A weight discretization paradigm for optical neural networks[0]

E. Fiesler[1,2], A. Choudry[3], and H.J. Caulfield[2]

[1]Department of Computer Science / [2]Center for Applied Optics /
[3]Department of Electrical and Computer Engineering
University of Alabama in Huntsville
Huntsville, Alabama 35899, U.S.A.

ABSTRACT

Neural networks are a primary candidate architecture for optical computing. One of the major problems in using neural networks for optical computers is that the information holders: the interconnection strengths (or weights) are normally real valued (continuous), whereas optics (light) is only capable of representing a few distinguishable intensity levels (discrete). In this paper a weight discretization paradigm is presented for back(ward error) propagation neural networks which can work with a very limited number of discretization levels. The number of interconnections in a (fully connected) neural network grows quadratically with the number of neurons of the network. Optics can handle a large number of interconnections because of the fact that light beams do not interfere with each other. A vast number of light beams can therefore be used per unit of area. However the number of different values one can represent in a light beam is very limited. A flexible, portable (machine independent) neural network software package which is capable of weight discretization, is presented. The development of the software and some experiments have been done on personal computers, while the major part of the testing has been done using a super computer.

1. INTRODUCTION

1.1. Background and definitions

Perceptron-like neural networks can be trained by teaching them patterns. A pattern consists of a set of elements (*pixels*). The word pixel, short for picture element, does not nessecarily imply that the patterns are images. Non-numeric input (e.g. visual, auditive, features) has to be converted to numeric input by some mapping before it can be presented to the neural network. Each pixel can assume a continuous or a discrete value. In the discrete case, the possible set of pixel values is often limited. Typical pixel value sets, used in artificial neural networks, are: $\{0, 1\}$ and $\{-1, 1\}$. Usually, a pattern is presented to the neural network by feeding each of the pixels to a different input neuron, i.e. a neuron of the first layer of the neural network. Therefore the number of input neurons is equal to the number of pixels in the pattern.

Many artificial neural networks consist of two phases : a *training* or *learning phase*, and a *recall* or *use phase*. During the training phase patterns are presented to the network. The interconnection strengths (also called synaptic strengths or weights) of the neural network are adapted conform these patterns by means of a neural network learning rule (as for example the backward error propagation learning rule). When the weights are stabilized, the network is called *fully trained*. During the recall phase input patterns are presented to the neural network. Based on the fixed weights, corresponding outputs, which are the activation values of the neurons of the highest layer, are generated by the network. This form of neural network training is called *off-line training*. Off-line training is often crucial, since it separates the normally time consuming training from the recall process and therefore speeds up the use of the neural network tremendously.

In *supervised learning*, or neural network learning rules with a 'teacher', two patterns are presented to the network: an input pattern and a *target pattern* (the 'teacher') which is the desired output for the neural network.

[0]This document has been published in the Proceedings of the International Congress on Optical Science and Engineering, volume SPIE-1281, pages 164–173, The International Society for Optical Engineering Proceedings, Bellingham, Washington, U.S.A., 1990, ISBN: 0-8194-0328-8. The final version of this document has been accepted for publication in IEEE Transactions on Systems, Man, and Cybernetics (IEEE-SMC) under the title: "A Universal Weight Discretization Method for Multi-Layer Neural Networks".

In these networks the total output of the network has to converge towards the target pattern; i.e. the activation values of the output neurons have to converge towards the pixel values of the target pattern. In *auto-associative learning* the input patterns are the same as the target patterns. Auto-associative learning is therefore used to train the neural network to remember a set of patterns. One of the main applications of auto-associative learning is image reconstruction or recalling a pattern if only a partial or disabled input is available. For example: a knowledge base which can handle incomplete data.

In *hetero-associative learning* the input and target patterns are usually different. Hetero-associative learning is therefore used to train the network to associate each of the input patterns with its corresponding target pattern. For example: association of geological information of a certain geographical area with the presence of fossil fuels there.

1.2. Problem definition

Discretization is essential for all kinds of implementations of neural networks, since most information media used can only discriminate a small set of intensity levels. For optical implementations (optical computing[1]) but also for analog electronic implementations[2]. For these reasons, weight discretization is investigated.

The research goals for the work presented in this paper are to develop a discretization method for back-propagation neural networks, to create a software environment for the simulation of neural network weight discretization, to test the discretization method by computer simulations.

1.3. Prior work

1.3.1. Hopfield model

In his famous 1982 paper, John J. Hopfield[3] studied a 'clipped' weight matrix (T_{ij}). He replaced T_{ij} by ± 1, the algebraic sign of T_{ij}. The purposes were to examine the necessity of a linear synapse supposition (by making a highly nonlinear one) and to examine the efficiency of storage. He found little performance (which is the ability of the neural network to learn (and recall) a certain amount of information.) degradation. The number of recallable patterns was (analytically) $\frac{2}{\pi}$ of the number with linear T_{ij}'s. Thus severe discretization causes only mild degradation. To restore performance, the number of neurons would have to be increased by $\frac{\pi}{2}$.

1.3.2. Winner-take-all-models

Stirk et al.[4] addressed a variety of non-Hopfield models from the viewpoint of performance sensitivity to analog optical inaccuracies. The results for simple winner-take-all networks are bad. Furthermore "big N" cases ($N = 64$) are significantly worse than "small N" ($N = 16$) cases, where N is the number of neurons in the input layer. Optics seems advantageous over electronics only for very large N, say, 10^4 to 10^6. This means optics is accurate enough only for small N, but small N is probably better done electronically.

1.3.3. Farhat's adaptive method

In a paper showing how to implement large neural networks ($10^3 - 10^4$ neurons), Farhat et al.[5] reformulate the Hopfield model for two-dimensional inputs and outputs and four-dimensional interconnects. They clip the interconnection matrix in various ways $\{0, 1\}$, $\{-1, 1\}$, $\{-1, 0, 1\}$, etc. and find that with "adaptive thresholds" the $\{0, 1\}$ interconnection pattern (easy to implement optically and electronically) can achieve the same performance level as a multivalued interconnection pattern. In effect, they have restored the $\frac{2}{\pi}$ loss observed by Hopfield by using adaptive thresholds.

1.3.4. Summary of prior work

What is known from these prior studies is that some neural network designs are far more prone to discretization errors than others and that compensatory methods such as adding more neurons or allowing adaptive thresholds may restore the performance of the network.

2. DISCRETIZATION OF BACK-PROPAGATION NETWORKS

2.1. Approach

Among the multi-layer neural network learning rules capable of both auto-associative, and hetero-associative learning, the backward error propagation learning rule[6,7], also known as back-propagation or error propagation, is the most widely used and is simple to use[8]. The back-propagation learning rule was therefore chosen for the experiments. The experiments are based on the discretization methods which are described in paragraph 2.2.

The number of neurons per layer can vary; N_l indicates the number of neurons in layer l ($1 \leq l \leq L$), where L is the total number of layers (or slabs) in the network including the input and the output layer. The interconnection weights between two layers of a neural network can be represented by a matrix $W_{l,ij}$, here l represents the level of the matrix ($1 \leq l < L$). The level l is the ordinal number of the lower one of the two layers connected by $W_{l,ij}$. The indices i and j determine the ordinal number of the neuron in the lower and upper layer respectively. The weights ($W^c_{l,ij}$), as used in ordinary back-propagation models, can theoretically assume any (continuous) value:

$$-\infty < W^c_{l,ij} < \infty.$$

The used discretization methods produce discrete weights ($W^d_{l,ij}$). In general there are D discretization levels, where D is a finite integral number. Since the set of desired discretization values can be mapped on any sequence of numbers using a bijection, any set with the same cardinality will satisfy. In this paper the choice is made for a sequence of consecutive integers equally divided among positive and negative numbers :

$$W^d_{l,ij} \in \left\{ n - \left\lfloor \frac{D+1}{2} \right\rfloor \;\middle|\; n = 1, 2, ..., D \right\}.$$

Discretization of the weights will impair the performance of the neural network, because there is a loss of information capacity. This is compensated by increasing the number of neurons and/or the number of hidden layers of the network. In other words: discretized weights contain less information than continuous ones; this is compensated by using more of them. The used discretization methods are discussed in the next paragraph.

2.2. Three discretization methods

2.2.1. The multiple-thresholding method

The multiple-thresholding method is the simplest of the three discretization methods used. It starts by fully training the neural network, using the back-propagation learning rule; i.e. iterate (over steps 2 till 4 of the algorithm in appendix A) until is reached *convergence* (i.e. reaching of the convergence criterion (see paragraphs 1.1. and 3.1.) or another limiting factor). Then discretize the continuous weights into discrete valued weights using a nonlinear function (usually a multiple-threshold). The weight matrices so obtained are referred to as the *discrete network*. The original set of weight matrices with continuous weights is called the *continuous network*. Chiueh and Goodman[9] have applied this method using three discretization levels. They found that about 15%-50% of the networks did not work.

2.2.2. The direct discretization method

In the direct discretization method, the neural network is initialized with discrete weights, which have random values within the discrete range. The forward propagation is similar to the normal back-propagation learning rule (step 2 of the algorithm in appendix A). During the backward propagation (step 3 of the algorithm in appendix A), the weights are updated only if the difference in weight ($\Delta W^d_{l,ij}$) is big enough to change the weight into one of the other possible discrete values. This method does not work for the standard back-propagation learning rule.

2.2.3. The continuous-discrete learning method (CDLM)

This new developed method (a preliminary version of this work has been presented by Fiesler, Choudry and Caulfield[10]), schematically shown in figure 1, starts off with the multiple-thresholding method (paragraph 2.2.1.,

and (a) to (f) in figure 1). Next the original input pattern (a) is fed (h) into the discrete network (g). The outputs obtained by forward propagation (step 2b of appendix A) are compared (i) with the target pattern (e) and the errors (δ's) are back-propagated (j) through the **continuous network (c)**. Next, the weights of the continuous network are discretized (f) as before and the process starts all over again until the system reaches convergence. The fully trained discrete network (g) can now be used for the recall phase.

This approach leads to an increase in the total number of iterations needed. The process can be speeded up by skipping the full training of the continuous network, since starting with a fully trained continuous network is convenient, but not necessary.

3. EVALUATION OF THE DISCRETIZATION METHODS

3.1. The back-propagation model used

The experiments performed are based on the back-propagation learning rule. The characteristics of the back-propagation model used are: it is usually fully connected between adjacent layers (*interlayer connections*), has no *intralayer connections* i.e. connections between neurons in the same layer, and no *supralayer connections* i.e. connections between neurons that are not in adjacent layers nor in the sama layer [7:figure 8.3]. The following assumptions are made: a negative weight is inhibitory, a zero weight means no connection and a positive weight is excitatory. Zero weights offer the possibility of having no connection between certain neurons in spite of fully connectedness.

The patterns used to train the network were free of noise. They are presented to the neural network as a set of pairs of patterns. Each pair consists of an input pattern and its corresponding target output pattern. A pattern consists of a rectangular matrix of pixel values (height $\times$ width) which is mapped onto the one dimensional set of input neurons (the neurons in layer one). Let h^i be the height of the input patterns and w^i the width (i stands for input). The pixel value of input pattern j ($p_{j,mn}$) is mapped on input neuron $(m-1)w^i + n$, where m indicates the row of the pixel in the pattern ($1 \leq m \leq h^i$) and n the column ($1 \leq n \leq w^i$). The patterns in the set, which are presented in the order they are provided by the user, are fed repeatedly into the input neurons of the neural network until convergence is reached. The convergence criterion used is: when all the activation values of the output neurons reach their ϵ-range. An ϵ-range is the range near a desired output, determined by the *deviation parameter* (ϵ). The deviation parameter is the maximum amount that an output activation value may deviate from the target pattern value.

3.2. Implementation

3.2.1. Software specification

In order to perform the discretization experiments with all the necessary flexibility, a portable (machine independent) back-propagation software environment was developed using the PASCAL programming language[11]. The main part of the software has been developed on a personal computer. When some of the experiments took more than 24 hours to run on the personal computer, changing over to a Cray X-MP/24 supercomputer seemed a good idea.

Some of the flexibility criteria for the software environment were : the capability of handling both auto-associative and hetero-associative learning, changing the pattern size (height and width), the number of patterns, the learning rate (η), the number of layers, the number of neurons per layer (for each hidden layer), the number of discretization levels, the deviation parameter, and the maximum number of iterations for the (continuous-discrete) learning method, also the ability of choosing a discretization method, an initialization scheme for the weights, and a pixel value set.

The most important outputs of the simulation system (for both the continuous and the discrete network) are:

- the stop criterion : whether the desired output is reached (within the ϵ-range) or the number of iterations reached its maximum
- the output values (activation values of the output neurons) after each iteration, if desired

- the number of iterations made
- the number of errors made (output activation values that reached undesired values)
- the number of output neurons that did not reach the desired output (within the ϵ-range)
- the maximum deviation (between actual output activation value and the desired output value)

The user can choose which outputs are desired for specific experiments.

3.2.2. Methodology

The most promising discretization method is the CDLM, because the direct discretization method does not work, and the CDLM easily outperforms the multiple-thresholding method because the first includes and improves the second method.

Two approaches were taken to test both the CDLM and the multi-threshold method. First a systematical 'search' through the state space of possible experiments. The starting position was the smallest network possible and using two discretization levels, since this is the preferred number for most neural network implementations. The next variable to vary is therefore the pattern size which is the same as the input size. Then both auto- and hetero-associative learning were tested. The number of patterns was the next variable to vary. This meant starting off with a two layer system, which would be enlarged in further experiments. The number of possible experiments was growing exponentially, a second approach was taken.

Here, the collection of patterns was fixed. This means a fixed number of patterns, a fixed pattern size, a fixed number of input and output neurons and, in this case, a choice was made for hetero-associative learning. The central parameter in this approach is the number of discretization levels.

3.2.3. Parameter definition

This paragraph discusses the parameters which were kept constant in most of the experiments. For perfect recall (i.e. output activation values are within their ϵ-range), using noise free inputs, it turned out that the higher the learning rate the faster the convergence. Besides dedicated experiments, the value of the learning rate (η) was kept at 0.5[12]. The value used for the deviation parameter (ϵ) is 0.05. Random values in the range $[-0.1, 0.1]$ were used to initialize the weights. The pixel value set used is $\{-1, 1\}$. For the nonlinear function required in both the multiple-thresholding method and the CDLM, a multiple threshold with rounding off to the nearest pixel value was used. A typical figure for the maximum number of iterations is 20,000. The local thresholds (Θ's) or biases[7] were kept zero.

In the second approach a number of variables were fixed in order to limit the state space. In these experiments, hetero-associative learning, three by five pixel patterns, a learning rate of 0.5, and two to four layers were used.

3.3. Results

"Being able to learn and perfectly recall (associate) a set of noise free patterns" is taken as a measure for comparing the performances of continuous and discrete networks.

What could be expected intuitively, is confirmed by the experiments: the performance of the CDLM is better than the performance of the multiple-threshold method. The outputs of the multiple-thresholding method were often outside the ϵ-range. Sometimes, wrong results were obtained when rounding(-off to the nearest pixel value) was applied to the outputs. In general rounding can be used to obtain a $\{0, 1\}$-result from an output neuron that did not converge into the ϵ-range.

The CDLM on the other hand usually achieves much better results. The first approach (see paragraph 3.2.2.) emphasized the performance restoration of the neural network using the minimum number of discretization levels, which is two. In the case of associating one set of two patterns by the simplest network of one input neuron, one output neuron and a variable number of hidden layers and neurons in them, a two level discretization works very well (see figure 2). The two layer network does not converge to a value within the ϵ-range but gives the right answer

after rounding. The performance of a three layer system with one neuron in the hidden layer is worse than the two-layered network. But adding neurons to the hidden layer increases the performance. With five neurons in the hidden layer the ϵ-range of 0.05 is reached. In figure 3 the situation for four layers is depicted. In order to reach ϵ-accuracy, the minimum number of neurons needed in the hidden layers is (5&3), (2&4), and (1&5) neurons in the second & third layer. If two patterns are stored, the graph (see figure 4) is less smooth but the same behavior can be observed. Fourteen hidden neurons are needed in the second layer to reach ϵ-accuracy. Note that for the three layered network with one or two neurons in the hidden layer faulty results are produced when rounding is used. In order to reach the right outputs after applying rounding in the case of four layers, the minimal number of units needed in the hidden layers is (2&9), (3&6), (4&5), (5&4), (6&4), (7&4), (8&3), and (9&4) neurons in the second & third layer. However, rounding gives sometimes wrong results for some hidden layer sizes larger than these minima. The number of extra neurons needed to restore the performance is relatively high. Other results showed that this relative overhead became smaller for bigger networks.

In some of the smaller networks the activation values of the output neurons remained constant during the discrete training. In these cases the performances of the multiple-thresholding method and the CDLM are equal.

The emphasize of the second approach is on comparing results using different numbers of discretization levels. The pattern set consists of nine pairs of character-like patterns. The continuous network could perfectly associate them all after 372 iterations.

# discr. levels	# iterations	# non-converg.	max. abberation
2	10000	67	0.88
3	10000	18	0.50
5	10000	2	0.12
7	69	0	0.00
9	47	0	0.00

This table shows that if the number of discretization levels increases, the number of non-converging outputs decreased. A perfect recall was obtained at seven discretization levels. Further increase leads to a decrease in the number of iterations needed for a perfect recall. This observation can also be made for the continuous network: adding neurons to the network leads to a faster convergence for the continuous network (less iterations needed).

Sometimes the performance of the network reached a maximum, without reaching total convergence. In order to compensate for this, the observed maximum performance is stored and used as final result.

If the CDLM starts with a full training of the continuous network, the number of iterations needed for training the discrete network varies from one to a number of iterations comparable to the number of iterations needed for the training of the continuous network. In this case, the total number of iterations needed for the CDLM is therefore one to two times that of the continuous network.

In general: addition of a new layer to the network, without increasing the total number of neurons in the network, results in a performance degradation. This can be explained by looking at the total number of interconnections before and after the addition of a new layer. If a fully connected two-layered network contains N_1 neurons in its input layer, and N_L in its output-layer, the total number of weights is $N_1 \cdot N_L$. The number of neurons needed in an additional (=hidden) layer to have the same number of connections is $\frac{N_1 \cdot N_L}{N_1 + N_L}$.

4. CONCLUSIONS

Of the three discretization methods proposed, the CDLM works better than the multiple-thresholding method, and the direct discretization method is unusable. A portable neural network software environment has been created for performing the discretization experiments. As intuitively expected, the lower the discretization (more discretization levels), the better the performance of the neural network. But, using two discretization levels, as desired by optical and electronic implementations, give reasonable results. The results of a two layer neural network are usually good enough when using the CDLM.

If the CDLM starts with a fully trained continuous network, the number of iterations will be one to two times that of the number needed for the full training of the continuous network.

5. ACKNOWLEDGEMENTS

The authors like to thank Drs. Ing. Vincent J. Harrand and Dr. J. Patrick Ryan and the members of the local neural network interest group for their helpful critique and fruitful discussions on this matter. They are also grateful to visiting professor Dr. Ir. Jerry Smith for his critical remarks and Drs. Jeroen van der Zijp for proofreading this paper.

† This work is partially supported by TBE (agreement dated 10/21/86) and ONR (contract # N00014-86-K-0591).

APPENDIX A

Back(ward Error) Propagation; the formulas:

In this appendix, $a_{l,i}$ represents the activation value of the neuron i in layer l of the neural network, and t_j is the target pattern value which corresponds to neuron j in the output layer.

Back-propagation consists of the following steps :

(1) Initialize the weights ($W^c_{l,ij}$) and offsets ($\Theta_{l,j}$).

(2a) $a_{l,i}$:= input to the i-th neuron of the input layer.

(2b) Forward propagation :

$$a_{l,j} := \frac{1}{1 + e^{-(\sum_{i=1}^{N_{l-1}} W^c_{l-1,ij}\ a_{l-1,i}\ -\ \Theta_{l-1,j})}} \qquad \text{for } 2 \le l \le L$$

(3) Backward propagation :

$$\Delta W^c_{l,ji} := \eta\ \delta_{l+1,j}\ a_{l,i}$$

where

$$\delta_{l,j} := \begin{cases} (t_j - a_{l,j}) a_{l,j} (1 - a_{l,j}) & \text{if } l = L-1 \\ a_{l+1,j}(1 - a_{l+1,j})(\sum_{k=1}^{N_{l+2}} \delta_{l+2,k} W^c_{l+1,kj}) & \text{if } 1 < l < L-1 \end{cases}$$

next, add $\Delta W^c_{l,ji}$ to $W^c_{l,ji}$

(4) IF no(t enough) convergence THEN GOTO (2a)

REFERENCES

1. H. John Caulfield, Jason Kinser, and Steven K. Rogers, Optical Neural Networks, *Proceedings of the IEEE*, volume 77, number 10, pages 1573-1583, October 1989.

2. A.P. Thakoor, J.L. Lamb, A. Moopenn, and John Lambe, "Binary synaptic connections based on memory switching in a-Si:H", *Neural networks for computing*, American Institute of Physics (A.I.P.) conference proceedings 151, pages 426-431, editor: John S. Denker, at: Snowbird, Utah, April 13-16, 1986, American Institute of Physics, New York, New York, 1986, ISBN: 0-88318-351-X.

3. John J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", *Proceedings of the National Academy of Sciences of the U.S.A.*, volume 79, pages 2554-2558, April 1982.

4. Charles W. Stirk, S.M. Rovnyak and R.A. Athale, "Effects of noise on an Optical Implementation of an Additive Lateral Inhibitory Network", *Proceedings of the IEEE first international conference on neural networks*, volume III, pages 615-623, editors: Maureen Caudill and Charles Butler, at: San Diego, California, June 20-24, 1987, SOS Printing, San Diego, California, 1987, IEEE Catalog Number 87TH0191-7.

5. N.H. Farhat, S. Miyahara and, K.S. Lee, "Optical analog of two-dimensional neural networks and their application in recognition of radar targets", *Neural networks for computing*, American Institute of Physics (A.I.P.) conference proceedings 151, pages 146-152, editor: John S. Denker, at: Snowbird, Utah, April 13-16, 1986, American Institute of Physics, New York, New York, 1986, ISBN: 0-88318-351-X.

6. Paul Werbos, *Beyond regression: New tools for prediction and analysis in the behavioral sciences*, Ph.D. dissertation, Harvard University, 1974.

7. David E. Rumelhart, James L. McClelland, and the PDP Research Group, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, Volume 1: Foundations, The MIT Press, Cambridge, Massachusetts, ("A Bradford Book"), 1986, ISBN: 0-262-18120-7 or 0-262-18123-1 (set of two volumes).

8. Robert Hecht-Nielsen, "Neurocomputing: picking the human brain", *IEEE Spectrum*, volume 25, number 3, pages 36-41, March 1988.

9. T.D. Chiueh and R.M. Goodman, "Learning algorithms for neural networks with ternary weights", presented at the First Annual Meeting of the International Neural Network Society, Boston, Massachusetts, September 5-10, 1988, abstract: *Neural networks*, volume1, supplement 1, page 166, 1988.

10. E. Fiesler, A. Choudry, and H.J. Caulfield, "Weight discretization in backward error propagation neural networks", presented at the First Annual Meeting of the International Neural Network Society, Boston, Massachusetts, September 5-10, 1988, abstract: *Neural networks*, volume1, supplement 1, page 380, 1988.

11. Kathleen Jensen and Niklaus Wirth, *PASCAL User Manual and Report*, Springer Verlag, New York, New York, 1978 (corrected printing), ISBN: 0-387-90144-2 or 3-540-90144-2 (Berlin, Germany).

12. Maureen Caudill, "Neural Networks Primer", part II, *AI Expert*, volume 3, number 2, pages 55-61, February 1988.