In *International Conference on Neural Networks*,
Perth, Western Australia, 1995, IEEE Press.

Cascade Correlation: Derivation of a More Numerically Stable Update Rule

George H. John
Computer Science Department
Stanford University
Stanford, CA 94305
gjohn@cs.Stanford.EDU
http://robotics.stanford.edu/~gjohn

ABSTRACT

We discuss the weight update rule in the Cascade Correlation neural net learning algorithm. The weight update rule implements gradient descent optimization of the *correlation* between a new hidden unit's output and the previous network's error. We present a derivation of the gradient of the correlation function and show that our resulting weight update rule results in slightly faster training. We also show that the new rule is mathematically equivalent to the one presented in the original Cascade Correlation paper and discuss numerical issues underlying the difference in performance. Since a derivation of the Cascade Correlation weight update rule was not published, this paper should be useful to those who wish to understand the rule.

1. Introduction

The Cascade Correlation algorithm of Fahlman & Lebiere (1990) is an elegant, simple, and powerful tool for classification and regression. It is a constructive method for training a neural network; initially, there are no hidden units, only input units and output units. The algorithm automatically adds hidden units in order to fit the training data more closely. This frees the user from having to guess at a good network topology. Because of this freedom, and because of the computational strengths of Cascade Correlation, it has begun to enjoy widespread use in applications (McKenna, Ricketts, Cairns & Hussein 1992, Yamamoto & Zenios 1993).

At the core of the algorithm is the novel objective function used to train new hidden units, the *correlation* function. When new hidden units are added, their weights are repeatedly adjusted to maximize this objective function using some optimization algorithm requiring first-order derivative information. In fact, the algorithm uses either gradient descent or quickprop, an optimization method borrowing ideas from Gauss-Newton methods (Fahlman 1988, Gill, Murray & Wright 1981). To use these methods, one must first mathematically derive the equations for the first derivatives of the objective function with respect to the parameters (the weights, in this case).

In Fahlman & Lebiere (1990), the derivative of correlation with respect to the weights is given, but we have found that this equation for the derivative exhibits undesirable numerical behavior. We have derived a new equation for the gradient that is mathematically equivalent to the original, but which yields slightly better performance.

In Section 2 we describe the Cascade Correlation learning algorithm. Section 3 gives a derivation of our equation for the first derivatives of the correlation function, which differs in form from the derivative presented in the original paper. We then show that the two expressions for the first derivative are equivalent, and discuss the numerical properties leading to the superiority of our derivative expression. Section 4 demonstrates the similar but different performance of the update rule using the different expressions of the first derivative, showing the slight superiority of the method presented here, and Section 5 concludes the paper.

2. The Cascade Correlation Algorithm

Cascade Correlation is a method for both building a network (adding nodes) and for training (or optimizing) the weights of a network. *Cascade* refers to the method for building the networks—each output unit receives information from all input and hidden units, and when new hidden units are added they receive input from all network inputs and all preceding hidden units. The output of a unit is equal to the dot product of its weights and its inputs, transformed by a sigmoid, Gaussian, or linear activation function. Figure 1 shows the "cascading"

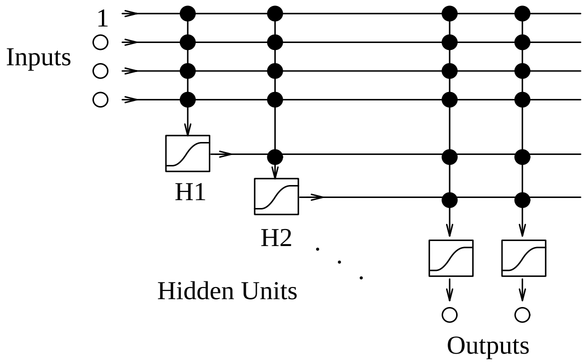

Fig. 1: The Cascade Correlation Learning Architecture.

architecture of a Cascade correlation network.

Correlation refers to the method used to train the weights of new hidden units. The output unit weights are always trained separately to minimize mean-squared error on the training set. When adding a new hidden unit, its weights are adjusted to optimize the correlation of its output with the errors of the previous network (without the new hidden unit). Thus, when added to the network, the new hidden unit's output should be helpful in correcting the errors of the previous network. The new hidden unit is trained by itself—no other weights are modified.

To begin defining terms and symbols precisely, each output unit is trained to minimize error, where the error of a particular output unit o is defined as $E_o = \sum_p (t_{p,o} - O_{p,o})^2$, where E_o is the error of output unit o, $t_{p,o}$ is the target or desired value for output unit o when the net is presented with pattern p, and $O_{p,o}$ is the observed value of output unit o when the net was presented with pattern p. This is simply the sum-squared error criterion.

If the resulting error after training the output units (using backpropagation or quickprop) is too high, a new hidden unit, called the *candidate unit*, is added. (The "too high" threshold and other parameters are discussed thoroughly in the documentation included in the Cascade Correlation software provided by White (1995).) The inputs to the candidate unit are the bias, all net inputs, and the outputs of all previous hidden units. We might expect to retrain the entire network using backpropagation to minimize the error on the training set, but this is not the case. Rather, Cascade Correlation fixes all weights in the network except for the candidate unit input weights. They it adjusts the weights of the candidate unit to maximize the *correlation* between its output and the errors of the output units before the candidate was added. After the hidden unit is trained it is added to the net, and the output units are retrained to minimize error.

The idea behind correlation is that if the new hidden unit could perfectly predict the errors of the previous network, it could be combined with the previous network to predict the output on each training sample exactly. Even if the new hidden unit does not perfectly correlate with the errors, it will still be useful in reducing the error of the network. Thus, correlation with the error of the output units is a reasonable metric for usefulness of the new hidden unit. Fahlman and Lebiere define correlation[1] as

$$S = \sum_o \left| \sum_p (V_p - \overline{V})(E_{p,o} - \overline{E_o}) \right| \tag{1}$$

where $E_{p,o}$ is the *residual output error* of output unit o for training pattern p, which is equal to $t_{p,o} - O_{p,o}$. V_p is the candidate unit's output on pattern p. $\overline{V}$ is equal to the average over all patterns p of V_p, and similarly $\overline{E}_o$ is the average over all patterns of $E_{p,o}$.

In backpropagation, gradient descent is used to minimize the mean-squared error objective function. Cascade Correlation uses gradient descent to find weights for the candidate units that maximize correlation with the previous net's errors as follows:

$$\begin{aligned} \Delta w_i &= \eta \frac{\partial S}{\partial w_i} \\ \frac{\partial S}{\partial w_i} &= \sum_o \sigma_o \sum_p f' I_{i,p} (E_{p,o} - \overline{E_o}) \ , \end{aligned} \tag{2}$$

where σ_o is equal to the sign of the $|\cdots|$ term in the correlation equation. In the next section we present a derivation of $\frac{\partial S}{\partial w_i}$ and show that it is equivalent to Equation 2.

3. The $\frac{\partial S}{\partial w_i}$ Controversy

In this section, we derive a new equation for $\frac{\partial S}{\partial w_i}$ and show that it is equivalent to the one presented in Fahlman & Lebiere (1990). However, because of the limits of floating-point hardware, the equations perform differently in practice, and we discuss the different numerical properties of the two formulas.

3.1. Derivation of $\frac{\partial S}{\partial w_i}$

Since there seems to be no published derivation of the Cascade correlation update rule, it seems worthwhile to provide a reasonable amount of detail in the derivation:

$$S = \sum_o \left| \sum_p (V_p - \overline{V})(E_{p,o} - \overline{E_o}) \right| \tag{3}$$

[1] This is actually the sum of the absolute covariance between the candidate unit and the previous net's error over all output units. The difference between covariance and correlation is simply a scaling term, so we will not discuss the statistical definitions further.

$$
\begin{aligned}
\frac{\partial S}{\partial w_i} &= \frac{\partial \sum_o \left|\sum_p (V_p - \overline{V})(E_{p,o} - \overline{E_o})\right|}{\partial w_i} \\
&= \sum_o \frac{\partial \left|\sum_p (V_p - \overline{V})(E_{p,o} - \overline{E_o})\right|}{\partial w_i} \\
&= \sum_o \sigma_o \frac{\partial \sum_p (V_p - \overline{V})(E_{p,o} - \overline{E_o})}{\partial w_i} \\
&\quad \text{where } \sigma_o \stackrel{\text{def}}{=} \operatorname{sgn} \sum_p (V_p - \overline{V})(E_{p,o} - \overline{E_o}) \\
&= \sum_o \sigma_o \sum_p \frac{\partial (V_p - \overline{V})(E_{p,o} - \overline{E_o})}{\partial w_i} \\
&= \sum_o \sigma_o \sum_p \frac{\partial (V_p - \overline{V})}{\partial w_i}(E_{p,o} - \overline{E_o}) \\
&\quad [E_{p,o} \text{ does not depend on } w_i] \\
&= \sum_o \sigma_o \sum_p \left(\frac{\partial V_p}{\partial w_i} - \overline{\frac{\partial V}{\partial w_i}}\right)(E_{p,o} - \overline{E_o}) \qquad (4)
\end{aligned}
$$

Now we just have to compute $\partial V_p / \partial w_i$. $V_p = f(\sum_i w_i I_{p,i})$, where f is the activation function (typically the sigmoid), and $I_{p,i}$ is the ith input to the candidate unit for the pth pattern.

$$
\begin{aligned}
\partial V_p / \partial w_i &= \frac{\partial f(\sum_i w_i I_{p,i})}{\partial w_i} \\
&= f' \frac{\partial (\sum_i w_i I_{p,i})}{\partial w_i} \\
&= f' I_{p,i}
\end{aligned}
$$

Plugging this into equation 4, we get:

$$
\frac{\partial S}{\partial w_i} = \sum_o \sigma_o \sum_p (f' I_{p,i} - \overline{f' I_i})(E_{p,o} - \overline{E_o}) \qquad (5)
$$

This disagrees with Fahlman & Lebiere (1990), which gives the result shown in Equation 2 (repeated in Equation 7 below).

3.2. The Equivalence of the Two Derivative Equations

Since an earlier version of this paper, Fahlman (personal communication) has pointed out that the update rule of Equation 5 is equivalent to the one presented in the original paper. Where the two methods differ in practice it is only because the method shown here is less susceptible to numerical errors. To recap, here are the two equations:

$$
\begin{aligned}
\frac{\partial S}{\partial w_i} &= \sum_o \sigma_o \sum_p (f' I_{p,i} - \overline{f' I_i})(E_{p,o} - \overline{E_o}) \quad (6) \\
\frac{\partial S}{\partial w_i} &= \sum_o \sigma_o \sum_p f' I_{p,i}(E_{p,o} - \overline{E_o}) \ . \qquad (7)
\end{aligned}
$$

Equation 6 gives the expression derived in this paper, while Equation 7 gives the expression reported in the original paper.We now show that the two are equivalent.

$$
\begin{aligned}
\frac{\partial S}{\partial w_i} &= \sum_o \sigma_o \sum_p (f' I_{p,i} - \overline{f' I_i})(E_{p,o} - \overline{E_o}) \\
&= \sum_o \sigma_o \sum_p f' I_{p,i}(E_{p,o} - \overline{E_o}) \\
&\quad - \sum_o \sigma_o \sum_p \overline{f' I_i}(E_{p,o} - \overline{E_o}) \\
&= \sum_o \sigma_o \sum_p f' I_{p,i}(E_{p,o} - \overline{E_o}) - \overline{f' I_i} \sum_o \sigma_o 0 \\
&= \sum_o \sigma_o \sum_p f' I_{p,i}(E_{p,o} - \overline{E_o})
\end{aligned}
$$

3.3. Numerical Properties

Numerically, subtracting the mean from each $f' I_{p,i}$ leads to more stability, since more bits of precision can be kept. For example, take $f' I_{1,i} = 1.00000001$ and $f' I_{2,i} = 1.00000002$. When multiplied by another number, the least significant bits may be lost, but if we first subtract the mean, we get $f' I_{1,i} - \overline{f' I_i} = -5 \times 10^{-8}$ and $f' I_{2,i} - \overline{f' I_i} = 5 \times 10^{-8}$, which allows us to retain full precision.

These intuitive arguments are supported by Neely (1966), who gives a nice discussion of the numerical issues arising in the computation of some common statistics: mean, standard deviation, and correlation. The original cascade correlation derivative corresponds to Neely's P_1 correlation statistic, while our derivative corresponds to his P_2 statistic. By running various numerical experiments, Neely concluded "It is evident ... that the correlations calculated using P_1 are in error for all [experiments]." P_2 gave much better estimates, although it exhibited greater instability than yet more sophisticated methods.

4. An Empirical Comparison

We ran two versions of Cascade Correlation, the standard algorithm which uses Equation 7 and JCascor which uses Equation 6. Experiments were performed using the Cascade Correlation software provided by White (1995). All parameters are set to their default values. The data is normalized by scaling linearly so that each attribute's range is $[0, 1]$. At the end of each trial, the total number of candidate unit epochs is recorded. Results are summarized in Table 4. (The sonar and vowel datasets are from White & Fahlman (1994), and the rest are from Murphy & Aha (1994).)

Each algorithm was run 20 times on each dataset. For each run, the total number of candidate unit

Table: 1: Experimental results, showing the average number of candidate unit epochs for each trial of Cascor and JCascor and the significance level at which we accept the hypothesis that the JCascor algorithm requires fewer candidate training epochs (using the Student's t-test).

Dataset	Cascor	JCascor	P(t)
Breast-Cancer	359±15	350±11	68%
Pima-Diabetes	2125±30	2080±24	87%
Sonar	64.3±2.6	66.9±2.2	23%
Vowel	849±20	842±18	59%
Vote	90.2±8.0	91.3±6.3	45%
Vote1	308±11	307±13	52%

epochs (the number of epochs spent training using the correlation objective function) was totaled. Table 4 reports the average number of epochs over all runs, plus or minus the standard deviation of the average. For each dataset, we use an unpaired t-test to test the hypothesis that the JCascor algorithm used fewer epochs than the Cascor algorithm. In no case can we accept this hypothesis with at least 90% confidence, although in most cases JCascor did require fewer epochs. Additionally, note that in all problems except for Vote1, JCascor had lower variance in the number of epochs than Cascor, indicating more stability.

Originally our hypothesis was that JCascor would require fewer epochs, and that the difference would be significant. (This was based on Neely's (1966) results.) There are two possible reasons why we did not get such results in practice. The most probable cause is the iterative gradient descent algorithm – small numerical errors in the calculation of correlation or its derivative will only result in a step of the optimization being in the wrong direction. However, this is of little consequence, because when optimizing an unknown function, taking a step in the direction of the gradient is only a heuristic. Numerical errors in calculating the derivative of the objective function do not accumulate but rather are overcome by the deviations of the objective function's surface itself from the linear model assumed by the gradient descent algorithm. More aggressive optimization algorithms, such as Quickprop (Fahlman 1988), might be more sensitive to errors in the gradient, and might be more improved by using Equation 6.

5. Conclusion

We have presented a derivation of the candidate unit weight update rule in the Cascade Correlation algorithm, resulting in an expression equivalent to the one in the original paper but with better numerical properties. Since issues of computational statistics are not often studied in the neural network literature, we decided to empirically determine the significance of the correlation expression used in the algorithm. Numerical precision seems to be insignificant in this case—the iterative gradient descent algorithm seems forgiving of small numerical errors. The results from an unpaired t-test on several datasets show that the new weight update rule does not result in significantly fewer training epochs at the 90% confidence level.

Acknowledgments

The author is supported by a National Science Foundation Graduate Research Fellowship. Thanks to Scott Fahlman for comments and corrections, and to Jerry Friedman for the reference to Neely.

References

Fahlman, S. E. (1988), An empirical study of learning speed in back-propagation networks, Technical Report CMU-CS-88-162, Carnegie Mellon University.

Fahlman, S. E. & Lebiere, C. (1990), The cascade-correlation learning architecture, *in* D. Touretzky, ed., "Advances in Neural Information Processing Systems", Vol. 2, Morgan Kaufmann.

Gill, P. E., Murray, W. & Wright, M. H. (1981), *Practical Optimization*, Academic Press, New York.

McKenna, S. J., Ricketts, I. W., Cairns, A. Y. & Hussein, K. A. (1992), Cascade-correlation neural networks for the classification of cervical cells, *in* "IEE Colloquium on Neual Networks for image processing applications", IEE, London, UK, pp. 51–54.

Murphy, P. M. & Aha, D. W. (1994), "UCI repository of machine learning databases", Available by anonymous ftp to `ics.uci.edu` in the `pub/machine-learning-databases` directory.

Neely, P. M. (1966), "Comparison of several algorithms for computation of means and correlation coefficients", *Communications of the ACM* **9**, 496–499.

White, M. (1995), "Cascade correlation learning algorithm v1.2", Available by ftp to `ftp.cs.cmu.edu` in the `/afs/cs/project/connect/code/supported` directory.

White, M. & Fahlman, S. E. (1994), "Neural-bench benchmark collection", Available by anonymous ftp to `ftp.cs.cmu.edu` in the `/afs/cs/project/connect/bench` directory.

Yamamoto, Y. & Zenios, S. A. (1993), "Predicting prepayment rates for mortgate-backed securities using the cascade-correlation learning algorithm", *The Journal of Fixed Income* **2**(4), 86–96.