CASCADE ERROR PROJECTION LEARNING ALGORITHM

Tuan A. Duong*†, Allen R. Stubberud†, and Taher Daud*
*Center for Space Microelectronics Technology
Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109
† Department of Electrical Engineering, University of California at Irvine
Irvine. CA 92717

Abstract:

In this paper, we workout a detailed mathematical analysis for a new learning algorithm termed Cascade Error Projection (CEP) and a general learning frame work. This frame work can be used to obtain the cascade correlation learning algorithm by choosing a particular set of parameters. Furthermore, CEP learning algorithm is operated only on one layer, whereas the other set of weights can be calculated deterministically. In association with the dynamical stepsize change concept to convert the weight update from infinite space into a finite space, the relation between the current stepsize and the previous energy level is also given and the estimation procedure for optimal stepsize is used for validation of our proposed technique.

The weight values of zero are used for starting the learning for every layer, and a single hidden unit is applied instead of using a pool of candidate hidden units as in the cascade correlation scheme. Therefore, simplicity in hardware implementation is also obtained. Furthermore, this analysis allows us to select from other methods (such as the conjugate gradient descent or the Newton's second order) one of which will be a good candidate for the learning technique. The choice of learning technique depends on the constraints of the problem (e.g., speed, performance, and hardware implementation); one technique may be more suitable than others. Moreover, for a discrete weight space, the theoretical analysis presents the capability of learning with limited weight quantization. Finally, 5- to 8-bit parity problems are investigated; the simulation results demonstrate that only three hidden units are required to learn a 5-bit parity problem perfectly and to learn a 6-bit parity with just one pattern error. Four hidden units are sufficient for a 7-bit parity problem with no error and for an 8-bit parity problem with one pattern error. We have restricted the learning to a fixed 100 epoch iterations for each single-layer perception (each single hidden unit) learning. In addition, with 3- to 4-bit weight resolution, it is demonstrated that this technique is capable of learning reliably 5- to 8-bit parity problems by incorporating additional hidden units (up to a maximum of 20).

I Introduction

There are many ill-defined problems in pattern recognition, classification, vision, and speech recognition which need to be solved in real time [1-3]. One of the most attractive features of the neural network is a massively parallel processing topology that offers tremendous speed specially when implemented in hardware. Generally, neural network approaches in hardware face two main obstacles:

(1) difficulty of network convergence due to the learning algorithm itself as well as the limited precision of the devices;

(2) high cost of implementing hardware to truly mimic the synapse and neuron transfer functions dictated by the algorithm.

Furthermore, the convergence and the implementable hardware have a mutual correlation to each other; for example, the convergence of the learning network depends on the weight resolution available in synapse [4-6], and the cost of implementation of each bit in synapse grows, at least doubly, in silicon area, power, and connectivity [7-8]

In this paper, CEP learning algorithm is presented. It offers a simple learning method using a one-layer perception approach and a deterministic calculation for the other layer. Such a simple procedure offers a fast, reliable, and hardware implementable learning algorithm. To validate the new learning theory of CEP, simulations for 5- to 8-bit parity problems are investigated in weight quantization of a floating point machine (32-bit for float and 64-bit for double precision) and limited weight quantizations (3- to 6-bit weight resolution) of VLSI hardware.

II Cascade Error Projection: Mathematical Foundation

We describe the new learning architecture along with its mathematical foundation. The architecture consists of two sub-networks: one uses perception learning (master network) and the other uses deterministic calculations (slave network). The architecture starts out as a single layer perception and adds hidden units when needed, one after another.

Assume that the network contains n hidden units (see Fig. 1) and the learning cannot be improved any further in the energy level. At this point, a new hidden unit (n+1) is added to the network.

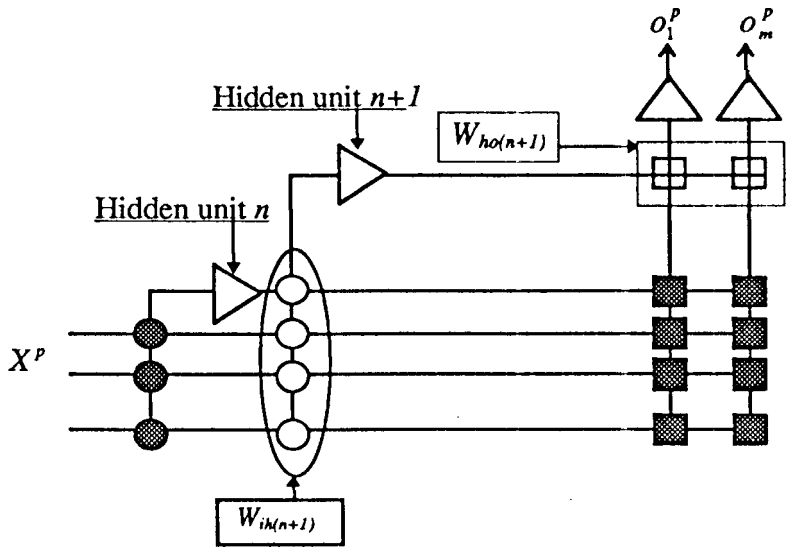

Figure 1: Schematic diagram for CEP learning with a newly added hidden unit (n+1). Blank circles and squares are the weight components that are determined by iterative learning and calculation, respectively.

N is the dimension of the input space, n+1 is the dimension of the expanded input space (n+1 is dynamically changed and is based on the learning requirement), and m is the dimension of the output space, P is the number of training patterns. Finally, f is a sigmoidal transfer function which is defined by:

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}}$$

Other notations are defined as follows:

$\varepsilon_o^p = t_o^p - o_o^p(n)$ denotes the error for an output index o and training pattern p between target t and the actual output $o(n)$. n indicates the output with n hidden units in the network.

f'$_o^p$(n) denotes the output transfer function derivative with respect to its input index o and the training pattern p.

$f_h^p(n+1)$ denotes the transfer function of hidden unit $n+1$ for a training pattern p.

X^p denotes the input pattern p.

The energy function is defined as follows:

$$E(i) = \sum_{p=1}^{P} E^p(i) = \sum_{p=1}^{P}\sum_{o=1}^{m}(t_o^p - o_o^p(i))^2 = \sum_{p=1}^{P}\sum_{o=1}^{m}(\varepsilon_o^p)^2$$

The difference of energy between the network with n hidden units and the network with $n+1$ hidden units can be obtained as,

$$\Delta E = E(n) - E(n+1) = \sum_{o=1}^{m}\{-w_{ho}^2\sum_{p=1}^{P}[f'^p_o f_h^p(n+1)]^2 + 2w_{ho}\sum_{p=1}^{P}[\varepsilon_o^p f'^p_o f_h^p(n+1)]\}$$

where $w_{ho} f_h^p(n+1)$ is small. This assumption is needed for nonlinear transformation function only.

As proved in ref. 9, the maximum energy reduction between hidden unit n and a newly added hidden unit $n+1$ with respected to w_{ho} is:

$$\max\{(\Delta E)_{Who}\} = \sum_{p=1}^{P}\sum_{o=1}^{m}\{\varepsilon_o^p f'^p_o f_h^p(n+1)\} \quad \text{when } w_{ho} = \frac{\sum_{p=1}^{P}\varepsilon_o^p f'^p_o f_h^p(n+1)}{\sum_{p=1}^{P}[f'^p_o f_h^p(n+1)]^2} \qquad (1)$$

Let

$$r = \begin{bmatrix} \frac{1}{m}\sum_{o=1}^{m} f'^1_o\{t_o^1 - o_o^1\} \\ \cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots \\ \cdots\cdots\cdots\cdots \\ \frac{1}{m}\sum_{o=1}^{m} f'^P_o\{t_o^P - o_o^P \end{bmatrix}$$

Then, $r \in [-1,1]^P$.

and

$$F_h(n+1) = \begin{vmatrix} f_h^1(n+1) \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ f_h^P(n+1) \end{vmatrix}$$

We can rewrite equation (1) using a matrix notation as follows:

$$AE = m\Gamma^T F_h(n+1) \qquad (2)$$

From (1) and (2), the energy reduction is dependent on a match between Γ and $F_h(n+1)$. The technique to match Γ with $F_h(n+1)$ can include, e.g. perception learning with gradient descent, maximum correlation or covariance with gradient ascent, conjugate gradient, and Newton's second order method. Therefore, the learning network performance really depends on the learning technique chosen for matching the error surface Γ and $F_h(n+1)$, In equation (2), let $f'^p_o(n)= 1$; then it can be rewritten as:

$$\Delta E = \sum_{p=1}^{P} \{ f_h^p(n+1) \frac{1}{m} \sum_{o=1}^{m} (t_o^p - o_o^p(n)) \} \qquad (3)$$

If we maximize AE in equation (3) by using the maximum correlation or covariance technique, then the learning algorithm is identical to that of Cascade Correlation. Thus a strong mathematical basis has been established for the algorithm as originally reported in ref. 10.

III Cascade Error Projection Learning Algorithm

a) Learning approach:

The new energy function is defined as:

$$\Phi(n+1) = \sum_{p=1}^{P} \{ f_h^p(n+1) - \frac{1}{m} \sum_{o=1}^{m} (t_o^p - o_o^p) f'^p_o \}^2$$

The weight updates between the inputs (including the weights by expanded inputs) and the newly added hidden unit is calculated as follows:

$$\Delta w_{ih}^p(n+1) = -\eta \frac{\partial \Phi(n+1)}{\partial w_{ih}^p(n+1)}$$

and the updated weight value for the synapse between the hidden unit h and the output unit o can be calculated by,

$$w_{ho}(n+1) = \frac{\sum_{p=1}^{P} \varepsilon_o^p f'^p_o f_h^p(n+1)}{\sum_{p=1}^{P} [f'^p_o f_h^p(n+1)]^2}$$

b) Simulation

1) Problems:

Using this technique, we have solved, 5- to 8-bit parity problems with (1) no limited weight quantization (floating point 32-bit for single. precision and 64-bit for double precision); and, (2) the limited weight quantization from 3-to 6-bits.

2) *Parameters*

The learning rate η is used and is set to decrease linearly as: $\eta_{new} = \eta_{old} - .01 * \eta_0$, where η_0 = initial learning rate.
For our simulation, the parameter values of table I are

Table I: Values of initial learning rate and a used in simulation for different parity problems and bit-resolution of synapses.

	5-bit parity	*6-bit parity*	*7-bit parity*	*8-bit parity*
Floating-point math ine Weight	$\eta_0 = 1.0$ a = N/A	$\eta_0 = 1.0$ a = NIA	$\eta_0 = 0.4$ a = N/A	$\eta_0 = 0.4$ a = N/A
3-bit Weights	$\eta_0 = 1.0$; a = .0024810	$\eta_0 = 1.0$; a = .016597	$\eta_0 = 1.0$; a = .008766	$\eta_0 = 1.0$; a = .004101
4-bit Weights	$\eta_0 = 1.0$; a = .0016467	$\eta_0 = 1.0$; a = .010858	$\eta_0 = 1.0$; a = .008218	$\eta_0 = 1.0$; a = .004101
5-bit Weights	$\eta_0 = 1.0$; a = .0016467	$\eta_0 = 1.0$; a = .010858	$\eta_0 = 1.0$; a = .008163	$\eta_0 = 1.0$; a = .004217
6-bit Weights	$\eta_0 = 1.0$; a = .0016467	$\eta_0 = 1.0$; a = .010858	$\eta_0 = 1.0$; a = .008163	$\eta_0 = 1.0$; a = .004217

3) Conversion technique (round-off technique)

In continuous weight space, the weight quantization can be considered as infinite. However, in hardware, weight quantization is always finite and limited Therefore, it is necessary to convert the weight updates Aw to a finite weight quantization Aw*. As proved in ref. 9, learning can be done with limited weight quantization as long as the difference between Aw and Aw* is viewed as equivalent independent white noise (round-off conversion technique) and the stepsize which is used to convert from Aw to Aw* must not be fixed. The dynamical stepsize can be roughly estimated as follows:
In continuous space, the energy reduction is:

$$\mathrm{AE} = \sum_{p=1}^{P} \sum_{o=1}^{m} \varepsilon_o^p f'^p_o f_h^p (n+1)$$

During learning, limited weight quantization value, Aw* directly affects the output of the $(n+1)^{th}$ hidden unit $f_h(n+1)$. It is expressed as:

$\tilde{f}_h^p(n+1) = f_h^p(\sum_{i=1}^{N+1} \tilde{w} x_i + \sum_{j=1}^{n} \tilde{w} x_h(j))$, $\tilde{w}$ *is a* weight *component in* finite weight space.

The reduction of energy in discrete (finite) weight space [9]is:

$$\Delta\tilde{E} = \tilde{E}(n) - \tilde{E}(n+1) = \sum_{p=1}^{P}\sum_{o=1}^{m} \varepsilon_o^p f'^p_o \tilde{f}_h^p(n+1) \quad (5)$$

Our main focus is finding the conversion factor (stepsize) based on known parameters (e.g. previous energy E(n)). This will lead to an estimation of the conversion factor, and hence, enhanced learning with limited weight quantization.

$$\Delta\tilde{E} \propto \tilde{f}_h^p(n+1) \quad (6)$$

Ignoring the nonlinear characteristic, it is roughly estimated that:

$$\tilde{f}_h^p(n+1) \propto \tilde{w} \propto stepsize(n+1) \quad (7)$$

From (5), (6), and (7) one can express

$$stepsize(n+1) \propto \tilde{E}(n) \quad (8)$$

The expression in (8) is a critical step in estimating the dynamical stepsize which is dependent on the previous energy of the network. In other words, the expression can be written as:

$$stepsize(n+1) = \alpha\tilde{E}(n)$$

The value of α can be obtained for each application through simulation (e.g. Table 1)
The weight update Aw is converted into the equivalent available weight quantization Aw*. The conversion can be summarized as follows:

$$\Delta w^*_{jh}(n) = \begin{cases} stepsize(n) * \mathrm{int}(\frac{\Delta w_{jh}}{stepsize(n)} - 0.5) \text{ if } (\frac{w_{jh}(n)}{stepsize(n)} + \mathrm{int}(\frac{\Delta w_{jh}(n)}{stepsize(n)} + 0.5)) < 2^B \text{ and } \Delta w_{jh}(n) > 0 \\ stepsize(n) * \mathrm{int}(\frac{\Delta w_{jh}(n)}{stepsize(n)} - 0.5) \text{ if } (\frac{w_{jh}(n)}{stepsize(n)} + \mathrm{int}(\frac{\Delta w_{jh}(n)}{stepsize(n)} - 0.5)) \le -2^B \text{ and } \Delta w_{jh}(n) < 0 \\ 0 \qquad \text{Otherwise} \end{cases}$$

4) *Simulation results:*

As noted earlier, we are solving 5-,6-,7-, and 8-bit parity problem with different synaptic resolution. We compare the results of higher and lower synaptic resolution to show the robustness of such an algorithm for hardware implementation. The input and output highs are 0.8, and the lows are -0.8. The neuron transformation function is a hyperbolic tangent

function. Zero values are used for the initial weights of each newly added hidden unit; therefore, it is not needed to conduct extra runs for each problem. A 100 epoch iterations learning is applied to each hidden unit for the weight between the input and the current hidden unit only.

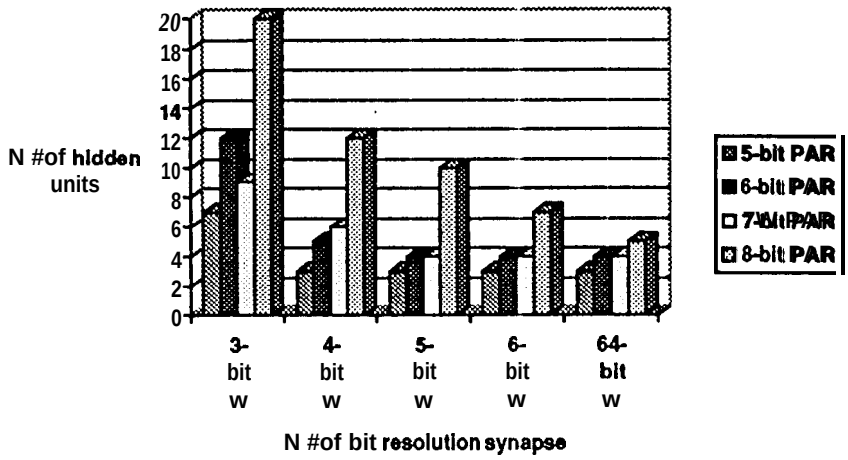

Figure 2: The chart shows CEP learning capability and the number of hidden units required to correctly solve 5- to 8-bit parity problems using round-off technique. x axis represents weight quantizations (3-6 and 64-bit) and y axis shows the resulting number of hidden units (limited to 20). Each hidden unit learning is provided with 100 epoch iterations. As shown, a lager number of hidden units compensate for the lower weight resolution.

Table II: *The number of iterations required for a* ***single*** *perception learning for different parity problems to obtain the* ***convergence****.*

	5-bit parity	6-bit parity	7-bit parity	8-bitparity
3-bit weights	*700 epochs*	*1200 epochs*	***900 epochs***	*
4-bit ***weights***	*300 epochs*	*500 epochs*	***600 epochs***	***1200 epochs***
5-bit weights	*300 epochs*	*400 epochs*	***400 epochs***	***1100 epochs***
6-bit ***weights***	*300 epochs*	*400 epochs*	***400 epochs***	***700 epochs***
64-bit weights	300 *epochs*	400 ***epochs***	400 *epochs*	***600 epochs***

* *Did not converge completely* ***within*** *the 20 hidden units* limit. *The percentage error was 136% after a* ***total*** *of 1600 epochs* of learning iterations.

IV. Conclusions

In this paper, we have shown that CEP is a reliable technique for both software- and hardware-based neural network learning. From this analysis, it is shown that the CC algorithm is a special case and can be understood in greater depth with this analysis. Moreover, the theoretical analysis provides us with the general framework of the learning

architecture, and the particular learning algorithm can be independently studied for its suitability for a given application associated with given constraints specific to each problem. For example, for hardware implementation CEP is advantageous, but for software, covariance or Newton's second order method is more advantageous). For the CEP learning algorithm, the advantages can be summarized as follows:

- A fast and reliable learning technique
- An easy implementation in hardware
- A low weight resolution requirement in weight space
- A robust model in learning neural networks

Acknowledgments:

The research described herein was performed by the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology and was jointly sponsored by the Ballistic Missile Defense Organization/Innovative Science and Technology Office (BMDO/IST), the Office of Naval Research (ONR), the Advanced Research Projects Agency (ARPA), and the National Aeronautics and Space Administration (NASA). The authors would like to thank Dr. A. Thakoor for useful discussions.

References:

[1] T. A. Duong, T. Brown, M. Tran, H. Langenbacher, and T. Daud, "Analog VLSI neural net work building block chips for hard ware-in-the-loop learning," *Proc. IEEE/INNS Int'l Join Conf. on Neural Networks*, Beijing, China, Nov. 3-6, 1992.

[2] T. A. Duong et. al, "Low Power Analog Neurosynapse Chips for a 3-D "Sugarcube" Neuroprocessor," *Proc. of IEEE Intl' Conf. on Neural Networks*(ICNN/WCCI), Vol III, pp. 1907-1911, June 28-July 2, 1994, Orlando, Florida.

[3] B.E. Boser, E. Sackinger, J. Bromley, Y. LeCun, and L.D. Jackel, "An Analog Neural Network Processor with programmable Topology," *IEEE Journal of Solid State Circuits*, vol. 26, NO. 12, Dec. 1991.

[4] P. W. Hollis, J.S. Harper, and J.J. Paulos, "The effects of Precision Constraints in a Backpropagation learning Network," *Neural Computation, vol.* 2, pp. 363-373, 1990.

[5] M. Hoehfeld and S. Fahlman, "Learning with limited numerical precision using the cascade-correlation algorithm," *IEEE Trans. Neural Networks*, vol.3, No. 4, pp 602-611, July 1992.

[6] T. A. Duong, S.P. Eberhardt, T. Daud, and A. Thakoor, "Learning in neural networks: VLSI implementation strategies," chapter of Handbook of Fuzzy logic and neural networks, Ed: C.H. Chen, Umass (to be published), 1995.

[7] S.P. Eberhardt, T.A. Duong, and A.P. Thakoor, "Design of parallel hardware neural network systems from custom analog VLSI "building-block" chips," *IEEE/INNS Proc. IJCNN*, June 18-22, 1989 Washington D.C., vol. II, pp. 183.

[8] T. A. Duong, S. P. Eberhardt, M. D. Tran, T. Daud, and A. P. Thakoor, "Learning and Optimization with Cascaded VLSI Neural network Buitding-Block Chips," *Proc. IEEE/INNS International Join Conference on Neural Networks*, June 7-11,1992, Baltimore, MD, vol. I, pp. 184-189.

[9] T. A. Duong, A. Stubberud, and T. Daud , "Cascade Error Projection learning theory" *submitted to Neural Computation*, 1995.

[10] S. E. Fahlmann, C. Lebiere, "The Cascade Correlation learning architecture," in *Advances in Neural Information Processing Systems 11*, Ed: D. Touretzky, Morgan Kaufmann, San Mateo, CA, 1990, pp. 524-532.