22. Rumelhart D., Hinton, G. and Williams R., Learning internal representation by error propagation, in Parallel Distributed Processing Vol. 1, (1986) 318–362, : M. I .T. Press, Cambrigde, MA,
23. Sprecher D., On the structure of continous functions of several variables, Trans. Am. Math. Soc. 115 (March 1965) 340–355,
24. Sugeno M. and Kang G.T., Structure indentyfication of fuzzy model, Fuzzy Sets and Systems 28 (1988) 15–33,
25. Takagi T. and Sugeno M., Fuzzy identification of systems and its applictaions in modeling and control, IEEE Transaction on Systems, Man, and Cybernetic, 15 (1985) 116–132,
26. Tsukamoto Y., An approach to fuzzy reasoning method, in Advances in Fuzzy Set Theory and Applications, Madan M. Gupta, Rammohan K. Ragade and Ronald R. Yager (eds.) North-Holland, Amsterdam, (1979) pp. 137–149,
27. Wang L. - X., Fuzzy systems are universal approximators in Proc. IEEE 1922 Int. Conf. Fuzzy Systems, San Diego CA, March (1992) pp. 1163-1170,
28. Weigl M. and Kosiński W. A neural network system for approximation and its implementation, in Materialy Konferencyjne I Krajowej Konferencji Sieci Neuronowe i Ich Zastosowania, Proc. I-st National Conference Neural Networks and their Applications, Kule, 12 - 15. IV, 1994, vol. 2 (1994) pp. 485–491, Częstochowa,
29. Widrow B. and Lehr M., 30 years of adaptive neural networks: perceptron, madaline, and backpropagation, Proc. of the IEEE, 78 (9) (1990) 1415–1442,
30. Żurada J., Multilayer perceptron networks; Selected aspects of training optimization in Materialy Konferencyjne I Krajowej Konferencji Sieci Neuronowe i Ich Zastosowania, Proc. I-st National Conference Neural Networks and their Applications, Kule, 12 - 15. IV, 1994, vol. 1, (1994) pp. 82–103 Częstochowa,

Intelligent Information Systems III
Proceedings of the Workshop held in
Wigry, Poland, 6-10 June, 1994

pages 443- 458

Fast Algorithms for Neural Network Design and Training

Bogdan Maciej Wilamowski

University of Wyoming, Laramie WY 82071-329, USA
wilam@uwyo.edu

Abstract. A few algorithms for fast design and training of neural networks are presented. First the methods of improvements of the backpropagation algorithm are discussed. By the introduction of the new method of gradient computation fast convergency was obtained and the problem of "flat spots" was practically eliminated. The input pattern transformation for easy training and separations are presented and this leads to a significant simplification of the required neural network. A very efficient method for design of neural network, for classification purposes, is also shown. All presented concepts are explained with examples.

1 Introduction

When the backpropagation algorithm [1, 2, 3] was discovered, many researchers hoped for a broad usage of multilayer neural networks. This expectation was not totally fulfilled. Due to local minima and "flat spots" it is difficult to find adequate solutions for many of the practical problems. Sometimes, a solution can be obtained after more than a hundred thousand iterations. Often, when the initial weights are chosen unfortunately, no solution can be found. Many improvements were suggested for the backpropagation algorithm, but problems with the convergency of the backpropagation algorithm are still prevailing. Some researches have chosen an unsupervised training method like, WTA [4] with further labeling. Interesting results were obtained with the "quickprop" [5] or cascade-correlation algorithm. Some researches were experimenting with the general regression networks [6], or probabilistic networks [7, 8]. In this work, other improvements for the backpropagation algorithm are proposed [9, 10] by a modification of the way how the neuron gain is computed. Interesting results were obtained, as it is shown in section 3 but many practical problems still are difficult to solve using just the backpropagation algorithm. In section

4, the input space transformation is proposed for easy pattern classification. This way the neural network training can be significantly simplified. Also the training procedure is faster and simpler. In some cases the network can be designed analytically, as it is shown in section 5. In section 6 a very simple neural network is described, which can be used for rapid prototyping. This concept uses the simplest feedforward version of the counterpropagation network, originally proposed by Hecht-Nilsen [11] and then fully developed by Zurada [12]. Further modification of the network is also described in section 6. This method is illustrated in a few practical examples, including the problem of the two spirals.

2 Modifications of the gradient computation in the backpropagation algorithm

The backpropagation algorithm, commonly employed for the training of multilayer neural networks, suffers from a slow asymptotic convergence rate. For the sigmoidal bipolar activation function

$$f(net) = \frac{2}{1 + \exp(-\lambda\, net)} - 1 \tag{1}$$

the gradient (slope) is computed as a derivative of (1)

$$g_1 = \frac{2\exp(-\lambda\, net)}{1 + \exp(-\lambda\, net)} = 0.5\,(1 - f^2) \tag{2}$$

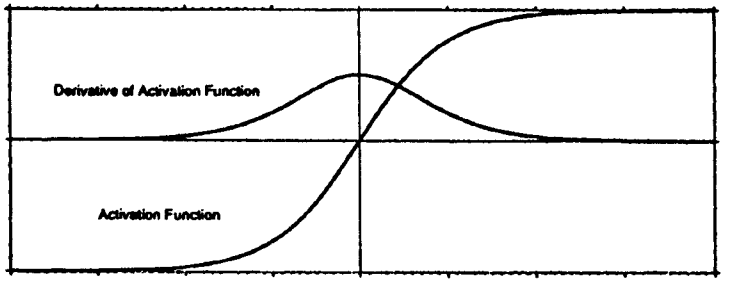

Fig. 1. Modified sigmoidal activation function and its derivative using equations (1) and (2) for bipolar neurons.

The activation function and it's gradient are illustrated in Figure 1. Only a small error propagates back for networks with large net values. Convergence is even slower for neurons with high gains (steep activation functions). Typical cases for convergence using the "XOR" neural network with $\lambda = 1.0$ and $\eta = 0.2$ and randomly chosen initial

weights are shown in Figure 2 where the best 10 of 20 randomly chosen weights were used. For most cases, convergence was reached in less than 500 iterations. However, when the initial weights are chosen unfavorably, it is extremely difficult to recover the proper state during the learning procedure as shown in Figure 3. To begin with unfavorable weights, the neural network was trained into saturation and then the desired output were changed to their opposite values. In most of these cases, the standard backpropagation algorithm did not converge at all.

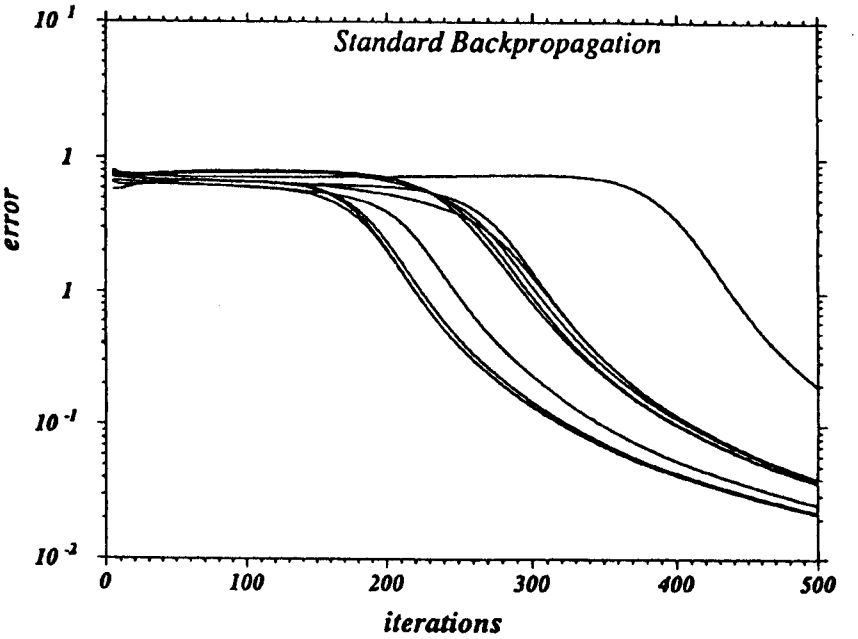

Fig. 2. Global error as function of iterations for the "XOR" neural network using the standard back-propagation algorithm with learning constant $\eta = 0.2$ and neuron gain $\lambda = 1.0$ for ten best of 20 randomly chosen initial weights.

In the backpropagation algorithm, the weight changes are proportional to the error propagating from the output through the slopes of activation functions and through the weights. For large *net* values the derivatives of the activation functions, as well as the back propagating error signals, are very small (Fig. 1). Convergence of the learning process can be improved by changing how the error propagates back through the network. It is proposed that for the purpose of error propagation, the slope (gradient) of the activation function is calculated as the slope of the line connecting the output value with the desired value rather than the derivative of the activation function at the output value.

$$g_2 = \frac{f_{desired} - f_{actual}}{net_{desired} - net_{actual}} \tag{3}$$

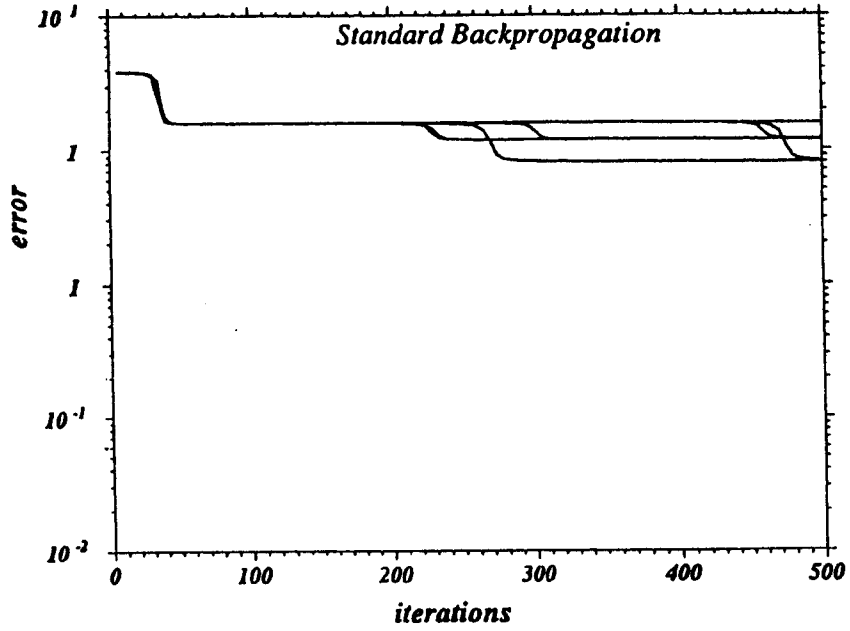

Fig. 3. Global error as a function of iterations for the "XOR" using unfavorable data. The network was trained into saturation and then the desired outputs were changed to their opposite values (maximally wrong values).

This is illustrated in Fig. 4. Note that if the output value is close to the desired value, the calculated slope corresponds to the derivative of the activation function, and the algorithm is identical with the standard backpropagation formula. Therefore, the "derivative" is calculated differently only for large errors, when the classical approach significantly limits error propagation.

After modifying the standard backpropagation algorithm, the convergence ratio was greatly improved. Figure 5 shows results for unfavorably chosen initial weights (the same weights used in Figure 3). However, when this modified algorithm was used with random weights, (favorable starting conditions as in Figure 2), the rate of convergence was slower compared to the convergence using the standard backpropagation algorithm. This undesired effect of the modified algorithm is more significant for cases with small net values and large output errors. In this case, the gradient g2 computed using equation

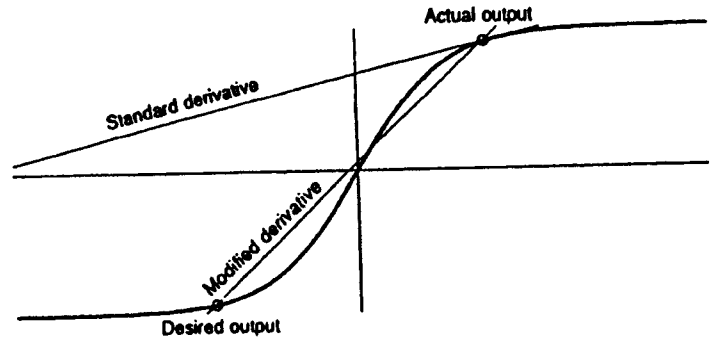

Fig. 4. Illustration of the modified derivative computation using the slope of the line connecting the points of actual output and desired output.

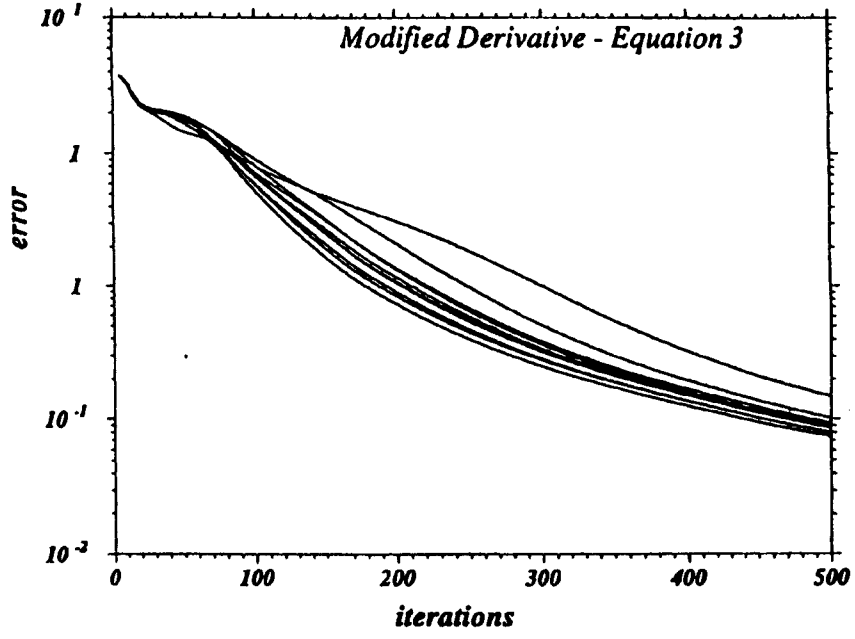

Fig. 5. Global error as a function of iterations for the "XOR" neural network using the modified back-propagation algorithm using equation 3 for gradient computation with and the same unfavorable starting weights as in Fig. 2

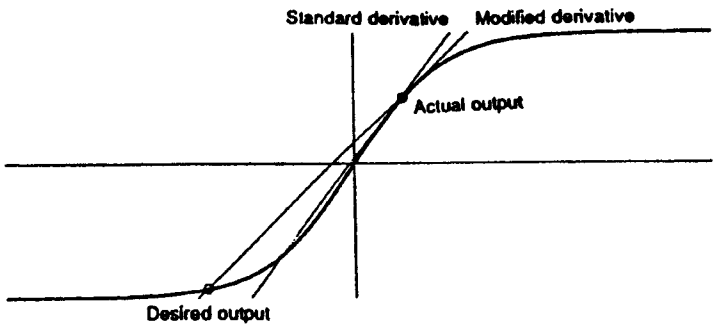

Fig. 6. Illustration of gradients g1 and g2 obtained for small net values.

(3), is larger than g1 which was obtained using equation (2) (as in the traditional back-propagation algorithm). This is illustrated in Figure 6.

To compensate for this effect, the effective gradient was computed using the formula

$$g_{eff} = \sqrt{\frac{g_1^2 + g_2^2}{2}} \tag{4}$$

where g_1 is the slope obtained as the derivative of activation function (Eq. (2)) and g_2 is the slope obtained using the modified formula (3). Equation (4) guarantees that the larger of two gradients, g1 and g2, will dominate. Figure 7 shows the results for the "XOR" example with the same initial weights and learning parameters as in Figures 3 and 5, but equation (4) was used for computation of the effective gradient.

Similar results were obtained with other neural network structures such as binary number classifiers, parity bit finder and others. In all cases, significant improvement of convergence was noted, especially in networks with large neural gains and with unfavorable chosen starting weights.

It was also observed that in cases with small gains $\lambda = 1$, modification of the gradient computation did not improve the convergence. This is due to the fact that in cases where the output values are far from saturation, the slope computed using the modified method (Figure 4) is smaller than the slope computed as a simple derivative which causes the error to propagate slowly. This corresponds to lowering the effective learning constant.

3 Input space Transformation

A single neuron separates the input space into two categories. The separation is done, dependent on the input space dimensions, by line, plane or hyperplane. The multilayer neural network inherits this single neuron feature. Therefore, in the first layer, input patterns are always separated by lines, planes or hyperplanes. If input patterns are, so call, linearly nonseparable, then a multilayer neural network is required. Also, in this case, the input patterns are separated in the first layer by lines, planes or hyper planes. In

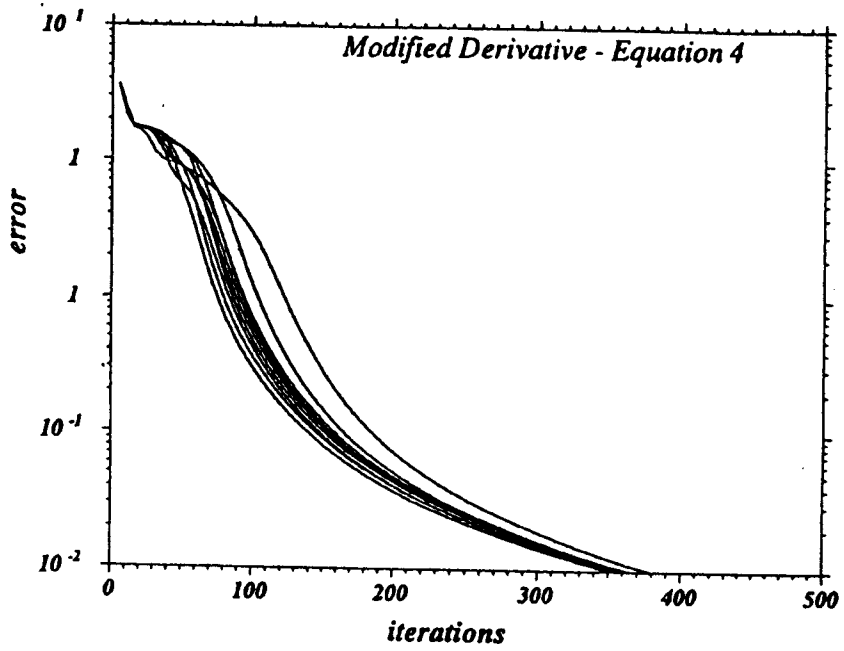

Fig. 7. Global error as a function of iteration for the "XOR" neural network using the effective gradient (equation (4)). The same unfavorable weights and learning constant were used as in Fig. 3 and Fig. 6.

many practical cases it is required to separate a cluster surrounding a certain exemplary pattern. To separate clusters, it seems to be more desirable to be able to separate clusters by circle, sphere or hypersphere. This type of problems, are very difficult to solve using neural networks.

The purpose of this work is to introduce a transformation of input patterns, such that one sigmoidal type neuron is able to separate a cluster in the transformed input space. This task can be accomplished by transforming input space on the sphere where each cluster can be cut out by plane. In the similar manner as any fragment of an apple surface can be easily separated single cut. A similar concept of using a sphere for easy pattern separation was introduced by Kohonen [4]. The input pattern is transformed into a sphere by normalizing the length of the input vectors to unity. Unfortunately, in this approach the important information about the input vector magnitude is lost and not used for pattern recognition.

In order not to lose an information about input patterns, the dimensionality of the input space must be increased by one. Once the input space is transformed into a sphere, or a hypersphere, the radius of the sphere is fixed and equal for convenience, to the unity. In the case of $n + 1$-dimensional hypersphere with the fixed radius the number of

independent variables (angles) is n. Therefore the simplest transformation approach is to use the components of the input vectors of the input Cartesian space as angles of the transformed hypersphere.

There are many possible transformations possible

$$\begin{aligned} z_1 &= \sin(kx_1) \\ z_2 &= \cos(kx_1)\sin(kx_2) \\ z_3 &= \cos(kx_1)\cos(kx_2)\sin(kx_3) \\ &\cdots\cdots\cdots\cdots \\ z_n &= \cos(kx_1)\cos(kx_2)\ldots\cos(kx_{n-1})\sin(kx_n) \\ z_{n+1} &= \cos(kx_1)\cos(kx_2)\ldots\cos(kx_{n-1})\cos(kx_n) \end{aligned} \tag{5}$$

or

$$\begin{aligned} z_1 &= \cos(kx_1) \\ z_2 &= \sin(kx_1)\cos(kx_2) \\ z_3 &= \sin(kx_1)\sin(kx_2)\cos(kx_3) \\ &\cdots\cdots\cdots\cdots \\ z_n &= \sin(kx_1)\sin(kx_2)\ldots\sin(kx_{n-1})\cos(kx_n) \\ z_{n+1} &= \sin(kx_1)\sin(kx_2)\ldots\sin(kx_{n-1})\sin(kx_n) \end{aligned} \tag{6}$$

where $x_1, x_2, \ldots x_n$ are Cartesian coordinates in the original input space and $z_1, z_2, \ldots z_n$, z_{n+1} are Cartesian coordinates of the input patterns transformed on the hypersphere of the $n+1$ dimension. The k is the scaling factor equal to $\alpha_{MAX}/(x_{MAX} - x_{MIN})$ where α_{MAX} is the angular size of the part of the sphere used, and $x_{MAX} - x_{MIN}$ is the maximum range of the input variable. If different values of k are used for different dimensions then clusters are separated by ellipsoids, instead of spheres. Transformation formulas (5) and (6) are only two examples out of many possible transformations to a sphere in the n+1 dimension space. Neither equation (5) or (6) can be easily implemented in hardware. In many practical cases such hardware implementation is not required, because data can be easily transformed by means of the computer, before training or recognition is taking place.

4 Design Example with Transformation of the Input Space

Let us consider a simple two dimensional case with three clusters to be separated as Fig. 8 illustrates. First task, is to estimate the center of gravity of each cluster. In example from Fig. 8 these centers are: [5, 6], [-2.4, 3.8], and [3,-5]. Let us assume to use one quarter of the sphere $k = \pi/34$. Using transformation (5) the centers of gravity in augmented z space are given

$$\begin{aligned} z_1 &= \sin(kx_1) \\ z_2 &= \cos(kx_1)\sin(kx_2) \\ z_3 &= \cos(kx_1)\cos(kx_2) \end{aligned} \tag{7}$$

The separation planes should be normal to the vectors given by (7). Equations for the separation planes can be easily found using the point and normal vector formula:

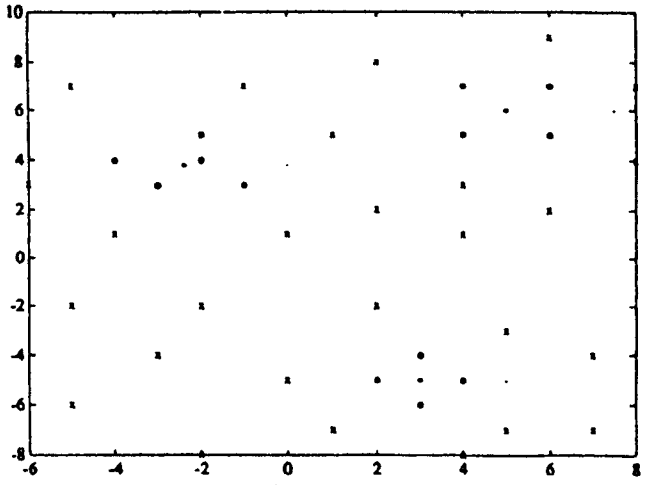

Fig. 8. Two dimensional input space with three clusters.

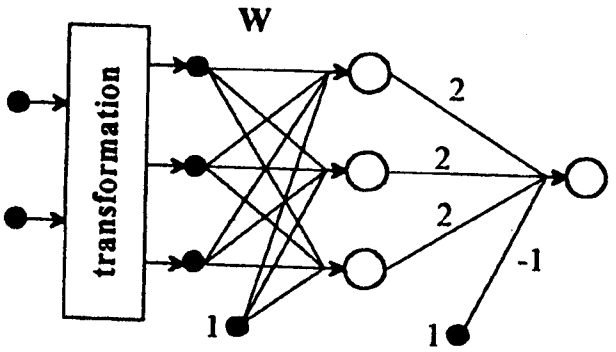

Fig. 9. Network structure for classification of clusters shown in Fig. 8.

$$z_{CG}^T\ (z - z_E) = 0 \tag{8}$$

where z_E is a point transformed from the edge of the cluster (see Fig. 8). From equation (8) all weights for the first layer of the neural network can be found analytically as

$$W = \begin{bmatrix} 0.4794 & 0.4955 & 0.7243 & -0.9689 \\ -0.2377 & 0.3603 & 0.9020 & -0.9713 \\ 0.2955 & -0.4580 & 0.8384 & -0.9801 \end{bmatrix} \tag{9}$$

Fig. 9 shows the neural network structure. One can see that the neuron in the second

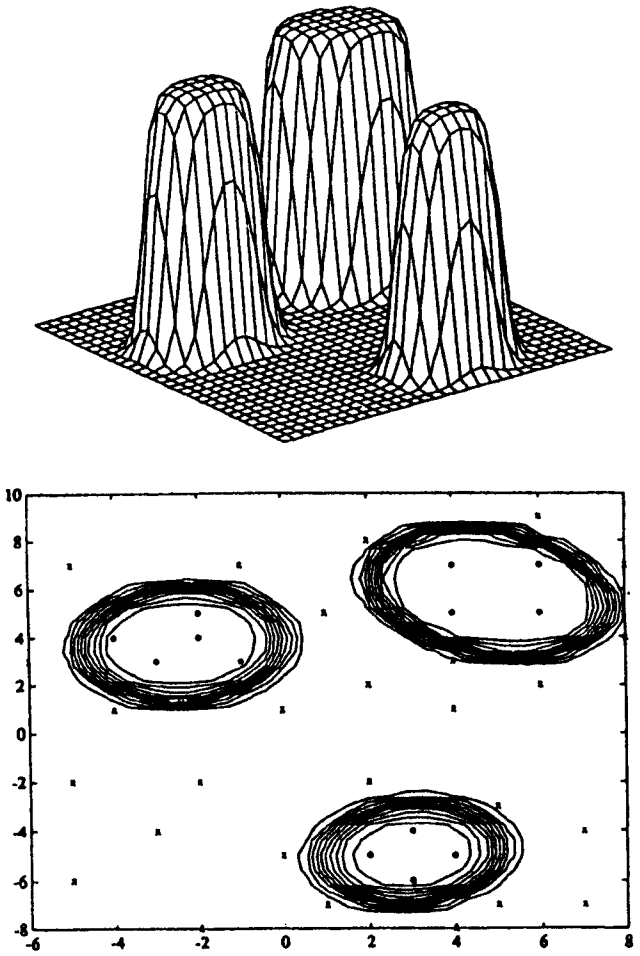

Fig. 10. Input-output mapping of the network shown in Fig. 9 (a) mesh plot, (b) contour plot.

layer perform the "OR" operation only. Fig. 10 shows the mapping from input **x** space into output value y.

5 Simple neural network for classification purposes

Some advanced neural networks [13], known as general regression networks [6], or probabilistic networks [7, 8], require a number of hidden neurons equal to the number of

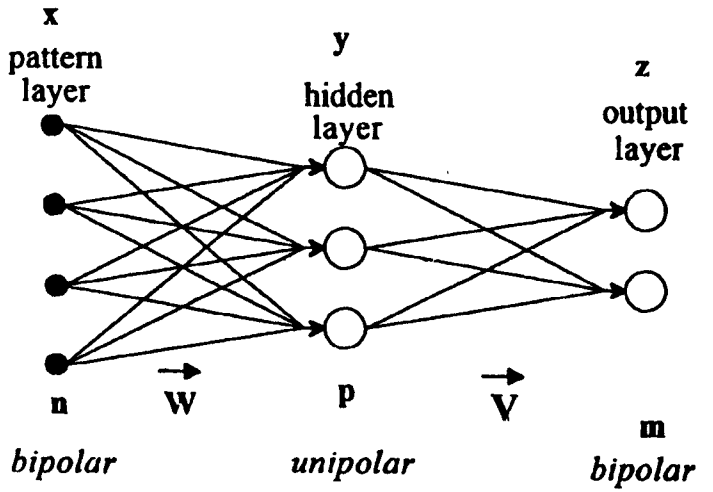

Fig. 11. Feedforward version of the counterpropagation network.

training samples. A similar concept can be used for simple neural network structures where the number of the hidden neurons should be equal to the number of clusters. This network for rapid prototyping is based on the counterpropagation network [11]. The simplest form of this network can be used as an easy classifier. Such possibility was mentioned by Wasserman [14] and then elaborated by Zurada [12]. Similar neural networks, which were not derived from the counterpropagation structure, were also, later, independently developed [15].

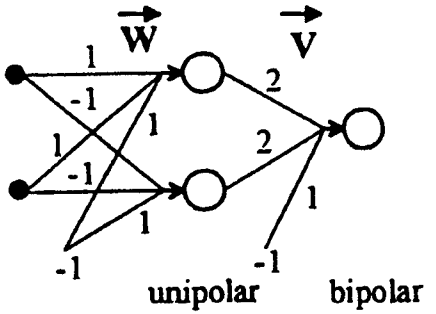

Fig. 12. Network structure for the "XOR" problem.

The basic structure of the feedforward version of the counterpropagation network is

shown in Fig. 11. [12]. This network consists of the Kohonen WTA layer with unipolar neurons and the outstar Grossberg layer. When binary input patterns are considered, then input signals are either -1 or $+1$. For the given cluster only one neuron, the winner, in the hidden layer is activated. The weights of this neuron $w_j = x_j$ are exactly equal to the pattern corresponding to the center of cluster x_j. If the binary input pattern x_i is applied to the network the net value for the $j-th$ neuron is

$$net = w_j^T x_i = n - 2\,HD(w_j, x_i) \tag{10}$$

where n is the dimension of the input pattern and $HD(w_j, x_i)$ is the Hamming distance between column vectors w_j and x_i. If the neuron is required to react also for the patterns within a given Hamming distance HD to the stored pattern, the threshold of this neuron should be set to

$$w_{j\ th} = n - 1 - 2\,HD_{jr} = n - 1 - d_j \tag{11}$$

where HD_{jr} denote the radius of attraction or radius of $j-th$ cluster, and d_j is the cluster diameter. Thresholds in the hidden layer should be set in such way that only one neuron in the hidden layer can be activated at a given time. The hidden layer has an unipolar activation function. The higher the neuron gain in the hidden layer the better the performance of the network, the best performance can be obtained with hard threshold neurons. No activation function is required for the output layer or it can be a bipolar activation function without a threshold (the threshold is set to zero).

The network shown in Fig. 11 can be also considered as an analog memory. For any binary input (address) certain analog values (stored in the weights of output layer) can be recovered. Moreover, input patterns (address) need not to be exact. The correct analog value will be recovered from the memory if the input pattern is close enough (within HD) to the actual stored address. Further modifications of the network shown in Fig. 11 are also possible. For proper operation, the input patterns need not to be binary as suggested in [12, 15], but they can be analog vectors with the same magnitude as in classical Kohonen layer [4], or the input pattern transformation, as shown in section 4, can be used.

The network, which was briefly described above, is very easy to design, and all weights are given a priori. The weights in the first layer are equal to the input patterns, and weights in the output layer are equal to the output patterns. Obviously, both input and output patterns which are stored in the network could be the centers of gravity of the stored clusters. Not all input or output patterns should be stored. The only parameters which require some computations are the thresholds in the hidden layer. The value of the threshold should be chosen accordingly to the diameter of a cluster using equation (11).

In case of pattern classification into two categories, the network can be further simplified. Instead of setting the number of neurons in the hidden layer equal to the number of patterns, the number of neurons in the hidden layer should be equal to the number of patterns to one category. The output layer has just one neuron and to perform the "OR" operation all of its weights should be equal to two, and the threshold should be equal to one. This particular weight arrangement is chosen with assumption of the smallest integer threshold and weights.

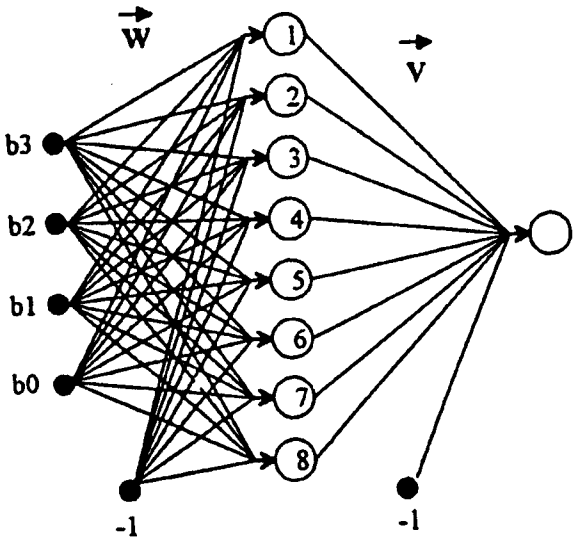

Fig. 13. Network structure for the parity four problem.

Such simple network can be used for rapid prototyping. The "XOR" case can be used as an example. Assume that patterns [-1,-1] and [1,1] should be classified to +1 category and remaining patterns to the -1 category. The network structure for the "XOR" case is shown in Fig. 12. Thresholds in the hidden layer are computed from equation 6 assuming $HD = 0$. Note, that if this problem is design using the feedforward counterpropagation network, then four neurons would be required in the hidden layer. Fig. 13 shows the neural network structure for the parity 4 problem where the weights in the hidden layer are given by

$$W = \begin{bmatrix} -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \end{bmatrix} \tag{12}$$

Also in this case only 8 neurons are required in the hidden layer while for counterpropagation network 16 neurons would be required.

Another illustrative example is the problem of the two spirals, which is known as very difficult for the backpropagation network. Using the proposed approach, this problem can be easily solved. Let us assume that the two spirals shown in Fig. 14(a) should be separated into two categories. Let us also assume that the spiral marked with squares should be classified into the +1 category and the remaining patterns into the -1 category. Therefore, the number of hidden neurons is equal 28. Using the 8 level resolution for both x and y axis, the three bits are required for each axis. For proper analog operation a

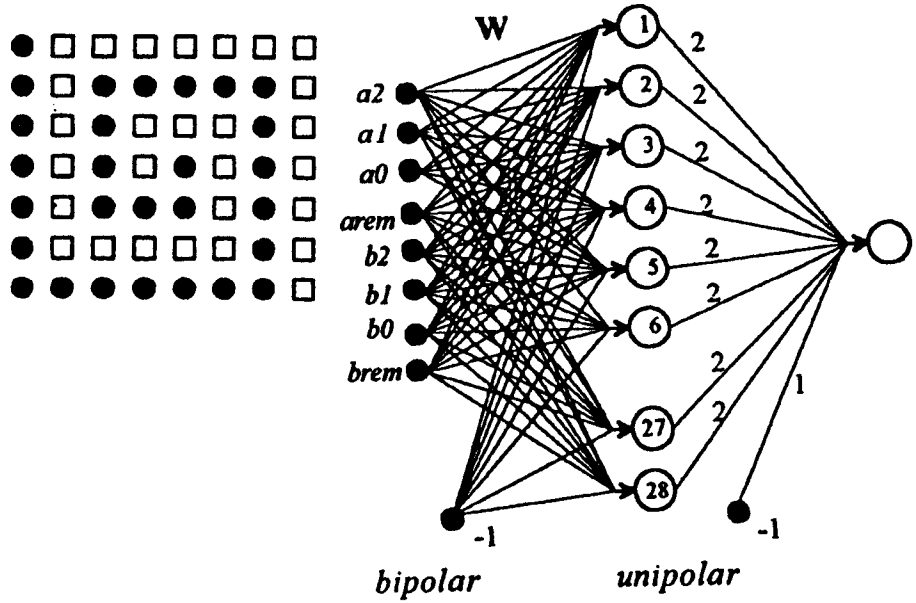

Fig. 14. Problem of two spirals (a) - pattern distribution, (b) - network structure.

remainder of the AD conversion is also supplied as a fourth input for each axis. The remainder is defined as

$$rem = abs\left(x - \sum_{i=0}^{2} 2^i \, b_i\right) \tag{13}$$

Thus, the network has six binary inputs and two analog ones. The network structure is shown in Fig. 14(b). Assuming $HD = 1$, thresholds for the hidden layer are set to be equal five. Fig. 15 show the nonlinear mapping obtained the designed network.

6 Conclusion

Modification of the backpropagation algorithm using formula (4) with the combination of (2) and (3) for determining of the slope of the activation function enhances the convergence of the learning procedure. The effect of this modified algorithm on the speed of convergence was examined with various examples.

These results show that the standard backpropagation algorithm converges very poorly, or does not converge at all, when the weights are chosen unfavorably (maximally wrong) and when gains on the neurons are high. However, results show significant improvement with the modified algorithms. Note that for efficient classification of patterns, the neural networks with high gain values λ have to be used. In such practical cases, the standard backpropagation algorithm has difficulty converging.

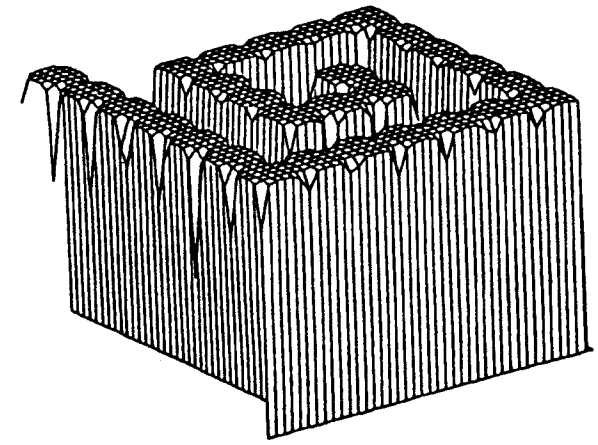

Fig. 15. Input-output mapping for the problem of two spirals.

The described in section 5 procedure for two dimensional input space can be easily generalized for the multidimensional case. Original patterns in input Z space have to be transformed into augmented space Z. Equations (5) or (6) can be used for the transformation. Then the standard backpropagation method can be used for network training.

As one can see, from the presented in section 6 examples, the approach based on the counterpropagation network is very fast and the resultant neural network is comparable or even simpler than backpropagation networks.

References

1. Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning internal representations by error propagation. In Parallel Distributed Processing, Cambridge, MA: MIT Press. 1 , (1988) 318-362.
2. Werbos P. J.: "Beyond regression: new tool for prediction and analysis in the behavioral sciences," Ph. D diss, Harvard University 1974.
3. Werbos, P. J.: Back-propagation: Past and future. Proceeding of International Conference on Neural Networks, San Diego, CA, 1 , (1988) 343-354.
4. Kohonen T.: Adaptive, Associative, and Self-Organizing Functions in Neural Computing. Applied Optics 26 , (1987) 4910-4918.
5. Fahlman, S. E. . Faster-learning variations of back propagation: An empirical study. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), Proceedings of the 1988 Connectionist Models Summer School, San Mateo, CA. (1988) 38-51.
6. Specht D. F.: A general regression neural networks.IEEE Trans. Neural Networks 2 (1991) 568-576.

7. Maloney S. P.: The Use of Probabilistic Neural Networks to Improve Solution Times for Hul-To-Emitter Correlation Problems, International Joint Conference on Neural Networks, New Jersey: IEEE Press. 1 (1989) 289-294.
8. Specht D. F.: Probabilistic Neural Networks. Neural Networks, 3 (1990) 109-118.
9. Torvik L. M., Wilamowski B. M.: Modification of the Backpropagation Algorithm for Faster Convergence. 1993 International Simulation Technology Multiconference November 7-10, San Francisco, also in proceedings of Workshop on Neural Networks WNN93 (1993) 191-194.
10. Wilamowski B. M., Torvik L. M.: Modification of Gradient Computation in the Back-Propagation Algorithm. ANNIE'93 - Artificial Neural Networks in Engineering, St. Louis, Missouri, November 14-17, 1993; also in Intelligent Engineering System Through Artificial Neural Networks, ASME Press 3 (1993) 175-180.
11. Hecht-Nielsen R.: Counterpropagation Networks. Appl. Opt. 26 (1987) 4979-4984. and Hecht-Nielsen R: Applications of Counterpropagation Networks. Neural Networks. 1 (1988) pp. 131-139.
12. Zurada, J. M.: Introduction to Artificial Neural Systems, West Publishing Company 1992.
13. Wasserman P. D.: Advanced Methods in Neural Computing. Van Nostrand Reinhold, New York (1993).
14. Wasserman P. D.: Neural Computing Theory and Practice, Van Nostrand Reinhold, New York (1989).
15. Hao H. J., Tan S., Vandewalle J.: Bipolar Pattern Association Using a Two-Layer Feedforward Neural Network. IEEE Trans. On Circuit and Systems-I, CAS1-40 (1993) 943-946.

Intelligent Information Systems II
Proceedings of the Workshop held in
Wigry, Poland, 6-10 June, 199

pages 459- 47

A low-dimensional model of coordinated eye and head movements

A. W. Przybyszewski*, A. H. Clarke
Vestibular Research Lab., HNO Klinik, Klinikum Benjamin Franklin,
Freie Universität Berlin
Dept. of Cognitive and Neural Systems, Boston University, MA 02215

Abstract. Eye and head movement data, were recorded under head-fixed and head-free conditions, and compared with theoretical results obtained using a non-linear model of eye-head coordination. The model explains slow, or pursuit movement correlated closely to target movement, and saccades, or quick phases of eye movement. Eye movement under head-fixed conditions was modeled by an externally forced Duffing equation, whilst properties of head movement are described by a second externally forced Duffing equation with lower eigen frequency. In the more natural, head-free conditions where both eye and head movements are used synergetically to pursue a visual target, the vestibulo-ocular reflex (VOR) is represented by coefficients defining the mutual coupling between these two oscillatory systems. In the present model, the oscillator that models eye movement has an inhibitory influence on head movement; head to eye coupling coefficients are included to model the influence of the VOR mechanism. Individual eye and head movement patterns in different subjects can be adequately modeled by altering the coupling coefficients. In order to adequately simulate those changes introduced by microgravity conditions, the coefficients defining eye-head coordination (mutual coupling) must be changed. It may be hypothesized that such changes in the neurovestibular system could introduce the instability in eye-head coordination, which is known to lead to space sickness.

1 Introduction

Coordinated eye and head movements are employed extensively to direct visual attention during everyday activity. Indeed, the eye may be considered the most active of all human

* Supported by a Whitehall grant S93-24