8-1991

An Improved Algorithm for Neural Network Classification of Imbalanced Training Sets

Rangachari Anand
Syracuse University

Kishan Mehrotra
Syracuse University, mehrtra@syr.edu

Chilukuri K. Mohan
Syracuse University, ckmohan@syr.edu

Sanjay Ranka
Syracuse University

Recommended Citation

Anand, Rangachari; Mehrotra, Kishan; Mohan, Chilukuri K.; and Ranka, Sanjay, "An Improved Algorithm for Neural Network Classification of Imbalanced Training Sets" (1991). *Electrical Engineering and Computer Science - Technical Reports*. 104.
https://surface.syr.edu/eecs_techreports/104

SU-CIS-91-29

An Improved Algorithm for Neural Network Classification of Imbalanced Training Sets

R. Anand, K.G. Mehrotra, C.K. Mohan, and S. Ranka

August 1991

School of Computer and Information Science
Syracuse University
Suite 4-116, Center for Science and Technology
Syracuse, New York 13244-4100

An Improved Algorithm for Neural Network Classification of Imbalanced Training Sets

R. Anand, K. G. Mehrotra, C. K. Mohan and S. Ranka

August 15, 1991

Abstract

In this paper, we analyze the reason for the slow rate of convergence of net output error when using the backpropagation algorithm to train neural networks for a two-class problems in which the numbers of exemplars for the two classes differ greatly. This occurs because the negative gradient vector computed by backpropagation for an imbalanced training set does not point initially in a downhill direction for the class with the smaller number of exemplars. Consequently, in the initial iteration, the net error for the exemplars in this class *increases* significantly. The subsequent rate of convergence of the net error is very low. We suggest a modified technique for calculating a direction in weight-space which is downhill for both classes. Using this algorithm, we have been able to accelerate the rate of learning for two-class classification problems by an order of magnitude.

1 Introduction

Classification, the assignment of an object to one of a number of predetermined groups, is of fundamental importance in a number of areas ranging from image and speech recognition to the social sciences. Consequently, a number of statistical classification techniques have been developed, based primarily on Bayes' rule.

In the classification problem we assume that a pattern, can belong to exactly one of several classes. We are provided a training set consisting of sample patterns which are representative of all classes along with class membership information for each pattern. Using the training set, we deduce rules for membership in each class and create a classifier which can then be used to assign other patterns to their respective classes according to these rules.

One connectionist approach to the classification problem, which has gained popularity in recent years, is the use of backpropagation-trained [10] neural networks. Backpropagation, based on the method of steepest descent [6], is one of the most widely used training algorithms for feed-forward neural networks. Since these networks can be taught arbitrary non-linear mappings, it is relatively straightforward to adapt them for pattern classification tasks [5].

Although backpropagation has enjoyed wide popularity, it has been observed that the rate of convergence of error is very low in many applications. Consequently, several researchers have devised modifications to the backpropagation algorithm to increase the convergence rate. The general approach has been to vary the learning rate dynamically during training in order to maintain it at the largest value that will not cause oscillations [13] [2]. Attempts have been made to learn from a subset of the patterns to determine the network size and initialize the weights to reduce training time [12].

When training a network with backpropagation for a two-class problems in which the numbers of exemplars for the two classes differ greatly (i.e. the training set is *imbalanced*), we have observed that the rate of convergence of net output error is especially low. In an imbalanced training set, the class with more exemplars is called the *dominant* class while the other is called the *subordinate* class. Imbalanced training sets do occur frequently in practice.

In this paper, we show that the low rate of convergence of net error occurs because the negative gradient vector computed by backpropagation for an imbalanced training set does not initially decrease the error for the subordinate class. Consequently, in the initial iteration, the net error for the exemplars in the subordinate class *increases* significantly. The subsequent rate of convergence for the exemplars of the subordinate class is very low. To solve this problem, we suggest a modified technique for calculating a direction in weight-space which is downhill for both classes. Using this algorithm, we have been able to accelerate by an order of magnitude the rate of learning for two-class classification problems.

In section 2 of this paper, we consider the standard backpropagation algorithm and present an analysis of the MSE which points towards the reasons of the above mentioned drawbacks. In section 3, we present a modified backpropagation algorithm which performed significantly better than the standard backpropagation algorithm. A comparison of the two algorithms is made in section 4 for three examples and analysis is presented in section 5.

2 Backpropagation and classification problems

Although backpropagation has enjoyed wide popularity, it has been observed that the rate of convergence is often very low in many applications. Consequently, several researchers have devised modifications to the backpropagation algorithm to increase the rate of convergence of error. Vogl, et al. [13], suggest that the learning rate be modified during training depending on the rate of convergence of error. Anderson [2] suggests that every weight in a network should be given its own learning rate and and that these learning rates be varied during training.

We have observed that net error often converges especially slowly when training networks with the standard backpropagation algorithm for two-class problems with imbalanced training sets. In these problems, we have also found that the net error for exemplars in the dominant class is reduced rapidly in the first few iterations but net error for the subordinate class increases considerably. The subsequent rate of decrease of net error for the subordinate class is very low.

Typical behavior of the errors is shown in figure 1 where the net error of the subordinate and dominant class are plotted. A logarithmic scale is used for the X-axis in order to highlight the large change in net error that occurs in the first iteration. We analyze the cause of this phenomenon in section 2.2. Mathematical results are presented only for networks with one hidden layer.

2.1 Definitions

In order to explain the reasons for the observed phenomenon, it is necessary to reproduce some of the well known properties of feed-forward networks. In this section, we define these concepts and introduce necessary notation.

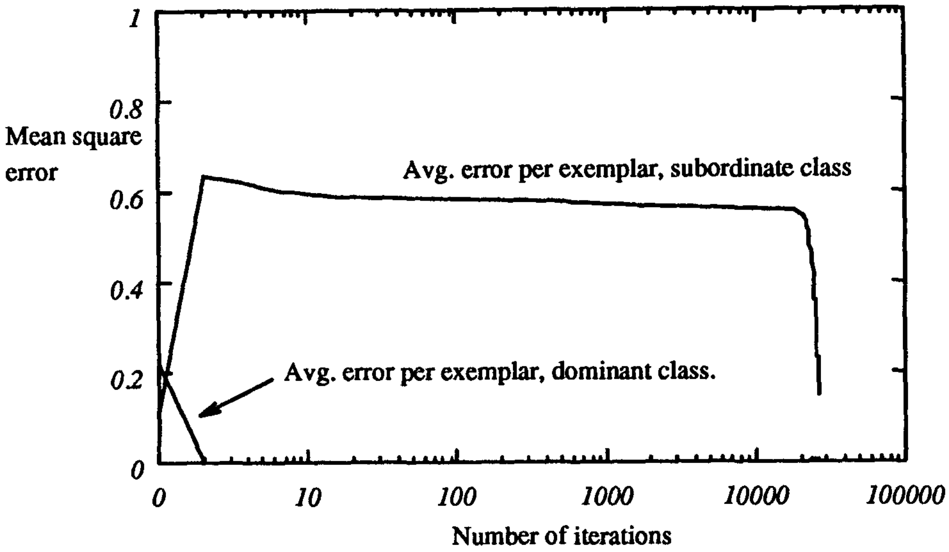

Figure 1: Net errors for dominant and subordinate classes after each iteration during a training session.

Network architecture: A schematic diagram of a feed-forward network is shown in figure 2. The nodes in the network are organized in the form of layers. There are no interconnections among nodes in the same layer. The output of each node in one layer feeds into all nodes in the next layer through weights. We will consider networks with only one node in the output layer since we focus on two-class problems in this paper.

The hidden layers are numbered in increasing order away from the output layer as shown in the diagram. No computation is performed by the input layer: it merely receives the input pattern and distributes the components to the last hidden layer. We shall use the term *downstream* to mean "towards the output layer".

The output from the network is *clamped* during training. If the target for a pattern of class 1 is $1 - \epsilon$ but the output is greater than $1 - \epsilon$, then the output is clamped to $1 - \epsilon$. Similarly for a pattern of class 2, if the target is td but the network output is *less* than ϵ, then the output is clamped to ϵ. The clamp is used to implement the modified penalty function suggested by Sontag and Sussmann [11]. They observe that backpropagation is less likely to get stuck in local minima when the output is clamped during training. Clamping is particularly desirable in classification problems because it makes no sense to say that an error has occurred when the network gives an output

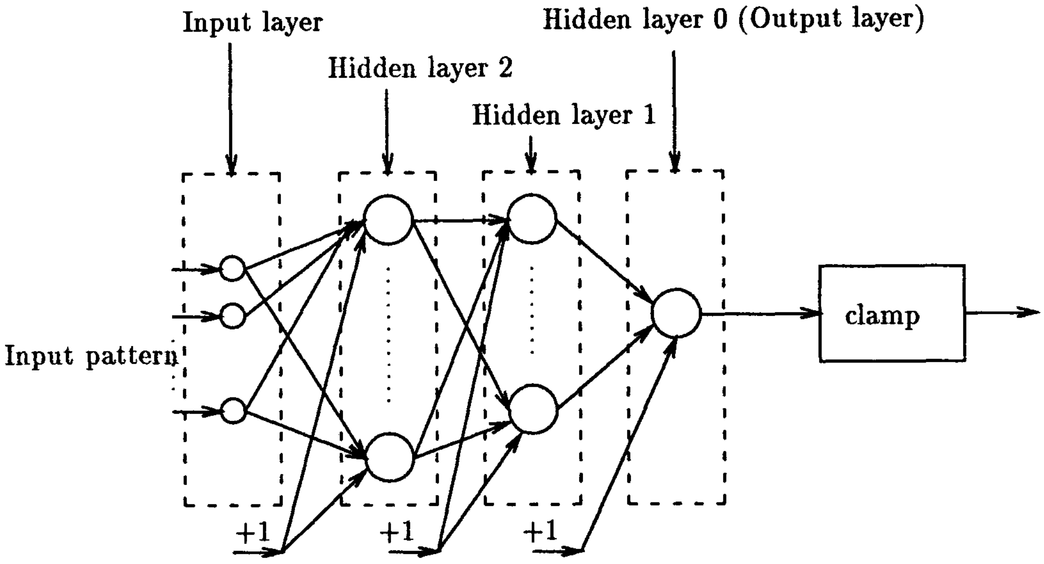

Figure 2: A multilayer feed-forward network for a two-class problem.

greater than $1-\epsilon$ when the target is $1-\epsilon$ (i.e., the network classifies samples with very low error).

Notation: To fix the notation, we consider a backpropagation network with one hidden layer (HL) shown in figure 3. There are $I+1$ nodes in the input layer for input patterns of length I; the additional node represents the bias, θ, in the function $\frac{1}{1+e^{-(\mathbf{w}\cdot\mathbf{x}+\theta)}}$ computed at each node. The HL contains $L+1$ nodes including a node for the bias term. Since we deal only with two-class problems in this paper, we assume that there is only one node in the output layer, which we call the output node.

The exemplars of class C_k form the set

$$T_k = \{(\mathbf{x}_j^{(k)}, t_j^{(k)}): \quad j = 1, \ldots, n_k\}: \quad k = 1, 2$$

The input vector for the jth exemplar of the kth class (i.e. the (j,k)th exemplar) is

$$\mathbf{x}_j^{(k)} = (x_{j,1}^{(k)}, \ldots, x_{j,I+1}^{(k)})$$

where $x_{j,I+1}^{(k)} \equiv 1$ and the target values are $t_j^{(1)} = 1-\epsilon$ and $t_j^{(2)} = \epsilon$. The training set T, for a two class problem is $T_1 \cup T_2$.

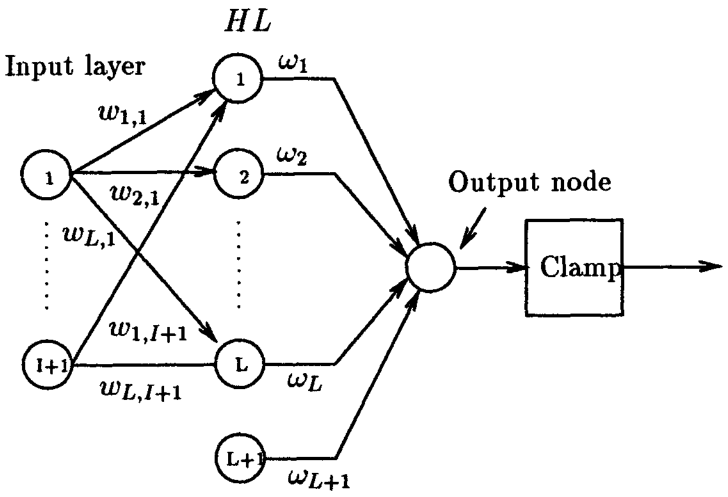

Figure 3: Notation for identifying nodes and weights in a network.

The outputs of HL can be collectively written as

$$\mathbf{y}_j^{(k)} = (y_{j,1}^{(k)}, \ldots, y_{j,L+1}^{(k)})$$

where $y_{j,L+1}^{(k)} \equiv 1$. The output of the network (i.e. that of the output node) is given by $z_j^{(k)}$.

In this network, the weight assigned to the link from the rth node of the input layer to the sth node of the HL is denoted by $w_{s,r}$. The weights on the links from the input layer to the sth node in HL are collectively denoted by

$$\mathbf{w}_{(s)} = (w_{s,1}, \ldots w_{s,I+1}).$$

We collectively refer to all weights between the input layer and HL by

$$\mathbf{w} = (\mathbf{w}_{(1)}, \ldots, \mathbf{w}_{(L)}).$$

The weight of the link from the sth node of the hidden layer to the output node is denoted by ω_s. All such (ω_s) weights are collectively denoted by $\boldsymbol{\omega}$, i.e.,

$$\boldsymbol{\omega} = (\omega_1, \ldots, \omega_{L+1}).$$

Finally, all weights of the network are denoted by $\mathbf{W}$:

$$\mathbf{W} = (\boldsymbol{\omega} , \mathbf{w}).$$

Gradients: We express the net error for the entire training set, $E(\mathbf{W})$, in terms of the net errors for subsets T_1 and T_2 denoted by $E_1(\mathbf{W})$ and $E_2(\mathbf{W})$ respectively:

$$\begin{aligned} E(\mathbf{W}) &= E_1(\mathbf{W}) + E_2(\mathbf{W}) \qquad (1) \\ E_k(\mathbf{W}) &= \sum_{j=1}^{n_k} f(t_j^{(k)}, z_j^{(k)}) \text{ for } k = 1, 2 \end{aligned}$$

Where f is the penalty function:

$$f(t_j^i, o_j^i) = \begin{cases} (t_j^i - o_j^i)^2 & \text{if } ((t_j^i = 1 - \epsilon) \wedge (o_j^i < t_j^i)) \vee ((t_j^i = \epsilon) \wedge (o_j^i > t_j^i)) \\ 0 & \text{otherwise} \end{cases}$$

The gradient, $\nabla E(\mathbf{W})$, of the error function $E(\mathbf{W})$ can be expressed in terms of the gradients for $E_1(\mathbf{W})$ and $E_2(\mathbf{W})$:

$$\nabla E(\mathbf{W}) = \nabla E_1(\mathbf{W}) + \nabla E_2(\mathbf{W}) \qquad (2)$$

In each iteration of the standard backpropagation algorithm, we compute $\nabla E(\mathbf{W})$, the gradient vector of the error surface. Since net error decreases most rapidly in the direction exactly *opposite* to that pointed to by the gradient vector, we move the weights in the direction of $-\nabla E(\mathbf{W})$.

Backpropagation is summarized in the following equation:

$$\mathbf{W}(m+1) = \mathbf{W}(m) - \lambda \nabla E(\mathbf{W}(m))$$

where $\mathbf{W}(m)$ is the weights of the network at the beginning of the mth iteration, and λ, a positive constant, is the learning rate. Some modifications to backpropagation vary the learning rate during the training process [2] [13].

A vector $\mathbf{v}$ is said to point in a *downhill* direction for $E(\mathbf{W})$ if

$$\mathbf{v} \cdot (-\nabla E(\mathbf{W})) > 0$$

In other words, the angle between $\mathbf{v}$ and $-\nabla E(\mathbf{W})$ is less than 90°.

Weight change computation: The hidden node outputs $\{y_{j,1}^{(k)}, \ldots, y_{j,L}^{(k)}\}$ are computed as follows:

$$y_{j,s}^{(k)} = \frac{e^{\mathbf{x}_j^{(k)} \cdot \mathbf{w}_{(s)}}}{1 + e^{\mathbf{x}_j^{(k)} \cdot \mathbf{w}_{(s)}}}, \quad \text{for } s = 1, \ldots, L, \;\; k = 1, 2, \text{ and } j = 1, \ldots, n_k.$$

The network output $z_j^{(k)}$ is obtained with the following equation:

$$z_j^{(k)} = \frac{e^{\mathbf{y}_j^{(k)}.\boldsymbol{\omega}}}{1 + e^{\mathbf{y}_j^{(k)}.\boldsymbol{\omega}}}, \text{ for } k = 1, 2, \text{ and } j = 1, \ldots, n_k.$$

Due to the nature of the sigmoid function, $\frac{1}{1+e^{-u}}$, the values $y_{j,1}^{(k)}, \ldots, y_{j,L}^{(k)}$ and $z_j^{(k)}$ are always positive and in the range (0, 1).

All weight changes consist of a product of the error signal for a node and the output of another node. The weight change in ω_s due to the (j,k)th exemplar is given by:

$$\begin{aligned} \Delta\omega_s^{(j,k)} &= \lambda \times \text{Error signal of output node} \times \text{Output of } s\text{th node of } HL \\ \Delta\omega_s^{(j,k)} &= \lambda \left((t_j^{(k)} - z_j^{(k)}) z_j^{(k)} (1 - z_j^{(k)}) \right) \left(y_{j,s}^{(k)} \right) \qquad (3) \end{aligned}$$

for $s = 1, \ldots, L+1$, $j = 1, \ldots, n_k$ and $k = 1, 2$. This corresponds to $\lambda \frac{\partial E^{(j,k)}(\mathbf{W})}{\partial \omega_s}$. Similarly, the weight change in $w_{r,s}$ due to the (j,k)th exemplar is given by:

$$\begin{aligned} \Delta w_{r,s}^{(j,k)} &= \lambda \times \text{Error signal of } r\text{th node of } HL \times \text{Output of } s\text{th input node} \\ \Delta w_{r,s}^{(j,k)} &= \lambda \left((t_j^{(k)} - z_j^{(k)}) z_j^{(k)} (1 - z_j^{(k)}) \omega_r \right) \left(x_{j,s}^{(k)} \right) \qquad (4) \end{aligned}$$

for $r = 1, \ldots, L$, $s = 1, \ldots, I+1$, $j = 1, \ldots, n_k$ and $k = 1, 2$.

The contribution of the (j,k)th exemplar to the gradient vector, $\nabla E^{(j,k)}(\mathbf{W})$ is:

$$\nabla E^{(j,k)}(\mathbf{W}) = (\Delta\boldsymbol{\omega}^{\;(j,k)}, \Delta\mathbf{w}^{(j,k)}) \qquad (5)$$

where $\Delta\boldsymbol{\omega}^{\;(j,k)} = (\Delta\omega_1^{(j,k)}, \ldots, \Delta\omega_{L+1}^{(j,k)})$ and similarly $\Delta\mathbf{w}^{(j,k)} = (\Delta\mathbf{w}_1^{(j,k)}, \ldots, \Delta\mathbf{w}_L^{(j,k)})$.

Finally, the gradient vector $\nabla E_k(\mathbf{W})$ is defined as follows:

$$\nabla E_k(\mathbf{W}) \;=\; \sum_{j=1}^{n_k} \nabla E^{(j,k)}(\mathbf{W}) \quad k = 1, 2. \qquad (6)$$

2.2 Analysis of the standard BP

In this section, we present a mathematical analysis of the slow rate of convergence of net error.

Theorem 1 *If all inputs to a feed-forward network with one hidden layer are positive, then for every weight in the network, the weight change in the first iteration of backpropagation has the same sign for all exemplars of class C_1 and the opposite sign for all exemplars of class C_2.*

Proof: In equations 3 and 4, we find that the sign of the weight change for any weight in the network due to the (j, k)th exemplar depends only on the term $(t_j^{(k)} - z_j^{(k)})$ since the weights are the same for all exemplars and all $\mathbf{x}$, $\mathbf{y}$ and z have only positive values. Since $(t_j^{(k)} - z_j^{(k)})$ is non-negative for class C_1 and non-positive for class C_2, the result follows. □

Discussion: In the statement of theorem 1, we have assumed that all inputs to the network are positive.

The assumption of positive inputs is not restrictive since all inputs can be made positive by a simple translation. In most applications of backpropagation it is desirable to transform the inputs to belong to $[0,1]^I$ with a simple transformation. If this is not done, a single large input value often dominates the output through a sigmoid function, slowing down the rate of convergence of net error.

We have observed that error signals are attenuated as they travel backwards through the randomly initialized network in the first iteration. Hence the changes prescribed for the weights in the upstream hidden layers are very small.

Consequently, even in the case when inputs to the network are negative, the results of theorem 1 generally hold since the weight changes for the hidden layer weights are small compared to the weight changes in the output layer weights as discussed in the sequel.

We have also assumed that the network has only one hidden layer. Even in in networks with more than one hidden layer, we have observed that the expected magnitude of the error signals of the nodes in HL_1 are approximately the same. Therefore, in general, we expect that the signs of the weight changes in the second hidden layer will be different for exemplars of each class. This leads us to believe that theorem 1 will continue to hold for networks with more than one hidden layer. Experiments and numerical calculations have supported this observation.

Theorem 2 *Under the assumptions of theorem 1,*

$$\nabla E_1(\mathbf{W}) \cdot \nabla E_2(\mathbf{W}) < 0.$$

Proof: The dot product of $\nabla E_1(\mathbf{W})$ with $\nabla E_2(\mathbf{W})$ is:

$$\nabla E_1(\mathbf{W}) \cdot \nabla E_2(\mathbf{W}) = \sum_{s=1}^{L+1} \Delta\omega_s^{(1)} \Delta\omega_s^{(2)} + \sum_{r=1}^{I+1} \sum_{s=1}^{L} \Delta w_{r,s}^{(1)} \Delta w_{r,s}^{(2)}$$

From theorem 1, we find that in each pair, $(\Delta w_i^{(1)}, \Delta w_i^{(2)})$ and $(\Delta w_{r,s}^{(1)} \Delta w_{r,s}^{(2)})$, one of the terms is positive and the other term is negative. Hence the dot product is always negative. In geometrical terms, the angle between $\nabla E_1(\mathbf{W})$ and $\nabla E_2(\mathbf{W})$ is greater than 90°. □

Suppose $\mathcal{E}(\cdot)$ denotes the expectation with respect to weights $\mathbf{W}$, and $\mathcal{E}_{\boldsymbol{\omega}}\ (\cdot)$ denotes the conditional expectation with respect to $\boldsymbol{\omega}$ while $\mathbf{w}$ remains fixed.

Theorem 3 *The expected values of the squares of the lengths of the gradient vectors satisfy:*

$$\frac{\mathcal{E}\|\nabla E_1(\mathbf{W})\|^2}{\mathcal{E}\|\nabla E_2(\mathbf{W})\|^2} \approx \frac{n_1^2}{n_2^2}$$

Proof: In the following proof sketch, only the leading term of each expected value is considered. A more detailed proof is given in the appendix.

The square of the length of $\nabla E_k(\mathbf{W})$ is:

$$\|\nabla E_k(\mathbf{W})\|^2 = \sum_{s=1}^{L+1} (\Delta\omega_s^{(k)})^2 + \sum_{r=1}^{I+1} \sum_{s=1}^{L} (\Delta w_{s,r}^{(k)})^2.$$

From lemma 3.a of the appendix, we obtain

$$\begin{aligned} \mathcal{E}_{\boldsymbol{\omega}}\ (\Delta\omega_s^{(k)})^2 &= \sum_{j=1}^{n_k} \sum_{l=1}^{n_k} \mathcal{E}_{\boldsymbol{\omega}}\ (\Delta\omega_{j,s}^{(k)} \Delta\omega_{l,s}^{(k)}) \\ &\approx \left(\frac{2t_j^{(k)} - 1}{8} \right)^2 y_{j,s}^{(k)} y_{l,s}^{(k)} n_k^2, \end{aligned}$$

and from lemma 3.b, the expected value of $y_{j,s}^{(k)} y_{l,s}^{(k)}$ with respect to $\mathbf{w}$ is approximately $\frac{1}{4}$. Thus,

$$\mathcal{E}\left(\sum_{s=1}^{L+1} (\Delta\omega_s^{(k)})^2 \right) \approx \frac{(2t_j^{(k)} - 1)^2 (L+1)}{256} n_k^2.$$

The expected values of $(\Delta w_{s,r}^{(k)})^2$ are negligible (see lemma 3.a). Hence,

$$\mathcal{E}\left(\|\nabla E_k(\mathbf{W})\| \right)^2 \approx \frac{(2t_j^{(k)} - 1)^2 (L+1)}{256} n_k^2; \quad k = 1, 2.$$

Since $\frac{(2t_j^{(k)}-1)^2(L+1)}{256}$ is the same for both values of k, the desired result holds. □

Discussion In theorem 3, we have shown that the expected lengths of the gradient vectors $\nabla E_k(\mathbf{W})$ are proportional to the sizes of the training sets, n_k. This would imply that, in general, the length of the gradient vector of the dominant class (class 2) will be very large when $n_2 \gg n_1$ [1]. Typically observed $\nabla E_1(\mathbf{W})$ and $\nabla E_2(\mathbf{W})$ are depicted in figure 4.

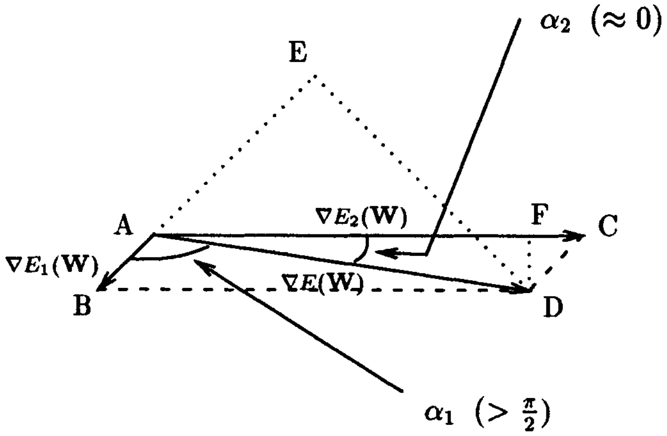

Figure 4: Relationship between gradient vectors $\nabla E_1(\mathbf{W})$, $\nabla E_2(\mathbf{W})$ and $\nabla E(\mathbf{W})$.

The length of vector $\nabla E_2(\mathbf{W})$ is much larger than the length of $\nabla E_1(\mathbf{W})$, therefore $\nabla E(\mathbf{W}) \approx \nabla E_2(\mathbf{W})$.

Theorem 4 (Ostrowski) [7] *If* v *is a unit vector, then there exists a constant* λ *such that*
i) $\mathbf{W}' = \mathbf{W} + \lambda \mathrm{v}$ *and*
ii)$E(\mathbf{W}') < E(\mathbf{W})$
if and only if v *is a downhill vector for* $E(\mathbf{W})$.

Standard backpropagation tells us that AD (refer to figure 4) is the best direction to follow to reduce $E(\mathbf{W})$. However, the effect of moving in the direction of AD can

[1] Since all weights are uniformly distributed, the variance of the square of length will be small and by Chebyshev's inequality [1] the stated result will hold.

be measured in terms of AF for $E_1(\mathbf{W})$ and AE for $E_2(\mathbf{W})$. But AE points in the uphill direction of $E_1(\mathbf{W})$. Consequently by Ostrowski's theorem, if $\mathbf{W}$ is changed in the direction AD then $E_2(\mathbf{W})$ will decrease significantly and $E_1(\mathbf{W})$ will increase significantly. The magnitudes of the changes are proportional to the lengths of AF and AE.

It has been observed that rate of convergence of backpropagation is often very low when the output error is high. The reason for this behavior can be explained easily by analysis of the error signal for the output node:

$$(t-z)z(1-z)$$

A large error ($|t-z| \approx 1$) implies that either $z \approx 0$ or $z \approx 1.0$. In either case, one of the last two terms in the above expression will have a low value and due to this reason, the amount of change in weights will be small.

In summary, we have observed that if $\mathbf{W}'$ denotes the *new* weight vector obtained by changing the weight by moving in the direction of AD, $E_1(\mathbf{W}')$ the net error of the subordinate class and $E_2(\mathbf{W}')$, the net error of the dominant class, then,

1. After the first iteration, $E_1(\mathbf{W}')$ is high and $E_2(\mathbf{W}')$ is low.
2. Since $E_1(\mathbf{W}')$ is high, the error signals from the output node will have a small magnitude and rate of convergence of error is slow. Likewise, since $E_2(\mathbf{W}')$ is small, the rate of change of $E_2(\mathbf{W}')$ will be very slow.
3. Consequently, standard BP will make a major improvement in reducing the net error in the first step and will likely get stuck in a slow mode of error reduction.

In addition to the magnitude of the gradient vector, the actual weight change for each weight in the network also depends on the learning rate λ. Since we use a fixed learning rate in backpropagation, the usual approach is to find, by trial and error, the largest value of λ which does not cause oscillation. In the context of imbalanced training sets, however, we have found that increasing the learning rate does not necessarily increase the rate of convergence of net error.

The reason for this behavior lies in the increase in $E_1(\mathbf{W})$ which occurs in the first iteration. By increasing λ, we also increase the value of $E_1(\mathbf{W})$ after the first iteration. As we have noted previously, this causes the rate of convergence of $E_1(\mathbf{W})$ to decrease. Experimental results are summarized in figures 9 and 10.

3 Modified backpropagation

From the results in section 2.2, it is clear that $-\nabla E(\mathbf{W})$ does not always point in the best direction to minimize error for *both* classes in a two-class problem. The main feature of our modification is to compute a descent vector, $\mathbf{v}$, which points in a downhill direction for both classes i.e. $\mathbf{v}$ satisfies

$$-\mathbf{v} \cdot \nabla E_k(\mathbf{W}) \;<\; 0, \text{ for } k = 1, 2 \tag{7}$$

and takes the place of the gradient vector in the backpropagation algorithm:

$$\mathbf{W}(k+1) = \mathbf{W}(k) - \lambda \mathbf{v}.$$

We propose to set the direction of $\mathbf{v}$ so that $\mathbf{v}$ bisects the angle between $-\nabla E_1(\mathbf{W})$ and $-\nabla E_2(\mathbf{W})$:

$$\frac{-\nabla E_1(\mathbf{W})}{\| - \nabla E_1(\mathbf{W})\|} \cdot \mathbf{v} = \frac{-\nabla E_2(\mathbf{W})}{\| - \nabla E_2(\mathbf{W})\|} \cdot \mathbf{v}$$

(See figure 5). Unless the angle between $-\nabla E_1(\mathbf{W})$ and $-\nabla E_2(\mathbf{W})$ is exactly 180°, we are always guaranteed to find a downhill direction for both E_1 and E_2.

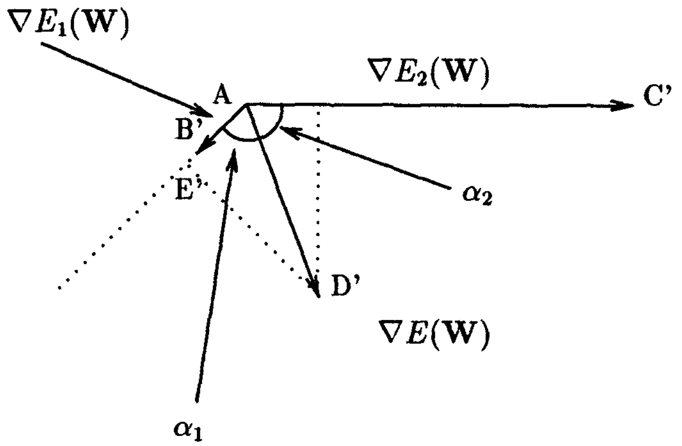

Figure 5: Direction of gradient vectors in modified algorithm.

The above method is not the only choice for computing a suitable descent vector. A descent vector can be any vector that makes an angle less than $\frac{\pi}{2}$ with both AB and AC. The main reason for using the bisector is that it is simple to compute and is guaranteed to point in a downhill direction for both classes.

The proposed algorithm does not suffer from the deficiencies of the standard backpropagation stated previously. Both E_1 and E_2 continue to follow the downhill path at each iteration rapidly. Therefore the proposed algorithm will, in general, be faster. Empirical verification of the above observations is presented for three examples in section 4.

Magnitude of proposed descent vector: We have investigated two schemes for computing the magnitude of the descent vector $\mathbf{v}$:

1. Proportional to the means of the magnitudes of $\nabla E_1(\mathbf{W})$ and $\nabla E_2(\mathbf{W})$:

$$\|\mathbf{v}\| = \frac{1}{2}(\|\nabla E_1(\mathbf{W})\| + \|\nabla E_2(\mathbf{W})\|) \tag{8}$$

2. Proportional to the same magnitude as would be computed by standard backpropagation for $E(\mathbf{W})$:

$$\|\mathbf{v}\| = \|\nabla E_1(\mathbf{W}) + \nabla E_2(\mathbf{W})\| \tag{9}$$

We shall refer to these formulae as method 1 and method 2 respectively. Our experience with examples, described in the next section, indicates that net error converges somewhat faster with method 2.

4 Numerical results

In this section, we compare the performance of modified backpropagation with standard backpropagation for three different classification problems. We first present some details of the three classification problems and then summarize the results in figure 8.

4.1 Example 1 (Grid)

The patterns in the training set are two dimensional and are uniformly randomly generated, with no overlap between the classes. The patterns occur in 25 clusters as

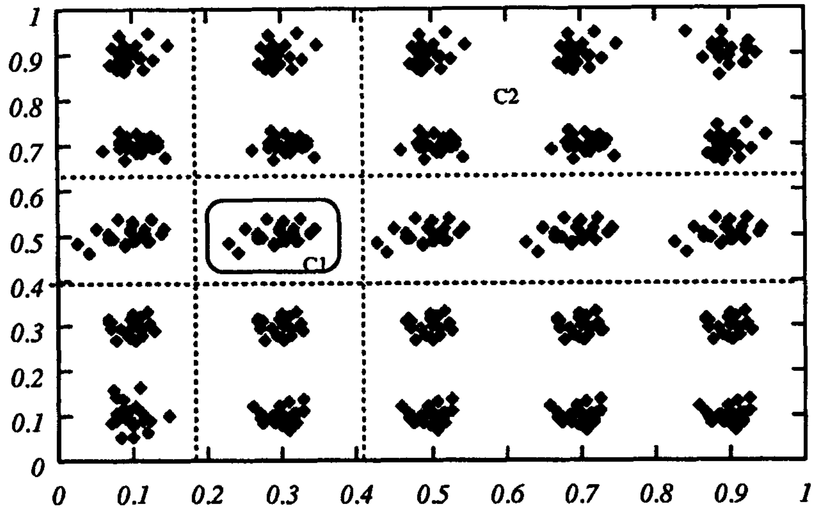

Figure 6: Location of classes $\mathcal{C}_1$ and $\mathcal{C}_2$ for example 1. Dotted lines show that four hidden units are sufficient for this problem.

shown in figure 6. Patterns that belong to the subordinate class, $\mathcal{C}_1$, lie within an interior cluster. Class $\mathcal{C}_2$, the dominant class, consists of the points in the remaining 24 clusters. Thus, $n_1 = 25$ and $n_2 = 600$.

A single-output network with one hidden layer containing 4 nodes was used for this problem. The target value for exemplars in class $\mathcal{C}_1$ was 0.9 while the target for class $\mathcal{C}_2$ was 0.1. We used 4 nodes in the hidden layer since 4 decision surfaces are required to separate patterns of class $\mathcal{C}_1$ from class $\mathcal{C}_2$ (shown with dotted lines in figure 6). A learning rate of $\lambda = 0.01$ was used for all runs. In each experiment, training was started from the same randomly generated set of initial weights for the standard as well as the modified backpropagation algorithms.

The average error per exemplar for classes $\mathcal{C}_1$ and $\mathcal{C}_2$ during a typical training run is shown in figure 7 for both standard and modified backpropagation.

4.2 Example 2 (Speech)

The data used in this example is for a speech recognition problem and was obtained from the UCI repository of machine learning databases and domain theories. The input patterns are 10 element floating point vectors representing vowel sounds which belong to one of 11 classes. There are 45 exemplars for each class. We have derived

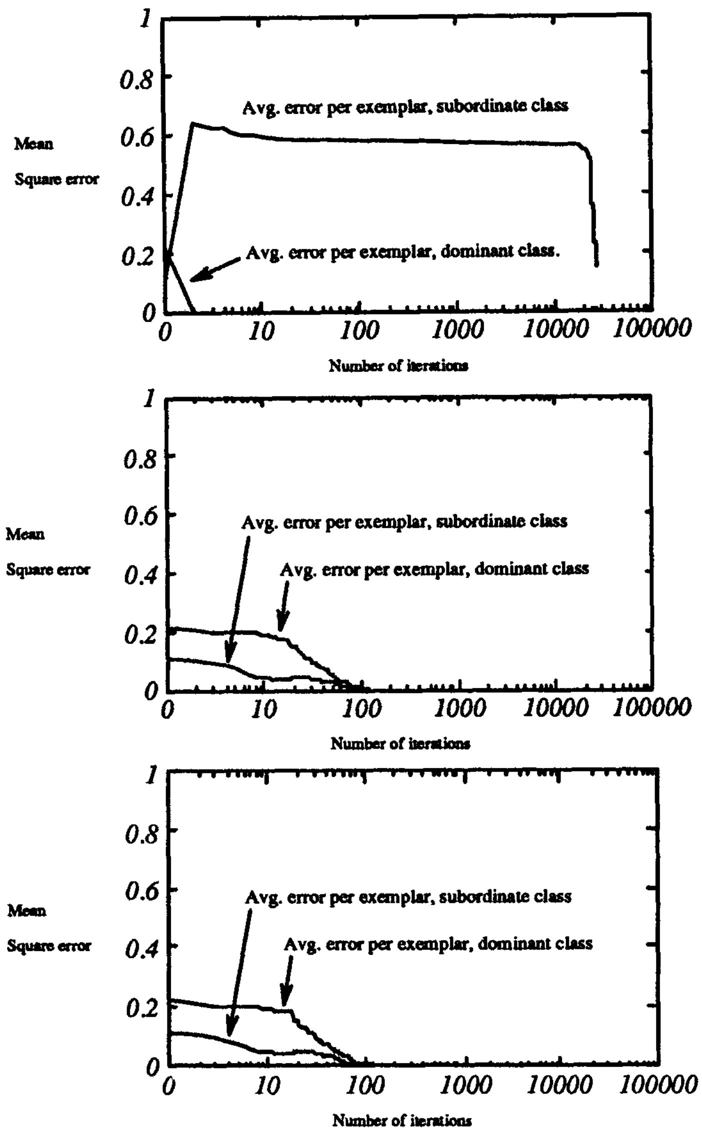

Figure 7: Average error per exemplar of dominant (C_1 and subordinate C_2 classes during training (Plots shown are for experiment 1 of example 1). *Top*: Standard backpropagation. *Middle*: Modified backpropagation, method1. *Bottom*: Modified backpropagation, method2.

a two-class problem from this 11-class problem: Class $\mathcal{C}_1$, the subordinate class, contains exemplars for the vowel sound in "hid". Class $\mathcal{C}_2$, the dominant class, contains exemplars for the remaining 10 vowel sounds. Thus, we have $n_1 = 45$ and $n_2 = 450$. As in the previous examples, the patterns were translated and scaled in order to lie within $[0,1]^{10}$.

A single-output net with one hidden layer of 20 nodes was trained for this problem with the learning rate $\lambda = 0.01$. In each experiment, the same set of random initial weights were used for both standard backpropagation as well as the modified algorithm. Training was stopped when only three exemplars remained misclassified.

4.3 Example 3 (Fisher's Iris data)

In this example, we analyzed the well-known Fisher's Iris data set [4]. Although this is actually a three class problem with 50 exemplars for each class, we have converted it to a two class problem as follows: Class $\mathcal{C}_1$, the subordinate class, consists of the exemplars for *Iris Versicolor.* Class $\mathcal{C}_2$, the dominant class, contains exemplars for *Iris Setosa* and *Iris Virginica.* Thus, $n_1 = 50$ and $n_2 = 100$. The original patterns were translated and scaled so as to lie within $[0,1]^4$. We used a network with one hidden layer of 4 nodes and the learning rate λ was set to be 0.05. In each experiment, the same set of random initial weights were used for both standard backpropagation as well as modified backpropagation. Training was stopped when only two exemplars remained misclassified.

4.4 Summary of results

The results of the three experiments are shown in figure 8. In general, we find that method 2 is faster than method 1 and both are considerably faster than standard backpropagation. The speedup obtained with the modified backpropagation appears to be greatest for problems with highly imbalanced training sets, but even if the imbalance ratio is only 2, as in the case of example 3, the average speedup is greater than 5.

Example	Experiment	Standard BP	Modified BP		Speedup	
			Method 1	Method 2	Method 1	Method 2
1	1	43755	147	116	297.6	377.2
	2	19532	313	869	62.4	22.5
	3	23376	252	172	92.8	135.9
	4	22340	388	232	57.6	96.3
	5	21130	337	175	62.7	120.7
2	1	2280	170	108	13.4	21.1
	2	2210	237	116	9.3	19.1
	3	2340	282	383	8.3	6.1
	4	1910	277	124	6.9	15.4
	5	1960	197	100	9.9	19.6
3	1	1500	215	163	6.9	9.2
	2	1390	501	363	2.8	3.8
	3	1410	110	466	12.8	3.0
	4	1470	222	260	6.6	5.7
	5	1420	530	273	2.7	5.2

Figure 8: The number of iterations required for the number of misclassifications to decrease to acceptable level.

λ	Num. Iter.	$\frac{E_1(\mathbf{W})}{q_1}$ after first iter.
0.01	43755	0.486510
0.02	18318	0.726782
0.03	11160	0.792245
0.04	8674	0.806312
0.05	5991	0.809228
0.055	5416	0.809645
0.06	5910	0.809836
0.065	7939	0.809924
0.07	12588	0.809965

Figure 9: The effect of varying λ for example 1 experiment 1 using standard backpropagation. Column 2: The number of iterations needed for the all exemplars to be correctly classified. Column 3: Mean square error for class C_1 after the first iteration.

5 Comparison of execution times

The standard backpropagation algorithm consists of two steps:

1. Evaluation of $\nabla E(\mathbf{W})$.

2. Weight adjustment $\mathbf{W}' = \mathbf{W} + \lambda \nabla E(\mathbf{W})$.

In the modified backpropagation also, two gradient vectors $\nabla E_1(\mathbf{W})$ and $\nabla E_2(\mathbf{W})$ are computed but the time to compute these two vectors is exactly equal to the amount of time needed to compute $E(\mathbf{W})$. In this step, the only difference between the standard and proposed backpropagation is that we need to store two gradient vectors.

The only additional computation in the proposed backpropagation is in evaluating the descent vector with equation 7. The additional overhead for computing the descent vector in the proposed algorithm is negligible compared to the time needed to compute the gradient vectors. Since our algorithm generally requires far fewer iterations for the error to converge, we achieve a good speedup in run times.

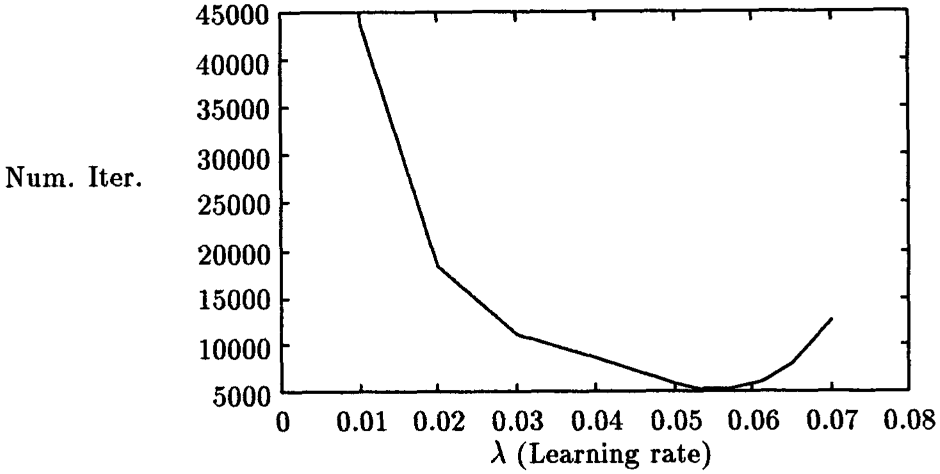

Figure 10: The influence of λ on the number of iterations needed to correctly classify all exemplars in example 1 experiment 1 using standard backpropagation.

In example 2 (speech recognition), the actual time taken by standard backpropagation is:

$$\text{Number of iterations} \times 495 \times 8.2 \ \ \text{milliseconds}$$

on a SUN SPARCserver 490, whereas the time taken by the modified backpropagation algorithm is

$$\text{Number of iterations} \times ((495 \times 8.2) + 2.2) \ \ \text{milliseconds.}$$

Thus in the proposed algorithm, it takes only 2.2 milliseconds per iteration to compute the descent vector which is negligible compared to the time to compute the gradient vectors.

6 Concluding remarks

In this paper, we have analyzed the reason for low rates of convergence of backpropagation for two class problems with imbalanced training sets for two-class problems. We then propose a modified version of the standard backpropagation algorithm which is significantly faster for such problems.

We have observed that although the net error of the dominant class decreases in the first iteration of standard backpropagation, the net error of the subordinate class actually *increases* significantly. The subsequent rate of decrease of net error of the subordinate class is very low. We show that this phenomenon occurs because the gradient vector computed by standard backpropagation for a randomly initialized network points in a downhill direction only for the dominant class.

The main feature of our modification to standard backpropagation is that we compute a descent vector which points in a downhill direction for both classes. Hence, net errors for both the dominant and subordinate classes are decreased by moving the weights in the direction of the descent vector.

We have compared the performance of standard and modified backpropagation for three two-class problems with varying degrees of imbalance in their training sets. The speedup obtained with modified backpropagation appears to be greatest for problems with highly imbalanced training sets, but even if the imbalance ratio is low, as in example 3 (Fisher's Iris data), the average speedup is greater than 5.

We plan to extend our results to multiclass problems as well. One difficulty that we have often encountered in multiclass problems is that even when the average error per exemplar is small, the probability of misclassification for one or more classes is very high. Another difficulty that we have observed with multiclass problems is the extremely low rate of convergence of error. We are currently trying to explain these phenomena in a manner similar to that described in this paper.

References

[1] Hogg, R. V., and Craig, A. T., "Introduction to mathematical statistics", The Macmillan Company, 1971.

[2] Jacobs, R. A., "Increased rates of convergence through learning rate adaptation", *Neural Networks*, Vol. 1, 1988, pp 295-307.

[3] Jacobs, R. A., Jordan, M. I., and Barto, A. G., "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks", COINS Technical Report 90-27, University of Massachusetts, 1990.

[4] James, M. "Classification Algorithms", John Wiley and Sons, 1985.

[5] Kohonen, T., Barna, G., and Chrisley, R. "Statistical Pattern Recognition with Neural Networks: Benchmarking Studies", *Proceedings of the International Conference on Neural Networks*, 1988, Vol-I, pp 61-68.

[6] Kowalik, J., and Osborne, M. R., "Methods for unconstrained optimization problems", American Elsevier Publishing Company Inc., 1968.

[7] Ostrowski, A. M., "Solution of equations in Euclidean and Banach Spaces", Academic Press, 1973.

[8] Pierre, D. A., "Optimization theory with applications", pg 297, John Wiley and Sons, Inc., 1969.

[9] Plaut, D. C., and Hinton, G. E., "Learning sets of filters using back-propagation", *Computer Speech and Language"*, Vol. 2, 1987, pp 35-61.

[10] Rumelhart, D. E., and McClelland, J. L. "Parallel Distributed Processing, Volume 1", MIT Press, 1987.

[11] Sontag, E. D., and Sussmann, H. J. "Backpropagation separates when perceptrons do", *Proceedings of the International Conference on Neural Networks*, 1988, Vol-I, pp 639-642.

[12] Waymaere, N. and Martens, J-P., "A fast and robust learning algorithm for feedforward neural networks", *Neural networks*, Vol. 4, No. 3, 1991, pp 361-370.

[13] Vogl, T. P., Mangis, J. K., Rigler, A. K., Zink, W. T., and Alkon, D. L., "Accelerating the Convergence of the Back-Propagation Method", *Biological Cybernetics*, Vol. 59, 1988, pp 257-263.

A Appendix

Since we shall repeatedly encounter complicated functions of random variables, we follow the procedure outlined below to find an approximation of their expected values.

Let $g(u)$ be some function of random variable u. Suppose we wish to obtain $\mathcal{E}(g(u))$ where $\mathcal{E}(\cdot)$ denotes the expected value. Then, using the Taylor's series expansion of $g(u)$ with respect to u upto three terms, about $\mathcal{E}u = \mu$, we get,

$$\begin{aligned}
\mathcal{E}(g(u)) &\approx \mathcal{E}\left(g(\mu) + g'(\mu)\cdot(u-\mu) + \frac{1}{2}(u-\mu)\cdot g''(\mu)\cdot(u-\mu)\right) \\
&= g(\mu) + \frac{1}{2}g''(\mu)\,\mathcal{E}\left((u-\mu)^2\right) \quad \text{for scalar } u \\
&= g(\mu) + \frac{1}{2}\sum_i\sum_j g''_{ij}(\mu)\,\mathcal{E}((u_i-\mu_i)(u_j-\mu_j)) \quad \text{for vector } u
\end{aligned}$$

$g(\mu)$, the leading term of the right hand side approximates the expected value of $g(u)$ when expansion is considered only upto the first two terms.

A.1 Expected values of functions of weight changes

Lemma 3.a

$$\begin{aligned}
\mathcal{E}_{\omega}\ \left[\Delta\omega_s^{(j,k)}\Delta\omega_s^{(l,k)}\right] &= y_{j,s}^{(k)}y_{l,s}^{(k)}\left\{\frac{(2t^{(k)}-1)^2}{64}\right. \\
&\qquad \left. -\frac{1}{64\times 12}\sum_{i=1}^{L+1}\left[(2t^{(k)}-1)^2\left((y_{j,i}^{(k)})^2+(y_{l,i}^{(k)})^2\right) - y_{l,i}^{(k)}y_{j,i}^{(k)}\right]\right\} \\
\mathcal{E}_{\omega}\ \left[\Delta w_{s,r}^{(j,k)}\Delta w_{s,r}^{(l,k)}\right] &= \frac{x_{j,r}^{(k)}x_{l,r}^{(k)}}{192}(2t^{(k)}-1)^2
\end{aligned}$$

Proof: To establish (3.a), we follow the procedure outlined previously and in addition, we use $\mathcal{E}(\omega\ ) = \mathbf{0}$, $\mathcal{E}(\omega_s^2) = \frac{1}{3}$ and $\mathcal{E}(\omega_s\omega_{s'}) = 0$. These equalities holds because the ω_s are stochastically independent random variables uniformly distributed between [-1, 1]. The $y_{j,s}^{(k)}$ are constants when expectation is taken with respect to ω .

If we confine our attention only to the leading terms,

$$\begin{aligned}
\mathcal{E}_{\boldsymbol{\omega}}\ \left[\Delta\omega_s^{(j,k)}\Delta\omega_s^{(l,k)}\right] &= y_{j,s}^{(k)}y_{l,s}^{(k)}\left\{\frac{(2t^{(k)}-1)^2}{64}\right\} \\
\mathcal{E}_{\boldsymbol{\omega}}\ \left[\Delta w_{s,r}^{(j,k)}\Delta w_{s,r}^{(l,k)}\right] &= 0
\end{aligned}$$

□

The following lemma helps obtain the expectations of the above values with respect to the hidden layer weights:

Lemma 3.b

$$\begin{aligned}
\mathcal{E}_{\mathbf{w}}(y_{l,s}^{(k)}y_{j,s}^{(k)}) &= \frac{1}{4}+\frac{\mathbf{x}_j^{(k)}\cdot\mathbf{x}_l^{(k)}}{48} \\
\mathcal{E}_{\mathbf{w}}((y_{l,s}^{(k)})^2(y_{j,s}^{(k)})^2) &= \frac{1}{16}+\frac{1}{6\times 32}\left(\|\mathbf{x}_j^{(k)}\|^2+2\mathbf{x}_j^{(k)}\cdot\mathbf{x}_l^{(k)}+\|\mathbf{x}_l^{(k)}\|^2\right) \\
\mathcal{E}_{\mathbf{w}}((y_{l,s}^{(k)})^3y_{j,s}^{(k)}) &= \frac{1}{16}+\frac{1}{64}\left(\|\mathbf{x}_j^{(k)}\|^2+\mathbf{x}_j^{(k)}\cdot\mathbf{x}_l^{(k)}\right)
\end{aligned}$$

Proof: The proof for lemma 3.b is similar to that of the previous lemma including $\mathcal{E}(\mathbf{w}_s)=\mathbf{0}$, $\mathcal{E}(w_{s,r}^2)=\frac{1}{3}$, and $\mathcal{E}(w_{s,r}w_{s,r'})=0$ for similar reasons. Once again, the first terms on the r.h.s. of each expression gives the leading term; e.g. $\mathcal{E}_{\mathbf{w}}(y_{l,s}^{(k)}y_{j,s}^{(k)})=\frac{1}{4}$. □

Theorem 3: *The ratio of the expected square length of the gradient vector satisfies*

$$\frac{\mathcal{E}(\|\nabla E_1(\mathbf{w})\|^2)}{\mathcal{E}(\|\nabla E_2(\mathbf{w})\|^2)}\approx\frac{n_1^2A_1}{n_2^2A_2}$$

where

$$A_k=\frac{1}{64}\left\{C_1^{(k)}+C_2^{(k)}\|\overline{\mathbf{x}^{(k)}}\|^2+C_3\tau+C_4\tau^2+C_5\Theta\right\}$$

and

$$\begin{aligned}
\overline{\mathbf{x}^{(\mathbf{k})}} &= \frac{1}{n_k}\sum_{j=1}^{n_k}\mathbf{x}_j^{(k)} \\
\tau &= \frac{1}{n_k}\sum_{j=1}^{n_k}\|\mathbf{x}_j^{(k)}\|^2
\end{aligned}$$

$$
\begin{aligned}
\Omega &= \tau^2 \\
\Theta &= \frac{1}{n_k}\left(\sum_{j=1}^{n_k} \|\mathbf{x}_j\|^2 \mathbf{x}_j^{(k)}\right)\cdot \overline{\mathbf{x}^{(\mathbf{k})}} \\
C_1 &= \left\{\frac{(2t^{(k)}-1)^2}{96}(80+12L-L^2)+\frac{(L+4)^2}{192}\right\} \\
C_2 &= \left\{\frac{(2t^{(k)}-1)^2}{48}(L+16)-\frac{((2t^{(k)}-1)^2+1)}{1152}(L^2+4L)\right\} \\
C_3 &= \left\{-\frac{(2t^{(k)}-1)^2}{1152}(L^2+6L)+\frac{L}{1152}\right\} \\
C_4 &= \left\{\frac{L(L-1)}{27648}\right\} \\
C_5 &= \left\{-(2t^{(k)}-1)^2\frac{L(L-1)}{27648}\right\}
\end{aligned}
$$

Proof: Let us consider $\mathcal{E}(\|\nabla E_k(\mathbf{W})\|^2)$.

$$
\begin{aligned}
\|\nabla E_k(\mathbf{W})\|^2 &= (\nabla\boldsymbol{\omega}^{\ (k)}, \nabla\mathbf{w}^{(k)})\cdot(\nabla\boldsymbol{\omega}^{\ (k)}, \nabla\mathbf{w}^{(k)}) \\
&= \sum_{s=1}^{L+1}(\Delta\omega_s^{(k)})^2 + \sum_{r=1}^{I+1}\sum_{s=1}^{L}(\Delta w_{r,s}^{(k)})^2
\end{aligned}
$$

where $\Delta\omega_s^{(k)} = \sum_{j=1}^{n_k}\Delta\omega_s^{(j,k)}$ and $\Delta w_{r,s}^{(k)} = \sum_{j=1}^{n_k}\Delta w_{r,s}^{(j,k)}$ hence,

$$
\|\nabla E_k(\mathbf{W})\|^2 = \left[\sum_{r=1}^{L+1}\sum_{j=1}^{n_k}\sum_{l=1}^{n_k}\Delta\omega_s^{(j,k)}\Delta\omega_s^{(l,k)}\right] + \left[\sum_{r=1}^{I+1}\sum_{s=1}^{L}\sum_{j=1}^{n_k}\sum_{l=1}^{n_k}\Delta w_{r,s}^{(j,k)}\Delta w_{r,s}^{(l,k)}\right] \quad (10)
$$

We first take the conditional expectation of each term over $\boldsymbol{\omega}$ keeping $\mathbf{w}$ fixed. These values are obtained from lemma 3.a. In the next step, we find the expectation of these values over $\mathbf{w}$ using lemma 3.b and substitution in equation (10) and a lengthy but straightforward simplification [2] gives:

$$
\mathcal{E}_{\mathbf{W}}(\|\nabla E_k(\mathbf{W})\|^2) \approx \frac{n_k^2}{64}\left\{C_1^{(k)} + C_2^{(k)}\|\overline{\mathbf{x}^{(k)}}\|^2 + C_3\tau + C_4\tau^2 + C_5\Theta\right\}
$$

[2] The simplification was verified using the MACSYMA system.

Note: Coefficients C_4 and C_5 are negligible for two reasons:

(a) The denominators are very small;

(b) if all $x_{j,s}^{(k)}$ are between 0 and 1 then their higher order terms keep getting smaller and smaller as the power increases.

Hence $A_1 \approx A_2$.