BackPercolation

ASSIGNING LOCAL ERROR IN FEEDFORWARD PERCEPTRON NETWORKS

Mark Jurik

SET ZOOM SCALE TO 134% … TO VIEW THE GRAPHICS CORRECTLY

DO NOT PRINT … AS GRAPHICS ARE DESIGNED TO BE VIEWED ON YOUR MONITOR

INTRODUCTION

The promise of fast, general purpose, nonlinear adaptive mapping functions useful in pattern recognition tasks (classification, identification, system control) has gained a resurgence in interest now that computer processing is relatively inexpensive. In various problems involving pattern recognition, the overall task may call for prediction, estimation, interpolation or extrapolation from the data at hand. To accomplish this, the analyst may try to model an unknown system or function.
Neurocomputing techniques can be applied to the system identification problem using adaptive algorithms for either parameter or functional estimation. In neurocomputing, data is typically processed by one or more homogeneous arrays of simple processors (also called processing elements or cells) possessing nonlinear behavior and operating in parallel. The arrangement of these cells constitutes a neural network.

One type of neural network is the multilayered feedforward perceptron net (MFP). Within it, data is modified by sequentially traversing in a strictly forward direction through one layer of cells after another, going through each layer only once. The data can be modified radically by each layer. As shown in figure 0, the first layer of processors may extract subfeatures that are then processed in the second bank to form even more abstract features, and so on, eventually leading to the desired output.

Multilayer feedforward perceptron networks with as few as a single hidden (intermediate) layer of processing elements (PE's) with appropriately smooth activation functions are capable of arbitrarily accurate approximations to arbitrary mappings [HSW1] and their derivatives. In fact, these networks can approximate functions that are only piecewise differentiable.[HSW2]. This may have numerous applications in system identification and control [See1],[NaPa], pattern recognition and classification [See2], and nonlinear filtering [Lipp]. Neural networks can be trained to perform a variety of different tasks, including credit card fraud detection [Schw], word error detection during decoding of block codes [JePr], sonar classification [GoSe], engine exhaust diagnostics[1], equities trading[2], and airline seating[3]. In general, perceptron networks map m-dimensional input data to n-dimensional output data, where typically m>n.

In all cases the parameters that govern the network's behavior need to be determined, either explicitly or adaptively. The nonlinear nature of perceptron networks makes the calculation of the network's parameters in a single-step very difficult. Therefore, adaptive techniques, whereby the network is repeatedly presented the input and desired output data patterns, are more popular.

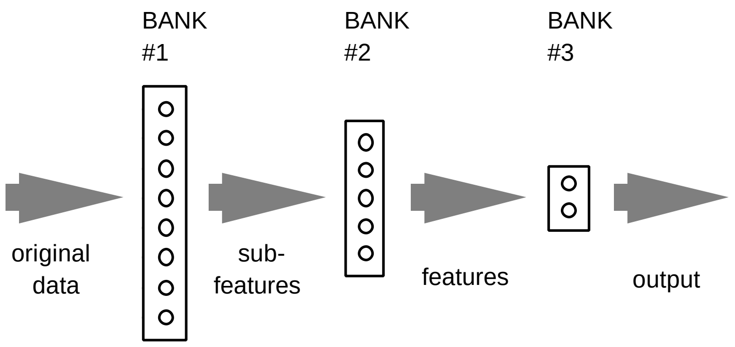

Figure 0. *In a MFP neural network, the input data can be transformed into increasingly abstract patterns.*

There exist many adaptive techniques for finding parameters that minimize a function. In perceptron networks those methods that modify estimates of each parameter within a PE by utilizing only information about the PE's parameters perform PE-local computations and are classified as <u>PE-local techniques</u>. Methods utilizing information about all the weights simultaneously (as in matrix inversion) are classified as <u>global techniques</u>. Other techniques could exist that fall between these two extremes. However, the more local the computations, the more easily one may anticipate its implementation in distributed, parallel processing hardware.

In addition to its mathematical simplicity, this may be another reason why BackPropagation [Werb],[Rume],[Gorm],[SeRo], a very local iterative technique, is the most popular method for training feedforward multilayer perceptron networks. However, the two biggest disadvantages of Back-Propagation (BP) include very slow learning and frequent inferior solutions when the problem to be solved is moderately complex or the number of nodes in the net becomes large. [Waib]

Numerous speed-up modifications to the BP algorithm exist, including momentum [Rume], Delta-Bar-Delta [Jaco], QuickProp [Fahl], conjugate gradients [KrSa], extended Kalman filtering [SiWu], Gain BP [GaSb], Gram-Schmidt signal decorrelation [Orph], and a new generation of self-reconfiguring feedforward networks [FaLe],[Ash]. The first four methods are relatively simple to execute and have delivered improvement in convergence speed for some applications, but as this report will show, the new proposed algorithm has performed better in all cases. The next two methods are significantly more complex and require matrix inversion, a non-local process that is considered disadvantageous when implementing the algorithm in distributed, parallel processing hardware. Signal decorrelation techniques may be advantageous but extensive benchmarking results are not available. Self-reconfiguring techniques, whereby the neural network adds more cells on an as-needed basis, may be impractical in real-time modeling of nonstationary processes because, as the process changes over time, the network adjusts by becoming ever larger without bounds.

The author proposes a new algorithm, called Backpercolation (Perc), for training feedforward perceptron networks. To a large extent, Perc satisfies the following useful constraints:

1. training stability does not degrade when there are many hidden layers,
2. training does not require non-local calculations (such as matrix inversion),
3. training does not automatically increase the number of elements in the architecture,
4. weights converge quickly toward attaining arbitrarily accurate output.

The basic Perc algorithm involves only PE-local calculations and does not employ momentum, matrix inversion, nested iterative searches or self-reconfiguration. However, Perc may be useful to self-reconfiguring perceptron networks when each reconfiguration requires weight retraining.

BENCHMARK COMPARISONS

This section covers experimental results of the Perc algorithm applied against the following types of benchmark training tasks: N-Bit Parity, N-M-N Encoder/Decoder, N-N-N Linear Channel, Multiplexer and Symmetry Detection. Perc's convergence speed on each training task is compared (in table 2) to the best published result from a collection of other perceptron training techniques as found in the technical literature. (See table 3.)

The N-Bit Parity task requires training a network to output the parity of the binary input vector. For example, 2-bit parity is equivalent to the XOR logic function on 2 input values. In this case, the network acepts two binary input values, a and b, and outputs the binary value c = (a XOR b). Table 1 maps out all the legitimate a,b,c combinations. In order to present standardized values to the network, the logic value 0 is replaced by -1.

a	b	c
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

Table 1 c = a XOR b

The N-M-N Encoder/Decoder task requires training a network to replicate its N binary input pattern with its N cell output. At any one time, only one input cell is set to true (true= +1) and the remaining N-1 cells are set to false (false= -1). For each of the N patterns to be presented to the network, a different cell is set true. The information in these patterns must squeeze through a hidden layer of only M cells, M<N. Kruglyak [Krug] has shown how to solve the N-2-N bit encoder problem

graphically for any value of N. However, whether or not a perceptron network could find the solution for very large N is unknown. The greater the ratio N:M, the more the network must compress the input data. Nonetheless, Perc can find a solution with 100% probability of success to the 12-2-12 arrangement in less than 600 epochs of training. To date, the 12-2-12 represents a compression ration of 6:1, which may be the largest ratio reportedly solved by a perceptron network through gradient-based weight adaptation.

For the N-N-N Linear Channel task, all three layers have N cells. The network is required to replicate any vector pattern of N continuously variable (floating point) input values with its N output cells, providing, of course, the randomized input values range between +1 and -1. Only the 10-10-10 arrangement has been seen in the technical literature by the author, wherein 100 randomized training patterns were presented in each epoch.

For the Multiplexer task, the input layer has N + 2^N cells and the output layer has only one cell. The N input lines can represent a binary encoding, or address, of any of the other 2^N input lines. The desired output of the network is the input to the cell designated by the address line. Only the 6-6-1 arrangement has been seen in the technical literature by the author.

For the Symmetry Detection task, the input layer has 2N cells and the output layer has only one cell. The network is to detect if the first N inputs are identical, in reverse arrangement, to the second N inputs. For example, the input pattern (0,0,1,1,0,0) would require an output of 1 (symmetry is found), whereas the input 0,0,1,0,1,1 would require an output of -1 (symmetry is not found). For Perc, the standardized input values were +1 and -1.

In the symmetry detection experiment, N=3. Consequently, there are only 2^N=8 uniquely symmetrical patterns and 2^{2N}=64 possible input combinations. As a result, a symmetric pattern would occur in training only 8/64 = 12.5% of the time. If not compensated for this, the network would learn to bias its output toward the decision of no symmetry in a manner similar to a Bayesian classifier. [RRKOS],[Wan] Therefore, when training with Perc, the author has arranged the training patterns to alternate between one that is symmetric and one that is not. This necessitated adding to the pattern database copies of the symmetric patterns in order to have equal number of symmetric/ nonsymmetric examples. This almost doubled the number of training patterns in an epoch from 64 to 120. In order to have a fair comparison of training speed, (which really should measure the total number of pattern presentations to a network), the posted number of epochs for Perc in table 2 is twice the actual average number of epochs that occured during experimentation.

In the following table, the CONFIG column contains the number of cells in the input, hidden and output layers respectively. The D term indicates that the network included direct weighted connections between the input and output cells. The G term indicates that the hidden cells warping function was gaussian rather than sigmoidal. The ERROR column contains the training cutoff error tolerance, whereupon the magnitude of the output error of each output cell for each training pattern must be less than the value stated for training to cease. The symbols a and b indicate the version of Backpercolation employed. The % column specifies the fraction of 50 training episodes that produced successful convergence (i.e., when error fell below the cutoff.) EPOCHS is the average number of training epochs required to get successful convergence. PARADIGM describes the training method used for comparison. The symbol ? in this column indicates that the author is not aware of any published report on this specific type/configuration/error combination, but wishes to publish Perc's results anyway for the reader's benefit.

TRAINING TASK		BACKPERCOLATION				BEST OTHER PARADIGM	
TYPE	CONFIG.	ERR	a,b	%	EPOCHS	EPOCHS	PARADIGM
Parity	2-2-1 XOR	0.1	a	100	**8**	**73**	Conjugate Gradient
	3-3-1	0.1	a	100	**18**	**247**	Gram Schmidt
	2-1-1 D	0.1	a	100	**8**	**24**	Cascade Corr. D,G
	3-1-1 D	0.1	a	100	**6**	**32**	Cascade Corr. D,G
	4-4-1 D,G	0.1	a	100	**5**	**66**	Cascade Corr. D,G
	8-8-1 D,G	0.1	a	100	**28**	**357**	Cascade Corr. D,G
	6-6-6-1	‡1	b	80	**93**	**5713**	Solis / Wets Optiimize
Encoder - Decoder	8-2-8	0.4	a	100	**88**	**103**	Quickprop
	8-2-8	0.1	a	100	**194**	**?**	?
	8-3-8	0.4	a	100	**22**	**22**	Quickprop
	8-3-8	0.1	a	100	**28**	**68**	Conjugate Gradient
	10-2-10	0.4	a	100	**246**	**?**	?
	10-2-10	0.1	a	100	**472**	**?**	?
	10-5-10	0.4	a	100	**15**	**14**	Quickprop
	10-5-10	0.1	a	100	**19**	**71**	Conjugate Gradient
	12-2-12	0.4	a	100	**545**	**?**	?
Linear Channel	10-10-10	0.1	a	100	**8**	**125**	Super-SAB
Multiplex	6-6-1	0.1	b	100	**24**	**137**	Delta-Bar-Delta
Symmetry	6-2-1	0.3	b	84	**2 x 62**	**198**	Gain Backpropagation

TABLE 2

Comparisons in convergence speed between Perc and other paradigms.

Note 1: The training cutoff occurred when the squared error of the output cell, summed across all the training patterns was less than 0.0002.

One final point: the tanh(μ) warping function used by Perc has a dynamic range [-1,+1] that is twice as large as that used by most networks, which is [0,+1]. For fair comparison of output error between these two ranges, the actual error (d_k - ϕ_k) of cells using tanh(μ) is scaled down by 50%. In this way, an output error of 0.2 from a cell with the larger range is equivalent, in relative terms, to an output error of 0.1 from a cell with the smaller range.

TRAINING METHOD	DESCRIPTION
Conjugate Gradient (CG)	The direction of weight adjustment combines both the gradient of the global error surface and the prior direction of adjustment, in such a way that the new direction is conjugate to the old direction. Uses the Polak-Ribiere method. [KrSa]
Gain Backpropagation (GBP)	Cost function to be mimimized is based on both current and past data. GBP requires updating and inverting an exponential time averaged auto-correlation matrix of the input data as each pattern is presented to the network. [GaSb]
Delta-Bar-Delta (DBD)	Each weight has its own learning rate that increases linearly as long as the weight's direction of change does not rapidly alternate, in which case learning rate decreases exponentially. [Jaco]
QuickProp (QP)	Each weight change moves the system state toward the center of a more shallow sloped error surface. Local valleys in the error surface are assumed to be parabolic, thereby permitting quick approximation of location of zero-slope region. [Fahl]
Super-SAB (SS)	Each weight has its own learning rate that increases exponentially as long as the weight's direction of change does not alternate, in which case learning rate decreases exponentially. Decrease rate is faster than increase rate. [Toll]
Cascade Correlation (CC)	Hidden layer cells are added one at a time and their weight values are not changed after the cell has been added. Input weights are set to maximize the covariance between new cell's output and network's output error. [FaLe]
Gram-Schmidt (GS)	Input data vector to each layer is first preprocessed by Gram-Schmidt orthogonalization. BackPropagation is modified so that gradients can travel backward through the Gram-Schmidt layers. [Orph]
Solis/Wets Optimization (SWO)	Weight vector for the entire network is modified by the random optimization method of Solis & Wets. [Baba]

Table 3

Brief description of the perceptron training methods that were compared to Backpercolation.

K. Miscellaneous properties

Tunnelling Through Local Minima

One of the biggest problems with using a strict gradient descent down the global error surface is that if the surface is nonlinear and full of local minima then the process could get stuck in a minima that yields poor network performance. Perc also uses gradient descent, but on the local level. Figures 12a and 12b suggest pretty strongly that the latter method allows the network to tunnel through minimas in the global error surface. Figure 12a shows how the absolute value of the error of the output cell, averaged across all training patterns in the 6-2-1 symmetry detection experiment, increased during training for 30 consecutive epochs before descending to a lower error. Figure 12b shows this phenomenon occuring twice.

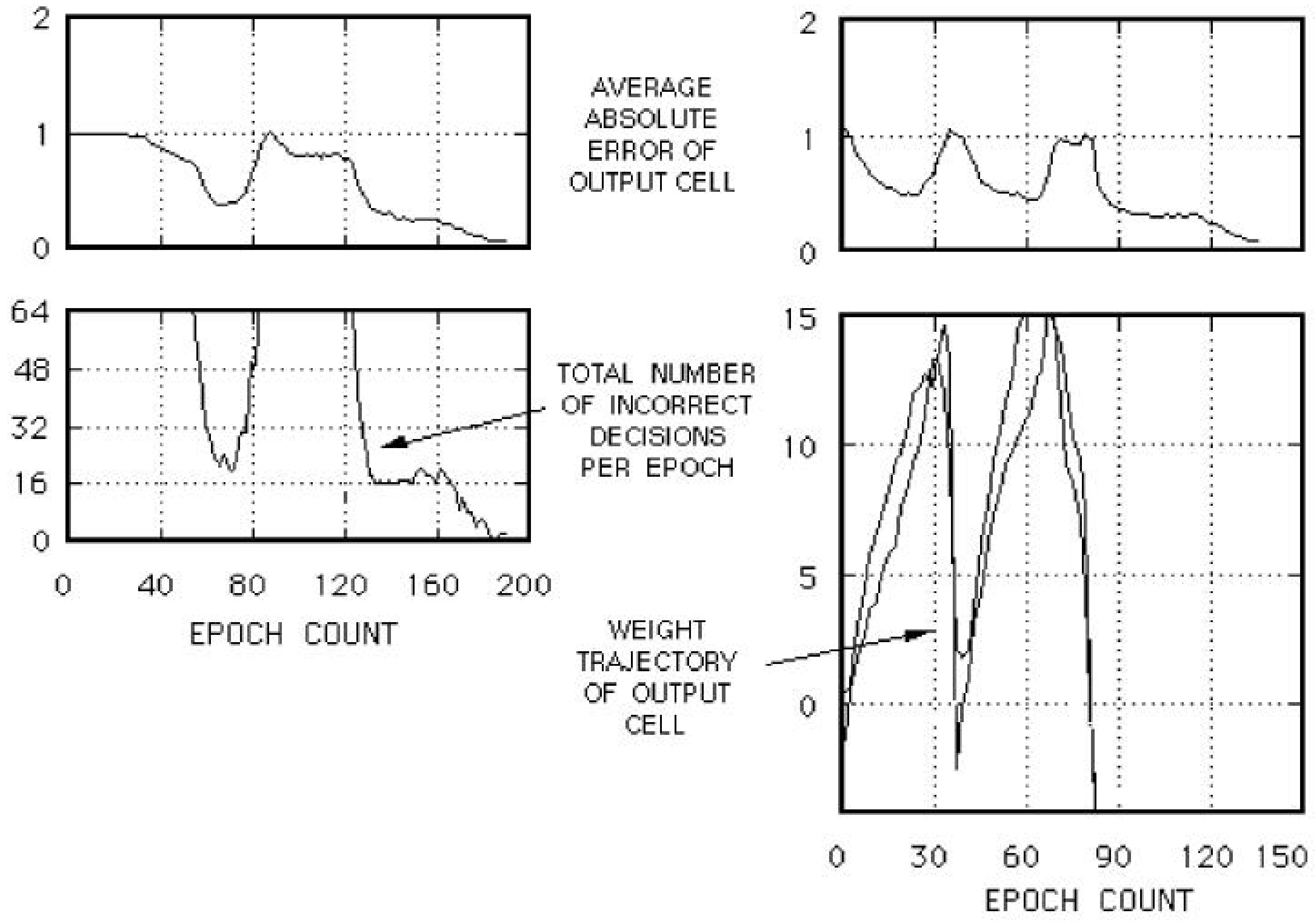

FIGURE 12a *Output of 6-2-1 network showing how it traversed up the global error surface for 30 consecutive epochs of training*.	**FIGURE 12b** *Output of 6-2-1 network showing how it twice traversed up the global error surface during training.*

Weight Trajectories

Fahlman and Lebiere [FaLe] suggest that one factor contributing to BP's slow convergence is what they call the "moving target problem". They state: *"because all the weights in the network are changing at once, each hidden unit sees a constantly changing environment. Therefore, instead of moving quickly to assume useful roles in the overall problem solution, the hidden units engage in a complex dance with much wasted motion*."

All the charts in figure 13 illustrate that weight trajectories under Perc training are engaged in very little wasted motion. Trajectories are fairly clean and display symmetry when there is symmetry in the problem itself. Much wasted motion does occur if the error amplification factor l is too large, causing instability.

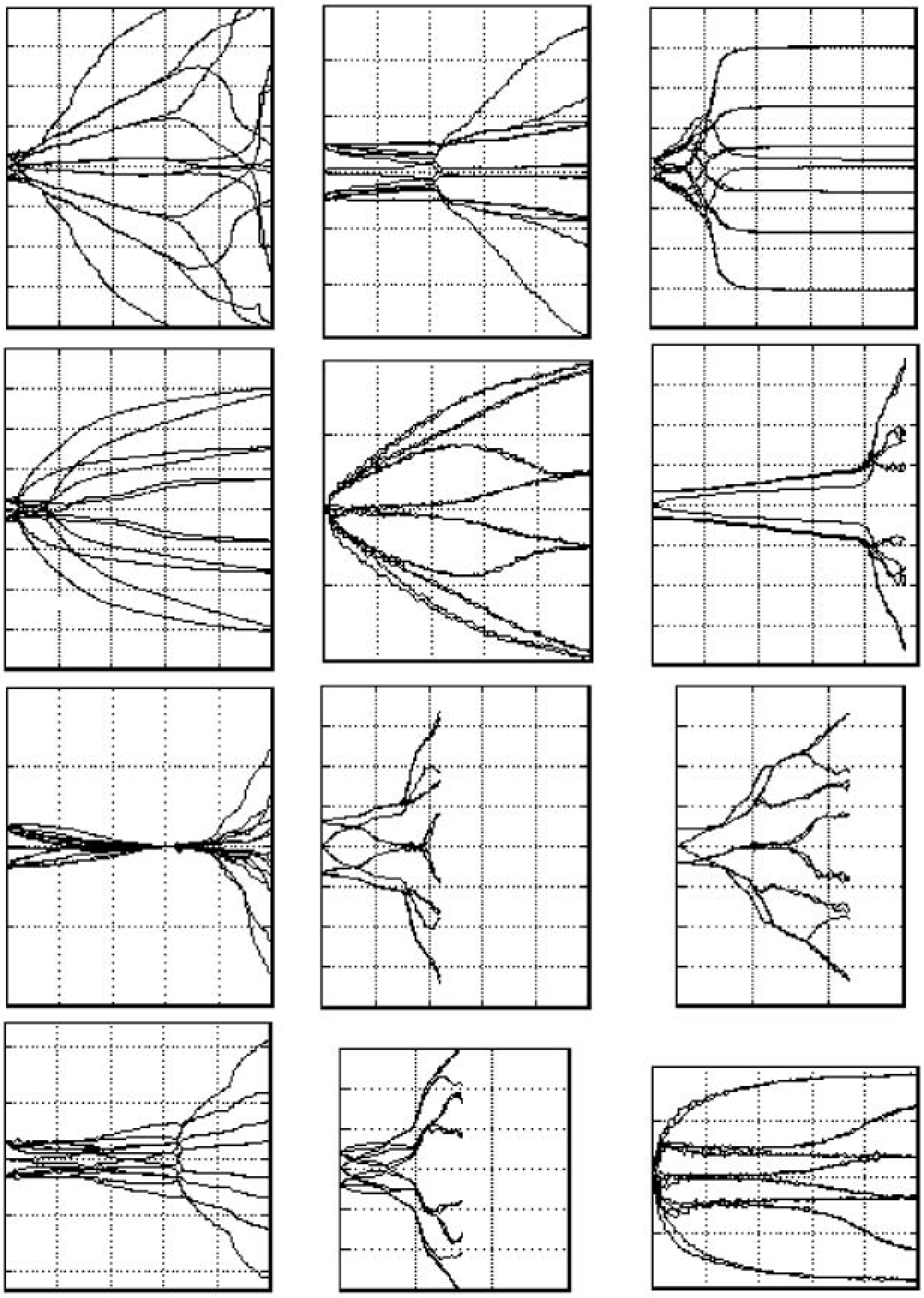

Figure 13 - *Selected trajectories of weight values versus epoch count*

REFERENCES

[Ash] Ash T., ÒDynamic Node Creation in Backpropagation NetworksÓ, UCSD Technical Report ICS 8901, University of California, San Diego, 1989.

[Baba] Baba N, ÒA new approach for finding the global minimum of error function of neural networksÓ, *J. Neural Networks*, v2, pp367-273, 1989

[Fahl] Fahlman S., ÒFast-Learning Variations on Back-Propagation: An Empirical StudyÓ, in Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufman Publishers, 1989.

[FaLe] Fahlman S., Lebiere C., ÒThe Cascade-Correlation Learning AlgorithmÓ, in Neural Information Processing Systems 2, Morgan Kaufman Publ. 1990

[GaSb] Gawthrop P, Sbarbaro D, ÔStochastic Approximation and Multilayer Perceptrons: The Gain Backpopagation AlgorithmÓ, *J. Complex Systems*, v4, 1990, CompSys Pub, Champaign IL, USA

[Gorm] Gorman R., et al., "Analysis of Hidden Units in a Layered Network Trained to classify Sonar Targets", *J. Neural Networks*, vol 1, pp75-89, 1988

[GoSe] Gorman R., Sejnowski T., ÒLearned Classification of Sonar Targets Using A massively Parallel Neural NetworkÓ, IEEE trans. ASSP, v36, #7, july 88

[Hayk] Haykin S, Adaptive Filter Theory, Prentice Hall, 1986

[HSW1] Hornik K, Stinchcombe M, & White H, ÒMultilayer feedfoward networks are universal approximatorsÓ, *Neura Networks*, v2, p359-366, 1989, Pergamon Press.

[HSW2] Hornik K, Stinchcombe M, & White H, ÒUniversal approximation of an unknown mapping and its derivatives using multilayer feedforward networksÓ, *Neural Networks*, v3, pp 551-560, 1990, Pergamon Press.

[Jaco] Jacobs R., "Increased Rates of Convergence through Learning Rate Adaptation", *J. Neural Networks*, v1, p295-307, 1988

[JePr] Jeffries C., Protzel P., "High Order Models for Error Correcting Code", SPIE Technical Symposium, Orlando Florida.

[KrSa] Kramer A., Sangiovanni A., "Efficient parallel learning algorithms for neural networks", in Neural Information Processing Systems 1, Morgan Kaufman Publ., 1989

[Krug] Kruglyak L, ÒHow to solve the N bit encoder problem with just two hidden unitsÓ, *Neural Computation* 2, 339-401, 1990.

[Lipp] Lippman R, ÒAn introduction to computing with neural nets, IEEE ASSP Magazine, April 1987.

[Luen] Luenberger D., Linear and Non-linear Programming, Addison-Wesley, 1986.

[NaPa] Narendra K, Parthasarathy K, ÒIdentification and Control of Dynamical Systems using neural networksÓ, *IEEE trans. Neural Networks,* v1, #1, March 1990.

[Orph] Orphanides S., ÒGram-Schmidt Neural NetsÓ, in *Neural Computation*, v2, #1, Spring 1990, MIT Press.

[RRKOS] Ruck,Rogers,Kabrisky,Oxley,Suter, ÒThe Multilayer perceptron as an approximation to a Bayes Optimal Discriminant FunctionÓ, *IEEE trans Neural Networks*, v1, #4, December 1990. 296-298

[Rume] Rumelhart D., et al., Parallel Distributed Processing: Explorations in the microstructure of Cognition, vol 1. , MIT press, 1988.

[Schw] Schwartz T, ÒChase Manhattan uses neural nets for credit card fraud detectionÓ, *Synapse Connection*, Mar/April 1990

[See1] See special issue on neural networks in control systems in *IEEE Control Systems Magazine*, v10, #3, April 1990.

[See2] See Proceedings of IEEE Intnl. Joint Conf. Neural Networks, 1988,1989,1990.

[SeRo] Sejnowski T., and C. R. Rosenberg, NETtalk: A Parallel Network that Learns to Read Aloud, Johns Hopkins Univ. Tech. Report, JHU/EECS-86/01, 1986.

[SiWu] Singhal S., Wu L., "Training multilayer Perceptrons with the Extended Kalman Algorithm", in Neural Information Processing Systems, Morgan Kaufman,1989

[TaJu] Tao M, Jurik M, ÒImproved Back-Propagation Algorithms: A systems control and identification approachÓ, Proc. IEEE conference Multidimensional Signals and Parallel Processing, Asilomar Calif., 1988.

[Toll] Tollenaere T, ÒSuperSAB: Fast Adaptive Back-propagation with good scaling propertiesÓ, Æ. Neural Networks, v3, pp561-573, 1990

[Waib] Waibel A., et al., "Phoneme Recognition Using Time Delay Neural Networks", ATR internal report TR-I-0006, Oct 30, 1987.

[Wan] Wan E, ÒNeural network classification: a Bayesian InterpretationÓ, *IEEE trans Neural Networks*, v1, #4, December 1990. 303-305

[Werb] Werbos P.,"Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences", thesis in applied math, Harvard University, Aug 1974.

[Whit] White R, "The learning rate in back-propagation systems: an application of Newton's method", Proc. IEEE IJCNN, June 1990, p I-679.

[WiSt] Widrow B., Stearns S., Adaptive Signal Processing, Prentice-Hall, 1985.

1 Under current research at NASA, Lewis Research Center, USA.

2 Reportedly used by Morgan Stanly Bank in New York City.

3 Several U.S. Airlines are using one neural net to predict the passenger seating demand for a certain day and another net to map the projected demand into optimal seating allotments.