High Order and Multilayer Perceptron Initialization

G. Thimm and E. Fiesler

Abstract— **Proper initialization is one of the most important prerequisites for fast convergence of feed-forward neural networks like high order and multilayer perceptrons. This publication aims at determining the optimal variance (or range) for the initial weights and biases, which is the principal parameter of random initialization methods for both types of neural networks.**

An overview of random weight initialization methods for multilayer perceptrons is presented. These methods are extensively tested using eight real-world benchmark data sets and a broad range of initial weight variances by means of more than 30,000 simulations, in the aim to find the best weight initialization method for multilayer perceptrons.

For high order networks, a large number of experiments (more than 200,000 simulations) was performed, using three weight distributions, three activation functions, several network orders, and the same eight data sets. The results of these experiments are compared to weight initialization techniques for multilayer perceptrons, which leads to the proposal of a suitable initialization method for high order perceptrons.

The conclusions on the initialization methods for both types of networks are justified by sufficiently small confidence intervals of the mean convergence times.

Keywords— **neural network initialization, random weight initialization, initial weight, weight initialization, initial weight distribution, comparison of weight initialization methods, high(er) order neural network, high(er) order perceptron, sigma-pi connection, activation function, learning rate, multilayer perceptron, optimization, connectionism, real-world benchmark**

I. Introduction

THE LEARNING SPEED of multilayer and high order perceptrons[1] depends mainly on the initial values of its weights and biases, its learning rate, its network topology, and on learning rule improvements like the momentum term. The optimal values for these parameters are usually unknown *a priori* because they depend mainly on the training data set used. In practice it is not feasible to perform a global search for obtaining the optimal values of these parameters, as the convergence behavior of the network might change significantly for small changes in the initial weights, as was demonstrated by J. F. Kolen and J. B. Pollack [3]. An extensive search for the optimum values requires therefore much more overhead than performing a relatively small number of simulations using non-optimal values. Furthermore, current mathematical techniques are insufficient for a complete theoretical study of the learning behavior of these neural networks. Nevertheless, it is important to have a good approximation of the optimal initial value of these parameters; or with the words of J. F. Kolen and J. B. Pollack: to start the learning process in the "eye of the storm," to reduce the required training time.

Several weight initialization methods for multilayer perceptrons have been suggested. The simplest method among them is random weight initialization, which is often preferred for its simplicity and its ability to produce multiple solutions, as the weights may, due to their initial randomness, converge to various attractors [3]. Other methods involve extensive statistical and/or geometrical analysis of the data and are therefore very time consuming. The most rigorous among those is the pseudo-inverse method for perceptrons, which, besides being limited to linear separable data, has several other drawbacks (see [4]). Some other weight initialization methods are based on special properties of a network that can not be applied to high order or multilayer perceptrons, as for example the weight initialization technique for radial basis function networks by J. C. Platt [5].

D. E. Rumelhart *et al.* observed that if all weights in a neural network are initialized with zero, they have the tendency to assume identical values during training. They therefore proposed random weight initialization to avoid this undesired situation by *breaking the symmetry* [6]. However, the efficiency of this method depends much on the initial weight distribution. Several researchers therefore proposed random weight initialization methods. An overview of these methods is presented in section II, and their performance is evaluated in section IV-B.1.

In order to obtain a thorough insight in the initialization characteristics of high order networks, which have not been studied before, numerous experiments were performed, varying the following parameters:

- the shape of the initial weight distribution: uniform, normal or Gaussian[2], and a novel distribution which is uniform over the intervals $[-2a, -a]$ and $[a, 2a]$, and zero everywhere else,
- the variance of the initial weight distribution,
- the order and topology of the network, and
- the activation function.

The results of these experiments is a simple weight initialization method using an application independent variance. In section V, this method is compared to methods developed for multilayer perceptrons, in order to profit from experiences of other researchers and to determine a best method for higher order perceptrons.

The authors are with IDIAP, P.O. box 592, CH-1920 Martigny, Switzerland; E-mail Georg.Thimm@idiap.ch

[1] For a definition of high order neural networks and associated references see [1] and [2].

[2] Neural network weights are often assumed to be normally distributed [7].

II. Weight Initialization Techniques for Multilayer Perceptrons

S. E. FAHLMAN PERFORMED an early experimental study on the random weight initialization scheme for multilayer perceptrons. Based on this study, he proposed to use a uniform distribution with a range of $[-1.0, 1.0]$, but found that the best weight range for the data sets in his study varied from $[-4.0, 4.0]$ to $[-0.5, 0.5]$ [8].

Other researchers tried to determine the optimal weight range using network parameters:

L. Bottou uses an interval $[-a/\sqrt{d_{in}}, a/\sqrt{d_{in}}]$, where a is chosen is such way that the weight variance corresponds to the points of the maximal curvature of the activation function (which is approximately 2.38 for a standard sigmoid), and d_{in} is the fan-in (or in-degree) of a neuron, without justifying this interval further in a theoretical manner. L. Bottou trains the neural network only on speech data and does not compare this method with others [9].

J. W. Boers and H. Kuiper initialize weights using a uniform distribution over the interval $[-3/\sqrt{d_{in}}, 3/\sqrt{d_{in}}]$, without any mathematical justification. They state that this interval performed the best on their speech data [10].

F. J. Śmieja uses uniformly distributed weights which are normalized to the magnitude $2/\sqrt{d_{in}}$ for each node. The thresholds of the hidden units are then initialized to a random value in the interval $[-\sqrt{d_{in}}/2, \sqrt{d_{in}}/2]$ and the thresholds of the output nodes are set to zero. He obtained these values from reasoning about hyperplane spin dynamics, and did not validate his method by experiments [11].

L. F. A. Wessels and E. Barnard describe two weight initialization methods. The first method sets the initial weight range to a value which assumes that the output of the network and the output patterns have the same variance. The second method puts equally distributed decision boundaries in the input space (without considering input or output patterns), which produces initial weights for the first interlayer weight matrix. The weights of the second interlayer weight matrix are set to 1.0. They compared both methods on generalization for three data sets. They found that the second method performed better in terms of generalization. However, they did not compare convergence speeds [12].

An approach similar to the first method of L. F. A. Wessels and E. Barnard was introduced by G. P. Drago and S. Ridella [13]. They aim at avoiding flat regions in the error surface by restricting the number of neurons with absolute activations greater than 0.9. They developed a simple formula to estimate the best weight initialization scheme for multilayer perceptrons and showed for three data sets that this scheme uses satisfactory good initial weight ranges. The weights are uniformly distributed over the interval $[-a, a]$, with $a = 1.3/\sqrt{1 + E[x^2]}$ for the input layer and $a = 1.3/\sqrt{1 + 0.3d_{in}}$ for the output layer (assuming that all input values x have the same expected value E).

Y. Lee, S.-H. Oh, and M. W. Kim showed theoretically that the probability of prematurely saturated neurons (small weight changes cause only negligible changes of the neuron output) in multilayer perceptrons increases with the maximal value of weights. They conclude that a smaller initial weight range increases the learning speed of multilayer perceptrons. Simulations performed using two data sets confirm their reasoning, but they disregard that learning speed also decreases for weight ranges that are too small. Y. Lee *et al.* do not suggest an optimal weight range [14].

P. Haffner, A. Waibel, H. Sawai, and K. Shikano use a normal initial weight distribution. Unfortunately they do not compare their approach to others, give details, or justify it mathematically [15].

R. L. Watrous and G. M. Kuhn compared a Gaussian distribution to a uniform distribution and found differences on the conditioning of the Jacobian matrix of a neural network, but found no relation to the convergence speed [16].

D. Nguyen and B. Widrow use a multilayer perceptron with piecewise linear activation functions as an approximation of a network with logistic activation functions. Based on this simplification, they calculated an optimal length of $\sqrt[d_{in}]{N_2}$ for the randomly initialized weight vectors and an optimal bias range of $[-\sqrt[d_{in}]{N_2}, \sqrt[d_{in}]{N_2}]$ for neurons in the hidden layer, where N_2 is the number of hidden nodes. The weights of the neurons in the output layer are randomly initialized in the interval $[-0.5, 0.5]$, without any justification given [17].

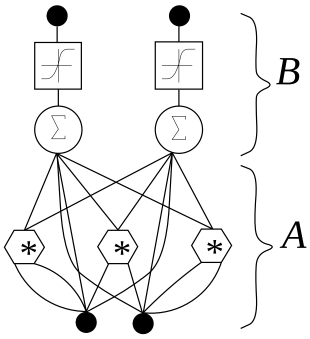
Y. K. Kim and J. B. Ra calculated a lower bound for the initial length of the weight vector of a neuron to be $\sqrt{\eta/d_{in}}$, where η is the learning rate [18].

Besides these random weight initialization methods, some non-random methods are described here for completeness.

A mixture between a random weight initialization scheme and the pseudo inverse method was developed by C.-L. Chen and R. S. Nutter for perceptrons with one hidden layer. First, the weights in the first interlayer weight matrix of the network are initialized with random values. Then, the weights in the second interlayer weight matrix are calculated using the pseudo inverse method applied to the activation values of the hidden layer. C.-L. Chen *et al.* refined this technique further by alternating the adjustment of the first interlayer weight matrix in a backpropagation-like process with the mentioned method of calculating the second interlayer weight matrix. These adjustments are repeated until a convergence criterion is reached, after which the backpropagation training begins. The authors report faster training in number of backpropagation cycles [19], but they disregard the computational complexity of the matrix inversions.

T. Denoeux and R. Lengellé initialize a one hidden layer perceptron with prototypes. This method requires a transformation of the input patterns to vectors of unit length and increased size. Additionally, prototypes have to be found by a cluster analysis. The authors reported improvements in training time, robustness versus local minima, and better generalization [20].

Fig. 1. A two-layer high order perceptron.



III. High Order Perceptrons

HIGH ORDER PERCEPTRONS are high order neural networks [21], having only unidirectional interlayer connections. They are also a generalization of Sigma-Pi networks [6], which are multilayer perceptrons having high order connections, and functional link networks [22].

A high order connection connects a set of neurons in one layer to neurons in the next layer (marked A in figure 1). Each connection applies its specific *splicing function* to the activation values of the lower layer. The number of activation values combined by the splicing functions determines the order of the connection and the connection with the highest order determines the order ω of the network. A network is called a *full (n-th order) network*, if all possible interlayer connections up to this order are present. The network shown in figure 1 is, for example, a full two layer second order network.

The results of the splicing functions are fed, together with the activation values of the lower layer, into the next layer of neurons (marked B). Each neuron consists of a summation unit and an activation function (depicted by a Σ and a symbolized function, respectively).

High order perceptrons can be trained using the backpropagation algorithm, with possible extensions such as a momentum term [6] or flat spot elimination [8].

From now on in this publication, only two layer high order perceptrons are considered. The splicing function used in this study is multiplication, but other functions are also conceivable.

IV. The Simulations

A SIMULATION CONSISTS of initializing a neural network and applying the online backpropagation algorithm, alternated by convergence tests, until (non-)convergence is observed. A number of simulations starting with the same initial conditions is called an experiment.

The experiments are performed with two major aims: firstly, to see whether the performance of a network changes for different types of initial weight distributions, and secondly to find the optimal initial weight variance, depending on the activation function of the output neurons and the network order.

Each experiment consists of at least 50 simulations. The number of simulations per experiment was increased until the size of the 95% confidence interval for the mean convergence time permitted a sound conclusion. The confidence intervals were calculated under the assumption that the mean convergence time is student-t distributed.

For the simulations performed, a suboptimal learning rate is used, as it is too laborious and computing time consuming to find the optimal learning rate for each combination of data set and network, and as the learning rate and initial weight variance seem to have an independent influence on the learning speed. Because of the suboptimal learning speed, the results do not necessarily allow a comparison between different activation functions and experiments reported elsewhere, as the maximal possible learning rate may differ largely from the one actually used. For example, a third order network has, for the solar data with a shifted/scaled logistic output function, an optimal learning rate of about 0.05. In contrast, the same network and data set, except for using now a standard logistic output function, has an optimal learning rate of about 4.0, and converges in about the same number of iterations.

Some of the convergence criteria chosen in these simulations are rather crude and not related to the task to be solved. This is done in the aim to reduce the high computational expense, which still was several months of Sparc 10 CPU time.

A. The Data Sets

Most of the data sets used, and shortly described below, are obtained (if not stated otherwise) from an anonymous-ftp server at the University of California [23], which also contains further references and documentation. In the following list of data sets, the two numbers in brackets behind the name of the data set are the number of input and output values, respectively.

Solar (12,1) contains the sun spot activity for the years 1700 to 1990. The task is to predict the sun spot activity for one of those years, given the activity of the preceding twelve years ($\hat{=}$12 real valued inputs). The data are scaled to the interval $[0,1]$ and the networks trained to a mean square error of 0.06 (0.08 for first order perceptrons).

Wine (13,3) is the result of a chemical analysis of wines grown in a region in Italy derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. A wine has to be classified using these values, which are scaled to the interval $[0,1]$. The output patterns use boolean values, encoded as $+1$ and -1 and the trained networks are required to correctly classify 90% of the patterns (80% for first order perceptrons).

CES (2,1) is the output of the constant elasticity of a substitution production function for thirty pairs of labor and capital input (see [24], pages 195 and 210). The patterns have two real valued inputs and one real valued output, none of them scaled and the networks are trained to a mean square error of 0.08 (0.14 for first order perceptrons).

Servo (12,1) was created by Karl Ulrich (MIT) in 1986 and contains a very non-linear phenomenon: predicting the rise time of a servomechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages. The input is coded into two groups of five boolean values each, and two discrete inputs, one assuming four, the other five values. The output is real valued, and like all real valued inputs, scaled to the interval $[0, 1]$. The networks are trained until a mean square error of 0.08 is reached (0.14 for first order perceptrons).

Vowels (20,5) is a subset of 300 patterns of the vowels data set, obtainable via ftp from cochlea.hut.fu (130.233.168.48) with the LVQ-package (lvq_pak). An input pattern consists of 20 unscaled cepstral coefficients obtained from continuous Finnish speech. The task is to determine whether the pronounced phoneme is a vowel, and, in the case it is, which of the five possible ones. The boolean output values are encoded as $+1$ and -1 and the networks are required to recognize 90% of the patterns (65% for first order perceptrons with a linear sigmoidal function, 80% for other first order perceptrons).

Auto (7,1) concerns city-cycle fuel consumption of cars in miles per gallon, to be predicted in terms of 3 multivalued discrete and 4 continuous attributes. All values are scaled to the interval $[0, 1]$ (incomplete patterns have been removed) and the networks are trained to a mean square error of 0.06.

Glass (9,1) consists of 8 scaled weight percentages of certain oxides and a 7 valued code for the type of glass (window glass, head lamps etc.). The output is the refractive index of the glass, scaled to $[0, 1]$ and the networks trained to a mean square error of 0.03 (0.04 for simple perceptrons).

Digits (256,10) consists of 500 handwritten digits (50 patterns for each of the ten digits) of the *NIST Special Database 3* [25]. Each digit was scaled to fit into an image of 16x16 points, and each pixel is represented by an eight bit value. The input values are scaled to the interval $[-1, 1]$ and the boolean output values are encoded as $+1$ and -1 and the networks are required to classify 95% of the patterns correctly.

B. The Experiments for Multilayer Perceptrons

In the aim to validate and compare the performance of the random weight initialization techniques for multilayer perceptrons mentioned in section II, a large number of experiments has been performed using the data sets listed in the previous section. The network topology used has one hidden layer which is fully interlayer connected to both input and output layer. The network has no intralayer or supralayer connections, and all activation functions in the hidden and output layer are hyperbolic tangents. No optimization technique was used for training.

For each data set, a sequence of experiments with uniform initial weight distributions of a varying variance were performed (100 simulations per experiment). The outcome of these experiments was used to determine the overall best weight variance as a reference for comparing the random weight initialization schemes of L. Bottou, F. J. Śmieja, G. P. Drago *et al.*, and D. Nguyen *et al.* It should be noted that D. Nguyen *et al.* do not seem to use a bias in the output layer. However, neither [17] nor the references mentioned in it state why. To make the simulations fair (leaving out the bias makes learning more difficult), a bias is used in the all simulations reported in this publication.

Fig. 2. Performance of multilayer perceptrons for a uniform weight distribution over the interval $[-a, a]$, with $a = \sqrt{3\sigma^2}$.

	N_2	η	10E-6 to 0.001	0.005	0.01	0.05	0.1	0.2	0.5	0.7	1.0	2.0	3.0	4.0
Solar	5	0.3	∘	∘	∘	∘	∘	∘	∘	●	×	∘	×	×
Wine	6	0.2	∘	∘	∘	∘	∘	∘	∘	∘	∘	●	×	×
CES	4	0.1	∘	∘	∘	∘	∘	∘	○	●	∘	×	×	×
Servo	3	0.1	∘	∘	∘	○	○	●	∘	∘	∘	∘	∘	×
Vowels	20	0.05	∘	∘	∘	∘	∘	∘	●	○	∘	×	×	×
Auto	3	0.3	∘	∘	∘	○	○	●	∘	∘	∘	×	×	×
Glass	6	0.6	∘	∘	∘	○	●	∘	∘	∘	∘	∘	∘	∘
Digits	30	0.3	∘	○	●	∘	∘	∘	×	×	×	×	×	×

B.1 The Results for Multilayer Perceptrons

The outcome of the experiments for the multilayer perceptrons are shown in tables 2 and 3. These tables list in the first three columns the name of the data set, the number of neurons in the hidden layer N_2, and the learning rate η. The subsequent columns, labeled with the initial weight variances in table 2 and names in table 3, respectively, contain the outcome of the experiments. The names in table 3 refer to the random weight initialization schemes described in section II. A '●' in these columns corresponds to the smallest mean number of required online learning cycles until convergence (an online learning cycle is a presentation of all patterns with a weight update after each presentation), a '○' represents a mean which is too close to the best value to statistically conclude that this value is worse. A '∘' means that the network always converged but in a significantly longer time. A '×' means that some or all simulations in an experiment did not converge in the prescribed number of itterations (which was set to at least three times the smallest number needed for convergence).

The best fixed value for a initial weight variance averaged over these experiments is 0.2, but this value is not necessarily the optimal value for a specific data set.

The on average best method for the determination was presented by L.F.A. Wessels.

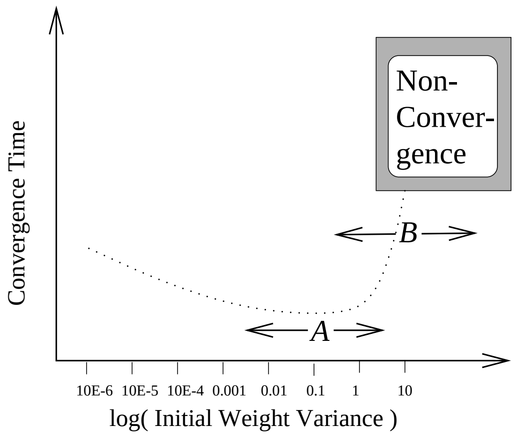
B.2 Analysis of the simulations for Multilayer Perceptrons

The average convergence behavior of a multilayer perceptron is depicted in figure 4, where region A indicates the optimum initial weight variances that have been encountered and region B non-convergence. As the curve is flatter on the left side of the optimal initial weight variance than on the right, the loss in performance is much

Fig. 3. Random weight initialization with the methods of other authors.

	N_2	η	Bottou	Wessels	Śmieja	Drago	Nguyen
Solar	5	0.3	∘	◯	∘	∘	∘
Wine	6	0.2	∘	◯	◯	∘	∘
CES	4	0.1	∘	◯	∘	∘	∘
Servo	3	0.1	◯	◯	∘	∘	∘
Vowels	20	0.05	∘	◯	∘	∘	∘
Auto	3	0.3	◯	●	◯	∘	◯
Glass	6	0.6	◯	•	∘	∘	∘
Digits	30	0.3	◯	∘	∘	∘	●

Fig. 4. The average behavior of a multilayer perceptron in convergence speed for changing the initial weight variance.



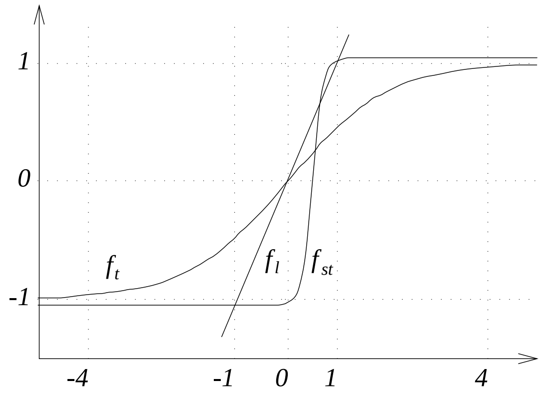
more tolerable for initial weight variances smaller than the optimal value as compared to bigger variances. Moreover, non-convergence was only encountered for simulations using initial weight variances bigger than the optimal value.

The rather small differences obtained for an initial weight variance of 0.2 as compared to the optimal result, suggests to use this value for a simple weight initialization method. A comparison between the results for this simple method and table 3 shows that the weight initialization method of L. F. A. Wessels *et al.*, which uses the same weight variances as the method of J. W. Boers, performs the best[3]. This is also valid compared to an initial weight variance of 0.2, with a saving of between 1% and 50% training cycles.

Remark: Some of the weight initialization methods presented in section II scale the upper and lower bound of the initial random weight interval by the reciprocal square root of the fan-in. This corresponds to scaling the initial weight variance by the reciprocal of the fan-in. Hence, these methods assume a negative correlation between the fan-in of a neuron and the best initial weight variance. This correlation can not be confirmed or rejected by the results of the experiments; more experiments with other data sets are necessary to do so.

[3] The raw data have not been included for readability of the paper

Fig. 5. The activation functions.



$$\begin{aligned} f_l(x) &= x \\ f_t(x) &= \frac{2}{1+e^{-x}} - 1 = tanh(\frac{x}{2}) \\ f_s(x) &= 1.05\ tanh(5.05x - 2.54) \end{aligned}$$

C. The Experiments with High Order Perceptrons

The networks used in the simulations are usually full and the biases are initialized with a random value of the same distribution as the weights. The only exception is the network trained on the digits data set. This network includes all first order connections and only second order connections with both inputs corresponding to different pixels in the same row or the same column in the image. This configuration should allow the extraction of sufficient features to learn the digits. Training sessions on the in section IV-A described digits data set gave an acceptable recognition of untrained digits, despite the small training set used.

The three different initial random weight distributions used are: uniform on the interval $[-a, a]$ (with $a = \sqrt{3\sigma^2}$), normal (restricted to an absolute value of $3\sigma^2$), and uniform over the intervals $[-2a, -a]$ and $[a, 2a]$ (with $a = 3\sigma^2/7$) while zero everywhere else. The three types of activation functions used are: a linear f_l, a hyperbolic tangent f_t, and a scaled/shifted hyperbolic tangent f_s, shown in figure 5. The use of the function f_s was motivated by several ideas: the scaling in the direction of the y-axis prevents the weights from becoming very big and thus cause the same effect as for example scaling the output data to $[-0.9, 0.9]$ and the change of the steepness and the shifting of the sigmoid in the direction of the x-axis were used to force the outcome of the summation step in the neurons to be in the interval $[0, 1]$. Also, experiments with this activation function where performed to see, whether a relation between a deformation of the activation function and the optimal initial weight range exists. The only optimization technique applied to speed up learning is flat-spot elimination.

Fig. 6. Performance of high order perceptrons for a uniform weight distribution over the interval $[-a, a]$, with $a = \sqrt{3\sigma^2}$.

	ω	f	η	10E-6	0.0001	0.001	0.005	0.01	0.05	0.1	0.2	0.5	0.7	1	2
Solar	1	l	0.1	●	○	●	○	∘	∘	∘	∘	∘	∘	∘	
	2	l	0.1	○	●	○	∘	∘	∘	∘	∘	∘	∘	∘	
	3	l	0.1	●	●	∘	∘	∘	∘	∘	∘	∘	∘	∘	
	1	t	0.7	●	●	●	●	●	∘	∘	∘	∘	∘	∘	
	2	t	0.7	∘	●	○	∘	∘	∘	∘	∘	∘	∘	∘	
	3	t	0.7	●	●	○	○	○	∘	∘	∘	∘	∘	∘	
Wine	1	t	0.5	∘	∘	∘	∘	∘	∘	∘	∘	∘	○	○	●
	2	t	0.5	∘	∘	∘	∘	∘	∘	∘	∘	●	×	×	×
	1	s	0.02	●	○	○	∘	∘	×	×	×	×	×	×	
	2	s	0.02	●	●	●	○	∘	×	×	×	×	×	×	
CES	1	l	0.1	○	○	○	○	○	●	∘	∘	∘	∘	∘	
	2	l	0.1	○	●	○	∘	∘	∘	∘	∘	∘	∘	∘	
	1	t	0.2	∘	∘	∘	∘	○	○	∘	●	∘	∘	∘	
	2	t	0.2	●	○	●	●	○	○	∘	∘	∘	∘	×	
	1	s	0.01	●	○	∘	○	∘	×	×	×	×	×	×	
	2	s	0.01	○	●	∘	∘	∘	×	×	×	×	×	×	
Servo	1	l	0.01	○	●	○	∘	∘	∘	∘	∘	∘	∘	∘	
	2	l	0.01	○	●	●	○	∘	∘	∘	∘	∘	∘	∘	
	1	t	0.03	∘	∘	∘	∘	∘	●	○	∘	∘	∘	∘	
	2	t	0.03	●	●	●	●	●	∘	∘	∘	∘	∘	∘	
Vowels	2	l	0.03	●	∘	∘	∘	∘	∘	∘	∘	∘	∘	∘	
	1	t	0.5	∘	∘	∘	∘	∘	∘	○	○	●	○	∘	
	2	t	0.5	○	○	∘	○	○	○	○	●	∘	∘	×	
	1	s	0.03	○	●	○	○	∘	∘	×	×	×	×	×	
	2	s	0.03	●	∘	∘	×	×	×	×	×	×	×	×	
Auto	2	l	0.03	○	●	○	∘	∘	∘	∘	∘	∘	×	×	
	2	t	0.1	●	○	●	∘	○	∘	∘	∘	∘	∘	∘	
Glass	1	l	0.1	○	●	●	●	●	∘	∘	∘	∘	∘	∘	
	2	l	0.005	●	●	∘	∘	∘	∘	×	×	×	×	×	
Digits	2r	l	2E-4	●	∘	∘	∘	∘	×	×	×	×	×	×	

Fig. 7. Performance of high order perceptrons for a normal distribution restricted to the interval $[-3\sigma^2, 3\sigma^2]$.

	ω	f	η	10E-6	0.0001	0.001	0.005	0.01	0.05	0.1	0.2	0.5	0.7	1	2
Solar	1	l	0.1	●	●	∘	●	∘	∘	∘	∘	∘	∘	∘	
	2	l	0.1	○	●	○	○	○	∘	∘	∘	∘	∘	∘	
	3	l	0.1	●	●	●	∘	∘	∘	∘	∘	∘	∘	∘	
	1	t	0.7	○	○	○	○	●	○	∘	∘	∘	∘	∘	
	2	t	0.7	○	○	○	●	●	○	∘	∘	∘	∘	∘	
	3	t	0.7	○	●	○	○	○	∘	∘	∘	∘	∘	∘	
Wine	1	t	0.5	∘	∘	∘	∘	∘	∘	∘	∘	∘	∘	○	●
	2	t	0.5	∘	∘	∘	∘	∘	∘	∘	∘	∘	∘	●	×
	1	s	0.02	○	○	●	○	○	∘	×	×	×	×	×	
	2	s	0.02	●	●	∘	○	○	∘	×	×	×	×	×	
CES	1	l	0.1	●	●	●	●	○	●	●	∘	∘	∘	∘	
	2	l	0.1	○	○	○	○	●	∘	∘	∘	∘	∘	∘	
	1	t	0.2	○	∘	○	○	○	○	○	○	●	∘	∘	
	2	t	0.2	○	○	○	○	○	●	∘	∘	∘	×	×	
	1	s	0.01	∘	∘	∘	●	∘	∘	∘	×	×	×	×	
	2	s	0.01	○	○	○	○	●	∘	∘	×	×	×	×	
Servo	1	l	0.01	∘	●	○	∘	∘	∘	∘	∘	∘	∘	∘	
	2	l	0.01	●	●	●	●	●	∘	∘	∘	∘	∘	∘	
	1	t	0.03	∘	∘	∘	∘	∘	●	●	∘	∘	∘	∘	
	2	t	0.03	○	○	○	○	○	●	∘	∘	∘	∘	∘	
Vowels	2	l	0.03	∘	∘	∘	●	∘	∘	∘	∘	∘	∘	∘	
	1	t	0.5	∘	∘	∘	∘	∘	∘	∘	○	○	●	○	
	2	t	0.5	∘	∘	∘	∘	∘	∘	●	○	∘	∘	∘	
	1	s	0.03	●	○	∘	○	∘	∘	∘	×	×	×	×	
	2	s	0.03	∘	●	∘	×	×	×	×	×	×	×	×	
Auto	2	l	0.03	○	○	○	●	●	∘	∘	∘	∘	×	×	
	2	t	0.1	○	●	○	○	○	∘	∘	∘	∘	∘	∘	
Glass	1	l	0.1	∘	○	●	○	○	∘	∘	∘	∘	∘	∘	
	2	l	0.005	●	●	●	○	∘	∘	∘	×	×	×	×	

C.1 The Results of the Simulations for High Order Perceptrons

Tables 6, 7, and 8 show, besides entries already explained in section IV-B.1, the order ω of the fully interlayer connected network and the activation function f. The entries in tables 7 and 8 with a gray background are those showing a significant difference with table 6.

C.2 Analysis of the Simulations for High Order Perceptrons

THE MINIMAL CONVERGENCE TIMES for all three initial weight distributions show no difference of statistical significance, mainly their corresonding initial weight variances differ. The average behavior of the learning time as a function of the initial weight variance, which is depicted in figure 9, is explained as follows. The main difference for the three distributions is the value of the weight variance where the convergence time starts increasing drastically. This "edge" (point A) is roughly at the same location for both the uniform distribution and the uniform distribution over two intervals, but slightly shifted to higher variances for the normal distribution. As the optimal weight variance (point B) is similarly displaced, the performance of two networks, initialized with two different weight distributions of the same variance, is difficult to compare. This might explain the better performance for a Gaussian initial weight distribution in the report of P. Haffner. For the various combinations of data set, network order, etc., the optimal weight variance was encountered in region D, whereas non-convergence was, if at all, only observed in region C.

As the three different initial weight distributions yield no significant difference in network performance, only the commonly used uniform distribution is considered from now on. For the shifted/scaled logistic and the linear activation functions, the best fixed weight variance is about 10^{-4} (which corresponds to an interval $[-0.017, 0.017]$). For the logistic activation function, the best value for the weight variance depends a lot on data set and network order. In general, the performance with optimal initial weight variance differs not much more than about 10% from the results obtained with a variance of 10^{-4} or even smaller. Therefore a variance of 10^{-4} may be used as a simple application independent random weight initialization scheme. This initialization scheme is also justified by a smaller risk: a network performs nearly as good for an initial weight variance smaller than the optimum. The loss in performance for choosing the initial weight variance too small is much less significant than it is for multilayer perceptrons.

Fig. 8. Performance of high order perceptrons for a uniform weight distribution over the intervals $[-2a,-a]$ and $[a,2a]$, with $a = \sqrt{3\sigma^2/7}$.

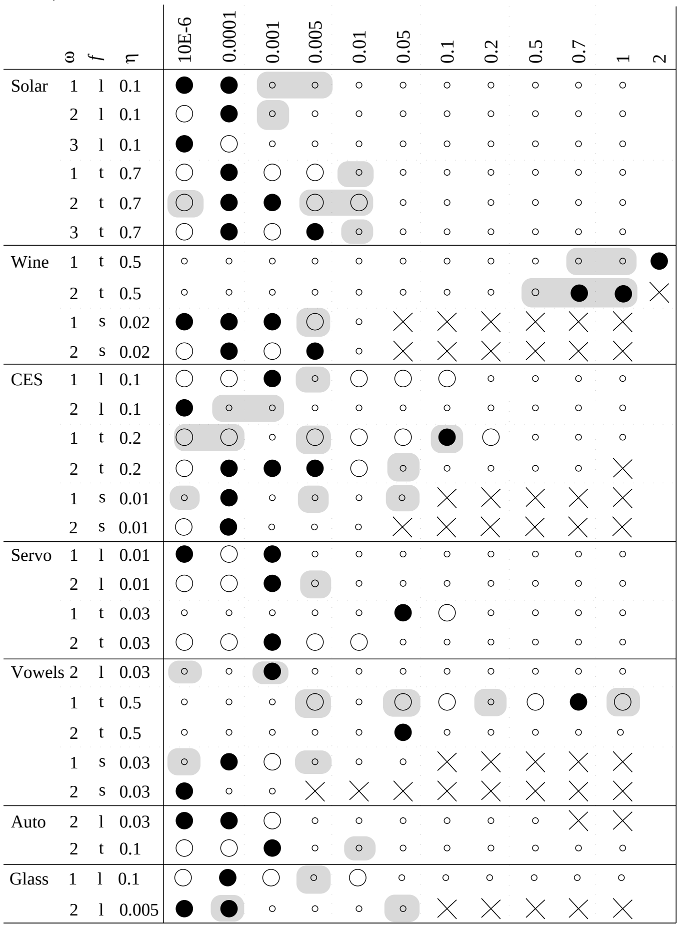
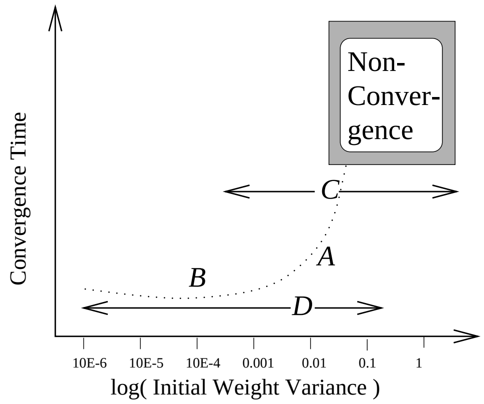


Fig. 9. Average behavior of a higher order perceptron in convergence time for changing initial weight variance.



The experiments confirm also that the data set itself has a large influence on the optimal initial weight variance: for the solar, wine, and servo data sets, the networks have about the same size for the same order, but the optimal value for the weight variance differs a lot for the network with the logistic activation function. Further, the optimal value for the initial weights remained for some data sets nearly unchanged for different net orders or even different activation functions, while it changes greatly for other sets. It remains the question, which attribute of the data sets causes this behavior.

V. Weight Initialization Techniques for Three Layer Perceptrons Applied to High Order Perceptrons

The most remarkable fact is that the rules based on observations (the "rules of thumb"), which seem to perform well for multilayer perceptrons, do not apply directly to high order perceptrons. Uniform weight distributions with an initial weight range of $[-1,1]$ (which corresponds to a variance of about 0.33) or bigger are definitely a poor choice for most of the examples considered. The mean convergence time is for some of the examples more than four times higher than the best initial weight range found or, even worse, they do not converge in a reasonable time.

The approaches using an interval $[-a/\sqrt{d_{in}}, a/\sqrt{d_{in}}]$ with a more or less arbitrary constant a, do not outperform the weight initialization with a fixed variance of 0.0001 or vice versa. Nevertheless, one would expect that more sophisticated methods for random weight initialization perform better than a scheme with fixed initial weight variance.

One such a sophisticated scheme was described by L. F. A. Wessels [12], and is here recalculated for a second order net with linear activation functions. The method tries to initialize the weights in such a way that the variance $\sigma^2_{\hat{y}}$ of the network output is equal to the expected variance σ^2_y of the target patterns.

A random weight initialization allows the calculation of the variance $\sigma^2_{\hat{y}}$ depending on the network topology (N_1 is the number of neurons in the input layer, w_i a weight, x_i an input value):

$$\begin{aligned}
\sigma^2_{\hat{y}} &= E[\hat{y}^2] - E^2[\hat{y}] = E[\hat{y}^2] \\
&\qquad \text{as weights and input values are independent} \\
&= E[\sum_{i,j=1}^{N_1} w_i w_j x_i x_j + \sum_{i,j,k,l=1}^{N_1} w_{ij} w_{kl} x_i x_j x_k x_l] \\
&= \sum_{i=1}^{N_1} E[w_i^2] E[x_i^2] + \sum_{i,j=1}^{N_1} E[w_{ij}^2] E[x_i^2 x_j^2] \\
&\qquad \text{as } E[w_{ij} w_{jk}] = 0 \text{ if } i \neq k \text{ or } j \neq l \\
&= E[w^2] (\sum_{i=1}^{N_1} E[x_i^2] + \sum_{i,j=1; i\neq j}^{N_1} E[x_i^2] E[x_j^2] + \sum_{i=1}^{N_1} E[x_i^4]) \\
&\qquad \text{as all weights have the same distribution}
\end{aligned}$$

$$= E[w^2]\left(\sum_{i=1}^{N_1}\frac{1}{12}+\sum_{i,j=1;i\neq j}^{N_1}\left(\frac{1}{12}\right)^2+\sum_{i=1}^{N_1}\frac{1}{80}\right)$$
$$= \frac{5{N_1}^2+64N_1}{720}E[w^2]$$

This assumes that all w_i and all x_i are independent random variables in the interval $[-1,1]$ and $[0,1]$ respectively, and therefore $E[x^2]=1/12$ and $E[x^4]=1/80$, as shown by the following equations:

$$\begin{aligned} E[x^2] &= \int_{-\infty}^{\infty}(x-\tfrac{1}{2})^2 F_{uniform\ [0,1]}dx \\ &= \int_0^1 (x-\tfrac{1}{2})^2 dx = \tfrac{1}{12} \\ E[x^4] &= \int_{-\infty}^{\infty}(x-\tfrac{1}{2})^4 F_{uniform\ [0,1]}dx \\ &= \int_0^1 (x-\tfrac{1}{2})^4 dx = \tfrac{1}{80} \end{aligned}$$

The variance σ_w^2 of a variable w is equal to the expectation of its squared value $E[w^2]$:

$$E[w^2] = \frac{\int_{-\infty}^{\infty} w^2 * F_{uniform,\sigma}(w)dw}{\int_{-\infty}^{\infty} F_{uniform,\sigma}(w)dw} = \frac{\int_{-\sqrt{3\sigma_w^2}}^{\sqrt{3\sigma_w^2}} w^2 dw}{\int_{-\sqrt{3\sigma_w^2}}^{\sqrt{3\sigma_w^2}} 1 dw}$$
$$= \frac{\frac{1}{3}\cdot 2\cdot \sqrt{3\sigma_w^2}^3}{2\cdot\sqrt{3\sigma_w^2}} = \sigma_w^2$$

Therefore the initial condition, equation $\sigma_{\hat{y}}^2=\sigma_y^2$, can be fulfilled by setting

$$\sigma_w^2 = \frac{720}{5{N_1}^2+64N_1}\sigma_y^2$$

Where σ_y^2 is equal to $1/3$ for boolean data and data which are scaled to the interval $[-1,1]$, and $1/12$ for data which are scaled to the interval $[0,1]$.

Similar formulas can easily be calculated for networks of different orders. The evaluation of these formulas gives an optimal initial weight variance between 0.3 and 0.02 for the examples considered in this publication. These values are usually much too high as compared to the results listed in the tables. No effort was therefore done to solve the same problems for the other (non-linear) activation functions.

For the activation functions f_t and f_s the heuristic of L. Bottou comes the closest to the optimal weight variance and may therefore be considered as being better. Comparing the variances suggested by his heuristic to table 6, one finds that his method may be improved by using about one tenth of the suggested variance.

VI. Conclusion

THE EXPERIMENTS SHOW that a suitable and convenient weight initialization method for high order perceptrons[4] with identity activation function[5] is a random initialization with a rather small variance of about 10^{-4} (which corresponds to a weight range of $[-0.017, 0.017]$). If a hyperbolic tangent is used as activation function, the best performance is obtained for the interval $[-a/\sqrt{d_{in}}, a/\sqrt{d_{in}}]$, where a is chosen such that the weight variance corresponds to one third of the distance between the points of the maximal curvature of the activation function[6] (this is approximately 0.8 for an unscaled hyperbolic tangent with steepness one). The "rules of thumb" which perform well for multilayer perceptrons are not suitable for high order perceptrons (which can be explained by their different topologies).

The steepness (and/or the horizontal shift) of the activation function has an important influence on the convergence time of high order perceptrons. A mathematical study, partly inspired by this research, showed that changes of the steepness of the activation function may be compensated for by adapting the initial weight range and the learning rate [26]. More precisely, two networks with the same topology converge in the same time and have the same response for every input if their learning rates η and $\hat{\eta}$, the steepness of their sigmoidal functions β and $\hat{\beta}$, and their weight vectors $\mathbf{w}$ and $\hat{\mathbf{w}}$ fulfill the condition

$$\frac{\beta}{\hat{\beta}} = \frac{\hat{\mathbf{w}}}{\mathbf{w}} = \sqrt{\frac{\hat{\eta}}{\eta}}.$$

This condition permits to extrapolate the weight initialization schemes to neural network implementations which for example do not use a standard sigmoidal function, as it is the case in optical implementations of neural networks (see [27]). However, for the weight initialization scheme of L. F. A. Wessels it can be shown that it already adapts correctly the initial weight range for changes of the steepness, as the point of maximal curvature is moving inverse linearly with the steepness of the sigmoidal function. But the condition says also that if the steepness of the sigmoidal function is changed, the learning rate needs to be adapted. More precisely, it has to be divided by $(\beta/\hat{\beta})^2$ in order to obtain the same convergence time as the network with the standard sigmoidal function. Thus the combination of the results in this paper and the adaption rule above are together a very powerful tool for the determination of the initial weight range of multilayer or high order perceptrons.

The large difference between the best initialization schemes found for the high order perceptrons and the multilayer perceptrons might be explained by the (non-)presence of a hidden layer. Other researchers observed a difference in the convergence time and best learning rate between multilayer perceptrons with one or two hidden layers, which can be understood as a modification of the initial weight range (see above), an indication that the number of hidden layers has an influence on the best initial weights.

On the other hand, the shape of the initial weight distribution of three rather different distributions showed no or only very little effect on the optimal convergence time of

[4] The results of this study apply in part to standard (first order) perceptrons, since high order perceptrons are a generalization of standard perceptrons.

[5] A linear activation function of steepness one.

[6] This results in about one tenth of the weight variance.

high order perceptrons. The main effect observed is a dislocation of the optimal value for the initial weight variance. There is consequently no preference for one of the three distributions as the optimal learning speeds are similar.

For multilayer perceptrons with one hidden layer, the weight initialization method of L. F. A. Wessels *et al.* performed on average the best, but the performance of the methods proposed by J. W. Boers *et al.*, L. Bottou, F. J. Śmieja, G. P. Drago *et al.*, and D. Nguyen *et al.* are nearly as good. An fixed weight variance of 0.2, which corresponds to a weight range of [-0.77, 0.77], gave the best mean performance for all the applications tested in this study. This performance is similar or better as compared to those of the other weight initialization methods. Despite the fact that the method of L. F. A. Wessels *et al.* and L. Bottou apply the same initial weight variances for the experiments performed for this publication, the first should be preferred, as it scales the initial weight variances depending on the activation function (due to the calculation of the network output variance).

The experiments show that the best initial weight variance for both types of neural networks is determined by the data set. Consequently, some reasoning on the data set has to be included in the determination of this value, if better values than those proposed in this publication are desired. On the other hand, an initial weight variance close to the optimal value is often acceptable, as the impact on the number of required learning cycles is not too big for small deviations in variance. In general, the loss in convergence speed for both types of neural networks is bigger when too high a variance is chosen than when too small a variance is chosen, as compared to the optimal value.

The evaluation of the experiments performed in order to find the best weight initialization scheme for high order and multilayer perceptrons includes the calculation of confidence intervals for the mean convergence time. This is a much more reliable measure than simply counting the number of simulations performed, as used in most other publications. Moreover, the simulations showed that some data sets require many more simulations to obtain a sufficiently small size of the confidence interval than others. In the experiments performed in this research, these numbers varied between 50 and 2,000. The authors encourage other researchers to report their results in a similar way.

The experiments were performed without any convergence time decreasing optimizations like the use of a momentum term or an adaptive learning rate since they involve constants which will further increase the dimension of the problem of best initial network configuration. However, the authors expect those techniques to have an insignificant influence on the optimal weight initialization variance (or range) since are designed to "short-cut" the trajectory the weights follow on the error surface towards the finally assumed minimum, which decreases the convergence time more or less proportionally to the non-optimized case.

Acknowledgments

The authors want to thank the Institute for Logic, Complexity, and Deduction Systems at the University of Karlsruhe for the supply of computing power, which allowed the performance of the simulations.

References

[1] E. Fiesler, "Neural network classification and formalization", *Computer Standards & Interfaces*, special issue on Neural Network Standardization, J. Fulcher, Ed., vol. 16, no. 3, pp. 231–239, June 1994, North-Holland/Elsevier. ISSN: 0920–5489

[2] G. Thimm, R. Grau, and E. Fiesler, "Modular object-oriented neural network simulators and topology generalizations", in *Proceedings of the International Conference on Artificial Neural Networks (ICANN'94)*, M. Marinaro and P. G. Morasso, Eds., London, U.K., 1994, vol. 1, pp. 747–750, Springer-Verlag. ISBN: 3-540-19887-3

[3] J. F. Kolen and J. B. Pollack, "Back propagation is sensitive to initial conditions", Tech. Rep. TR 90-JK-BPSIC, Laboratory for Artifical Intelligence Research, Computer and Information Science Department, 1990.

[4] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the theory of neural computation*, vol. I of *Computation and Neural Systems Series; Santa Fe Institute Studies in the Sciences of Complexity; Lecture notes*, Addison-Wesley Publishing Company, The Advanced Book Program, Redwood City, California, 1991. ISBN: 0-201-51560-1

[5] J. C. Platt, "Learning by combining memorization and gradient descent", in *Advances in Neural Information Processing Systems*, R. P. Lippman et al., Ed., San Mateo, 1991, vol. III, pp. 714–720, Morgan Kaufmann.

[6] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: Foundations, The MIT Press, Cambridge, Massachusetts, 1986. ISBN: 0-262-18120-7

[7] I. Bellido and E. Fiesler, "Do backpropagation trained neural networks have normal weight distributions?", in *ICANN'93; Proceedings of the International Conference on Artificial Neural Networks*, S. Gielen and B. Kappen, Eds., London, U.K., 1993, pp. 772–775, Springer-Verlag.

[8] S. E. Fahlman, "An empirical study of learning speed in back-propagation networks", Tech. Rep. CMU-CS-88-162, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, September 1988.

[9] L.-Y. Bottou, "Reconnaissance de la parole par reseaux multi-couches", in *Neuro-Nîmes'88; Proceedings of the International Workshop on Neural Networks and Their Applications*, 1988, EC2 and Chambre de Commerce et d'Industrie de Nîmes, pp. 197–217. ISBN: 2-906899-14-3

[10] E. J. W. Boers and H. Kuiper, "Biological metaphors and the design of modular artificial neural networks", Master's thesis, Leiden University, Leiden, The Netherlands, August 1992.

[11] F. J. Śmieja, "Hyperplane 'spin' dynamics, network plasticity and back-propagation learning", GMD report, GMD, St. Augustin, Germany, November 28, 1991.

[12] L. F. A. Wessels and E. Barnard, "Avoiding false local minima by proper initialization of connections", *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 899–905, November 1992.

[13] G. P. Drago and S. Ridella, "Statistically controlled activation weight initialization (scawi)", *IEEE Transactions on Neural Networks*, vol. 3, no. 4, pp. 627–631, July 1992.

[14] Y. Lee, S.-H. Oh, and M. W. Kim, "An analysis of premature saturation in back propagation learning", *Neural Networks*, vol. 6, pp. 719–728, 1993.

[15] P. Haffner, A. Waibel, H. Sawai, and K. Shikano, "Fast back-propagation learning methods for neural networks in speech", Technical Report TR-1-0058, ATR Interpreting Telephony Research Laboratories, 1988.

[16] R. L. Watrous and G. M. Kuhn, "Some considerations on the training of recurrent neural networks for time-varying signals", in *Second Workshop on Neural Networks for Speech Processing*, M. Gori, Ed., Trieste, Italy, 1993, Università di Firenze, pp. 5–17, Edizioni LINT Trieste S.r.l.

[17] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive

weights", in *Proceedings of the International Joint Conference on Neural Networks (IJCNN) San Diego*, Ann Arbor, Michigan, 1990, IEEE+INNS, vol. III, pp. 21–26, IEEE Neural Networks Council; Edward Brothers.

[18] Y. K. Kim and J. B. Ra, "Weight value initialization for improving training speed in the back propagation network", in *International Joint Conference on Neural Networks*. 1991, vol. 3, pp. 2396–2401, IEEE.

[19] C. L. Chen and R. S. Nutter, "Improving the training speed of three-layer feedforward neural nets by optimal estimation of the initial weights", in *International Joint Conference on Neural Networks*. 1991, vol. 3, pp. 2063–2068, IEEE.

[20] T. Denoeux and R. Lengellé, "Initializing back propagation networks with prototypes", *Neural Networks*, vol. 6, pp. 351–363, 1993.

[21] Y. C. Lee, G. Doolen, H. Chen, G. Sun, T. Maxwell, H. Lee, and C. L. Giles, "Machine learning using a higher order correlation network", *Physica D: Nonlinear Phenomena*, vol. 22, pp. 276–306, 1986. ISSN: 0167-2789

[22] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989. ISBN: 0-201-12584-6

[23] P. M. Murphy and D. W. Aha (Librarians), UCI Repository of machine learning databases [Machine-readable data repository], ftp access: ics.uci.edu:/pub/machine-learning-databases, 1994.

[24] G. G. Judge, W. E. Griffiths, R. C. Hill, and T.-C. Lee, *The Theory and Practice of Econometrics*, Wiley Series in Probability and mathematical statistics. John Wiley and Sons, 2 edition, 1985.

[25] M. D. Garris and R. A. Wilkinson, *NIST Special Database 3*, National Institute of Standarts and Technology, Advanced System Division, Image Recognition Group, February 1992.

[26] G. Thimm, P. Moerland, and E. Fiesler, "The Interchangeability of Learning Rate and Gain in Backpropagation Neural Networks", To appear in *Neural Computation*, 1996.

[27] I. Saxena and E. Fiesler, "Adaptive multilayer optical neural network with optical thresholding", *Optical Engineering*, vol. 34, no. 8, pp. 2435–2440, August 1995.