Softprop: Softmax Neural Network Backpropagation Learning

Michael Rimer
Computer Science Department
Brigham Young University
Provo, UT 84602, USA
E-mail: mrimer@axon.cs.byu.edu

Tony Martinez
Computer Science Department
Brigham Young University
Provo, UT 84602, USA
E-mail: martinez@cs.byu.edu

Abstract — Multi-layer backpropagation, like many learning algorithms that can create complex decision surfaces, is prone to overfitting. Softprop is a novel learning approach presented here that is reminiscent of the softmax explore-exploit Q-learning search heuristic. It fits the problem while delaying settling into error minima to achieve better generalization and more robust learning. This is accomplished by blending standard SSE optimization with lazy training, a new objective function well suited to learning classification tasks, to form a more stable learning model. Over several machine learning data sets, softprop reduces classification error by 17.1 percent and the variance in results by 38.6 percent over standard SSE minimization.

I. Introduction

Multi-layer feed-forward neural networks trained through backpropagation have received substantial attention as robust learning models for classification tasks [15]. Much research has gone into improving their ability to generalize beyond the training data. Many factors play a role in their ability to learn, including network topology, learning algorithm, and the nature of the problem at hand. Overfitting the training data is often detrimental to generalization and can be caused through the use of an inappropriate objective function.

Lazy training [12,13] is a new approach to neural network learning motivated by the desire to increase generalization in classification tasks. Lazy training implements an objective function that seeks to directly minimize classification error while discouraging overfitting. Lazy training is founded upon a satisficing philosophy [9] where the traditional goal of optimizing network output precision is relaxed to that of merely selecting hypotheses that produce rational (correct) decisions. Lazy training has been shown to decrease overfitting and discourage weight saturation in complex learning tasks while improving generalization [13,14]. It has performed successfully on speech recognition tasks, a large OCR data set and several benchmark problems selected from the UCI Machine Learning Repository, reducing average generalization error over training of optimized standard backpropagation networks using 10-fold stratified cross-validation.

In this work a method for combining standard backpropagation learning and lazy training is presented that we call *softprop*. It is named after the softmax exploration policy in Q-learning [19], combining greedy exploitation and conservative exploration in an optimization search. This exploration policy tends to be effective in complex problem spaces that have many local minima. This technique is shown to achieve higher accuracy and more robust solutions than either standard backpropagation or lazy training alone.

A background discussion of traditional objective functions and the lazy training objective function is provided in Section II. The proposed softprop technique is presented in Section III. Experiments are detailed in Section IV. Results and analysis are shown in Section V. Conclusions and future work are presented in Section VI.

II. Motivation for Lazy Training

To generalize well, a learner must use a proper objective function. Many learning techniques incorporate an objective function minimizing *sum-squared-error* (SSE). The validity of using SSE as an objective function to minimize error relies on the assumption that sample outputs are offset by inherent gaussian noise, being normally distributed about a cluster mean. For function approximation of an arbitrary signal, this presumption often holds. However, this assumption is invalid for classification problems where the target vectors are class codings (i.e., arbitrary nominal or boolean values representing designated classes).

Error optimization using SSE as the measure has been shown [8] to be inconsistent with ultimate sample classification accuracy. That is, minimizing SSE is not necessarily correlated to achieving high recognition rates. In [8], a monotonic objective function, the *classification figure-of-merit* (CFM), is introduced for which minimizing error remains consistent with increasing classification accuracy. Networks that use the CFM as their criterion function in phoneme recognition are introduced in [8] and further considered in [5]. They are, however, also susceptible to overfitting.

The question of how to prevent overfitting is a subtle one. When a network has many free parameters local minima can often be avoided. On the other hand, networks with few free parameters tend to exhibit better generalization performance. Determining the appropriate size network remains an open problem [7].

The above objective functions provide mechanisms that do not directly reflect the ultimate goal of classification learning, i.e., to achieve high recognition rates on unseen data. Numerous experiments in the literature provide examples of networks that achieve little error on the training set but fail to achieve high accuracy on test data [2, 16]. This is due to a variety of reasons, such as *overfitting* the data or having an incomplete representation of the data distribution in the training set. There is an inherent tradeoff between fitting the (limited) data sample perfectly and generalizing accurately over the entire population.

Methods of addressing overfit include using a holdout set for model selection [18], cross-validation [2], node pruning [6, 7], and weight decay [20]. These techniques seek to compensate for the bias of standard backpropagation learning [11] in specific situations. For example, as overly large networks tend to overfit, node pruning seeks to improve accuracy by simplifying network topology. Forming network ensembles can also reduce problems in the inductive bias inherent to gradient descent. Ensemble techniques, such as *bagging* and *boosting* [10], or *wagging* [3], are more robust than single networks when the errors among the networks are not closely correlated.

There is evidence that the magnitude of the weights in a network plays a more important role to generalization than the number of nodes [4]. Optimizing SSE tends to a saturation of weights, often equated with overfitting. It follows that overfit might be reduced by keeping the weights smaller. Weight decay is a common technique to discourage weight saturation. Another simple method of reducing overfit is to provide a maximum error tolerance threshold, d_{max}, which is the smallest absolute output error to be backpropagated. In other words, for a given d_{max}, target value, t_k, and network output, o_k, no weight update occurs if the absolute error $| t_k - o_k | < d_{max}$. This threshold is arbitrarily chosen to indicate the point at which a sample has been sufficiently approximated. Using an error threshold, a network is permitted to converge with much smaller weights [17].

A. Lazy Training

Retaining smaller weights can be accomplished more naturally through lazy training. Lazy training only backpropagates an error signal on misclassified patterns. Previous work [12, 13] has shown how applying lazy training to classification problems can consistently improve generalization.

For each pattern considered by the network during the training process, only output nodes credited with classification errors backpropagate an error signal. As this forces a network to delay learning until explicit evidence is presented that its state is a detriment to classification accuracy, we have dubbed this technique *lazy training* (not to be confused with *lazy learning* approaches [1]). Often, an objective function is used in backpropagation training that tends to a saturation of the weights. That is, it tends to encourage larger weights in an attempt to output values approaching the limits of 0 and 1. Lazy training does not depend on idealized target outputs of 0 and 1. As such, it is biased toward simpler solutions, meaning that smaller weight magnitudes (even approaching zero) can provide a solution with high classification accuracy. This approach allows the model to approach a solution more conservatively and discourages overfit.

B. Lazy Training Heuristic

The lazy training error function is as follows. Let N be the number of network output nodes (distinct class labels). Let o_k be the output value of the k^{th} output node of the network ($0 \le o \le 1$, $1 \le k \le N$) for a given pattern. Let T designate the target output class for that pattern and c_k signify the class label of the k^{th} output node. For target output nodes, $c_k = T$, and for non-target output nodes, $c_k \ne T$. Non-target output nodes are called *competitors*. Let $o_{T\max}$ denote the highest-outputting target output node. Let $o_{\sim T\max}$ denote the value of the highest-outputting competitor. The error, ε_k, back-propagated from the k^{th} output node of the network is defined as

$$\varepsilon_k \equiv \begin{cases} o_{\sim T\max} - o_k & \text{if } c_k = T \text{ and } (o_{\sim T\max} \ge o_{T\max}) \\ o_{T\max} - o_k & \text{if } c_k \ne T \text{ and } (o_k \ge o_{T\max}) \\ 0 & \text{otherwise} \end{cases} \quad . \qquad (1)$$

Thus, the target output backpropagates an error signal only if there is some competitor with an equal or higher value than it, signaling a misclassification. Non-target outputs generate an error signal only if they have a value equal to or higher than $o_{T\max}$, indicating they are also responsible for the misclassification. The error value is set to the difference in value between the target and competitor nodes.

Lazy training of a network proceeds at a different pace than with standard SSE minimization. Weights are updated only through necessity. Hence, a pattern can be considered “learned” with any combination of output values, providing competitors output lower values than targets. Training only nodes that directly contribute to classification error allows the model to relax more gradually into a solution and avoid premature weight saturation.

The output nodes can in effect collaborate together to form correct decisions. When the target output node presents a sufficient solution value in a local area of the problem space (i.e. its value is higher than for non-target nodes), competitor outputs do not need to work at redundantly modeling the same local data (i.e., approximate a zero output value). Consequently, they are able to specialize and break complex problems up into smaller, simpler ones. Whereas a fixed error threshold causes training to stop when output values reach a pre-specified point (e.g. 0.1 and 0.9), lazy training implements a *dynamic error threshold*, halting training on a given pattern as soon as it is classified correctly. Keeping weights smaller allows for training with less overfit and greater generalization accuracy.

C. Adding an error margin to lazy answers

When lazy training, it is common for the highest outputting node in the network to output a value only slightly higher than the second-highest-firing node (see Figure 1). This is true for correctly classified samples (to the right of 0 in Figure 1), and also for incorrect ones (to the left of 0). This means that most training samples remain physically close to the decision surface throughout training. An error margin, μ, is introduced during the training process to serve as a confidence buffer between the outputs of target and competitor nodes. Using the sigmoid function, the error margin is bounded by [–1, 1]. For no error signal to be backpropagated from the target output, an error margin requires that $o_{\sim T\max} + \mu < o_{T\max}$. Conversely, for a competing node k with output o_k, the inequality $o_k + \mu < o_{T\max}$ must be satisfied for no error signal to be backpropagated from k.

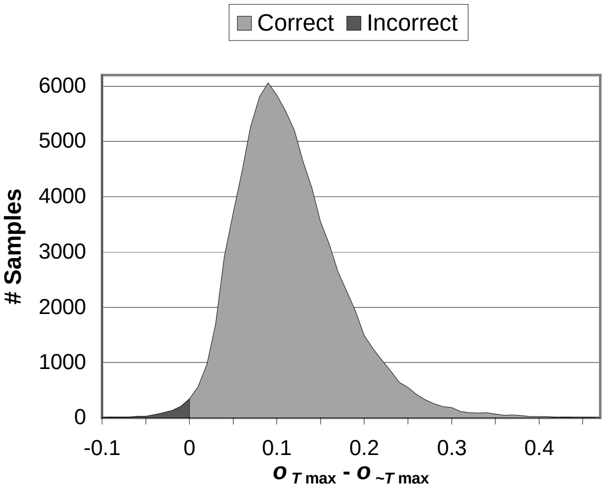

Fig. 1. Network output margin of error after lazy training.

Requiring an error margin is important since the goal of learning in this instance is not simply to learn the training environment well but to be able to generalize. This is especially important in the case of noisy problem data. During the training process, μ can be increased gradually and might even be negative to begin with, not expressly requiring correct classification at first. This gives the network time to configure its parameters in a more uninhibited fashion. Then μ is increased to an interval sufficient to account for the variance that appears in the test data, allowing for robust generalization.

At the extreme value of μ equal to 1, lazy training becomes standard SSE training, with output values of 1.0 and 0.0 required to satisfy the margin. Since a margin of 1 can never be obtained without infinite weights, an error signal is always backpropagated on every pattern.

III. Softprop Heuristic

The softprop heuristic performs a novel explore-exploit search of the solution space for multi-layer neural networks. Softprop exchanges the use of a single pure objective function with a mixture taking advantage of both lazy training and SSE minimization at appropriate times during the learning process. The heuristic is as follows:

> For each epoch, let the lazy training error margin $\mu = t/T$, where $t \in \{0, 1, 2, \ldots\}$ is the current epoch and T is the maximum number of epochs to train.

Softprop causes a smooth shift from lazy training to SSE minimization as the search progresses. The lazy exploration phase first steers the decision surface toward a general problem solution without saturating network weights prematurely. Then, as learning tends toward SSE exploitation, the distance of the decision boundary from proximate patterns is maximized. The practical aspect of this approach is analogous to simulated annealing, where a Boltzmann stochastic update is used with an update probability “temperature” that is gradually reduced to allow the network to gradually settle into an error minimum.

The complexity of softprop is equivalent to that of standard SSE optimization and lazy training and converges in comparatively as many epochs.

IV. Experiments

A. Data sets

Several well-known benchmark classification problems were selected from the UC Irvine Machine Learning Repository (UCI MLR). The problems were selected so as to have a wide variety of characteristics (size, number of features, complexity, etc.) in order to demonstrate the robustness of the learning algorithms. Results on each problem were averaged using 10-fold stratified cross-validation.

B. Training parameters

Experiments were performed comparing the SSE and lazy training objective functions against the proposed softprop heuristic. Feed-forward multi-layer perceptron networks with a single, fully-connected hidden layer were trained through on-line backpropagation. In all experiments, weights were initialized to uniform random values within the range [-0.3,0.3]. The learning rate was 0.1 and momentum was 0.5. Networks trained to optimize SSE used an error threshold (d_{max}) of 0.1.

Feature values (both nominal and continuous) were normalized between zero and one. Training patterns were presented to the network in a random order each epoch. The same initial random seed for network weight initialization and sample shuffling was used for all experiments on a given data set.

SSE and lazy training continued until the training set was successfully learned or until training classification error ceased to decrease for a substantial number of epochs. The softprop schedule was set for an equivalent number of epochs. A holdout set (between 10-20% of the data) was randomly selected from the training set each fold to perform model validation. The model selected for test evaluation was the network epoch with the best holdout accuracy.

Network architecture was optimized to maximize generalization for each problem and learning heuristic. Pattern classification was determined by *winner-take-all* (the class of the highest outputting node is chosen) on all models tested.

V. Results

Table 1 lists the results of a naïve Bayes classifier (taken from [21]), standard SSE backpropagation, lazy training, and softprop on the selected UCI MLR corpus. Each field lists first the average holdout set accuracy using 10-fold stratified cross validation. The second value is the variance of the classification accuracy over all ten runs. The best generalization and variance for each problem is bolded.

On average, an optimized backpropagation network minimizing SSE is superior to a naïve Bayes learner on the above classification problems. Lazy training obtains a significantly higher accuracy over SSE training. Interestingly, the SSE minimizing network achieves an SSE up to two orders of magnitude lower than that of the selected lazy trained network, a moot point because SSE is simply a means to an end, not the ultimate measure of optimality. However, this serves to illustrate that the SSE and lazy approaches each perform radically different searches of the problem space.

Softprop performed better than both lazy training and simple SSE backpropagation, reducing classification error by 17.1% and had the best overall accuracy. Softprop is particularly effective in learning noisy problems (e.g. *sonar*) where premature saturation of weights could trap the network in a local minimum.

Decreasing classification error is a worthy achievement, but of possibly even greater import is the fact that softprop has a significant overall reduction in the variance of classification error over the ten cross-validation folds. Lazy training shows a minor overall reduction in standard deviation of error over SSE backpropagation. Softprop provides a larger reduction of 38.6%. This supports the softprop approach as being more robust.

TABLE I

Results on UCI MLR data sets using 10-fold stratified cross-validation

Data set	**Bayes**	**SSE**	**Lazy**	**Softprop**
ann	**99.7** **0.1**	98.25 0.54	97.92 0.55	98.29 0.43
bcw	93.6 3.8	96.78 2.05	96.87 3.76	**97.07** **1.61**
ionosphere	85.5 4.9	88.03 6.12	**90.60** **4.80**	89.17 4.93
iris	94.7 6.9	93.33 7.30	**95.33** 4.27	**95.33** **3.06**
musk2	97.1 0.7	99.38 **0.21**	**99.44** 0.40	99.23 0.48
pima	72.2 6.9	**77.47** 3.75	76.69 5.22	76.69 **2.37**
sonar	73.1 11.3	77.40 10.77	81.73 14.08	**83.65** **8.67**
wine	94.4 5.9	94.94 8.04	96.63 4.58	**98.88** **2.29**
Average	88.79 5.06	90.70 4.85	91.93 4.74	**92.29** **2.98**

VI. Conclusions and Future Work

The softprop heuristic of gradually increasing the required margin of error between classifier outputs, reflecting a steady shift between classification error exploration and SSE exploitation, was shown to be superior to either optimization of SSE or classification error alone. Softprop reduces classification error over a corpus of machine learning data sets by 17.1% and variance in test accuracy by 38.6%.

While the parameters of the SSE backpropagation learner had been extensively optimized, due to time constraints little parameter tuning was done on the softprop heuristics. It is possible that by optimizing the learning parameters even more significant improvements could be shown. Providing specialized exploration policies for local areas of the parameter space by dynamically setting a particular μ for each pattern will be considered. In this way, local learning can proceed at different speeds depending on the local characteristics of the problem domain. As learning progresses, the values for the local μ can be learned and refined according to need. We will experiment with the feasibility of relaxing the restrictions of our search by allowing a negative-valued μ. This in essence provides a way to “tunnel” through difficult, inconsistent, or noisy portions of the problem space in order to escape local minima and might assist in achieving more optimal solutions.

REFERENCES

[1] David W. Aha, editor, *Lazy Learning*, Kluwer Academic Publishers, Dordrecht, May 1997.

[2] Andersen, Tim and Tony R. Martinez, "Cross Validation and MLP Architecture Selection", *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'99*, CD Paper #192, 1999.

[3] Andersen, Tim and Martinez, Tony, "Wagging: A learning approach which allows single layer perceptrons to outperform more complex learning algorithms", *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'99*, CD Paper #191, 1999.

[4] Bartlett, Peter L., "The Sample Complexity of Pattern Classification with Neural Networks: The Size of the Weights is More Important than the Size of the Network", *IEEE Trans. Inf. Theory*, 44(2), 1998, pp. 525-536.

[5] Barnard, Etienne, "Performance and Generalization of the Classification Figure of Merit Criterion Function", *IEEE Transactions on Neural Networks*, 2(2), March 1991, pp. 322-325.

[6] Castellano, G., A. M. Fanelli and M. Pelillo, "An empirical comparison of node pruning methods for layered feed-forward neural networks", *Proc. IJCNN'93-1993 Int. J. Conf. on Neural Networks*, Nagoya, Japan, 1993, pp. 321-326.

[7] Castellano, G., A. M. Fanelli, and M. Pelillo, "An iterative pruning algorithm for feed-forward neural networks", *IEEE Transactions on Neural Networks*, vol. 8 (3), 1997, pp. 519-531.

[8] Hampshire II, John B., "A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 1, No. 2, June 1990.

[9] Simon, Herbert, "Theories of decision-making in economics and behavioral science," *American Economic Review*, XLIX (1959), 253.

[10] Maclin, R and Opitz, D, "An empirical evaluation of bagging and boosting", *The Fourteenth National Conference on Artificial Intelligence*, 1997.

[11] Mitchell, Tom, *Machine Learning*. McGraw-Hill Companies, Inc., Boston, 1997.

[12] Rimer, M., Andersen, T. and Martinez, T.R., "Improving Backpropagation Ensembles through Lazy Training," Proceedings of the *IEEE International Joint Conference on Neural Networks IJCNN'01*, pp. 2007-2112, 2001.

[13] Rimer, Michael, "Lazy Training: Interactive Classification Learning," Masters Thesis, Brigham Young University, April 2002.

[14] Rimer, M. Martinez, T.R. and D. R. Wilson, "Improving Speech Recognition Learning through Lazy Training," to appear in Proceedings of the *IEEE International Joint Conference on Neural Networks IJCNN'02*.

[15] Rumelhart, David E., Hinton, Geoffrey E. and Williams, Ronald J., "Learning Internal Representations by Error Propagation", Institute for Cognitive Science, University of California, San Diego; La Jolla, CA, 1985.

[16] Schiffmann, W., Joost, M. and Werner, R., "Comparison of Optimized Backpropagation Algorithms", *Artificial Neural Networks*, European Symposium, Brussels, 1993.

[17] Schiffmann, W., Joost, M. and Werner, R., "Optimization of the Backpropagation Algorithm for Training Multilayer Perceptions", University of Koblenz: Institute of Physics, 1994.

[18] Wang, C., Venkatesh, S. S., and Judd, J. S., "Optimal stopping and effective machine complexity in learning", in Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, vol. 6, Morgan Kaufmann, San Francisco, 1994, pp. 303-310.

[19] Watkins, C., and Dayan, P. "Q-learning", *Machine Learning*, vol. 8, 1992, pp. 279-292.

[20] Werbos, P., "Backpropagation: Past and future", *Proceedings of the IEEE International Conference on Neural Networks*, IEEE Press, 1988, pp. 343-353.

[21] Zarndt, Frederick, "A Comprehensive Case Study: An Examination of Machine Learning and Connectionist Algorithms," Masters Thesis, Brigham Young University, 1995.