximation. Cancelation is usually encountered at later stages of learning, when secondary symmetries in the training set become apparent. Examples of such a signals will be given in the next chapter, where a criterion for discovering the cancelation example sets will be formulated.

It is important to point out, that networks with zero–centered sigmoid neurons suffer much more from cancelation phenomena than nonsymmetrical sigmoid networks. The reason can be illustrated with the multiplicative term from equation (3–12) (see also figure 3–6a,b). Since by the zero–centered sigmoid the product of the nodal transfer and its derivative have an opposite sign, if the negative example (example with a minus sign) is inputted to the network. In contrast, the multiplicative term takes only positive values for non–symmetrical sigmoid (equation (3–12) and also figure 3–6c). The assumption that the input example should be negative or positive has only a illustrative meaning. If all the training examples are scaled in the interval [0...1], the symmetrical initialization around the zero point translates them to the negative and the positive part of the sigmoid nonlinearities.

‘

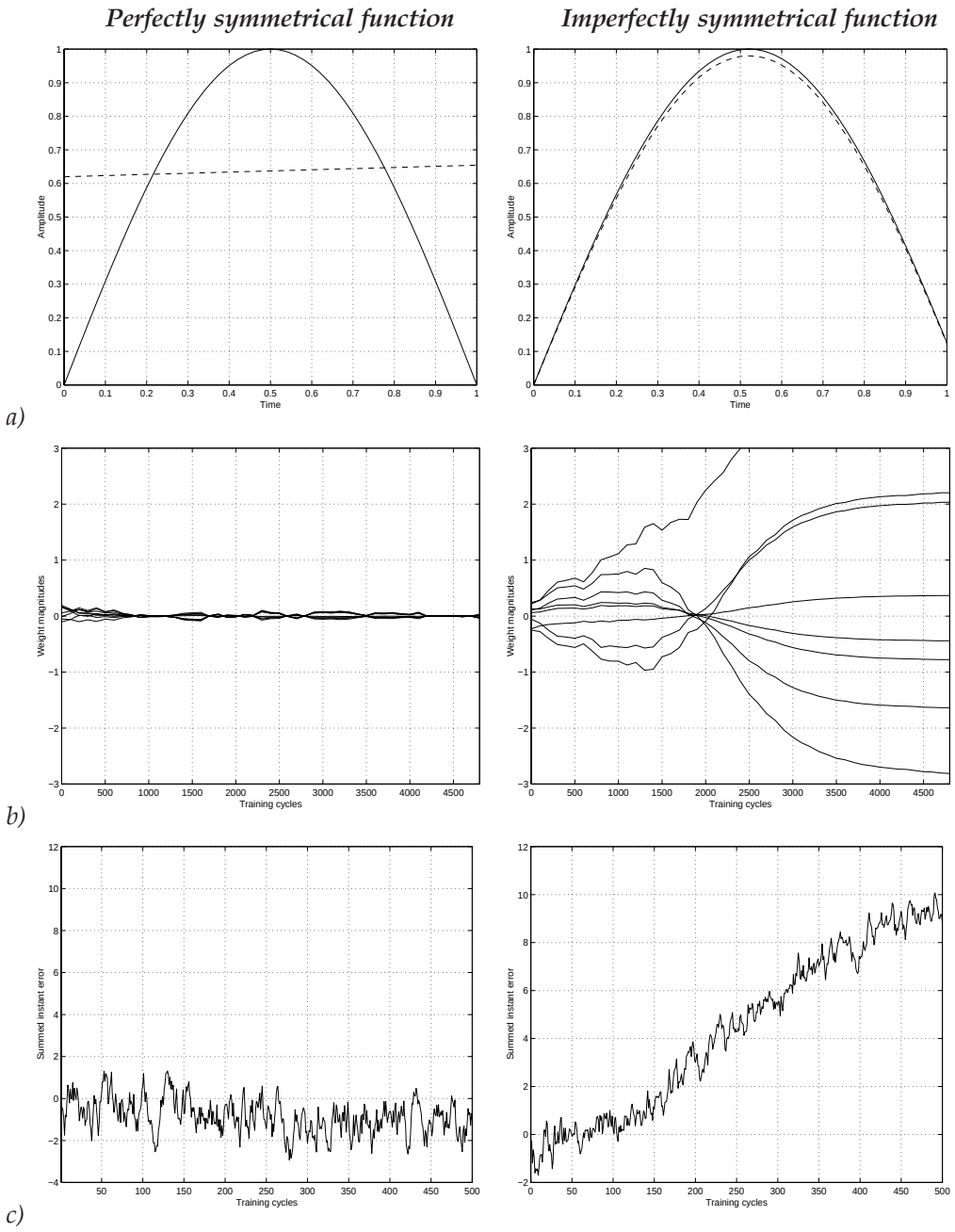


Figure 3–14: *Training for a perfect and an imperfect symmetrical function: a)target function (solid) and approximation (dashed); b)weight vector evolution; c) summed instant error.*

3.3.3.2 Error landscape symmetries and degradations.

The described symmetries in the network cause indecisiveness of the learning algorithm either because of equal possibilities to choose between a multidimensional subspaces or because of the flatness of the error relief in the later stages of learning. All the described cases in sections 3.1.1, 3.2 and 3.3.2 may be relegated to three cases: zeroing of the weights, also known as a degradation phenomenon, saturation, and balancing of the weight values. Within the network, hidden units represent the momentary grasp which the neural system has on the problem at hand. The functions, represented by the hidden units, form the internal representation of the problem. This internal representation can have a different degree of distributedness. With radial–basis networks, the internal representations are very localized – there is a high degree of correspondence between the training signal features and the different nodes of the hidden layer. It has been generally accepted, that the single hidden layer multilayer perceptron with a sigmoid activation does not have a localized representation, although there are similar internal mechanisms, which guide the MLP and RBF feedforward networks, trained with back–propagation algorithm. In the recent paper of Weaver [155], this misconception is addressed. It is shown that the internal representation of MLP can be made as local as desired.

The globality or such distributivity of the internal representation implies phenomena such as interference and cross–coupling between the hidden units, which results in problems of local minima or nearly flat regions in the error function, arising from near cancellation of the effects of different weights. The cancelation phenomena, expressed by parallel and nearly–zero symmetrical trajectories or degraded weight vectors can be caused by many different factors, working in combination. Crucial is the influence of problem features and its representation. Few difficult problem–induced cases were shown in previous sections.

The degradation concerns the capacity of the network. During learning, network parameters are storing the information, contained in the examples. A certain capacity measure can be associated with the network, which limits the information that can be stored in it. The capacity of the network, or also known as the degree of freedom of the network, is determining the complexity of the problem, which the network can fit. Correspondingly, in order to ensure proper working of the network, a first condition is to design a network with a large enough capacity to fit the problem.

The capacity of the network is often related to the number of units (respectively parameters), which store the information from the examples. For MLP the number of the hidden layer units are controlling its capacity. A reasonable conclusion, derived from this fact, is that a basic design principle will be to construct networks with a very large number of hidden neurons. Such a solution will contradict the conclusions in Section 3.1.1 that a too wide hidden layer increases the symmetry of the network parameters. As already explained in Section 3.1.1, the symmetries in the network parameters can additionally enforce degradation. In cases, where there are not additional conditions for degradation, the overparametrising can cause decreased generalization capabilities of the network: the network tries to map even the noise in the training data.

During training, due to symmetries in the patterns or other reasons, degradation of the network can occur. This is equivalent to disconnection of links between neurons. The individual connections are stuck–at–0 and remain there practically for ever. An exam-

ple for such stuck–at–0 faults is shown in figure 3–15a. Effectively, this phenomenon is equivalent to reducing the capacity of the network. In extreme situations it can apply to all the neurons in the network. An example of such a complete degradation is illustrated in figure 3–15a. A partial degradation of the weight vector, when the network brings most of the weights to zero, and the remaining weights evolution is shown in figure 3–15b for training a network, initialized with small random weights from a symmetrical distribution, with samples from a symmetrical signal.

The behavior of the network, trained to map a signal, which has symmetrical components, is shown in figure 3–15c via the weight vector evolution. Here, the main tendency of the signal has been learned by the network, but the learning process does not progress any further. The adaptation of the weights has stopped, and most of them have taken the two overlapping parallel trajectories. (In the picture all the 8 hidden–to–output weights are plotted). Another way of pattern presentation of the same signal brings most of the weights to zero and leaves 3 to adapt until they map the feature in their capacity.

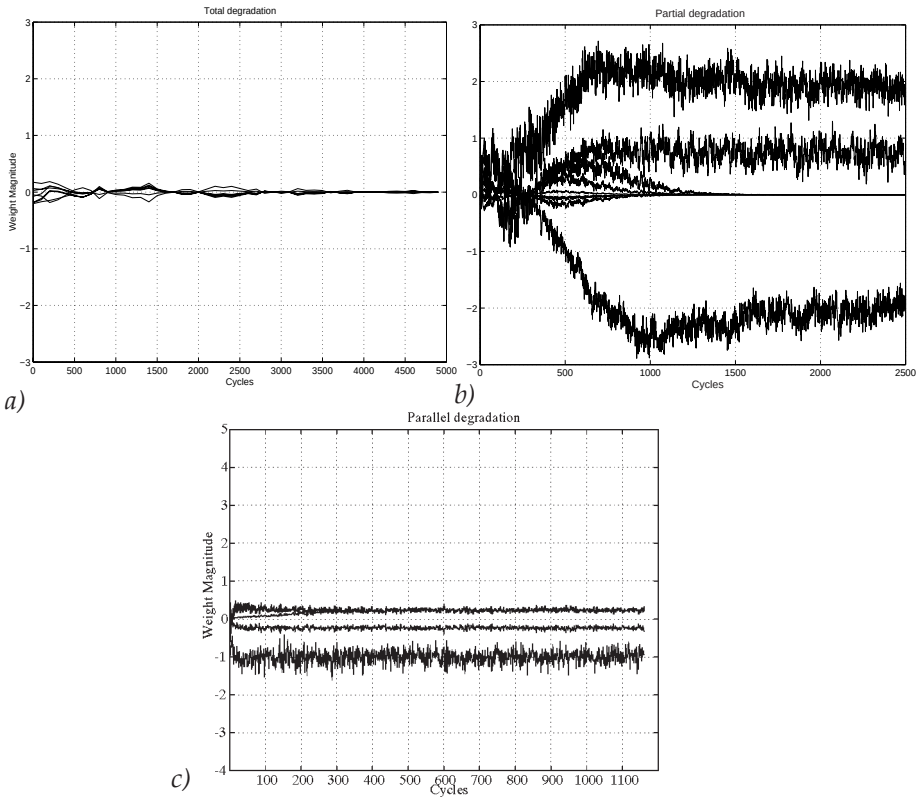


Figure 3–15: *Examples of degraded weight evolution.*

The observed phenomena is said to be similar to degradation, i.e. zeroing (disconnecting) the weight values, but not equivalent. The actual degradation does not appear, when the causes for such a degradation appear at later stages of learning evolution. Instead, the values of the weights, which have reached already a certain level but differ-

ent from zero, does not evolve any further. An example of such a degradation–like phenomenon without zeroing is shown in figure 3–15c. In the error landscape paradigm this corresponds to a stay in a flat area, different from those caused by the zero–weight.

In all the visualized cases there are two important features of the learning process to be observed: firstly, the weight values stabilize at a certain value, and secondly, most of them go into overlapping trajectories. Correspondingly, there is no distributedness in the internal representation. The lacking distributedness of the internal representations is equivalent to a decreasing network capacity: a network, able to map a complex problem, degrades to approximate only some feature of the training signal, such as the main tendency or periodicity. How to avoid these poor learning results will be discussed in the following chapter.

3.4 Knowledge, Randomness and Symmetry.

The perspective considered so far suggests that the learning process is as much dependent on the problem, (i.e. the knowledge which is being presented) as on the object (the network which is going to process it) and on the learning algorithm (which makes the incorporation of the problem into the network possible). The network implies the two design principles: Symmetry (due to the highly parallel architecture, initialization principles and used transfers), and Randomness (introduced by the randomized algorithm, parameter selection principles and additional noise injections). The network, incorporating Symmetry and Randomness, and the Knowledge about the problem interact during the learning process.

3.4.1 How things came to bear.

Knowledge comes in many disguises. It can be an implicit part of persons in an organization. By training people on the job, they can become knowledgeable about the tasks to be performed and on their function in the company or in any other socio–economic structure. But, when questioned, the persons involved may have problems in expressing their knowledge in a form that is accessible to others.

Classical artificial intelligence starts with knowledge acquisition: the process of getting knowledge out of people and into a model. This is a discipline in its own right. Questionnaires must be composed with considerable care to make sure that the gathered data can be simply transformed into explicit knowledge. Such knowledge is then represented in logical expressions to support logical reasoning.

This in turn brings the structure of new knowledge representation in terms of a neural network into play. Machine learning discusses this theme as an uncovering of *deep knowledge* in contrast to the *shallow knowledge* that results from *black–box modeling*. Where a system is described as a function or a relation on the input/output behaviour, it appears as a black–colored box of which no content is visible. The knowledge about the system is limited to its outward appearance. When more of the internal structure becomes known, the knowledge deepens.

As a view on the internal structure becomes available in neural learning by the captured knowledge on the hidden nodes of the network, it lends itself very well to the retrieval

of deep knowledge. Putting the three classical process parts (knowledge, acquisition and model) together gives a simplified view on a neural system as a feedback system. Only when separately the captured model is falsified does the feedback from the model to data set come in operation.

For neural networks these system elements are rather called data set, example presentation and network structure. This terminology shows that it is rather a structural view. As we aim to uncover the nature of the learning process, this will not fit nicely to our purpose. On the contrary, we need a terminology that reflects the behavioral aspects of the process.

3.4.1.1 Taking a different view.

In this thesis, we propose to distinguish between knowledge, symmetry and randomness. We have uncovered the symmetry problem as a major factor in monitoring the sampling strategy. Symmetry can be present in the data set and in the network and will require different decisions on sampling. Randomness may influence the effect of symmetry but will largely show its impact through the network structure. In other words, both are highly related but will bring different views on repair. It is the aim of this thesis to show that the *KRS–model* is closer to reality and will therefore lead to better ways to ensure learning reliability.

The available knowledge over the problem to be solved, instantiated in the input/target samples, gives the major guiding line of the learning process. The stochasticity, or the randomness, introduced in the neural system, and the symmetry are two other crucial factors, which influence the search for a successful and reliable itinerary. The symmetries in the landscape, signified by crests, valleys and flat–regions, presume equally probable itineraries. In other words symmetry gives an equal chance to move in several directions and therefore leads to indecisiveness. On the other hand, randomness is supposed to help the network escape from such a dilemma. It helps the learning algorithm to move away from the current stationary spot. In other words, symmetry and randomness define an antagonistic system of forces for anyone who travels the road of knowledge.

Symmetry can be dominant in the beginning of, but also at specific moments during, learning. Randomness (for instance as stochastic variable in the learning algorithm or as additional noise at the network input, output or parameters [3]) is then required to force the solution to follow alternative itineraries. When the amount of randomness is not sufficient to balance the effect of symmetry, learning will not be completed: instead of being adapted to ensure the right mapping between input/output data strings, the initial parameters will eventually become zero. If the noise (the randomness) of the system is dominant, learning will also be unsuccessful, because the network will rather learn the noise than the exemplified knowledge. The fundamental issue of learning is therefore the attaining and maintenance of a functional balance between symmetry and randomness directed by the examples (the knowledge).

To facilitate an optimal strategy in handling symmetry and randomness in relation to captured knowledge, it is important to know more about them. Though randomness and knowledge presentation are subject of constant interest [3], [67], [102], symmetry has not been elaborated in a systematic manner. Figure 3–16 suggests that symmetry can be categorized from two different perspectives, concerning either the place (the network or the problem) or the effect (the resulting relief of the error surface) of its appear-

ance. The error landscape displays essentially two meanings to the word “symmetry”: either the scenery itself or the choice in travel direction can be symmetrical. Such different meanings come about when studying the landscape as an interaction between the network structure and the problem. The error landscape results from network topology and value settings, while the travel direction stems from the structure of the problem as presented to the network during learning in request from the learning algorithm.

Having a look at neural networks as constructed in practice, it can be seen that randomness goes together with symmetry in a splendid variety: in Hopfield and Kohonen networks, randomly initialized neurons are connected in a symmetrical lattice; in feedforward networks, permutational symmetrical topologies are initialized with random parameters, uniformly distributed in a symmetrical range around the origin.

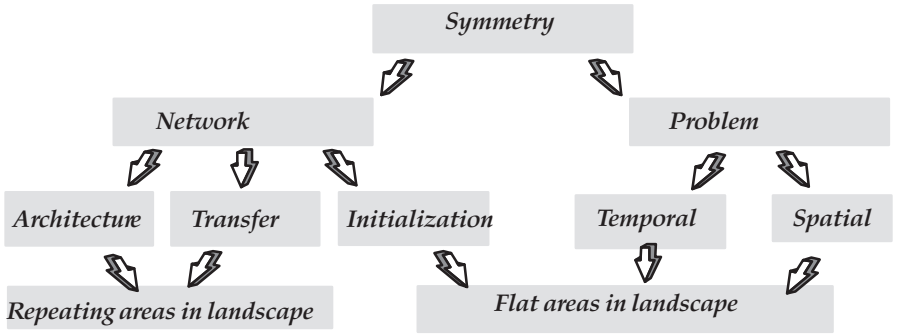


Figure 3–16: *The different faces of symmetry.*

Within the Knowledge/Randomness/Symmetry framework, finding the best learning trajectory is equivalent to ensuring the best ratio between these 3 components during the entire learning evolution (Figure 3–17). If the proper ratio of these three components is destroyed in the certain stage of learning, a way should be found to increase its presence.

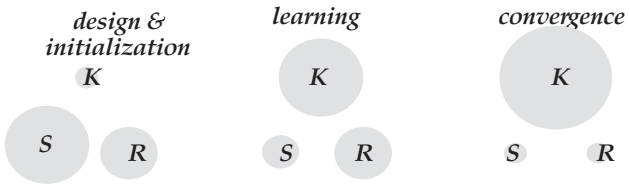


Figure 3–17: *Evolution of the KRS–system.*

3.4.1.2 Learning stages and reliability.

The introductory section of this chapter puts forward two major reasons for bad learning performance. The first one is related to the initial phase of training. As mentioned earlier, the network and the problem are completely separated in the beginning of the learning process. The network is constructed in the symmetrical fashion (it is “clearminded”) to be able to take a path in every direction. From the error landscape point of view the learning process starts in the area situated symmetrically between the different learning subspaces. During the initial training phase the guidance from the learning algorithm

and the training patterns are trying to break the symmetrical state of the network and to force the remaining search in one of the possible subspaces. The bearer of this initial symmetry is the highly parallel architecture and the choice of the initial state.

Once the initial symmetry is broken, the learning algorithm starts its gradient search in one subspace. As proved many times, a lot of difficult (mainly flat) areas can be met in this phase of the learning process. By overcoming these hurdles, the learning process can reach the final stage and tune its state to the satisfactory optimum (Figure 3–18).

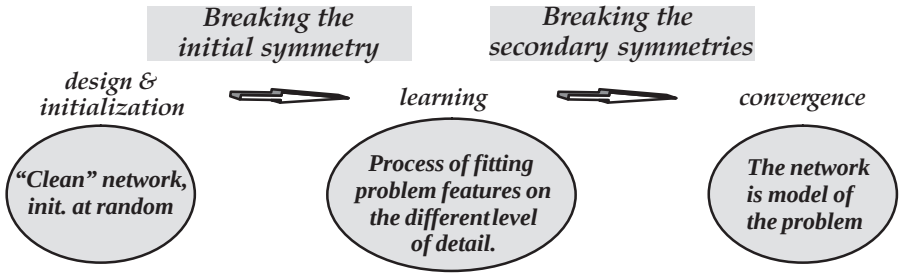


Figure 3–18: *The stages of neural learning.*

The error surface paradigm provides an integral view on the 3 phases of the neural learning process. These phases are all represented as surface parts, which should be traversed. In the initial phase the learning process starts in the area, equidistantly (symmetrically) situated between the subspaces, when all possible (equivalent) solutions can lay. The duration of the network's stay in this area depends on external factors such as noise (the random fluctuations), and on the problem samples, which tend to extract the neural system from the equilibrium state. This phase is known also as the *initial symmetrical phase*.

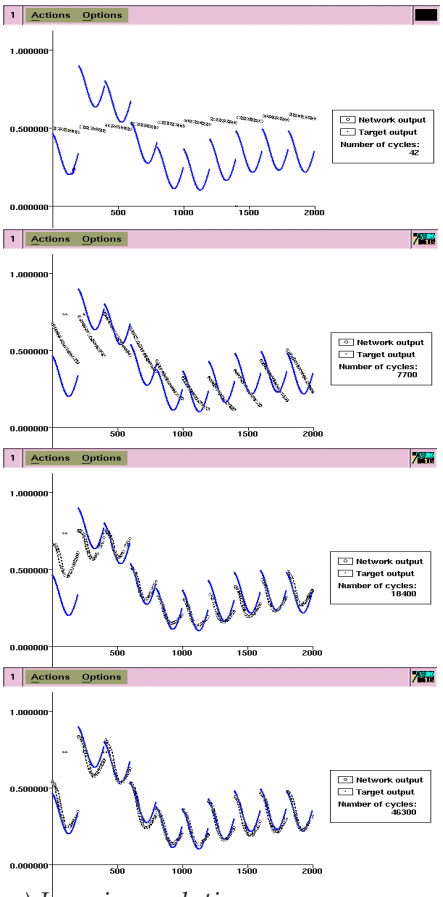
The equalities (the symmetries) in the network interfere with the problem to be learned and with the learning algorithm. If the nature of the problem to be learned is such that in combination of the network symmetries it disables the learning algorithm to make decisions for a long time (i.e. the signal has a cancelation nature), the network comes into a state, which is similar to degradation. In line with nomenclature from fault tolerance literature, *network degradation* is the phenomenon, when some of the connections of the network are practically disconnected – i.e. brought to zero.

In some simple cases, for instance in the investigation of a simple symmetrical function, this is really happening. For more complicated signals, as shown at figure 3–15c, the weight values are not nulled. Here the signal is a high–frequency sinewave super–imposed on a low–frequency wave. As the single sinewave can be satisfactorily learnt, one might expect the composition to be learnable too. This is unfortunately not true. Instead, the network has learned an element of the signal, its major tendency or its periodicity, and stops when a 'difficult' signal part is met. When analyzing this behavior, one finds that the remainder of the problem to be learnt has a distinct, local symmetry which precludes a succesful result.

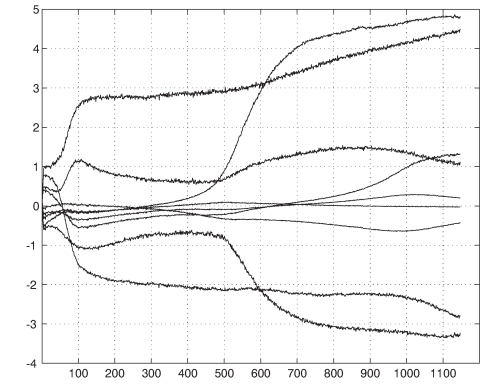
In the third stage of learning, the optimization is done in most of the parameter directions. The values of the active weights are in a quite wide range and symmetry is not a stopping factor for the tuning process anymore. But the large parameter values can

make the learning algorithm to take too large steps and may even bring unlearning with a growing error.

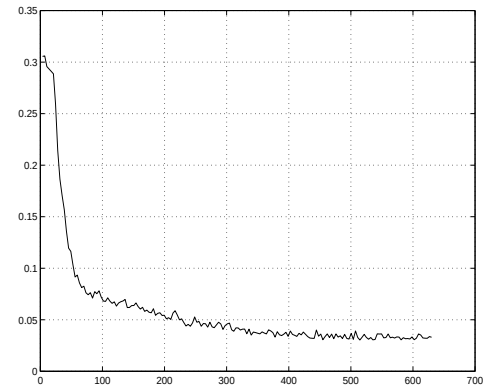
These three phases are illustrated in figure 3–19. It gives glimpss on the approximation of a complex signal at four time stamps: after 42, 7700, 18,400 and 46,300 training cycles, corresponding approximately to 0, 3,9, 23 epoches. In this succession of pictures, one can easily discern how the approximation process moves from global appearance to local detail. A closer look at the beginning of symmetry breaking is provided by the curves on weight and error evolution.



a) Learning evolution



b) Weight evolution



c) Error evolution

Figure 3–19: *Stages in a learning process.*

3.4.1.3 Towards a KRS measure.

In case of an unsuccessful experiment a common practice is to reconstruct the experiment and restart training process. A range of alterations can be devised ranging from changing the network to changing the data set. The wide variety in options makes it very

hard to find the optimal variation. Often the presence of potential problems goes unnoticed.

In order to get the problem into grip, a balance in the KRS–field should be kept. Changing on–line the learning speed has been used with occasional success, but as we pointed out this only comes to avail when the speed of error backpropagation is at stake. In the KRS–view this is not a remedy but only soothes the pain. A more structured method can be based on finding an optimal extraction of the knowledge. It is for this reason that we rather discuss learning behavior as a knowledge transformation process governed by the two antagonistic influences: symmetry and randomness.

The critical issue is therefore: how can the antagonism be quantified? This is not an easy thing to decide. Firstly, we need to show that the KRS–model can explain a number of known anomalies in a generalized setting. Secondly, we need to come with a practical way to fight against this anomalies. Once this has been done, there is ground to pursue a more in–depth, analytical formulation. In this thesis, we limit ourselves to the first–hand solution. It will be found in a next chapter that the behavior of a non–learning process is so much different from a learning one, that a simple qualifier will suffice to improve drastically the network performance. This in turn validates that our model hypothesis might be correct enough to pave the ground for further research.

3.4.2 The KRS model.

So far our critique on the current methods of symmetry breaking is that they are either problem–specific, or concern one phase (usually the beginning) of the learning process. Alternatively we propose to solve the global learning reliability problems by keeping the proper KRS–ratio during the entire learning period. In a sense this idea is borrowed from *active learning* [121], which presumes that the learning algorithm has control over what part of the problem domain it receives information about. Our suggestion is to base this control on the instantaneous check of the training set structure and to reorder it if necessary. This way, not only the initial high symmetry will be destroyed, but also the long plateaus of the training error can be shortened and shallow valleys escaped. As a result, reliable convergence and learning duration can be ensured in difficult to learn tasks (for instance when cancelation signals are trained) or the training time decreased, when ordinary signals are learned. The other advantage of this proposal is that the problem is not violated by adding extra noise to it.

The occurrence and impact of neural network symmetries as seen on the global error surface have been reviewed and analyzed from the learning evolution point of view. In summary, the symmetries in network architecture and transfer are forming repeating parts in the error landscape and are of importance only in the beginning of the learning process, i.e. before the specialization of the neurons occurs. These symmetries are predictable and common for most networks. Normally they can not harm the learning process. On the other hand, when the problem symmetries are present, together with high initialization symmetries, the structural equalities cannot be destroyed and thus the specialization of the representation neurons (learning) is likely to fail or to suffer from a too long and unpredictable learning duration.

All the reviewed literature concerns either artificial problems (attempts to learn geometrical or algebraic functions, that contain a lot of internal symmetries), or networks

of importance rather for theory than for practice (parity machines, committee machines) which introduce artificial symmetries as well. In [10] it is shown how symmetries can affect real–life problems.

Moreover the suggested strategies for breaking the symmetry have the following weak points: (a) they often change the complexity of the problem to be learned by introducing additional noise – a remedy that can harm the learning quality near the convergence point; (b) the solution they suggest is in most cases problem–dependent; (c) often they concern only one moment in which the repair should take place.

3.4.2.1 Looking into the mirror.

Symmetry is a fundamental property, where the conciseness of a mathematical model is at stake. Symmetry can be found in almost any natural process and subsequently used to model part of the process and generalize for the total. Analytical algebra gives a variety of examples of how symmetry can be used by a range of laws, such as on commutation, reflection, transition and association.

We have seen however that symmetry can also have negative effects. By the occurrence of a "longest run" phenomenon, a network can be brought to confusion rather than to knowledge. Evidently it makes no sense to change the problem to be trained. The issue is rather that the presentation order of the data set should allow for breaking the symmetry during learning.

We have also discussed the role of randomness. It means to break the symmetry but should keep the implied knowledge intact. This leads to the idea that symmetry and randomness are to be in balance to allow for the transmission of knowledge between data set and network. From [17] we quote

> ***"In our test we observed three phases with smooth boundaries. At the beginning of the adaptation (a fully open system) there is not enough knowledge to make exploration decisions, so random selection is as good as reflective exploration. In the next phase the accuracy can be significantly improved by making good exploration decisions. In the third phase, so much is known about the system, that it becomes closed, and new examples don't have much influence on accuracy. In this phase exploration becomes less effective. The boundaries between such phases are not obvious. We summarize conditions when exploration would not be advisable: (1) when the agent has learned enough about the system so that it is no longer an effective open system; (2) when the overall learning set is going to be completely learned anyway."***

We see the same theme recurring in a number of publications. For instance, Heisterman looked at the use of genetic algorithms for optimization problems [75]. After an initial phase, where Monte Carlo analysis might be used to create a sensible set of genes, he uses only a limited set of genetic operators to close in to the optimal solution. However, when the optimum is nearly found, he changes over to straight gradient search.

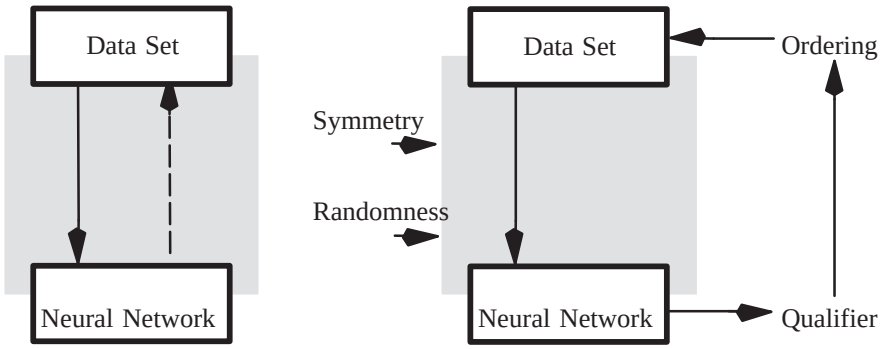
We interprete this as a steady change in balance between symmetry (making for a number of local extrema) and randomness (allowing to ignore the local attraction). One should mark, however, that symmetry and randomness are not independent parameters. We have seen that symmetry can be caused by the sampling of a signal. By behavioral symmetry, the same effect is due to the finite precision arithmetic in the network itself.

The influence of the resulting quantization error is commonly called noise and therefore a contribution to randomness.

The importance of the above cited work is that it shows how simple criteria can be found to indicate a phase transition. To accomplish this, we need to find a qualifier that gives a true indication of the present overall ratio between symmetry and randomness in the presence of knowledge. We will call this the KRS–ratio.

3.4.2.2 The role of the qualifier.

To depict the role of the qualifier, we will first take a look at the basics of the learning process, as shown in figure 3–20a. The content of the data set is sampled and brought to the attention of the neural network over some learning algorithm. This learning is by large a uni–directional process. Learning problems often go unnoticed and only under catastrophic circumstances will the need arise to make changes in the data set (re–ordering; augmentation; enrichment). This will happen off–line; the data set is intuitively improved untill the learning seems to succeed.



(a) Classical Learning (b) KRS–based Learning

Figure 3–20: *A look at the learning process.*

Our research aims to uncover the nature of learning problems and to acknowledge the uncovered parameters: symmetry and randomness. The situation is bi–facial. An asymmetrical signal on an asymmetrical networks seems learnable. However, when either the signal or the network is symmetrical, problems may occur. While when symmetry is present in both data set and neural system, accidents are bound to happen. For this reason, we see symmetry (and likewise randomness) as attributes of the overall learning (Figure 3–20b).

Because of their omni–presence, the influence of symmetry and randomness will be hard to quantify directly. If ever, information theory will be the carrier rather than pure stochastics. But sofar this theory has not been applied successfully to the learning process [48], we will rather attempt to derive an empirical rule. This rule aims to indicate learning problems as soon as possible. Compared to the classical approach it is viewed that many short duration fail attempts are to be prefered to the single long awaited disaster.

In other words, we aim to find an easy qualifier that measures indirectly the effect of symmetry and randomness on learning reliability, irrespective of their nature. Such a

qualifier should be fit to steer decisions on how to change the presentation of the data set. Rather than to allow for an arbitrary selection from the data set, we propose here a willful ordering of the data set: the presentation set. We will not attempt to change the learning algorithm.

This then poses the next problem: in which way a data set should be re–ordered to achieve a better learning reliability. In general, this is a combinatorial optimization problem that can not be solved on–line. As in this thesis the focus is on the KRS relation and the ordering problem is only of secondary importance, we will discuss a greedy short–cut. In the first instance, we will assume symmetry to have either a local or a global impact. By judiciously enlarging and shrinking a windowed view on the data set the effect of symmetry in the current learning process can be eliminated.

4 Example selection.

The previous chapter puts forward, that the major obstacles for a reliable itinerary on the error surface (and for the learning reliability, respectively) stem from the peculiar error relief forms such as plateaus or valleys between the solution subspaces, creating conditions in which adequate learning is difficult to achieve even if it is theoretically feasible. The dynamics of the learning process may include situations where further progress is halted on such relief forms. The key to learning success is founded on the order, wherein the examples are presented. The most common way of example ordering is to select them at random from the set of available examples. We argue that this is not sufficient and introduce alternative sampling strategies. Particularly we propose a notion to allow for automation in this problem–driven example selection: the cancelation criterion. This criterion will be formulated and verified to show its potential for enhancing the learning reliability.

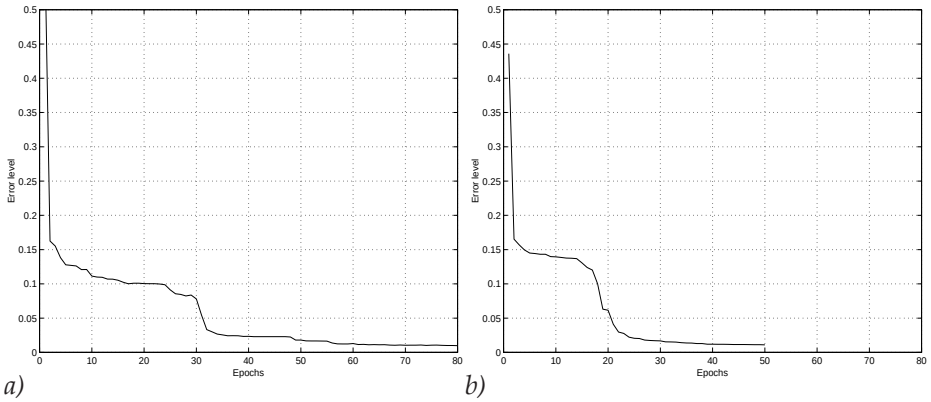
The nature of iterative learning on a randomly initialized and (in terms of the selection of training examples) arbitrarily developed network is such that a precise replication of the experiment is practically impossible. Every experiment is dependent on two factors outside the actual problem definition: (a) the choice of the initial parameters, that position the network at a certain point on the error surface, and (b) the order of presented examples that direct the itinerary on this surface.

It is widely known that an unsuitable initialization of the network can be a reason for its bad performance [107]. We claim, that the occasional failure of the learning algorithm is rooted in both above named factors: not only wrong initialization, which makes the learning algorithm to start from difficult to escape areas on the error surface, but also wrong guidance of the example flow in a way that precludes the escape from difficult error surface's relief forms. In an attempt to generalize, it can be observed that the second case implies the first one: if proper guidance of the learning algorithm can find a way through a difficult landscape during learning, it is certainly possible to surmount a difficult initial position.

This claim can be supported with a very simple experiment scheme. Two groups of experiments are performed with the same network until the cost function for every single run drops below 0.01. Both experiments start in the same initial point with the same training set, ordered according to a different rule. The outcome of the two groups of runs show a difference in training time and in learning trajectory. Typical trajectories from both experiments with average performance (regarding to training duration) are visualized by error function development in figure 4–1.

This experiment shows, that the learning process depends more on the way training examples are presented than on its starting position. This implies, that a successful and reliable convergence of the learning process can be achieved by only controling the *presentation order* of the examples during learning. The choice of a starting position can be made according to widely accepted initialization rules.

So far we have put forward that the effort to construct a reliable learning trajectory can be spent entirely on directing the itinerary of the learning process. In contrast, ensemble methods (in their simplest form) imply starting from different points in the weight space and choosing the network (the solution) which takes the best trajectory. Such an approach does not answer to the question how to proceed if there is not a good solution found or if the initial metastable state is not escapable for some reasons (see for illustration the analysis of a symmetrical function in a shallow valley in section 3.3.2.1).



a) *b)*

Figure 4–1: *The error function for experiments with examples of the same training set ordered in different manners. Both experiments are started from the same initial point in the multi–dimensional weight vector space.*

We have characterized the choice of initial parameters and of the example presentation order as dynamical aspects of neural learning. Within the error surface paradigm the process dynamics relate to the route, that the learning algorithm takes on this surface on its way to the optimal solution, while the surface shape is static. Since the route is determined to a large extent by the example stream, normally constructed according to some randomization principle, the impact of randomness on neural learning will be discussed in detail with respect to both its weak and strong points. Though random sampling seems advantageous in helping the network to escape a state of symmetry, it can not ensure that the same learning trajectory will be repeated twice. In the first instance, this means, that the learning experiment can not be replicated. Consequently, this is a cause for bad reliability: the learning duration can not be guaranteed.

Therefore alternatives to the random sampling scheme will be discussed here. In search for dependencies between the learning quality and the example stream presented, some experiments are performed with alternative ways of selecting training examples. It illustrates the influence of different pattern presentations on the learning reliability, but foremost suggests a framework for solving a number of learning problems, i.e. ensuring the reliability in terms of higher probability for convergence.

Though a closed–form analytical expression to predict network behavior as a function of example presentation order has not been found, frameworking the interdependence between the example presentation and the weight changes and considering the peculiarities of the error landscape leads to a practically acceptable criterion: the *KRS–ratio*. It is found out, that if the KRS–ratio converges to zero within one epoch, there is a high

chance for cancelation to appear. The suggested example reorderings are increasing the KRS–ratio development and correspondingly resolve cancelation situations. In accordance with the created cancelation criterion, a sample selection strategy is developed and its feasibility is shown.

4.1 The basics of sampling.

The creation of a training set that ensures optimal functionality from the learning process has many aspects. As the continuum to be modeled is infinite, taking representative examples involves a sampling operation: a finite set of examples is extracted according to a *sampling algorithm*, such that the set (a) can be learned and (b) allows for generalization. The properly trained neural network has then become an executable model of the infinite continuum.

Characteristic of a sampling algorithm are the sampling rate and the sampled interval. The relation between these two has been worded for information processing by *Nyquist* in his *Sampling Theorem*. It states that the information as present in the infinite continuum can be contained in the finite example set. However, the Nyquist Theorem is necessary but not sufficient for neural networks. It derives its significance from the fact, that it gives the minimum amount of examples that must be present in the example set. It is generally possible to reduce the size of the training set, as reported in [69] [146]. But in some applications obtaining the training samples is costly. Important in these cases is to find the smallest number of examples that lead the network to a good result [80].

The next question is therefore, whether all examples in the set must be used? It is clear that according to the Nyquist theorem again a minimum number of the examples must be used. The related question is therefore: if the example set is redundant, which examples must be chosen? This question has far reaching consequences, as it implicitly allows for a discreticized but potentially infinite input space. For the moment, we rather restrict ourselves in the number of ways to optimize learning.

Therefore we assume that an example set is available and will be fully used for learning. So next to the above physical concept of sampling data points from a signal, we will also view sampling as a data selection technique. From the pool of examples, we take all examples in a deliberate order and present them individually to the network for the purpose of learning. In other words, by sampling the data set is transformed into a *presentation set*, the difference being that in the latter the examples are already queued for presentation.

4.1.1 Sampling techniques.

The presentation order of the examples is of greatest importance for the success of learning. For a long time, a random selection of examples has been prefered as it was noted that by repeatedly presenting the examples in a fixed order forces the network to remember this order too. Often, this order has no real meaning and leads therefore to false learning. By presenting the examples in a random order, the network will never see the same sequence and therefore learns from the individual examples only.

In the following, we will start from this *random sampling* (RS) technique as a point of reference. Subsequently we introduce some variations to find that next to its obvious

advantages it also has some drawbacks that make for a reliability problem. It is essentially this reliability problem that we aim to solve in this thesis. Therefore a brief review of existing neural sampling schemes is made first.

4.1.1.1 Random sampling.

Most neural algorithms apply a number of random choices during their operation. Hence, even for a fixed input, different runs may give different results in time and accuracy. This nicely fits the definition of a randomized algorithm, according to [92]:

Def. 4−1: A *randomized algorithm* is one that receives in addition to its input data a stream of random bits that it can use for the purpose of making random choices.

There are two major advantages of randomized algorithms. Firstly, often the execution time or space requirement of a randomized algorithm is smaller than that of the best deterministic algorithm known for the same problem. Secondly, all the existing randomized algorithms are extremely easy to understand and implement [92]. Randomization of a known deterministic algorithm with a detrimental worst–case behavior converts it often to an algorithm that performs well with high probability on every possible input.

Randomized algorithms gather information about the distribution of their input data by drawing random samples. In the case where the input data are fixed it is useful to randomize the order at which they are presented. In this sense, the random transformation of the *example set* from data set to presentation set has been performed by a randomized algorithm.

In general, there are no tangible rules in neural theory relating the signal to be learned and the nature of training patterns. Thus random equidistant sampling gives in most cases a satisfactory result. Moreover, the random factor is crucial for the work of all learning algorithms of a stochastic nature. The founders of the backpropagation algorithm have suggested in [129] that not only the network parameters but also the training examples should be chosen randomly. The motivation for this choice is based on the features of the most commonly used optimization method – the *gradient descent*. Gradient descent is performed on the local error function:

$$f(\boldsymbol{w}, z^{\mu}) = -\nabla_{w} e(\boldsymbol{w}, z^{\mu}) \tag{4–1}$$

The stochastic gradient descent on a local scale approximates the deterministic gradient descent on a global error landscape:

$$E(\boldsymbol{w}) = \lim_{T\to\infty} \frac{1}{T} \sum_{\mu=0}^{T-1} e(\boldsymbol{w}, z^{\mu}) \tag{4–2}$$

Another reason to choose the training samples randomly is to reproduce the density of the underlying distribution. If there is no further information available, this choice is very reasonable, but it usually results in a suboptimal training performance. Beside its easy implementation, random example presentation supports a framework for theoretical investigation of the generalization properties of neural networks. It is generally refered to as “Learning from examples” (Section 1.2.1). Analytically, learning from examples method is examined in many studies, among which are [13], [41] and [46]. A complete description of this approach for analyzing neural networks’ generalization is given by Poggio *et al.* [123].

Random sampling assumes that training examples are arbitrarily chosen, and that the learning process evolves under its own dynamics. In this case, it can be said that the neural network is a passive learner: the neural network has no control over the construction of the presentation set.

4.1.1.2 Alternative sampling schemes.

The intuitive alternative to randomization is that of selecting the training examples to follow the time order of the extracted signal samples. This happens for instance when the natural order of presentation is of importance (as in time–series prediction tasks) or when this is the only possible way of obtaining the examples (like in on–line learning scenarios).

Both choosing the examples for training at random from the available samples as well as presenting them to the network in their natural order are passive learning schemes: training algorithms do not have control over the example selection and presentation process. The most broadly applied neural algorithms are passive. The Hopfield Network, for example, takes its pattern set as a whole and so does also the Back–Propagation algorithm in its batch version. In the stochastic backpropagation, examples to be learned are randomly selected [46], which is also a passive learning technique.

A lot of studies show superior results when training examples are chosen in some systematic way. Generally, the decision on which examples are selected, is made by the training algorithm. This is the reason to refer to this training scheme as active learning. Active algorithms applied to neural networks aim to ensure success of the neural learning process by optimizing the information coming from the environment. They are either oriented towards the strategy of pattern presentation or to the selection of the best training set. Accordingly, there are two distinct classes of techniques for choosing training examples. The first group assumes that the network is partially trained on a set of previously acquired examples. This class of techniques is known as *active sampling* or *progressive learning* [46] and can be defined as the task of adding new examples to the set of available examples. The second class of active learning techniques is known as *active selection* or *informative learning* [80] and implies selection of training examples from the set of available examples.

In the active learning scheme the necessary number of training samples heavily depends on their distribution over the input set. Properly selected, these actions can drastically reduce the amount of fatalities and the computation time required for learning to be completed. As reported by Morgan and Boulard [106], one can produce results by using only a small fraction of the available examples, that are close to those obtained when all available data are used. Another reason for using a specific strategy of pattern selection is that network performance can be drastically improved (Cloete and Ludik [39]). This latter result, together with [7], [121] etc., establishes neural active learning as an alternative for passive learning from random examples.

In summary, it can be said that the sampling of a signal to be learned as well as the selection of the examples for actual training are important characteristics of the training process. Different manners of sampling or example presentation require changes in the training algorithm, but not in the optimization principles. Therefore, active and passive learning are terms concerning not the training process in general, but the way of example selection and its benefits to the learning success. Different sampling and ordering methods are relevant to one of the sampling schemes, as summarized in figure 4–2.

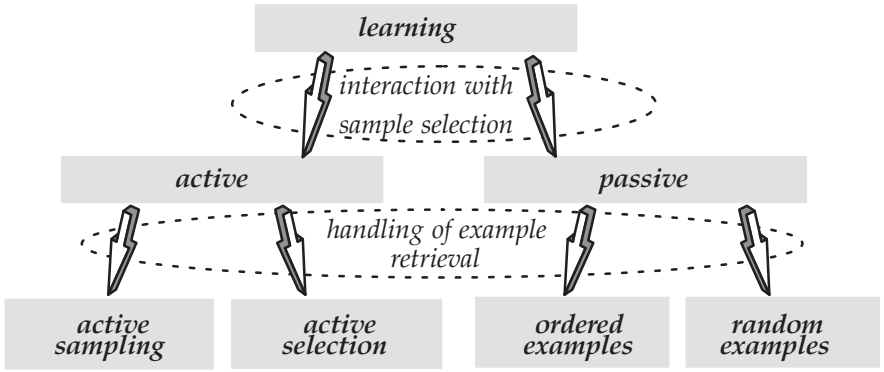


Figure 4–2: *Classification of the learning process with respect to example presentation.*

4.1.2 Neural active learning.

Active learning has been studied independently by researchers in many different areas such as neural networks, robotics, computational learning theory, experiment design, information retrieval, and reinforcement learning. It addresses the effects of pattern selection and presentation on the learning process. The active learning algorithms applied to neural networks aim to ensure success of the neural learning algorithm by selecting the optimal information from the environment. They are either oriented to the strategy of pattern presentation or to the selection of the best training set.

4.1.2.1 Active selection (Informative learning).

A logical way of actively selecting the training examples is: to choose from a large and redundant set of samples those, that will allow one to achieve the ultimate goal of learning the problem at hand. The reduction in the number of training samples can be necessary for several reasons. Firstly, reducing the training set size will eventually speed up learning. Lets assume that the number of selected examples is much smaller than the number of overall available samples. Thereby learning can be expected to have a shorter duration, because it is accepted that the training time is proportional to the number of examples to be learned. Secondly, obtaining the training samples is costly in some applications. Then finding the smallest number of examples that lead the network to a good result, is of importance [80].

The training set reduction can be achieved either by an incrementally growing training set [121], by selecting samples or queries [14] or by skipping examples in the training set [2] [69] [146].

If the training set is not carefully chosen, learning can result in bad generalization. Then it is important to select a concise but efficient (in terms of generalization) training set. Learning schemes, that make it possible to select training examples such that the redundancy contained in them is avoided and generalization abilities are not worsened, are called *query learning*. It contains two non–separable tasks: data modeling and data selection.

The *data modeling* process starts with choosing a small data set from the problem to be learned and consists of fitting this data into a neural network model by an optimiza-

tion method [126]. The model should be accurate, but not over–fitted. One way to fulfill this restriction is to base the model selection process on cross–validation techniques [19] [89] [93]. The so–obtained model paves the ground for the *data selection* process, where an optimization criterion is defined and computed. This criterion can for instance correspond to the model's estimate of the squared distance between its actual output and the expected correct output for a new data point. The data that minimizes this criterion is then added to the training set.

After the data model has been selected, there are three different approaches to find new data. The first approach is applicable when a neural network model has been established using already seen input/output examples and the exploration of the data space will include an incremental expansion of the training set. If the input value of a new example is unknown, it is necessary to calculate the input value to be queried. Usually, this new value is calculated stochastically. This first approach is called *Query Construction* [141], but, if there exists a string of random input values that can be queried, it is normally referred to as *Query Filtering* [121]. The process of adding more examples to the training set is often called *data subset selection* [30], [163].

In every training set there are "difficult to learn" training examples – the network learns them least and in approximate borders. Some work in neural active selection aims to improve the learning and generalization capabilities of the network by modifying the presentation frequency of patterns [30] [108].

In [46], a focus on particular examples is envisaged, which constitutes active learning, to escape from situations where the network is stuck in a local minimum. For all the studies mentioned above, numerous criteria exist to select the pattern to be learned, but, in all cases, the general principle is to choose the example that will give the maximum of new information about the environment. The way of implementing the method of changing the frequency of pattern presentation is the influence of each example of the training set to be weighted [30] [108].

4.1.2.2 Active sampling (Progressive learning).

Learning can be considered as an intrinsically temporal process: the environment is not learned as a whole and at once, but in several stages. As seen in figure 3–19, the network approximates the median of the samples at the very beginning of the learning process. Later the network learns the global tendency of the target function (Figure 3–19b). In the following steps the network approximates small details from the target. Finally, the complete signal is learned by the network (Figure 3–19d).

This subdivision of the learning process can be considered as an advantage, since it gives the possibility of learning progressively. Progressive learning (or also known as active sampling) assumes, that the training set is not learned at once. The final goal to learn the posed problem is achieved by fulfillment of intermediate tasks: the *partial learning*. There are not many studies on that aspect of progressive learning in neural networks, although it was evoked some years ago [46].

Active sampling is essentially concerned with determining the distribution of training examples, which may differ from the distribution according to which examples occur naturally in the learning environment. The modified distribution is determined by making use of a priori knowledge about at least one of three objects: (a) the computational model i.e. the neural network architecture, (b) the learning algorithm as largely determined by the error criterion used for training and the learning rule by which this error

is minimized, and (c) the learning task which includes the environmental distribution and any a priori knowledge of the target function. In some cases, there is enough knowledge about the learning task and the network model, and thereby it is possible to determine the optimal sampling distribution in advance, without need for active sampling [5] [43]. Active sampling determines the sampling distribution according to the result of learning upon previous examples. In this sense the network is an active participant in its training.

The most well–known study on progressive learning is done by Elman [55]. He solves a grammatical prediction task. Sentences, generated by an artificial grammar, are presented to a simple recurrent network, which must predict at each time the probability of occurrence of every word in a dictionary. When the corpus of sentences is presented as a whole to the network, training fails. Conversely, when the training set is divided into two subsets (one containing easy sentences and the other containing complex sentences) and when the proportion of complex sentences increases progressively in the same training cycle, then the training succeeds.

The improvements achieved by progressive learning can be intuitively interpreted from the shape of the error surface. The error surface is entirely determined by the training set and the network topology and transfer. As discussed in the previous chapter, the surface is the same during the whole learning period in the classical passive scheme of back–propagation learning. In active learning, the error surface changes continuously, since the training set changes. Consequently, when beginning with an easy subset, the gradient dynamic applied in a first error surface easily and rapidly leads close to a minimum of the surface, which places the network's weights in a good configuration to learn the second subset, and this is repeated at each stage, until the end of the learning. This explanation is related to studies about initialization of the weights in back–propagation networks [83].

Such studies on progressive learning are very interesting, but there are some reasons to suppose that their validity is limited. Firstly, they suppose that the decomposition of the example space in successive training sets is known a priori. Secondly, the training set has to be changed. In [55] the next training set is selected every five epoches. In other experiments, there is a threshold on the error below which the system takes the following subset. This threshold is diminished at each subset, either by taking a priori defined values [39] or according to some law [39] [45]. In all cases, a priori information is given, that has a strong influence on the results. Thirdly, most experiments are made on recurrent networks [54] , where the design of consecutive training sets is based in the analysis of the temporal sequences; see [132] for some rather disappointing experience.

The historically first approach in progressive learning is the *combined subset training* [30] [45]: the network is trained on a subset of random patterns selected from the whole training set until a certain criterion is reached. Then a second subset is added to the first one, and so on until the whole training set is learned. Cloete and Ludik [40] suggest also an increased subset training, which implies a multiplication of the training set size by a fixed number. Another limitation of this approach arises from the problem to predict the difficulty of a task by the number of examples to be learned. Thus, the first subset can be quite hard to learn, almost as hard as the whole training set.

The natural development of the active sampling method is to build *consecutive training* sets by increasing not the size of the training set, but its difficulty. This is achieved in

[39], where the network is first trained to count from 0 to 1, then to 3, then to 7 and lastly to 15. The obtained results are significantly better than with a combined subset training. Similar experiments, but with different motivation, are reported in [55]. The author shows in a task of complex sentence prediction, that starting with a training set containing short sentences and increasing the length of the sentences allows the network to learn the whole environment, whereas it is unable to learn it otherwise. In the third approach, partial training as suggested by Jacobs [83], several intermediate tasks are presented to the network (before it learns the final task).

An interesting experiment is described by Fahlman [56], where the recurrent cascade correlation architecture is used to learn the Morse code. Learning is achieved by presenting the patterns from the easiest characters together with the most difficult ones. Implicitly an important part of pre–processing has been performed here, as the Morse code already takes the importance of characters into account. The code provides a sequence of dots and strokes for each character, such that the most frequent characters are assigned the shortest sequence. This makes it clear beforehand which characters are easy and which are difficult.

4.1.2.3 Active learning implementation principles.

A widely accepted and practical way to implement active learning principles is in the framework of so–called *optimal experiment design* [6], as popularized in statistics. This framework is based on a technique of Maximum Likelihood Estimation. The work in this direction has been carried out in [42] [44] [99]. An integrated prediction error has been used as the objective function for selecting a new data point. This error function assumes that the data points have asymptotically a normal distribution [121], of which the parameter values must still be found. It is possible to use a Markov Chain Monte Carlo simulation [115] to establish this measure.

Another framework to be considered is the *sequential optimal recovery* [13]. This type of analysis does not make probabilistic assumptions, but concentrates on the worst–case behavior of the algorithm. It has been subjected to a detailed study in the area of function approximation, in which one tends to compute the maximum possible error for the guidance of selecting the next data point. All the methods mentioned so far can be described with the Bayesian inference paradigm [142].

As described in section 2.3.1.1, an advanced approach for neural network training to overcome the dependence of the learning process from the choice of initial state is the committee method. This method copes very well with the Bayesian paradigm. When a committee of neural network models is obtained as a result of the data modelling process, there is an expected disagreement on where to sample for new data in the problem domain. This disagreement can be expressed as an *ambiguity measure* [93], or by the estimated output variances [89] [124].

The choice of committee model differs with each application area. Bootstrapping techniques have been used by [53] for the time–series prediction task. It was found that a better generalization performance and a more accurate estimation of output error bars is achieved [27] [149]. This confirms our expectation that pre–structuring of the presentation data based on a windowing technique is worthwhile. This idea will be further detailed later in this chapter. But first we need some more background information.

4.2 Alternative example selection schemes.

The alternative sample selection scheme, as will be motivated and developed here, aims to increase the reliability of neural learning. For this purpose it should correspond to the concept of reliability, namely achieving robustness and effectiveness through finding an optimal training trajectory. For making the learning itinerary reliable and always reaching a satisfactory solution, it should be ensured that even difficult error landscape areas will be passed. As elaborated on in previous chapters, such are very flat and very curvacious regions. Normally, a randomized training algorithm can escape flat areas and shallow minima or valleys; for a specific group of signals, which will be named *cancelation signals* here, such areas are usually an unsurpassable obstacle. The experiment in section 3.3.3 analyses one simple example of this signal group; others will follow in the remainder of this thesis.

There is not a learning method that can solve all potential problems. We attempt to get over unreliable convergence by a proper example selection scheme. Here it will be shown how selecting the training examples in a different manner can change the learnability of cancelation signals. Additionally, as was shown by the opening experiment of this chapter, the learning speed of an arbitrary signal may be increased by orders of magnitude through a proper example reordering.

Since the order of presenting the training examples is crucial for improving learning reliability and speed, the impact of example streams on the learning process has to be revealed first. After defining the overall characteristics of a cancelation signal further in this section, the cancelation training set will be introduced for the training itself. It predicts the learnability of a particular training sequence. The main point here is, that even if the signal has a cancelation nature, presenting the extracted training samples in a special order can avoid a number of learning problems. In order to define the way of selecting the training examples a simple criterion, called *KRS–ratio*, is developed. Its simplicity makes it easy to calculate for the current training sequence and thus to be implemented on–line in the training algorithm.

Calculating the KRS–ratio predicts which signals will be difficult to train. The learning difficulties are investigated by systemizing the results of many experiments with signals with different level of cancelation. Manners for increasing the KRS factor are easily derived from the features of the optimization method and random processes. The results of training large numbers of sets, produced by reordering the extracted samples in a special manner, are confirming the usefulness of the chosen resampling strategies.

4.2.1 Example presentation order and learning success.

The analysis of the data–driven nature of the learning process requires a corresponding framework. Here only general directions will be given to provide a basis for the sampling method which will be proposed in a further section. Moreover, the general analysis and solution for above described learning problems will be made within this framework.

Following the definition, given in 1.2.1, the learning algorithm's task is encoded in the set of examples z^{μ}. Finding the function $f(x) \in \mathcal{F}$ which maps the problem best is equivalent to discovering the conditions in which differences between the responses,

given by the network and by the supervisor, to the same input vector are smallest. This difference for the pattern $z^\mu = (x^\mu, y^\mu)$ is denoted as error $e(z^\mu, w)$ in equation (2–1). The algorithm processes the sequence of examples z^μ until the network error (2–3) falls below some given magnitude.

If z^μ is a randomly drawn example from the training set $\{z^1, z^2, \ldots, z^N\}$, the total error function can be formulated as:

$$E \equiv \frac{1}{N}\sum_{\mu=1}^{N} p(z^\mu).[f(x^\mu, \boldsymbol{w}) - y^\mu]^2 \qquad (4\text{–}3)$$

where $p(z^\mu)$ is the probability that an example z^μ is presented, y^μ is the target part of this example, and $f(x^\mu, w)$ is the network response. The widely used random pattern presentation method requires that in one epoch all training examples are presented. Correspondingly, every example is equally likely to be presented to the network.

4.2.1.1 Impact on the learning success.

This implies that the optimization can be performed on the development of the weight values in time. The probability that example z^μ is presented can be expressed by the probability that it is the example with number t, taken for training the neural system. This correspondence indicates that the conclusions on the effect of consequence of pattern presentation can be derived by calculating the probability characteristics of the training set.

Learning reliability is to be judged on the trajectory which the learning process development makes on the error surface. As it was shown by equation (3–8), the changes of the error value with respect to the network parameters (weights) is equivalent by value and opposite by sign to the changes of the weights in time. This derivation is very helpful, since the change of the weights in time is related to the presentation order of the training examples. In order to give more insight to that statement, let us remember that the weight vector values at moment t+1 depend on those at moment t and on the weight change, caused by the example with sequence number t:.

$$\boldsymbol{w}_{kj}(t+1) = \boldsymbol{w}_{kj}(t) + \Delta_{z(t)}\boldsymbol{w}_{kj} \qquad (4\text{–}4)$$

where $\Delta_{z(t)}\boldsymbol{w}_{kj}$ is the weight change when the t^{th} randomly selected example z(t) is supplied to the network. For the on–line mode the weight change after the presentation of the t^{th} random pattern is:

$$\Delta_{z(t)}\boldsymbol{w}_{kj} = -\eta\nabla e(z(t), \boldsymbol{w}) = \eta\delta_k(t)o_j(t) \qquad (4\text{–}5)$$

The generalized delta rule for updating the weights $w_{kj}(t)$ includes also a momentum term, contributing the influence of the previous weight changes to the current one, i.e. of the change the previous examples caused to the weight values:

$$\Delta_{z(t)}w_{kj}(t) = \alpha\Delta_{z(t-1)}w_{kj}(t-1) + \eta\delta_k(t)o_j(t) \qquad (4\text{–}6)$$

In order to see the effect of the sequence of patterns on the synaptic weights it is necessary to sum the changes which every pattern causes on the weight values:

$$\Delta_{z(t)}w_{kj}(t) = \Delta_{z(t-1)}w_{kj}(t-1) + \ldots + \Delta_{z(1)}w_{kj}(1). \qquad (4\text{–}7)$$

From equations (4–6) and (4–7) the current change of the weight value, caused by the current example $z(t)$, can be represented as a time series.

$$\Delta_{z(t)} w_{kj}(t) \;=\; \eta \sum_{n=0}^{t} \alpha^{t-n} \delta_k(n) o_j(n). \qquad (4\text{–}8)$$

This form of the generalized delta rule has been used for analyzing the behavior of a symmetrical function in the initial shallow valley (equation (3–27)). The notation $\Delta_{z(t)} \boldsymbol{w}_{kj}(t)$ is made here to show explicitly the dependence of the weight values on the training examples. In general such a notation makes the description less distinct, and will be avoided in further sections by substituting with $\Delta \boldsymbol{w}_{kj}(t)$. Besides, $\delta_k(n)$ in equation (4–8) can be substituted, leaving

$$\Delta w_{kj}(t) \;=\; \eta \sum_{n=0}^{t} \alpha^{t-n} (y(n) \;-\; f(x(n), w)) \varphi_k{}'(\vartheta(n)) o_j(n). \qquad (4\text{–}9)$$

As already explained, $f(x, w) = \varphi(\vartheta(t))$. Thus equation (4–9) will change into:

$$\Delta w_{kj}(t) \;=\; \eta \sum_{n=0}^{t} \alpha^{t-n} (y(n) \;-\; \varphi(\vartheta(n))) \varphi_k{}'(\vartheta(n)) o_j(n). \qquad (4\text{–}10)$$

In summary, the learning algorithm (a) chooses in an arbitrary way the examples z^μ, (b) orders them to a pattern stream $z(t)$, and (c) transforms the pattern stream $z(t)$, (where t is the natural numbering $1, 2, ...t...$ of the examples in the order of their appearance) to the sequence of weight values $\boldsymbol{w}_{kj}(t)$, formed by adding a weight change $\Delta_{z(t)} \boldsymbol{w}_{kj}$ to the previous weight value. In this sense the two streams $\boldsymbol{w}_{kj}(t)$ and $\Delta_{z(t)} \boldsymbol{w}_{kj}$ contain equivalent information:

$$\{z^1, z^2, ..., z^\mu, \ldots, z^N\} \longrightarrow \{z(1), z(2), \ldots, z(t), ...\} \longrightarrow |\{w_{ji}(1), w_{ji}(2), \ldots, w_{ji}(t), ...\} \longrightarrow$$

$$\left\{\Delta_{z(1)} w_{ji}, \Delta_{z(2)} w_{ji}, \ldots, \Delta_{z(t)} w_{ji}, ...\right\} \qquad (4\text{–}11)$$

This reasoning is valid in all the example presentation manners, which ensures an equal probability $p(z^\mu)$ for every example z^μ to be presented. The described equations will help for instance to imagine when the example stream leads to weight degradation and how this can eventually be avoided.

Later in this thesis, we will develop the idea that structuring of the example presentation ordering is necessary to avoid learning problems. From the above equivalence between sample presentation ordering and weight value change, one may therefore rightfully deduce that structuring the example set implies a structuring of the network. In this sense, the technique to be developed can be viewed as a first step in the direction of modular synthesis of neural networks with a guarantee on learning by construction. This makes our technique to a viable alternative to the “knowledge annealing” technique advocated by [144].

4.2.1.2 Resampling schemes.

Although an indication about the changes of the trajectory on the global error surface, characterized with the change of the weight vectors, is given by the data stream $z(t)$, this indication is not straightforward. But certainly, some useful conclusions can be drawn.

The most general conclusion about the dependence between the weight changes and the differences between subsequent examples follows directly from equation (4–9): large

weight changes are proportional either to an abrupt change of input stimuli or to a large error (large differences between the target signal and the network output). In contrast, a stabilization on the weight oscillations is to be expected when the target signal is almost learned or when the input values are changing smoothly. The learning trajectory as well as the weight plots have one global path of changes, but on the short term they are exposed to small fluctuations [160]. In the long run, large changes in the weight values can be expected when consequent examples cause weight changes with the same sign (in the same direction). This obvious conclusion, derived from equation (4–10) will be considered when attempting to direct the learning trajectory in the forthcoming sections.

By now the importance of the inputs has been discussed in neural literature in terms of saliency or sensitivity analysis. The two basic methods for inputs importance estimation are known as predictive importance and casual importance. *Predictive importance* is concerned with the increase of generalization error when an input is omitted from a network [101]. By *casual importance* estimation the inputs are manipulated in order to find out how much the outputs will change [101]. A common conclusion of the sensitivity analysis methods is that different measures of importance are likely to be used in different applications of neural networks.

4.2.1.3 Bootstrap resampling.

Bootstrapping is used in neural research as a method for estimating generalization error based on ”resampling”. Since resampling is a typical feature of the bootstrap method, it can be used to asses the impact of sampling variation in the data set on the trained weights.

The bootstrap method generally requires no major assumptions other than simple random sampling and finite variance. This allows us to investigate the sample variability in a statistics of interest, by making repeated computations by a randomly drawn samples with the same probability distribution as the initial samples. The distribution of the statistic of interest can then be observed over the resampling experiments. Under appropriate conditions the distribution of the resampled statistics corresponds to the sampling distribution of the statistics of interest.

In [15], the bootstrap method has been used to assess the sampling variation. First a large number N of pseudo samples of size n have been drawn independent of the original sample and the statistical characteristics of every such sample are used to draw the conclusion for the original statistics. This bootstrap methodology allows for the identification of those variables that are truly predictive for the target. Our interest in this work is that it gives a statistical validation of the observed relationships between the input information, processed by the neural network and its target.

4.2.2 Example stream features.

Our experience shows, that some signals are more difficult to be learned by random pattern selection than others, see for example the discussion in Section 3.3.3. The perfectly symmetrical signal in figure 3–14 will never converge to the desired value, while the “easier” signal with imperfect symmetry leads almost effortless to a satisfactory solution. Since the difference in the two experiments lies only in slight changes of the contents of the target patterns set, one may expect, that there is a relation between target signal features and its learnability, correspondingly with its reliability.

The analysis of the experiment illustrated in figure 3–14 brings us to direct our attempts on the definition of signals that can cause reliability problems. Moreover, it suggests a way to overcome reliability problems, caused by such signals, by supplying the training samples in a way that the correspondent weight updates will not cancel each other. The experiment in figure 4–1 at the beginning of this chapter shows the influence on speeding up the learning process of different manners to order the examples of a training set. This latter experiment concerns a signal, which is learnable by a "standard" random pattern selection scheme.

The described experiments from figures 3–14 and 4–1 have been especially chosen to illustrate the potential impact on learning speed and reliability of the ways to select the training samples. An interesting question is whether the approximation of an arbitrary signal is affected by the example selection manner. If the relation between signal learnability and its structure, the way it is sampled or the order of presentation of the samples can be found, then learning quality and reliability is to be considerably improved. Such a relation, expressed by a (preferably simple) criterion will allow an easy check on whether the current training set can be learned easily, and – if not – how its learnability can be improved. This check can be made before training starts, which will give an indication whether learning problems can be expected with this signal if it is trained by means of random sampling. Reliability problems can appear as well during training.

4.2.2.1 Definition of a cancelation training set.

In Section 3.3.3, the behavior of a symmetrical function in the initial shallow valley has been analyzed. The initial shallow valley is a peculiarity especially for networks, constructed by zero–centered sigmoid neurons. As can be concluded by the analysis in section 3.3.3:

> ***If the training set consists of pairs of samples which cancel their impact on the progress along the learning itinerary, a reliable learning by means of random example selection can not be guaranteed.***

It is to be expected, that if the learning process meets a very flat error region, such a training set will bring prolonged training times as well. In both cases there is a large chance for the learning process to be of a poor quality. For instance in the approximation task from section 3.3.3 optimization can be directed to the optimal point on the bottom of this valley. In a flat region, such a signal will cause an itinerary which rather follows the lows of the random process (Figure 2–12b) than to take a direction, which corresponds to a better mapping of the training problem. The formalization of this learning problem, as well as the made experiments, brings us to generalize that this effect appears by a wider group of signals which have the potential to bring the learning process to an unwanted ending. A basis for determining the range of signals, which have the potential to cause reliability problems with a cancelation nature, will be the observations on the global weight vectors behavior (respectively on the taken learning trajectories) when unreliable learning occurs.

Different weight behaviors, associated with non–learning are (1) degradation of the network parameters; (2) all the parameters go to a few parallel trajectories; and (3) saturation of the network units. The first two are observed when the cancelation situation is present.

An intuitive rule for reliable learning in this case is to keep the weight vectors in the range such that neither saturation of the network units, nor network degradation occurs.

This implies, that the internal representation should be kept distributed enough not to allow a degradation of the weight vector values. Of course, this might not be true for networks that need pruning. For some tasks, networks with a very large amount of weights are designed, and the learning process prunes out the unnecessary parameters.

Interpreted through the error landscape paradigm, degradation of the network takes place, when a high energy valley, or local minima is reached. The optimization goes into the direction of the local lowest point. Degraded to a few parameters, the network is able to learn a detail of the problem, corresponding to this high energy stationarity.

The observed effects of the existence of a cancelation phenomena during training have been analyzed by a typical example in approximating a symmetrical training set. In this case degradation is not an obvious phenomenon since the weight vectors have no time to grow initially and loose potential due to cancelation. The degradation in this case can not be distinguished from the lack of initial symmetry breaking. Here an attempt will be made to generalize the analysis made for a symmetrical signal to cases where a secondary cancelation of the training examples and respectively network degradation will appear.

As discussed already, the degradation can be encountered in two forms. The first one is characterized by the typical behavior of the network parameters to approach a zero point which, when mapped on the global error surface paradigm, corresponds to a multi–dimensional high energy valley, at which most of the network parameters reach its bottom. The bottom of a multi–dimensional valley can have several non–zero directions, which correspond to the non–zeroed network parameters. The second form of degradation is characterized by all network parameters taking a few parallel trajectories.

Lets go back to the assumption, made for analyzing a symmetrical signal, that n uniformly distributed samples are taken from the input signal. Let the input signal is scaled in x–direction between –1 and 1. Then there are $n/2$ samples x_i in the range [–1 0], and $n/2$ samples in [0 +1], such that for every sample x_i there exist a sample x_j, $i \neq j$, for which $x_i = -x_j$. To every input tuple x_i, x_j, $(x_i = -x_j)$ corresponds y_i, y_j, tuple of the target signal, such that $y_i = y_j$.

The explanation of the cancelation effect was made on the assumption, that two corresponding elements of a tuple are presented in subsequent order. This way the impact, which the second example from the tuple will have on the learning development, is canceling the impact from the first one. In practice the probability that two such examples are presented in subsequent order is not very high. But since two subsequent examples are most often on different sides of the zero point, in the longer run the summary impact of the example presentation is zero.

Equation (4–10) will be used in order to generalize the appearance and influence of this phenomenon for an arbitrary signal. In it the weight adaptation Δw was expressed as a sum of the changes, caused from a sequence of training examples. This sum can be divided into two sub–sums:

$$g_1(t) = \eta \sum_{n=0}^{t} \frac{\alpha^{t-n}}{2} y(n)\varphi'(\vartheta(n))o_j(n) \qquad (4\text{–}12)$$

$$g_2(t) = \eta \sum_{n=0}^{t} \frac{\alpha^{t-n}}{2} \varphi(\vartheta(n))\varphi'(\vartheta(n))o_j(n) \qquad (4\text{–}13)$$

This division of the weight adaptation time series is helpful to reveal the impact of the target sequence when the input sequence does not lead to meaningful changes in the network output, i.e. when a degradation of the weight vectors occurs. Only the first sum depends on the training signal (equation (4–12). The second sum is externally dependent only on the flow of input values (equation (4–13)). A similar description has been used for the analysis of the experiment in section 3.3.3 from equation (3–28).

By analogy with this analysis, it can be expected, that cancelation problems will appear when the first sum (equation (4–12) converges close to zero for one training epoch. With the assumptions made for random sampling this implies that the training signal has little impact on the training process. What the network produces as an output is in this case a combination of the mean of the signal and the impact of the input sequence. When the example sequence is arbitrarily ordered, the network approximates only the signal mean if the training has just begun, or stays at the already learned part feature of the signal in later stages of learning.

Therefore, in order to find out the impact of the target signal on the training process (in peculiar error surface areas), we need to analyze only the first additive component g_1. If equation (4–12) converges close to zero during one epoch:

$$\lim_{t \to N} g_1(t) \rightarrow 0 \qquad (4\text{–}14)$$

then it can be expected that the examples in the signal to be learned will cancel each other's impact even in shorter intervals due to the additive properties of random, zero–symmetrical sequences.

To determine the signals that will satisfy (4–14) let us look at this equation into more detail. The constants α and η can be ignored, since they are not adaptable for the "standard" way of performing backpropagation. Except on the target values, the equation (4–12) (which is implied in (4–14)) depends on the multiplicative term ψ, which for the i–th example is:

$$\psi^i = \varphi'(\vartheta(x_i))o_j(x_i) \qquad (4\text{–}15)$$

Therefore it depends indirectly on the input distribution. As in the symmetrical case it is convenient to scale the input signal between –1 and 1 and the target signal y between –1 and 1 for the zero–centered sigmoid transfer and 0 and 1 for the logistic one. This choice is made because of the active parts of the corresponding transfer functions. Since the scaling factor for the target signal y is not of importance for the condition (4–14), it is chosen to be [0...1].

Then if two subsequent training examples $z^{2i} = (x^{2i}, y^{2i})$ and $z^{2i-1} = (x^{2i-1}, y^{2i-1})$ have equal but opposite values for the products of target and ψ values (equation (4–16)), they will cancel to a large extent the impact of each other's presentation.

$$\psi^{2i} y^{2i} = -\psi^{2i-1} y^{2i-1} \qquad (4\text{–}16)$$

If the complete training set consists of such sample tuples, their sum will be equal and describes the condition in which cancelation will appear.

$$\sum_{i=1}^{N/2}(y^{2i}\psi^{2i} - y^{2i-1}\psi^{2i-1}) = 0 \qquad (4\text{–}17)$$

As can be seen from the plot in figure 4–3 of overall ψ_z–values of a zero–centered sigmoid network, there is a very high probability that the last two multiplicative terms of equation (4–12) provide an equal probability for the network to give positive or negative output. When this altering of the sign of network output is present in a short series of steps, the values are subtracted from each other. Thus only on the third multiplicator from equation (4–12), the target sample set determines whether training will lead to degradation.

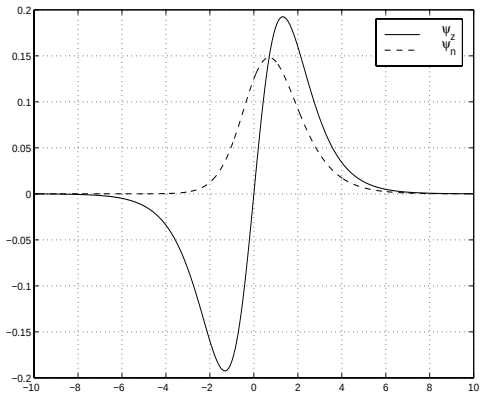


Figure 4–3: *Adaptation curves for different sigmoids, when training signal is excluded.*

The second sum, determining the network adaptation vectors, expressed by equation (4–13) follows the changes in the input signal. If the impact from the targets is mutually canceled, and the input samples are scaled between –1 and 1, it can be expected that equation (4–13) will give a zero summary result for one training epoch. Then equation (4–17) can be reduced to:

$$\sum_{i=1}^{N} y_i\psi_i = 0 \qquad (4\text{–}18)$$

This equation for a cancelation check is in general not suitable for prediction of the learnability (or reliability) of a signal, because the ψ values are not known. Our aim is to derive an approximate criterion, which depends entirely on the extracted training examples. For a symmetrical signal it is possible to substitute the ψ_i values with x_i values, as discussed in the following subsection.

4.2.2.2 Symmetrical signals and cancelation.

For every y–symmetrical signal the following dependence is valid:

$$\int_{-1}^{0} - y(x)dx + \int_{0}^{1} y(x)dx = 0 \qquad (4\text{–}19)$$

If sampled equidistantly and scaled under the assumed conditions, the input values x_i and x_{i+1} (corresponding to the tuple values y_i and y_{i+1}) have opposite signs and equal magnitude. This implies, that the x–values of the examples are contributing effectively only with their sign to the adaptation process, if presented sequentially. Then (4–18) can be represented in the following way:

$$\sum_{i=1}^{2N} y_i \mathcal{G}(x_i) \;=\; 0 \qquad (4\text{–}20)$$

where the function $\mathcal{G}(x)$ is as follows: $\mathcal{G}(x) \;=\; \begin{cases} 1 \text{ if } x \;\geqq\; 0 \\ 0 \text{ if } x \;=\; 0 \\ -\,1 \text{ if } x \;<\; 0 \end{cases}$

By sampling uniformly and at random, the probability that examples with numbers i and $i + 1$ have opposite valued x parts is very high. Cancelation will take place, if the x–coordinates of the drawn samples alter often. As was already shown in a number of experiments, cancelation can appear after a part of the approximation process has already been done. At later stages of the learning process, after some details of the signal or its main tendency are already mapped onto the network, equation (4–20) can be valid for the unlearned part.

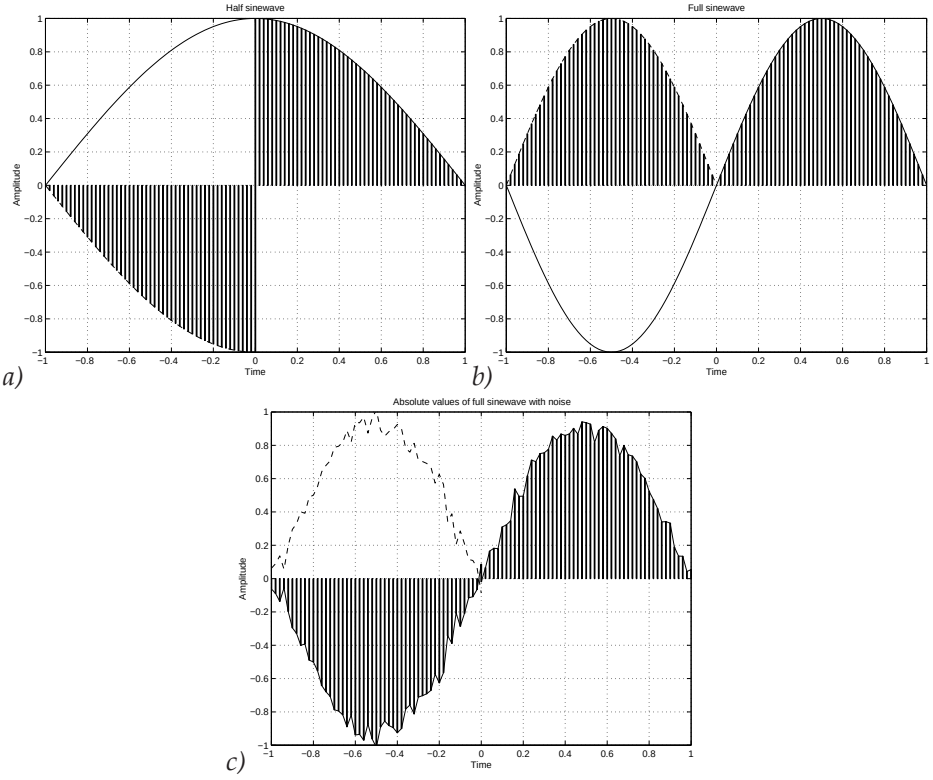


Figure 4–4: *Fast detection of cancelation in sinewaves.*

Equation (4–19) gives an illustrative rule, which allows to determine the danger for learning reliability in simple signals. For a half sinewave scaled as shown in figure 4–4a, equation (4–21) is valid. If substituted in equation (4–19), the integration will lead to a zero result and thereby signifies the occurrence of cancelation. A similar reasoning can be made for the signal from figure 4–4c, which consists of two positive sinewaves with additional uniformly distributed random noise.

$$y_1(x) = \sin(\frac{1}{2}\pi x + \frac{1}{2}\pi) = \sin(\frac{1}{2}\pi(x + 1)) \qquad (4–21)$$

In contrast, the full sinewave in figure 4–4b does not satisfy the continuous cancelation as can be concluded by equation (4–22). Once substituted in eq. (4–19), the integration will provide a non–zero result, which signifies the exclusion of cancelation.

$$y_2(x) = -\sin(\pi x + \pi) = \sin(\pi(x + 1)) \qquad (4–22)$$

The corresponding experiments confirm that the first and the third signal have cancelation nature, which reveals by random sample selection.

The way shown to predict reliability, respectively learnability, of simple signals is convenient because it is illustrative. In real–life applications the signal shape is usuallynot known. The predictions about the possible reliability problems should be made on the basis of the available (often noisy) examples. In such a case calculating the integral from equation (4–21) is not possible.

4.2.2.3 Cancelation criterion.

Neural systems learn from examples taken from the training signal by means of some sampling rule. The extracted samples determine what the network can learn about the task, represented by this signal. As was elaborated in the previous section the acquisition of the knowledge, contained in the extracted samples, might be obstracted by a restriction on the interaction between network and learning algorithm, which can be revealed by the specific representation of some tasks, i.e. a specific example set. The definition of a cancelation training set relates the signal, or more specifically the extracted example set, to its learnability. The definition assumes that the training examples are presented in a random manner.

Beside the contents of the extracted example set, its ordering (for instance the current randomization) can cause variation of the learning performance. Further in this section, experiments that show how different orderings of the same training set can considerably change the training duration and repeatability (and even avoid some learning problems) will be shown. Furthermore, resampling based on signal variability can bring a different outcome when repeating an experiment. Therefore, instead of checking the learnability of a signal (which we consider equivalent to checking the extracted example set) it is more convenient to examine the current ordering of the training examples and to change it if necessary. Still, checking in advance the cancelation properties of the signal remains a necessary step to determine whether further resampling or example reordering will be necessary in a particular case.

Both examinations (of the learnability of a signal, and of the current sample sequence) can be done in the same way if the cancelation criterion is expressed through the training examples only. Such a representation of the cancelation criterion will determine what type of example reordering will lead to improved learning. This will be the basis for creating an adequate sampling strategy.

The definition of a cancelation training set relates the signal to the potential and the certainty it can be learned. Therefore the formalization of the cancelation danger only through the training examples will help controling learnability and learning reliability, respectively.

Equation (4–20) determines the range of signals, which in terms of random uniform sampling (which presumes altering of the x–value sign at almost every iteration) has the potential to cause failure of the training process. The conclusion, which can be drawn from the previous section and from the corresponding experiments is, that the existence and subsequent ordering of pairs of training examples, which exactly cancel each other's impact, is not a strict condition determining the learning quality. If the summary effect of a few sequential target value presentations from the negative region with respect to the *x*–axis is equal to the summary impact of the target examples from the positive *x*–range, then equation (4–18) will reach the zero point. As a result the weight values continue to oscillate around the state they have reached already or degrade to zero.

As was shown before (see figure 4–1) reordering of the samples from the same training set can not only improve the quality considerably, but also speed up the learning process and ensure a repeatable training duration. In order to be able to test whether the constructed training set (after a randomization or special manner of sampling or ordering of the examples) has the potential to cause learning problems, the cancelation criterion should be constructed in a way, that will encounter the sequential order of the presented examples. Again the training set is assumed to be scaled within the [–1, 1] range on its x coordinate and within (0,1) range on its y–coordinate. This scaling helps to define the cancelation properties of the training set and may be independent of the scaling of the examples, which will be used for actual training.

The analysis of a symmetrical training set in a shallow valley, made in the previous chapter, was based on the assumption that two nearly equidistant (measured from the beginning of the coordinate system) samples are supplied sequentially to the network. The difference between their *y*–values ($y^i - y^{i-1}$) is approximately zero. A subtraction of their corresponding *x*–values ($x^i - x^{i-1}$) will give the doubled x^i–value. On the small time scale weight vectors have slow dynamics: they are adapted by small amounts and rarely change their sign. Therefore the change of the value of the network function is small, and its sign is determined by the sign of the training examples. In the beginning of the training process the parameter values are small enough to provide noticeable changes in the current values of $g_2(t)$ terms (see eq.(4–13)).

In summary, if two equidistant patterns of a symmetrical signal are supplied to the network in the beginning of the training process in consecutive order, the following set of equations is valid:

$$\begin{cases} x^i = -x^{i+1} \\ y^i = y^{i+1} \\ {g_1}^i(t) = -{g_1}^{i+1}(t) \\ {g_2}^i(t) = {g_2}^{i+1}(t) \end{cases} \tag{4–23}$$

In the beginning of the training process the values of $g_2(t)$ are negligible, because of the small parameters values. In this case, the initial symmetry breaking can occur only

due to the influence of the elements of the $g_1(t)$ sum. The instant values of $g_1(t)$, corresponding to equidistant examples, are canceling each other impact being approximately equal by value and opposite by sign.

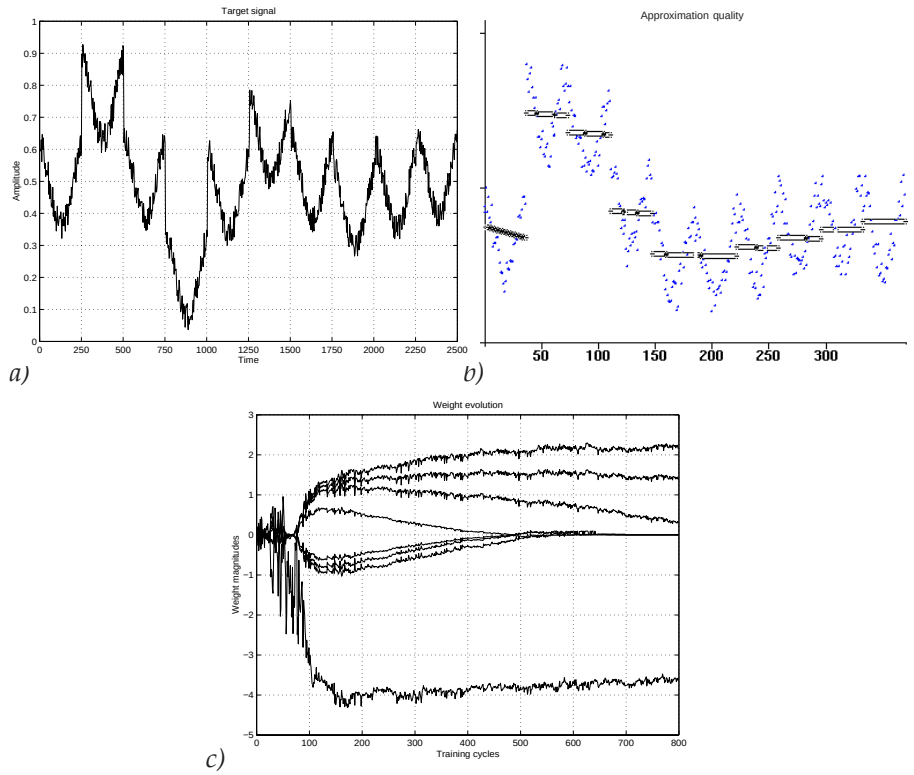


Figure 4–5: *An example for secondary cancelation.*

If the training set does not contain equal y–values to corresponding x–values, i.e. when

$$\begin{cases} x^i = -x^{i+1} \\ y^i \neq y^{i+1} \\ g_1{}^i(t) \neq -g_1{}^{i+1}(t) \\ g_2{}^i(t) = g_2{}^{i+1}(t) \end{cases} \tag{4–24}$$

is valid, then the calculated $g_1(t)$ values are different and the initial symmetry breaking occurs easily, since the adaptation steps are big enough due to the non–cancelation subsequent $g_1(t)$ values. Such a case is shown in figure 4–5c, where the weight vector symmetry is easily broken. In tangible terms a broken initial symmetrical phase corresponds to the learned main tendency of the signal, as shown in figure 4–5b. After the mapping of the main signal tendency, the combination network–signal shows cancelation properties. In the not learned part of the signal, y–values correspond to opposite and comparatively close values of the x–vector. This contradicting information stops further adaptation (when training is performed on the zero–centered sigmoid transfer network).

In both cases it is useful to define a directional coefficient of the changes in the signal that should be learned. The sequential presentation of two examples which have equal teacher values and different stimuli vectors will cause contradictory (canceling each other) adaptation steps. If the growing (decreasing) direction of a signal is taken, which will correspond to a non–zero direction coefficient value, the consequent adaptation steps will provide meaningful changes. The so–calculated ratio prevents from contradiction when for x–samples that are close in time correspond y–values that are close in magnitude. In many smooth signals, such a correspondence between x– and y–couples of values exists and there is in fact no conflict situation. In conclusion it can be stated, that a small direction coefficient ratio, indicated by a very small value of the fraction (4–25), is an indication for a cancelation situation.

$$\left|\frac{y_i - y_{i+1}}{x_i - x_{i+1}}\right| \ll 1 \tag{4–25}$$

This fraction takes into account the ambiguity, which appears when two stimuli, having the potential to cause similar $g_2(t)$ adaptation components, correspond to equal teacher values, which provide similar but opposite $g_1(t)$ components. A cancelation occurs rarely in two subsequent training steps, but even in a short training interval the normalized sum of the non–absolute values of all such tuples of examples is:

$$KRS(z) = \frac{1}{M}\sum_{i=0}^{M}\frac{y_i - y_{i+1}}{x_i - x_{i+1}} \tag{4–26}$$

where M is the number of the performed training iterations. M can span from one to few presentations of the training set. If for an arbitrary training set the sum of all such a fractions for the already ordered (or just randomized) training groups of N samples converges to a value near zero,

$$\lim_{M \to N} KRS(z) \to 0 \tag{4–27}$$

it can be expected that the so–ordered training set will bring the network to degradation and the learning process will be unreliable.

Equation (4–26) represents the sum of the direction coefficients for the current sampling of a signal. If condition (4–27) is valid for a training set, this training set can not be learned by means of random example selection. For example, if we apply the cancelation criterion to the signals from the experiment visualized at figure 3–14, then the KRS–ratio, calculated for the first signal, converges to zero. The development of the KRS–ratio corresponding to the second signal predicts that this signal can be learned if the extracted training examples are randomly ordered. In figure 3–14b the KRS–ratio of the sinewave, trained on the interval $[0, 2\pi]$, is shown. It predicts problemless training, which is confirmed by the corresponding experiments.

Actually the three ratios indicate the network learning performance on the same signal, exemplified on different ranges. The three example sets are representative: they contain enough information so that the internal generator of data can be learned by the network. Correspondingly, it is indifferent which of them will be used for training the network. The three KRS–ratios converge to different values. It could be expected, that the highest KRS–ratio ensures faster training, since it predicts that there are less examples, that have the potential to cancel each other's impact. Indeed, the performed experiments have shown, that the complete sinewave is much easier to be learned than the one,

trained on the interval $[0, 0.95\pi]$ or the noisy half sinewave. This observation suggests, that beside the learnability, the KRS–ratio development is related to the training duration and to the learning reliability, respectively. This dependence will be investigated in the forthcoming section.

The KRS–plots, shown in figure 4–6, predict learnability of very simple signals. The training process is entrapped in a stationary state in the very beginning of the training process. Once escaped from this state, there are no further risks for the learning success. By more complex signals the learnability, respectively the reliability problem, can appear in the beginning as well as in later stages of training, when the network has already learned to map a feature of a signal. Our goal is to investigate both learnability and learning reliability in realistic learning scenarios. The following investigations will be made by signals, which have the potential to cause prolonged learning or stand–still situations in later stages of learning, and not when an initial symmetry breaking still has not occurred.

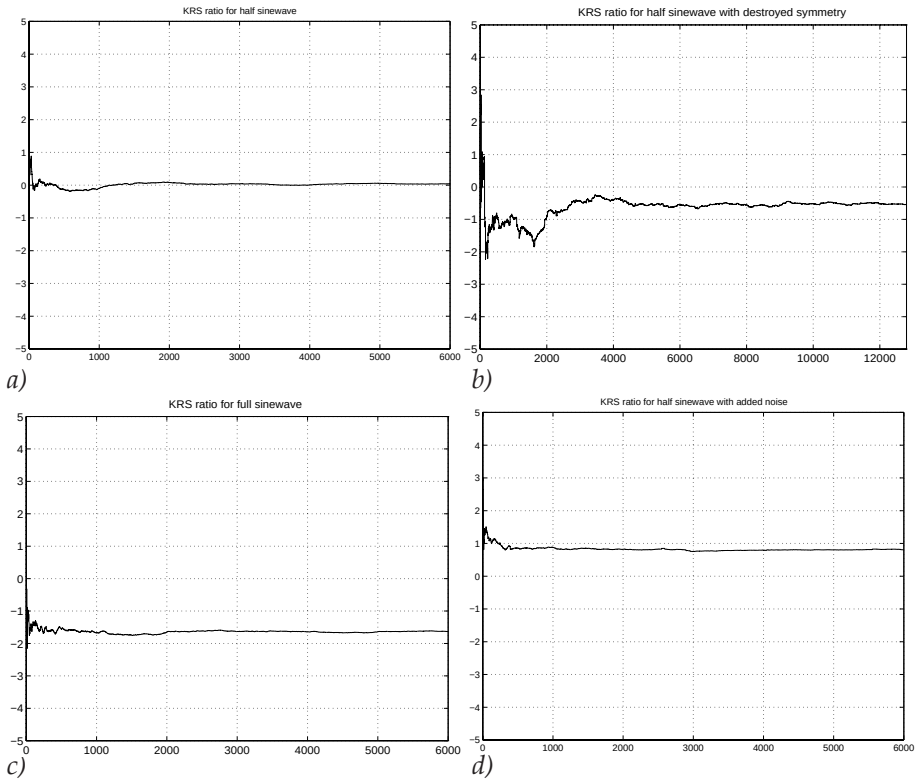


Figure 4–6: *The KRS–ratio evolution of the training sets, extracted by random sampling of the function* $sin(x)$ *on intervals* $[0, \pi]$, $[0, 95\pi]$ *and* $[0, 2\pi]$ *and with added noise.*

4.2.3 Reliability of cancelation signals.

The previously defined cancelation criterion aims to predict the learnability of a signal or of the extracted and (most often randomly) ordered training set. It gives a quantitative

estimate whether the created training sequence will cause bad learning. As discussed in chapter 2, the learnability of a signal is the main condition for estimation of its reliability. The other aspects of the learning reliability are training time and mapping accuracy, i.e. the degree to which a signal has been learned.

To estimate the reliability of cancelation signals, the mapping accuracy will be taken as a fixed value. The evaluation will be made on (a) whether a signal has been learned to the desired (fixed) degree, and if so, (b) how long the achievement of this goal took and whether the learning duration and reproducibility of the experiment with duration of the same range can be guaranteed.

In terms of the introduced cancelation criterion, a degree of cancelation will be defined. This degree will be computed in percentage of cancelation examples. The connection between the degree of cancelation and the measured reliability aspects will help to understand whether and how the existing training set can be learned faster by resampling or reordering of the training examples. Moreover, whether the example reordering can guarantee the training time (i.e. the replicability) of an experiment.

As already discussed in chapter 2, the stochastic nature of neural learning does not allow exact evaluation of reliability of the neural learning process. In an attempt to quantify the learning success and duration, a statistical investigation over the signals with unreplicable convergence will be done. The investigated cancelation signal is shown at figure 3–14. This signal has been chosen for higher practical convenience of the experimental results than could be provided by a symmetrical signal. This signal overcomes the initial symmetrical phase and shows cancelation behavior in later stages of learning. As already explained, cancelation concerns the effect of long–term presentation of training examples, which have the potential to provoke changes in the network state with equal but opposite strength. This phenomenon becomes noticeable when the error surface contains shallow valleys and flat areas, which can entrap the learning algorithm. The level of cancelation corresponds to the degree of internal symmetry and can be measured in percentage of examples with cancelation potential.

4.2.3.1 Periodicity.

As elaborated in other studies [10] [21], periodical signals may contain a risk for learning reliability. We agree, that for such signals cancelation can appear more often than by others. Since a periodical signal will give a more realistic view on the cancelation problem than simple symmetrical signals, our investigation will be continued on such signals. The periodical signal from figure 4–7 is artificially created to allow easy control over the content of cancelation examples in the extracted training set. To generate example sets with an a priori known level of cancelation, two different fragrancies of variation are added. The main tendency of the signal is a fading sinusoid, and the additive components with higher frequency are providing the required percentage of cancelation examples. A signal with a high level of cancelation will correspond to

$$f(x,y) = sin(\pi x)exp(-x) + cos(\pi y).$$

For the sake of convenience, uniformly distributed noise is added. The noise level is less than 10% of the magnitude of the training signal. The relation between x, y and t is as follows: $t = \frac{T}{X}x + y$, where $T, X, Y,$ are correspondingly the total number of samples and the maximum number x and y values ($T = XY$). The corresponding KRS–plots predict that the first signal will be more difficult to learn. We expect that the can-

celation additive components will cause bad learning in a single experiment and unreliable performance when subsequent trainings are performed. If during training the difference between the signal and the learned part satisfies the criterion in (4–26), it can be expected, that further optimization will be impossible. Every periodical training signal has almost such a potential. We expect that there are similar reasons, which cause degradation in on–line learning, when the new coming information causes forgetting of the already learned mappings.

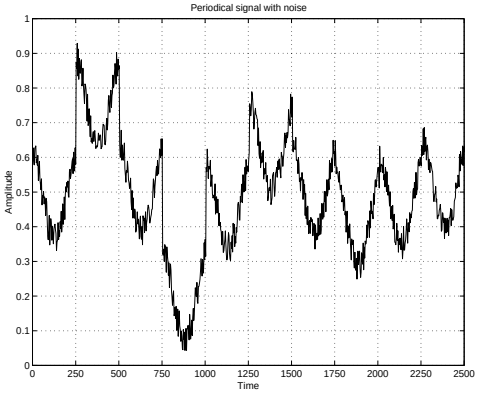
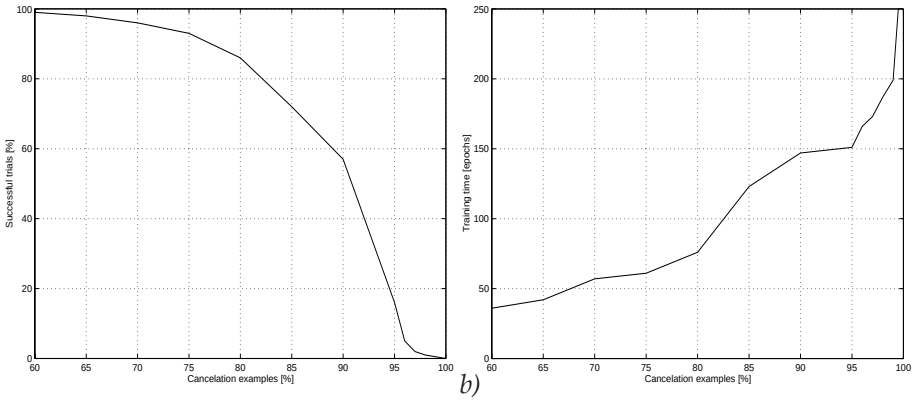


Figure 4–7: *The cancelation signal investigated.*

4.2.3.2 Mean and variance of training duration.

The learning time and the success of approximation of this signal has been extensively experimented with. The percentage of cancelation examples in the extracted training sets varies from 10% to 100%. For every particular number of cancelation examples 200 different training sets are extracted, ordered and applied.

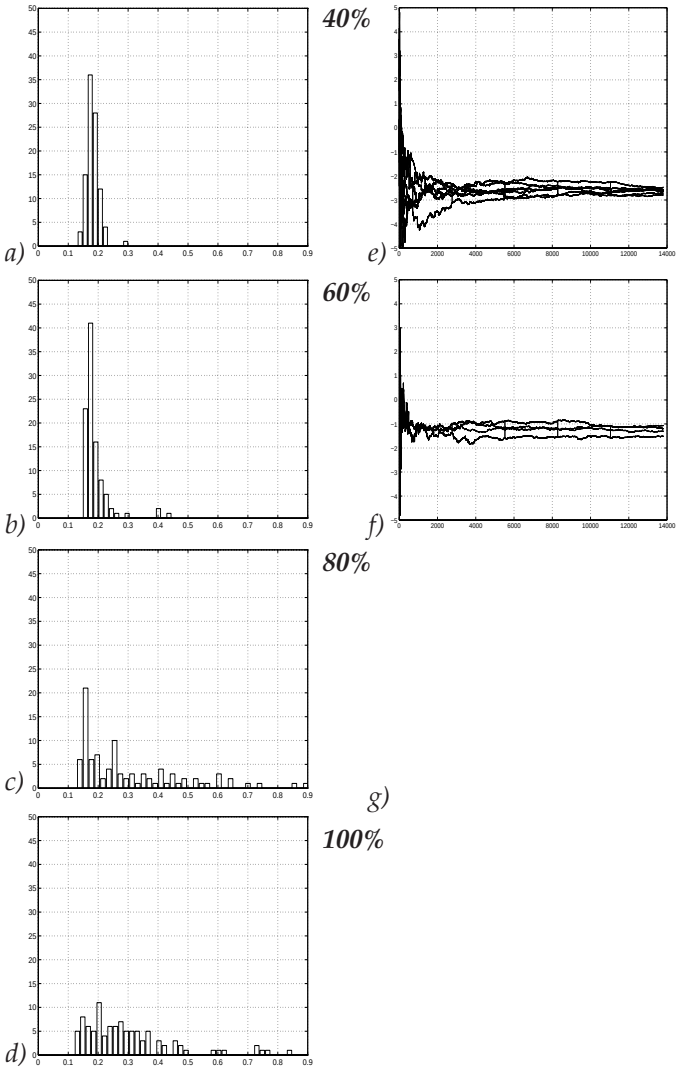


a) *b)*

Figure 4–8: *Network generalization performance.*

In figure 4–8a the results of this investigation into the effect of cancelation pattern sets on network reliability are depicted. The performance of the network on subsequent experiments with differently randomized training sets is plotted against the perceptual

presence of cancelation in the examples. It can be seen, that once a certain amount of cancelation patterns is present in a training set, the percentage of successful trials decreases quadratically and the experiment becomes non–reproducible. Simultaneously the training duration increases drastically. This is shown in figure 4–8b. The learning duration is plotted versus the percentual presence of cancelation examples after all the non–learned experiments are discarded.



5 Algorithms for windowed sample selection.

In chapter 3, the principal difficulties of the gradient algorithm on error surface areas have been discussed. Based on the behavior of the example determined trajectory, the relation between the idiosyncrasies of a signal, as present in the extracted training sequence, and its learnability has been established through the cancelation criterion. The provided analysis suggests sampling strategies, which successfully can remedy such cancelation problems, as shown further in chapter 4. Here the cancelation is to be prevented by adapting the criterion created for learnability and reliability enhancement to concrete signals and signal groups. The feasibility of the developed method is discussed within the developed theoretical framework and some comparative results are shown.

The neural method requires that a real–world problem is represented as a set of examples of its behavior. In the previous chapter it is shown that every signal and corresponding extracted training set have a certain degree of cancelation. The degree of cancelation in an example set is an important factor which determines its learnability and reliability. The analysis of the relation between reliability and degree of cancelation, as expressed by the KRS–ratio, suggests several ways for reliability enhancement. All of them require, that the KRS–ratio within one epoch should be kept high to prevent the learning development from cancelation.

In this chapter, various possibilities for realization of the so far developed strategy for reliability enhancement via example reordering will be specified. Their advantages by training signals with varying complexity and structure will be shown. Furthermore, there are training situations, which require additional effort for surmounting the states of indecisiveness and slow progress during learning. Summarizing all the possible cancelation situations and creating a tangible set of rules, which can ensure learning success by any of them gives a practical validation of this work.

The method created for reliability enhancement and the corresponding active training algorithms, which will be developed in further sections, have their weak points and pose some learning problems if applied in a wrong way. In later sections the main drawback of the training algorithms will be discussed as well.

In the following, we will first discuss the cancelation signal groups in more detail. The investigated signals are divided in groups according to the complexity of the cancelation problem that might appear. The simplest treatment is required by symmetry and parity problems, defined by [104] as second–order problems. This leads to a further formalization of the principal sample selection algorithm. Periodicity of a signal is shown to be a drawback of the regression method, when the least squares qualification, by far mostly used in neural learning optimization criterion, is applied. In analogy to the described second–order problems, degradation of the network due to cancelation appears as well when approximating periodical signals. The effect of periodicity is elaborated

to find an extended algorithm for sample selection. The ideas which are coming on foreground by treatment of periodical signals reveal the danger of cancelation and the strategies for treatment of a broader group of signals, which we name *general cancelation* signals. In such cases cancelation can creep in unnoticeably and result in poor/prolonged approximation. Finally we will review some practical cases.

5.1 Cancelation signal groups.

Many of the real–world signals have a cancelation nature. If training data are provided from a continuous process, every extracted training set will catch a part of the signal with different cancelation potential. This causes training to have different duration.

The purpose of this study is to show that statistical effects by data presentation can set in without further ado, but, when becoming apparent, a relevant treatment should be applied. Therefore, it is important to predict the range of situations where unreliable learning is expected to take place. Although the created cancelation criterion gives a good indication when learning problems are expected in a particular case, relevant quality enhancement methods will be obtained by dividing the cancelation signals in groups with respect to the complexity of the existing cancelation situation. Some examples are shown in appendix A.

5.1.1 Second–order problems.

In the well–known book of Minski and Papert on “Perceptrons” [104] the order of a predicate is defined as the size of the largest conjunction in the minimal sum of products logical form for that predicate. This implies that when both conjunction and alteration are predicates of order one, XOR is a predicate of order two. The generalization of XOR to more than two inputs is parity, which is not of finite order. The *group–invariance theorem* [104] generalizes the order of a problem using the arguments of symmetry. The order of different geometric functions has been defined, and related to the learning problems. The major conclusion from this important study is that successful learning is more dependent on the problem to be learned than on the learning mechanism.

From a practical viewpoint, pattern symmetry and parity are often given as examples of difficult to learn second–order problems. Their basic characteristic is that single pixels alone do not carry information about the solution of the problem. Sufficient information can be extracted from pixel pairs that are related to mirror symmetries or that are corresponding pairs in the 2–dimensional parity problem. It is a problem for a least squares criterion to do the optimization when such ambiguous information is to be learned. Further on in this section some examples of symmetry and parity problems are shown.

5.1.1.1 Some elementary observations.

Cancelation effects are introduced in this thesis by the half sinewave. A number of experiments have been performed with different symmetrical signals as well; these experiments confirm nicely our proposition, that every symmetrical signal will get a learning process into cancelation. Figure 5–1 visualizes two second–order learning tasks and their corresponding KRS–ratio development during random sampling.

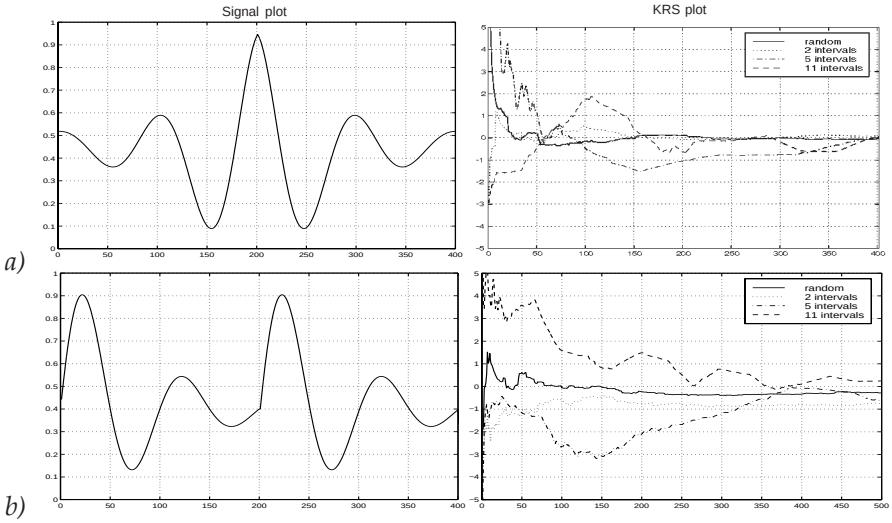


Figure 5–1: *An example of two signals with second–order problems.*

Variations of the KRS coefficients development when the signal is sampled at random within different number of training subintervals are shown with non–solid lines. The training set was divided into 2, 5 and 11 intervals, respectively.

Since the neural method is never used on such a 'clean' signals in practice, an interesting question is whether the cancellation will disappear once noise is added to the signal. This reasoning has to be contrasted to the noise injection learning methods, that imply adding an input or weight noise during training. They logically improve not only the generalization ability of the network, but also the learnability of this signal. We do not try to prove that the noise in the training data (in our range of reasoning in the target signal) will improve learnability, which is obviously so, but to show, that although there is a noise in the signal, it still has high cancelation potential due to the statistical characteristics of the examples (see Appendix B), and therefore a learning failure is possible. In the next experiment, noise is added to the target signal. We may expect that, if the added noise is with higher amplitude, or the number of training samples is not big enough, there is a chance the cancelation properties of the signal are decreased and hence learnability improved.

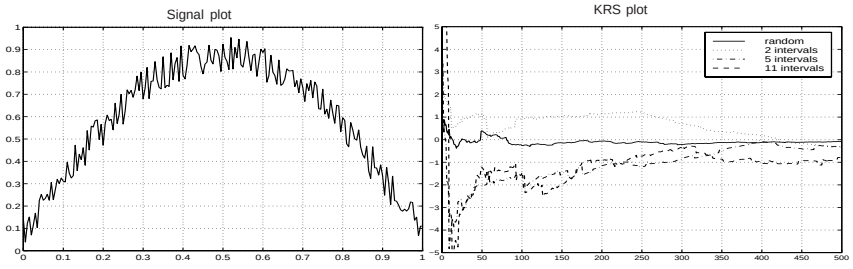


Figure 5–2: *Prediction of learning a sinewave with noise.*

In figure 5–2, random noise with normal distribution and amplitude up until 10 percent of the amplitude of the signal itself is added to the half sinewave. Learning difficulties are present, if training is performed by arbitrary presentation of the training examples. Logically, when noise with normal distribution is added to a symmetrical signal, the chance for cancelation grows with the number of the extracted training patterns, since the distribution of random samples is becoming more uniform.

Simple second–order problems are characterized by a prolonged initial symmetrical phase. The KRS graphs for random example selection by all the signals (the solid lines) predict that differentiation of the internal representations to a specific solution will not occur for all these signals. By the signal from figure 5–1 a degradation of the network parameters may occur, even after the initial symmetrical phase is broken. This result is due to the used optimization method which will be generalized in the forthcoming section.

5.1.1.2 Influence of sampling.

So far, training has been performed by means of random sampling of signals, which have explicit symmetry or can be divided into contradicting pairs of examples. In figure 5–3 a cancelation target set is extracted from signals, which by means of random equidistant sampling does not show cancelation properties explicitly. In other words, sampled symmetry sets can be created by choosing not–equally spaced contradictory patterns. We have created such training sets artificially (Figure 5–3a,b), but a number of pattern selection methods or practical sampling recommendations, efficient otherwise, can also end up with creating a cancelation example set (Figure 5–3c).

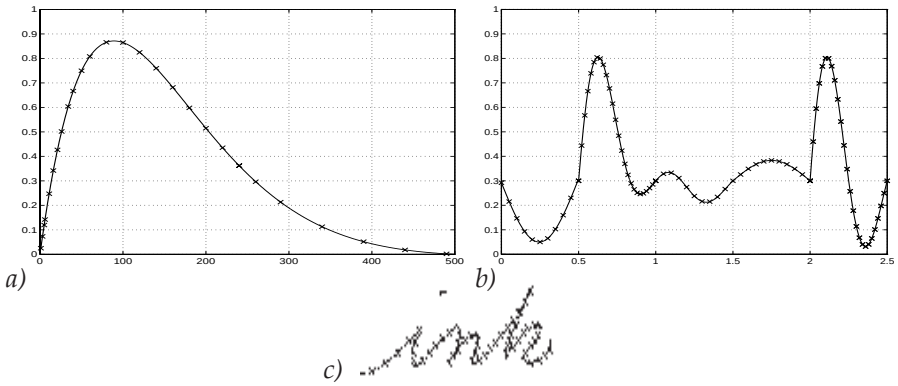


Figure 5–3: *Functions from which symmetrical training can be extracted.*

Handwritten word recognition (HWR) is a typical example of when sampled symmetry can creep in through the choice of practical sampling strategy. Handwritten Word Recognition, also called isolated handwritten word recognition, deals with the problem of machine reading of handwritten words, generally with the assistance of a lexicon of all valid words. A handwritten word is typically scanned in from a paper document and made available in the form of a binary or grayscale image to the recognition algorithm for Off–line HWR. The problem differs from On–line HWR where the writing surface is frequently an electronic notepad or a tablet, and where temporal information (the tra-

jectory of the pen as it traces the word) is available to the recognition algorithm, which attempts to recognize the writing as it is being written.

Figure 5–3c illustrates an often used approach for sampling the text by Handwritten Word Recognition. The complete word is a training signal. Depending on the style of the writer all up– and down–going lines might be carrier of important information. This is the reason why the so–defined signal (the word contour) is divided on a sequence of *strokes*: line segments between two sign changes in the writing direction. Each stroke is characterized by a 5–tuple: the starting point, three equally spaced intermediate samples and the end point. Ensembles of strokes can be identified as characters, which in turn can be assembled to words. The normalizing storage of strokes by 5–tuples, that blurs away the differences in angular writing raises symmetries in the created training set. For instance, the first character of the word in figure 5–3c will be found from an upgoing and a downgoing line at different angles, but with a symmetrical representation on stroke level.

5.1.2 Cancelation and periodicity.

Intuitively, a periodical signal is expected to multiply the difficulties, which appear in learning a symmetrical and other second–order problem. A study done by Bishop [21] derives the difficulties of the neural method to approximate periodical signal in analogy with the conceptually close problem of regression in statistics.

For treatment of some periodical signal, division of the signal into windows and training with randomly drawn samples within those windows is not sufficient. There are still difficulties for the network to break the symmetry or to prevent itself from unlearning. The reason is that the order of the problem is decreased but is still high enough to provoke ambiguous “lessons” which the teacher signal gives and thus high enough to cause unlearning.

Of course, if the reasons for bad learning are only and entirely dependent on the order of the problem, the neural method should not be able to learn any signal, since a polynomial, that changes its direction K–1 times, has a degree of at least K.

Further a simple technique is proposed, which breaks the order of the problem by dividing the input space in a way to prevent from invariance and indecisiveness. Increasing the abruptness of the input space has been suggested otherwise by saliency analysis methods. Here the relation between the order of the problem, its cancelation potential and necessity of increased abruptness is investigated.

5.1.2.1 Least squares optimization for periodic signals.

The learning process of artificial neural networks is very similar to the way many statistical models do estimation. Modeling a continuous function of input variables is comparable to the regression problem, well–known in statistics.

If neural networks are placed in the same framework as statistical models for classification and regression, the assumption should be made, that there are N independently chosen pairs of inputs x and targets y, whose distribution is unknown. The set of all such couples $\{z^\mu \equiv (\boldsymbol{x}^\mu, y^\mu) : \mu = 1, 2, ...N\}$ is the example set. The goal of network training is not simply to learn the example data set z but rather to model the underlying data generation process. In this sense the neural task is to learn to predict the target y if the un-

known input x is given to the network. The most general description of a process which generates the data is in terms of the joint probability distribution $p(x, y)$ over input and target variables. This joint probability distribution can be represented as:

$$p(x, y) = p(y|x)p(x) \tag{5–1}$$

where $p(x)$ is the unconditional distribution of the inputs and $p(y|x)$ is the conditional distribution of the target variable when the input vector $\boldsymbol{x}$ is given. The task already formulated for the neural method to predict the value of y for unseen x is achieved if the conditional distribution $p(y|x)$ is known. Modeling this distribution is done by optimizing the probability the network to give an output y when input x is given for the complete input space. There are many ways to optimize this probability [19]. The basic optimization methods are founded on the Maximum Likelihood Principle. If optimization should be done for the data set D, containing N examples $\{x^\mu, y^\mu\}$, which are drawn independently from the same distribution, the likelihood function can be defined as a multiplication of these probabilities. The probability for the complete data set D when the weight vector w is given is represented by:

$$L(D|\boldsymbol{w}) = \prod_{\mu=1}^{N} p(y^\mu|x^\mu) \tag{5–2}$$

where $L(D|\boldsymbol{w})$ is a likelihood function. The different choices in error function arise from different assumptions about the form of the conditional distribution $p(y|x)$. The least squares approach corresponds to finding the maximum likelihood for the special case in which $p(y|x)$ is modelled by a Gaussian distribution which is spherically symmetric in y–space and has a x–dependent mean [21]. Translating this representation to an Euclidean space makes least squared optimization inappropriate for periodic targets.

5.1.2.2 Cancelation effects by the periodical signals.

The above mentioned difficulty of the regression method to perform optimization with least mean squares is conceptually close to the problem which appears with symmetrical signals. After calculating the cancelation coefficient for a periodical signal, it often indicates that the signal is difficult to be learned. Going back to the experiment, which visualizes the stages of training process, it can be observed that first the most global feature of the signal (the main tendency) is learned by the network. If the signal is periodical, the remaining part to learn contains a continuous sinusoid (cosinusoid), and correspondingly is expected to have a high cancelation potential. Secondary cancelation due to periodicity of the signal will cause convergence problems.

The training sets, shown in figure 5–4, represent such synthetic periodical signals and their corresponding KRS–ratios. Synthetic signals are expected to have much more training problems in terms of cancelation, which causes degradation of the effective network. This is so because there are on average much more equalities designed into such a signal and thus ambiguities for the training method and consequent unlearning.

This analysis and the experiments with synthetic signals show, that degradation–like phenomena are eased by the periodicity of the signal, because of the reasons explained earlier in this section. We claim that cancelation is mainly due to the specific property of the examples and their ordering, and can be detected by calculating the KRS–ratio. The investigation over many periodical sets, extracted from the same signal but with different cancelation content, illustrates that.

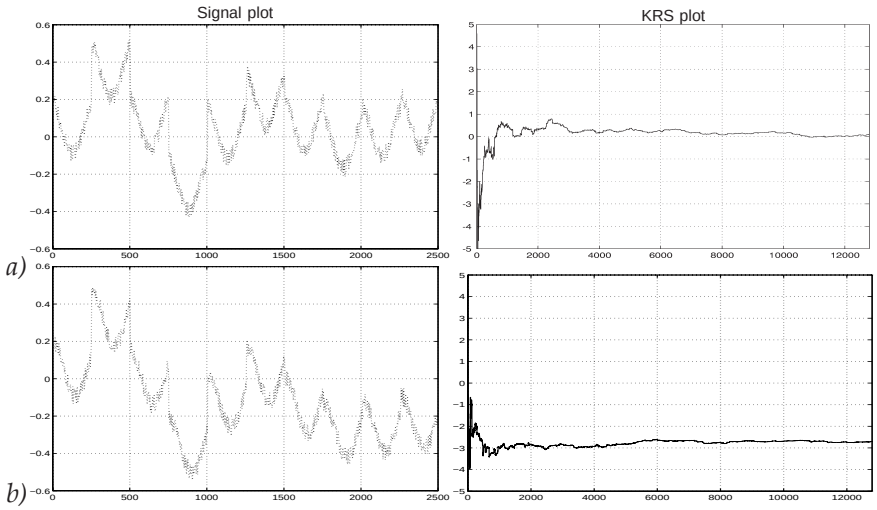


Figure 5–4: *Examples of synthetic periodical signals.*

A good illustrative example is the synthetic signals example sets from figure 5–4. Both signals are periodical, but the second one does not have a KRS–ratio development, which predicts cancelation. Repeating training of the second example set by means of random sampling shows reliable learning behavior. More details about learnability and learning reliability of training sets extracted from the same signal but having different percentages of cancelation examples has been presented in section 4.2.3 (Figure 4–8). When a cancelation is present in the training set, the signal periodicity increases the training difficulties and requires additional treatment when training.

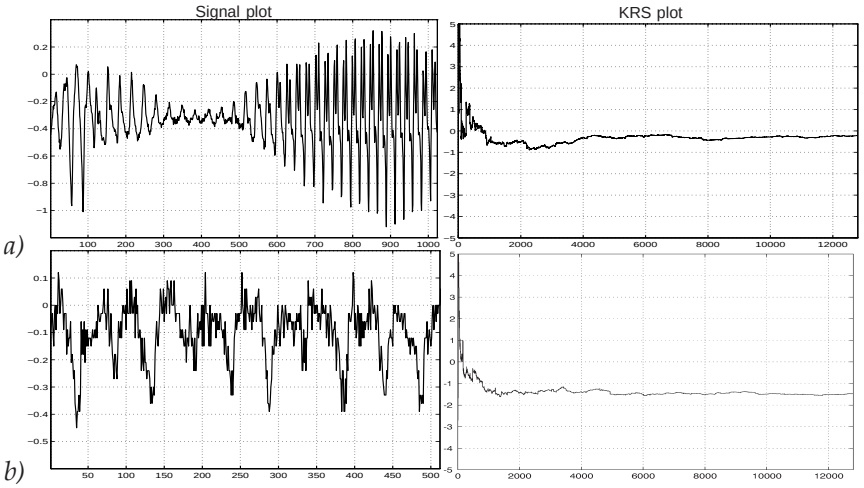


Figure 5–5: *Examples of periodical signals, recorded on a power–generator.*

A good example is the symmetrical signal from figure 5–1. Simple techniques, such as moving the window from which training samples are drawn, do not help avoiding the

cancelation effect. A simple division of the target sequence on intervals is also not a reliable method for training a cancelation signal. Making the signal input space more harsh by dividing it on parts, synchronized with the period of the signal ensures fast learning with low variance when training differently randomized training sets.

The signals shown in figure 5–5 are recorded during the emergence working mode of a power generator. The first one (Figure 5–5a) contains a large percentage of cancelation examples and its approximation usually fails when a random equidistant sampling is done on it and the training is performed with a network with zero–centered sigmoids. In contrast, the signal in figure 5–5b does not have a cancelation nature despite of its periodicity. Anyway, increasing its reliability in terms of speeding up the convergence process can be done by the training algorithm, proposed further on.

In contrast with simple second–order problems, cancelation can appear for more complex tasks in the beginning of training or may cause a network degradation in later stages. After differentiation of the hidden nodes has began and the corresponding fan–out connection has grown, the secondary cancelation prunes them and brings the network to some suboptimal solution. This secondary cancelation stems from the features of the learning development.

As discussed in section 3.4 and illustrated in figure 3–19, a learning process has several stages: the main tendency or another major feature of a signal is learned first; second level of detail is grasped by the network next; finally the smallest details of a signal are mapped. This implies, that after a certain level of mapping has been reached, further adaptation can be stopped because the system–task has come to a level of indecisiveness, because the task (signal) features, which has to be grasped next have a high degree of cancelation.

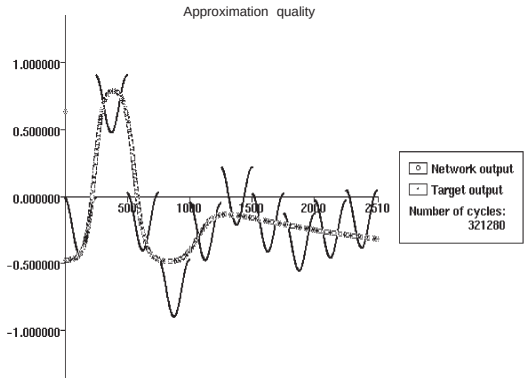


Figure 5–6:*The approximation quality of a signal by which secondary cancelation appears.*

An illustration of such a case is shown in figure 5–6. Even after exhaustive training, the quality of approximation does not improve a lot, because the next learning stage implies mapping with a high cancelation. The weight vector picture is similar to that in Fig.3–15b. A conclusion, which we have arrived at, is that the periodicity itself is not causing reliability problems, if there is not high level of cancelation in the examples.

5.2 Improved learnability.

As frequently implied in this thesis, learning problems that have network degradation as a final effect may emanate from a range of numerical and methodological problems during training. The central theme in this thesis is the cancelation phenomenon: the examples as newly presented to the network cancel the contribution to the learning process, made by previously presented examples. The neural training process can suffer even more, when the signal is periodic. In such cases there are additional similarities, introduced by the inability of the applied distance measure to represent a 1–to–1 correspondence for this group of signals in the Euclidian space. In general, additional problems, that increase the impact of cancelation are (a) the order of the training problem, which can be reduced by division of the input stream, and (b) numerical problems that cause insufficient *differentiability* of the input examples. In common practice, when there is no other information about the features of a signal $S(x, y)$ to be approximated, the training of the neural network is performed by presenting the x–coordinate to the network together with the function value, i.e. the y–coordinate. Since the signal has to be scaled in the transfer function range (usually [0, 1] or [–1, 1], the difference between two subsequent x–coordinates from the initial example order becomes very small, when the number of training patterns grows. In general this should not be a big problem. When the complexity of a signal grows, and especially when there are cancelation structures in it (like the signals in figure 5–4 and 5–5), there are a x–values which differ very little within the interval with a cancelation y–part. The differentiation between the training patterns within this range is hampered and the network learns only the dominating tendency, i.e. takes the average value in this range.

The division of the training signal on intervals with respect to the target signal can ease this learning problem in some cases, but it will not help, when the selection intervals for training include cancelation structures, since the corresponding input values (within this interval) differ very little. Presentation of the training examples in intervals forces such a hardly distinguishable x–values to be presented subsequently and in this way to increase the symmetry of the training system. The straight line output is expected to be seen as a response of the network in this range, analogous to the analysis of the simple symmetrical function (Section 3.3.3).

In this section the peculiarities of example ordering by second–order problems and the necessary extensions for periodical and general cancelation signals will be elaborated on. To alleviate the reliability problems by those groups of signals, some improvements of the learning method will be shown in the following. The handling of such problems will be scrutinized here in two parts. First, we will develop the basic algorithm for learning improvement. Subsequently, this algorithm is extended to handle secondary cancelation and periodic issues also.

5.2.1 Improved learnability for second–order problems.

An empirical investigation into learnability and learning reliability in general has been performed in the previous chapter. Multiple experiments, performed with different training sets, extracted according to the same sample selection scheme from a signal are summarized.

As suggested from empirical and theoretical analyses in the previous chapter, simple second–order approximation problems can be solved by destroying the contradictions in the training examples that cause unlearning of recently learned information. Approximation of simple symmetrical signals can be resolved in several ways. They are all aimed at decreasing the cancelation effect by increasing the discriminatory ability of the network, initially designed in a symmetrical way.

Assuring discrimination by destroying the contradictions, that emanate from learning second–order problems on highly parallel and symmetrically built architectures, is an easy task for a particular realization. As discussed in section 4.3.2, for learning a symmetrical signal it is sufficient to move the window from which the training examples are drawn. In this way the number of cancelation examples can be decreased. An alternative way for decreasing the cancelation potential of the training set is by simply "cutting of" a piece of a signal: the trained network is able to generalize for the non–trained part.

These intuitive methods have little practical significance. Basically, the shape of real–life signals is not always known. Furthermore, the power of neural learning is that the conclusions can be made on basis of examples from known areas of the problem. Thus approximation of mathematically formulated signals does not require an example–driven method such as neural learning.

The practical significance of our reliability improvement methods is not only restricted in the need for explicit knowledge about the nature of the signal, but also in the complexity growth of a signal. For instance the signal in figure 5–4 will not be better learned from a moved sampling window, due to the implied periodicity.

The benefit of the suggested solution for learning symmetrical signals is the practicality of increasing the KRS–ratio and breaking the indecisiveness of the neural system, respectively. This experience will be used to solve cancelation problems that appear in complex signals. A better treatment strategy for more complex higher–order problems or for signals in which cancelation becomes apparent in later stages of training is a systematic use of various sample reordering strategies.

Training with reordering methods, as formalized in section 4.3, shows considerable differences in the KRS–ratio development, as illustrated in figure 5–1b and figure 5–2 with non–solid lines. The plots correspond to arbitrarily chosen window sizes: 2, 5, or 11. They are giving good results for these particular signals. In general, the learning algorithm should involve calculation of KRS development characteristics and change the window size and/or the kind of reordering if necessary.

Since the training algorithm determines how to choose the training examples, it is an active selection algorithm, according to the definition, given in 4.1. This active training method will be named *windowed active training*, because the suggested resampling techniques for improved learnability are based on dividing the training set on intervals or on groups of examples, within which there is a low probability that cancelation example pairs to be picked up in a small time–scale.

5.2.1.1 An algorithm for active training.

The active training algorithm, which enables us to solve cancelation problems, is based on the stated KRS criterion. The developed resampling strategies help to increase the

KRS–ratio. The active training algorithm provides practical rules for finding out when resampling is necessary and what kind of (re–) sampling strategy should be applied.

The usefulness of our cancelation criterion has several aspects. Firstly, it gives an indication about the learnability of a signal in general and the training examples as created and ordered for the current epoch. Secondly, if the signal is learnable in general, the criterion can predict whether it or the current training set have a high degree of cancelation and correspondingly prolong the learning time.

These aspects of the cancelation criterion become apparent during the interaction between training set from one side and a randomized optimization method operating on highly symmetrical structures from the other side (Figure 3–20). The practical realization of this interaction should be incorporated in the learning algorithm. This can be achieved as follows:

- Check whether the training signal contains a high percentage of cancelation examples.
- If learning reliability or learnability in general is endangered, a resample strategy should be applied.
- As suggested in 4.3.1, a practical way to increase the KRS–ratio is by dividing of the training set on intervals and further picking randomly the training intervals and/or the elements in them. The number of selection intervals can be determined by starting with one, comprising the complete training set, and incrementing the number of intervals (i.e. dividing the training set). After every incremental step it should be checked whether the reordered training sets will have cancelation problems as well.
- When the cancelation check finds a division of the training set that predicts good training results, the network can start learning with the current number of intervals.

The evaluation of a particular sample stream can be achieved easily at any moment during training. Learning problems in terms of premature stopping of the adaptation process will appear only if the learning trajectory passes peculiar areas on the error surface. This is very likely to occur in the beginning of the learning process. However, at any further stage the adaptation may be stopped or enormously delayed as well. A good indication for existence of a cancelation situation – cancelation training set and difficult error surface area – is a training period with constant error (if the small fluctuations of the error function are neglected) as illustrated in figure 3–14e.

In terms of the active training algorithm construction, the last consideration implies a regular check for plateaus of the error function – if the training error does not change significantly for a long period, a new division of the training set on intervals should take place. If the latest reordering causes a decrease of the KRS–ratio, a decremental step of the number of training intervals should be made. In summary, the suggested active training algorithm differs from any passive training algorithm with the following actions:

- Before training starts: Check for high cancelation content (low KRS–ratio) in the extracted training set. If there is a risk for learning reliability, divide the training set on intervals, until KRS plot predict reliable learning.

- If during training the error value does not change for a long time, a new division of the training set with a corresponding KRS check should take place.

Taking a top–down scheme of division of the training set on intervals is in accordance with the KRS model (Section 3.4.2): In the beginning of the training process there is a high symmetry in the solution space, since the learning process is free to choose its traveling direction in accordance with the training problem encoded in the examples. In this training stage a large group of selected examples may direct the learning itinerary in the same global direction for a longer period without forcing the network to reach a local suboptimal state. Selecting large groups of non–contradictory examples will rather break the initial high symmetry. In later stages of the training process, decreased window sizes will increase the chance of obtaining satisfactory minimal states by avoiding jump–overs of local, possibly satisfactory minima.

5.2.1.2 Algorithm 1.

From the signal to be learned $S(z)$, where i^{th} sample from the signal is characterized by the value y of a signal at a moment x: $z^i = (x^i, y^i)$. The complete example set $\boldsymbol{z}$ can be represented as follows: $\boldsymbol{z} \equiv \left\{(x^i, y^i)\right\}_{i=1}^{N}$ is extracted by representative (for instance equidistant) sampling. In case of approximation or classification tasks, the examples in the training set $\boldsymbol{z}$ are randomly permuted, and they form a data stream $\boldsymbol{R_1} \equiv Perm_1(\boldsymbol{z})$. To predict the potential for a reliable training of a signal a cancelation criterion requires a large enough number of training examples. In order to avoid the uncertainty of the predicted cancelation content, due to the random ordering of the examples, the predicted cancelation is made on basis of more than one randomization of the training examples. The number of permutations depends on the size of the extracted training set. Every different permutation of the example set $\boldsymbol{z}$ forms a different data sequence $\boldsymbol{R_1} \equiv Perm_1(\boldsymbol{z})$, $\boldsymbol{R_2} \equiv Perm_2(\boldsymbol{z})$,..., $\boldsymbol{R}_s \equiv Perm_s(\boldsymbol{z})$. They can be gathered in a longer sequence as:

$$R = R_1 \# R_2 \#,..., \# R_s \tag{5–3}$$

The value of s is empirically determined and depends on the length of the training set. Basically, if the KRS–ratio calculated for one ordering of the training set is high enough, that training sequence is good for the current training epoch. More than one ordering of the example set can be made before training to start in order to insure, that the training problem is not due to the current (pseudo)–randomization. The pre–training cancelation check calculates the KRS coefficients with the training sequence, containing the elements from the s randomizations as given in equation (5–3):

$$KRS(\boldsymbol{z}) = \frac{1}{sN - 1} \sum_{j=1}^{sN-1} \frac{y^{j+1} - y^j}{x^{j+1} - x^j} \tag{5–4}$$

The potential danger of unreliable learning is present if this ratio goes close to zero:

$$\lim_{|z| \to \infty} KRS(\boldsymbol{z}) \to 0 \tag{5–5}$$

If condition (5–5) is satisfied, the training set $\boldsymbol{z}$ is divided into p subintervals. The new example sequence is made in accordance with the chosen reordering scheme. The KRS–ratio is now calculated with s new training sequences, ordered according to the subinterval scheme. If the example stream created in this way converges to a value dif-

ferent from zero, the training can start with this ordering of the training examples. Otherwise, the size of the training interval should be decreased until the cancelation criterion predicts reliable learning.

Algorithm 1:

[1] **Read the data set $z \equiv \{(x^i, y^i)\}_{i=1}^{N}$. Define the window size m to be equal to the number of training examples N.**

[2] **Make a training sequence R, consisting of several (s) data set permutations of z: $R_1 \equiv Perm_1(z)$, $R_2 \equiv Perm_2(z)$,..., $R_s \equiv Perm_s(z)$.**

[3] **Calculate the KRS coefficient for the so−created training sequence $R = R_1 \# R_2 \# ..., \# R_s$.**

[4] **If the KRS coefficient does not converge to a value close to 0, i.e. $\lim_{sN \to \infty} KRS(z) \to 0$, then go to [7]; otherwise**

[5] **Decrease the window size m to be equal to N/p, where p is preferably an integer number.**

[6] **Divide the data set z on p equal parts. Create a new training sequence R according to one of the sampling schemes specified in Section 4.3. The new training sequence should contain the same number s of epoches. Go to [7].**

[7] **Train by choosing the example stream within an epoch according to the sampling scheme with the chosen number of intervals. End.**

This algorithm gives the sequence of actions which should be taken to construct a trainable sequence. It is constructed in the way it should be applied before training starts. If during training the plateau of the training error is encountered, this algorithm has to be applied again. The initial data set $z \equiv \{(x^i, y^i)\}_{i=1}^{N}$, this time will be one with the current division on training windows.

5.2.2 Coping with periodicity and general cancelation.

This thesis has elaborated on the assumption that the success of neural learning and particularly of MLP networks depends more on what has to be learned than on the learning method itself. Many successful applications are based on the selection of best features for training. Feature extraction is a major data preparation step for classification tasks. In general, more specifically in approximation and prediction, the information about the neural task is encoded in the target values, since the input example stream most often encodes temporal information or the position of the sample in the signal.

The here developed algorithms try to give a global strategy for example stream reordering that can ensure reliable training. Our practical rules for training are valid for both cases: ad–hoc learning and construction of specific features. In Algorithm1, the changes of the example stream with respect to the target signal are embedded. As discussed earlier in this section and in 4.3.2, this can be insufficient for more complex signals. Changes in the input can provide for additional improvement of the learnability. As already discussed in Section 4.3.2, signal changes in the input stream have to be directed to increase its differentiability in case of cancelation. This is the reason why in Algorithm 2, which has to solve more complex cancelation cases, it is endeavored to increase the input stream differentiability.

In Section 4.3.2 it is shown how the weight adaptation depends on the input values and on the difference between subsequent inputs. On basis of this dependence, various practical ways to do that can be derived. They are related to the characteristics of the training task. Further on, several ways for input stream construction will be shown.

As discussed earlier, both periodical and general cancelation signals give rise to an additional difficulty when training without a priori knowledge about the input–output space. The constraints, introduced (1) by existing cancelation structures within relatively smaller ranges of the overall signal space together with (2) the decreased discriminability because of the similarities in the input values, increase the symmetry in the learning system and create extra conditions for bad reliability performance. The suggested windowing method alone does not always alleviate the cancelation problem in this case. On the contrary, it can make it stronger since it forces the values from the same training interval to be supplied in a sequential order and thus decreases the differentiability provided by the input space. For instance, in periodical signals there are additional similarities, introduced by the inability of the applied distance measure to represent 1–to–1 correspondence for this group of signals in the Euclidian space. The enhancement suggested in section 4.3.2 of the discriminability of the input stream has a particular realization for periodical signals – besides ensuring that two subsequent input vectors have a different impact on the network state, they should provide for a non–ambiguous input/output correspondence. In a similar way, additional training problems within general cancelation signals can be prevented.

For every particular task, there is more information available when a certain event (certain example of the task behavior) has happened. For instance, for periodical signals, one may prior identify repeating phases as: the season, month, or week in stock market prediction tasks, the frequency a machinery rotates with, the standardized frequency of power generation. For ECG signals the kind of the pulses of the heart beat by a healthy person, the duration of the different phases etc. is known. To alleviate the network mapping process such additional information can be used instead of or together with time, order or any other “standard” available knowledge, encoded in the input stream.

A priori information can be supplied to the network as an additional input stream. For instance, approximating a power signal (cf. further in this section), can have couples of values for the time and phase information as an input vector and the signal, measured at the output of the diagnosed device, as an output.

Such a case is illustrated in figure 5–17a. Here, the division of the input space on intervals stems for cancelation. As a result of long training, the network succeeds in learning the R pulse of the signal, where the differentiability of the input–output space is higher, but fails for the other signal components. We now present an algorithm, which summarizes in a working consequence the considerations discussed so far.

5.2.2.1 Algorithm 2.

Algorithm2 combines all the considerations for training signals with high cancelation potential and arbitrary complexity made so far. The signal to be trained with this algorithm can contain any periodical or cancelation structure in it. The changes with respect to the “normal” training procedure (as described in Section 1.2.1) are made before and during the training phase. Before training starts, there are two basic steps: (a) the order

of the training examples should be determined in accordance with the training signal, and (b) the kind and division of input information is to be made.

Algorithm1 gives the basic rules for finding out when and what kind of example reordering should take place. It can be applied directly to simple cancelation problems. In such cases it is usually enough to find the number of division intervals, by which cancelation will be less and to train with this fixed number of intervals. When the complexity of a signal increases, the corresponding error relief is becoming more complex as well, and the optimization process may require further tuning. As indication when new example reordering will be necessary, the encountered plateaus of the error function are used. If the optimization process does not progress for a long period, the current manner of example sequencing is not efficient anymore. A new change of the number of division intervals can improve the performance on the current error landscape area. For instance, if currently a deeper valley or a flatter plateau should be passed, the learning trajectory should be guided to make a longer shift in the same direction. Respectively, the examples which will ensure that the time series (4–9) have less oscillations, have to be chosen. If the smaller size of windowing intervals gives a better KRS–ratio, a step backwards to a more optimal number of intervals should be made. Mapped to the training algorithm, this implies that during the learning phase a check whether there is a plateau of the thinning error should be made.

Algorithm2 has to deal with complex cancelation signals. It gives the additional steps, which have to be made in order to discover and make explicit hidden cancelation in the training sequences. It refers to Algorithm1 as a example reordering step. When such a reordering should be made is controlled in a slightly changed way: the cancelation potential has to be calculated with respect to all training inputs. If one of them has a bad KRS–ratio, the information it provides will not have a positive effect on the training process: it will not contribute to the knowledge acquisition process and will have a symmetrising effect on the influenced network part. Furthermore, if the different inputs have KRS–ratios which differ very much, the ones with a higher ratio will dominate the training process development.

The reorderings of the training set made during training are supposed to break hidden symmetries in the data streams. It is possible, that there is one symmetry breaking process, and afterwards the training goes smoothly to the solution with the required optimality. More than one abrupt change in the error curve, respectively in the weight values, are also possible.

In later stages of learning there could not be a direct indication about the present cancelation by the KRS outcome, since it gives an indication about the current ordering of the entire training set. The difference between the entire training set and the already mapped part of a signal can provide for unreliable learning. This can happen if there is hidden cancelation in the substructures of a signal. Further development of the *KRS* method could be able to give an indication of the risks of secondary cancelation. For now, a plateau of the error function is searched during the training process. Note that there is not a search for the plateau of the error surface, but for a plateau of the error function, which may indicate that the training process is in any kind of stationary area (valley, local minima plateau, etc.).

The separate steps of the windowed sampling method with divided inputs are described in the following algorithm:

Algorithm 2:

[1] **Read the available data $D_n \equiv \{(x^i, y^i)\}_{i=1}^{N}$.**

[2] **Divide (if necessary) the input example stream on two $x^{1}{}_i$ and $x^{2}{}_i$ $D_n \equiv \{(x^{1}{}_i, x^{2}{}_i, y^i)\}_{i=1}^{N}$, so that the KRS−ratio of the output with respect to the both inputs KRS_1 and KRS_2 have increased values.**

[3] **Check for cancelation in the data stream with respect to the both inputs, if the training examples are supplied to the network in a random order. If there is a high cancelation potential with respect to one or to both of the inputs, apply Algorithm1. Otherwise the examples should be supplied to the network in accordance with the current division on windows.**

[4] **Start training with the example sequences, as defined in 3.**

[5] **If during training a plateau of the error function appears, apply *Algorithm1*, until a satisfactory solution is found.**

[6] **End.**

The suggested *Algorithm2* implies three steps which make it different from the passive learning methods: (a) choice of a proper example reordering scheme and sampling interval size, (b) adaptation of the number of selection intervals during the learning phase, and (c) proper division of the input space. The possible drawbacks when applying this algorithm will be discussed and illustrated with the outcome of relevant experiments. Two real–world problems that show the benefits of the algorithm will be discussed in more detail further on. Then the outcome of a series of experiments will be summarized in a table, to show the improved results in terms of average training duration and variance for signals trained with and without the help of *Algorithm2*.

5.2.2.2 Experiments with interval size.

The suggested algorithms give good results for the majority of signals we have experimented with. Some of the results obtained are shown in table 5–1, which give a comparison with the results, obtained with the same signals but trained with the “standard” backpropagation algorithm. To show the advantages of the developed sample reordering methods a few experiments will be discussed in more detail here. Moreover, there are certain risks that applying the sample selection algorithm can bring worse results if used wrongly. In the following the most common cases of faults by using the sample selection algorithm will be described.

First the weight vector development, when changing the number of training intervals incrementally, will be shown for the training signals of figure 3–14a and figure 4–7. Figure 5–7a,b visualizes the corresponding weight changes. The number of network parameters is the same in both cases, but the weight vector development differs structurally due to the different complexity of the training tasks. For the sake of clarity of the results, the recordings are made at every 100 training iterations, until the error level of 0.02 has been reached.

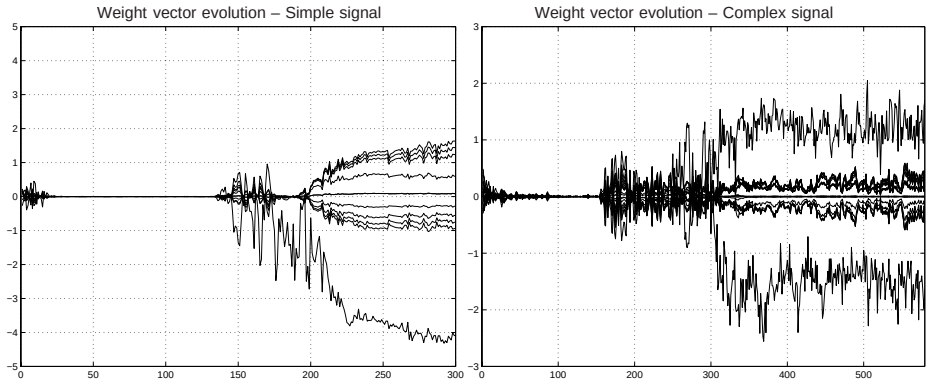


Figure 5–7: *The development of weight vectors when incrementally changing the number of training intervals for the signals from a) Figure 3–14a b) Figure 4–7.*

Training of both signals starts with random selection of the training examples. After about 3 epoches, all the weight values are becoming negligible for both signals. The training error remains constant for a few epoches, which is detected by the training algorithm as a plateau of the error function. This indicates that the sample order should be changed. The division of the original training set on subsets is made according to the RSRI sample ordering scheme. The weight values are brought back from (almost full) degradation to normal for training magnitudes. The higher complexity of the second signal requires longer training. During this extension period, the error plateau has to be broken one more time through a new division on subintervals. For the illustrated case this is made at about 10 epoches (about 250 hundreds of iterations in the plot).

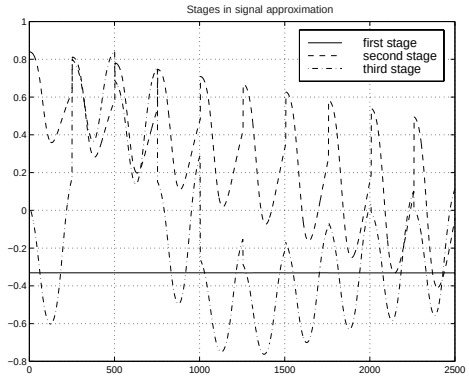


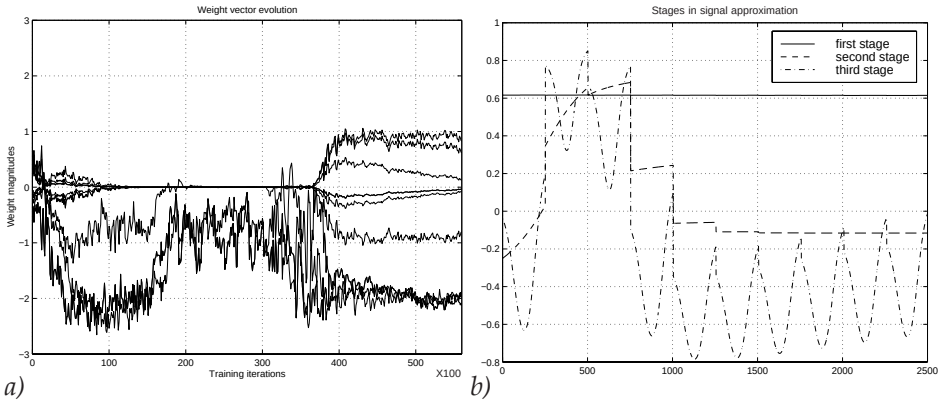
Figure 5–8: *Approximation quality after the corresponding change in the interval size.*

An interesting question is what has already been learned at the moment, when the training set is newly ordered. For the first training set (Figure 4–7) the network output is a straight line before the reordering of the examples takes place. The result of learning the signal from figure 4–7, corresponding to the weight vector development in figure

5–7b, is given at figure 5–8 before the first and the second reorderings of the example set and shortly after that (i.e. 4, 10 and 14 epoches, corresponding to 100, 250 and 350 hundred iterations in the plots). Before the initial symmetrical state is broken, the network response is a straight line. At the following training period, between 180 and 300 hundreds of iterations at the weight vector plot (Figure 5–7b), the network has learned the periodical changes in the signal. The third period of development of the weight vectors corresponds to the dash–dotted plot of the signal. In this stage the network maps the precise position of repeating signal elements.

The KRS–ratio and the development of the KRS–plot predicts when the training signal will have cancelation problems. The current status of using this indicator for learnability of a particular signal is as follows: *If the KRS coefficient is low, a reordering of the training set should be done.*

For many signals the level of this coefficient does not rise very much after various example reorderings, but is high enough to indicate learning without cancelation. Thus, there is a different KRS–value for every signal above which learning success and reliability can be guaranteed. There is still some empiricism in defining this value a priori.



a) *b)*

Figure 5–9: *An example of impropriate example reordering during learning.*

This drawback of the KRS–criterion leaves room for sometimes making a wrong selection of subinterval size (and number respectively). This can lead to secondary training problems. To show the impact on learning progress, a wrong selection of training intervals will be forced, by intentionally choosing a wrong subdivision of the example set. Changing the selection subinterval ranges with a constant increment makes such a learning situation possible.

In figure 5–9a the weight changes are visualized when the number of subintervals after the resampling is equal to the signal period. Within one period, the training sequence has cancelation properties. The neurons and the corresponding weights which have not been degraded before the wrongly chosen interval division comes into use, now show a tendence to unlearn. The division has been made after 5 epoches (150 hundreds iterations in the plot). After the algorithm has detected a new plateau in the training error, a new division of the training set at 10 epoches (300 hundreds iterations) brings finally a successful approximation.

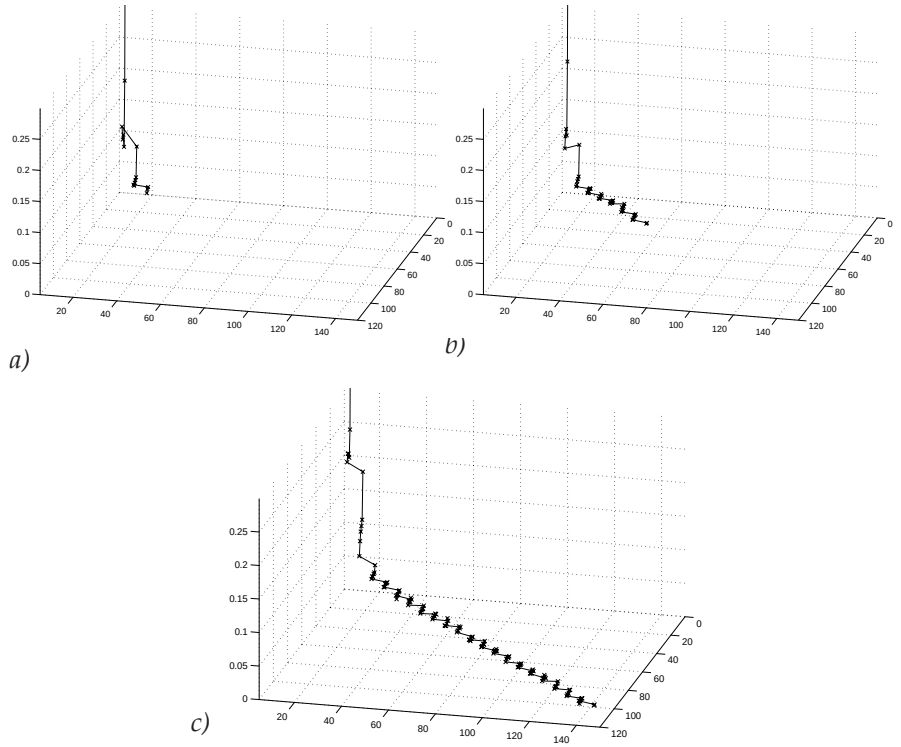


Figure 5–10: *Three most typical error functions by training the signal from figure 4–7.*

Figure 5–10 illustrates the three most probable learning trajectories when training the signal from figure 4–7 with division of the training set in subintervals. The trajectories represent the three most typical developments of the training process when applying Algorithm 2. They show the training error in dependence of the current number of intervals (on the x–axis) and the training epoches on the y–axis. The initial division on training subintervals is made in accordance with the calculated KRS–ratio. Further division on subintervals is initiated when very negligible changes of the error values for a long period of training are detected.

The conclusion from these experiments is, that the biggest descent of the error level happens in the very beginning of the training process. The third graphic shows, that after the first large change the learning curve decreases almost linearly when further divisions on subintervals are made. In this case, it is not clear whether continuously increasing the number of intervals contributes for faster training. Since the neural experiments remain irreplicable, this can hardly be checked. A better approach to show that is through making explicit the relation between the KRS–ratio: the error plateaus and the necessity to change the training intervals.

5.2.2.3 Constructing input streams.

The drawback when selecting a reordering scheme in accordance with the target space have been discussed in general in the previous section. Here special attention will be paid to the construction of example streams. There are a few intuitive principles for doing that: (a) the input stream should not provide information that contradicts the target stream, and (b) there should not be a contradiction between the different input streams, and (c) the input streams should not introduce additional ambiguities. The possible drawbacks of the input space division method can be related to either their interdependence with the subdivision of the target space, or the domination/contradiction of input streams.

As described in the previous section, foreseeing local cancelation and decreased differentiability helps to avoid additional training difficulties, as caused by periodic and general cancelation signals. The corresponding changes in the example streams aim to increase the discriminability of the input space in a way that will not harm the training process. In case of periodic signals, a logical way to do that is by including the periodicity of the signal, which is most often known [9] as an extra input information to the network. Such a reorganization of the input space is expected to improve learning because of the following reasons. Firstly, the division of the input signal on two different streams increases the differentiability and the abruptness of the input/output space and thereby contributes to an early escape from an initial symmetrical phase. As frequently discussed earlier, the initial symmetrical phase, though encountered in every learning problem, is especially ostensible and long when cancelation or low discriminability takes place. Secondly, using a staircase additional input, synchronized with the periodicity of the signal, has always a positive effect on training process itself, because inputting a feature of the signal eases the mapping process a lot.

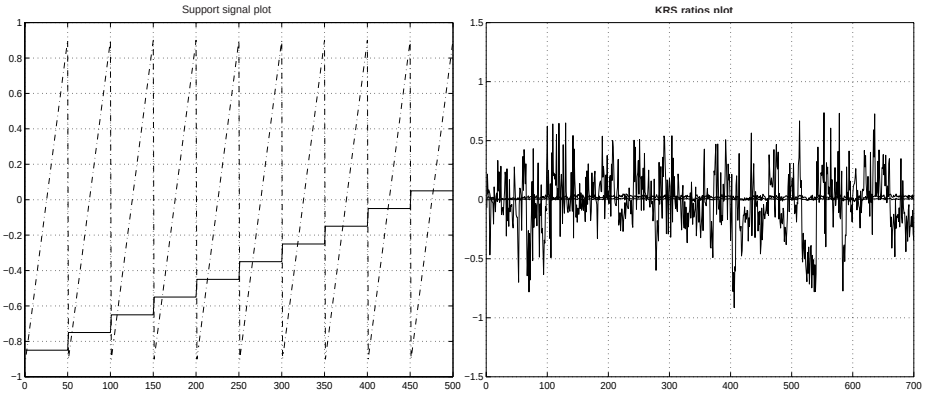


Figure 5–11: *Training a periodic signal plus supporting differentiator.*

These two reasons for increasing the abruptness of the input/output space by dividing the input signal on two streams have general applicability for speeding up learning and ensuring reliable convergence for an arbitrary training signal. Since the two different inputs have their own contribution to the creation of the activation pattern (the internal representation of the problem space) in the network, they will contribute to the symme-

try breaking process separately. Correspondingly two separate KRS–ratios have to be calculated.

Such type of input space division has been done for experiments with various support signals. In figure 5–11, the KRS–ratio development is shown for the periodic signal shown in figure 5–4, supported by a simple x–coordinate input. The plot which is closest to zero represents the resulting KRS coefficient development for an input stream, taking the scaled x–coordinate values. Obviously, such an input is less favorable. A support signal with a staircase appearance (solid line in figure 5–11) does not have a very positive effect on training either, as can be concluded from its KRS–coefficient graphic (the second closest to the x–coordinate plot). The largest impact on the training development has the saw–tooth signal, represented on figure 5–11 with a dash–dotted line and leading to the most obiquitous KRS coefficient development. Further improvements of the structure of the input streams will be shown in later sections.

It should be noted that the contribution of the third signal alone is insufficient for training because it creates similarities in the input stream, i.e. different target examples have the same corresponding inputs. Such a construction of the example streams will create as many ambiguities in the neural system as the cancelation in the target stream of examples does. An example of such an input stream is the one shown in figure 5–11a. To avoid the contradictions that this stream of examples will introduce to the learning system, an additional input that distinguishes this equivalence is needed. Either the x–coordinate of a signal, which differentiates the two targets in time, or an input stream, which adds a level (bias) to every saw tooth can be used as such input information. The staircase input stream, as shown in figure 5–11a, is the simplest variant of such an input.

The specific contribution of input stream division on cancelation prevention is that calculating the KRS–ratio for the input that carries the main signal will predict a very high cancelation potential if the target signal has been windowed to include cancelation structures. We are aware that overcoming the cancelation is a much easier task. A proper division of the training signal soon provides satisfactory results, while a bad ratio between the target subinterval and the input space division can increase the training time substantially and lessen the approximation quality due to the decreased active capacity of the network.

We can easily construct a signal with an improper KRS–ratio by selecting the number of target intervals equal to a period of the signal (and correspondingly synchronized with the changes in the input stream). As an example, see the variations of the weight vector values in figure 5–7b and figure 5–9a. These two pictures show the changes in the weight vector when the wrong interval selection is made either in the beginning of (Figure 5–7b) or during (Figure 5–9a) the training process for the signal in figure 4–7.

For the two experiments the number of selection intervals have been varied incrementally on constant time intervals with a predefined step. The first experiment starts with 1 interval (i.e. the complete training set). Since the training set has very low KRS–ratio at about 3 epoches for both signals, all the weight values quickly become negligible. The original training set is then divided on subintervals and the training is continued by supplying the training examples according to RSRI resampling scheme. The weight values are brought back from (almost full) degradation to a normal for the training magnitudes.

For the second experiment the training is started with the interval size which corresponds to a high KRS–ratio. The initial symmetrical phase is broken in the very early stage and the differentiation of the weights grows. Later on, the number of intervals of the target signal is made equal to number of division intervals of the input space. Since in the separate parts there is a very high cancelation, most of the hidden weight values go to zero and the effective size of the network decreases. During this stage of the training process the network output can map only the already learned global tendency of the signal, as shown at figure 5–9b with the dashed line. During the simulations it has been observed that the approximation quality has been slightly degraded in comparison with the results seen after the initial symmetry breaking. After 15 training epochs (corresponding to 30000 iterations in the plot), the number of intervals was increased again, and the training process has developed until it reaches a satisfactory solution.

A simple way to decrease the interdependence between the input and output streams even more is to select arbitrarily the onset point for every new window from the target signal.

In summary it can be said that in windowing as well as in distributing the information between multiple inputs, the training process risks converging in a very suboptimal solution. A clear indication of a suboptimal solution is weight vector degradation. The risks, when the target set is windowed, are in zeroing the weights because of a wrongly chosen number of training intervals, which increases the cancelation within the interval. Most of the values of the corresponding weight vector during training with a wrongly chosen training interval degrade to zero or a certain fixed value (often the output bias value). This corresponds to unlearning of already learned information. Changing again the number of resampling intervals brings the training process to its normal functioning. Correspondingly, there should be protection within the resampling algorithm to prevent a wrong choice of training intervals.

Up until now, the periodicity of the signal has been used to increase the KRS–ratio of the input/output space. Any other division of the input space has a similar effect. But since the supplied input has a very high impact on the learning success, using information that contradicts the nature of the training problem can be harmful for the training process, since it can increase the non–linearity of the problem. Furthermore, studying such a voluntary division of the input space does not make a lot of sense, since in practice one will rather use the available knowledge for the problem in order to construct an input stream (input feature).

5.3 Two real–life problems.

The suggested algorithm for windowed sample selection was developed on the basis of statistical investigation over the artificially created signals. To show its potentials for solving practical problems, experiments with a real life signals, which include high cancelation, are presented. As discussed before, signals with periodicity, as well as signals whose established manner of sampling provides with highly symmetrical training sets, imply a risk for the reliable learning performance.

The importance of this contribution is illustrated by the observation that training problems can often appear although the task complexity is easy to handle by the chosen neu-

ral structure. Reliability is hereby taken as the ability to learn in a stable, reproducible way. For applications in an industrial setting, such a reliability should ensure real–time, hazard–free behavior. A typical example can be found in the diagnosis of power generators, as discussed next.

5.3.1 Diagnosis of turbo–generator.

The fault–diagnosis of power generators has a long and rich history. As a compound of electrical and mechanical parts resists a sufficiently detailed analytical representation that aids in the subsequent localization and repair the diagnosis is usually performed by very experienced persons. However, these persons tend to become a rare item on the personnel market and the quest for an intelligently system is on.

Destabilization of a turbo–generator by shaft torsion can be estimated during its operational lifetime by vibroacoustic measurements [72]. Spectral analysis shows that every mechanical change in a generator system causes changes in its vibroacoustic frequency spectrum. Peak values in this spectrum can be associated with specific changes in turbogenerator operation. Hence the development of a certain malfunction can be monitored and therefore the moment when it will become fatal can be predicted in a very early phase. However, as only a limited number of faults can be comfortably learnt, the creation of new cases must be constantly nurtured.

The practical significance of a classical vibration–based fault monitor and report AI–system is limited as:

- the amount of on–line information is too large for comfortable handling,
- the count of all probable failures is too high for simultaneous monitoring, and
- only known faults can be classified.

The first of these arguments stresses the need for on–line data processing. The recorded example signals of simulated or real faults have to be learned by the network. The signal, shown in figure 5–12a, is recorded during the emergence working mode of a power generator. In order to discover in one training sessionwhether this signal has the cancelation potential it is first tested by the cancelation criterion. A high degree of cancelation has been discovered, as shown in figure 5–5b. If the training set created by equidistant sampling of this signal is supplied to the network at random, it leads to a poor approximation. A division of the input space on two streams similar to these from figure 5–11 and proportional to the periodicity of the signal provides for a result as shown in figure 5–12c, if the training set is presented at random. The approximation quality does not improve with the time, as illustrated by the weight vector development. The weights have taken two parallel trajectories and are oscillating in small areas, as shown in figure 5–12d. A breaking of this symmetrical phase does not occur when the training set is presented in arbitrary order. The periodicity of the signal is learnedbecause of the domination of the input information, synchronized with the periodicity of the signal. In this case the network output is governed by only 3 hidden neurons, which can successfully approximate the periodicity of the signal.

Although the suggested KRS–ratio indicates when a reliable learning can be expected, it does not provide tangible rules to construct the example streams. For a division of the example stream in accordance with the target signal, the *Algorithm1* uses developed in

the previous chapter sample selection strategies. We have shown that a proper selection of input data streams can increase the learning speed and the reliability enormously. Furthermore, some input streams improve the learning performance better than others. The KRS criterion can verify that the corresponding combination target/input causes no reliability problems. Choices for input streams remain a designers' task, and their decisions are influenced by the accumulated expertise and the availability of relevant information about the specific problem.

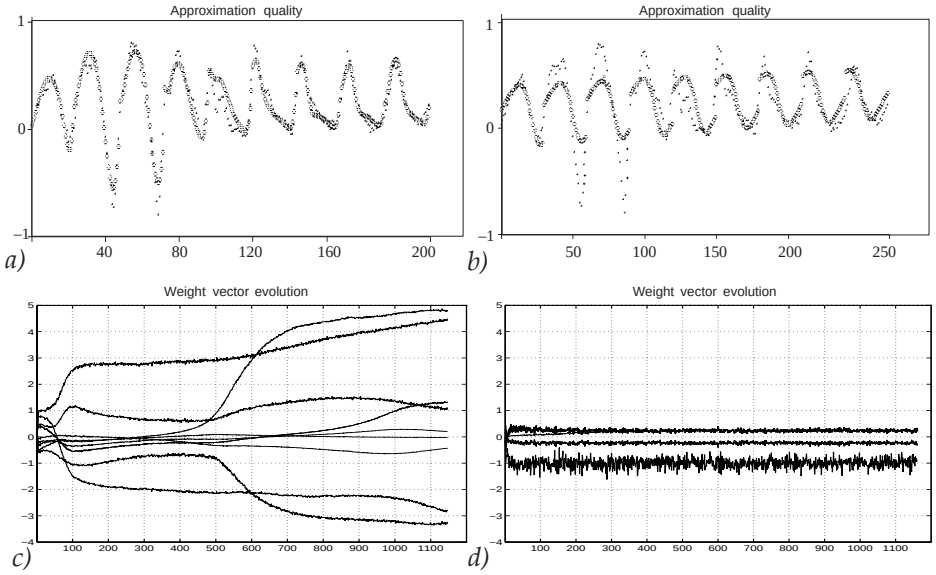


Figure 5–12: *c) weight changes during cancelation–free training, leading to the signal approximation shown in a). d) weight changes during cancelation training leading to the signal approximation shown in b).*

Because of the known periodicity of the power signal, it is not necessary to choose a selection interval size with the KRS–ratio which calculates adaptively for variety of subinterval divisions. The correct selection interval size can be found by dividing the training set into parts defined by its local extrema. The first training sequence is determined by selecting the size of the training intervals to take the range between subsequent odd local extrema. The number of the subintervals and elements within an subinterval are chosen randomly during the training. The training sequence created in this manner showed higher degrees of cancelation, as expected from the fact that the resampling was performed in a wrong manner. The training sequences, created by decreasing the size of the sample selection subintervals show lower and lower degree of cancelation. Figure 5–12c illustrates the weight changes when a non–cancelation training set is extracted. In this case the learning quality is quite satisfactory, as shown at 5–12b.

Further improvements in the learning speed and the replicability of its duration can be achieved by choosing appropriate input streams. An example of input streams for improved reliability is shown in figure 5–13. A typical facet of this application of a neural

network in an industrial environment is the occurrence of new frequency contributions from new failures as well as the shift in existing frequencies because of wear and ageing. For the diagnosis it is required to perform the learning at regular intervals. These on–line requirements make extensive pre–processing impossible, while on the other hand reliability is of utmost importance to guarantee hazard–free operation.

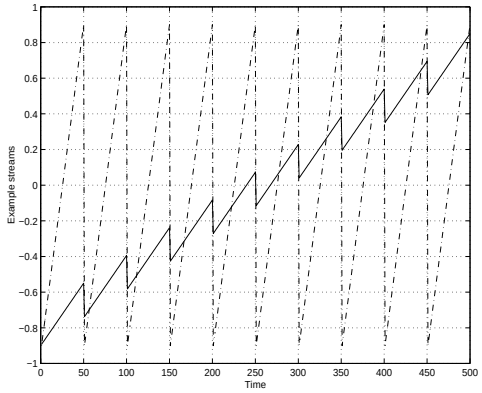


Figure 5–13: *Input streams for improved learning performance.*

5.3.2 QRS detection.

In this section the results with a known medical application will be discussed: the approximation of the heart beat signal as used for *QRS detection*. For the rendering of the problem and for the description of the state–of–the–art, we have used material from [90].

Heart disease is currently the major cause of mortality. More than half of the people with heart disease die within two hours of an attack, that may start as angina pectoris, myocardial infarction, or congestive failure. The most susceptible category is formed by men in the age of 40 to 50 years. A physician could monitor such cases better if equipped with highly integrated medical instrumentation, that should be safe, reliable, accurate, and robust.

The QRS detector represents a building block for critical care electrocardiogram (*ECG*) monitoring and also heart rate monitoring during activities as jogging, aerobics or cycling [73], [23]. The device could improve the reliability of arrhythmia diagnosis and implanted pacemakers monitoring. The assignment is to specify and verify a neural QRS detector on artificial ECG test data representing the normal heart function.

5.3.2.1 Physiological facts about ECG.

The ECG signal originates from the cardiac muscle where electrical impulses are employed to control the mechanical actions of the heart muscle [26], [50]. The heart is a four chambered pump consisting of left and right atria and ventricles. In the resting phase (*diastole*) blood enters through the atria and passively fills ventricles. At the end of the diastole, the atria contract to assist in filling the ventricles. In the active phase (*systole*) blood is pumped by contraction of ventricles to pulmonary artery and aorta.

Mechanical activities of the heart are controlled by two neural centers, namely the sinoatrial node (SA) and the atrioventricular node (AV). The major neural pacing center (SA) generates impulses to initiate systolic contractions and sends them through nerve fibers to the AV node which delays the pulse before sending it to the Bundle of His, then to the Branch Bundles and finally to the Purkinje Fibers which conduct the impulse to all parts of ventricles. SA and AV form a hierarchical system in which the SA node normally assumes control of the beating rate (60–80/minute). In case of SA failure, the AV node assumes control at much slower pace (40–60/minute).

The ECG is composed of three major sections, the P, QRS, and T waves. The P section represents atria muscle depolarization, the QRS complex represents contraction of the ventricular muscles, and the T section represents only an electrical event with no mechanical counter part. Normally, the heart beats 72 times per minute. This rate (not the coordination) changes under special psychophysical conditions (stress, relaxation, sleep, exercise). Slowed and accelerated heart rate are called bradycardia and tachycardia, respectively. During tachycardia, the heart rate can be doubled or even tripled.

The main causes of arrhythmias are malfunctioning of pacemaker cells, blocks in electrical pathways, and abnormal electrical impulse generations from within sections of the cardiac muscle. Resulting conditions are flutter, fibrillation, and heart block. Flutter is a combination of much less coordinated muscle actions than the normal and rapid contractions of heart muscle. An extreme version of flutter is called fibrilation. Atrial fibrillation is far less serious than the ventricular one, since the heart continues to function, but up to 30% less efficiently. Ventricular fibrilation can cause death of heart muscle. Heart block occurs as a consequence of interruption of electrical pathways of the heart causing discoordination of ventricular and atrial rhythms.

5.3.2.2 Artificial sample generation.

In order to obtain learning and test data for the neural QRS detector developed here, an ECG simulator is needed. It consists of a heart simulator and four noise insertion models namely for power line noise, respiration noise, baseline shift, and EMG. All parameters for the simulators mentioned above are assembled in one parameter file, that contains also information on the number of heart beats and the sampling frequency.

The heart signal consists of 9 parts, wherein 5 peaks occur: the p–, q–, r–, s– and t–peak. Most prominent is the r–peak, because of its significant amplitude; therefore the detection of the r–peak frequency is often used for a first characterization of the heart beat.

As shown in 5–14, a single heart beat starts with a flat part (called here “before p”) leading to the p–peak. A flat part is characterized by its duration (or width) and its signal level (or amplitude); a peak is characterized by its duration, its maximum amplitude and the displacement of the peak within the assigned time interval. All these numbers have a typical value and a variance. After the p–peak follows again a flat part, that leads to the q–peak. Immediately thereafter occurs the r–peak and the s–peak. Lastly we find a flat part, the t–peak and a concluding flat part. This basic pattern can be repeated a specified number of times (”number of beats”) and will be discretisized in time by a sampling signal of a specified frequency (”sample frequency”). The alternative specification of “heart beat frequency” should be somewhat in–line with the specification of the heart beat signal parts.

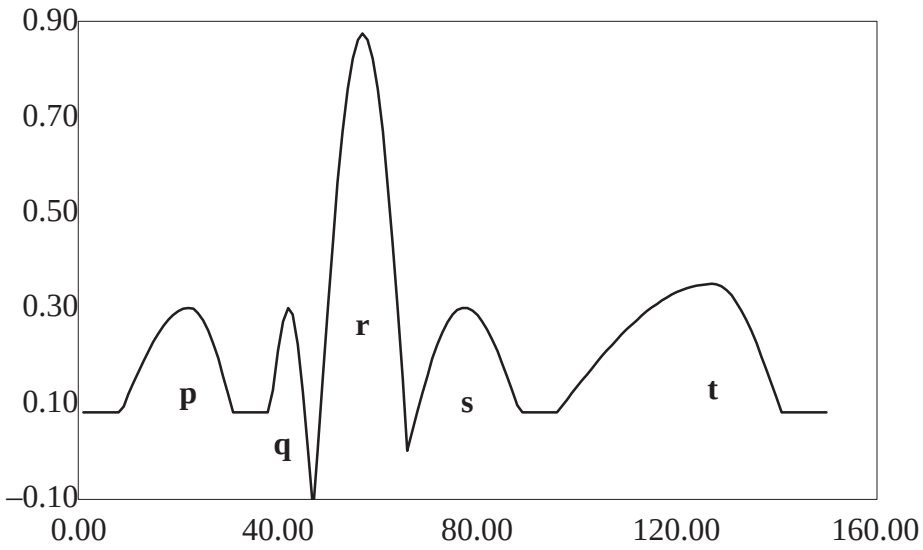


Figure 5–14: *Somewhat beautified rendering of a single heart beat.*

The ECG simulator calculates ECG samples and writes them to an output file together with a phase flag which indicates the current phase (PQRST) of a heart beat. The heart simulator produces P, QRS, and T waves according to a selected heart beat frequency (beats/min) and the shape of the waves. The shape description consists of time interval width [ms], amplitude [mV], and peak displacement from the start of time intervals [ms]. The general shape of a peak is sine–like.

A number of disturbances can be introduced to reflect the practical noisy measurement conditions, such as there are:

- power line noise, producing the influence of external electrical equipment (typ. 50 Hz.);
- respiration noise, producing the effect of other bodily parts in function;
- base line shift, producing the effect of shifting earthing conditions due to sudden moves of the body to sensor attachment;
- EMG noise, producing random effects on the heart signal;

5.3.2.3 Towards a solution.

The classical DSP–oriented approaches for QRS detection are based on signal amplitude measurements and are therefore oversensitive to noise – especially abrupt base line shift and EMG noise [128]. To reduce unwanted effects of false QRS detection i.e. to detecting QRS where no actual QRS exists, several preprocessing techniques are used. In this section, we focus on neural network solutions for ECG interpretation. Main points of interest are preprocessing, network type, inputs, outputs, and data used. Some comments are also included.

In only a few cases of ECG interpretation raw data are used i.e. the data are used without preprocessing. Normally the raw ECG signal is filtered by low and high–pass filters

[151], [161], [82] using signal processors [82] and/or FFT [151]. Some additional data transformation procedures like normalization, centering, base line shift reduction, and R peak detection are also in use. The next step is feature extraction by classical means [162]. Features normally include QRS width, QRS amplitude, QRS offset, T slope, T prematurely [162], power spectral density [151], or 37 standard ECG variables by the HP program (like sex, age, and a set of nonlinear functional transforms of the input parameters) [25].

Many authors use already existing ECG data bases like the American Heart Association Database (AHA) [162], the MIT–BIH arrhythmia database [161], [34], or the University of Leuven ECG database [25], which makes it hard to compare the results. The second source of ECG data are Holter tapes [82], self–sampled data, and/or simulated data including ACG anomalies.

5.3.2.4 Approximation by windowed sample selection.

The QRS signal as described in the previous section poses two types of problems in achieving a successful approximation. First, since it is periodic, learning difficulties due to the periodicity can be observed. Secondly, since various substructures within a single period of the signal show a high level of cancelation, the known ways of representing examples from the input/output space will manifest such additional learning difficulties. Since the learning problem by periodical signals have been elaborated on already, in this section more attention will be spend on overcoming the learning difficulties by general cancelation signals. This is the reason to train one QRS period first.

An ad–hoc learning scenario provides for an unsatisfactory learning performance with both types of networks: zero–centered sigmoid network gives a straight line output after an exhaustive training, since the logistic transfer network shows poor approximation. The outcome of an experiment with the second type of network after about 47000 presentations of the training set (more than sufficient for a signal with such complexity) is shown in figure 5–15.

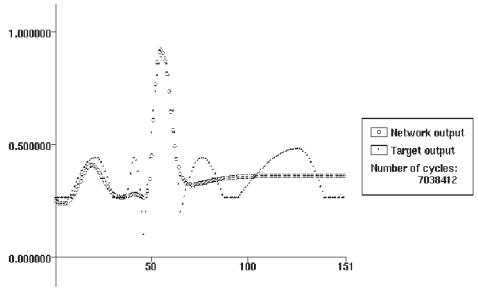


Figure 5–15*: The results by ad–hoc training with logistic network.*

In line with the reasoning made so far in this thesis a division of the target signal on intervals is made as a first attempt to improve the learning quality. A comparable approximation quality to this shown in figure 5–15 is achieved by the windowed sample selection algorithm about 50 times faster, even with a symmetrical network. The learning curves for an experiment outcome with a average performance for logistic sigmoid network, trained with randomized training set and zero–centered sigmoid network

trained with WSS algorithm are shown in figure 5–16. The solid line that shows a dramatic improvement of the training time, corresponds to training, performed with the windowed sample selection algorithm. The learning curve for a zero–centered transfer network trained by means of random example selection is not shown, since such a network has not reached a comparable approximation quality after practically endless training.

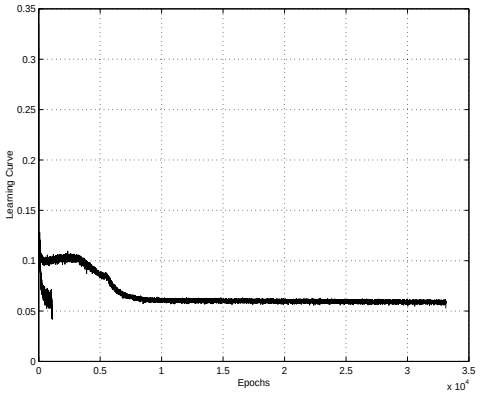


Figure 5–16*: The learning curves for standard backpropagation a) and WSS training b).*

An attempt at further improvements of the training process in terms of approximation quality and speed are made through division of the input stream as an additional step to the windowed sample selection procedure.

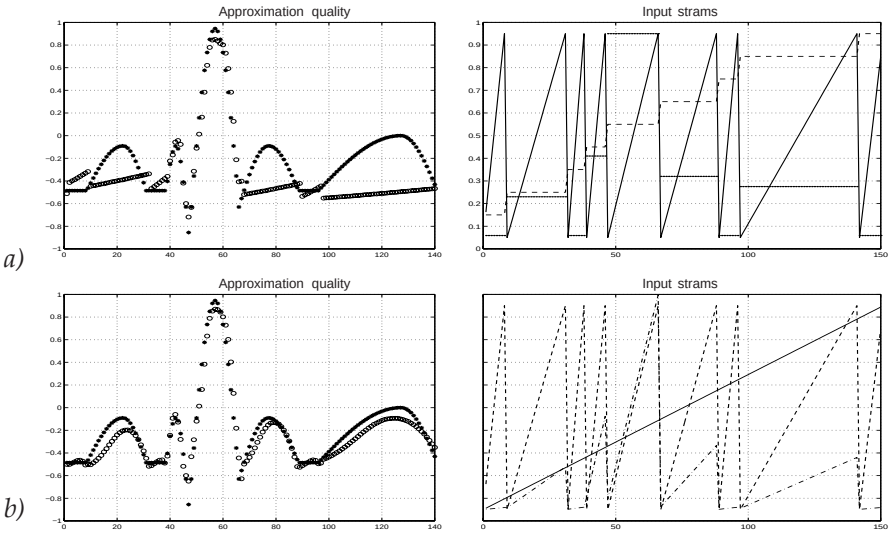


Figure 5–17*: The generalization results (left) with corresponding input sets (right) .*

The division of the input stream can be made in several manners. For instance, the ECG signal from figure 5–17a can be trained by blindly dividing the input space on parts and increasing the discriminability of the input/output space. The training process requires very big amount of training time to find a satisfactory solution, since the input information can contradict the problem features. When recorded, the ECG signals have to be synchronized, i.e. there is time information available. Furthermore, the average duration of the different QRS phases is known. This allows one to use this information for input space division instead of training with blindly determined division of the input space. This feature of the ECG signal is equivalent to the knowledge about the period of the periodical signals. The main difference from a periodical signal is that the input streams are divided into unequal parts by QRS signal.

In periodic signals, risks of introducing an additional symmetry to the learning system are caused by making the existing cancelation structures too explicit: within each subinterval of the input stream there is a corresponding cancelation structure of the output stream. Training either by means of random example selection or by windowing of the target stream, so that the cancelation structures are included in these windows, results in poor approximation. An additional input with a staircase appearance provides for an equal input values to corresponding equal targets. Choosing a subdivision of the target space which includes high cancelation, makes such input entirely useless. Randomized selection will also provide cancelation learning, if not in general, than for learning of a specific structures.

Similarly, unreliable performance is expected if the signal contains structures with high degree of cancelation (as seen in the different pulses of a QRS signal). Such a signal we call a general cancelation signal, since neither symmetry nor periodicity is explicitly present in it (here we referr to a one puls of a QRS signal). A QRS signal sampled in a longer time scale than one period possesses an additional risk for cancelation due to the periodically repeating QRS pulses.

A series of experiments with QRS signals was carried out. These experiments ranged from training by a random selection of examples, through sample ordering with respect to the target stream, and to reordering with respect to both input and output streams. Within one QRS period a danger of unreliable learning exists when the window includes a cancelation structure. However, this condition can be strengthened if one of the inputs accentuates on this feature. In the case of the ECG signal, such a danger can be foreseen, if for an extra input are used the phases of the heart rhythm. Within a few of the phase durations there are obvious conditions for cancelation.

5.4 Discussion.

This chapter gives a practical validation of the developed methodology for reliable training of a MLP network. After the introduction of the neural method, the second chapter gives major directions on how network learnability can be enhanced. The third chapter narrows the problem area by unifying it to a manifestation of indecision in the network operation. The fourth chapter develops the strategy for avoiding learnability and reliability problems in general on the basis of reordering the streams of examples supplied to the network during training. This strategy gives many possibilities for their

realization, which need to be concretisized for practical use. Creating an example reordering strategy on the basis of the developed KRS–criterion leaves a range of opportunities for practical realization. This variety requires for an algorithmic implementation of the developed sampling strategies for problems with different complexity and features, provided in the fifth Chapter.

The concrete realization of the resampling strategy developed varies more with the features of the signal rather than with its complexity. The supporting arguments for this statement are implied in many places in this thesis. The reliability problems are solved with respect to the complexity of the cancelation situation present in the problem. Among the possible ways to cope with simple cancelation cases is to move the window on a signal from which training samples are drawn. The sampling strategy developed suggests an alternative way for handling cancelation signals by dividing the training set on intervals, within which the extracted samples will ensure unproblematic training. This technique gives positive results in a wider range of learning situations. It remedies certain reliability problems cancelation signals of simple shape, but might be insufficient in more complex cases.

Additionally, by division of the training, there are risks for the training process to converge to a suboptimal solution. If the location and the size of windows is wrongly selected, it is likely that the weights will degrade to zero. Consequently, the potential for mapping the dynamics of the internal generator will decrease, even the already learned information can be forgotten. Though changing the window definition brings the training process back to its normal function, the question over what the proper choice should be, remains unanswered. The elaborated KRS–paradigm suggests that there is a high degree of symmetry at the beginning of a learning process and that guiding the learning trajectory for a long time in the same direction will not increase the risk of jumping over a satisfactory minimum. It is advisable that the sizes of windows with non–cancelation examples shrink at later stages of the training.

As periodic signals naturally imply additional risks for degradation phenomena to occur, the approach of handling simple cancelation signals is extended to cope with periodicity as well. Simple techniques such as moving the selection interval of the training examples will not help because periodicity keeps the cancelation content of the examples at a high level.

Increasing the abruptness of the input/output space is a well–known technique in neural sensitivity analysis methods. We create abruptness within the input streams in order to break the symmetrical states. By doing that, the network is forced to learn the variability of the input streams. This implies that any change in the input stream of examples should be made in accordance with the variability of the signal. Furthermore, if additional inputs are introduced, it should be checked whether the information of some input is not dominant over others. Such a check is easily performed on the basis of the KRS criterion.

In general, the construction of input strings of examples is synchronized with a feature of the signal. This way the network task is facilitated considerably. The periodicity of a signal is often known, although the signal itself should be learned from examples [9]. By using the periodicity in the signal as an additional input, its learnability and speed should be improved, since (a) extra information is available for the learning process, and (b) internal symmetries are made explicit, and thus manageable.

The risks posed by splitting the input space are in choosing input sequences with widely differing KRS–ratio (the KRS coefficient with respect to one of the inputs is approximately zero, and the other has a different value), which induce the network to learn the information from the input with the bigger KRS values while suppressing the other. Hence the success or failure of the training process can be directly related to the internal representation.

The comparison of the training performance of the active sampling algorithms created with the backpropagation algorithm in its standard form are shown for several signals, which are plotted in appendix A under the corresponding number. For reliability of the outcomes, trainings with every signal are repeated 100 times. On the basis of these repetitions, the median performance and variance of the repetitions are calculated. Since all the signals have a high cancelation content, their training with equivalent (zero–centered transfer network) did not bring to the desired approximation quality after exhaustive training. These cases are denoted in the table with a dash. For a comparison, the results of ad–hoc training with a logistic sigmoid network are drawn beneath. Calculating the variance of the results, obtained with a logistic sigmoid network is not relevant for the comparison.

Table 5–1: *Median and variance of the training duration of experiments with 100 trials each.*

Approximation Quality	**Windowed Sample Selection**		**Backpropagation**	
	symmetric network		*asymmetric network*	*symmetric network*
Signal No (see Appendix A)	Median [epoches]	Variance	Median [epoches]	Median [epoches]
1	16.7	2.357	195	–
2	47.1	6.4000	125.9	–
3	53.8	1.8763	287.5	635.2
4	398.2	$3.52\ 10^3$	3200	–
5	638.5	$1.31\ 10^6$	7500	–
6	206.8	$4.96\ 10^3$	33000	–

6 Closing remarks.

This final chapter consists of three parts. Firstly, overall basic ideas and observations are summarized and the major conclusions of this work are restated. The contribution of this thesis is listed explicitly next. This summary underlines the necessity for knowledge engineering in neural learning. Furthermore, it makes explicit some interesting parallels between the internal processes, that take place by unreliable training and by on–line learning. In relation to this finding, we are convinced that further studies on this topic might lead to find possible ways of applying neural technology in cases, where learning can not be confined to an off–line phase. The ideas about future research are summarized in the third section of this chapter.

6.1 Justification.

Theoretical investigations in neural methodology have shown very promising results. There are also many successful applications at present. However, the enthusiasm is generally declining. Although the proven theorems state that every signal can be approximated with a two–layer feedforward network, it does not detail constructively how to achieve that. For this reason, some scientists state that a plateau in neural research is reached. This belief is based on the fact that progress in applied neural network research is slower than desired. This is undermining the definite potential to obtain better solutions. We conjecture, that neural technology does have the potential to solve a much wider spectrum of problems. This is demonstrated by a modest attempt to improve the performance of one class of networks.

Neural operation depends on the neural system as well as on the problem to be learned. Since learning tasks with similar complexity can cause varying degrees of difficulty for a particular realization, we have directed our attention to the search for a relation between the task to be learned and neural performance. The first interesting observation is that the learnability of a signal is proportional to the replicability (within borders allowed by randomized computations) of the neural experiment. This is the reason why we have been varying signal structures which show either bad learning or unreliable performance. An attempt for quantization of this relation is made with the definition of the cancelation criterion that predicts the learnability of a signal on basis of the extracted examples. Although this criterion needs some adjustments when applied to different signals (for instance, a reliable relation between the number of training examples and the cancelation ratio has to be found), it generally provides a good indication of the learnability of a signal.

Furthermore, it is shown, that changing the ordering and the structure of the example streams changes the cancelation ratio and improves drastically the learnability and the learning reliability of a signal. This result has driven us to search for an optimal ordering of the example streams in order to improve the reliability for learning particular signals.

So far a simple example reordering technique has manifested the potential to alleviate "difficult" training cases. Additionally, example reorderings have shown that the learning duration and its repeatability can be improved drastically for non–cancelation signals as well. As a consequence, we have largely used the "difficult" cases to illustrate the impact of reordering and leave the overall evaluation to the reader's own experience.

The summary made so far concerns our work on the task level. Analysis of the internal mechanisms of the learning process, that lead to bad learning quality, outlines another line of thought. The interaction between the learning task and the neural system can naturally be expressed by the error landscape paradigm: the multidimensional surface, constructed according to the features of the neural system and the training task, represents the information about the current state of their interaction. It is discussed in the introduction to this thesis, that the neural network adapts itself until it becomes a model of the task to learn. In the error landscape perspective every point on the landscape corresponds to a certain degree of this modeling. The network has become a perfect model of the task when the zero level of the error energy landscape has been reached.

In the error landscape oriented way of thinking, training difficulties are expressed via the dynamics of the system – the trajectory, which the learning process follows on its road of intermediate adaptation to reach the minimal energy state. In practice, optimization is performed under the assumption that the error landscape is locally quadratic – i.e. it can be approximated with up to second–order terms of the Taylor expansion:

$$E(W) = E(W^*) + \nabla E|_{W^*} + \nabla^2 E|_{W^*} \qquad (6\text{–}1)$$

The optimal point W^* can be found when ∇E and $\nabla^2 E$ are known. Once the current adaptation point is located, ∇E and $\nabla^2 E$ are recomputed using local information and the minimum is relocated until the correct one is found. The success of this approach in finding the minimal energy point depends on the shape of the error function. If the quadratic approximation is a fair choice, finding good minima is fast and reliable. In feedforward network training, the error function appears to have a large number of flat areas: valleys and plateaus of different dimensionality. If the network state is drawn in such a flatness, most of the parameters are becoming very small – the Hessian matrix is nearly singular.

We have shown that reordering of the example stream can prevent the training difficulties, that arise from singularities of the Hessian matrix. Additionally, we have concluded that the same kind of example reordering can remedy cancelation problems in both types of difficult surface reliefs: very curvacious and very flat forms. The reason is that during cancelation the impact of the target signal on the learning progress is minimal. Therefore the network state (the learning trajectory) changes only in very small steps. In case of valleys and shallow minima, this leads the optimization to the lowest local energy point. The optimization progresses scarcely on surface areas with a high degree of flatness. In both cases the effective length of consequent steps can be enlarged by forcing them to go in one and the same direction. This way the flatnesses can be traversed faster, and shallow hollows can be jumped over.

Optimal reordering is intimately coupled to signal characteristics. We have pursued a conflict–resolving approach, not fully unlike resampling in statistical model selection [37]. In the reordering algorithm, a criterion as previously formulated and experimen-

tally verified is applied to predict the cancelation potential of an existing example stream. If the potential is too high, the stream will be recursively k–fold divided until the potential has dropped enough.

Signal characteristics, such as periodicity and various cancelation substructures, are often complex cases of unbounded behavior. We have shown that complex cancelation can be alleviated by division of the input example space, and therefore decreasing the order of the input–output correspondence and thus decreasing the level of ambiguity and cancelation in the signal, respectively. Such a spatial division can be computed and/or generic, leading to a *generic feature* definition (Figure 6–1). For a class of signals (or domain of problems), some generic features can be defined that are useful for the differentiation of the specific signal. In a sense, this can be viewed as implied feature extraction and leads to concepts in modular neural network design.

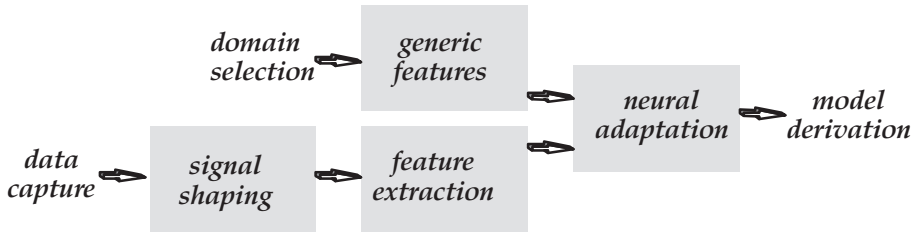


Figure 6–1: *The improved neural processing chain.*

6.2 Contributions of this thesis.

The major contribution of this thesis is the methodology created for the enhancement of the learning reliability and learnability in general. Some aspects are considered: (a) the theoretical analysis of the reasons for unreliable learning; (b) the criterion, that predicts when bad learning will take place; and (c) the example reordering strategies with respect to the input and output example streams and (d) the corresponding algorithms that train cancelation tasks successfully. In the following, we will list a number of such advances:

- [**Reliability estimation**, section 2.3] Among the various ways for neural reliability enhancement we have chosen to analyze and guide the trajectory, that the learning process draws on the multi–dimensional error surface. Correspondingly, the reliability estimation method have to consider a multitude of such trajectories. In order to enhance the quality of an arbitrary learning trajectory, the statistical characteristics of the whole multitude have been considered. The learning reliability is measured by the median and the variance of this multitude. Although this is not a very compact way of measuring the learning reliability, it gives a fair idea about the flow of the training process. Later on, example reordering methods are chosen and optimized further on with respect to the smallest median and variance. It can be observed, that such trajectories correspond to the error rate curve with frequent steep drops.

- [**Symmetry**, section 3.1] The analysis of the possible reasons, causing either bad training outcomes or results that can not be repeated within the desired borders, concludes that the major preconditions for reliability problems are found to be different symmetries in the learning system. Various symmetries in the neural system can cause indecisiveness of the learning process. They result in flatnesses of different dimensionality of the error landscape relief. One–dimensional flatness has the shape of a valley; the higher dimensional flatnesses are imagined as planes. Our contribution is in finding an unified method for fast escaping planes as well as shallow valleys. This unified method suggests that the higher–dimensional flatness should be passed by forcing the algorithm to take steps in one global direction. In the case of shallow valleys and minima, the same technique will cause the sidewalls of the valley to be jumped over. To avoid jumping over satisfactory low minima, which can be met in later stages of the training process, the division on intervals is performed by only decreasing the interval size (or the passages in the same direction).

- [**Neural transfer**, section 3.2] The most important symmetry breaking factors implied in the neural method are the different randomizations and the training task. Additionally, the asymmetrical adaptation factor, which is a property of logistic sigmoid networks, contributes to the symmetry breaking process. The reliability problems due to symmetrical states are more likely to occure if the integral over the product of the nodal transfer and its derivative $\varphi.\varphi'$ equals zero over the range of its activation, as it is true for zero–centered sigmoid networks. Correspondingly, for zero–centered transfer networks, there is a larger potential for unreliable learning performance than for logistic transfer networks, since there is an increase of the symmetry in the neural system.

- [**Internal contradiction**, section 3.3] Cancelation in the neural system appears by a specific aspect of the training task: the training examples cancel each others impact on the learning process. The behavior of cancelation signals in particular (stationary) error surface areas has been analyzed. It has been shown that in such cases the impact of the target signal is negligible. Thus, the neural system is guided by the input examples only, which are values on the range [–1...1] (resp. [0...1]), selected in a random manner. Therefore, (a) the training process in such areas can be described by the laws of random processes, and (b) the learning progress is negligible, since the target signal has no substantial effect on the adaptation, and (c) it causes the weight vectors to degrade: the effective capacity of the network decreases. Cancelation can appear if either the complete training set or the particular ordering of the training examples has cancelation properties.

- [**Cancelation criterion**, section 4.2] When a presentation set may cause problems that will lead to lengthened training or even paralysis, it will be of advantage to predict this to happen. When having sufficient prediction at hand, one may refrain from learning as chances are that this will be to no avail anyhow. The created cancelation criterion gives a way to predict whether the current

training set will suffer from cancelation. Up till now learning problems have been solved either by changing the training algorithm or by restarting the network. Furthermore, the cancelation criterion can be used on–line in the training process to show whether the current sequence of training examples (the training set ordered for the current training iteration, for instance by randomization) will prove to be ineffective (degradative training).

- [**Example order in presentation set**, section 4.3] Pure random selection reveals the degrading nature of cancelation signals. We note that even improved algorithms do not always prevent degradation. We have therefore analyzed different constructions of a presentation set from the same example set. It is shown that even difficult examples can be learned using a "windowed sampling" strategy, where the location of the windows can be on–line changed based on a cancelation criterion.

- [**Complexity & periodicity**, section 5.1] For more complex signals (for instance periodic ones), we find that cancelation effects can appear during later stages of training: when some features of the posed task have already been learned. In such cases reordering of the training set with respect to the target stream of examples is insufficient for inputs in the following situations. First, when samples are with equal absolute x–value but opposite sign or with many close x–values corresponding to equal target values. Second, when the low differentiability of the input stream does not contribute a lot to the symmetry breaking process. Since the impact of the target is minimal on cancelation signals, the input stream(s) should move the neural system from the clearminded (symmetrical) state to a model of the training task.

- [**Generic features**, section 5.2] To handle also complexity and periodicity, pre–knowledge about the composition of the signal can be used. We solve the training problem by adding generic features that allow the overall training to handle the cancelation problems in isolation. An example of such a feature is the period in periodical signals or the phase of a signal, containing cancelation structure in itself. Such features need not be artificial, but may sometimes be created autonomously and therefore still fit the idea of an adaptive model. For instance, the period indication can be created by a lock–in amplifier (which in turn can be neurally implemented).

Based on these contributions, we have suggested an active training algorithm and demonstrate in a number of examples that, where classical training does not succeed or gives rise to training of variable duration, the use of reordering in the presentation set based on prediction of potential cancelation with additional generic features will lead to fast and stable learning. This algorithm is stated here as a condensed notation for the integrated impact of the previously gained insights and created advances. It is clearly not definitive, as still a large number of research issues needs to be resolved.

6.3 Suggestions for future research.

Following the previous lines of thought there are two possibilities for further expansion of this research. First, all the results and observations about the impact of the example streams on learnability and learning reliability suggest that further progress in neural learning has to be found in the direction of knowledge construction and representation. As we claim to have sufficiently uncovered the origin of much that has given neural networks a bad name (notably unpredictably long learning), it makes sense to rise beyond the current insight. On the algorithm level we have shown the effectiveness of creating a good presentation set from the available examples: (a) adapting the window size to the level of cancelation or (b) setting the scope of relevance by adding generic features. Between these extremes, there are many practical opportunities for construction of presentations. For instance, increasing the training speed and correctness can simply be done by training with overlapping windows: the origin of the training subinterval is chosen arbitrarily, and with this origin the new selection subinterval is spanned (Figure 6–2).

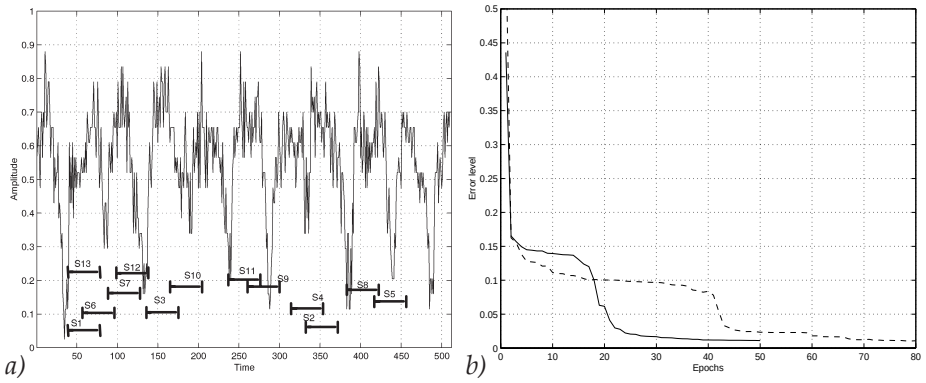


Figure 6–2: *a) The training signal. With the lines S1–S13 are shown the first 13 example selection subintervals. Obviously, they overlap. b) Error decay by training the signal with division of the example space on overlapping (the solid line and non–overlapping (the dashed line) windows.*

Other opportunities are to discard the empirical elements by finding the threshold above which the cancelation ratio will reliably predict good learning performance. Up until now it has been concluded that the exact value, above which the KRS–ratio predicts reliable learning depends on the training signal. More work can be done in defining ways to divide the input streams, so that they can ensure optimal learning of complex signals and not force the neural system to false mappings.

The analysis of the internal mechanisms that guide the learning process suggest another direction for further research. Between the development of the training process by signals with a high level of cancelation and on–line learning scenarios exists a number of similarities. First, the pictures of the parameter vectors for both critical training situa-

tions are similar – a degradation of the effective capacity of the network appears, expressed by nullification/equalization of the most of the network parameters. Second, by improving the training performance by cancelation signals, only the example stream has been manipulated.

Similarly, for on–line training scenarios the only possibility for improving the quality of the learning development is by transformations of the example streams. It can be expected that the developed techniques, which can escape the described training difficulties only by reordering the example stream, will help deal with on–line training information. The similarities in the internal learning mechanisms of those two problem areas suggest, that the techniques for improved reliability of learning by cancelation signals can be used when the training examples are coming on–line from a process and enable the algorithms of backpropagation type to be used for applications where learning can not be confined to an off–line phase.

It comes as no surprise, that the presentation set is the key for reliable learning. An inferior presentation set may often go unnoticed if it is used for testing such networks that are very vulnerable, for instance networks with zero–centered transfer. An optimal presentation set will always work, even with sensitive networks. Having isolated such sensitive arrangements, we are fully equipped to define optimal example sequences for an arbitrary training task.

In a sense, we have used a geometrical interpretation for discussing the effect of learning. Nevertheless, the theory of dynamical systems has not been mentioned here. As the operation of neural networks is based on the exploitation of non–linear functions, chaotic behavior is likely to occur. In fact, the conclusion that without further precautions a neural network will seemingly take different routes towards different results has often been claimed as a sign for such behavior. As we have shown that by proper design such behavior can be eliminated to the benefit of the engineering issues, it seems that this claim is ill–founded.

Still, there is reason for caution. The fact that we can eliminate the appearance of ill–behavior does not imply that chaos is not there. It only shows that we can direct the training path at will. The way in which we accomplish this is by a deliberate division of the input space. For this we regretfully do not have a clear geometrical interpretation. However, it bears resemblance to the way in which chaotic systems tend to be analyzed. This brings us to to a last category of research questions (the most speculative ones): is it possible to formulate the learning process as a dynamic system?

As we have introduced generic features to aid the learning towards a reliable behavior, it seems reasonable to assume that a chaotic description of the training can be based on such general signal characteristics. What we have observed as indecisiveness can also be interpreted as bifurcation. Where we attempt to handle such indecisiveness by adding generic features can then be seen as a pre–selection. Where bifurcation is allowed for, a dynamical system tends to grow into a state of real confusion. This is apparently also true for neural networks. Vice versa, the generic features and/or window definitions can be seen as a labelling of the bifurcations, such that choices in the path selection can be made.

A long reaching goal of future research can therefore be seen as using dynamical systems theory to guide in the selection of useful features, not aimed for providing core

information on the specific signal but rather aimed to give support in the overall learning process.

References.

[1] Albertini, F. and Sontag, E., "For Neural Networks, Function Determines Form", *Neural Networks* **6** , pp. 975–990, 1993.

[2] Allred, L. and Kelly, G.E., "Supervised Learning Techniques for Backpropagation Networks", *Proceedings IJCNN* ***I***, pp. 721–728, 1989.

[3] An, G., "The Effects of Adding Noise During Backpropagation Training on a Generalization Performance", *Neural Computation* **8**, pp. 643–674, 1996.

[4] Annema, A.J., *Analysis, modelling and implementation of analog integrated neural networks*, Ph.–D. thesis (Twente University, The Netherlands) 1994.

[5] Ash, A., and Hedayat, A., "An introduction of design optimality with an overview of the literature", *Communication in statistics, Part A –Theory and methods*, 7, 1295–1325.

[6] Atkinson, A.C., and Donev, A.N., *Optimal Experimental Designs* (Oxford Science publications) 1992.

[7] Atlas, L., Cohn, D., Ladner, R., El–Sharkawi, M. A., Marks II, R. J., Aggoune, M. E., and Park. D. C., "Training Connectionist Networks with Queries and Sampling",pp. 566–573, in: Touretzky, D.S. (Ed.), *Advances in NIPS **2***, (Morgan Kaufmann Publishers, San Mateo) 1990.

[8] Baldi, P. and Hornik, K., "Neural networks and principal component analysis: Learning from examples without local minima", *Neural Networks **2***, pp. 53–58, 1989.

[9] Barakova, E.I., Spaanenburg, L., and Zaprjanov, J., "Neural fault diagnosis of a turbogenerator by vibroacoustic data", *Proceedings ICSPAT'95* (Boston, USA) pp. 1454–1458, 1995.

[10] Barakova, E.I., and Spaanenburg, L., "Selective Sampling for Reliable Neural Signal Approximation", *Proceedings NICROSP'96* (Venice, Italy) pp. 183–193, 1996.

[11] Barakova, E.I., and Spaanenburg, L., "Symmetry: Between Indecision and Equality of Choice", in: J.Mira R. Moreno–Diaz, J. Cabestany (eds.), "Biological and Artificial Computation: From Neuroscience to Technology", *Lecture Notes in Computer Science **1240***, pp. 903–912.

[12] Barber, D., Saad, D., and Sollich, P., "Finite size effects in on–line learning of multilayer neural networks", *Europhysics letters* **34**, pp. 151.

[13] Baum, E.B., and Haussler, D., "What size net gives valid generalization", pp. 81–90 in: Touretzky, D. (Ed.), *Advances in NIPS **1*** (Morgan Kaufmann, San Mateo) 1989.

[14] Baum, E.B., "Neural Algorithms That Learn in Polynomial Time from Examples and Queries", *IEEE Trans. on Neural Networks, pp. 5–19, 1991*.

[15] Baxt, W.G., and White, H., "Bootstrapping confidence intervals for clinical input variable effects in a network trained to identify the presence of acute miocardial infarction", *Neural Computation* **7**, pp. 624–638, 1995.

[16] Belfore, L.A., and Johnson, B.W., "The Fault–Tolerance of Neural Networks", *The international Journal of Neural Networks Research and Applications **1***, pp. 24–41, 1989.

[17] Beyer, U., and Smieja, F., "Learning from examples using reflexive exploration", 17 pages, 1993.

[18] Biehl, M., and Schwartze, H., "Learning by Online Gradient Descent", *Journal of Physics. A: Mathematical and general* **28,** pp. 643–656, 1995.

[19] Bishop, C.M., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[20] Bishop, C.M., "Training with noise is equivalent to Tikhonov Regularization". *Neural Computation* **7**, pp. 108–116, 1995.

[21] Bishop, C.M., and Nabney, I.T., "Modeling Conditional Probability Distributions for Periodic Variables", 1996.

[22] Blum, E.K., "Approximation of Boolean Functions by Sigmoidal Networks: Part I: XOR and Other Two–Variable Functions", *Neural Computation* **1**, pp. 532–540, 1989.

[23] Börger, H.H., *EKG–Information* (Steinkopff Verlag, Darmstadt) 1986.

[24] Bolt, G.R., *Fault Tolerance in Neural Networks*, Ph.D. Thesis (University of York, U.K.) 1992.

[25] Bortolan, G. e.a., "Design of Neural Networks for Classification of Electrocardiographic Signals", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society **12***, No. 3 (1990).

[26] Conover, M.B., *Pocket Nurse Guide to Electrocardiography* (The C.V. Mosby Company, USA) 1986.

[27] Breiman, L., "Bagging predictor". *Technical Report **TR–421*** (University of California at Berkeley, USA) 1994.

[28] terBrugge, M.H., Nijhuis, J.A.G., and Spaanenburg, L., "License–Plate Recognition", in: Jain, L.C. and Lazzarini (Eds.) *Intelligent Techniques in Character Recognition: Practical Applications* (CRC Press) 1998.

[29] terBrugge, M.H., Nijhuis, J.A.G., Jansen, W.J., Drenth, H., and Spaanenburg, L., "On the representation of data for optimal learning", *Proceedings ICNN'95* ***VI*** (Perth, Western Australia) pp. 3180–3184, 1995.

[30] Cachin, C., "Pedagogical pattern selection strategies", *Neural Networks* ***7***, No. 1, pp. 175–181, 1994.

[31] Calabro, S.R., *Reliability Principles and Practices* (McGraw–Hill Book Company, Inc.) 1962.

[32] Carpenter, G.A., Grossberg, S., "Learning, Categorization, Rule Formation, and Prediction by Fuzzy Neural Networks," in Chen, C.H., ed. *Fuzzy Logic and Neural Network Handbook*, NY: McGraw–Hill, pp. 1.3–1.45, 1996.

[33] Carter, M.J., Rudolph, F. and Nucci, A., "Operational Fault Tolerance of CMAC Networks", pp. 340–347, in: Touretzky, D.S. (Ed.), *Advances in NIPS* ***2*** (Morgan Kaufmann, San Mateo) 1990.

[34] Casaleggio, A. e.a., "Neural Network for Automatic Anomalous QRS Complex Detection", *Computer in Cardiology* (Chicago, USA) 1990.

[35] Chen, A.M., Lu, H., and Hecht–Nielsen, R., "On the geometry of feedforward neural networks error surfaces", *Neural Computation* ***5***, No. 6, pp. 910–927, 1993.

[36] Chen, J.R., and Mars, P., "Stepsize variation for accelerating the backpropagation algorithm", *Proceedings of Int. Joint. Conf. on Neural Networks* ***I*** (Washington D. C., USA) pp. 673–678, 1990.

[37] Cherkassky, V., and Mulier, F., *Learning from data* (John Wiley) 1998.

[38] Cheung, R. K., Lustig, I. and Kornhauser, A.L., "Relative effectiveness of training set patterns for backpropagation", *Proceedings of Int. Joint. Conf. on Neural Networks* ***I*** (San Diego, USA) pp. 601–604, 1990.

[39] Cloete, I., and Ludik, J., "Increased complexity training" , in: J. Mira, J. Cabestany and A. Prieto (eds.), Proceedings IWANN'93, *Lecture Notes in Computer Science* ***686***, 1993.

[40] Cloete, I., and Ludik, J.. "Incremental training strategies", *Proceedings ICANN* (Sorrento, Italy).

[41] Cohn, D., and Tesauro, G.. "How Tight are the Vapnik Chervonenkis Bounds?" *Neural Computation* **4**, No. 2, pp. 249–269, 1992.

[42] Cohn, D.A., “Neural network exploration using optimal experiment design”, *Neural Networks* ***9***, No. 6, pp. 1071–1083, 1996.

[43] Cohn, D.A., ”Queries and exploration using optimal experiment design,” in: Cowan, J. et al. (Eds.), *Advances in NIPS* ***6*** (Morgan Kaufmann, San Fransisco) 1994.

[44] Cohn, D.A., Ghahramani, Z., and Jordan, M.I., “Active learning with statistical models”, *Journal of Artificial Intelligence Research* ***4***, pp. 129–145, 1996.

[45] Cotrell, G.W., and Tsung, F.S.. “Learning Simple Arithmetic Procedures”, in: Barnden, J.A., Pollack, J.B. (eds.), “High–Level Connectionist Models”, *Advances in Connectionist and Neural Computation Theory* ***1***, 1991.

[46] leCun, Y., “Generalization and network design strategies”, *Proceedings of Connectionism in Perspective* (North Holland, Amsterdam) 1989.

[47] Darken, C. and Moody, J., “Note on learning rate schedules for stochastic optimization”, pp. 832–838, in: Lippmann, R.P. et al. (Eds.), *Advances in NIPS* ***3***, (Morgan Kaufmann Publishers, San Mateo) 1991.

[48] Deco, G., and Obradovic, D., *An information–theoretic approach to neural computing* (Springer Verlag, Germany) 1996.

[49] Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jakel, L., and Hopfield, J., “Large Automatic Learning Rule Extraction and Generalization”, *Complex Systems* **1,** 1987.

[50] Dubin, M.D., *Rapid Interpretation of EKG's* (Cover Publishing Company, Tampa, U.S.A.) 1989.

[51] Dudar, G., and Rose, K., “The Effect of Quantization on Multilayer Neural Networks”, *IEEE Transactions on Neural Networks* ***6***, No.6, 1995.

[52] Eduards, P.J., and Murray, A.F., *Analogue Imprecision in MLP Training* (World Scientific Publishing, Singapore) 1996.

[53] Efron, B. and Tibshirani, R., *An Introduction to the Bootstrap* (Chapman and Hall, New York) 1993.

[54] Elman, J.L., “Finding Structure in Time”, *Cognitive Science*, pp. 179–211, 1990.

[55] Elman, J.L., “Learning and development: the importance of starting small”. *Cognition*, pp. 71–99, 1993.

[56] Fahlman, S.E., and Lebiere, C., “The Cascade–Correlation Learning Architecture”, *Technical Report* ***CMU–CS–90–100***, (Carnegie–Mellon University, USA) 1991.

[57] Feller, W., *An introduction to probability theory and its applications* (John Wiley) 1968.

[58] Fletcher, R., *Practical Methods of optimization* (John Wiley & Sons) 1987.

[59] French, R. "Dynamically constraining Connectionists Networks to Produce Distributed, Ortogonal Representations to Reduce Catastrophic Forgetting", *Proceedings 16th Annual Cognitive Sciences Conference **5***, pp 207–220, 1984.

[60] Gardner, E., "The Space of Interactions in Neural Networks Models", *Journal of physics A: Mathematical and general* **21**, pp. 257–270, 1988.

[61] Gardner, E., and Derida, B., *Journal of physics A: Mathematical and general* **22**, No. 12, pp. 1983, 1989.

[62] Geman, S., Bienenstock, E., and Doursat, R., "Neural Networks and the Bias/Variance Dilemma", *Neural Computation* **4**, pp. 1–58, 1992.

[63] Geva, S. and Sitte, J., "A cartpole experiment benchmark for trainable controllers", *IEEE Control Systems*, pp. 40–51, 1993.

[64] Girosi, F., Jones, M., and Poggio, T., ''Regularization Theory and Neural Networks Architectures'', *Neural Computation **7***, pp. 219–269, 1995.

[65] Goutte, C., "Behaviour in 0 of the neural networks training cost", *Neural Processing Letters*, to appear in 1998.

[66] Goerick, C., and vonSeelen, W., "On Unlearnable Problems Or A Model for Premature Saturation in Backpropagation Learning", *Proceedings EANN'96*, 1996.

[67] Grandvalet, Y., and Canu, S., "Comments on Noise Injection into Inputs in Back Propagation Learning", *IEEE Transactions on Systems, Man and Cybernetics **25***, No. 4, 1995.

[68] Guo, T.–H., and Nurre, J., "Sensor failure detection and recovery by neural networks", *Proceedings IJCNN **I***, pp. 221–226, 1991.

[69] Haffner, P., Waibel, A., Sawai, H., and Shikano, K., "Fast Back–Propagation Learning Methods for Neural Networks in Speech", *ATR Technical Report **TR–I–0058***, 1988.

[70] Hafner, S., "Einsatz von Künstlichen Neuronalen Netzen zur Signalverarbeitung im Kraftfahrzeug am Beispiel spezifischer Motorsteuerungsprobleme", *Fortschritt–Berichte VDI **12*** , No. 349 (VDI Verlag, Frankfurt, Germany) 1998.

[71] Hamey, L.G.C., "Results on Weight Configurations that are not Local Minima in Feed–Forward Neural Networks", *Proceedings 7th Australian Conf. on Artificial Neural Networks.*

[72] Hammons, T.G., "Impact on Shaft Torsionals in Steam Turbine Control", *IEEE Transactions on Energy Conversion* ***4***, No. 2, pp. 143–149 (1989).

[73] Hampton, J.R., *EKG–leicht gemacht* (Jungjohann Verlagsgesellschaft, Stuttgart) 1991.

[74] Haykin, S., *Neural Networks: a comprehensive foundation* (MacMillan) 1994.

[75] Heisterman, J., *Genetische Algorithmen*, Teubner–texte zur Informatik **9** (Teubner Verlag, Germany) 1994.

[76] Heskes, T.M., and Kappen, B., *Physical Review A* **44**, pp. 2718.

[77] Hinton, G.E., "Learning Distributed Representations of Concepts", *Proc. 8ht Annual Conf. Cogn. Sci.Society* (Amherst, USA) pp. 1–12, 1986.

[78] Holmstrom, L., and Koistinen, P., "Using adaptive noise in backpropagation training", *IEEE Transactions on Neural Networks* ***3***, pp. 24–28, 1992.

[79] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators", *Neural Networks* **2**, pp. 359–366, 1989.

[80] Hwang, J.N., Choi, J.J., Oh, S., Marks II, R.J., "Query–Based learning applied to Partially Trained Multilayer Perceptrons", *IEEE Transactions on Neural Networks* ***2***, pp. 131–136, 1991.

[81] Hush, D.R., Horne, B., and Sales, J.M., "Error surfaces for multilayer perceptron", *IEEE Transactions on Systems, Man and Cybernetics* ***22***, No. 5, pp. 1152–1161, 1992.

[82] Iwata, A. e.a., "A Fast Analyzing System For Holter Recording Using Digital Signal Processores And Neural Networks", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society* ***12***, No. 2, 1990.

[83] Jacobs, R.A., "Initial Experiments On Constructing Domains of Expertise and Hierarchies In Connectionist Systems", *Proceedings Connectionist Models Summer School*, pp. 144–153, 1988.

[84] Jenkins, R.E., and Yuhas, B.P., "A simplified neural network solution through problem decomposition: The case of the Truck Backer–Upper", *IEEE Transactions on Neural Networks* ***4***, No. 4, pp. 718–720, 1993.

[85] Jordan, F., and Clement, G., "Using the Symmetries of Multilayered Network To Reduce the Weight Space", *IEEE ICNN* **2**, pp. 391–396, 1991.

[86] Keegstra, H., Jansen, W.J., Nijhuis, J.A.G., Spaanenburg, L., Stevens, J.H., and Udding, J.T., "Exploiting Network Redundancy for Lowest–Cost Neural Network Realizations", *Proceedings ICNN'96* (Washington D.C., USA) pp. 951–955, 1996.

[87] Keene, S., Musa, J.D., and Keler, T.W., *Developing Reliable Software in the Shortest Time*, Video Tutorial Course (IEEE Press) 1995.

[88] Kemslay, D.H., and Martinez, T.R., "A survey of neural network research and fielded applications", *Int. Journal of Neural Networks **2***, No. 2/3/4, pp. 123–133, 1992.

[89] Kindermann, J., Paass, G., and Weber, F., "Query construction for neural networks using the bootstrap", *Technical Report* (German National Research Centre for Information Technology, Bonn, Germany) 1995.

[90] Kistner, A., Klofutar, A., and Beutelschieℬ F. "Neuronale Netze in der Signalverarbeitung", in S. Hafner (ed.) *Neuronale Netze in der Automatisierungstechnik* (R. Oldenbourg Verlag Munchen Wien) 1994.

[91] Kolen, F., and Pollack, J.B., "Back Propagation is Sensitive to Initial Conditions", in: Touretzky, D. (ed.) *Advances in Neural Information Processing Systems" **4*** (Morgan Kaufmann, San Francisco, CA) 1992.

[92] Krap, R., "An introduction to randomized algorithms", *Discrete Applied Mathematics* **34,** pp 165–201, 1991.

[93] Krogh, A., and J. Vedelsby, J., "Neural network ensembles, cross validation, and active learning", pp. 231–238, in: Tesauro, G. et al. (Eds.), *Advances in NIPS **7*** (MIT Press, Cambridge) 1995.

[94] Kurkova, V., and Kainen, P., "Functionally Equivalent Feedforward Neural Networks", *Neural Computation **6***, No. 3, pp. 553–558.

[95] Kwon, O.–J., and Bang, S.–Y., "Comments on 'The effects of quantization on multilayer neural networks' ", *IEEE Transactions on Neural Networks **9***, No. 4, pp. 718–719, 1998.

[96] Lee, B.W., and Kim, S.W., "Required dynamic range and accuracy of electronic synapses for character recognition applications", *Proceedings IEEE ISCAS*, pp. 1545–1548, 1992.

[97] Lee, Y., Oh, S.H., and Kim, M.W., "An analysis of premature saturation in backpropagation learning", *Neural Networks **6***, pp. 719–728, 1993.

[98] Lisboa, P.J.G. and Perantonis,S.J., "Complete solution of the local minimain the XOR problem", *Neural Systems **2***, pp. 119–124, 1991.

[99] MacKay, D.J.C., "Information–Based Objective Functions for Active Data Selection". *Neural Computation **4***, pp. 590–604, May 1992.

[100] Marr D., *Vision* (W.H.Freeman, New York) 1982.

[101] Masters, T. *Practical Neural Network Recipes in C++*, San Diego: Academic Press (1994).

[102] Matsuoka, K., "Noise injection into inputs in Backpropagation learning", *IEEE Transactions on Systems, Man and Cybernetics* **22**, nr. 3, 1992.

[103] Meijer, P.B.L., *Neural network applications in device and subcircuit modelling for circuit simulation*, Ph.–D. thesis (Eindhoven University, The Netherlands) 1996.

[104] Minski, M., and Papert, S., *Perceptrons* (MIT Press, Cambridge) 1988.

[105] MIT, commercial flyer (Aachen, Germany) 1997.

[106] Morgan, N., and Boulard, H., "Generalization and parameter estimation in feedforward neural nets: Some experiments", pp. 630–637, in: Touretzky, D.S. (Ed.), *Advances in Neural Information Processing Systems **2*** (Morgan Kaufmann, San Mateo) 1990.

[107] Mukherjee, S., *Neural Network Training Algorithms Based on Quadratic Error Surface Models*, Ph.D. Thesis (Cornell University, USA) 1997.

[108] Munro, P.W., "Repeat Until Bored: A pattern selection strategy", pp. 1001–1008, 1992.

[109] Musa, J.D., Iannino, A., and Okumoto, K., *Software Reliability* (Mc. Graw–Hill, USA) 1993.

[110] Neußer, S. Nijhuis, J.A.G., Spaanenburg, L., Höfflinger, B., Franke, U., and Fritz, H., "Neurocontrol for lateral vehicle guidance", *IEEE Micro **13***, No.1, pp. 57 – 66, 1993.

[111] Nguyen, D., and Widrow, B., "The truck backer–upper: An example of self–learning in neural networks", *Proceedings IJCNN* **II**, pp. 357–363, 1989.

[112] Nijhuis, J.A.G., *An engineering approach to neural system design*, Ph.–D. thesis (Catholic University of Nijmegen, The Netherlands) 1992.

[113] Opitz, R., personal communication 1997.

[114] Phatak, D.S. and Koren I., "Complete and Partial Fault Tolerance of Feedforward Neural Nets", *IEEE Trans. on Neural Networks **2***, No. 6, pp. 446–456, 1995.

[115] Paass, G. and Kindermann, J., "Bayesian query construction for neural network models", pp. 441–450, in: Tesauro, G. et al. (Eds.), *Advances in NIPS **7*** (MIT Press, Cambridge) 1995.

[116] Partridge, D., and Yates, W.B., "Engineering Multiversion Neural Net System", 1995.

[117] Partridge, D., "Network Generalization Differences Quantified", *Neural Networks **2***, No. 9, pp. 263–271, 1996.

[118] Perrone, M.P., "General Averaging Results for Convex Optimization", pp. 364–371, in: Mozer, M.C., et al. (Eds.), *Proceedings Connectionists Models Summer School,* (Lawrence Erlbaum) 1993.

[119] Perrone, M.P. and Cooper, L.N., "When networks disagree: ensemble methods with hybrid neural networks", pp. 126–142, in: Mammone, R.J. (Ed.), *Artificial Neural Networks for Speech and Vision* (Chapman & Hall, U.K.) 1993.

[120] Peterson, C., "Mean field theory neural networks for feature recognition, content addressable memory and optimization", *Connectionists Science* **3**, pp. 3–33, 1991.

[121] Plutowski, M., and White, H., "Selecting concise Training Sets from Clean Data", *IEEE Trans. on Neural Networks* **4**, No. 2, March 1993.

[122] Protzel, P. W., Palumbo, D.L., and Arras, M.K., "Performance and Fault–Tolerance of Neural Networks for Optimization", *IEEE Trans. on Neural Networks **4***, No. 4, July 1993.

[123] Poggio, T. and Girosi, F., "Networks for approximation and learning", *Proceedings of the IEEE* **78**, pp. 1481 –1497, 1990.

[124] RayChaudhuri, T., and Hamey, L.G.C., "An algorithm for active data collection for learning Feasibility study with neural networks", *Technical Report 95–173C* (Macquarie University, Department of Computing) 1995.

[125] Reed, R., Oh, S., and Marks, R.J., "Regularization using juttered training data.", *Proceedings Int. Joint Conf. Neural Networks **III*** (Baltimore) pp. 147, 1992.

[126] Riedmiller, M., "Advanced supervised learning in multi–layer perceptrons — from back–propagation to adaptive learning algorithms", *Int. Journal of Computer Standards and Interfaces **5***, 1994.

[127] Riegler, P., and Biehl, M., "On–line backpropagation in two–layered neural networks", *Journal of Physics. A: Mathematical and general* **28,** pp. 507–513, 1995.

[128] Roy, S.C., "A Testable CMOS QRS Detector and Arrhythmia Monitor", *Technical Report TR90–45* (MCNC, USA) 1990.

[129] Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation", in: Rumelhart, D.E., and Wil-

liams, R.J. (Eds.), *Parallel Distributed Processing **1***, Ch. 8 (MIT Press, Cambridge) 1986.

[130] Saad, D., and Solla, S.A., "On line learning in soft committee machines", *Physical Review E* **52**, nr. 4, pp. 4225–4243.

[131] Saad, D., "Explicit Symmetries and the Capacity of Multilayer Neural Networks'', *Journal Physics A **27***, pp. 2719–2734, 1994.

[132] Sang, E.F.T.K., *Machine Learning of Phonotactics*, Ph.–D. thesis (Groningen University, The Netherlands) 1998.

[133] Schonewille, B., "Collision Avoidance: an application of autonomous neural control", *IMS Technical Report* (Instutut für Mikroelektronik Stuttgart) 1992.

[134] Schraudilph, N.N., "On centering Neural Networks Weight Updates", in Müller, Orr, and Caruana (Eds.), *Tricks of the Trade: How to Make Neural Networks Really Work (working title)*, Lecture Notes in Computer Science, Springer Verlag, Berlin 1998.

[135] Scott–Cardell, N., "Why Some Feedforward Networks Cannot Learn Some Polynomials", *Neural Computation*, **6,** pp. 761–766, 1994.

[136] Sejnowski, T.J., Kienker, P.K., and Hinton, G.E., "Learning symmetry groups with hidden units: Beyond the perceptron", *Physica* **22D**, pp. 260–275, 1986.

[137] Segee, B.E. and Carter, M.J., "Comparative Fault Tolerance of Parallel Distributed Processing Networks", *IEEE Transactions on Computers **43***, No. 11, 1994.

[138] Sharkey, A.J.C. and Sharkey, N.E., "How to improve the Reliability of Artificial Neural Networks", 1996**.**

[139] Shawe–Taylor, J., "Symmetries and discriminability in feedforward network architectures", *IEEE Transactions on Neural Networks* **4**, No. 5, pp. 816–826, 1993.

[140] Siggelkow, A., Nijhuis, J.A.G., Neußer, S., and Spaanenburg, L., "Influence of hardware characteristics on the performance of a neural system", pp. 697 – 703, in: Kohonen, T. et al. (Eds.), *Artificial Neural Networks* **1** (North–Holland, Netherlands) 1991.

[141] Sollich, P., and Sadd, D., "Learning from queries for maximum information gain in imperfectly learnable problems", pp. 287–294, in: Tesauro, G. et al. (Eds.), *Advances in NIPS **7*** (MIT Press, Cambridge) 1995.

[142] Sollich, P., "Query construction, entropy, and generalisation in neural network models", *Physical Review E **49***, pp. 4637–4651, 1994.

[143] Spaanenburg, L., deGraaf, J., Nijhuis, J.A.G., Stevens, J.H., and Wichers, W., *Neural understanding of low–resolution images*, pp. 237–248, in: (Hafner, S., Kiendl, H., and Schwefel, H.–P.) *Computational Intelligence: Neuronale Netze, Evolutionäre Algorithmen u. Fuzzy Control im industriellen Einsatz*, VDI Berichte **1381** (VDI Verlag, Düsseldorf) 1998.

[144] Spaanenburg, L., Ter Haseborg, H., and Nijhuis, J.A.G., *Fuzzy diagnosis of float–glass production furnace*, pp. 197–206, in: (Reusch, B.) *Computational Intelligence: Theory and Applications*, Lecture Notes in Computer Science **1226** (Springer Verlag, Berlin) 1997.

[145] Spaanenburg, L., Murray, A.F., vanSchaik, F.A., Tryba, V., and Verleysen, M., "Analog Synapse Memory Technology", *Deliverable DR2–F3 of Basic Research Action no. 3049 (Nerves)*, June 1991.

[146] Strand, E.M., and Jones, W.T., "An active Pattern Set Strategy for EnhancingGeneralization While Improving Backpropagation Training Efficiency", *Proceedings IJCNN **I*** (Baltimore, USA) pp. 830–834, 1992.

[147] Sussmann, H.J., "Uniqueness of the weights for minimal feedforward nets with a given i/o map", *Neural networks* **5**, pp. 589–593, 1992.

[148] Thimm, G., and Fiesler, E., "High Order and Multilayer Perceptron Initialization", Accepted for publication by *IEEE Transactions on Neural Networks*, 1996.

[149] Tibshirani, R., "A comparison of some error estimates for neural network models", *Technical Report* (University of Toronto, Canada) 1995.

[150] Tikhonov, A.N., "On solving incorrectly posed problems and method of regularization", *Doclady Akademii Nauk USSR*, **151**, pp. 501–504, 1963.

[151] Tsai, Y.S. et al., "An Experiment on ECG classification using Back–Propagation Neural Network", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society **12***, No 3 (1990).

[152] Vapnik, V., "Principle of risk minimization for learning theory", *Advances in Neural Computing*, 1992.

[153] Warkovski, F. , Leenstra, J., Nijhuis, J. and Spaanenburg, L., "Issues in the Test of artificial Neural Networks", *Digest ICCD'89*, pp. 487–490, 1989.

[154] Watkin, T.L., Rau, A., and Biehl, M., "The Statistical Mechanics of Learning a Rule", *Review of modern Physics* **6**, No. 2, 1993.

[155] Weaver, S., Baird, L., and Polycarpou, M.M., "An Analytical Framework for Local Feedforward Networks", *IEEE Transactions on Neural Networks* **9**, No. 3, pp. 473–482, 1998.

[156] Weigend, A., Rumelhart, D., and Huberman, B., "Generalization by Weight–Elimination with Application to Forecasting", pp. 875–882, in: Lippmann, J., and Touretsky, D.S. (Eds.), *Advances in NIPS* **3** (Morgan Kaufmann) 1991.

[157] West, A., Saad, D., and Nabney, I., "The learning dynamics of a universal approximator", pp. 288–294, in: Mozer, M.C. et al. (Eds.), *Advances in NIPS **9*** (MIT Press, Cambridge) 1997.

[158] Weyl, H., *Symmetry* (Princeton University Press) 1952.

[159] Wiegerinck, W., and Heskes, T., "How dependencies between successive examples affect on–line learning", to appear in *Neural Computation*, 1996.

[160] Wiegerinck, W., and Heskes, T., "On–line learning with time–correlated patterns," *Europhysics Letters* **28**, pp. 451—455, 1994.

[161] Xue, Q., e.a., "Training of ECG Signals in Neural Network Pattern Recognition", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society **12***, No. 3, 1990.

[162] Yeap, T.H., e.a., "ECG Beat Classification by a Neural Network", *Annual International Conference of the IEEE Engineering in Medicine and Biology Society **12***, No. 3, 1990.

[163] Zhang, B.T., "Accelerated learning by active example selection", *International Journal of Neural Systems **5***, No. 1, pp. 67–75, 1994.

List of Tables.

List of Figures.

List of Symbols.

e	local error
f	network function
E	value of RMS error criterion
H	number of hidden neurons
N	number of examples
o	output of hidden neuron
w_{ij}	weight for signal from j to i
x	input vector
x^i	input vector in example i
x_j	j−th element of the input vector
y	output vector
y^i	output vector in example i
y_j	j−th element of the output vector
z	example set
z^i	example i from example set
ε	instantaneous squared error
φ	transfer function
η	learn factor
θ	bias
ϑ	hidden output function

List of Abbreviations.

ANN	Artificial Neural Network
EBP	Error Backpropagation
ECG	Electro−cardiogram
HW	Hardware
KRS	Knowledge, Randomness and Symmetry
MDCP	Mean Direction Coefficient Predictor
MLP	Multilayer Perceptron
OS	Ordered Sampling
OSRI	Ordered Sampling within Randomly Selected Intervals
RMS	Root Mean Square
RS	Random Sampling
RSRI	Random Sampling within Randomly Selected Intervals
SW	Software
WSS	Windowed Selective Sampling

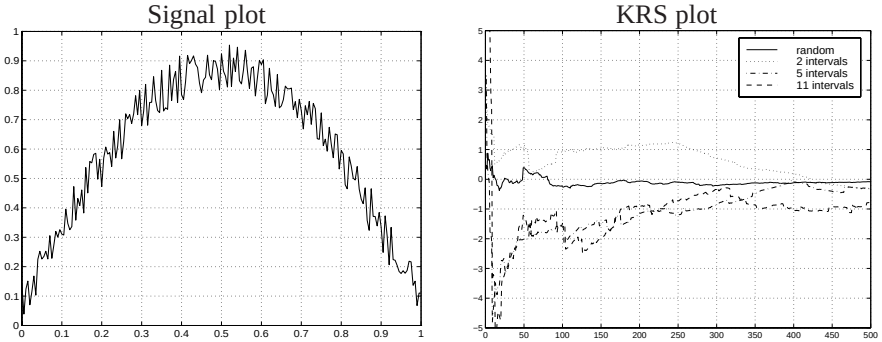
Appendices.

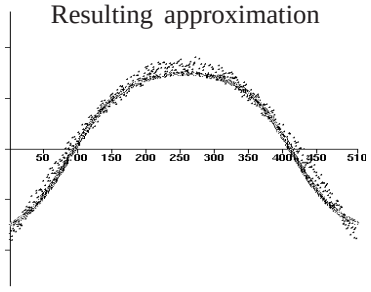
Appendix A: KRS–experiments.

The following figures represent different training cases, which has been performed to show the results from sample reordering algorithm. Every case is described by three figures. The first shows the original training signal; the second shows KRS plots by different example reorderings. The third plot represents either a snapshot from the Interact visualizer, or the generalization curves when testing. Some of the plots also appear in various places in the thesis. The systematization made here aims to give a quick overview of more made experiments and an easy way for their comparison and to allow a easy reference to various experiments, when the pictures are not really necessary to be plotted within the text. One is refered to table 5–1 for more numerical data.

A.1 Signal No 1.

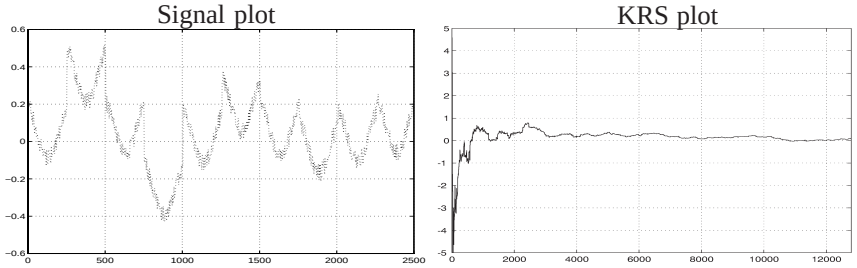
Sinewave with noise (see also figure 5–2).
It shows, how even the addition of noise on the input signal does not remove the potential internal cancelation. However, dividing the signal into intervals removes the tendence to paralysis.

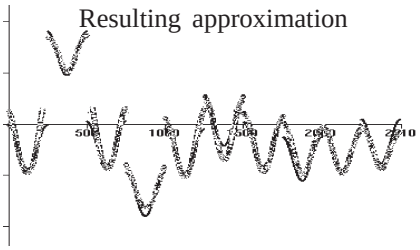




A.2 Signal No 2.

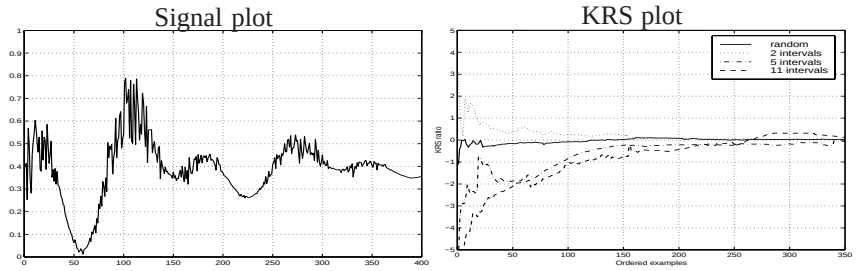
Complex goniometrical signal (see also figure 4–7).
The envelope of the signal is learned very fast, but then the internal cancelation prohibits any further progress.

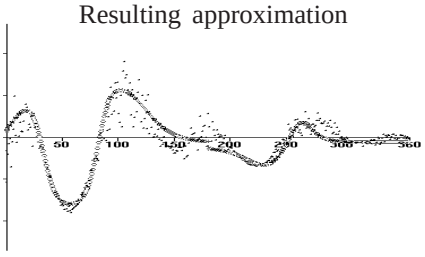




A.3 Signal No 3.

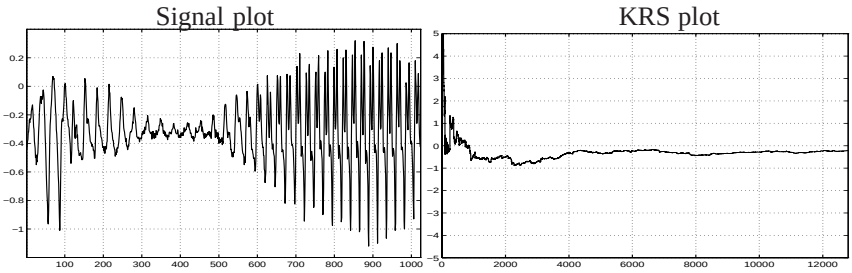
Not appearing elsewhere in this thesis.
An arbitrary signal with local symmetry and additional noise has learning problems unless the input space is divided in small enough intervals.

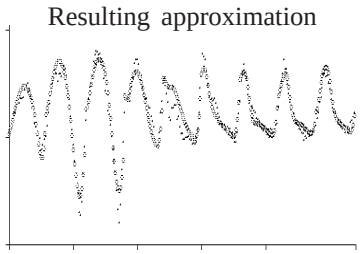




A.4 Signal No 4.

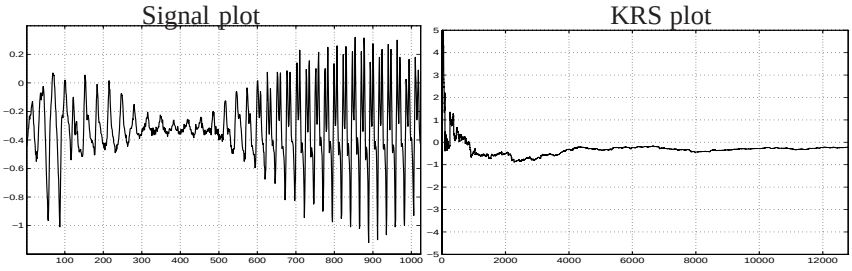
Power generator signal (see also figure 5–5a).
Approximation for the signalpart corresponding to the first 400 timesteps is limited to the training of the envelope.

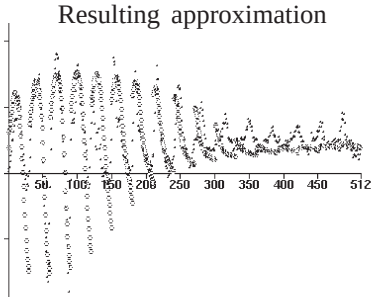




A.5 Signal No 5.

Power generator signal (see also figure 5–5a).
Approximation for the signalpart corresponding to the first 512 timesteps will not even learn the envelope for longer time fragments.

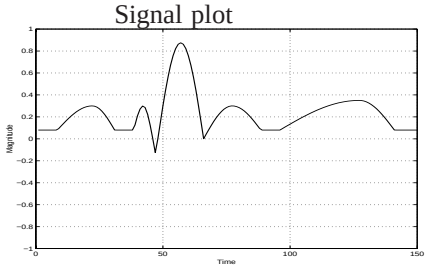




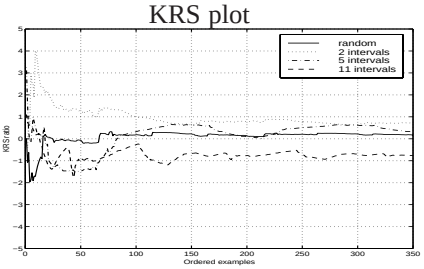
A.6 Signal No 6.

Single QRS signal (see also section 5.3.2).
The signal with some global but foremost local symmetry becomes (better) trainable when the number of intervals is raised.

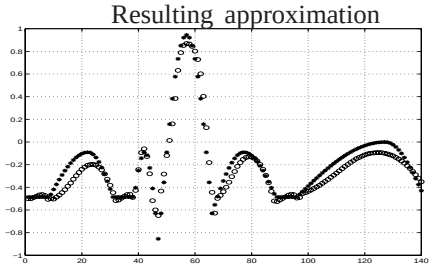
Signal plot



KRS plot



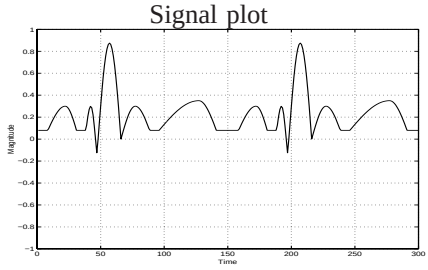
Resulting approximation



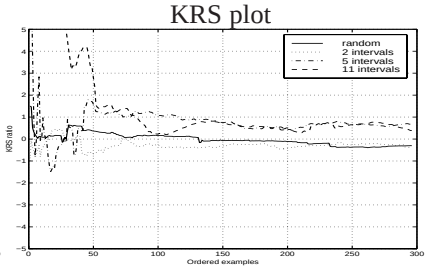
A.7 Signal No 7.

Double QRS signal (see also section 5.3.2).
For the larger signal length the global symmetry is less apparent and training is already performed for the smaller interval numbers.

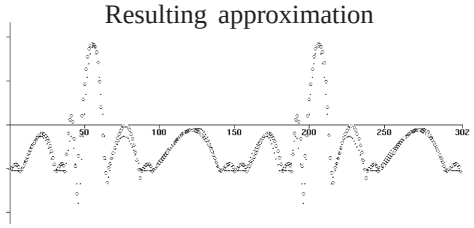
Signal plot



KRS plot



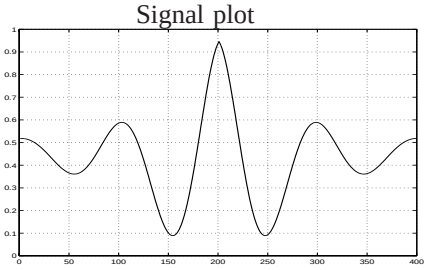
Resulting approximation



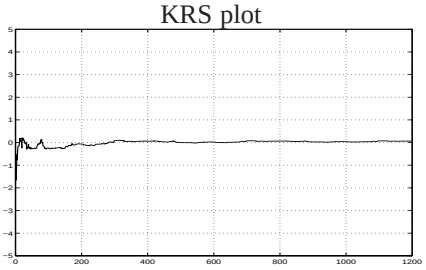
A.8 Signal No 8.

Symmetrical signal with second–order problem (see also figure 5–1a).
A complex signal with high global symmetry will have already have problems at the onset of learning.

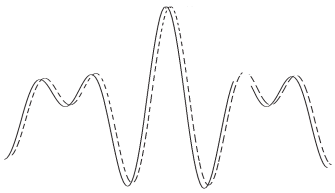
Signal plot



KRS plot



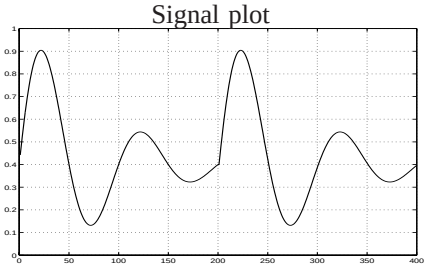
Resulting approximation



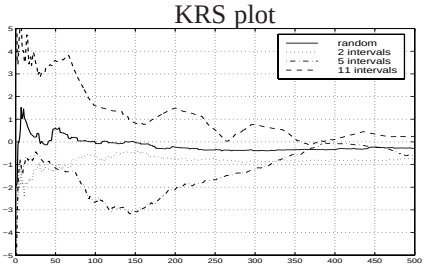
A.9 Signal No 9.

Asymmetrical signal with second–order problem (see also figure 5–1b).
A complex signal with little global symmetry will encounter local symmetries later in the training process.

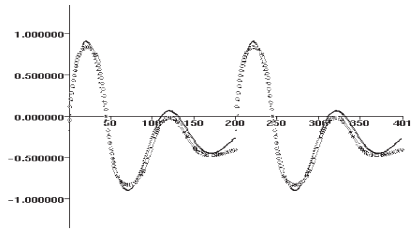
Signal plot



KRS plot



Resulting approximation



Appendix B: The Random walk.

There are numerous intuitions about the outcome of myriad experiments with events of a random nature, which happen to be wrong. The random walk theory reveals this misjudgment and is a basis for more advanced theories. For the analysis, made in this thesis, several properties of a random walk are interesting. First, it is of use to know how often the successive cumulative gains are becoming zero. This quantification is related to getting a notion of how fast network parameters can degrade to the zero point during adaptation. Second, it is of interest how the random walk depends on the length of the step. This gives inside to the problem why the initial symmetrical phase, when the network parameters has small values, is the most often encountered degradation problem. The third interesting question is what the spread and distribution of the endpoints by many random walks is, which will help to understand how many experiments, which because of the reasons explained in the thesis are governed nearly by the lows of the random processes, have the chance to escape the stationary areas, and how this chance is related to the number of training examples in a cancelation set.

B.1 The random walk in one direction.

A ideal coin–tossing game is a well accepted way to describe the random walk problem. The outcomes of individual tosses are represented geometrically on a rectangular coordinate system with horizontal t–axis and vertical y–axis. Every point on this coordinate system has as an abscis the number p of the current trial and as an ordinate the partial sum of the previous coin tossings s_p. All the partial sums draw a path $(s_1, s_2, \ldots, s_p)$. Every path is the outcome of a random walk experiment. Correspondingly the statistical characteristics of the multitude of paths can be quantified.

The probability, that at epoch n the path S has reached the point r, is denoted by $p_{n,r}$.

$$p_{n,r} = P\{S_n = r\} = \binom{n}{\frac{n+r}{2}} 2^{-n} \tag{B–1}$$

To the investigations, made in this thesis, the case is interesting, when the point r is the zero point. In the theory of random walks, this case is known as a *return to the origin.*

A return to the origin occurs at epoch k, if $S_k = 0$. Here k is necessarily even, and for $k = 2\nu$ the probability of a return to the origin equals $p_{2\nu,0}$. Because of its frequent occurrence this probability it will be denoted by $u_{2\nu}$.

$$u_{2\nu} = \binom{2\nu}{\nu} 2^{-2\nu} \tag{B–2}$$

The probability, that the first return to the origin occurs at epoch $2n$ is given by the following equation:

$$f_{2n} = \frac{1}{2n-1} u_{2n} \tag{B–3}$$

Another quantifiable characteristic for a set of paths is the spread of the endpoints. It is defined by the probability, that the maximum of a path of length n leading to point $A = (n, k)$ and having a maximum $\geq r$ with $k \leq r$ is denoted by $p_{n,2r-k} = P\{S_n = 2r - k\}$.

B.2 Generalizing the random walk.

All the conclusions made in the previous section concern the one–dimensional random walk by which the step length equals to unity. They can be illustrated with the outcome of a coin–tossing game. Here generalizations for multi–dimensional random walks as well as for random walks with unequal step length will be made.

In a two–dimensional random walk it can be imagined, that a particle moves in unit steps in the four directions, parallel to the $t-$ and $y-$ axes. For a particle which starts from the beginning of the coordinate system, there are four possible positions which have real–valued coordinates. Similarly, in three dimensions every point has six neighbors. The random walk is then specified by the corresponding four or six probabilities. The generalizations will be made for a symmetric random walk, where all four or six directions are equally probable. The probability of a return to the origin is:

$$u_{2\nu} = \frac{1}{4^{2n}} \sum_{k=0}^{n} \frac{(2n)!}{k!k!(n-k)!(n-k)!} = \frac{1}{4^{2n}} \binom{2n}{n} \sum_{k=0}^{n} \binom{n}{k}^2 \qquad \text{(B–4)}$$

Equation (B–4) can be generalized for a higher dimensional case.

By the generalized one–dimensional random walk the restriction, that the particle moves in unit steps is avoided. In this case at each step the particle shall have the probability p_k to move from any point x to $x + k$. where the integer k can be zero, positive or negative. This generalization is also known as *sequential sampling*.

B.3 Interpretation.

The theory of random walks is not (or at least not directly) applicable to neural networks. On each presentation, the weights will be adapted. In other words, each state of the neural network represents a different history and therefore a permuted input set will lead to a different state. Nevertheless, the characteristics of a non–learning neural network seem comparable. This can be interpreted in the following way. When a neural network is still in a state of infancy, it makes no difference what this state is: by providing a canceling input stream all that has been learned can be unlearned. It is only when a neural network has matured and “knows” what to do, that the different streams have a reduced input.

It is clear, that a neural random walk requires an enhanced theoretical model. The reason why this has not been attempted here, is largely that we intended to eliminate the occurrence of a longest ruin by construction rather than by analysis. Despite that, such a model will still be a welcome addition to the theory of neural design.

Appendix C: Software.

As illustration to the provided algorithms, we supply here the basic routine that allows for the reordering schemes as discussed in this thesis.

C.1 The Permutation procedure.

NR_PATTERNS: the overall number of patterns
UINT_MAX: largest unsigned integer

```
void            Permutation (int nr, int ptype)
/*_____________________________________________________________________ */
/* AIM:         select interval and permute the elements within          */
/*                                                                       */
/* INPUT:       int      nr       window count                          */
/*              int      ptype    permutation style:  INDEXSWAPPED      */
/*                                                    BOTHSWAPPED       */
/*                                                    VALUESSWAPPED     */
/*                                                    PERMWithOFFSET    */
/*_____________________________________________________________________ */
{               int      i;                    /* example index within window */
                int      val;                      /* example to be exchanged */
                int      j;                     /* support variable in exchange */
                int      k;                           /* exchangable example */
                int      size;                     /* # patterns per window */
                int      memint[NR_PATTERNS];              /* example set */
                int      pl[NR_PATTERNS];                /* presentation set */
                int      offset;                /* index of window start */
                int      ii;                             /* window index */
                time_t   t;                               /* clock time */
 unsigned       int      seed;                           /* random value */

/* Initialize for random value generation  ..................................... */
                time (&t);
                seed = t%UINT_MAX;
                srand48 (seed);

/* Initialize the presentation array in increasing index order  ................ */
/*      size: the number of patterns to be permuted ..........................*/
                if (nr > 1) size = NR_PATTERNS / nr;
                else        size = NR_PATTERNS;

                for (i = 0; i < NR_PATTERNS; i++)  pl[i] = i;


/* Perform the different  permutations  ........................................ */
```

```
/* PERMWithOFFSET delivers  offset..offset+size   or
 offset..NR_PATTERNS−1; 0..offset+size MOD NR_PATTERNS .   */
        if (ptype == PERMWithOFFSET)
        {     for (ii = 0; ii < nr; ii++)
              {         offset = (int) floor(NR_PATTERNS *drand48());
                    for (i = 0; i < size; i++)
                    {   val     = (int) floor(((size−i)*drand48()));
                        val     = (offset+val)% NR_PATTERNS;
                        k       = (offset+i)  % NR_PATTERNS;

                      /* swap pl[offset+i] and pl[offset+val] ..... */
                        j = pl[k]; pl[k] = pl[val]; pl[val] = j;
                    }
              }
        }
        else if ((ptype == BOTHSWAPPED) ||
                 (ptype == VALUESWAPPED))
        {       for (k = 0; k < nr; k++)
                {     for (i = 0; i < size; i++)
                      {   val = k * size + (int) floor(((size−i)*drand48()));

                        /* swap pl[N−i] pl[val] ..............................  */
                          j  = pl[(k+1)*size−i−1];
                          pl[(k+1)*size−i−1] = pl[val];
                          pl[val] = j;
                      }
                }
        }
        for (i = 0; i < nr; i++) memint[i] = i;

        if ((ptype == BOTHSWAPPED) ||
           (ptype == INDEXSWAPPED))
        {   for (i = 0; i < nr; i++)
            {       val = (int) floor(((nr−i)*drand48()));

                   /* swap m[N−i] m[val] ..................................  */
                    j  = memint[nr−i−1];
                    memint[nr−i−1] = memint[val];
                    memint[val]    = j;
            }
        }
}
```

Index of Terms.

M

N

O

P

Q

R

S

T

U

W

Samenvatting.

Het leren staat centraal in de studie naar de werking van neurale netwerken. In de loop der jaren zijn vele leeralgoritmen voorgesteld om een optimaal netwerk te verkrijgen, maar steeds weer blijkt het leerproces van wisselende kwaliteit en tijdsduur te zijn. Dit proefschrift onderzoekt de oorzaak van dit onvermogen tot experimentele reproduceerbaarheid en laat zien hoe met geringe ingrepen toch een consistent resultaat verkregen kan worden.

Neurale netwerken mogen zich in een groeiende populariteit verheugen. Weliswaar is van de oorspronkelijke verwachtingen om computers in imitatie van de biologie te maken weinig overgebleven, op andere gebieden heeft zich echter een technologie ontwikkeld die met minder hoogdravende doelen wel een markt–acceptatie heeft bewerkstelligd. Hiermee hebben zich de neurale netwerken als bruikbare technologie geetableerd, waarmee een zoveelste echec in haar bestaansgeschiedenis is voorkomen.

Al in de vijftiger jaren heeft de biologie als inspiratiebron gediend bij de ontwikkeling van nieuwe rekenconcepten. Een verdere ontwikkeling is tot staan gebracht toen Minsky aantoonde dat met concurrerende technologieen betere resultaten voor de toenmalige problemen te verkrijgen waren. De herontdekking van het Error Back–Propagation algoritme geeft in de tachtiger jaren aanleiding tot een herbezinning, met name als rekenwijze op grote parallelle computers. Begin negentiger jaren lijkt ook dit in het vergeetboek te komen.

Inmiddels is echter een derde stroom opgekomen die het neurale netwerk ziet als een methode om kennis te verwerven uit ongestructureerde gegevens. Met name is de aandacht gericht op beeldgegevens, gedragsgegevens en productie gegevens. Deze gebieden worden gekarakteriseerd door een onvolledige kennis, die door voorbeelden aangevuld zouden kunnen worden door een reele modellering van de werkelijkheid.

Tegenwoordig zien we dat neurale netwerken algemeen geaccepteerd zijn op een aantal deelgebieden van de industriele praktijk. Voor de sturing van staalwalsen zijn neurale regelaars wereldwijd in gebruik gekomen, in de voorspelling van consumenten gedrag worden veel neurale functies toegepast en ook voor de visuele inspectie van productie processen is duidelijke vooruitgang geboekt. Niet altijd wordt overigens het neurale model als zodanig ook in gebruik genomen; vaak dient het vooral als een geschikt beginpunt voor een verdere klassieke ontwikkeling.

Ondanks dit alles blijft vaak de klacht genoemd worden dat de ontwikkelingsgang zelfs voor de expert onbetrouwbaar is. Met name de leertijd van het netwerk laat een grote variatie zien, terwijl soms na lange tijd het probleem niet leerbaar blijkt. Het loont dus de moeite zich verder te verdiepen in de leerproblemen en methoden te ontwikkelen die een betere garantie bieden op een uiteindelijk resultaat. Het voorliggende proefschrift stort zich dan ook op deze problematiek:

Kan het leren van een neuraal netwerk gestabiliseerd en gegarandeerd worden?!

Eerst wordt daarvoor een definitie van betrouwbaarheid ontwikkeld. Betrouwbaarheid is een bekend begrip in de ingenieurswereld, maar behoeft aanpassing voor het thema

neurale netwerken. Al in 1991 hebben Nijhuis en Spaanenburg aangegeven dat de betrouwbaarheid van een kunstmatig neuraal netwerk anders dan in haar biologische evenbeeld niet vanzelf tot stand komt. Wij geven aan dat de betrouwbaarheid niet alleen in de fout–tolerantie maar tevens in de reproduceerbaarheid van het leerproces zichtbaar wordt.

Vervolgens gaan we op zoek naar de interne fenomenen die deze gebrekkige reproduceerbaarheid veroorzaken. De oplossing is gericht op het vinden van de interactie tussen de factoren, die het leerproces in zijn algemeenheid bepalen: (a) de symmetrie in probleem– en oplossingsruimte, (b) de schijnbaar willekeurige keuzes, waarop de specificering en veralgemenisering van het neurale ontwerp gebaseerd is, en (c) de soort en hoeveelheid kennis, die uit de leervoorbeelden afgeleid moet worden.

Uit een vergelijking met het gebruik in een aantal verwante gebieden blijkt dat symmetrie veelal wenselijk is voor een compacte mathematische notatie van een verkregen model, maar dat, als dit model nog tot stand moet komen (zoals in het neurale systeem), het creatie proces door een overmaat aan symmetrie tot een vorm van onbeslisbaarheid kan worden gebracht dat een goed resultaat in de weg staat.

Het blijkt dat veel leerproblemen op het symmetrie begrip terug te voeren zijn. Niet alleen kan aan het begin van het leren reeds modelvorming onmogelijk worden, maar blijkt verder in het proces zelfs ”ontleren” niet bij voorbaat uit te sluiten. Ontleren of gedeeltelijk leren ontstaat in die gevallen waarbij de systeem symmetrie dominant aanwezig. Bij de meeste leerproblemen lijkt van zo’n dominantie niet direct sprake te zijn. Maar bij nadere beschouwing blijkt het leren soms merkbaar vertraagd en leidt herhaling van het experiment tot uiteenlopende rekentijden (en soms zelfs van uitkomsten). De mate van reproduceerbaarheid van rekentijden heeft een directe relatie met de leerbaarheid van het probleem en kan gebruikt worden als een indicatie (waarschuwing) voor intrinsieke leerproblemen.

Het verwerven van kennis kan opflakkeren, branden, smeulen en uitdoven. Een brand heeft materiaal, zuurstof en de juiste temperatuur nodig. Zuurstofgebrek zal de brand doen uitdoven. Daarentegen voert wind steeds weer nieuwe zuurstof toe en wakkert daarmee de brand aan, terwijl zij een beginnend brandje eerder door afkoeling tot uitdoven zal brengen. Alles bij elkaar een zeer gecompliceerd samenspel, waarbij iedere factor zowel positief als negatief kan werken, en daardoor een wiskundige beschrijving niet triviaal maakt. In de Griekse natuurfilosofie is dan ook eerst het redeneermodel geformuleerd om later via vele experimenten tot een verdere detaillering te komen.

Voor de kennis uitdoving, die in deze thesis bestudeerd wordt, nemen we dezelfde beproefde aanpak. We formuleren eerst het KRS–model: een denkschema waarin het verband tussen de kennis in het netwerk, de ruis in de keuze van de voorbeelden en de symmetrie in de neiging tot leren tot uitdrukking komt. Aan de hand van dit schema kunnen we een beter inzicht verkrijgen in de wijze waarop voorbeeldkeuze en leren op elkaar inwerken.

Vanuit de experimentele ervaring kunnen we aan dit model het begrip KRS–verhouding toevoegen. Het geeft verdere informatie over de omstandigheden waarin kennis verwerving binnen het netwerk plaatsvindt en met name wanneer deze omstandigheden minder gunstig zijn voor het beoogde doel. Daarmee is het begrippenarsenaal gebouwd

en is het inzicht gescherpt waarmee getracht kan worden om zulke omstandigheden generiek te verbeteren.

In een eerste aanpak worden relatief eenvoudige signalen behandeld waarbij een oplossing nagestreefd wordt door bij voorbaat de onzekerheid in de leerproces ten gevolge van symmetrie effecten te verbreken. We concluderen vervolgens dat daarmee nog geen alomvattend resultaat verkregen is. Leerproblemen kunnen immers ook pas later ontstaan.

De vele resultaten in ons onderzoek ondersteunen de conclusie, dat de interne opbouw van het signaal een belangrijkere rol spelen in de leer proces dan het neurale systeem zelf. Dat is terug te zien in experimenten met signalen van vergelijkbare ingewikkeldheid die een verschillende betrouwbaarheid tonen als de mogelijkheid tot uitdoving aanwezig is. We komen dan tot een algemene benadering, waarin het signaal (de signalen) zelf behandeld te worden voor een beter leerresultaat. De voorgestelde methodiek is gebaseerd op een stelselmatige lokalisatie van de problematiek door gebruik te maken van de KRS–verhouding. D.w.z. in de veelheid van keuzes voor de presentatie volgorde van de voorbeelden kiezen we bij voorkeur die volgorde die potentieel de minste problemen zal geven.

Het onderzoek aan signalen met een hoge ingewikkeldheid wijst erop dat de tijdsinformatie (of positie van het voorbeeld) vanuit het signaal uitdovingseffecten kan veroorzaken. Dat gebeurt bij voorbeeld vaak bij periodieke signalen. We komen daarmee tot een derde bijdrage aan de verbetering van het leerproces. Vanuit de algemene aard van signalen kunnen karakteristieken afgeleid worden, die als ”feature” bij het leerproces het interne gebeuren zullen verbeteren. Ze dienen er vooral toe om een ongebreidelde proliferatie van uitdoving bij voorbaat een halt toe te roepen.

Door het proefschrift heen wordt een aantal probleemgevallen van diverse kanten belicht. Het betreft daarbij zowel academische speeltjes, realistische artefacten als daadwerkelijke realiteit. Regelmatig hebben we eenvoudige voorbeelden gebruikt, die speciaal ontworpen zijn voor de experimentele verificatie van de oorspronkelijke werkhypothese. Deze zijn zo ingericht dat, als het verwachte fenomeen zou bestaan, dit onmiddellijk en saillant tot uitdrukking zou komen. De moeilijkheid daarbij is dat zelfs de geringste afwijking van de beoogde experimentele inrichting al tot minder duidelijke resultaten kan leiden.

Verder zijn praktijksituaties op sterk gereduceerde schaal gebruikt om in te zien of en in hoeverre de bevestigde hypotheses in werkelijkheid niet door andere effecten dusdanig overschaduwd zouden worden, dat van daadwerkelijke realiteitswaarde geen sprake zou zijn. De opgedane ervaring doet vermoeden dat ons probleem ten grondslag zou kunnen liggen aan een aantal reeds eerder gerapporteerde probleemsituaties, waarvan tot nu toe slechts aan verbetering via alternatieve leeralgoritmes was gedacht. In een enkel geval is zelfs door de betrokken onderzoeker geconcludeerd dat het probleem onoplosbaar is.

Tenslotte wordt het betoog geverifieerd in zowel de diagnose van een turbo–generator als in de herkenning van een QRS–signaal. Dit zijn voor ons slechts willekeurige voorbeelden, die voornamelijk gekozen zijn om hun algemene belang en brede bekendheid. Op beide gebieden zijn voldoende en werkzame oplossingen gepresenteerd in de breedte van de toegepaste literatuur. Hier illustreren deze voorbeelden slechts, hoe

werkzaam de door ons voorgestelde wijze van omgaan met voorbeelden is. Het laat zien dat het proefschrift een bijdrage levert aan de verdere verdieping van het inzicht in de werking van kunstmatige neurale netwerken, gericht op het bouwen van kwalitatief hoogwaardige en betrouwbare intelligente systemen.

Acknowledgements.

It is with pleasure that I thank my colleagues, my friends and my family for their support and encouragement throughout the years taken to complete this work.

Firstly, I wish to thank my supervisor, Professor L. Spaanenburg. Ben, thank you for your genuine interest, untiring support and guidance throughout all the stages of this work. Your enthusiasm and the quality of your advice, I have found invaluable. I also wish to thank Dr. J. Nijhuis, my referent, for many invaluable discussions and his timely comments. I have learned much from both of you.

A warm thank to my husband Tino Lourens for the many discussions we had over different aspects of my work, and for the enduring faith and encouragement over the last five years. I am very grateful to you for your emotional support and loving attention even at times when I hardly deserved it.

I am indebted also to Wim Wiegering and his colleagues from SNN in Nijmegen, and also to David Saad, Ton Coolen, and Michel Biehl, for your advice and guidance. I wish to thank also the members of the Reading Committee, Prof. Kistner, Prof. van Bokhoven, and Prof. Nerbonne for their invaluable remarks and suggestions on my thesis.

To my colleagues within the Department of Informatics and Mathematics both past and present, thank you all. Peter Cruisinga and Rix Groenboom, Mark ter Brugge and Rienk Venema, without your friendship, the burden of this work would have been far greater.

Most importantly, I thank my mother Bojka, for her unquestioning support and love throughout my education and upbringing. I am much indebted to you.

IPA Dissertation Series number: 1999–05