In *International Conference on Artificial Intelligence (IC-AI'99), June 28-July 1 1999, Las Vegas.*

RE-VISITING BACKPROPAGATION NETWORK OPTIMIZATION: TOWARDS MAXIMALLY PRUNED NETWORKS

A Goh and SSR Abidi
USM Computer Sciences
11800 Penang, Malaysia
Email: **alwyn@cs.usm.my**

Abstract

Backpropagation (BP) Neural Network (NN) error functions enable the mapping of data vectors to user-defined classifications by driving weight matrix modifications so as to reduce classification error over the training data set. Conventional BP error functions are usually only implicitly dependant on the weight matrix, however an explicit *penalty* term can be added so as to force numerically insignificant weights closer to zero. In our investigation, BP training is undertaken as a prelude to a pruning stage that selectively removes functionally unimportant weight matrix elements, thereby resulting in sparser network connectivity more suited for subsequent rule extraction. This paper investigates the usage of several error and activation functions in the effort to produce maximally *clean* network connections.

Keywords: Neural Networks, Backpropagation, Network Optimisation, Network pruning, Penalty term.

1 Introduction

NNs are usually designed with the number of input and output nodes fixed respectively by the feature descriptor and classification dimensionality, hence network optimization is equivalent to determining the size of the hidden layer and associated connective linkages; which can be obtained by node addition to an initially undersized layer, or via the reverse process of node elimination starting from an oversized hidden-layer. This paper adopts a refinement on the latter approach [1, 2, 3], in which inter-node connections—as opposed the nodes themselves—are progressively removed until classification performance degrades below the acceptability threshold. The objective is therefore to obtain a *sparse* but accurate (with respect classification) weight matrix.

Sparse NN connectivity is directly linked to the occurrence of numerically insignificant connection weights, a process which is encouraged by the incorporation of an adjustable *penalty* term [4, 5]. A properly calibrated penalty term causes small weight matrix elements to be exposed to a disproportionately large downward gradient thereby resulting in further numerical suppression. The subsequent network pruning involves subjecting the *trained* weights to a significance test which leads to the elimination of functionally unimportant weights from the NN.

The modified BP training and pruning algorithms featured in this paper follows the approach of Setiono [4], and extends the analysis to alternative NN architectures. Straightforward examination of the pruning condition suggests that certain connective architectures are inherently easier to optimize, which can be tested on the basis of experimental data. Our results will demonstrate that careful selection of certain NN architectural elements—activation and error functions in particular—

enhances training-pruning efficiency. Network pruning is a precursor to most methods to extract symbolic rules from learnt NN [5, 6, 7]; in this context, our results are particularly attractive since sparser weight matrices are expected to yield a more compact set of symbolic rules.

2 Basic Backpropagation: A Brief Overview

In order to correctly classify data vectors, the hidden-input and output-hidden network connections—represented by the weight matrices $\boldsymbol{W_{ji}}$ and $\boldsymbol{W_{kj}}$; with indices ($\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$) representing the input, hidden and output layers—are progressively adjusted so as to minimise global classification error. Weight modification proceeds in the direction opposite to the gradient of the error function with respect to the weights. The error gradient of such networks therefore depends on the following network features [3]:-

(a) *Error function*: In our analysis, we use both quadratic and logarithmic error functions. The latter is logarithmically divergent at the domain boundaries, and is therefore expected to result in BP networks with superior error convergence characteristics.

(b) *Node activation function*: The BP networks reported here feature both sigmoid (σ) and hyperbolic-tangent (δ) activations for hidden-layer nodes, while all output activations are sigmoidal. These BP networks will henceforth be denoted as δσ and σσ respectively.

In total, we analyse four basic NN architectures i.e. *δσ-quad, δσ-log, σσ-quad* and *σσ-log* with the suffix describing the error function, as compared against Setiono's [4] analysis which only features *δσ-log* networks. In our analysis, the weight matrices of trained networks are subsequently *pruned* to eliminate functionally insignificant weights.

3 Penalty Term in the Error Function

For network pruning, adding a penalty term to the error function whilst training the network is an accepted practice [4, 5, 9]. The *penalty* term is designed to further reduce weight matrix elements which are numerically small—in anticipation that such weights may be safely eliminated during post-training optimisation—without significantly affecting overall classification performance. The penalty function should exhibit a strongly localised minimum in the $\boldsymbol{w = 0}$ neighbourhood, and have a set of continuous parameters ($\boldsymbol{\alpha_i}$) which can be adjusted so as to *fine-tune* the width and steepness. Following Setiono, we employ the function:

$$P^{poly}(w) = \sum_{kj}\sum_{ji}\left[\alpha_1 \frac{\alpha_2 w^2}{1+\alpha_2 w^2} + \alpha_3 w^2\right] \qquad (1)$$

which has a single global minimum at $\boldsymbol{w=0}$ for parameter values satisfying $\boldsymbol{\alpha_1 >> \alpha_3}$ and $\boldsymbol{\alpha_2 >> 1}$. The gradient ∇P should also satisfy the following criteria:-

- negligible for numerically large $\boldsymbol{w}$ values
- large and negative for numerically small values near $\boldsymbol{w \cong 0}$,

the net effect of which causes the penalty effect to come into play only for weak inter-node connections.

The penalty functions used in this paper employ the numerical values $\boldsymbol{\alpha_3 = 10^{-4}\ \alpha_1}$ and $\boldsymbol{\alpha_2 = 10}$. In qualitative terms, the size of the $\boldsymbol{\alpha_1}$ parameter determines the relative importance of the penalty term with regards the conventional error function. Larger $\boldsymbol{\alpha_1}$ values are therefore associated with more aggressive weight matrix optimisation, on the other hand it also leads to a more *choppy* descent towards a global minimum in the error function. We speculate this is due to implicit $\boldsymbol{w}$ dependence (in the conventional error term) exercising a *smoothing* effect, thereby ensuring a lesser likelihood of *temporary* convergence into a local minimum on the error surface.

4 Optimization of NN Connectivity

We use the same weight matrix optimization criteria as Setiono [4, 5], which is based on a first-order Taylor expansion of the node output

with respect to variation in the weight matrix element of interest i.e.

$$y(w) - y(W_0) = \nabla_w y \Big]_{W_0} (w - W_0) \qquad (2)$$

Setting $w_0 = 0$ and comparing the right-hand side of (2) against some predetermined optimisation error ε_2 therefore allows us to determine whether a connective weight can be "safely" removed without jeopardising classification accuracy, which would also depend on the maximum allowable training error ε_1. The combined (training followed by optimisation) error must still result in correct classification ie $\varepsilon = \varepsilon_1 + \varepsilon_2 < 0.5$ for a sigmoidal output layer. A high optimisation tolerance ε_2 will result in a large number of zeroed weight matrix elements, on the other hand the correspondingly low training tolerance ε_1 means more training cycles—or perhaps failure to successfully train. After some experimentation, we determined that $\varepsilon_1 \cong \varepsilon_2$ results in an optimal balance between training time minimisation and connectivity optimisation. The objective is to achieve a sparse weight matrix as that would lead to a compact set of symbolic rules.

Imposing an upper-limit on the right-hand side of (2) and evaluating the node output gradient allows the optimisation conditions to be expressed as follows:-

$$|W_{kj}||W_{ji}| \le 16\varepsilon_2 \quad \text{and} \quad |W_{kj}||W_{ji}| \le 4\varepsilon_2$$

for W_{ji} elements (σσ and δσ, respectively)

$$|W_{kj}| \le 4\varepsilon_2$$

for W_{kj} elements (both cases) (3)

with the first condition—specifying constraints on W_{ji} pruning—dependant on all W_{kj} elements connected to the j-th hidden node. This dictates that hidden-input connections be eliminated prior to output-hidden optimisation. The weight removal algorithm is presented elsewhere, and will not be repeated here.

Note the four-fold difference in the W_{ji} elimination conditions for σσ compared against δσ networks used by Setiono [4, 5]. Equation (3) can be interpreted as a qualitative statement with regards the relative efficiency of σσ and δσ hidden-input weight matrix optimisation. This assertion constitutes the primary motivation for the experiments outlined in the following section—in which σσ and δσ NNs are trained and pruned using the same training datasets, and the resultant weight matrices compared. Networks with σσ architectures are expected to prune more effectively, given equal penalty function strengths.

5 Experimental Results

The experiments described below were designed to observe pruning efficiency in response to the following parameters:-

- choice of base error function: log vs. quad
- choice of activation functions: δσ vs. σσ
- variation in the penalty function strength: $\alpha_1 \in \{0, 0.005, 0.010, \ldots\}$, with α_2 and α_3 values selected as previously explained

The training datasets we used were selected from the University of California at Irvine (UCI) Machine Learning Repository, primarily for their size and simplicity. The dimensionality of the training datasets and the NNs used to represent them is tabulated below:-

Dataset		***Node layer size***			***Matrix size***	
Name	***Size***	***i***	***j***	***k***	W_{ji}	W_{kj}
Lenses	24	6	3	3	18	9
Balance	625	20	4	3	80	12

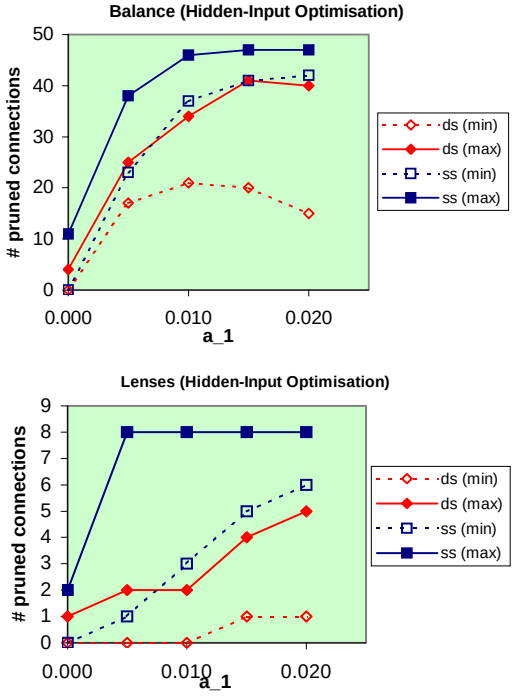
Table 1: Training Datasets

All NNs were coded using standard C++, and executed as batch-jobs on Windows-95 PCs. Connectivity optimization was only performed on successfully trained networks. We imposed an upper-limit cutoff—of order 10^6—with regards the number of training cycles, which resulted in the training-pruning process lasting anywhere from a few seconds to several minutes. With these constraints, it was found that:

- For the *balance* data-set, all δσ-quad and σσ-quad networks with $\alpha_1 > 0$ were not trainable.
- For the smaller *lenses* data-set, only δσ-quad networks were training-convergent.

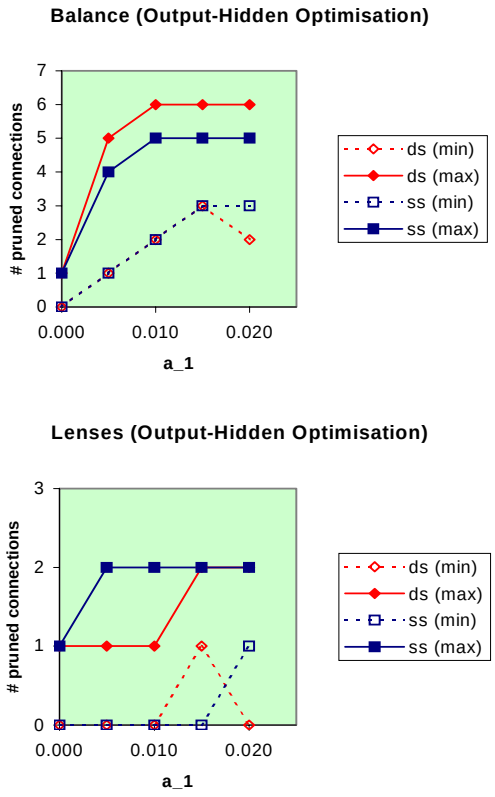
leading to the conclusion that NNs with quadratic error functions are significantly more difficult to optimize compared against those with logarithmic error functions. The remainder of this section will discuss the pruning of weight matrices produced by logarithmic error minimization.

For δσ-log and σσ-log NNs, training-pruning was performed using a sequence of increasingly strong penalty functions. It was determined that optimisation scales with α_1 only up to a certain point, beyond which further efficiency was not obtained. Even greater increases in α_1—resulting in error functions with a relatively large explicit (penalty) component—lead to NNs which demonstrate *choppy* error minimization, and which tend not to train properly. This trend was observed for δσ-log and σσ-log NNs trained on both datasets, and also for δσ-quad networks trained using the *lenses* data-set. Given the previously established training cycle cutoff, we found that both δσ-log and σσ-log networks with $\alpha_1 > 0.02$ were not training convergent. Graphs (1) and (2) below illustrate the relationship between weight matrix optimization and penalty function strength.



Graph 1: W_{ji} optimization for both datasets

Examination of data associated with hidden-input weight optimization yields another interesting conclusion, namely that σσ-log networks consistently prune more easily compared against their δσ-log equivalents. Our experiments indicate that *minimal* σσ-log optimisation is essentially equal or even slightly superior to *maximal* δσ-log pruning. This validates the qualitative accuracy of Equation (3), nevertheless we do not find σσ-log networks to be superior by a factor of four. One possible interpretation is to regard this inequality as the theoretical upper-limit for the comparative pruning efficiency of σσ and δσ networks, which cannot be attained in practice due to the probabilistic nature of the BP algorithm. It is therefore fairly safe to assert that σσ optimisation results in a sparser weight matrix.



Graph 2: W_{kj} optimization for both datasets

The output-hidden optimisation data shows no significant difference between the performance of σσ-log and δσ-log networks. This is consistent with the lower equation in (3),

which is applicable towards all networks featuring sigmoidal activation at the output layer. Graph (2) does however verify the effect of penalty function strength on overall pruning efficiency.

6 Concluding Remarks

The experimental results discussed in the previous section verifies the effectiveness of Setiono's algorithm in suppressing and subsequently eliminating small inter-node connections. In all experiments, it was found that pruning efficiency increases with penalty function strength but only up to an optimal point. We were thus able to obtain significantly more sparse weight matrices—with up to half of the original (fully-connected) network elements removed in some cases—while retaining high classification accuracy.

Penalty function usage seems to be far more effective when combined with logarithmic—rather than the more commonly used quadratic—error function. In fact, we found quadratic error NNs difficult (and sometimes impossible) to train even with an extremely weak penalty term. One possibility is that the explicitly-dependent penalty term leads to an overall error surface which contains more local minima, and is therefore more likely to *trap* the gradient descent. If this is the case, then logarithmic error NNs are certainly better equipped to handle addition of a penalty term due to the steeper overall error surface.

We also found that NN architecture has a major effect on optimization efficiency. Networks featuring hidden and output-layer ($\sigma\sigma$) sigmoidal activation had measurably superior connectivity pruning when compared against networks with hidden-layer tanh ($\delta\sigma$) nodes. This follows directly from the functional dependence of the hidden-layer nodes, which leads to a $\boldsymbol{W_{ji}}$ elimination condition which is easier to satisfy for $\sigma\sigma$ networks. As expected, there was no significant difference between $\sigma\sigma$ and $\delta\sigma$ networks with regards output-layer pruning efficiency.

In conclusion, $\sigma\sigma$-log NNs were ascertained to have the best training-pruning characteristics over the entire range of penalty function strengths. This combination of error and activation functions, in fact, almost always out-performs the $\delta\sigma$-log NNs used in Setiono's analysis—at least for the datasets under consideration. We are currently extending our investigation to larger datasets, both in terms of input-output nodes and training examples. The other area of interest is symbolic rule extraction, for which a larger degree of connective optimisation should correlate with a more compact rule-set. Preliminary results indicate this is usually the case, we will present a comprehensive analysis in a follow-up paper.

References

[1] FL Chung & T Lee (1992). *A Node Pruning Algorithm for Backpropagation Networks.* Int J Neural Systems, Vol. 3(3), pp. 301-314.

[2] ED Karnin (1990). *A Simple Procedure for Pruning Backpropagation Trained Neural Networks.* IEEE Trans Neural Networks, Vol. 1(2), pp. 239-242.

[3] R Watrous (1987). *Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimisation.* Proc IEEE 1st Int Conf Neural Networks, IEEE Press, NY, pp. 619-627.

[4] R Setiono (1997). *A Penalty Function Approach for Pruning Feedforward Neural Networks.* Neural Computation, Vol. 9 (1), pp. 185-204.

[5] R Setiono (1997). *Extracting Rules from Neural Networks by Pruning and Hidden Unit Splitting.* Neural Computation, Vol. 9(1), pp. 205-225.

[6] LM Fu (1994). *Rule Generation from Neural Networks.* IEEE Trans Systems, Man & Cybernetics, Vol. 24(8)

[7] R Setiono (1995). *Understanding Neural Networks via Rule Extraction.* Proc 14th Int Joint Conf Artificial Intelligence, Monreal, pp. 480-485.