Mach Learn (2006) 63:183–205
DOI 10.1007/s10994-006-6266-6

Classification-based objective functions

Michael Rimer · Tony Martinez

Received: 3 June 2005 / Revised: 4 November 2005 / Accepted: 11 November 2005 /
Published online: 3 March 2006

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 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 common error minimization heuristics, such as sum-squared error (SSE) and cross-entropy (CE), that do not explicitly minimize classification error. 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

Editor: Risto Miikkulainen

M. Rimer (✉) · T. Martinez
Computer Science Department, Brigham Young University,
Provo, UT 84602, USA
e-mail: mrimer@axon.cs.byu.edu

T. Martinez
e-mail: martinez@cs.byu.edu

1. Introduction

Artificial neural networks have received substantial attention as robust learning models for applications involving classification and function approximation (Rumelhart, Hinton, & Williams, 1985). This work proposes the use of classification-based (CB) objective functions to improve backpropagation, increasing generalization on complex classification tasks. The CB1 algorithm is presented as the main contribution. It is an example of a CB objective function suited to 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 misclassification. In doing so, it updates the network parameters as little as possible. This technique discourages weight saturation and overfitting and is conducive to higher accuracy in classification problems than optimizing with respect to common error 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%. 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.

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. Future work is outlined in Section 7 and conclusions are presented in Section 8.

2. Conventional objective functions

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 results 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 do not always decrease monotonically with the 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 target 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 naive 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 often leads to premature 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. One conventional approach uses "softer" targets like 0.1 and 0.9. This presents a less severe solution but still suffers from overfitting.

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

The validity of using common differentiable measures like 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 approximating the function of an arbitrary signal this presumption often holds. However, this assumption is invalid for classification tasks, where assigned real-valued target vectors are arbitrary values used to represent class labels. This suggests that other error 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 minimizing MSE might not be a reliable objective function for complex classification 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. The classification figure-of-merit (CFM) objective function was introduced in Hampshire II (1990) for learning classification problems when it was shown that SSE and CE errors are not necessarily correlated with classification accuracy. CFM separates the values of network outputs by as large a range as possible such that error minimization is monotonic with increasing classification accuracy. Like SSE and CE, this metric encourages weight saturation, which is often indicative of overfitting and detrimental to generalization (Bartlett, 1998).

2.2. Classification-based objective functions

Generalizing well, not minimizing error with respect to an objective function, is the goal of learning. LeCun's *digits* study mentioned above illustrates how objective functions can inaccurately reflect how well a problem has been learned. Hence, the objective function chosen for learning should approximate the true goal of the learner as closely as possible. CB1 more directly portrays how well the network has learned to classify the training patterns.

Similar to CFM, CB1 also attempts to increase the range between output activations. However, CB1 widens the range between outputs only when there is classification error.

When a misclassification is made, error is backpropagated only from those outputs that are credited with producing the error. Observe that this effectively narrows the gap between outputs, as they are transposed with respect to their traditional 0–1 target values. 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

There is an inherent tradeoff between fitting the (limited) training 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. A simple variant 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 called the approach classification-based training. Like Rankprop, CB1 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. With CB training, smaller weights near zero can provide an acceptable solution for classification tasks. Keeping weights smaller avoids problems caused by weight saturation.

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 ($0 \leq o \leq 1$ for sigmoid). Let o_k be the activation value of the kth output node in the network ($1 \leq k \leq N$). Let T designate the target class for the current pattern and c_k signify the class label of the kth output node. For target output nodes, $c_k = T$, and for non-target output nodes, $c_k \neq T$. 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,

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

The error, ε, back-propagated from the kth output node is then defined as

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

The error (1) represented in closed form is

$$\begin{aligned} \varepsilon_k \equiv\ & (o_{\sim T_{\max}} - o_k) I(\varepsilon_k = T \text{ and } (o_{\sim T_{\max}} \geq o_{T_{\max}})) \\ & + (o_{T\max} - o_k) I(c_k \neq T) \text{ and } (o_k \geq o_{T\max})) \end{aligned}$$

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 substantial 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 a higher value 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 outputs 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

When a pattern is already classified correctly, forcing output values closer to 0 or 1 often results in weight saturation and overfitting. 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 motivation 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 considerably 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

Figure 1 illustrates how sample variance in the training set can influence the decision surface arrived at using SSE and CB1 error functions on a two-class problem. Overfitting is minimized in CB training 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. In Fig. 1(b), the CB1 decision surface remains next to a noisy pattern. This is in contrast to conventional SSE training, where hard target values of 0 and 1 require pushing the decision surface as far away from all training points as possible, including noisy outliers (see Fig. 1(a)). Hence, test patterns in the area near the question mark falling next to an outlier of the competing class have a better chance of being correctly classified and network variance is substantially reduced.

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 Fig. 2). This is true for correctly classified patterns (those above 0 in Fig. 2), and

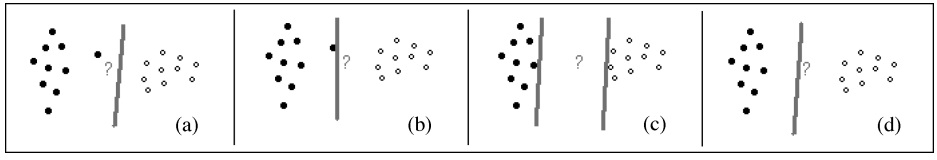


Fig. 1 Typical decision boundaries for SSE networks (a, d), CB1 networks without an error margin (c), and CB1 with a small error margin (b, d). CB1 induces a boundary more robust to noisy patterns

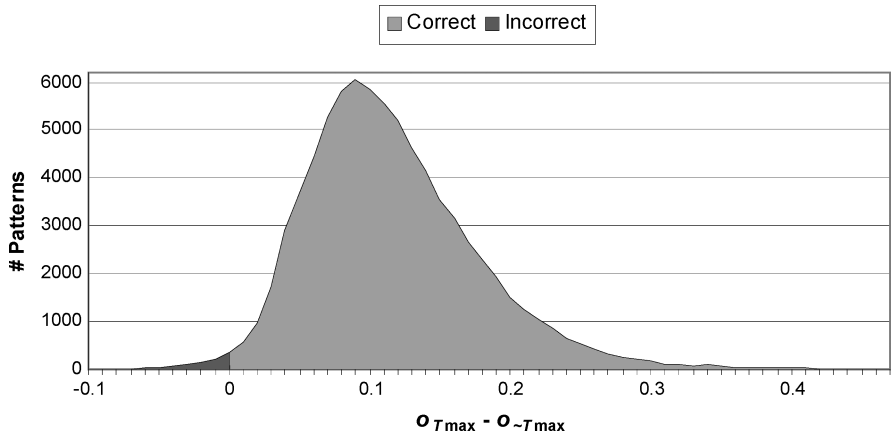


Fig. 2 Network output error margin after CB training on *OCR* data set. Values of output nodes are typically very close together, yet nearly all patterns are correctly classified

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 Fig. 1(c). According to the application this might not be desirable.

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

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

where min(·,1) and max(·,0) enforce the [0,1] error range of the logistic 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 Fig. 1(d)).

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 of 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

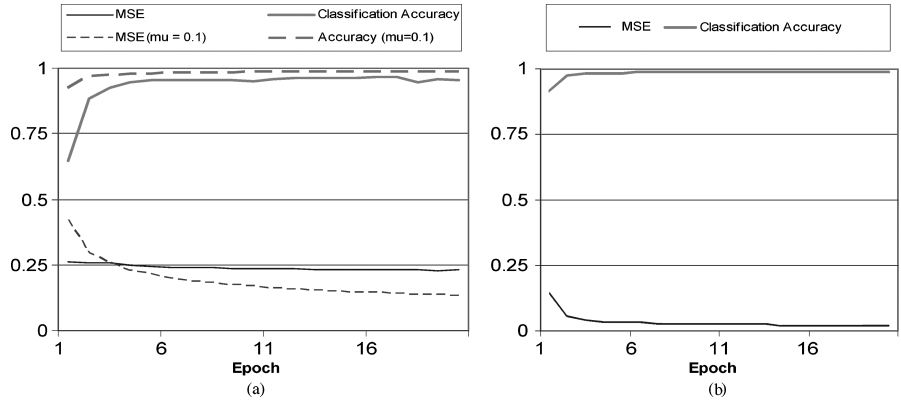


Fig. 3 (a) Classification accuracy and MSE during CB1 training. MSE decreases very slowly during training. (b) Classification accuracy and MSE during SSE optimization. MSE decreases quickly during training

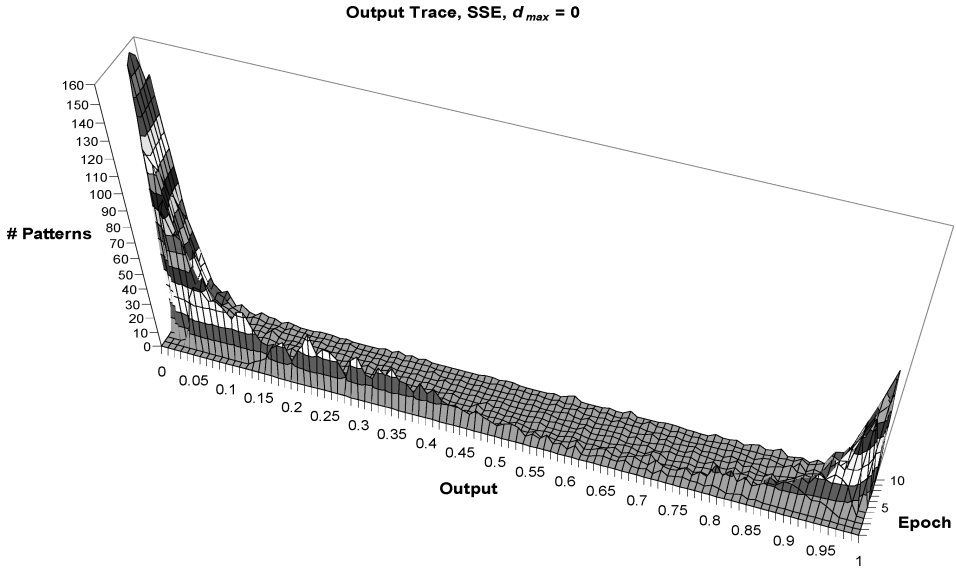


Fig. 4 Network output trace SSE optimization on *bcw* (first ten epochs). Weight saturation occurs after only a few training epochs

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:

- A single-output network on two-class problems (positive patterns are assigned a target value of 1.0, negative patterns are assigned a target value of 0.0)

- A single N-output network (one output per class on multi-class problems)
- N 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)

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 CB1.

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 roughly 370,000 alphanumeric character patterns and 47 symbolic classes, partitioned into 277,000 training patterns, 31,000 holdout set patterns, and 62,000 test patterns. Results on this data set were first presented in Rimer, Andersen, and Martinez (2001a).

Two network topologies were evaluated for learning *OCR*, a single N-output network and N single-output networks ($N = 47$ for *OCR*).

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* for each error function have been extensively optimized over the course of hundreds of empirical evaluations, comprised of tests of networks with topologies of one to three hidden layers, ten to five hundred hidden nodes per layer, learning rates from 0.01 to 0.5, and momentum from 0.0 to 0.99. We performed two sets of experiments on *OCR*. One tested multi-task learning (MTL), or using a single network with multiple output nodes, and the other used a separate network to learn each problem class. Pattern classification was determined by *winner-take-all* (the class of the highest outputting node is chosen) on all models tested.

For experiments presented here evaluating MTL, a hidden layer of 128 nodes was used. For experiments presented here training a separate, single output node network for each class label, each network has a single hidden layer of 32 hidden nodes. The learning rate was 0.1 and momentum was 0.7. Training was halted after 50 epochs (nearly 14 million patterns) of seeing no improvement in accuracy on the holdout set. The model selected for testing was the one with the best holdout set classification accuracy. Although we only present empirical results for this combination of parameter values, it is noted that relative accuracies among the error functions were typical and comparable over the 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 holdout error ceased to decrease for 500 consecutive epochs. The model selected for testing was the one with the best holdout set classification accuracy.

4.3. Results

4.3.1. OCR data set

Tables 2 and 3 display the results of standard SSE and CE backpropagation versus CB1 on *OCR*. *Train%* and *Test%* are the training and test set accuracy on the selected network model in percent, averaged over five training runs. How well each model generalizes is indicated by *Test% / train%*. *Train MSE* and *Test MSE* are the mean squared errors for the training and test sets on the epoch from which this model was chosen. Best values are printed in bold face.

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	21-30-3	21-30-3	21-30-3
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

These tests demonstrate that multi-task learning of *OCR* generalizes better than using a separate network to learn each problem 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 (i.e., test accuracy/train accuracy) is improved. Observe that with a single multiple-output network, test set accuracy is within a fraction of one percent of training set accuracy 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. Worse generalization was observed using networks with more hidden nodes. When training 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 *OCR* trains and generalizes better than SSE, and CB1 performs significantly better than both of these ($p < 0.0001$). Network models generated with CB1 also have improved generalization. Observe that CB1 has a much higher MSE than the other methods, yet overfitting is reduced and generalization is improved. This is an important point that is discussed further in Section 5.

Table 2 Results on *OCR* with *N* single-output node networks

Error function	Train%	Test%	test%/train%	Train MSE	Test MSE
SSE	99.28	97.86	.9857	**.0047**	**.0092**
CE	99.37	98.10	.9872	.0094	.0110
CB1 ($\mu = 0.05$)	**99.61**	**99.11**	**.9950**	.1830	.2410

Table 3 Results on *OCR* with one *N*-output node network

Error function	Train%	Test%	test%/train%	Train MSE	Test MSE
SSE	98.79	98.62	.9983	.0274	.0313
CE	99.09	98.91	.9982	**.0160**	**.0221**
CB1 ($\mu = 0$)	99.15	98.96	.9981	.1594	.1800
CB1 ($\mu = 0.1$)	**99.38**	**99.26**	**.9988**	.0992	.0968

Generalization with the best CB1 model is 0.73% greater than the best model trained with SSE and 0.53% greater that the best CE-trained model. Considering only multiple-output networks, error drops from 1.38% for SSE and 1.09% for CE to 0.74% for CB1, increases in accuracy of 0.64% and 0.35%, respectively ($p < 0.0001$). Considering only the multiple single-output network models, error drops from 2.14% with SSE and 1.90% with CE to 0.89% with CB1, increases in accuracy of 1.25% and 1.01%, respectively ($p < 0.0001$).

4.3.2. UCI MLDR data sets

Table 4 lists the results of a naïve Bayes classifier taken from (Zarndt, 1995), standard SSE and CE backpropagation, and CB1 with SSE and CE error updates (whether the error gradient, discussed in Section 3.1, is $o(1\text{-}o)$ or 1, respectively) on eight UCI MLDR classification problems. Results were gathered using 10-fold stratified cross validation and averaged over thirty randomly-initialized training runs.

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. An asterisk indicates a confidence within $p < 0.05$ that the highest accuracy is significantly better than the second highest. The last two columns indicate the difference in value between CB1 (with SSE or CE error gradient) and SSE and CE optimization. Here, a higher first value in each

Table 4 Results on selected data sets from UCI MLDR using 10-fold stratified cross-validation

Data set	Bayes	SSE	CE	CB1 SSE	CB1 CE	CB1 SSE –SSE	CB1 CE –CE
ann	**99.7***	98.25	98.33	97.62	*98.76*	−0.63	0.43
	0.1	0.54	0.53	0.47	*0.51*	−0.07	−0.02
bcw	93.6	96.96	97.06	*97.22*	**97.36***	0.26	0.30
	3.8	2.01	**1.81**	2.01	**1.81**	0.0	0.0
ionosphere	85.5	89.00	*90.80*	90.60	**90.88**	1.60	0.08
	4.9	4.72	4.64	**3.75**	*3.87*	−0.97	−0.77
iris	94.7	93.83	94.37	**95.47***	*95.37*	1.64	2.00
	6.9	5.68	5.87	*5.31*	**5.25**	−0.37	−0.62
musk2	97.1	99.06	98.56	*99.15*	**99.27**	0.09	0.71
	0.7	0.37	0.62	*0.36*	**0.29**	−0.01	−0.33
pima	72.2	76.26	76.11	*76.69*	**76.82***	0.43	0.71
	6.9	*4.24*	4.36	**3.43**	6.46	−0.81	2.10
sonar	73.1	76.06	78.87	*80.77*	**81.92***	4.71	3.05
	11.3	9.37	9.03	*9.02*	**8.60**	−0.35	−0.43
wine	94.4	96.29	96.74	**98.31***	*97.19*	2.02	0.45
	5.9	4.45	4.13	**3.49**	*3.47*	−0.96	−0.66
Average	88.79	90.69	91.35	*91.97*	**92.20**	**1.28**	**0.85**
	5.06	3.92	3.87	**3.48**	*3.79*	**−0.44**	**−0.08**

Note: Best values are shown in bold and second best in italics. Statistical significance of $p < 0.05$ of the most accurate model is signified by an asterisk (∗)

cell indicates greater improvement, and a lower second value indicates smaller standard deviation.

The average increase in classification accuracy is from 90.69% for SSE training to 91.97% for CB1 with SSE error gradient, a 1.28% decrease in error (significant to $p < 0.035$). Using CB1 with a CE error gradient demonstrated a 0.85% increase in accuracy over CE training, from 91.35% to 92.20%. 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 supports the hypothesis 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 (Duda, Hart, & Stork, 1999). CB1 incorporates a more direct minimization of misclassified patterns in gradient descent procedures by reducing error on only misclassified patterns.

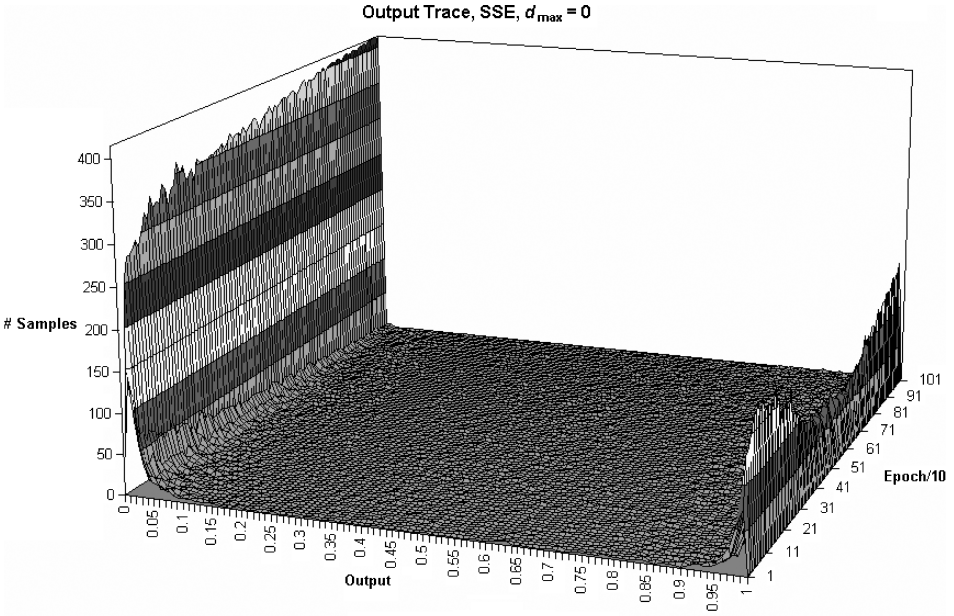
Since CB1 does not train using ideal target values like SSE and CE, MSE drops very slightly as training accuracy is improved (see Fig. 3(a)). This is in contrast to the strong, immediate drop in MSE illustrative of standard SSE optimization (see Fig. 3(b)). CE displays similar behavior to SSE optimization but is omitted from the following discussion for brevity.

In Fig. 3(a), rather than converging to zero, MSE remains just under 0.25 as training progresses with $\mu = 0$. With $\mu = 0.1$, MSE decreases to around 0.15. A large MSE is incurred by pattern outputs being very far away from the conventional 0–1 target values. Observe that a MSE of 0.25 is equivalent to a mean error of 0.5, which illustrates that output activations are close to 0.5 on average throughout training. This indicates that the weights for these outputs are close to zero. 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.

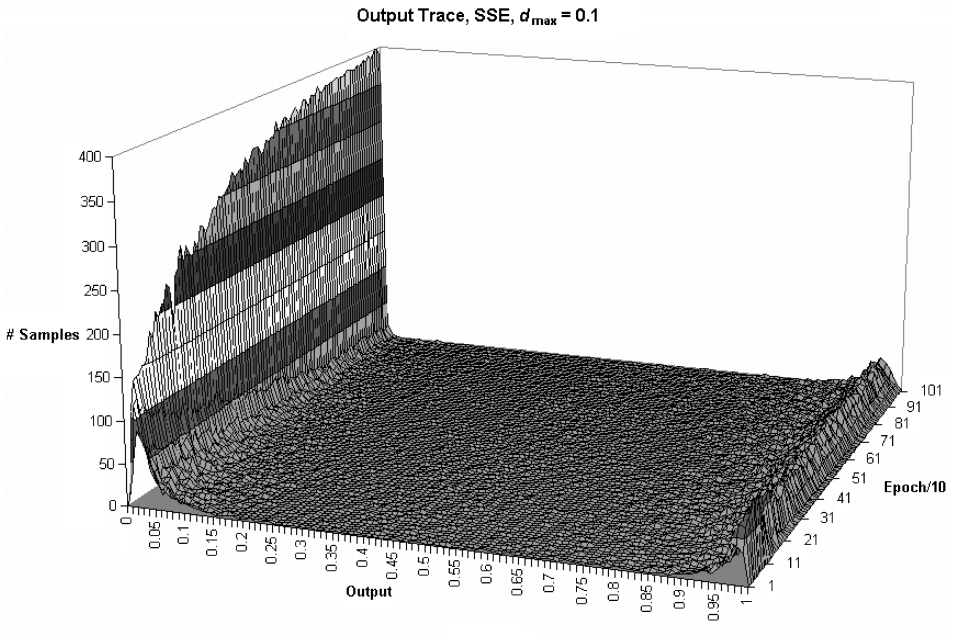
Figure 4–6 give insight into the behavior of the network during the learning process using four error functions. The surface plot shows a histogram of the values output by the network output nodes on the training patterns every epoch (Fig. 4) and every tenth training epoch (Figs. 5 and 6). Figures 4, 5(a) and (b) show learning minimizing SSE, and Figs. 6(a) and (b) show behavior during CB1 training. The results shown here are for the *bcw* data set, but such behavior is generally representative of all data sets tested.

In Fig. 4, it can be seen that SSE training forces the network to output values approaching 0 and 1 in fewer than ten epochs. It is noted that even after the first epoch, network outputs are already completely separate and distinct. Using a $d_{\max}$ of 0.1 reduces this tendency somewhat. Observe the flattened peaks for positive patterns in Fig. 5(b) that do not exist in 5(a).

CB training produces a starkly different behavior. In Fig. 6(a), it can be observed that all patterns output around 0.5 during the entire training process. In Fig. 6(b), incorporating a confidence margin of $\mu = 0.1$ widens the spread of output values, causing the output clusters of the two classes to visibly split apart as training progresses.



(a)



(b)

Fig. 5 Network output trace during SSE optimization on *bcw*. Network weights become quickly saturated during training

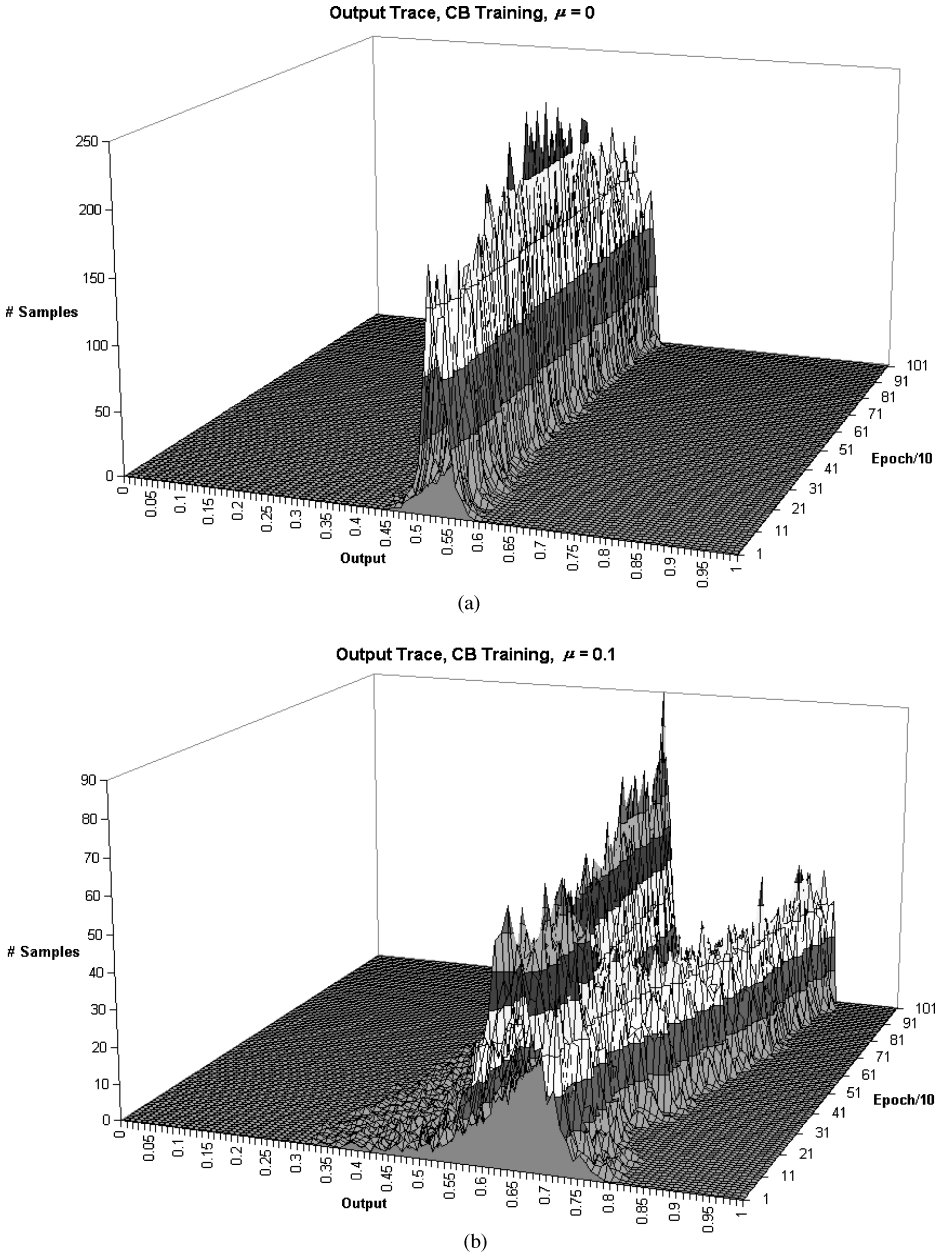


Fig. 6 Network output trace during CB training on *bcw*. Network weights do not become saturated during training

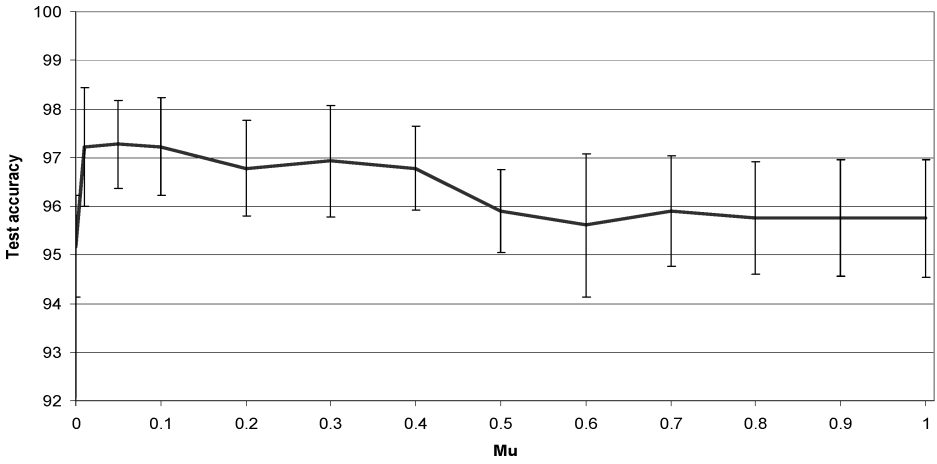


Fig. 7 10-fold CV results for CB training on *bcw* with μ. Small, non-zero values for μ typically demonstrate the best generalization

5.1. Empirical effects of an error margin

Figure 7 depicts the results of training with CB1 on *bcw* with values for μ ranging from 0 to 0.9. Each value bar shown is the averaged classification accuracy with standard deviation using 10-fold stratified cross validation.

This shows that CB1 is fairly robust to the selection of μ. Values for $\mu > 0$ cause the decision surface to be more removed from proximal test patterns than when $\mu = 0$ and have better generalization. Values for μ closer to 0 show the most improvement and values closer to 1 cause CB1 to revert proportionally to the behavior of standard SSE minimization. (Note that the accuracies shown for $\mu \sim 1.0$ do not match the accuracy for SSE training in Table 4 because the accuracies in Table 4 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 Fig. 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.

5.3. Effect of CB training on output values

CB1 training produces a final output distribution quite unlike that seen in Fig. 8. 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 Fig. 9). 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 SSE optimization but with only a fraction of the classification error. This suggests that the decision surface is

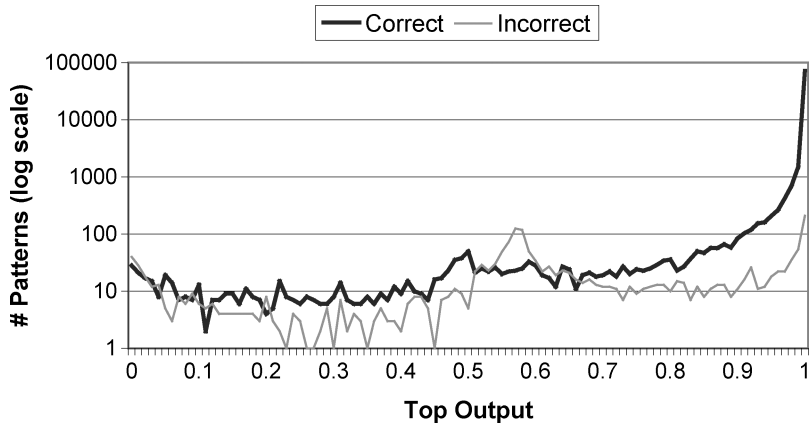


Fig. 8 Network outputs on *OCR* test set after SSE minimization are typically close to 1, indicative of large weights

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. 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

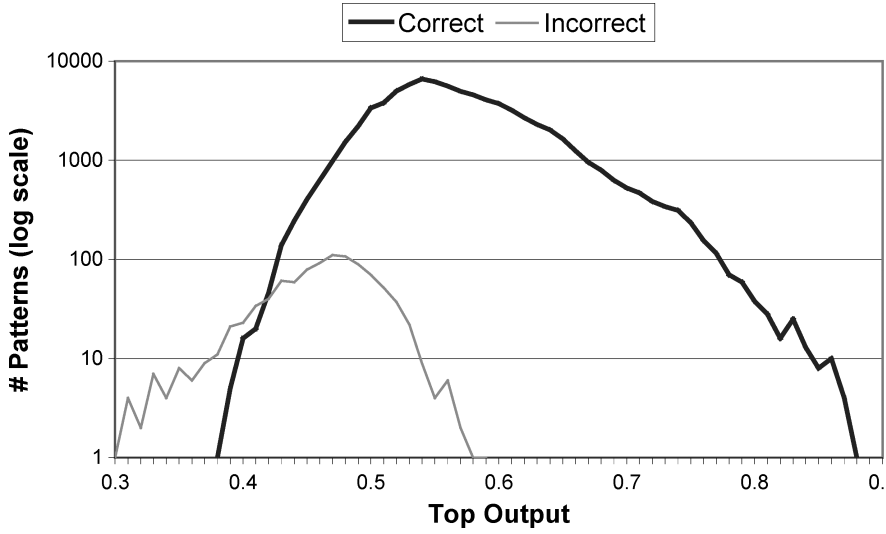


Fig. 9 Network outputs on *OCR* test set after CB1 training. Network weights are typically very small

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**

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.

That is, 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 (overly complex) networks using early stopping, weight decay or CB training, which allow networks to converge with smaller weights than normal, perform well. Although a network may have more parameters than strictly necessary, CB1 avoids superfluous weight updates when patterns are correctly classified. This results in lower network 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.

5.5. Multi-task learning with CB

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, 1997) and Caruana, Baluja and Mitchell (1996). Caruana shows how learning multiple tasks in conjunction 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). 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 $64 \times 32 \times 1$ architecture, (plus bias) yielding 2113 weight parameters in each network. In all, the model contains $2113 \times 47 = 99{,}311$ weights, whereas the best single network has a $64 \times 256 \times 47$ 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 was obtained 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.

5.6. 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. The number of epochs required to converge is similar for CB1 and conventional backpropagation training.

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 CB training deals with these issues is discussed.

6.1. Network complexity

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 regularization such as 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. CB1 actively attempts to find good solutions with weights remaining as small as possible to avoid saturation.

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 (Caruana, Lawrence and Giles, 2000).

Wang, Venkatesh and 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, 2001).

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) and Caruana, Lawrence and Giles (2000) 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. Overfitting

In taking all of the above issues into account, overfit is typically considered to be a global phenomenon. However, the degree of overfit can vary significantly throughout the input space. Caruana, 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 minimize overfit not only globally but also locally by not training on patterns that are already correctly classified.

A network's *bias* and *variance*, as defined in Geman and 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.

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). One way to reduce variance is by constructing a smoother decision surface. CB1 accomplishes this by discouraging patterns from affecting the shape and location of the decision surface more than is required for correct classification. SSE bias is acceptably increased, as CB training is used for classification tasks, not function approximation.

6.5. Coordinating objective function

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.

7. Future work

There are several directions that future research on CB training will take. CB training variants will be considered for batch and mini-batch learning. The effect of modifying the error margin, μ, in time and in space will be considered. Dynamically updating the value of the error margin as training progresses is a straightforward extension to be evaluated. Softprop, a learning approach combining CB1 and SSE optimization during training by means of the error margin, has shown improvement over CB1 in a preliminary study (Rimer & Martinez, 2004) and a thorough analysis will be presented in future work. Using a value for the error margin local to each training instance and intelligently updating these values as training progresses also shows promise. Also, 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.

8. Conclusion

CB training with the CB1 error function 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 applications. 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 test accuracy from 97.86 to 99.11% on a very large OCR data set.

References

Andersen, T., & Martinez, T. (2001). DMP3: A dynamic multilayer perceptron construction algorithm. *International Journal of Neural Systems*, *11:2*, 145–165.

Barnard, E. (1991). Performance and generalization of the classification figure of merit criterion function. *IEEE Transactions on Neural Networks*, *2:2*, 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*, 525–536.

Bianchini, M., Frasconi, P., Gori, M., & Maggini, M. (1998). Optimal learning in artificial neural networks: A theoretical view. In C. Leondes (ed.), *Neural network systems techniques and applications* (pp. 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. (1997). *Multitask learning*. Ph.D. Thesis, School of Computer Science, CMU.

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., lawrence, S., & Giles, C. (2000). Overfitting in neural networks: Backpropagation, conjugate gradient, and early stopping. *Proceedings of Neural Information Processing Systems*, *13*. MIT Press, 2001.

Chakraborty, G., Sawada, M., & Noguchi, S. (1997). Combining local representative networks to improve learning in complex nonlinear learning systems. *IEICE Trans. Fundamentals, E80-A*, *9*, 1630–1633.

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

Friedman, J. (1997). On Bias, Variance, 0/1-Loss, and the Curse-of-Dimensionality. *Data Mining and Knowledge Discovery, 1:1*, 55–77. Kluwer Academic Publishers.

Geman, S. & Bienenstock, E. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*, 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:*2.

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

Jacobs, R., Jordan, M., Nowlan, S., & Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*, 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)*, *6:2*, 117–126.

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

LeCun, Y., Denker, J. & Solla, S. (1990). Optimal brain damage. In D. Touretzky, (Ed.), *Advances in neural information processing systems (NIPS)* (vol. 2, pp. 598–605). San Mateo, CA: Morgan Kaufmann.

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

Rimer, M., Andersen, T., & Martinez, T. (2001a). Improving backpropagation ensembles through lazy training. *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'01* (pp. 2007–2012).

Rimer, M., & Martinez, T. (2004). Softprop: softmax neural network backpropagation learning. *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'04* (pp. 979–984).

Rumelhart, D. E., Hinton, G. E., & Williams, R. 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*, 299–313.

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,* (vol. 6, pp. 303–310). Morgan Kaufmann, San Francisco.

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.