Chapter 3

Artificial Neural Networks

In this chapter we present the basic principles and mechanisms of ANNs by means of an historical outline of the development of the field. As the biological model has played an important role in ANN research, and still is the source of a variety of new ANN models, we start with a short review of BNN properties. We conclude the following introduction focusing on the prominent *Error Back–propagation* algorithm for ANN training, which plays an important role in our system evolving ANSs.

3.1 Biological Neural Networks

Since the human brain has been identified as the center of thinking by Greek philosophers around 500 B.C., philosophers and scientists attempted to reveal structure and function of biological neural networks. But it was only at the end of the 19th century that the Spanish histologist *Cajal* was able to demonstrate that the nervous system is composed of interconnected cells (neurons) creating complex circuits. In the 1950s *Hodgkin*, *Huxley*, *Katz*, and *Eccles* were able to record electrical signals exchanged between neurons, and identified *Synapses* as biological connectors allowing the transmission of electrical signals from a source (*presynaptic*) to a sink (*postsynaptic*) neuron. In the 1980s methods became available to analyze the genetic control of differentiation of neurons and the expression of their functional properties (Shepherd, 1994), which is essentially what computer scientists began to use as a model for the artificial evolution of networks being at the center of this work.

According to Shepherd (1994, pp. 4–5) the main task of *Neurobiology* is to identify the functional and structural organization of the brain, and associate it with stages of information processing ultimately resulting in the physical

behavior of an organism. Barlow (1972) stated in his *Neuron Doctrine* that individual neurons represent specific events by hierarchically processing the information surveyed by the sensory input signals (Barlow, 1972). Consistent with Barlow's neuron doctrine researchers coined the terms *Pontifical Cell* (Sherrington, 1940), *Grandmother Cell* (Lettvin et al., 1959), and *Cardinal Cell* (Barlow, 1972) which should illustrate the one–to–one mapping of a neuron and a specific concept. However, it were not science, if researchers did not come up with the opposing concept of *Parallel Distributed Processing* advocating a one–to–many mapping between neurons and concepts having had great impact on the ANN community (Rumelhart et al., 1986). Todays solomonic view supports both hypotheses by classifying neurons into one of the two categories with possible intermediate (fuzzy) states always found in biological systems.

It is estimated that the human brain consists of approx. 10^{11} neurons with approx. 10^{14} connections. Hundreds of brain areas have been identified forming structural units that are interconnected at this higher level of organization. Though, many areas are associated with more or less specific tasks, it remains unclear, whether the structural units have a strict assignment to a function, or can be used for a variety of tasks. Again, the possibly unsatisfying answer to this question may be that both concepts are realized in biological neural networks including all variants of intermediate solutions.

A single neuron might seem to be a good candidate to serve as the atomic structural unit of a BNN, but even in terms of information processing a single biological neuron is an extremely complex structure, hence, it is also labeled *Microstructure*. Basically, a neuron sends a signal via its single *Axon* (output) and receives signals along the *Dendrites* (inputs) (Figure 3.1), however, the discrimination between axons and dendrites is not trivial, and many neurons send their signals over dendrites. Axons and dendrites of different neurons are coupled by electrical and/or chemical synapses. In electrical synapses ions carrying the signal directly traverse the small distance between the connected "cables", while in chemical synapses the electrical signal evokes a transmission of *Neurotransmitters*, which in turn generate an electrical signal on the receiving side. Axons, dendrites, and cell bodies (*Soma*) may be connected in any combination (Shepherd, 1994). Considering that some cells have hundreds and thousands of dendrites with a number of synapses on each of them, the complexity of a single neuron becomes apparent.

The electrical signals (*Action Potentials*) are generated and transmitted by complex processes in the neurons, where activation–dependent ion channels regulate the flow of charges through the cell's membrane involving K^+, Na^+, Ca^+, and Cl^- ions. At certain electrical states the neuron discharges and triggers an action potential with a maximal frequency of approx. 100 Hz,

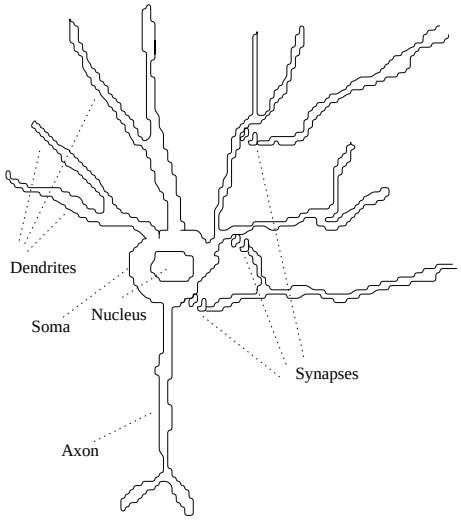

Figure 3.1: A vertebrate motoneuron with exemplary synaptic connections (after Shepherd (1994, p63)).

which compared to the clock rate of modern microprocessors is unbelievable slow, but the massive parallelism seen in the brain accounts for this deficit in signal speed.

An even higher degree of complexity at the neuron level is introduced by the fact that biological neurons are, of course, living structures changing their long– and short–term behavior under the influence of various *Neuromodulators* enabling *Synaptic Plasticity* (Shepherd, 1994). Above properties contribute to the astonishing plasticity of biological neural networks and could give some important clues for further improvement of ANNs. However, despite all the excitement the term ANN may imply, we should always keep in mind that most of todays ANN models are a high–level abstraction of a BNN. Essentially, the analogy is restricted to neurons (computational units) sending signals (numbers) over axons (connections) to other neurons, hereby transforming (processing) the signals.

3.2 A Brief ANN History

According to Haykin (1999, p38) it is generally agreed that the discipline of artificial neural networks was founded by McCulloch and Pitts (1943) with their work on neuron models capable of computing logical functions (McCulloch and Pitts, 1943). In this model excitatory connections $w_{j,i} = +1.0$ and inhibitory connections $w_{j,i} = -1.0$ are discerned. Inhibitory connections may

be absolute, completely switching off a neuron, or relative, attenuating the summed input signal. The activation function $a(x)$ is a threshold function generating binary signals.

The psychologist Donald Hebb (1949) suggested a fundamental hypothesis of learning in BNNs. In (Hebb, 1949) he says

> When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency as one of the cells firing B is increased.

The biological mechanisms of *Hebb Learning* have been explored many years later (in the 1970s), when *Long–Term Potentiation* (LTP) and *Long–Term Depression* (LTD) have been found to sustainably change synaptic properties. Hebb also suggested that the connectivity of BNNs changes with time and creates clusters of cooperating neurons organized in neuronal groups. Closely related to these ideas are two modern hypotheses in neuroscience: *Neural Darwinism* suggests that the neural pathways of biological brains are constantly sculptured by feedback of the environment, specifically during ontogeny (Edelman, 1987). Tononi and Edelman (1998) presented the *Dynamic Core* hypothesis proposing that brain areas are dynamically rearranged and allocated for the processing of specific tasks (Tononi and Edelman, 1998).

Although, Hebb's hypothesis originated strictly from the observation of BNNs, it is at the core of many prominent learning algorithms in ANNs, which can be derived in a pure mathematical way without using any biological inspiration. This may be viewed as an indication that the links between artificial and biological neural networks are stronger than generally conceded.

The psychologist Frank Rosenblatt (1958) presented the *Perceptron* model, which resembled basic organizational properties of BNNs. The perceptron network is composed of three layers of artificial neurons: a *Retina* (input layer), an Association (hidden) layer, and a *Response* (output) layer [1]. Retina and association layer are linked by feed–forward connections, while association and response layer also contain recurrent connections to previous layers. The real–numbered weights $w_{j,i}$ modulate the signals to a neuron, where the activation is modelled by a binary threshold function $a(x)$ (Rosenblatt, 1958). Many of the network architectures investigated and applied today are very similar to the perceptron, but in order to ease mathematical analysis Rosenblatt used single neurons as a simplified perceptron, which became the target of criticism of perceptrons in general.

[1] Today the expressions in parentheses are used.

Minsky and Papert (1969) published a number of proofs on the computational capabilities and limitations of perceptrons (Minsky and Papert, 1988). In condensed form the results of these analyses have been labeled as *Minsky's Paradox* stating that a single perceptron cannot learn the XOR function. However, later work has shown that *multi–layered* perceptrons (originally proposed by Rosenblatt) are universal approximators (Lippmann, 1987; Cybenko, 1989; Hornik et al., 1989).

Widrow and Hoff (1960) developed the *Adaptive LINear NEuron* (ADALINE) neuron interconnected in a *Multiple ADALINE* (MADALINE) network for adaptive filters in communication devices. Of specific importance for the development of ANN learning algorithms was the MADALINE's learning procedure based on minimization of the mean square error of ADALINEs (*Least Mean Square* (LMS) or *Widrow–Hoff* learning rule) (Widrow and Hoff, 1960). In essence the supervised Widrow–Hoff training is the basis of *Error Back–propagation* by solving the problem of algorithmic weight adjustment between an output and the directly connected previous layer. However, it took many more years until the credit assignment problem (the contribution of a specific neuron to network error) has been solved allowing supervised training of multi–layer perceptrons.

In the 1970s *Associative Memories* came into the spotlight of ANN researchers, e.g., (Anderson, 1972; Kohonen, 1972). Linear associators with an input and an output layer linked by weighted connections can be trained using concepts from linear algebra. The resulting weight update rules essentially constitute Hebbian learning, also termed *Correlative* learning in the context of associative memories.

In the 1980s the attraction to the ANN field regained momentum and a series of new ANN types has been proposed. Grossberg presented *Adaptive Resonance Theory* (ART) networks based on *Competitive* learning. The main idea behind ART is to escape the *Stability–Plasticity* dilemma of neural networks, i.e., the problem of learning new concepts, while not forgetting those already learned. Thus, in an ART network the response to input patterns is fed back to the input layer via recurrent connections. If input and response match each other, a resonance process is initiated strengthening the connections between input and response layer. If the input is unknown to the network, a new concept (assigned to an output neuron) is formed (Cohen and Grossberg, 1983).

Kohonen (1982) presented the *Self–Organizing Map* (SOM), a network type inspired by the sensory input mapping in BNNs (Willshaw and von der Malsburg, 1976), which is trained by unsupervised, competitive learning. The output layer is organized in a one–, or two–dimensional space inducing neighborship relations between output neurons. An input pattern propagated to

the output layer results in a winner neuron having the highest activation in the output layer. The weight vectors to the winner neuron and its neighbors are then shifted towards the input pattern vector (again a form of Hebbian learning). During training the input data are mapped to output neurons, whereby the topology of the input space is preserved. Each output neuron represents a region of the input space, i.e., a number of input values is represented by the weight vector to this neuron (*Vector Quantization*). The topology of the input space is preserved in the output layer (map), even when the dimension of the input space is reduced by the SOM (Kohonen, 1982).

The existence of recurrent connections in an ANN inevitably leads to dynamical phenomena of the system. Hopfield (1982) based the analysis of a recurrent network, where each neuron is connected to all other neurons, with symmetrically weighted connections on the *Spin Glass* model describing condensed matter with disordered magnetic moments. In this model the minimization of energy leads to a stable attractor of the dynamical system. When the weights of a network are viewed as energy of the network, energy minimization (training) results in connection weights allowing convergence of the activations of the network. Thus, (activation) patterns can be stored in stable attractors of the network constituting a *Content Addressable Memory*: a pattern similar to a stored pattern will cause the network to converge to the pattern in memory (if the applied pattern is in the basin of attraction) (Hopfield, 1982). It is unclear, if processes similar to the dynamics of a *Hopfield* net are fundamental for biological memories, but some researchers claim that chaos associated with dynamical systems plays an important role in BNNs (Freeman, 1994).

Arguably, the single most influential development in ANN history was the *Error Back–propagation* algorithm for multi–layer perceptrons reported in (Rumelhart et al., 1986). It solved the problem of assigning exact error terms to neurons in hidden layers, i.e., for each neuron the degree of its impact on the networks overall error can be calculated. Essentially, back–propagation is an optimization technique based on gradient descent, where the gradient of a high–dimensional function can be computed by an iterative procedure requiring only local information. The algorithm has already been reported in (Werbos, 1974) in a broader statistical context, but Rumelhart et al. (1986) worked out its specific merits in ANN training.

Since then, the multi–layer perceptron trained by error back–propagation has become the most utilized ANN type in most scientific and real–world applications (the standard neuron model employed in these networks is sketched in Figure 3.2). With an ANN software package at hand, users must "only" provide a set of problem–specific training data to teach a network. However, in order to achieve a well–performing network many parameters, e.g., number

of neurons, layers, and connections, or learning rates, have to be adjusted. For all of these ANN parameters no analytical rules exist to determine appropriate values, hence, ANN performance often relies on the expertise of human ANN designers. The automatic adjustment of ANN parameters, more generally components of an ANS, by evolutionary algorithms is at the center of this work. The general approach adopted is to combine evolution of ANS components with conventional ANN training that is based on error back–propagation being outlined in the following section.

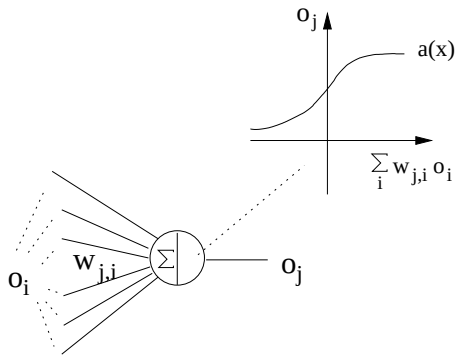

Figure 3.2: A standard artificial neuron with weighted sum of inputs Σ transformed by an activation function a(x).

3.3 Error Back–propagation

The specific mapping of an input pattern to an output pattern in an ANN is dependent on the network's weights $w_{j,i}$ and the neurons' activation function $a(x)$. In supervised learning known input/output patterns are used to determine the current error (deviation of actual network output from expected output). Hence, in a general form gradient descent towards minimal network error is given by

$$\Delta w_{j,i} = -\eta(\vec{\nabla}E)_{j,i}, \tag{3.1}$$

where $(\vec{\nabla}E)_{j,i}$ is the j,i–th component of the error gradient vector, η is the learning rate determining the step size in direction of the negative gradient, and $\Delta w_{j,i}$ is the resulting weight change. With hundreds and thousands of connection weights in a network solving Equation 3.1 becomes a complex, high–dimensional problem, as the network error function is highly non–linear (induced by the non–linear activation function). Error back–propagation

solves the problem by an iterative computation of the weight changes $\Delta w_{j,i}$ needing only local information from the two neurons linked by the connection weight. The method starts by calculating an error term at the network's output layer and computes the weight changes by back–propagating error terms through the network. Figure 3.3 shows a simple one–hidden layer network with all the parameters, which will be used for derivation of back–propagation learning.

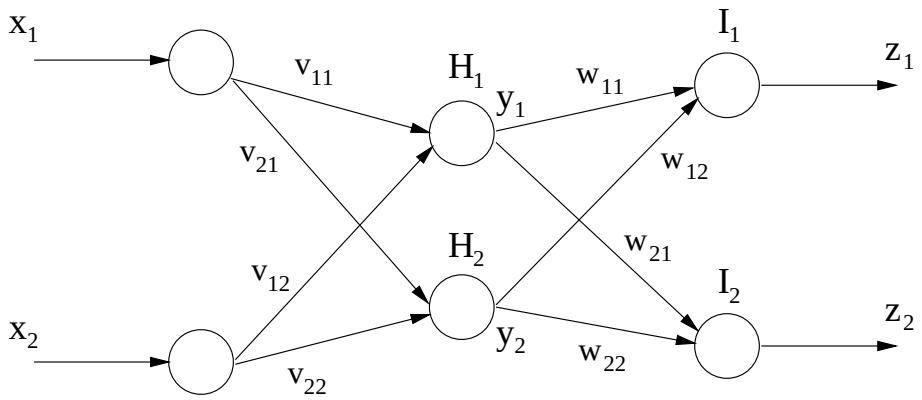

Figure 3.3: An exemplary feed–forward network with all the parameters relevant to back–propagation training (after (Patterson, 1996)).

The following definitions (Patterson, 1996) are valid for any one–hidden–layer network with m input neurons and n hidden neurons:

$$H_j = \sum_{i=0}^{m} v_{j,i} x_i \qquad I_k = \sum_{j=0}^{n} w_{k,j} y_j \tag{3.2}$$

with H_j and I_k being the sum of the weighted input signals of a hidden layer neuron and an output layer neuron, respectively. Note that the bias of each neuron is represented by an additional weighted connection to a neuron x_0 or y_0 (not shown in Figure 3.3) having a fixed activation of 1.0. By definition the input neurons do not transform the input signal, i.e., the activation function is the identity function and the bias is 0.0.

The input signal to a neuron is transformed by the activation function a to give

$$y_j = a(H_j) \qquad z_k = a(I_k), \tag{3.3}$$

where y_j and z_k are the neuron's activation.

The overall output error of the network is given by the sum of the error terms of the o output neurons

$$E^{(p)} = \frac{1}{2} \sum_{k=1}^{o} (t_k^{(p)} - z_k^{(p)})^2. \tag{3.4}$$

The superscript p (omitted in the following) indicates that $E^{(p)}$ is the error of a specific input pattern in the training set. An output neuron's error term is simply given by the square of the difference of the target value t (defined in the training set) and the actual output activation z. Though, the *Summed Square Error* (SSE) is the most common, any error term can be used for back–propagation, e.g., the error of the network on the validation set, or even an error signal generated by a human or a machine.

Referring to Equation 3.1 we now calculate the weight change of the connections between hidden and output layer $\Delta w_{k,j} = -\eta \frac{\partial E}{\partial w_{k,j}}$. As the error E is primarily a function of I_k, which in turn is a function of $w_{k,j}$, application of the chain rule of partial differentiation yields

$$\frac{\partial E(I_k(w_{k,j}))}{\partial w_{k,j}} = \frac{\partial E}{\partial I_k} \frac{\partial I_k}{\partial w_{k,j}} = \frac{\partial E}{\partial I_k} y_j. \tag{3.5}$$

Further use of the chain rule and Equation 3.4 give

$$\frac{\partial E(z_k(I_k))}{\partial I_k} = \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial I_k} = -(t_k - z_k) a'(I_k), \tag{3.6}$$

and defining a *Generalized Error Term* δ_k at the output layer as

$$\delta_k = (t_k - z_k)\, a'(I_k) \tag{3.7}$$

finally results in

$$\Delta w_{k,j} = -\eta \frac{\partial E}{\partial w_{k,j}} = \eta\, \delta_k\, y_j. \tag{3.8}$$

Equation 3.8 is essentially the Widrow–Hoff learning rule (setting $a'(I_k) = 1.0$) and has exactly the structure of the *Generalized Hebb Rule* $\Delta w_{k,j} = \eta\, z_k\, y_j$. In supervised back–propagation learning the generalized error term δ (provided by a teacher) substitutes the activation z_k of the unsupervised Hebb learning.

Calculating the weight changes for the connections between input and hidden layer, we again, start with the general gradient descent $\Delta v_{j,i} = -\eta \frac{\partial E}{\partial v_{j,i}}$, which gives

$$\frac{\partial E(H_j(v_{j,i}))}{\partial v_{j,i}} = \frac{\partial E}{\partial H_j} \frac{\partial H_j}{\partial v_{j,i}} = \frac{\partial E}{\partial H_j} x_i, \tag{3.9}$$

and

$$\frac{\partial E}{\partial H_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial H_j} = \frac{\partial E}{\partial y_j}\, a'(H_j). \tag{3.10}$$

Right here the problem of credit assignment, how to determine the influence of the activation of a hidden neuron on the overall error of the network, arises. It is solved by using the chain rule, again.

$$\frac{\partial E}{\partial y_j} = -\frac{1}{2}\sum_{k=1}^{o}\frac{\partial (t_k - f(I_k))^2}{\partial y_j} = -\sum_{k=1}^{o}(t_k - z_k)\, a'(I_k)\, w_{k,j} \tag{3.11}$$

Defining the generalized error term at the hidden layer as

$$\delta_j = a'(H_j)\sum_{k=1}^{o}\delta_k\, w_{k,j} \tag{3.12}$$

the distribution of errors at the output layer δ_k to the error term δ_j for each neuron in the hidden layer is described elegantly. The output errors are weighted by the connection weights and summed up at the output of each hidden neuron. During training the errors are propagated in a manner very similar to signals in network recall, however, in the opposite direction, which gives error back–propagation its name. Though, we have restricted the analysis to one–hidden layer networks, it can be easily extended to an arbitrary number of layers and to generalized multi–layer perceptrons allowing any connections in forward direction. Equation 3.12 enables the computation of an error term for any neuron, whose post–synaptic error terms are known.

In analogy to Equation 3.8 the final weight update rule for connections between input and output layer is given by

$$\Delta v_{j,i} = -\eta\,\frac{\partial E}{\partial v_{j,i}} = \eta\,\delta_j x_i. \tag{3.13}$$

Clearly, it can be seen that back–propagation requires only local knowledge of the network's parameters. In fact, for each weight update of a connection only two parameters, the activation of the source neuron and the generalized error term of the sink neuron, are needed. Compared to the general gradient, where most parameters of the network appear non–linearly in each component of the gradient vector, back–propagation is a very effective way to change the weight towards an error minimum. Still, there are a number of problems associated with the presented plain error back–propagation.

Although, the single weight updates are computationally cheap operations, the complexity of the ANN structure (number of neurons and connections), the size of the training data set, and the number of *Epochs* can make ANN training very expensive. Typically, a network is trained for some hundreds to some thousands of epochs. In each epoch (cycle) all patterns of the training data set are presented to the network sequentially.

Two modes of back–propagation learning can be discerned by the frequency of actual weight updates (Haykin, 1999). With *Batch* or *On–line Learning* the computed weight changes are accumulated for all patterns and are only applied after completion of an epoch. While this method allows an accurate calculation of the error gradient, it may suffer from redundant (identical) pattern in the training data set. *Sequential* or On–line learning changes the weights after each pattern, hence, randomness is introduced by the order of the patterns in the training data sets. Though, computation of the error gradient is only based on a single pattern, on–line learning is believed to converge faster, however, the differences may be small and dependent on the specific training data.

Like any optimization procedure back–propagation can be trapped in local minima of the error surface. While no improvement of back–propagation can guarantee to find the global error minimum, a number of enhancements aim at decreasing training time. A simple technique is *Gradient Reuse*, where the weight changes computed for a single epoch are repeated as long as the overall network error decreases. This technique is a first step towards adaptation of the learning rate during training, as a series of e epochs with reuseable weight changes effectively alters the learning rate η to $\eta' = e\,\eta$.

Evidently, the learning rate has a great influence on training time and convergence. A small learning rate enables sampling of the error surface with high resolution, which increases the chances to find any minimum, but also the possibility to end up in a local minimum. Moreover, it increases training time in flat regions of the error surface. A large learning rate quickly traverses flat regions, but it may also cause the algorithm to miss regions of high interest, namely, the global minimum or minima close to it. A simple, but effective adaptation of the learning rate is achieved by addition of a *Momentum* term α (Rumelhart et al., 1986) as given by

$$\Delta w_{j,i}(t) = -\eta \frac{\partial E}{\partial w_{j,i}} + \alpha\, \Delta w_{j,i}(t-1), \tag{3.14}$$

with $\Delta w_{j,i}(t-1)$ being the weight change in the previous epoch. The momentum term accelerates learning in flat regions of the error surface, while fine–tuning the search in "rough terrain". The trade–off for these benefits

is the increased probability to miss interesting minima close to flat regions (*overshooting*).

A number of algorithms based on back–propagation employ a more sophisticated adaptation of learning parameters during the learning process. A member of this family of algorithms is *Resilient Back–propagation* (RPROP) (Riedmiller and Braun, 1993), which will be described in more detail in the following section, as RPROP is commonly used to train evolved ANN structures in this work.

According to Equation 3.1 the computation of weight changes in back–propagation is based on a linear approximation of the gradient through the current point of the error surface. Additional information on the curvature of the surface would improve a training algorithm, as it would lead to a more exact path towards decreasing error. However, these improvements demand costly computation of the *Hessian* matrix containing all possible second derivatives of the high–dimensional error surface, inducing a non–local weight update scheme.

Various methods for the solution of these second-order optimization problems exist, e.g., the *Newton* method, but they become computationally impractical for larger networks. Even the simplified *Quasi–Newton* method only requiring estimations of the gradient has a computational complexity of $O(|C|^2)$ ($|C|^2$ being the number of connections) (Haykin, 1999), which leaves *Conjugate–Gradient* methods the only promising alternative to deal with training of larger networks based on second–order methods (Fletcher, 1987).

Secant methods try to reduce the computational cost of second–order methods by a rough estimation of the shape of the error surface. A representant of secant methods is *Quickprop* (Fahlman, 1988) assuming a local parabolic shape of the error surface resulting in the basic weight update

$$\Delta w_{j,i}(t) = \frac{g(t)}{c + g(t-1) - g(t)} \Delta w_{j,i}(t-1), \tag{3.15}$$

where $g = \frac{\partial E}{\partial w_{j,i}}$, and c is a constant eliminating the "flat spot" introduced by $g(t-1) = g(t)$. Here, the shape of the error surface is approximated by the secant given by the difference of the gradient of two consecutive epochs. A small difference, indicating the more linear region of the assumed parabolic shape, results in a larger weight change (following the linear gradient), while a larger difference is a sign for larger curvature, hence, a smaller weight change.

3.3.1 Resilient Back–propagation

A problem associated with standard back–propagation is the dependence of the generalized error term δ on the first derivative of the activation function. In large regions of the widely used sigmoidal functions the numerical value of the derivative can be very small resulting in a slowly converging ANN training. Jacobs (1988) has proposed four heuristics to improve the convergence rate of back–propagation (Jacobs, 1988):

1. Adaptation of network weights is controlled by connection–specific learning rates.
2. The learning rates are adapted to the error surface during training.
3. If the error gradient for consecutive training epochs does not change its direction (sign), the learning rate is increased.
4. If the error gradient changes its direction, the learning rate is decreased.

All of the above heuristics to improve back–propagation are incorporated in the *Resilient Back–propagation* (RPROP) algorithm (Riedmiller and Braun, 1993). Instead of calculating the weight change using the linear approximation of the gradient $\Delta w_{j,i}(t) = -\eta \frac{\partial E}{\partial w_{j,i}}$, for each weight a single variable $\Delta_{j,i}(t)$ governs the weight update, whose numerical value depends on the sign of the gradient, as given by

$$\Delta w_{j,i}(t) = \begin{cases} -\Delta_{j,i}(t) & \text{if } \frac{\partial E}{\partial w_{j,i}} > 0 \\ +\Delta_{j,i}(t) & \text{if } \frac{\partial E}{\partial w_{j,i}} < 0 \\ 0 & else. \end{cases} \tag{3.16}$$

The weight change $\Delta_{j,i}(t)$ is dynamically adapted during training increasing, when the gradient does not change its principal direction in two consecutive epochs, and decreasing, when the gradient changes its sign. Quantitatively, the adaptation is given by

$$\Delta_{j,i}(t) = \begin{cases} \eta^+ \Delta_{j,i}(t-1) & \text{if } \frac{\partial E(t-1)}{\partial w_{j,i}} \frac{\partial E(t)}{\partial w_{j,i}} > 0 \\ \eta^- \Delta_{j,i}(t-1) & \text{if } \frac{\partial E(t-1)}{\partial w_{j,i}} \times \frac{\partial E(t)}{\partial w_{j,i}} < 0, \end{cases} \tag{3.17}$$

where the constants η^+ and $\eta-$ are fixed to empirical values in the range of $0 < \eta^- < 1 < \eta^+$. The initial value of the weight change Δ_0 and the maximal weight change Δ_{max} are the two RPROP learning parameters, which will be evolved in experiments in Section 7. The training result is believed to be rather insensitive to both parameters, as the weight updates are adapted to

the error surface. However, a Δ_{max} being too large could result in a coarse sampling of the error surface missing regions of high interest.

Bibliography

Almasi, G. S. and Gottlieb, A. (1989). *Highly Parallel Computing.* Benjamin/Cummings. ISBN 0-8053-0177-1.

Anderson, J. A. (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14:197–220.

Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1994). An Evolutionary Algorithm that Constructs Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 5(1):54–64.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford University Press.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In Grefenstette, J. J., editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 101–111. Texas Instruments, Inc. and Naval Research Laboratory, Lawrence Erlbaum Associates.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In Grefenstette, J. J., editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21. Naval Research Laboratory, Lawrence Erlbaum Associates.

Balakrishnan, K. and Honavar, V. (1995). Evolutionary design of neural architectures – a preliminary taxonomy and guide to literature. Technical Report CS TR #95-01, Iowa State University, Department of Computer Science, Ames, Iowa 50011–1040, U.S.A.

Barlow, H. B. (1972). Single units and cognition: A neurone doctrine for perceptual psychology. *Perception*, 1:371–394.

Beasley, D., Bull, D. R., and Martin, R. R. (1993). An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69.

Blake, C. and Merz, C. (1998). http://www.ics.uci.edu/~mlearn/mlrepository.html. WWW Repository, University of California, Irvine, Dept. of Information and Computer Sciences.

Blickle, T. and Thiele, L. (1995). A Mathematical Analysis of Tournament Selection. In Eshelman, L., editor, *Genetic Algorithms: Proceedings of the 6th International Conference (ICGA95)*, San Mateo, California. Morgan Kaufmann Publishers, Inc.

Bornholdt, S. and Graudenz, D. (1992). General Asymmetric Neural Networks and Structure Design by Genetic Algorithms. *Neural Networks*, 5:327–334.

Branke, J. (1995). Evolutionary Algorithms in Neural Network Design and Training – A Review. In Alander, J. T., editor, *Proceedings of the First Nordic Workshop on Genetic Algorithms and their Applications*, pages 145–163.

Brindle, A. (1981). *Genetic Algorithms for Function Optimization.* PhD thesis, University of Alberta.

Bruce, J. and Miikkulainen, R. (2001). Evolving Populations of Expert Neural Networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 251–257, San Francisco. Morgan Kaufmann.

Caruna, R. A. and Schaffer, J. D. (1988). Representation and Hidden Bias: Gray vs. Binary Coding for Genetic Algorithms. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 153–161.

Castillo, P. A., Carpio, J., Merelo, J. J., Prieto, A., Rivas, V., and Romero, G. (2000). Evolving Multilayer Perceptrons. *Neural Processing Letters*, 12(2):115–128.

Cavaretta, M. J. and Chellapilla, K. (1999). Data Mining using Genetic Programming: The Implications of Parsimony on Generalization Error. In *Congress on Evolutionary Computation*, pages 1330–1337. IEEE.

Chalmers, D. J. (1990). The Evolution of Learning: An Experiment in Genetic Connectionism. In Touretsky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Proceedings of the 1990 Connectionist Summer School*, pages 81–90, San Francisco, CA. Morgan Kaufmann.

Chellapilla, K. and Fogel, D. B. (1999). Evolving Neural Networks to Play Checkers without Expert Knowledge. *IEEE Transactions on Neural Networks*, 10(6):1382–1391.

CMU-Repository (1993). ftp://ftp.cs.cmu.edu/afs/cs.cmu.edu/project/connect/bench/. WWW Repository, Carnegie Mellon University.

Coelho, A., Weingaertner, D., and von Zuben, F. J. (2001). Evolving Heteregeneous Neural Networks for Classification Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 266–273, San Francisco. Morgan Kaufmann.

Cohen, M. A. and Grossberg, S. (1983). Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks. *IEEE Transactions on Sytems, Man, and Cybernetics*, 13(5):815–826.

Cun, Y. L., Denker, J., and Solla, S. (1990). Optimal Brain Damage. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2, pages 599–605. Morgan Kaufmann.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

Darwin, C. (1872). *The Origins of Species.* Collier Books.

Davidor, Y. (1990a). Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383.

Davidor, Y. (1990b). Sub–Goal Reward and Lamarckism in a Genetic Algorithm. In *Proceedings of the European Conference on Artificial Intelligence.*

Davis, L. (1989). Adapting Operator Probabilities in Genetic Algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Mateo, California. Philips Laboratories, Morgan Kaufmann Publishers, Inc.

Davis, L., editor (1991). *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York. ISBN 0-442-00173-8.

de Garis, H., Korkin, M., Gers, F., Nawa, E., and Hough, M. (2000). Building an artificial brain using an FPGA based CAM–Brain Machine. *Applied Mathematics and Computation*, 111(2–3):163–192.

DeJong, K. (1975). *The Analysis and Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan.

Domingos, P. (1999). The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425.

Eaton, M. and Collins, J. J. (1996). A Comparison of Encoding Schemes for Genetic Algorithms. In *World Congress on Neural Networks*, pages 1067–1070. International Neural Network Society, Lawrence Erlbaum Associates, Inc.

Edelman, G. M. (1987). *Neural Darwinism – The Theory of Neuronal Group Selection.* Basic Books, New York.

Edwards, D., Brown, K., and Taylor, N. (2002). An Evolutionary Method for the Design of Generic Neural Networks. In *Proceedings of the 2002 World Congress on Computational Intelligence, Congress on Evolutionary Computation*, pages 1769–1774. IEEE.

Eigen, M. (1971). Selforganization of Matter and the Evolution of Biological Macro–Molecules. *Die Naturwissenschaften*, 58(10):465–523.

Fahlman, S. and Lebiere, C. (1990). The Cascade-Correlation Learning Architecture. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532. Morgan Kaufmann.

Fahlman, S. E. (1988). Faster learning variations on back–propagation. In *Proceedings of the 1988 Connectionist Models Summer School*, pages 38–51, San Mateo, CA. Morgan Kaufmann.

Field, P. (1996). *A Multary Theory for Genetic Algorithms: Unifiying Binary and Nonbinary Problem Representations.* PhD thesis, University of London.

Fletcher, R. (1987). *Practical Methods of Optimization.* Wiley, New York, 2nd edition.

Fogel, D. (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* IEEE Press, New York.

Fogel, D. B. (1991). *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling.* Ginn Press, Needham Heights.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution.* Wiley, New York.

Forrest, S. (1985). Documentation for Prisoner's Dilemma and Norms Programs that Use the Genetic Algorithm. Technical report, The University of New Mexico, Albuquerque, NM.

Fraser, A. S. (1957). Simulation of Genetic Systems by Automatic Digital Computers, I. Introduction. *Australian Journal of Biological Science*, 10:492–499.

Freeman, W. J. (1994). Neural networks and chaos. *Journal of Theoretical Biology*, 171:13–18.

Friedrich, C. M. and Moraga, C. (1996). An Evolutionary Method to Find Good Building Blocks for Architectures of Artificial Neural Networks. In *Proceedings of the Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge–Based Systems*, pages 951–956.

Futuyma, D. J. (1990). *Evolutionsbiologie*. Birkhäuser Verlag, Basel.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1994). *PVM 3 User's Guide and Reference Manual*. Oak Ridge National Laboratory.

Goldberg, D. E. (1983). *Computer–Aided Pipeline Operation Using Genetic Algorithms and Rule Learning*. PhD thesis, University of Michigan.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.

Goldberg, D. E. and Deb, K. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In Rawlins, G., editor, *Foundations of Genetic Algorithms*, San Mateo, CA. Morgan Kaufmann.

Goldberg, D. E. and Lingle, R. (1985). Alleles, Loci, and the Traveling Salesman Problem. In Grefenstette, J. J., editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 154–159. Texas Instruments, Inc. and Naval Research Laboratory, Lawrence Erlbaum Associates.

Grönroos, M. (1999). A Comparison of Some Methods for Evolving Neural Networks. In *Proceedings of the GECCO'99 Genetic and Evolutionary Computation Conference*, pages 1442–1449, San Francisco, CA. Morgan Kaufmann.

Gruau, F. (1992). Genetic Synthesis of Boolean Neural networks with a Cell Rewriting Developmental Process. In Whitley, D. and Schaffer, J. D., editors, *Proceedings of the Third International Workshop on Combinations of Genetic Algorithms and Neural Networks*, pages 55–74, Los Alamitos, California. IEEE Computer Society Press.

Hancock, P. J. B. (1992). Genetic Algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In Whitley, D., editor, *Proceedings of COGANN Workshop at the International Joint Conference on Neural Networks*. IEEE Press.

Hanson, S. and Pratt, L. (1989). Comparing biases for minimal network construction with back-propagation. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 1, pages 177–185. Morgan Kaufmann.

Harp, S. A., Samad, T., and Guha, A. (1989). Towards the genetic synthesis of neural networks. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 360–369, San Mateo, California. Philips Laboratories, Morgan Kaufmann.

Hasenjäger, M. and Ritter, H. (2002). Active learning in neural networks. In Jain, L. C. and Kacprzyk, J., editors, *New learning paradigms in soft computing*, pages 137–169. Physica–Verlag GmbH, Heidelberg, Germany.

Hassibi, B., Stork, D., and Wolff, G. (1993). Optimal Brain Surgeon and General Network Pruning. In *Proceeding IEEE International Conference on Neural Networks, San Francisco*, pages 293–299.

Haykin, S. (1999). *Neural Networks. A Comprehensive Foundation.* Prentice–Hall, Upper Saddle River, 2nd edition.

Hebb, D. O. (1949). *The Organization of Behavior.* Wiley, New York.

Hillis, W. D. (1990). Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234.

Hinton, G. (1986). Learning Distributed Representations of Concepts. *Proceeding of the Eighth Annual Conference of the Cognitive Science Society.*

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* MIT Press.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, 81:8429–8433.

Hornby, G. S. and Pollack, J. B. (2001). Body-Brain Co-evolution Using L-systems as a Generative Encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 868–875, San Francisco. Morgan Kaufmann.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366.

Huber, R. and Mayer, H. A. (1998). ERC – Evolutionary Resample and Combine for Adaptive Parallel Training Data Set Selection. In Jain, A. K., Venkatesh, S., and Lovell, B. C., editors, *14th International Conference on Pattern Recognition*, pages 882–885, Los Alamitos, California. International Association for Pattern Recognition, IEEE Computer Society.

Huber, R., Mayer, H. A., and Schwaiger, R. (1995). netGEN - A Parallel System Generating Problem-Adapted Topologies of Artificial Neural Networks by means of Genetic Algorithms. In *Beiträge zum 7. Fachgruppentreffen Maschinelles Lernen der GI-Fachgruppe 1.1.3, Forschungsbericht Nr. 580, Dortmund.*

Huber, R., Schwaiger, R., and Mayer, H. A. (1996). On the Role of Regularization Parameters in Fitness Functions for Evolutionary Designed Artificial Neural Networks. In *World Congress on Neural Networks*, pages 1063–1066. International Neural Network Society, Lawrence Erlbaum Associates, Inc.

Ishiguro, A., Tokura, S., Kondo, T., Uchikawa, Y., and Eggenberger, P. (1999). Reduction of the Gap between Simulated and Real Environments in Evolutionary Robotics: A Dynamically–Rearranging Neural Network Approach. In *IEEE Systems, Man, and Cybernetics Conference*, pages III – 239–244. IEEE.

Jacobs, R. A. (1988). Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks*, 1(4):295–307.

Jain, A. and Zongker, D. (1997). Feature Selection: Evaluation, Application and Small Sample Performance. *IEEE Transactions on PAMI*, 19(2):153–158.

Kallel, L. and Schoenauer, M. (1997). Alternative Random Initialization in Genetic Algorithms. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 268–275, San Francisco, CA. Morgan Kaufmann.

Kelly, K. (1994). *Out of Control - The New Biology of Machines.* Fourth Estate Ltd., London.

Kim, H. B., Jung, S. H., Kim, T. G., and Park, K. H. (1996). Fast Learning Method for Back–Propagation Neural Network by Evolutionary Adaptation of Learning Rates. *Neurocomputing*, 11(1):101–106.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, 4:461–476.

Kohonen, T. (1972). Correlation Matrix Memories. *IEEE Transactions on Computers*, C–21:353–359.

Kohonen, T. (1982). Self-organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43:59–69.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by means of Natural Selection.* Complex Adaptive Systems. The MIT Press, Cambridge, MA.

Krön, B., Steinsky, B., Strapetz, M., and Mayer, H. A. (2000). On the Number of Neural Networks. Technical Report, University of Graz (Department of Mathematics), University of Salzburg (Department of Computer Science), Austria.

Kwok, T.-Y. and Yeung, D.-Y. (1997). Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems. *IEEE Transactions on Neural Networks*, 8(3):630–645.

Kyngäs, J. and Hakkarainen, J. (1996). Predicting sunspot numbers with evolutionarily optimized neural networks. In *Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications*, pages 173–180.

la Maza, M. D. and Tidor, B. (1993). An analysis of selection procedures with particular attention paid to proportional and boltzmann selection. In Stephanie Forrest, U. o. N. M., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 124–131. University of Illinois at Urbana-Champaign, Morgan Kaufmann.

Lee, C.-H. and Kim, J.-H. (1996). Evolutionary Ordered Neural Network with a Linked–List Encoding Scheme. In Whitley, D., editor, *International Conference on Evolutionary Computation*, pages 665–669.

Lettvin, J. Y., Maturana, H. R., McCulloch, W. S., and Pitts, W. R. (1959). What the Frog's Eye Tells the Frog's Brain. *Proc. I.R.E.*, 47(11):1940–1951.

Likothanassis, D., Georgopoulos, E., and Fotakis, D. (1997). Optimizing the Structure of Neural Networks Using Evolution Techniques. In *Proceedings of 5th International Conference on Applications of High–Performance Computers in Engineering*, pages 157–168.

Lindenmayer, A. (1968). Mathematical Models for Cellular Interaction in Development. Parts I and II. *Journal of Theoretical Biology*, 18:280–299; 300–315.

Lippmann, R. P. (1987). An introduction to computing with neural nets. *IEEE Acoustics, Speech and Signal Processing*, 4(2):4–22.

Liu, Y. and Yao, X. (1996). Evolutionary Design of Artificial Neural Networks with Different Nodes. In *Proceedings of the Third IEEE International Conference on Evolutionary Computation*, pages 570–675.

Lodish, H., Baltimore, D., Berk, A., Zipursky, S. L., Matsudaira, P., and Darnell, J. (1995). *Molecular Cell Biology.* Scientific American Books, 3rd edition. ISBN 07167-2380-8.

Lovelock, J. (1988). *The Ages of Gaia: A Biography of Our Living Earth.* W. W. Norton.

Lund, H. H., Webb, B., and Hallam, J. (1997). A Robot Attracted to the Cricket Species Gryllus bimaculatus. In Husbands, P. and Harvey, I., editors, *Proceedings of Fourth European Conference on Artificial Life*, pages 246–255, MA. MIT Press/Bradford Books.

Mayer, H. A. (1998a). ptGAs–Genetic Algorithms Evolving Noncoding Segments by Means of Promoter/Terminator Sequences. *Evolutionary Computation*, 6(4):361–386.

Mayer, H. A. (1998b). Symbiotic Coevolution of Artificial Neural Networks and Training Data Sets. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Fifth International Conference Parallel Problem Solving from Nature*, pages 511–520, Berlin, Germany. Springer.

Mayer, H. A., Fürlinger, K., and Strapetz, M. (1999). Extraction of Compact Rule Sets from Evolutionary Designed Artificial Neural Networks. In *IEEE Systems, Man, and Cybernetics Conference*, pages I – 420–424. IEEE.

Mayer, H. A. and Huber, R. (1998). Evolutionary Training Data Sets with n–dimensional Encoding for Neural InSAR Classifiers. In Arabnia, H. R., editor, *International Conference on Imaging Science, Systems and Technology*, pages 161–168. CSREA, CSREA Press.

Mayer, H. A., Huber, R., and Schwaiger, R. (1996). Lean Artificial Neural Networks - Regularization Helps Evolution. In *Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications*, pages 163–172.

Mayer, H. A., Schmidbauer, J., and Stieglbauer, G. (2001a). EMMA – Architecture and Subsystems of a Mobile Autonomous Robot with a Preference for Soccer. In *Advances in Signal Processing, Robotics and Communications (RODLICS 2001)*, pages 310–316. WSES Press.

Mayer, H. A. and Schwaiger, R. (1997). Towards the Evolution of Training Data Sets for Artificial Neural Networks. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 663–666. IEEE Press.

Mayer, H. A. and Schwaiger, R. (1999). Evolutionary and Coevolutionary Approaches to Time Series Prediction Using Generalized Multi-Layer Perceptrons. In *Congress on Evolutionary Computation*, pages 275–280. IEEE.

Mayer, H. A. and Schwaiger, R. (2002). Differentiation of Neuron Types by Evolving Activation Function Templates for Artificial Neural Networks. In *Proceedings of the 2002 World Congress on Computational Intelligence, International Joint Conference on Neural Networks*, pages 1773–1778. IEEE.

Mayer, H. A., Somol, P., Huber, R., and Pudil, P. (2000). Improving Statistical Measures of Feature Subsets by Conventional and Evolutionary Approaches. In *Proceedings of the Joint IAPR International Workshops SSPR 2000 and SPR 2000*, pages 77–86. Springer.

Mayer, H. A., Strapetz, M., and Fuchs, R. (2001b). Simultaneous Evolution of Structure and Activation Function Types in Generalized Multi-Layer

Perceptrons. In *Proceedings of the 2001 WSES International Conference on Neural Networks and Applications*, pages 349–354. WSES Press.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Mehlhorn, K. and Naher, S. (2000). *Leda: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press.

Menczer, F. and Parisi, D. (1992). Recombination and Unsupervised Learning: Effects of Crossover in the Genetic Optimization of Neural Networks. *Network: Computation in Neural Systems*, 3(4):423–442.

Mendes, R., Cortez, P., Rocha, M., and Neves, J. (2002). Particle Swarms for Feedforward Neural Network Training. In *Proceedings of the 2002 World Congress on Computational Intelligence, International Joint Conference on Neural Networks*, pages 1895–1899. IEEE.

Michalewicz, Z. (1991). *Genetic Algorithms + Data Structures = Evolution Programs*. Artificial Intelligence. Springer, Berlin.

Michalewicz, Z., Dasgupta, D., Riche, R. G. L., and Schoenauer, M. (1996). Evolutionary Algorithms for Constrained Engineering Problems. *Computers & Industrial Engineering Journal*, 30(2):851–870.

Miller, G. F., Todd, P. M., and Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 379–384, San Mateo, California. Philips Laboratories, Morgan Kaufman Publishers, Inc.

Minsky, M. L. and Papert, S. A. (1988). *Perceptrons: Introduction to Computational Geometry*. MIT Press, Expanded edition.

Moriarty, D. and Miikkulainen, R. (1993). Evolving Complex Othello Strategies Using Marker–Based Genetic Encoding of Neural Networks. Technical Report AI93–206, The University of Texas at Austin, Department of Computer Sciences, Austin, TX 78712–1188.

Moriarty, D. and Miikkulainen, R. (1995). Discovering Complex Othello Strategies Through Evolutionary Neural Networks. *Connection Science*, 7(3–4):195–209.

Moriarty, D. E. and Miikkulainen, R. (1998). Hierarchical Evolution of Neural Networks. In *Proceedings of the 1998 IEEE Conference on Evolutionary Computation*, pages 428–433, Piscataway, NJ. IEEE.

Nolfi, S. (1997). Using emergent modularity to develop control system for mobile robots. *Adaptive Behavior*, 5(3–4):343–364.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics – The Biology, Intelligence, and Technology of Self–Organizing Machines.* MIT Press.

Nolfi, S., Miglino, O., and Parisi, D. (1994). Phenotypic Plasticity in Evolving Neural Networks. In *Proceedings of the First Conference From Perception to Action.* IEEE Computer Society Press.

Paredis, J. (1994). Steps towards Co–evolutionary Classification Neural Networks. In Brooks, R. and Maes, P., editors, *Proceedings Artifical Life IV*, pages 545–552. MIT Press / Bradford Books.

Patterson, D. W. (1996). *Artificial Neural Networks – Theory and Applications.* Prentice–Hall, Singapore.

Plutowski, M. (1994). *Selecting Training Examplars for Neural Network Learning.* PhD thesis, University of California, San Diego.

Potter, M. A. and De Jong, K. A. (2000). Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8(1):1–29.

Prechelt, L. (1994). PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report TR 21/94, Universität Karlsruhe, Fakultät für Informatik, 76128 Karlsruhe, Germany.

Radi, A. and Poli, R. (1999). Evolutionary Discovery of Learning Rules for Feedforward Neural Networks with Step Activation Function. In *Proceedings of the GECCO'99 Genetic and Evolutionary Computation Conference*, pages 1178–1183, San Francisco, CA. Morgan Kaufmann.

Reed, R. (1993). Pruning Algorithms - A Survey. *IEEE Transactions on Neural Networks*, 4(5):740–747.

Reeves, C. R. and Taylor, S. J. (1998). Selection of Training Data for a Neural Network by a Genetic Algorithm. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *5th International Conference Parallel Problem Solving from Nature*, pages 633–642, Berlin. Springer.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA.

Röbel, A. (1994). Dynamic Pattern Selection: Effectively training Backpropagation Neural Networks. In *Proceedings of the International Conference on Artificial Neural Networks, ICANN'94.*

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Rosin, C. D. and Belew, R. K. (2000). New Methods for Competitive Coevolution. *Evolutionary Computation*, 5(1):1–29.

Rosin, P. L. and Fierens, F. (1995). Improving Neural Network Generalisation. In *Proceedings of IGARSS'95, Firenze, Italy.*

Rueda, L. and Oommen, B. J. (2000). The Foundational Theory of Optimal Bayesian Pairwise Linear Classifiers. In *Proceedings of Joint IAPR International Workshops SSPR 2000 and SPR 2000 (LNCS 1876)*, pages 581–590. Springer.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 318–362. MIT Press.

Rumelhart, D. E., Widrow, B., and Lehr, M. A. (1994). The Basic Ideas in Neural Networks. *Communications of the ACM*, 37(3):87–92.

Sastry, K. and Goldberg, D. E. (2001). Modeling Tournament Selection With Replacement Using Apparent Added Noise. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 781, San Francisco, CA. Morgan Kaufmann.

Schaffer, J. D. and Morishima, A. (1987). An Adaptive Crossover Distribution Mechanism for Genetic Algorithms. In Grefenstette, J. J., editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 36–40. Naval Research Laboratory, Lawrence Erlbaum Associates.

Schwaiger, R. and Mayer, H. A. (1997). Genetic Algorithms to Create Training Data Sets for Artificial Neural Networks. In Alander, J. T., editor, *3rd Nordic Workshop on Genetic Algorithms and their Applications*, pages 153–161. Finnish Artificial Intelligence Society.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Sixth–Generation Computer Technology Series. Wiley, New York.

Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by Committee. In *Proceedings of the Fifth Workshop on Computational Learning Theory*, pages 287–294, San Mateo, CA. Morgan Kaufmann.

Shepherd, G. M. (1994). *Neurobiology*. Oxford University Press, 3rd edition.

Sherrington, C. (1940). *Man on his Nature*. Oxford University Press, Oxford.

Siddiqi, A. A. and Lucas, S. M. (1998). A comparison of matrix rewriting versus direct encoding for evolving neural networks. In *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pages 392–397, Piscataway, NJ. IEEE Press.

Smith, J. M. (1989). *Evolutionary Genetics*. Oxford University Press, New York. ISBN 0–19–854215–1.

Smith, R. E., Goldberg, D. E., and Earickson, J. A. (1991). SGA-C: A C–Language Implementation of a Simple Genetic Algorithm. TCGA Report 91002, The Clearinghouse for Genetic Algorithms, The University of Alabama, Department of Engineering Mechanics, Tuscaloosa, AL 35487.

Smith, T. M. C. and Philippides, A. (2000). Nitric Oxide Signalling in Real and Artificial Neural Networks. *British Telecom Technology Journal*, 18(4):140–149.

Sopena, J. M., Romero, E., and Alquézar, R. (1999). Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, pages 323–328.

Spears, W. M. and Anand, V. (1991). A Study of Crossover Operators in Genetic Programming. In Ras, Z. W. and Zemankova, M., editors, *Proceedings of the Sixth International Symposium on Methodologies for Intelligent Systems*, pages 409–418, Berlin. Springer.

Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, California. Philips Laboratories, Morgan Kaufman Publishers, Inc.

Thierens, D. (1996). Non-Redundant Genetic Coding of Neural Networks. In Whitley, D., editor, *International Conference on Evolutionary Computation*, pages 571–575.

Tononi, G. and Edelman, G. M. (1998). Consciousness and Complexity. *Science*, 282(5395):1846–1851.

Törn, A. and Žilinskas, A. (1990). Global Optimization. *Lecture Notes in Computer Science*, 350.

van Laarhoven, P. J. M. and Aarts, E. H. L. (1989). *Simulated Annealing: Theory and Applications*. Kluwer Academic Publishers Group, Dordrecht.

Vecci, L., Piazza, F., and Uncini, A. (1998). Learning and approximation capabilities of adaptive spline activation function neural networks. *Neural Networks*, 11(2):259–270.

Weingaertner, D., Tatai, V. K., Gudwin, R. R., and Zuben, F. J. V. (2002). Hierarchical Evolution of Heterogeneous Neural Networks. In *Proceedings of the 2002 World Congress on Computational Intelligence, Congress on Evolutionary Computation*, pages 1775–1780. IEEE.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, Cambridge, MA.

Widrow, B. and Hoff, M. E. (1960). Adaptive Switching Circuits. In *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record*, pages (4):96–104.

Willshaw, D. J. and von der Malsburg, C. (1976). How patterned neural connections can be set up by self–organization. *Proceedings of the Royal Society London*, B194:431–445.

Wolpert, D. H. and Macready, W. G. (1996). No Free Lunch Theorems for Search. Technical Report SFI–TR–95–02–010, The Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM, 87501.

Yao, X. (1993). Evolutionary Artificial Neural Networks. *International Journal of Neural Systems*, 4(3):203–222.

Yao, X. (1999). Evolving Artificial Neural Networks. *Proceedings of the IEEE*, 87(9):1423–1447.

Yao, X. and Liu, Y. (1997). A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 8(3):694–713.

Yu, T. and Bentley, P. (1998). Methods to Evolve Legal Phenotypes. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Fifth International Conference on Parallel Problem Solving from Nature*, pages 280–291, Berlin, Germany. Springer.

Zell, A., Mamier, G., Vogt, M., Mach, N., Huebner, R., Herrmann, K.-U., Soyez, T., Schmalzl, M., Sommer, T., Hatzigeogiou, A., Doering, S., and Posselt, D. (1994). *SNNS Stuttgart Neural Network Simulator, User Manual.* University of Stuttgart.

Zhang, B.-T. (1993). Self–development learning: Constructing optimal size neural networks via incremental data selection. Tech. Rep. No. 768, German National Research Center for Computer Science, Sankt Augustin.

Zhang, B.-T. (1994). Accelerated Learning by Active Example Selection. *International Journal of Neural Systems*, 5(1):67–75.

Zhang, B.-T. and Veenker, G. (1991). Neural Networks that Teach Themselves through Genetic Discovery of Novel Examples. In *Proceedings International Joint Conference on Neural Networks (IJCNN '91)*, pages 690–695. IEEE Press.

Ziemke, T., Carlsson, J., and Bodén, M. (1999). An Experimental Comparison of Weight Evolution in Neural Control Architectures for a 'Garbage-Collecting' Khepera Robot. In *Proceedings of the 1st International Khepera Workshop*. HNI–Verlagsschriftenreihe.