Classification-based Objective Functions

MICHAEL RIMER mrimer@axon.cs.byu.edu
TONY MARTINEZ martinez@cs.byu.edu
Computer Science Department, Brigham Young University, Provo, UT 84602, USA
Phone: (801) 422-6464 *Fax: (801) 422-0169*

Abstract. Backpropagation, similar to most learning algorithms that can form complex decision surfaces, is prone to overfitting. This work presents classification-based objective functions, an intuitive approach to training artificial neural networks on classification problems. Classification-based learning attempts to guide the network directly to correct pattern classification rather than using an implicit search of common error minimization heuristics, such as sum-squared-error (SSE) and cross-entropy (CE). CB1 is presented here as a novel objective function for learning classification problems. It seeks to directly minimize classification error by backpropagating error only on misclassified patterns from culprit output nodes. CB1 discourages weight saturation and overfitting and achieves higher accuracy on classification problems than optimizing SSE or CE. Experiments on a large OCR data set have shown CB1 to significantly increase generalization accuracy over SSE or CE optimization, from 97.86% and 98.10%, respectively, to 99.11%. Comparable results are achieved over several data sets from the UC Irvine Machine Learning Database Repository, with an average increase in accuracy from 90.7% and 91.3% using optimized SSE and CE networks, respectively, to 92.1% for CB1. Analysis indicates that CB1 performs a fundamentally different search of the feature space than optimizing SSE or CE and produces significantly different solutions.

Keywords: neural networks, backpropagation, classification, objective functions

"The discovery consists of seeing what everyone else has seen
and thinking what no one else has thought"

Albert Szent-Georgyi

1. Introduction

Artificial neural networks have received substantial attention as robust learning models for tasks including classification and function approximation (Rumelhart, Hinton, & Williams, 1985). Learning is no longer formulated simply as function approximation (Bianchini, Frasconi, Gori, & Maggini, 1998) and much research has gone into improving a model's ability to generalize beyond sampled data. Many factors play a role in a network's ability to learn, including network properties, the learning algorithm, and the nature of the problem being learned. Usually, overfitting the training data is detrimental to generalization. Developing a universal learning model for high-accuracy learning while avoiding perceptible overfitting over relevant (real world) problem domains remains elusive.

This work proposes the use of classification-based (CB) objective functions, an intuitive approach to backpropagation training on classification problems. Classification-based training provides a framework for improving generalization on complex real world classification problems, such as speech and character recognition.

This work presents CB1 as its main contribution, an example of CB objective functions for learning classification tasks. CB1 seeks to directly minimize classification error by backpropagating error only on misclassified patterns from output nodes that are responsible for the error. It updates the network parameters as little as possible in order to directly classify the training patterns. This technique discourages weight saturation and overfitting and is conducive to higher accuracy in classification problems than optimizing common objective functions, such as sum-squared-error (SSE) and cross-entropy (CE).

CB1 is shown to perform markedly better on a large OCR data set than an optimized backpropagation network learning with respect to SSE or CE, increasing classification accuracy from 97.86% and 98.10%, respectively, to 99.11%, a 58.4% decrease in error. Comparable increases in accuracy are achieved on several classification problems from the UC Irvine Machine Learning Repository, with an average increase in accuracy from 90.7% and 91.3% for optimized SSE and CE networks, respectively, to 92.1% for CB1 performing 10-fold stratified cross-validation. Analysis indicates that CB1 performs a fundamentally different search of the feature space than backpropagation optimizing SSE or CE and produces significantly different solutions.

1.1 Overview

A background discussion and comparison of common objective functions to CB1 is provided in Section 2. The CB1 heuristic is presented in Section 3. Experiments and analysis are given in

Section 4 and discussion in Section 5. Further discussion of learning issues with feed-forward backpropagation neural networks, overfitting, and how these relate to CB1 is presented in Section 6. The relevance of CB techniques to previous interactive training paradigms is discussed in Section 7. Conclusions and future work are presented in Section 8.

2. Classical Objective Functions

In multi-layer perceptron (MLP) neural network learning, network speed, complexity and size are important considerations. Over the last fifteen years, much effort has been put into developing optimized neural network learning models and techniques. Techniques, such as Quickprop (Fahlman, 1988) and RPROP (Riedmiller & Braun, 1993), seek to speed up learning by dynamically adjusting update parameters. Models that seek to generate network topologies that are more suited to learning a given problem are classified as adaptive learning algorithms (Anderson & Martinez, 1995; Anderson & Martinez, 1996; Fahlman, & Lebiere, 1990). Partially connected static architectures are also considered in (Chakraborty, Sawada, & Noguchi, 1997). These networks have fewer parameters and are therefore simpler and more efficient than fully connected networks yet are able to perform equally well.

However, as computing resources continue to increase the consideration of *generalization* stands out at the forefront. With sufficient capacity, a network is able to store all of the training patterns presented it, and can reproduce results exactly as if performing a table lookup. After a certain point in backpropagation learning, however, reducing training set error often accompanies an increase in test set error, illustrating the degradation in generalization that accompanies overfit as training continues.

There has been a tendency to base "better" results of novel training variants on measurements which are not conclusive of improved generalization. This is especially prevalent in earlier research (see (Tollenaere, 1990) and (Fahlman, 1988) for a good discussion of this), but still occurs today. Results in the literature on speed enhancements, for instance, often show how fast a new algorithm can converge to arbitrary accuracy on a training set. In essence, this demonstrates how easily new training algorithms can *overfit*, but says nothing about their ability to generalize.

Since gradient descent procedures, such as backpropagation, do not allow direct minimization of the number of misclassified patterns (Duda, Hart, & Stork, 1999) an error or objective function must be derived that will result in increased classification accuracy as objective error is minimized. Network output values must have a corresponding error measure derived by their deviance from target output values. Quantifying the output error provides a way for iteratively updating the network weights in order to minimize that error and thereby achieve more accurate output. However, error functions are not always monotonic with reduction in classification error, which is the real goal of the learner.

Classification of N classes is often viewed as a regression problem with an N-valued response, with a value of 1 in the nth position if the observation falls in class n and 0 otherwise (LeBlanc & Tibshirani, 1993). The values of *zero* and *one* can be considered *idealized* or *hard* target values. However, in practice there is no reason why class targets must take on these values.

To generalize well, a network must be trained using a proper objective function. Backpropagation training often uses an objective function that encourages making weights larger in an attempt to output a value approaching hard targets of 0 or 1 (±1 for the htan function). Using hard targets is a naïve way of training and creates several practical problems. Different portions of the data are learned at different times during training, and using hard targets not only leads to *weight saturation*, making it harder and slower to learn patterns that have yet to be learned, but also forces the learner to overfit on patterns that have already been learned. Using hard targets of 0.1 and 0.9 presents a less severe solution but still suffers from overfit.

Methods for overcoming problems resulting from the objective function include forming network ensembles. Ensemble techniques, such as *bagging* and *boosting* (Maclin & Opitz, 1997), or *wagging* (Andersen & Martinez, 1999), are more robust than single networks when the errors among the networks are not positively correlated (see Section 7).

Rankprop (Caruana, Baluja, & Mitchell, 1996) provides an alternative method to training with hard target values and empirically shows that it improves generalization. Rankprop records the output of the learner for each training pattern. It then sorts the patterns in the training set based on class, then according to output values. Thus, a rank of the patterns consistent with the current model is developed and used to define the target values on the next epoch. The idea behind Rankprop is that in the case of complex nonlinear solutions a simpler, *less nonlinear* function is provided to learn instead. The resulting simpler model often generalizes better. CB1 also provides a simpler function for the network to learn that leads to better generalization.

2.1 Common objective functions

Sum-squared error (SSE) or *mean-squared error* (MSE), a common statistical measure of optimality, is a natural choice for an objective function, being differentiable. The validity of using SSE as an objective function to minimize error relies on the assumption that pattern outputs are offset by inherent gaussian noise, being normally distributed about a cluster mean. For learning 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). This suggests that other metrics are more suited to classification problems.

In (LeCun, Denker, & Solla, 1990), a study of the *digits* problem revealed that heuristically reducing the number of network parameters by a factor of two increased training set MSE by a factor of ten, while generalization MSE increased by only 50%, and test set classification error actually decreased. This suggests that MSE is not the most reliable objective function for this or similar tasks. This also implies that comparison studies showing “improvements” through a reduction of SSE/MSE on classification tasks are not significant unless classification accuracy increases likewise.

Cross-entropy (CE) assumes *idealized* class outputs (i.e., target values of zero or one for a sigmoid activation) rather than noisy outputs as does SSE (Mitchell, 1997) and is therefore more appropriate to classification problems. However, error values using SSE and cross-entropy have been shown (Hampshire II, 1990) to be inconsistent with ultimate pattern classification accuracy. That is, minimizing CE as well as SSE is not necessarily correlated to high recognition rates.

The classification figure-of-merit (CFM) objective function was introduced in (Hampshire II, 1990) for learning classification problems. It provides a closer estimation of true classification accuracy, as minimizing error is monotonic with increasing classification accuracy. Networks that use the CFM as their criterion function are introduced in (Hampshire II, 1990) and further considered in (Barnard, 1991).

However, CFM does not determine localization in learning (Jacobs, Jordan, Nowlan, & Hinton, 1991), i.e., every network is trained on all patterns. It produces strong (cooperative) coupling between experts, but there could be redundant experts for each pattern. (Jacobs, Jordan, Nowlan, & Hinton, 1991) introduces competitive experts. A gating network makes a stochastic decision about which expert to select based on the input. This kind of system tends to devote a single expert to each training case.

The task of CFM is to separate network output values by as large a range as possible. Like SSE and CE, this tactic encourages weight saturation, which is often indicative of overfitting and detrimental to accuracy (Bartlett, 1998).

Information gain is especially useful with iterative network growth in mind (Andersen & Martinez, 2001b). Since it makes decisions that have the most affect on accuracy, it has the potential to avoid local minima early on in the training process. Rather than each learner trying to immediately fit local regions of a function (curve) to arbitrary accuracy, as one learner fits a small region sufficiently well, others stop trying to learn that part and can direct their attention to other areas still in need of learning.

A problem with information gain is that it does not clearly suggest how training is to proceed. Training is carried out either on repeated runs of random weight perturbations or by normalizing the learning rate for each class based on the prospective information to be gained from correcting an error on an incorrectly classified pattern of that class (Andersen & Martinez, 2001b). Therefore, usually only very simple models are trained to maximize information gain, such as perceptrons.

2.2 Classification-based objective functions

It must be stressed that minimizing an objective function is not the goal of classification learning. Rather, it must be viewed as the mechanism that guides the network in learning the concept. The above objective functions provide mechanisms that do not directly reflect the true goal of classification learning, which is to achieve high recognition rates on unseen data. A designer implementing a network to learn the task of face or voice recognition by minimizing SSE is not really interested in what value the SSE reaches, but how accurate the recognizer is. Additionally, inappropriate objective functions can be deceiving in portraying how well the network has learned the problem (e.g., LeCun's *digits* study mentioned above). This being the case, the objective function chosen for learning a given task should approximate the true goal of the learner as closely as possible.

CB1's general philosophy is similar to CFM in that it also attempts to increase the range between output activations. However, CB1 differs from CFM in that it widens the range between outputs only when there is classification error (that translates in actuality to a *narrowing* of the gap between outputs, as they are transposed with respect to their classical target values). When a classification error is made, error is backpropagated only from those outputs that are credited with producing the error. This approach allows the network to relax more conservatively into a solution and discourages weight saturation and overfitting.

3. CB1: A Classification-based Error Heuristic

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 (see Andersen & Martinez, 1995; Schiffmann, Joost, & Werner, 1993). There is an inherent tradeoff between fitting the (limited) data sample perfectly and generalizing accurately on the entire population (see Section 6.4).

There are several possible ways to process the network's output vector in calculating an error signal for backpropagation to fit the data properly. For instance, the difference between pre-defined target values and network activations over all the training patterns can be combined into a single batch update to minimize sum-squared error. Alternately, error can be backpropagated following each pattern presentation for on-line learning.

A simple variant to on-line training involves modifying the objective function by providing a maximum error tolerance threshold, d_{max}, which is the smallest absolute output error to be backpropagated. In other words, given $d_{max} > 0$, a target value, t_j, and network output, o_j, no network update occurs if the absolute error $| t_j - o_j | < d_{max}$. This threshold is arbitrarily chosen to represent a point at which a pattern has been sufficiently approximated. With an error threshold, the network is permitted to converge with smaller weights (Schiffmann, Joost, & Werner, 1993). More dynamic approaches, such as Rankprop (Caruana, 1995), avoid the use of pre-defined "hard" targets, setting ranked "soft" target values for the training patterns each epoch.

CB1, introduced here, considers the entire output vector of the network to determine the error of each output node. For each pattern considered, CB1 backpropagates error through the network only on misclassified patterns. As this technique forces networks to learn only when explicit evidence is presented that their state is a detriment to classification accuracy, we have dubbed the approach classification-based training.

CB1 is similar to Rankprop in that it avoids the use of hard target values. However, rather than providing soft targets, it avoids the use of predetermined target values all together. The objective of CB training is *not* to minimize the error between target and output values, but rather to produce output values that can be accurately translated to correct classifications. When target values are not required to guide training, the network is able to arrive at a solution more simply. With CB training, smaller weights, even approaching zero, can provide an acceptable solution for classification tasks. This avoids many of the problems accompanying weight saturation and overfit.

Network weights are updated during CB training exclusively to minimize classification error. When the network misclassifies a pattern, credit for the error is assigned to two sources. The first is the set of output nodes with higher output values than the target class node (resulting in the system outputting the wrong class value). The second is the target output node itself, which output a value too low to produce the correct classification. This approach is formalized as follows.

3.1 CB1 error function

Let N be the number of output nodes in a network. Let o designate the activation value of a node ($0 \le o \le 1$ for sigmoid). Let o_k be the activation value of the k^{th} output node in the network ($1 \le k \le N$). Let T designate the target class for the current 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*. Often, class labels are indicated in training by setting the target value of one output node high and setting the rest low. This restriction is not made here, as it is possible for more than one output node to act as a target node for a class label in the general case. However, for the remaining discussion standard 1-of-N target designations are assumed.

Let $o_{T\max}$ denote the value of the highest-outputting target output node, or formally

$$o_{T\max} \equiv \max \{ o_k : c_k = T \}.$$

Let $o_{\sim T\max}$ denote the value of the competitor outputting the highest o,

$$o_{\sim T\max} \equiv \max \{ o_k : c_k \ne T \}.$$

The error, ε, back-propagated from the k^{th} output node is then defined as

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

The error (1) represented in closed form is

$$\varepsilon_k \equiv (o_{\sim T\max} - o_k)\mathrm{I}(c_k = T \text{ and } (o_{\sim T\max} \ge o_{T\max})) + (o_{T\max} - o_k)\mathrm{I}(c_k \ne T \text{ and } (o_k \ge o_{T\max}))$$

where I is the indicator or characteristic function. Thus, a target output generates an error signal only if there is some competitor with an equal or higher value than $o_{T\max}$, signaling a potential misclassification. Non-target outputs likewise generate an error signal only if they have an output equal or higher than $o_{T\max}$, indicating they are responsible for the misclassification. The intuitive rationale behind this is that if the error is continually reduced on misclassified patterns, they will eventually be classified correctly.

The error delta used for backpropagation is

$$\delta_k = \varepsilon_k f'(o_k)$$

where $f'(o_k)$ is the standard error gradient, which is

$$f'(o_k) = o_k(1 - o_k)$$

for a sigmoid squashing function, and can be removed on output nodes when using cross-entropy (Joost & Schiffmann, 1998).

To illustrate how CB training works, consider a three-class problem. For a particular pattern, assume that the third class is the target. Traditionally, this translates into a target vector of (0, 0, 1). Assume that on this pattern, a 3-output network outputs (0.1, 0.2, 0.4). While the third output (the target) has significant squared error (0.36), the first two output values (the competitors) are sufficiently low, enough so that it is possible to extract the correct classification (the third class is chosen since its value is highest). Since the pattern is classified correctly, the network parameters remain unchanged.

Only if one of the competitors output higher than the target would a non-zero error signal be backpropagated from any of the output nodes. In the case that the network outputs (0.1, 0.4, 0.3), both the second and third output nodes would backpropagate error: the second since it outputs higher than the target node, and the third, since a competitor output a higher value than it. The error signal is set at the minimum amount possible to produce a correct classification.

3.2 Advantages of CB training

Repeatedly forcing output values closer to 0 or 1 in cases where pattern classification is already correct usually results in weight saturation and possibly overfit. This needlessly increases network variance (sensitivity to the training data), increasing classification error on test data. Training without idealized or predetermined target outputs allow a pattern to be potentially "learned" with any target node output, providing competitors output lower values. This insight is the driving philosophy behind CB training, which avoids this practice.

CB training of a network proceeds at a different pace than optimizing SSE or CE as the objective function. Weights are updated only through necessity. Backpropagating a non-zero error signal from *only* the outputs that directly contribute to classification error results in significantly fewer weight updates overall (observe that this number is proportional to the classification accuracy) and allows the model to relax more gradually into a solution. CB training learns only as much as required to remove misclassifications and thereby discourages overfitting. This approach is reminiscent of training with an error threshold; however, whereas a fixed error threshold causes training to stop at a pre-specified point, meaning weights must increase to a magnitude sufficient to achieve this threshold, CB training dynamically halts learning at the *first possible point* that correctly classifies a training pattern. This can be considered an implementation of a *dynamic error threshold* that is unique to each training pattern and network state.

3.3 Increasing the margin with CB training

Overfit is minimized in CB training in another regard because outliers (noisy patterns) have minimal detrimental impact to the decision surface's accuracy. This is because the target output is only required to output a value negligibly higher than the highest competitor before the training process stops updating the network parameters. This translates to halting the movement of the decision surface right next to the pattern (see Figure 1b). This is in contrast to classical

SSE training, where hard target values of 0 and 1 require pushing the decision surface as far away from all points as possible, even noisy outliers (see Figure 1a). Hence, a test pattern (represented by the question mark) falling immediately next to a noisy outlier belonging to a competing class has a better chance of being correctly classified. In other words, network variance is substantially reduced.

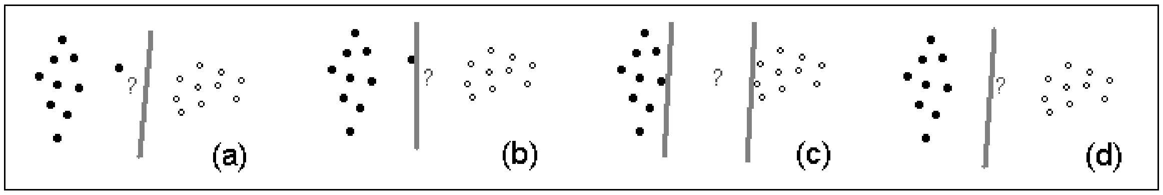


Figure 1. SSE decision surfaces (a,d) and CB decision surfaces (b,c,d).

When CB training, it is common for the highest outputting node in the network, which we will call $o_{\max}$, to output a value only slightly higher than the second-highest-outputting node (see Figure 4). This is true for correctly classified patterns (those above 0 in Figure 2), and also for misclassified ones (those below 0). This means that most training patterns remain physically close to the decision surface throughout training. In the absence of outliers, then, one would expect the heuristic to arrive at a decision surface similar to those portrayed in Figure 3c. According to the application this might not be desirable.

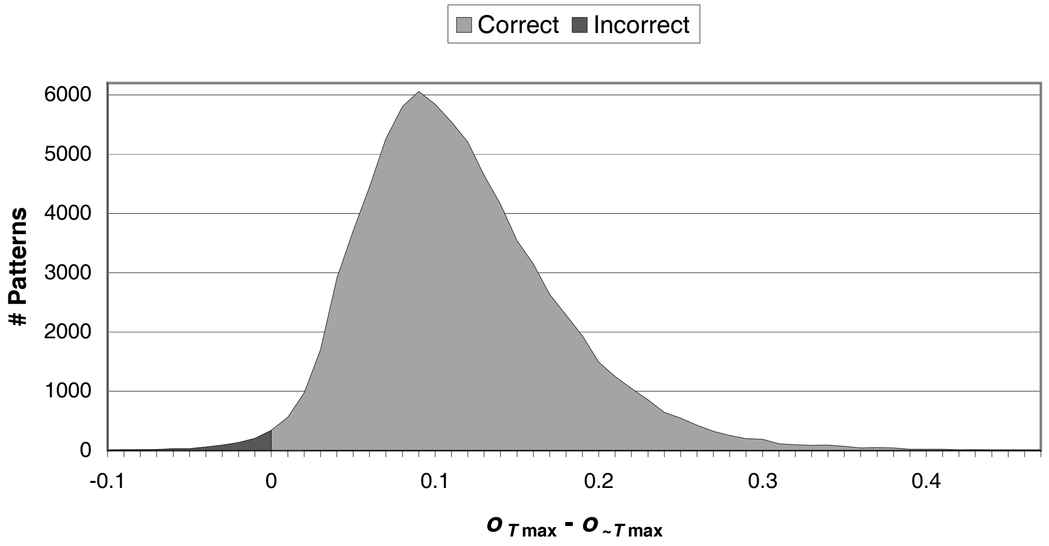


Figure 2. Network output error margin after CB training on *OCR* data set.

An error margin, μ, can be introduced during training that serves as a confidence buffer between the outputs of target and competitor nodes. The value for μ can range from –1 to +1 under the sigmoid function. 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 $o_k < o_{T\max} - \mu$ must be satisfied for no error signal to be backpropagated from k. This augmentation to (1) is presented as

$$\varepsilon_k \equiv \begin{cases} \min(o_{\sim T\max} + \mu - o_k, 1) & \text{if } c_k = T \text{ and } (o_{\sim T\max} + \mu \geq o_{T\max}) \\ \max(o_{T\max} - \mu - o_k, -1) & \text{if } c_k \neq T \text{ and } (o_k \geq o_{T\max} - \mu) \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

where min(·,1) and max(·,-1) enforce the [-1,1] error range of the sigmoid function. In this way, CB1 with a small μ (e.g. 0.1) approximates the SSE solution and the margin is maximized even in the absence of outliers (see Figure 1d).

During the training process, the value of μ can be altered 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 an even more uninhibited fashion. Then μ is increased to an interval sufficient to account for the variance that appears in the domain data, allowing for robust generalization. The value of μ can also be decreased, and remain negative as training is concluded to account for noisy outliers. A preliminary analysis of updating μ during training has shown promise (Rimer & Martinez, 2004).

Including a margin also decreases the amount of "classification oscillation" that occurs as outputs react to one another. When $\mu = 0$, patterns remain close to the decision surface during training. As training proceeds and the decision surface shifts around, patterns frequently slide back to the wrong side as the decision surface. Introducing a small, positive μ requires patterns to be situated further away from the decision surface and reduces the incidence of renewed misclassification, leading to quicker convergence. Observe that at the extreme value of $\mu = 1$, CB1 reverts to standard SSE training, with target values of 1.0 and 0.0 required for all positive and negative classes, respectively.

4. Experiments and Analysis

Neural networks were trained through backpropagation optimizing SSE and CE, and through CB1 to explore empirical advantages of CB training techniques. These models include:

- Single-output networks on two-class problems (positive patterns are assigned a target value of 1.0, negative patterns are assigned a target value of 0.0)
- Multi-output networks (one output per class)
- Independent single-output networks on multi-class problems (one per class)

Experiments were conducted over a variety of data sets with varying characteristics, differing by:

- Size of data set (150 instances to half a million)
- Number of features (two to hundreds)
- Number of labeled data classes (two to forty-seven)
- Complexity of data distribution (nearly linearly separable to highly complex)

Real world problems were drawn from the UC Irvine Machine Learning Database Repository (UCI MLDR) (Blake & Merz, 1998) and from a large database of machine printed characters gathered for OCR. This provides a vantage point from which to evaluate the robustness of the CB1 heuristic.

In empirical comparisons among different learning methods, appropriate training parameters were determined to optimize each model. For further conceptual analysis and illustration of the behavior of these systems, results of experiments using a range of parameters are provided.

4.1 Data sets

The performance of SSE versus CB training has been evaluated on an OCR data corpus (*OCR*) consisting of over 495,000 alphanumeric character patterns, partitioned into roughly 415,000 training patterns and 80,000 test patterns. This work was first presented in (Rimer, Anderson, & Martinez, 2001a).

Two network topologies were evaluated for learning *OCR*, a single *N*-output network and *N* single-output networks.

Additionally, eight well-known classification problems were selected from the UCI MLDR. Descriptions of the selected data sets are listed as follows:

ann – 7200 instances with 15 binary and 6 continuous attributes in 3 classes. The task is to determine whether a patient referred to the clinic is hypothyroid.

bcw – 699 instances with 9 linear attributes in 2 classes. The task is to detect the presence of malignant versus benign breast cancer.

ionosphere – 351 instances with 34 numeric attributes in 2 classes. This data set classifies the presence of free electrons in the ionosphere.

iris – 150 instances with 4 numeric attributes in 3 classes. This classic machine learning data set classifies the species of various iris plants based on physical measurements.

musk2 – 6598 instances with 166 continuous attributes in 2 classes. The task is to predict whether new molecules will be musks or non-musks.

pima – 768 instances with 8 numeric attributes in 2 classes. The predictive class in this data set is whether or not the tested individual has diabetes.

sonar – 208 instances with 60 continuous attributes in 2 classes. The task is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.

wine – 178 instances with 13 continuous attributes in 3 classes. The attributes give various parts of the chemical composition of the wine and the task is to determine the wine’s origin.

A single network with one output per class was used to learn these problems. Results on UCI MLDR problems were averaged using 10-fold stratified cross-validation.

4.2 Training parameters

Experiments were performed comparing the SSE, CE, and CB1 objective functions. Fully connected feed-forward networks with a single hidden layer trained through standard on-line backpropagation were used. In all experiments, weights were initialized to uniform random values in the range [-0.1,0.1]. Networks trained to optimize SSE and CE used an error tolerance threshold (d_{max}, described in Section 2) of 0.1.

Feature values (both nominal and continuous) were normalized between zero and one. Training patterns were randomly shuffled before each epoch. For each simulation, a random seed for network weight initialization and pattern shuffling was used across all networks tested.

Network learning parameters on *OCR* have been extensively optimized over the course of two years. For experiments presented here, we constrained each network to a single hidden layer comprised of 32 hidden nodes. The learning rate was 0.2 and momentum was 0.5. Training was halted after 500 epochs. Although we only present results for a single set of parameter values here, it is noted that results testing the above objective functions were typical and comparable over a wide range of learning parameters, network sizes and topologies tested.

On UCI MLDR data sets, network size was optimized to maximize generalization for each problem and error function. Optimized numbers of hidden nodes used for learning UCI MLDR data sets are listed in Table 1. Learning rate was 0.1 and momentum was 0.5 for all UCI MLDR problems. Training continued until the training set was successfully learned or until training classification error ceased to decrease for 500 epochs. The network model then selected for testing was the one with the best training set classification accuracy. An alternate set of experiments was run using a holdout set to perform model selection with comparable results. It has been omitted here for brevity. Pattern classification was determined by *winner-take-all* (the class of the highest outputting node is chosen) on all models tested.

Table 1. Network architectures on MLDR problems.
The number of input, hidden, and output nodes per network is shown.

Data set	**SSE Network**	**CE Network**	**CB1 Network**
ann	21-30-2	21-30-2	21-30-2
bcw	9-15-2	9-25-2	9-10-2
ionosphere	34-7-2	34-9-2	34-9-2
iris	4-1-3	4-1-3	4-1-3
musk2	166-5-2	166-5-2	166-5-2
pima	8-8-2	8-8-2	8-16-2
sonar	60-15-2	60-5-2	60-15-2
wine	13-16-3	13-8-3	13-16-3

4.3 Results

4.3.1 OCR data set

Table 2 displays the results of standard SSE and CE backpropagation versus CB1 on *OCR*. *Train %* and *Test %* are the final training and test set accuracy in percent. *Train MSE* and *Test MSE* are the mean squared errors for the training and test sets on the epoch for which the highest test set accuracy is achieved.

Table 2. Results on *OCR* data set.

Method	Train %	Train MSE	Test %	Test MSE
SSE (Multiple networks)	99.28	**.0047**	97.86	**.0092**
SSE (Single network)	98.40	.0225	98.38	.0335
CE (Multiple networks)	99.37	.0094	98.10	.0110
CE (Single network)	98.62	.0153	98.58	.0300
CB1 (Single network) $\mu = 0$	99.15	.1594	98.96	.1800
CB1 (Multiple networks) $\mu = 0.05$	**99.61**	.1830	**99.11**	.2410

The results on *OCR* show that multi-task learning (MTL), or using a single network with multiple output nodes, performs better than using a separate network to learn each class with SSE and CE objective functions. Even though training accuracy is lower on the SSE and CE multi-output networks than multiple networks, generalization is improved. Observe that training set accuracy is largely preserved on the test set when using a single multiple output network using any of the tested error functions. This occurs since little overfitting can occur in this size network when attempting to learn all classes simultaneously. A greater relative decrease in generalization was observed using networks with more hidden nodes. When using a separate network for each class, each network has much greater potential to overfit since there are many more network parameters. This behavior is exhibited to lesser degree with CB training.

Optimizing CE on this difficult classification problem trains and generalizes better than SSE, and CB1 performs significantly better than both. Network models generated through CB training have the capability of improving generalization even more. These tests also show that, although the final SSE for CB1 is 10-20 times greater than for SSE and CE optimization, the amount of overfitting is sharply reduced and generalization is improved. Since the networks learn together their errors are less correlated and the solution transfers well to unseen data.

Generalization error with the best CB1 architecture is 45.1% less than the best architecture trained with SSE and 37.3% less that the best architecture trained with CE. Considering only multiple-output networks, error drops from 1.62% for SSE to 1.42% for CE, and to 1.04% for CB1, error reductions of 35.8% and 26.8%, respectively. Considering only the multiple-network models, error drops from 2.14% with SSE to 1.90% with CE, and to 0.89% with CB1, error reductions of 58.4% and 53.2%, respectively.

4.3.2 UCI MLDR data sets

Table 3 lists the results of a naïve Bayes classifier (taken from (Zarndt, 1995)), standard SSE and CE backpropagation, and SSE and CE updates with CB1 on eight UCI MLDR classification problems. Results were gathered using 10-fold stratified cross validation. The first value in each cell is the average classification accuracy of the selected model. The second value is the standard deviation over all runs. The best generalization for each problem is bolded and the second best value is italicized.

Table 3. Results on selected data sets from UCI MLDR using 10-fold stratified cross-validation. Best values are shown in bold and second best in italics.

Data set	**Bayes**	**SSE**	**CE**	**CB SSE**	**CB CE**
ann	**99.7** **0.1**	98.25 0.54	98.33 0.53	97.62 0.47	*98.76* *0.51*
bcw	93.6 3.8	96.96 2.01	97.06 **1.81**	*97.22* 2.01	**97.36** **1.81**
ionosphere	85.5 4.9	89.00 4.72	*90.80* 4.64	90.60 **3.75**	**90.88** *3.87*
iris	94.7 6.9	93.83 5.68	94.37 5.87	**95.47** *5.31*	*95.37* **5.25**
musk2	97.1 0.7	99.06 0.37	98.56 0.62	*99.15* *0.36*	**99.27** **0.29**
pima	72.2 6.9	76.26 *4.24*	76.11 4.36	*76.69* **3.43**	**76.82** 6.46
sonar	73.1 11.3	76.06 9.37	78.87 9.03	*80.77* *9.02*	**81.92** **8.60**
wine	94.4 5.9	96.29 4.45	96.74 4.13	**98.31** **3.49**	*97.19* *3.47*
Average	88.79 5.06	90.69 3.92	91.35 3.87	*91.97* **3.48**	**92.20** *3.79*

The average decrease in classification error is from 9.31% for SSE training to 8.03% for CB training, a 13.7% decrease in error. An overall decrease in standard deviation also indicates that CB training is more robust to initial parameter values and pattern variance then SSE and CE optimization. This reflects the expectation that weight saturation and overfit is reduced and generalization is improved by CB training.

5. Discussion

Standard backpropagation and other gradient descent learning techniques do not consider or attempt to maximize the number of correctly classified training patterns (see (Duda, Hart, & Stork, 1999)). CB training incorporates a more direct minimization of misclassified patterns in gradient descent procedures by reducing error on only misclassified patterns.

CB1 does not modify weights to provide a monotonic decrease in a global error signal based on ideal target values using metrics as SSE or CE. In fact, during training SSE (MSE) often remains roughly constant as accuracy is improved (see Figure 3). Change in CE displays the same behavior as MSE but is omitted from this graph and the following discussion for clarity. This is in contrast to the steady drop in SSE illustrative of standard SSE optimization.

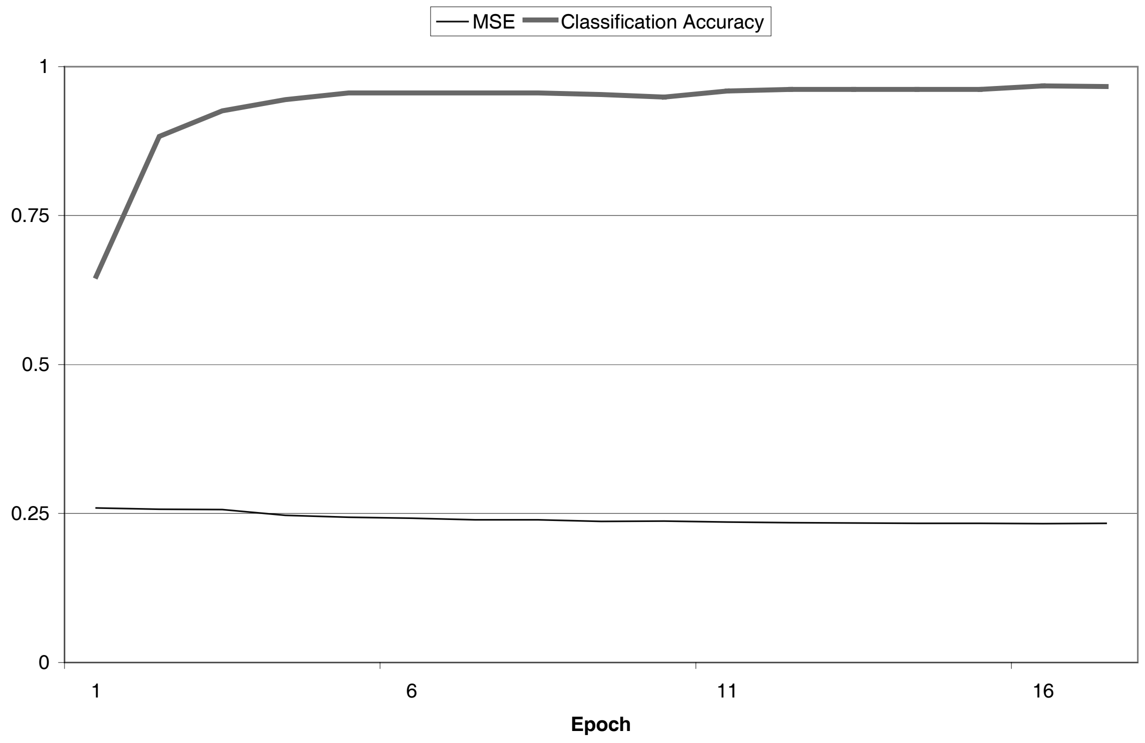


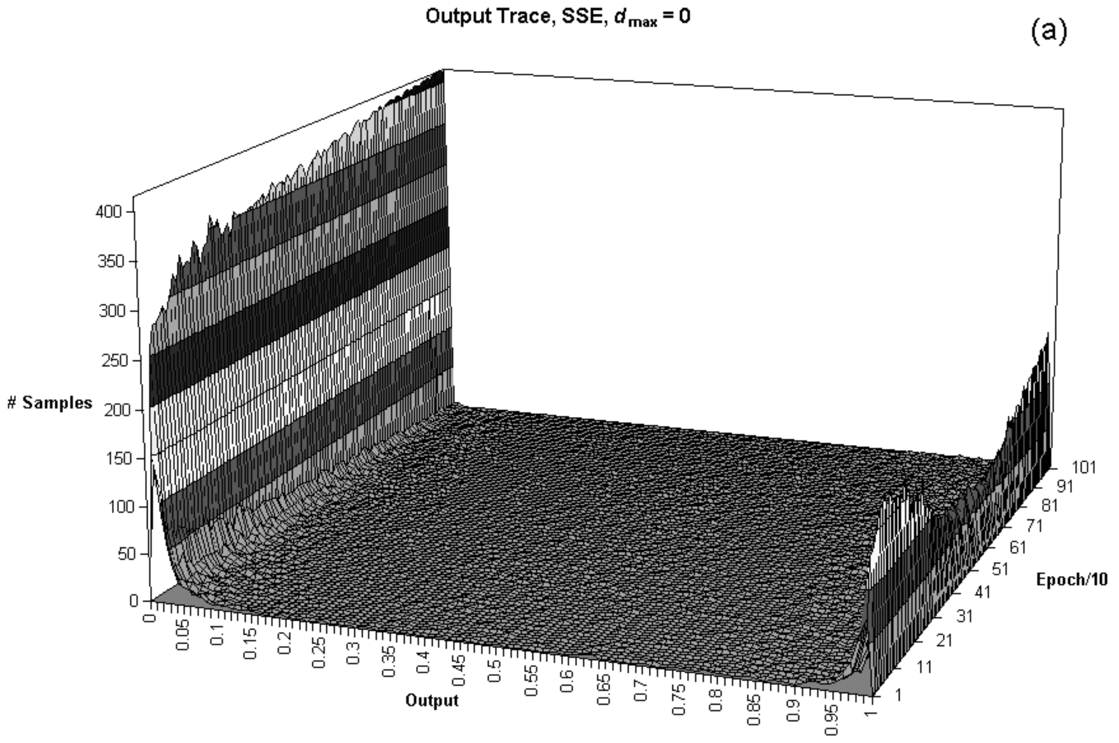
Figure 3. Classification accuracy and MSE during CB training.

SSE, rather than converging to zero, instead remains rather large. A MSE of 0.25 is equivalent to a mean error of 0.5, which illustrates that many output activations are about 0.5. This indicates that the weights for these outputs are close to zero. A large MSE is incurred by pattern outputs being very far away from their idealized target values. This suggests that CB training performs a fundamentally different search in feature space than standard SSE/CE optimization. It descends towards different minima and converges to a feature location physically distant from SSE/CE solutions. This also indicates that high-accuracy solutions exist where SSE are CE are about as high as when training starts on a network initialized to small random weights.

Figures 4 and 5 give insight into the behavior of the network during the learning process using four objective heuristics. The surface plot shows a histogram of the values output by the network output nodes on the training patterns every tenth training epoch. Figure 4a and 4b show learning minimizing SSE, and Figure 5a and 5b show behavior during CB training. The results shown here are only for the *bcw* data set, but such behavior is generally representative of all data sets tested.

Note that SSE training forces the network to output values approaching 0 and 1, even from the very first trace (the tenth epoch). Using a d_{max} of 0.1 reduces this tendency somewhat. Observe the flattened peaks for positive patterns in Figure 4b that do not exist in 4a.

CB training produces a starkly different behavior. In Figure 5a, it can be observed that all patterns output around 0.5 during the entire training process. In Figure 5b, incorporating a confidence margin of $\mu = 0.1$ widens the spread of output values, even causing the outputs of the two classes to visibly split apart as training progresses.



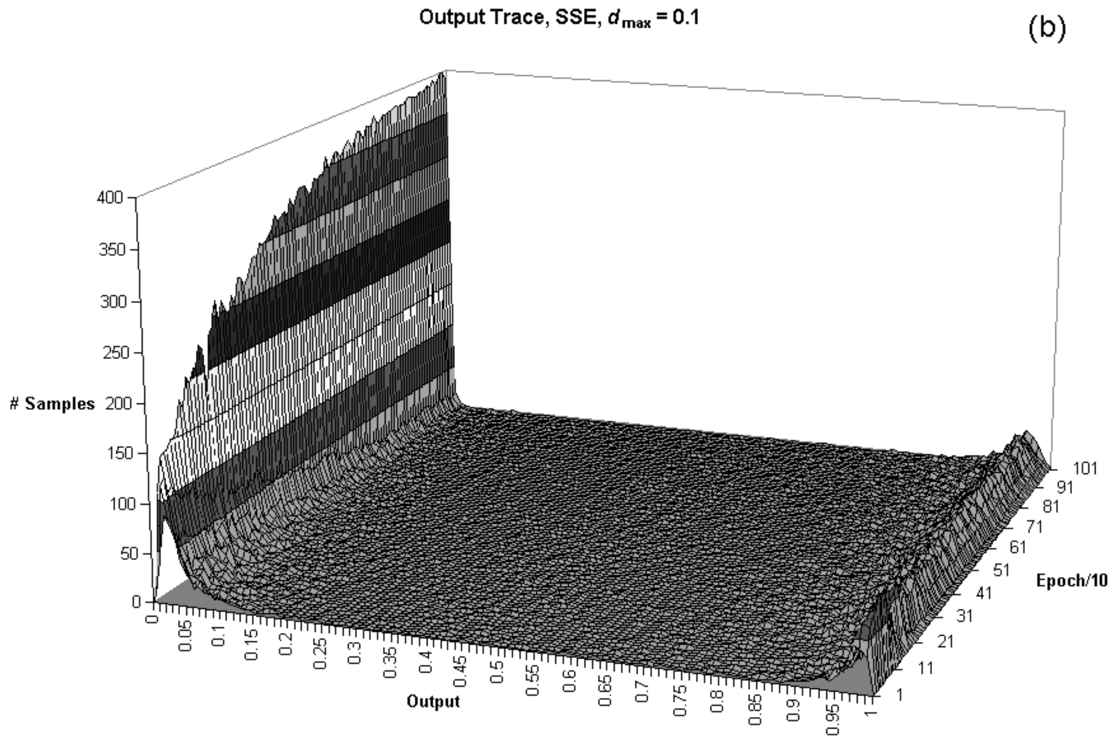
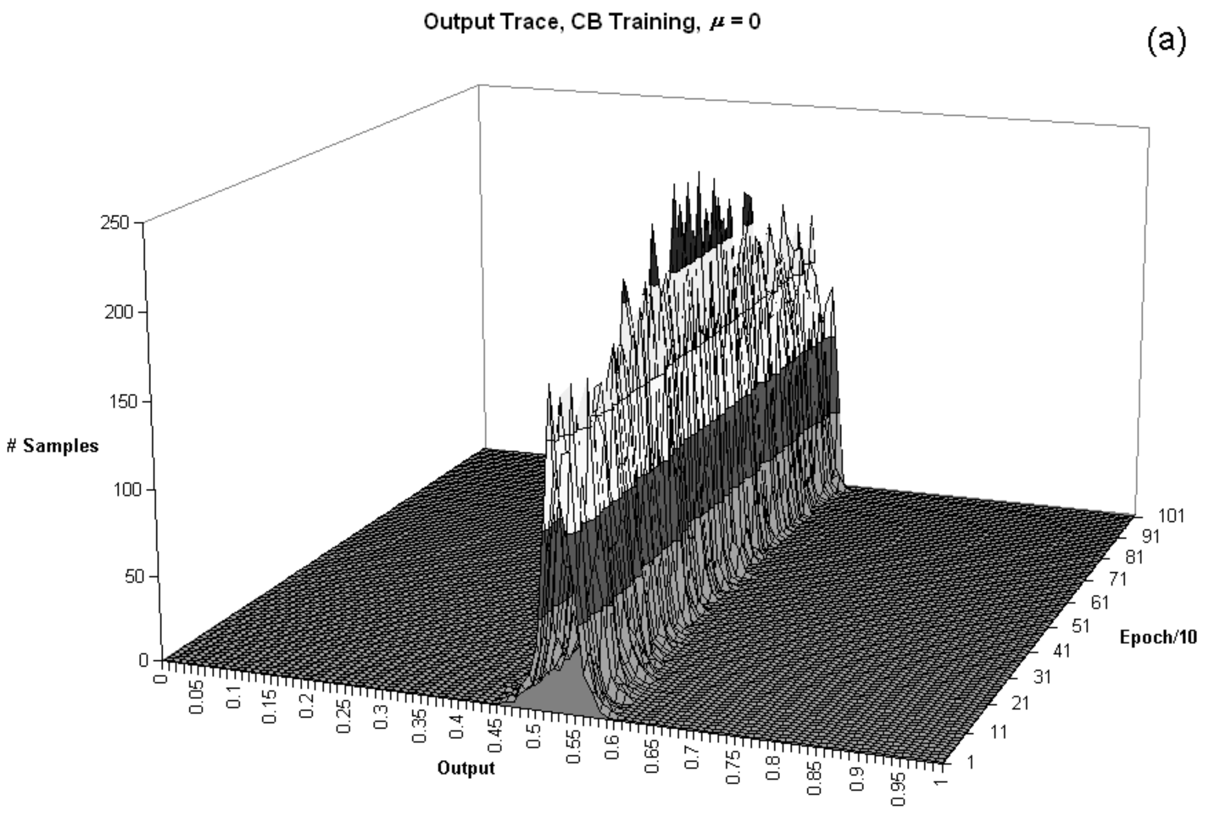


Figure 4. Network output trace during SSE minimization on *bcw*.



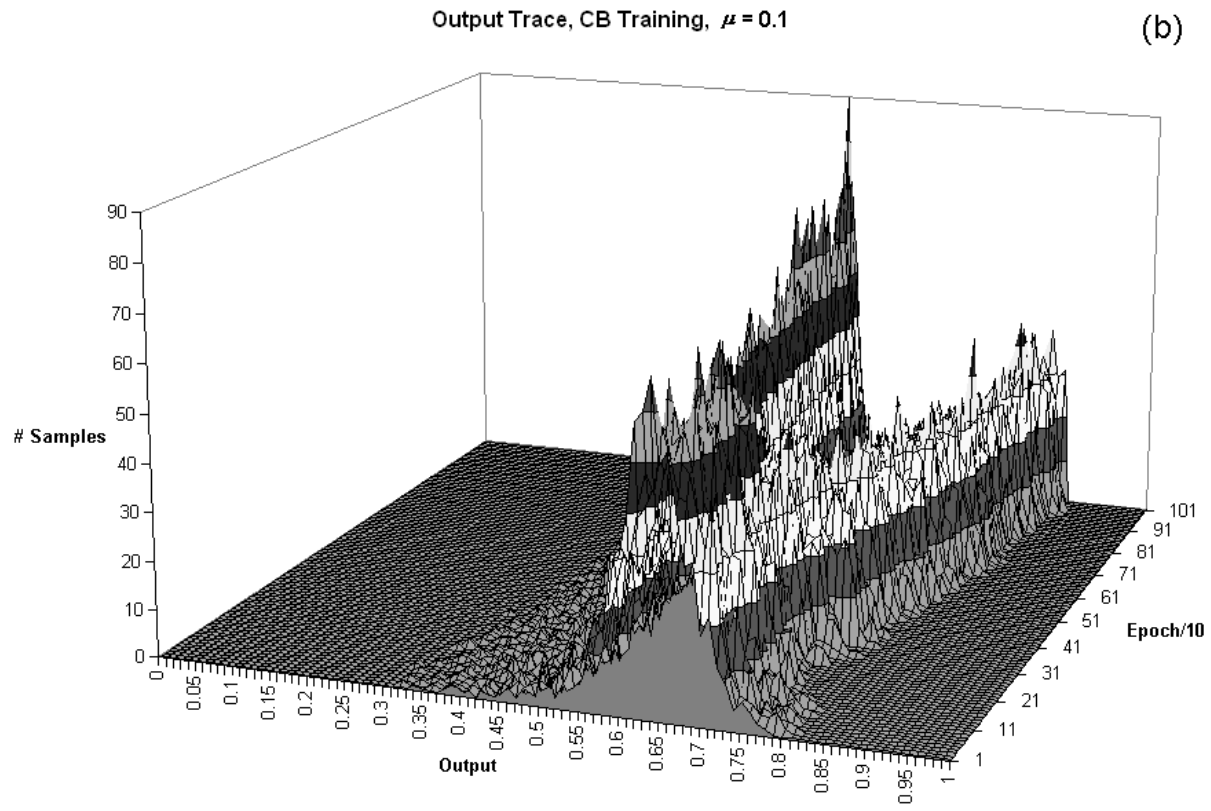


Figure 5. Network output trace during CB training on *bcw*.

5.1 Empirical effects of an error margin

Table 4 depicts the results of training with CB1 on *bcw* with values for μ ranging from 0 to 0.9. The top value is averaged classification accuracy and the bottom value is standard deviation using 10-fold stratified cross validation.

Table 4. 10-fold CV results for CB training on *bcw* with μ. Best results are in bold.

$\mu = 0.0$	0.01	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
95.17	**97.22**	**97.22**	**97.22**	96.78	96.93	96.78	95.90	95.61	95.90	95.75	95.75
2.09	2.42	**1.79**	2.01	1.96	2.29	**1.72**	**1.69**	2.94	2.26	2.32	2.40

These results show that CB1 is fairly robust to the selection of μ. A μ greater than 0 causes the decision surface to be further removed from test patterns in general and increases generalization as a result. Values closer to 0 show the most improvement and μ values closer to 1 cause CB1 to revert proportionally to the behavior of standard SSE training. (Note that the accuracy shown for $\mu \sim 1.0$ does not match the accuracy for SSE training in Table 3 because the accuracies in Table 3 are based on roughly optimized parameters for each error function, and CB and standard training have different optimal learning parameters.)

5.2 Effect of SSE on output values

Following a training run on *OCR* training to minimize SSE, winning network outputs on the test set were distributed as shown on the logarithmic scale in Figure 8. The network outputs were very close to 1.0 on the majority of the patterns. Only 2-3% of the patterns lie close to where the decision surface is located (implicitly at 0.5). The weights have grown in magnitude to the point that the dividing sigmoidal surface is very sharp.

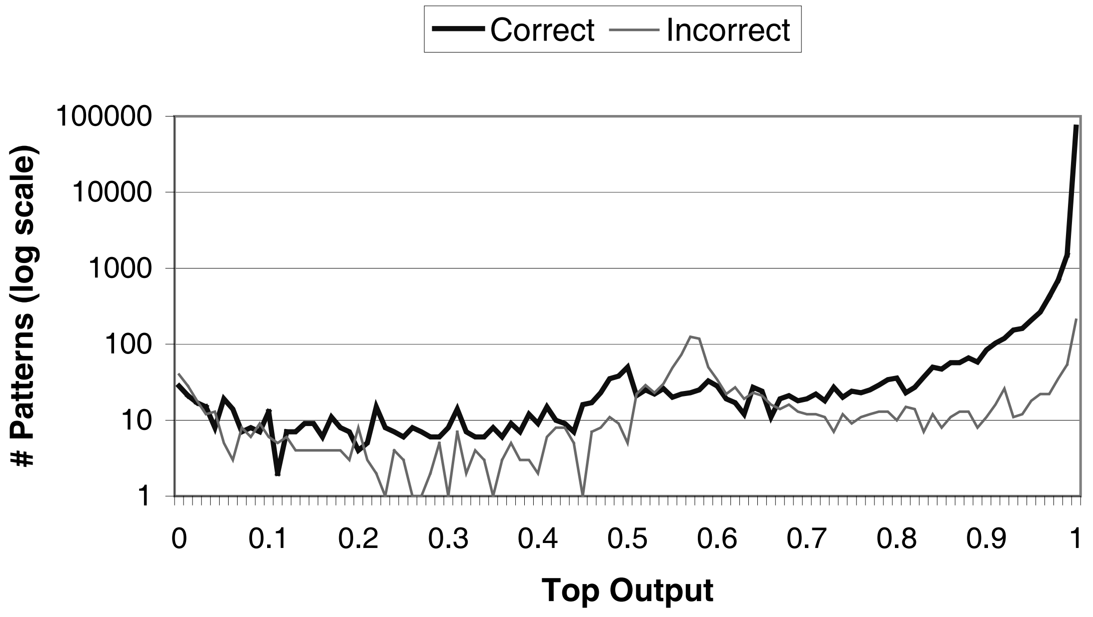


Figure 6. Network outputs on *OCR* test set after SSE minimization.

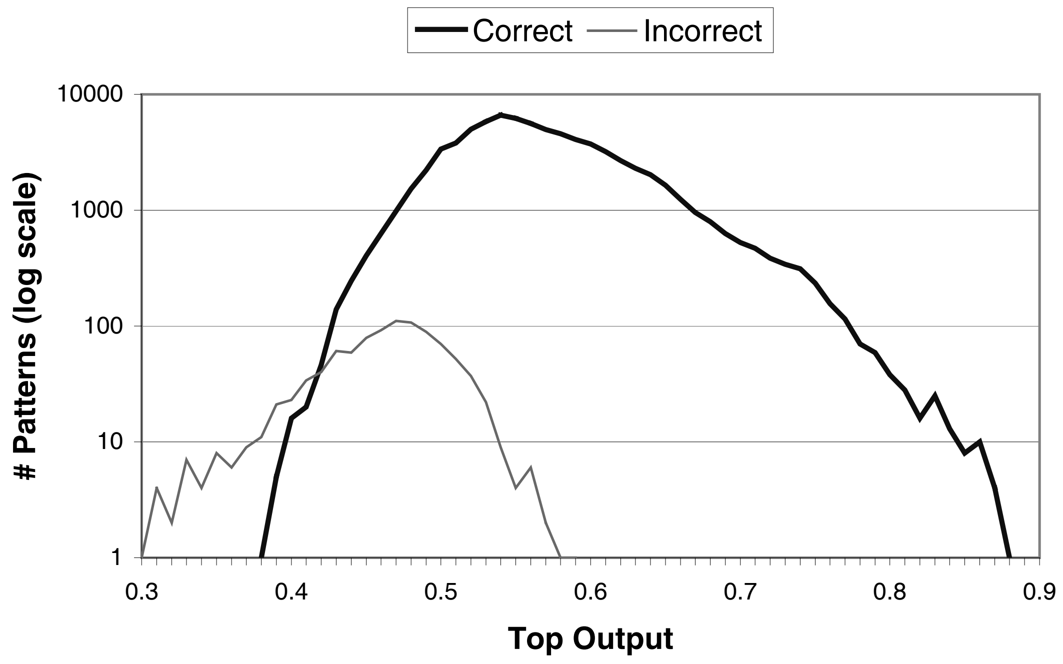


Figure 7. Network outputs on *OCR* test set after CB training.

5.3 Effect of CB training on output values

CB training produces a final output distribution quite unlike that seen in Figure 6. When networks only perform weight updates to prevent misclassification, instead of pushing the pattern outputs to one end of the output range or the other, the vast majority remains spread out just slightly above the decision boundary (see Figure 7). Pattern output distribution is roughly gaussian, reflecting an actual gaussian data distribution (i.e., gaussian noise in the *OCR* input features). There is a larger output variance than appears from standard SSE optimization but with only a fraction of the classification error. This suggests that the decision surface is much smoother and that network weights are not saturated. Misclassified patterns usually have outputs below 0.5 and are lower than the output for correctly classified patterns in the majority of cases.

5.4 Network complexity

At first, it seems counter-intuitive that networks outputting only around 0.5 will generalize so well. Ordinarily, training networks together allows a classifier to become more complex, prone to overfitting. According to Occam's razor, adding parameters to a network, beyond the smallest correct solution for a given problem, can be a detriment to the generalization ability of the network. This is similar to the claim that a network with higher learning capacity tends to "memorize" noise in the data, an undesirable trait.

However, it has been shown that the number of nodes in a network is not as influential as the *magnitude* of the weights (Bartlett, 1998). The topology, rather, serves more as a mechanism that lends itself to solving of certain problems, while the weights represent how tightly the network has fit itself to the (admittedly incomplete) training data distribution. Network complexity is further defined (Wang, Venkatesh, & Judd, 1994) as the number of parameters and the *capacity to which they are used in learning* (i.e., their magnitude). The authors show how network complexity is a generalization of Akaike's Information Criterion, which reveals

> *The generalization error of a network is affected not only by the number of parameters but also by the degree to which each parameter is actually used in the learning process.*

Occam's principle stands, in that it is best to make minimal *use* of the capacity of the network for encoding the information provided by the learning patterns (Wang, Venkatesh, & Judd, 1994). In light of this, it is understandable why training (often overly) complex networks using weight decay or CB training, which allow networks to converge with smaller weights than normal, perform well. Although the network has a large number of parameters, CB1 prevents further weight updates once patterns are correctly classified and results in low complexity. Hence, the possibility of overfitting is reduced in the training process.

The networks used in the *OCR* experiments (1 for each class) had 64 inputs, 32 hidden nodes and 1 output node, with 2080 weight parameters (plus 33 bias weights). The rows of Table 5 list the average magnitude of the weights in a network initialized with uniform random weights in the range [-0.3,0.3], after standard training, and after CB training, respectively. The columns denote the average magnitude of the bias weight on the hidden nodes, bias on the output node, average weight from input to hidden node, and from hidden to output node, respectively. The lowest weight magnitudes are bolded. The CB network has weights that are roughly two to four times larger than the initial random values, while SSE and CE training produce weights from ten to twenty times larger. The CB network is a simpler solution than the networks produced by backpropagation training optimizing SSE or CE.

Table 5. Average final network weight magnitudes.

Method	**Hidden Bias**	**Output Bias**	**Hidden Weight**	**Output Weight**
Initial	0.16	0.15	0.15	0.15
SSE	2.21	4.66	1.27	6.25
CE	2.56	4.95	1.43	4.16
CB1	**0.56**	**0.02**	**0.31**	**0.74**

5.5 Multi-task learning

Common training methods for learning multiple tasks involve training multiple networks separately, one for each task. However, learning the subparts of a complex problem separately may not be a good idea. Independent training of domain-specific experts is only marginally beneficial to the system as a whole. *Multi-task learning* (MTL), learning multiple problems simultaneously with a single multiple-output network, is described by Caruana (1993; 1995; 1996; 1997). Caruana shows how learning multiple tasks in conjunction (e.g., when learning how to automatically drive a vehicle, recognizing where the road's center line and/or left and right lanes are located in addition to deciding just how to steer) helps to avoid local minima and improve generalization. MTL performs better (learning tasks simultaneously) than learning tasks separately (Caruana, 1993). There are several reasons why MTL improves on single-task learning (STL). Advantages and disadvantages of using MTL over STL are discussed in (Caruana, 1993).

A problem that often occurs during training is due to the moving-target problem, and is referred to by Fahlman (Fahlman & Lebiere, 1990) as the *herd effect*. Suppose there exist two separate computational sub-tasks, A and B, which must be performed by the hidden nodes of a network. Any of the hidden nodes could handle either task, but since they have no way of communicating among themselves, they must decide independently which task to solve. If task A produces a more coherent error signal than B, there is a tendency for all the hidden nodes to focus on learning A. Once A is redundantly solved by all the nodes in the network, then they begin to work on B, which is now emitting the only significant error signal. If they move toward B all at once, then problem A reappears. Often the weights can be observed to oscillate for a prolonged period of time until the "herd" finally splits up and deals with both sub-tasks at once. This phenomenon, where weights oscillate back and forth, can be seen on complex problems (e.g., *2 spiral*). The herd effect can be observed not only among nodes in an MTL or STL network, but among networks being trained simultaneously in an ensemble as well.

5.6 CB learning with single vs. multiple networks

Using single-output networks to learn each class in the problem ensures each class is learned separately. Learning classes separately might allow easier analysis of solutions, whereas deciphering the meaning of network weights in a multi-output network is very difficult. However, there are advantages to CB training using a single multi-output network over separate single-output networks. Training a single network takes advantage of the benefits of MTL. Where problem hypotheses overlap, a single network can "reuse" nodes by taking advantage of redundant features. This produces a more compact solution than having to relearn redundant features in separate networks. In experiments on the *OCR* set 47 networks were trained. Each network had a 64x32x1 architecture, (plus bias) yielding 2113 weight parameters in each network. In all, the model contains 2113 x 47 = 99,311 weights, whereas the best single network has a 64x256x47 topology (plus bias). This equals 16640 + 12079 = 28,719 weights, a reduction in size of nearly three-and-a-half times. The practical implications of this are that not only is memory conserved, but classification speed is increased as well.

Caruana (1995) states one of the disadvantages of MTL is that, since tasks are learned at different times during training, it is difficult to know when to stop training. When training is stopped early, some tasks might not have been learned and generalization is often impaired as a result. Caruana's solution is to train the network until all tasks appear to be overfitting, or to take a separate snapshot of the network for each class, at the point where its validation accuracy is highest. However, taking several snapshots makes the solution much more unwieldy, and although the snapshot is taken at the point where accuracy is highest, there is no guarantee that overfitting has not already occurred in *some part* of the space for that class.

CB training solves both problems by naturally stopping training on tasks as they are learned, both within classes and among them. This helps in two ways: the solution can be kept small (using a single network), and overfitting is discouraged on two levels, both external to learning a class (overfitting a class because other classes have yet to be learned sufficiently) and internal to it (overfitting on localized regions of a class because other regions have yet to be learned).

CB training of multiple networks goes a step beyond MTL. Note that in Section 4.3.1, the best *OCR* test accuracy obtained was using multiple networks. It appears their increased computational ability (more network parameters) over the monolithic model enables them to be used as needed to find a better solution than with a single multi-output network, while CB training discourages abuse of the increased potential of the system to overfit. In addition to having specialized networks learning individual tasks at the same time, CB explicitly shares relevant information among the networks, in the form of output values, during training to coordinate their learning process. When networks are trained concurrently, rather than sequentially as in standard ensembles, they can take advantage of greater expressive power through interaction during the training process. Two or more networks can collaborate together to decide how learning is to proceed at any given point. Network interactivity is discussed further in Section 7.

5.7 Computational Cost

CB1 requires an O(n) search through the n network outputs to determine the highest target and competitor values. However, this additional overhead to the learning algorithm is negligible compared to the computation requirements of O(ih) for feed-forwarding a pattern vector and O(ihn) for backpropagation, where i is the number of inputs and h is the number of hidden nodes. In fact, CB1 saves O(ihn) time by omitting the error backpropagation step over correctly classified patterns.

6. Considerations in Neural Network Training using CB1

In this section, several issues are enumerated that must be considered when designing an effective neural network backpropagation learner. How each of these issues is dealt with, to some extent, has a significant effect on generalization. How the CB training philosophy presented in this work fits into these issues is discussed.

6.1 Altering Network Topology

Network topology plays a large role in achieving high generalization. Most commonly, solutions involve training a fully connected network. However, it has often been shown that partially connected networks perform as well or better than fully-connected ones. Pruning algorithms, such as Optimal Brain Damage (LeCun, Denker, & Solla, 1990) and Optimal Brain Surgeon (Stork & Hassibi, 1993), reduce the connectivity in an overly specified network, and construction algorithms (several are enumerated in (Andersen & Martinez, 2001b)) insert needed connections into a skeleton-network until sufficient function approximation is achieved.

6.1.1 Pruning Algorithms

For a fixed amount of data, networks with too many weights often do not generalize well. On the other hand, networks with too few weights will not have enough power to represent complex data accurately. The best generalization is often obtained by trading off the training error and the network complexity (LeCun, Denker, & Solla, 1990). If it is possible to reduce network complexity without reducing training error, then it is expected that generalization accuracy will improve.

Network complexity is defined (Wang, Venkatesh, & Judd, 1994) as the number of parameters and the capacity to which they are used in learning (i.e., their magnitude). A network with a few large weights may effectively be more complex than a network with a greater number of small weights. Hence, complexity can be reduced not only through pruning parameters, but also by reducing their values. A learning algorithm that aims at preserving small weights during training can aid in improving generalization. One example of this is performing weight decay (Werbos, 1988), which serves to weaken overly strong or saturated connections and in effect remove unused network connections. However, weight decay serves more as a recovery technique to repair the damage caused by minimizing the error function as weights tend toward saturation, rather than providing a heuristic that specifically aims at small-weight solutions. The CB training algorithm presented in this work actively attempts to find good solutions with weights remaining as small as possible to avoid saturation.

6.1.2 Growth Algorithms

Dynamic network construction algorithms typically start from a very simple basis and progressively add complexity until the training data are acceptably learned. Theoretically, a network can always be grown until it has perfectly learned the training data. However, at this point it often acts as a table lookup and exhibits poor generalization. Therefore in growing networks it is essential to choose a proper stopping point. Growth and pruning algorithms can be used in conjunction to first grow a network that empirically has the capability to learn the training data, and then prune nodes until accuracy on a holdout set begins to decrease.

Several growth methods append nodes after the current output node (Andersen & Martinez, 2001b), which is disadvantageous since the original output node was not trained specifically as a feature detector for use in the new network. Cascade Correlation (Fahlman & Lebiere, 1990) and DMP3 (Andersen & Martinez, 2001b) add nodes before the output node. CB training effectively teaches each output node to function as a feature detector that performs in conjunction with the other output nodes, rather than on its own. In this way, their mutual results can be combined into meaningful, non-redundant output.

6.2 Early Stopping

Early stopping strategies (Wang, Venkatesh, & Judd, 1994) commonly utilize network architectures that have the potential of being overly complex. Larger network architectures are likely to converge to a lower training error, but tend to produce higher error on test patterns. In order to avoid this, early stopping strategies try to determine when the problem has been learned sufficiently well, but not yet overfit (Andersen & Martinez, 2001b).

(Wang, Venkatesh, & Judd, 1994) shows that stopping learning before the global error minimum has the effect of network size selection. This can be accomplished through a number of methods, such as considering the accuracy of a validation, or holdout, set, and stopping training when the performance on the holdout set begins to degrade (Andersen & Martinez, 2001b).

CB training performs an "online" form of early stopping. Rather than stopping training completely when it is detected that the training set is being overfit, CB1 selectively omits training on *individual patterns* when backpropagating an error signal would not increase accuracy further.

6.3 Model Complexity

It is often believed that networks with too many degrees of freedom generalize poorly. This line of reasoning is based on two observations: (1) that a sufficiently large network is able to memorize the training data if training continues long enough, and (2) even with early stopping approaches, it is not apparent whether some form of overfit has occurred. By reducing the learning capacity of such a network, it is thereby forced to generalize as it no longer has the capability to memorize the training data.

Caruana (1997) points out that in order to perform a proper theoretical analysis of network capacity and generalization, the search heuristic must also be taken into account. Gradient descent search heuristics do not give all hypotheses an equal opportunity. The inductive bias of standard backpropagation is to start with a simple hypothesis (through usually small, random weights) and make the hypothesis more complex (by increasing the magnitude of the weights) until the network sufficiently learns the problem.

Thus, backpropagation is biased toward hypotheses with small weights, examining solutions with larger weights only as dictated by necessity. Excess network capacity does not necessarily hinder generalization, as learning stops as soon as possible. This stopping point is dictated in part by the objective function. During the first part of training, large networks behave like small networks. If they do not come to a satisfactory solution, they begin to perform less like small networks and more like mid-size networks, and so on. If a large network is too big, early stopping procedures will detect when generalization begins to degrade and halt training. At this point, the larger network performs similar to some smaller network. This means that generalization can be less sensitive to excess network capacity, and that using a network that is too small can hurt generalization more than using networks that are too large (Caruana, 1997).

The ability to perform online per-pattern stopping, combinable with standard early stopping techniques, enables CB training to be more robust in its management of excessively large networks. In our search for optimal network sizes in the experiments above, CB1 proved to be more robust to overly large numbers of hidden nodes than SSE and CE optimization.

6.4 Bias and Variance

A network's *bias* and *variance*, as defined in (Geman & Bienenstock, 1992), can be intuitively characterized as the network's test set generalization and its sensitivity to training data, respectively. There exists an inherent tradeoff between bias and variance, namely

> *The best generalization requires a compromise between the conflicting requirements of small variance and small bias. It is a tradeoff between fitting the training data too closely (high variance) and taking no notice of it at all (high bias)* (Sharkey, 1996).

Bias is the extent to which the network's output varies from the target function (the error), while variance is the sensitivity to the training data sampled in affecting generalization (the variance of the constructed hypothesis from the optimal Bayes hypothesis). An ideal function approximation network has low bias and low variance.

An ensemble with high variance tends to have low correlation of errors since each network arrives at a unique hypothesis. The ideal ensemble is a set of networks that do not show any coincident errors. In other words, each network has good generalization (low bias), and when a network is in error, the error is not shared by other outputs (high variance) and can be corrected through voting. If only one output is in error, it can be overruled by considering the majority of correct outputs of the remaining networks. However, ambiguity results when more than one (i.e., close to a majority) output is in error. Low bias and high variance is desirable in an ensemble, and having sufficiently high variance can make up for moderately high bias. Variance is commonly introduced into ensembles by varying the data presented to each network. This can be done through data sampling, disjoint training sets, adaptive resampling, providing different data sources, preprocessing, or a combination of these (Sharkey, 1996). CB training does not attempt to reduce error when it can be resolved by jointly considering the output values (as in ensembles), so as not to increase variance needlessly though overfitting.

Friedman illustrates that low SSE bias is not important for classification, and one can reduce classification error toward the minimal (Bayes) value by reducing variance alone (Friedman, 1997). CB training reduces variance among outputs by "over-smoothing" the decision surface. SSE bias is acceptably increased, as CB training is used for classification tasks, not function approximation.

Variance is controlled by the degree of over-smoothing – more smoothing creates less variance. CB training accomplishes over-smoothing by discouraging patterns from affecting the shape and location of the decision surface more than is required for correct classification.

6.5 Overfitting

The question of how to prevent overfit is a subtle one. When a network has many free parameters, not only can learning be fast, but also local minima can often be avoided. On the other hand, networks with few free parameters tend to exhibit better generalization performance (Castellano, Fanelli, & Pelillo, 1997). Determining the appropriate size network remains an open problem.

In taking all of the above issues into account, overfitting is typically considered to be a global phenomenon. However, the degree of overfitting can vary significantly throughout the input space. Lawrence and Giles (2000) show that overly complex MLP models can improve the approximation in regions of underfitting, while not significantly overfitting in other regions. However, their discussion is limited to function approximation tasks and not classification problems, which are affected in a different way by bias-variance tradeoffs (Friedman, 1997). CB training seeks to achieve minimal overfit not only globally but also locally by not training on patterns that are already correctly classified.

7. Interactive Training

It has been proposed (Wegner, 1997; Wegner & Goldin, 1999) that learning models that interact with an external environment (*e.g.*, another learner) have a greater theoretical power of expression than non-interactive models. To support this, (Weiss, 1999) shows that coupled agents perform much more efficiently than independent agents at complex learning tasks. The paradigm shift from optimized, but isolated, algorithms to interactive models reflects the current evolution in the philosophy of the field of computer science from procedure-oriented to object-oriented languages and single mainframes to networks of personal computers.

Neural network models that learn interactively are proposed to be superior over independent models. An implementation of interactive ensembles is presented in (Liu & Yao, 1999a; 1999b), where the networks in the ensemble are trained simultaneously with the inclusion of an additional error term that encourages negative error correlation among the networks. This generally provides some improvement. However, the field of interactive learning among neural networks is largely unexplored. CB1 is an original contribution to the budding field of interactive neural network learning.

Interactive methods can be performed on separate, specialized single-output networks or a single multi-output network (as discussed in Section 5.6), but discussion in this section will be limited to separate networks for simplicity.

7.1 Coordinating objective function

The philosophy of CB training has ties to multi-agent learning. Stirling points out that typically, optimal single agent solutions are not jointly optimal in multi-agent settings (Stirling & Goodrich, 1999). This is demonstrated for neural network architecture optimization schemes for voting (Andersen, Rimer, & Martinez, 2001a). It is shown that the optimal network architecture selected for a single network model is typically not the optimal architecture when many such networks are combined with voting techniques such as bagging. The optimal architecture when a single network is used is often much less complex than the optimal network architecture of several networks being combined into a voting ensemble.

Coordination among agents occurs if members of a multi-agent system use information concerning the existence, decisions, or perceived decision-making strategies of other agents. Coordination should provide a means of decision-making for each agent in the system as a function of both the agent's individual agenda and its relationships to the other members of the system (Stirling, Goodrich, & Frost, 1996). Coordinating means enabling a network to update its parameters based not only on its behavior, but the behavior of the other networks in the system. A network's complexity must be necessarily restrained when using standard learning heuristics to avoid overfitting. Coordinating network behavior using CB training allows the use of a more complex model without engendering overfit.

CB training provides coordination among multiple single-output networks, or among output nodes in a multi-output network. CB training illustrates the principle of *satisficing* (Herbert, 1959), where an aspiration level is specified, such that once that level is met, the corresponding solution is deemed adequate. CB training balances an output's *credibility*, or the exactness with which it can produce ideal target values for its class (*e.g.*, reducing SSE to zero), against its *rejectability*, or the risk of overfitting by doing so. A trade-off is created between exactness in individual class outputs and the classification accuracy of the system. An output node can satisfactorily perform less "ideally" with the understanding that the effectiveness of the entire system can be improved as a result. In relaxing the constraint of optimal credibility, resultant rejectability is reduced.

8. Conclusion and Future Work

CB training produces less overfit in gradient descent backpropagation training than optimizing SSE and CE. It produces simpler hypotheses than SSE and CE, increasing the probability of better generalization. Its robustness and superior generalization over SSE and CE backpropagation has been demonstrated on several real world data sets. On UCI MLDR problems there was an average increase in accuracy from 90.7% for optimized SSE networks to 92.1% for CB training performing 10-fold stratified cross-validation. Similarly, there was an increase in accuracy from 97.86% to 99.11% on a very large OCR data set.

There are many directions that future research on CB training will take. The effect of modifying the error margin will be considered. Implementing dynamic versions of these parameters that change over time or using values local to each instance in the training set presents a straightforward extension to be evaluated. Softprop, a learning approach combining CB1 and SSE optimization during training has shown further improvement in a preliminary study and a thorough analysis will be presented in future work.

Further studies of the effect of network size on CB training behavior and classification accuracy will be done. The effect of CB training in more sophisticated classification systems, where classification functions more complex than a single MLP are used, such as stacking a perceptron, MLP, or rule-based or search system onto an existing system (Rimer, Martinez, & Wilson, 2002), will be investigated.

CB training will be combined with other learning enhancements. Augmenting CB1 with weight decay to produce even "simpler" solutions, RPROP (Riedmiller & Braun, 1993) or Quickprop (Fahlman, 1988) techniques to improve the rate of convergence, *speed training* (Rimer, Anderson, & Martinez, 2001b) to speed up learning, methods for growing and pruning networks, and forming network hierarchies are approaches being considered. CB training variants will also be considered for batch learning.

It has been observed that classification errors between SSE and CB trained networks are highly uncorrelated. Ensembles combining SSE and CB trained networks will be analyzed with the expectation that this will further reduce test error.

References

Andersen, T., & Martinez, T. (1995). A Provably Convergent Dynamic Training Method for Multilayer Perceptron Networks. Proceedings of the *2nd International Symposium on Neuroinformatics and Neurocomputers* (pp. 77-84).

Andersen, T., & Martinez, T. (1996). Using Multiple Node Types to Improve the Performance of DMP (Dynamic Multilayer Perceptron). Proceedings of the *IASTED International Conference on Artificial Intelligence, Expert Systems and Neural Networks* (pp. 249-252).

Andersen, T., & Martinez, T. (1999). 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).

Andersen, T., Rimer, M., & Martinez, T. (2001a). Optimal Artificial Neural Network Architecture Selection for Voting. Proceedings of the *IEEE International Joint Conference on Neural Networks IJCNN'01* (pp. 790-795).

Andersen, T. & Martinez, T. (2001b). DMP3: A Dynamic Multilayer Perceptron Construction Algorithm. *International Journal of Neural Systems, 11*, No. 2 (pp. 145-165).

Barnard, E. (1991). Performance and Generalization of the Classification Figure of Merit Criterion Function. *IEEE Transactions on Neural Networks, 2*, No. 2 (pp. 322-325).

Bartlett, P. (1998). 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), (pp. 525-536).

Bianchini, M., Frasconi, P., Gori, M., and Maggini, M. (1998). Optimal learning in artificial neural networks: A theoretical view. In C. Leondes, (Ed.), *Neural Network Systems Techniques and Applications*, 1-51. Academic-Press.

Blake, C. & Merz, C. (1998). UCI Machine Learning Repository, http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.

Caruana, R. (1993). Multitask Connectionist Learning. Proceedings of the *1993 Connectionist Models Summer School* (pp. 372-379).

Caruana, R. (1995). Learning Many Related Tasks at the Same Time With Backpropagation. *Advances in Neural Information Processing Systems 7* (Proceedings of NIPS'94) (pp. 657-664).

Caruana, R., Baluja, S. & Mitchell, T. (1996). Using the Future to 'Sort Out' the Present: Rankprop and Multitask Learning for Medical Risk Evaluation. *Advances in Neural Information Processing Systems 8* (Proceedings of NIPS'95).

Caruana, R. (1997). *Multitask Learning*. Ph.D. Thesis, School of Computer Science, CMU.

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

Chakraborty, G., Sawada, M., & Noguchi, S. (1997). Combining Local Representative Networks to Improve Learning in Complex Nonlinear Learning Systems. *IEICE Trans. Fundamentals*, *E80-A*, No. 9 (pp. 1630-33).

Duda, R., Hart, P. & Stork, D. (1999). *Pattern Classification*, 2nd edition. Wiley, John & Sons, Inc.

Fahlman, S.E. (1988). Faster-learning Variations on Back-propagation: An Empirical Study. Proceedings of the *1988 Connectionist Models Summer School*, Morgan Kaufmann.

Fahlman, S., & Lebiere, C. (1990). The Cascade-Correlation Learning Architecture. In D. Touretsky (ed.), *Advances in Neural Information Processing Systems 2 (524-532)*. Morgan Kaufmann, San Mateo, California.

Friedman, J. (1997). On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery, 1*, no. 1 (pp. 55-77). Kluwer Academic Publishers.

Geman, S. & Bienenstock, E. (1992). Neural Networks and the Bias/Variance Dilemma. *Neural Computation, 4* (pp. 1-58).

Hampshire II, J. (1990). A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Transactions on Neural Networks, 1*, No. 2.

Herbert, S. (1959). Theories of decision-making in economics and behavioral science. *American Economic Review, XLIX* (p. 253).

Jacobs, R., Jordan, M., Nowlan, S., & Hinton, G. (1991). Adaptive Mixtures of Local Experts. *Neural Computation, 3* (pp. 79-87).

Joost, M. & Schiffmann, W. (1998). Speeding up Backpropagation Algorithms by using Cross--Entropy combined with Pattern Normalization. *International Journal of Uncertainity, Fuzziness and Knowledge-based Systems (IJUFKS), 6*, no. 2, (pp. 117-126).

Lawrence, S. & Giles, C. (2000). Overfitting and Neural Networks: Conjugate Gradient and Backpropagation. *International Joint Conference on Neural Networks*, Como, Italy (pp. 114–119).

LeBlanc, M. & Tibshirani, R. (1993). Combining estimates in regression and classification. *NeuroProse*.

LeCun, Y., Denker, J. & Solla, S. (1990). Optimal Brain Damage. In Touretzky, D. (ed.), *Advances in Neural Information Processing Systems (NIPS) 2* (pp. 598-605). San Mateo, Morgan Kaufmann Publishers Inc.

Liu, Y. & Yao, X. (1999a). Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics, 29*, no. 6 (pp. 716-725).

Liu, Y. & Yao, X. (1999b). Ensemble learning via negative correlation. *Neural Networks, 12*, no. 10 (pp. 1399-1404).

Maclin, R & Opitz, D. (1997). An empirical evaluation of bagging and boosting. *The Fourteenth National Conference on Artificial Intelligence*.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill Companies, Inc., Boston.

Riedmiller, M. & Braun, H. (1993). A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. Proceedings of the *IEEE Conference on Neural Networks.* San Francisco.

Rimer, M., Anderson, T., & Martinez, T. (2001a). Improving Backpropagation Ensembles through Lazy Training. Proceedings of the *IEEE International Joint Conference on Neural Networks IJCNN'01* (pp. 2007-2112).

Rimer, M., Anderson, T. & Martinez, T. (2001b). Speed Training: Improving Learning Speed for Large Data Sets. Proceedings of the *IEEE International Joint Conference on Neural Networks IJCNN'01* (pp. 2662-2666).

Rimer, M., Martinez, T., & Wilson, D. (2002). Improving Speech Recognition Learning through Lazy Training. Proceedings of the *IEEE International Joint Conference on Neural Networks IJCNN'02* (pp. 2568-2573).

Rimer, M. & Martinez, T. (2004). Softprop: Softmax Neural Network Backpropagation Learning. To appear in *IJCNN'04: Proceedings of the IEEE International Joint Conference on Neural Networks*.

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

Schiffmann, W., Joost, M., & Werner, R. (1993). Comparison of Optimized Backpropagation Algorithms. *Artificial Neural Networks*. European Symposium, Brussels.

Sharkey, A. (1996). On Combining Artificial Neural Nets. *Connection Science, 8* (3-4) (pp. 299-313).

Stirling, W., Goodrich, M., & Frost, R. (1996). Toward a theoretical foundation for multi-agent coordinated decisions. In *Proceedings of the Second International Conference on Multi-Agent Systems*, (pp. 345-352). Kyoto, Japan.

Stirling, W. & Goodrich, M. (1999). Satisficing games. *Information Sciences, 114* (pp. 255-280).

Stork, D. & Hassibi, B. (1993). Second order derivatives for network pruning: Optimal Brain Surgeon. In Sejnowski, T., Hinton, G. & Touretzky, D. (eds.), *Advances in Neural Information Processing Systems (NIPS) 5* (pp. 164-171). Morgan Kaufmann Publishers Inc., San Mateo.

Tollenaere, T. (1990). SuperSAB: Fast Adaptive Back Propagation with Good Scaling Properties," *Neural Networks, 3* (pp. 561-573).

Wang, C., Venkatesh, S., & Judd, J. (1994). Optimal stopping and effective machine complexity in learning. In Cowan, J., Tesauro, G., & Alspector, J. (eds.), *Advances in Neural Information Processing Systems, 6* (pp. 303-310). Morgan Kaufmann, San Francisco.

Wegner, P. (1997). Why Interaction is more powerful than algorithms. *Communications of the ACM.*

Wegner, P. & Goldin, D. (1999). Interaction, Computability and Church's Thesis. Accepted to the *British Computer Journal*.

Weiss, G. (ed.) (1999). *Multi-agent Systems, A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, Massachusetts.

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

Zarndt, F. (1995). A Comprehensive Case Study: An Examination of Machine Learning and Connectionist Algorithms. Masters Thesis, Brigham Young University.

List of Figures

List of Tables