OPTIMIZATION IN COMPANION SEARCH SPACES: THE CASE OF CROSS-ENTROPY AND THE LEVENBERG-MARQUARDT ALGORITHM

Craig L. Fancourt and Jose C. Principe
Computational NeuroEngineering Laboratory
Department of Electrical Engineering
University of Florida, Gainesville, FL 32611
email: fancourt@cnel.ufl.edu, principe@cnel.ufl.edu

Abstract

We present a new learning algorithm for the supervised training of multilayer perceptrons for classification that is significantly faster than any previously known method. Like existing methods, the algorithm assumes a multilayer perceptron with a normalized exponential (softmax) output trained under a cross-entropy criterion. However, this output-criteria pairing turns out to have poor properties for existing optimization methods (backpropagation and its second order extensions) because second-order expansion of the network weights about the optimal solution is not a good approximation. The proposed algorithm overcomes this limitation by defining a new search space for which a second-order expansion is valid and such that the optimal solution in the new space coincides with the original criterion. This allows the application of the Levenberg-Marquardt search procedure to the cross-entropy criterion, which was previously thought applicable only to a mean square error criteria.

1. INTRODUCTION

In multilayer perceptron (MLP) design, some emphasis has been placed on proper pairing of the network architecture and criterion [2], [3]. Typically, the criterion is dictated by the nature and statistics of the problem to be solved, while the final output transfer function is matched to the criterion so that the gradient is not attenuated by the derivative of the output function. In this paper, we are primarily concerned with a particular criterion-output pairing that is useful for classification; namely, the cross-entropy criteria and the normalized exponential (softmax) output function.

Unfortunately, little consideration has been given to the effect of such pairings on optimization techniques that utilize second-derivatives. We will show shortly that choosing the output transfer function on the basis of the gradient alone can cause the Hessian to become ill-conditioned, rendering invalid the quadratic assumption on which many advanced optimization search procedures are based.

This has led us to postulate that a conceptually appealing criterion may not necessarily be the best one for efficient optimization. This idea is not new. Based on information geometric considerations, the so-called natural gradient [4] utilizes in effect the gradient from the original criteria in conjunction with a Hessian-like term based on the Fisher information matrix.

Our approach is to find a companion search space that has a one-to-one mapping with the original search space (i.e. an optimal solution in one is an optimal solution in the other), but for which the quadratic approximation is always valid. The simplicity of the solution is startling. Conceptually, we envision a pseudo target at the *input* side of the final output nonlinearity, which we train under a mean squared error criterion. Unfortunately, we don't actually know what the pseudo target should be because it is a complex function of the data. For example, for the classification task, the pseudo target is a function of the separation of the classes and the proximity of an exemplar to a discriminant plane. Since we don't know the pseudo target, we use the Levenberg-Marquardt (L-M) approximation of the Hessian because it depends only on the operating point of the network and not on the (pseudo) target. This still leaves the question of how to estimate the gradient that clearly does depend on the pseudo target. Here, the proper choice of the criteria and output function, when properly paired, allows for backpropagating the gradient unattenuated through the final nonlinearity. That is, for weight update we use the gradient calculated at the output using the original criteria, and the L-M approximation to the Hessian calculated at the input to the final nonlinearity. This is shown symbolically in Figure 1(b) and is contrasted to the more traditional approach in Figure 1(a).

2. SIMULATIONS

Because the experimental results are so strong, we break from tradition and present them first. The benchmark data sets are classification tasks, which we divide into two categories.

2.1 Separable problems

The first category is data sets that are perfectly separable. Generally, the purpose of such data sets is to test the ability of a learning algorithm to find highly non-linear discriminant functions, rather than to test generalization

2.2 Non-separable problems

Most real world data sets are not perfectly separable and do not require discriminant functions as complicated as the previous data sets. Here, the goal is to test the ability of the algorithm to capture the underlying statistics of the problem, as indicated through the trained network's ability to generalize on previously unseen data generated from the same underlying distributions. We will not address the non-separable problem case in this paper.

.

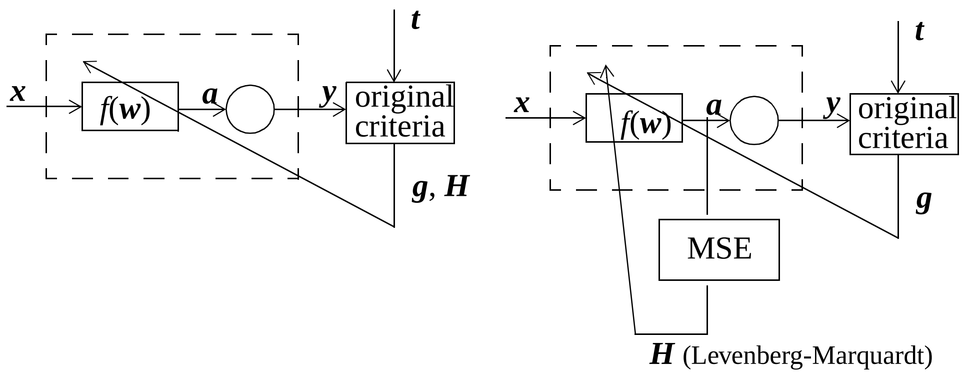

Figure 1. (a) standard optimization methods; (b) proposed hybrid optimization.

Table 1: Benchmark results: perfectly separable problems

Problem / Result	2-class spiral	4-bit parity	9-bit parity
Exemplars	194	16	512
MLP Architecture	2-40-1	4-4-1	9-9-1
Total # of weights	161	25	100
Runs converging (out of 100)	97	99	65
Mean # epochs (also back passes)	112	137	170
Mean # forward passes	333	394	491
Mean # matrix inversions	221	257	321

3. EXPOSITION OF THE ALGORITHM

We now present an intuitive development of the algorithm. Consider the network in Figure 1(a) where ***x*** is the input, ***a*** is the input to the final transfer function that produces the output ***y***, and ***t*** is the target (desired response). The criteria *J*(***y***,***t***) compares the output and target in some fashion. Assuming an infinite data set, the expectation of the criteria yields the cost: $C = E[J]$. Minimizing the cost with respect to the free weights, ***w***, in the network yields the best fit of the network to the data.

Newton's algorithm assumes that the cost in the region of weight space around the optimal weights can be well approximated by a 2nd order Taylor series expansion. If this is true, then it is easy to show that the weight update at each iteration is given by

$$\Delta \boldsymbol{w} = -\boldsymbol{H}^{-1} \cdot \boldsymbol{g} \qquad (1)$$

where ***g*** is the gradient and ***H*** is the Hessian, to be defined shortly.

3.1 Natural pairing between criteria and output function

The gradient of the cost function commutes with the expectation operator.

$$\boldsymbol{g} = \frac{\partial C}{\partial \boldsymbol{w}} = E\left[\frac{\partial J}{\partial \boldsymbol{w}}\right] \qquad (2)$$

where ***w*** is a vector of all the weights in the network. The derivative of the criteria is given by

$$\frac{\partial J}{\partial \boldsymbol{w}} = \left(\frac{\partial J}{\partial \boldsymbol{y}^{\dagger}} \cdot \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{a}^{\dagger}} \cdot \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{w}^{\dagger}}\right)^{\dagger} \qquad (3)$$

It can be shown that if the criteria is derived from the exponential family of distributions [5], then we can always find an output function such that

$$\frac{\partial J}{\partial \boldsymbol{y}^{\dagger}} \cdot \frac{\partial y}{\partial \boldsymbol{a}^{\dagger}} = (\boldsymbol{y} - \boldsymbol{t})^{\dagger} \Rightarrow \frac{\partial J}{\partial \boldsymbol{w}} = \frac{\partial \boldsymbol{a}^{\dagger}}{\partial \boldsymbol{w}} \cdot (\boldsymbol{y} - \boldsymbol{t}) \qquad (4)$$

and the gradient is proportional to the error between the network output and target:

$$\boldsymbol{g} = E\left[\frac{\partial \boldsymbol{a}^{\dagger}}{\partial \boldsymbol{w}} \cdot (\boldsymbol{y} - \boldsymbol{t})\right]. \qquad (5)$$

The advantage of finding a natural pairing between the criteria and output function is that the error is not attenuated as it is back-propagated through the dual of the output nonlinearity.

3.2 The true Hessian

From (2), the Hessian is

$$H = \frac{\partial^2 C}{\partial \boldsymbol{w} \partial \boldsymbol{w}^{\dagger}} = E\left[\frac{\partial^2 J}{\partial \boldsymbol{w} \partial \boldsymbol{w}^{\dagger}}\right] \qquad (6)$$

but

$$\frac{\partial^2 J}{\partial \boldsymbol{w} \partial \boldsymbol{w}^\dagger} = \frac{\partial \boldsymbol{a}^\dagger}{\partial \boldsymbol{w}} \cdot \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{a}^\dagger} \cdot \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{w}^\dagger} + \frac{\partial^2 \boldsymbol{a}}{\partial \boldsymbol{w} \partial \boldsymbol{w}^\dagger} \cdot (\boldsymbol{y} - \boldsymbol{t}) \tag{7}$$

where the second term in (7) is a tensor product. In the limit of an infinite data set, if the network outputs approximate the posterior probabilities, then it is easy to show [6] that this term goes to zero. That leaves us with the first term, which contains the derivative of the output function with respect to its input: $\partial \boldsymbol{y} / \partial \boldsymbol{a}^\dagger$. For a nonlinear transfer function, this represents an attenuation of the Hessian which, under certain circumstances, can cause the Hessian to become very small, in turn implying that the a second order Taylor expansion is no longer a good approximation. This can seriously impair the performance of second-order optimization methods, as we will see shortly. If we try to remove the effects of the derivative by using a linear output unit, then through (3) we are immediately led back to a mean squared error criteria:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{a}^\dagger} = \mathrm{I} \;\Rightarrow\; \frac{\partial J}{\partial \boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{t} \;\Rightarrow\; J = \|\boldsymbol{y} - \boldsymbol{t}\|^2 . \tag{8}$$

In other words, the only output-criteria pairing that neither attenuates the gradient nor the Hessian is a linear output and mean squared error criteria, respectively. For any other output-criteria pairing, we cannot simultaneously create a component that is linear in the backpropagation plane for both the calculation of the gradient and Hessian.

3.3 The Levenberg-Marquardt approximation to the Hessian

For a linear processing element (PE) and a MSE cost function, the L-M approximation to the Hessian yields directly [6],

$$J_{\text{mse}}(n) = \frac{1}{2} \sum_{k=1}^{K} (y_k - t_k)^2 \quad , \tag{9}$$

$$\frac{\partial}{\partial w_i} J_{\text{mse}}(n) = \sum_{k=1}^{K} (y_k - t_k) \cdot \frac{\partial a_k}{\partial w_i} \quad , \tag{10}$$

$$\frac{\partial^2 J_{\text{mse}}(n)}{\partial w_i \partial w_j} \approx \sum_{k=1}^{K} \frac{\partial a_k}{\partial w_i} \cdot \frac{\partial a_k}{\partial w_j} \quad . \tag{11}$$

Notice that the advantage of the L-M approximation is that the modified Hessian becomes only dependent on the state of the network, and independent of the target value.

For a nonlinear PE, there are two modifications to be considered as shown in (7), and in [6] algorithms are developed to handle nonlinearities. However, we can always use (11) allowing one to use the same L-M training algorithm for both regression and classification problems. This is dangerous for nonlinear PEs and MSE because the direction of the gradient is altered and there is no guarantee that the global minimum is obtained. Our goal was exactly to seek a cost function for which the gradient before a sigmoidal nonlinearity was undisturbed from the one obtained with a linear PE and MSE criterion such that (11) could be applied. We reasoned that in such case the direction of the gradient would be maintained, although the cost function for which we will be computing the Hessian was not the original cost function (i.e. the values of the gradient would differ). Of course, the ultimate test is how well the approximations hold up under simulation, but Table I shows that there is good reason to study further this idea.

4. APPLICATION TO A SOFTMAX OUTPUT AND A CROSS-ENTROPY CRITERION

We now show how this reasoning can be applied to a multilayer perceptron with a softmax output trained under a cross-entropy criteria. The cross entropy criterion

$$J = \sum_{n=1}^{N} J(n) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_k(n) \cdot \log[y_k(n)] \tag{12}$$

is used in conjunction with a softmax activation function at the final layer, where the softmax transfer function is given by

$$y_k = \frac{\exp[a_k]}{\sum_{l=1}^{K} \exp[a_l]} \tag{13}$$

where the a_k are the inputs to the softmax. Since derivatives distribute over summation, we can find the gradient and Hessian of the instantaneous cost, $J(n)$, and then sum the results over the entire data set. Dropping the temporal index for ease of reading, the ith component of the instantaneous gradient is

$$\frac{\partial}{\partial w_i} J(n) = -\sum_{k=1}^{K} \frac{t_k}{y_k} \cdot \frac{\partial y_k}{\partial w_i} \tag{14}$$

This is easily evaluated using the properties of the softmax. Since

$$\frac{\partial y_k}{\partial a_l} = y_k(\delta_{kl} - y_l) \tag{15}$$

it immediately follows that

$$\frac{\partial y_k}{\partial w_i} = \sum_{l=1}^{K} \frac{\partial y_k}{\partial a_l} \cdot \frac{\partial a_l}{\partial w_i} = y_k \left(\frac{\partial a_k}{\partial w_i} - \sum_{l=1}^{K} y_l \cdot \frac{\partial a_l}{\partial w_i} \right) \tag{16}$$

Plugging (16) into (14), the instantaneous gradient results

$$\frac{\partial}{\partial w_i} J(n) = \sum_{k=1}^{K} (y_k - t_k) \cdot \frac{\partial a_k}{\partial w_i} . \tag{17}$$

Note that this gradient is equivalent to a mean square error cost function where the softmax acts like a linear transfer function in the backpropagation plane and does not attenuate the backpropagated error. This is a great advantage over using a mean square error criteria in conjunction with sigmoid activation functions at the output layer. Although the outputs of both networks *tend* to sum to one *after* training, the attenuation of the error through the sigmoid layer slows down the training relative to an equivalently sized softmax network. This is well known in the literature [6]. The interesting part is when we compute the Hessian to this network and cost function.

4.1 Relationship between the exact and pseudo Hessian

The exact Hessian of the instantaneous cost function can be found by taking the partial derivative of (17) and using (16):

$$\frac{\partial^2 J(n)}{\partial w_i \partial w_j} = \sum_{k=1}^{K} \frac{\partial a_k}{\partial w_i} \cdot \frac{\partial a_k}{\partial w_j} \cdot y_k(1-y_k) - \sum_{\substack{k,l=1 \\ k \neq l}}^{K} \frac{\partial a_k}{\partial w_i} \cdot \frac{\partial a_l}{\partial w_j} \cdot y_k y_l + \sum_{k=1}^{K} (y_k - t_k) \cdot \frac{\partial^2 a_k}{\partial w_i \partial w_j} \quad . \tag{18}$$

If we assume that the second derivatives are approximately the same for all K outputs, then the third term is zero, since

$$\sum_{k=1}^{K} (y_k - t_k) \cdot \frac{\partial^2 a_k}{\partial w_i \partial w_j} \approx \frac{\partial^2 a}{\partial w_i \partial w_j} \sum_{k=1}^{K} (y_k - t_k) = 0 \tag{19}$$

and to the constraints that the outputs and targets sum to one. This should be contrasted with the analogous term in the Levenberg-Marquardt approximation for mean square error and sigmoid outputs. There, a summation over $y_k - t_k$ also results but, because there are no constraints on the outputs, the vanishing of the summation can only be justified when considering an expectation over the entire data set at the optimal solution. Thus, there is good reason to expect the L-M approximation to the Hessian to be better for a softmax output, at least early in the training. The latter caveat is required due to the vanishing of the Hessian at the optimal solutions previously mentioned.

Dropping the second-order derivatives and rearranging the remaining terms, there results

$$\frac{\partial^2 J(n)}{\partial w_i \partial w_j} \approx \sum_{k=1}^{K} \frac{\partial a_k}{\partial w_i} \cdot \frac{\partial a_k}{\partial w_j} \cdot y_k(1-y_k) - \sum_{\substack{k,l=1 \\ k \neq l}}^{K} \frac{\partial a_k}{\partial w_i} \cdot \frac{\partial a_l}{\partial w_j} \cdot y_k y_l \quad . \tag{20}$$

Note that the factor $y_k(1-y_k)$ in the first summation tends to leave small outputs unchanged and reduces large outputs. It will thus have a more uniform distribution across the outputs than y_k itself and can be ignored to a first approximation. In addition, if one output dominates the others, as would be expected once some learning has occurred, then all the cross terms $y_k y_l$ in the second summation will be small. Finally, we arrive at the approximation

$$\frac{\partial^2 J(n)}{\partial w_i \partial w_j} \approx \sum_{k=1}^{K} \frac{\partial a_k}{\partial w_i} \cdot \frac{\partial a_k}{\partial w_j} \tag{21}$$

which is exactly (11), obtained for the linear PE case. We call this approximation the pseudo Hessian.

5. CONCLUSIONS

Using this reasoning, we may be lead to expect that (21) would perform poorly because there are many approximations. However, the results presented in Table I show the power of the combination of cross-entropy criterion and the pseudo Hessian, so there is something fundamental here that needs to be further investigated. The results in Table I are orders of magnitude better than the ones presented by [2] and even better than the ones presented in [4] using the natural gradient. Unfortunately, Amari and co-workers never compared their algorithm with second order methods, just with the straight gradient descent.

Underlying this work is the hint that faster adaptation can be achieved in companion performance surfaces, that is, performance surfaces that preserve the location of the global minimum of the true performance surface but that have higher slopes everywhere. Since adaptation seeks the location of the minimum in parameter space while the value at the point is irrelevant, we the designers have an extra degree of freedom for fast adaptation. This reasoning leads to the design of new search methods that preserve the gradient direction of the original performance surface, but alter its “slopes”. The combination of the cross-entropy criterion and the L-M approximation of the Hessian seems to be such a combination, but others may exist.

Acknowledgment: This work was partially supported by NSF grant ECS-990394.

References

[1] Duda R. and Hart P., *Pattern classification and scene analysis*, John Wiley & Sons, New York, 1973.

[2] Moller M.F., A scaled conjugate gradient algorithm for fast supervised learning, *Neural Networks*, vol. 6, pp. 525-533, 1993.

[3] Osowski S., Bojarczak P., and Stodolski M., Fast second order learning algorithm for feedforward multilayer neural networks and applications, *Neural Networks*, vol. 9, no. 9, pp. 1583-1596, 1996.

[4] Park H., Amari S., and Fukumizu K., Adaptive natural gradient learning algorithms for various stochastic models, submitted to *Neural Networks*, 1999.

[5] Rumelhart D., Durbin R., Golden R., and Chauvin Y., Backpropagation: the basic theory, in Chauvin Y. and Rumelhart D. (eds.), *Backpropagation: Theory, Architectures, and Applications*, Lawrence Earlbaum Associates, Hillsdale, New Jersey, (1995).

[6] Bishop C., *Neural Networks for Pattern Recognition*, Oxford Press, 1996.