IDIAP RESEARCH REPORT

A New Margin-Based Criterion for Efficient Gradient Descent

Ronan Collobert *   Samy Bengio *

IDIAP–RR 03-16

February 11, 2004

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

* IDIAP, CP 592, 1920 Martigny, Switzerland, {collober,bengio}@idiap.ch

IDIAP Research Report 03-16

A New Margin-Based Criterion for Efficient Gradient Descent

Ronan Collobert Samy Bengio

February 11, 2004

Abstract. During the last few decades, several papers were published about second-order optimization methods for gradient descent based learning algorithms. Unfortunately, these methods usually have a cost in time close to $O(n^3)$ per iteration, and $O(n^2)$ in space, where n is the number of parameters to optimize, which is intractable with large optimization systems usually found in real-life problems. Moreover, these methods are usually not easy to implement. Many enhancements have also been proposed in order to overcome these problems, but most of them still cost $O(n^2)$ in time per iteration. Instead of trying to solve a hard optimization problem using complex second-order tricks, we propose to modify the problem itself in order to optimize a simpler one, by simply changing the cost function used during training. Furthermore, we will argue that analyzing the Hessian resulting from the choice of various cost functions is very informative and could help in the design of new machine learning algorithms. For instance, we propose in this paper a version of the Support Vector Machines criterion applied to Multi Layer Perceptrons, which yields very good training and generalization performance in practice. Several empirical comparisons on two benchmark data sets are given to justify this approach.

1 Introduction

Optimization by gradient descent is widely used by various machine learning algorithms such as back-propagation of the error in Multi-Layer Perceptrons (MLPs) and Radial Basis Functions, as suggested by Bishop (1995). Unfortunately, empirical evidences show that results obtained after training a model by gradient descent are often highly variable. For some problems and some models, training ends up efficiently into a good local optimum, while in some cases, results are often very bad.

Hence, in the last few decades, several researchers proposed various enhancements to classical gradient descent algorithms (see for instance a collection of them in the paper of LeCun et al., 1998). As shown by Fletcher (1987) and Battiti (1992), most of these enhancements focus on variations of second-order optimization methods, and thus have to compute the inverse of the Hessian of the cost function, which is the second order derivative of the cost with respect to pairs of parameters. Therefore, the time complexity of the resulting algorithms grows in $O(n^3)$ per iteration, and in $O(n^2)$ in space, where n is the number of parameters to optimize. Thus these algorithms become useless for very large data sets and models (this is the case for the original Newton method, but also for Quasi-Newton, Gauss-Newton, Levenberg Marquardt and Natural Gradient (see Amari, 1998) methods). Some enhancements which compute iteratively the inverse of the Hessian have been proposed (such as the Broyden-Fletcher-Goldfarb-Shanno method (BFGS) described by Press et al., 1992), but most of them still have a cost in $O(n^2)$ per iteration. In the end, most of the time, people rely on simple stochastic gradient descent which has a cost $O(n)$ per iteration, and which in general outperforms most other methods on large problems (as suggested by LeCun et al., 1998), even though it still often suffers from slow convergence. Some improvements have been proposed: for instance, a modified version of the Levenberg Marquardt algorithm uses the diagonal of the Hessian of the cost function; the Stochastic Meta Descent method introduced by Schraudolph (2002) proposes a trick to compute iteratively the product of the Hessian and a vector efficiently. However, in practice, they usually don't outperform a well-tuned stochastic gradient descent method on large problems (see LeCun et al., 1998).

A possible cause of this slow convergence of gradient descent algorithms could lie in the definition of the cost function itself. The aim of this paper is to show the influence of the choice of the cost function on the resulting efficiency of gradient descent algorithms. One way to analyze this, which will be explained in Section 3, is to observe the behavior of the Hessian matrix of the cost function. Careful analysis of the Hessian can in fact lead to the design of new cost functions for which gradient descent algorithms are more efficient. Indeed, the paper will show links between two apparently different cost functions, namely the Cross-Entropy criterion and the Support Vector Machine Criterion applied to MLPs, which both result in very simple Hessians.

The organization of the paper is as follows: in Section 2 we briefly introduce MLPs, mixtures of experts, and notations used in this paper, followed in Section 3 by a study of the local behavior of well-known cost functions illustrated by experimental evidence supporting the existence of a relation between simple Hessian and efficient gradient descent. In Section 4 we go further by showing relations between the Cross-Entropy criterion used in MLPs and the margin maximized in SVMs. Exploiting the margin idea leads in fact to a simple proposition to improve the performance of MLPs, which is verified in practice on two benchmark data sets. A short conclusion then follows.

2 Multi-Layer Perceptrons and Mixtures of Experts

We consider a two-class classification problem: given a training set of T examples $(x_t, y_t)_{t=1\dots T}$ with $(x_t, y_t) \in \mathbb{R}^n \times \{-1, 1\}$ (where x_t is the input vector of the t^{th} example, and y_t represents the corresponding class), we would like to find a function $f(\cdot)$ such that

$$f(x_t) > 0 \text{ when } y_t = 1 \text{ and } f(x_t) < 0 \text{ when } y_t = -1 \quad \forall t\,. \tag{1}$$

Our interest is to study functions which can be trained by gradient descent techniques. We will focus on two well-known models: Multi-Layer Perceptrons and Mixtures of Experts.

2.1 Multi-Layer Perceptrons

A Multi-Layer Perceptron (MLP) is a machine learning model which has been successfully applied to many classification and regression problems. The term "layer" refers to the fact that it represents a function which could be decomposed into several layers and computed in a feed-forward manner. We consider here an MLP with one hidden layer and N hidden units:

$$f(x) = b + \sum_{n=1}^{N} \alpha_n \, h(w_n^T \, x) \tag{2}$$

where $x \in \mathbb{R}^d$ is an input vector, $w_n \in \mathbb{R}^d$ are the weights of the hidden layer,[1] $\alpha_n \in \mathbb{R}$ are the weights of the output layer, and b is the bias of the output layer. $h(\cdot)$ is a transfer function which is usually a hyperbolic tangent. Note that it has been shown by Hornik et al. (1989) that MLPs with one hidden layer and hyperbolic tangent transfer functions are universal approximators of real valued functions.[2]

2.2 Mixtures of Experts

Mixtures of Experts (MEs), first introduced by Jacobs et al. (1991), represent function $f(\cdot)$ as a combination of simpler functions which are called "experts". More formally, given a input example x, the following decomposition is built:

$$f(x) = \sum_{k=1}^{K} g_k(x) f_k(x) \quad \text{with} \; \sum_k g_k = 1 \text{ and } g_k \geq 0 \;\forall k \tag{3}$$

where $f_i(\cdot)$ is the output function for expert i, and $g(\cdot) = \{g_1(\cdot) \dots g_K(\cdot)\}$ is the gater, which provides a real weight for each expert, given an input x. In this paper, the experts and the gater will be represented by MLPs. Since MLPs are universal approximators, MEs are also universal approximators. The underlying idea of MEs is that if the function to approximate is complex but can be easily decomposed into several simpler functions, each acting on a different input subspace, then MEs should be more appropriate, and the training of such models should be easier.

2.3 Gradient Descent

We consider in this section functions $f_\theta(\cdot)$ that depend on a real vector of parameters θ. If we suppose that $f_\theta(\cdot)$ is differentiable with respect to the parameters θ, one efficient way known to find the parameters θ such that $f_\theta(\cdot)$ verifies the separation conditions shown in (1), is to select a reasonable cost function $C(f_\theta(x),\, y)$ and to search for a θ^* that minimizes the cost over the training set

$$\theta^* = \arg\min_\theta \frac{1}{T} \sum_{t=1}^{T} C(f_\theta(x_t),\, y_t) \tag{4}$$

using gradient descent. One of the most common cost functions (described in details by Bishop, 1995) used for classification is the mean-squared error (MSE) which is half the squared difference between the output of the MLP and the desired class:

$$C(f_\theta(x),\, y) = \frac{1}{2}(y - f_\theta(x))^2 \; .$$

It is relatively easy to show that if we have an infinite amount of data, the minimum of the MSE criterion is obtained when $f_\theta(x)$ is equal to the posterior probability $p(y|x)$, as demonstrated by Bishop

[1] To simplify the notation, we suppose here that the last coordinate of the vector x is 1, and thus the bias of unit n is represented by the last value of the vector w_n.

[2] This implies that given a finite number of training examples x_i and a target function g, there exists an MLP f that can approximate g as close as desired, for all x_i.

(1995). Thus, the use of the MSE criterion seems to be relevant for classification. However, looking at it from a likelihood perspective, minimizing the MSE criterion is equivalent to maximizing a likelihood under the hypothesis that y is generated from a smooth function with added Gaussian noise. Since y is a binary variable, a Gaussian model is not really appropriate, and some people prefer to consider y as coming from a Bernoulli distribution. Therefore, if we want to minimize the negative log-likelihood (NLL) over the training set

$$\text{NLL} = -\frac{1}{T}\log(\prod_t p(y_t|x_t)),$$

where $p(y|x)$ is estimated by $\frac{1}{1+\exp(-yf_\theta(x))}$, we then obtain

$$\text{NLL} = \sum_t \log\left(1 + \exp(-y_t f_\theta(x_t))\right)$$

which leads to

$$C(f_\theta(x),\, y) = \log(1 + \exp(-y f_\theta(x)))\ . \tag{5}$$

This criterion is often called "cross-entropy criterion" (CE), for several reasons explained by Bishop (1995). The MSE and CE criteria will be used to analyze empirically and theoretically the influence of the cost function on the efficiency of the resulting gradient descent algorithm.

3 Local Behavior of Cost Functions

In this section, we analyze the impact of the cost function on the minimization problem given in (4). Both an empirical evaluation on two benchmark datasets and a theoretical analysis based on a second order Taylor approximation are given.

3.1 Experimental Setup

All experiments shown in this paper have been performed using two reasonably large data sets. The first one is the *UCI Forest* data set.[3] We modified the 7-class classification problem into a binary classification problem where the goal was to separate class 2 (the most numerous) from the other 6 classes, which leads to a well-balanced problem. We used 100,000 examples for training, 10,000 for validation (to select the hyper-parameters of the models) and 50,000 for testing.

The second one is the *UCI Letter* data set.[4] As for the Forest data set, we modified the 26-class classification problem into a binary classification problem where the goal was to separate the first 13 classes against the other 13 ones. This leads again to a well-balanced problem. The Letter data set comes already divided into a training set of 16,000 examples and a test set of 4,000 examples. We further divided the training set into an effective training set of 14,000 examples and a validation set of 2,000 examples.

3.2 Preliminary Results

In Table 1 we present a preliminary comparison between an MLP trained using the MSE and CE criteria, as well as a mixture of MLP-experts (ME) trained using the MSE criterion. Note that for each model, the shown number of hidden units is optimal according to the validation set. Moreover, all other hyper-parameters, such as the learning rate, were also selected according to the validation set.

Regarding the MLP trained with MSE on the Forest data set, it appears that the training performance is statistically significantly worse (with 99% confidence[5]) than an MLP trained with the CE

[3] The Forest data set is available on the UCI website at the following address:
`ftp://ftp.ics.uci.edu/pub/machine-learning-databases/covtype`

[4] The Letter data set is available on the UCI website at the following address:
`ftp://ftp.ics.uci.edu/pub/machine-learning-databases/letter-recognition`

[5] With a standard proportion test, assuming a binomial distribution for the targets, and using a normal approximation.

Forest			
Model	Cost Function	Train Err. (%)	Test Err. (%)
MLP (200HU)	MSE	13.7	13.9
MLP (500HU)	CE	10.7	11.1
ME (25HU per expert, 10 experts, 100HU for the gater)	MSE	9.4	9.8
Letter			
Model	Cost Function	Train Err. (%)	Test Err. (%)
MLP (500HU)	MSE	1.7	3.6
MLP (500HU)	CE	0.4	3.6
ME (50HU per expert, 20 experts, 25HU for the gater)	MSE	0.4	3.0

Table 1: Comparison between MLPs with two different criteria and a mixture of MLPs. Training and test errors represent classification error ratios.

criterion. After computation of the MSE for various number of hidden units, we observed on both data sets that it was impossible to decrease arbitrarily the training MSE when we had more than 500 hidden units. As MLPs are universal approximators, and as we were not able to reduce the training MSE, we conclude that there is an optimization problem. This optimization problem could explain the bad results, as compared to the cross-entropy criterion. This leads us to two crucial questions: *why does the CE criterion used with an MLP lead to an easier optimization problem compared to the MSE criterion?* Moreover, *why does a mixture trained with the same MSE criterion obtain statistically significantly better results on the training set (with* 99% *confidence for both data sets) than an MLP?*

3.3 Study of the Cost Function Landscape

As we suspect some optimization problems with the MSE criterion, we propose to study the *local* behavior of each cost function, and to compare them. Given a model $f_\theta(\cdot)$ where we want to optimize parameters θ, and given a vector of parameters θ^o, the cost function $E_{x,y}(\theta) = C(f_\theta(x), y))$ can be approximated with respect to θ around θ^o, by a second order Taylor expansion:

$$E_{x,y}(\theta) = E_{x,y}(\theta^o) + (\theta - \theta^o)^T \frac{\partial E_{x,y}(\theta^o)}{\partial \theta} + \frac{1}{2}(\theta - \theta^o)^T H_{x,y}(\theta^o) (\theta - \theta^o) + o(\|\theta - \theta^o\|_2^2) \quad (6)$$

where $\partial E_{x,y}(\theta^o)/\partial\theta$ and $H_{x,y}(\theta^o)$ are respectively the gradient and the Hessian matrix of $E_{x,y}$ with respect to θ, evaluated at θ^o. We use $\|.\|_2$ as the Euclidean norm for vectors, and $o(\|\theta - \theta^o\|_2^2)$ to represent a term negligible with respect to $\|\theta - \theta^o\|_2^2$. Remember that if we write $\theta = (\theta_1, \theta_2 \ldots)$, then the coefficient (i, j) of the Hessian matrix of the function $E_{x,y}$ is equal to

$$H_{x,y}^{i,j} = \frac{\partial E_{x,y}}{\partial \theta_i \, \partial \theta_j} \, .$$

Let us first note that since we are optimizing θ by gradient descent, it is clear that we don't want the derivative $\partial E_{x,y}/\partial\theta$ to be null when the example (x, y) is misclassified: we would be stuck on some undesirable plateau. Moreover, intuitively, we could say that the simpler the cost function $E_{x,y}(\cdot)$, the easier should be the optimization. Therefore, according to our local approximation (6), a simple Hessian $H_{x,y}$ could be better for the optimization.[6] Actually, as it is well-known that the Hessian is linked to convergence duration, as already suggested by LeCun et al. (1991), it seems quite

[6]Ideally, a null Hessian would be great, but not realistic. Hence, a sparse Hessian (with many 0 in the matrix) would be a good target.

reasonable to analyze it. Thus, we present in the following subsections an analysis of the gradient and the Hessian of the cost function, with respect to the weights to optimize. In fact, we propose a mathematical analysis of the "instantaneous" Hessian $H_{x,y}$, as well as an empirical evaluation of the "total" Hessian

$$H = \frac{1}{T} \sum_t H_{x_t,y_t} \tag{7}$$

of the cost function (4) which is the one that we really would like to minimize.

3.3.1 Mixtures of Experts and MSE

Let us first consider a mixture of MLPs $f(x) = \sum_{k=1}^{K} g_k(x) f_k(x)$ as given in (3). If we introduce v_i as the weight *vector* of f_i, then using the MSE criterion, the gradient is

$$\frac{\partial E_{x,y}}{\partial v_i} = -(y - f(x))\, g_i(x) \, \frac{\partial f_i(x)}{\partial v_i} \; .$$

This gradient is quite nice, because it behaves exactly as we would like: the first term $(y - f(x))$ is non-null when the example (x, y) is misclassified. Moreover, at least one gater output $g_i(x)$ must be non-null because these outputs sum to one. Hence, at least one expert will always receive some gradient when the example is misclassified.

Furthermore, the instantaneous Hessian *outside the block-diagonal* (one block for each pair of experts (i, j)) becomes:

$$\frac{\partial^2 E_{x,y}}{\partial v_i \, \partial v_j} = g_i(x) g_j(x) \frac{\partial f_i(x)}{\partial v_i} \left(\frac{\partial f_j(x)}{\partial v_j} \right)^T \quad (i \neq j) \; . \tag{8}$$

Recall that the basic idea of mixtures of experts is to try to partition the input space using a gater, in which case a given example would be only handled by one expert. Hence, for a given x, all $g_i(x)$ except one should be near zero. To verify what really happens in practice, we computed on the Forest data set the average of the product $g_i(x) g_j(x)$ over the training set, and over all pairs of distinct gater outputs $(i \neq j)$, with respect to the number of training iterations, as shown in Figure 1(a). It is clear that the product $g_i \, g_j$ tends to be very small in practice, and thus the Hessian (8) converges to zero outside the block-diagonal. To verify further this assertion, we also computed in Figure 1(b) the spectral norm

$$\left\| \frac{\partial^2 E_{x,y}}{\partial v_i \, \partial v_j} \right\|_2$$

for each block[7] (i, j) of the total Hessian (7) after 5 training iterations. This shows that the Hessian quickly becomes diagonal during training.

3.3.2 MLPs with MSE and CE

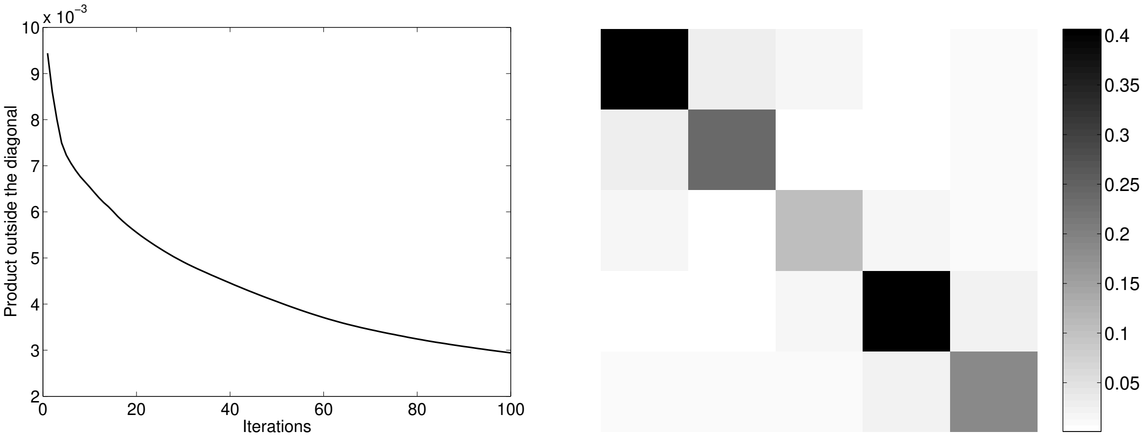
Let us now analyze the first derivative of the MLP given in (2) that would be respectively trained with the MSE and CE criteria. First the gradient when trained with the MSE criterion:

$$\frac{\partial E_{x,y}}{\partial w_i} = -(y - f(x))\, \alpha_i \, h'(w_i \, x) \, x \; ,$$

and then the gradient obtained with the CE criterion:

$$\frac{\partial E_{x,y}}{\partial w_i} = -p(-y|x)\, \alpha_i \, h'(w_i \, x) \, x \; . \tag{9}$$

[7] As each block is symmetric, the spectral norm corresponds to the largest absolute eigenvalue of this block.



(a) Average product of distinct pairs of gater outputs

(b) Spectral norm of each block in the Hessian matrix

Figure 1: Block-diagonality of the Hessian matrix: (a) shows that the term $g_i(x)g_j(x)$ $(i \neq j)$, which appears in the Hessian equation outside the block-diagonal, tends to very small values on average during training. This is on 100,000 examples of the Forest data set, with 10 experts, 25 hidden units per expert and 100 hidden units for the gater. (b) is a description of the Hessian matrix; each block corresponds to a pair of experts, and is represented by its spectral norm. The color of each block is associated with the corresponding value in the right column. This has been computed with 10,000 examples of the Forest data set, after 5 iterations, with 5 experts, 10 hidden units per expert and 25 hidden units for the gater.

Both are interesting because the first term $(y - f(x))$ or $p(-y|x)$ will be non-null if the example (x, y) is misclassified. Thus, if the initial values α_i of the output layer are non-null, all hidden units should receive a gradient.

Let us now compute the Hessian coming from the MSE criterion, using a vectorial notation, and outside the block-diagonal:

$$\frac{\partial^2 E_{x,y}}{\partial w_i \, \partial w_j} = \alpha_i \alpha_j \, h'(w_i \, x) \, h'(w_j \, x) \, x x^{\mathbf{T}} \quad (i \neq j) \ .$$

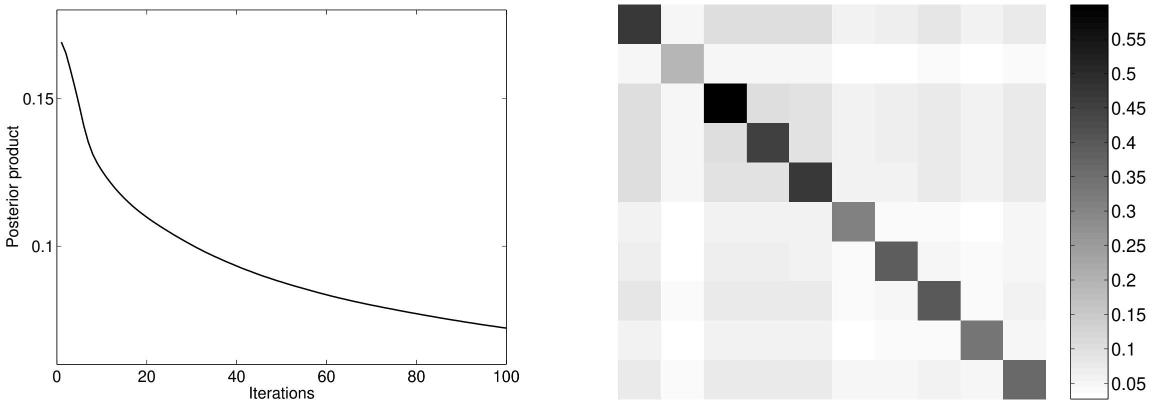
Note that there is no obvious reason for this Hessian to tend to zero, whereas if we compute the Hessian with the cross-entropy (CE) criterion we get:

$$\frac{\partial^2 E_{x,y}}{\partial w_i \, \partial w_j} = p(y|x)p(-y|x) \, \alpha_i \alpha_j \, h'(w_i \, x) h'(w_j \, x) \, x x^{\mathbf{T}} \quad (i \neq j) \ .$$

Here the term $p(y|x)p(-y|x) = p(y|x)(1 - p(y|x))$ will tend very quickly to zero, since we are training the MLP to maximize $p(y|x)$. This is verified in practice, as shown in Figure 2(a). It will push the Hessian obtained with the CE criterion toward almost block-diagonality (*one block for each unit*) (see Figure 2(b)), whereas the Hessian obtained with the MSE criterion remains full, as shown in Figure 3.

3.3.3 Why a Block-Diagonal Hessian Should Be Better

In the previous subsections we have seen empirically that when the Hessian induced by either the model or the criterion is block-diagonal, the resulting training performance is better. Let us now try



(a) Average product of posterior probabilities $p(y|x)p(-y|x)$

(b) Spectral norm of each block in the Hessian matrix

Figure 2: Block-diagonality of the Hessian matrix of an MLP trained with the CE criterion: (a) shows that the term $p(y|x)p(-y|x)$, which appears in the Hessian equation outside the block-diagonal, tends to very small values on average during training. This is on the Forest data set, with 100,000 training examples and 200 hidden units. (b) is a description of the Hessian matrix with 10 hidden units; each block corresponds to a pair of hidden units, and is represented by its spectral norm. The color of each block is associated with the corresponding value in the right column. It has been computed on the Forest data set with 10,000 training examples, after one iteration.

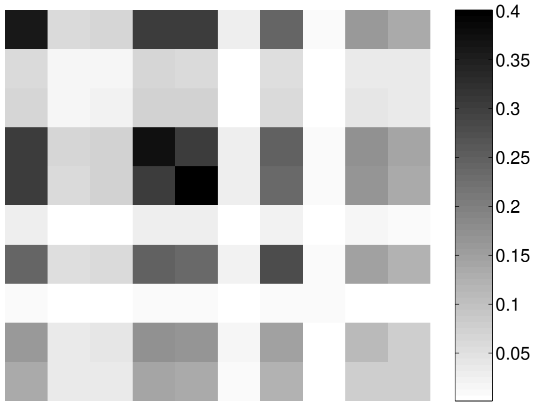


Figure 3: Hessian matrix for an MLP with 10 hidden units trained with the MSE criterion. Each block corresponds to a pair of hidden units, and is represented by its spectral norm. The color of each block is associated with the corresponding value in the right column. It has been computed on the Forest data set with 10,000 training examples, after 10 iterations.

to justify this empirical result. Let us consider a model $f_\theta(\cdot)$ where the parameter vector θ can be segmented into several sub-vectors $\theta = (\theta_1, \theta_2 \ldots \theta_n)$. Given a current state θ^o, if we consider the local quadratic approximation given by equation (6), and if we forget negligible terms with respect to $\|\theta - \theta^o\|_2^2$, the optimal weights $\theta^\star = (\theta_1^\star, \ldots \theta_n^\star)$ have to satisfy the following equality:

$$\frac{\partial E_{x,y}(\theta^o)}{\partial \theta} + H_{x,y}(\theta^o)\ (\theta^\star - \theta^o) = 0$$

which is equivalent to the system of equations

$$\frac{\partial E_{x,y}(\theta^o)}{\partial \theta_i} + H^{i,i}_{x,y}(\theta^o)\;(\theta^\star_i - \theta^o_i) + \sum_{j \neq i} H^{i,j}_{x,y}(\theta^o)\;(\theta^\star_j - \theta^o_j) = 0 \quad \forall i\;, \tag{10}$$

where we adopt the notation

$$\frac{\partial^2 E_{x,y}}{\partial \theta_i\, \partial \theta_j} = H^{i,j}_{x,y}\;.$$

In the ideal case where $H_{x,y}$ is block-diagonal, that is if $H^{i,j}_{x,y}$ is zero for $i \neq j$, this leads to

$$\frac{\partial E_{x,y}(\theta^o)}{\partial \theta_i} + H^{i,i}_{x,y}(\theta^o)\;(\theta^\star_i - \theta^o_i) = 0 \quad \forall i\;. \tag{11}$$

In other words, the optimal weights $\theta^\star_i$ are solution of independent systems of equations, and thus are locally independent. Therefore, the optimization problem is *much simpler* than with a full Hessian where the modification of only one weight would also affect the modification of all other weights. This could explain the significant difference in the training performance obtained with either an MLP trained with the CE or a mixture trained with the MSE criterion (where the Hessian is almost block-diagonal), compared to an MLP trained with the MSE criterion (where the Hessian remains full).

If $H_{x,y}$ is not truly block-diagonal, one can compute

$$\|\sum_{j \neq i} H^{i,j}_{x,y}(\theta^o)\;(\theta^\star_j - \theta^o_j)\|_2 \leq \sum_{j \neq i} \|H^{i,j}_{x,y}(\theta^o)\|_2\;\|\theta^\star_j - \theta^o_j\|_2$$

where $\|.\|_2$ is the Euclidean norm for vectors or the spectral norm for matrices. Therefore, the more the norm $\|H^{i,j}_{x,y}\|_2$ of the blocks of the Hessian outside the diagonal tends to zero, the more the approximation (11) of the system of equations (10) is accurate, and the simpler should be the corresponding optimization problem.

3.3.4 MLPs with Hyperbolic Tangent Output

It is well-known that adding a hyperbolic tangent at the output layer of the MLP proposed in (2) usually improves classification performances. Let us analyze here the effect of adding this hyperbolic tangent on the resulting optimization problem. Keeping the same notation, this is equivalent to using the following cost function:

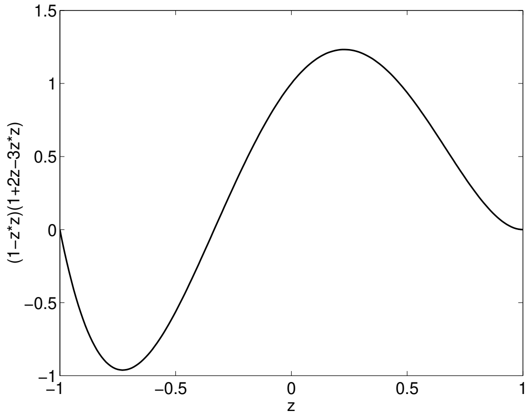
$$C(f(x),\, y) = \frac{1}{2}(y - \tanh(f(x)))^2\;.$$

Then computing the Hessian for each block, using the fact that $\tanh'(\cdot) = 1 - \tanh(\cdot)^2$ and using the shortcut $z = \tanh(f(x))$, we obtain outside the block diagonal:

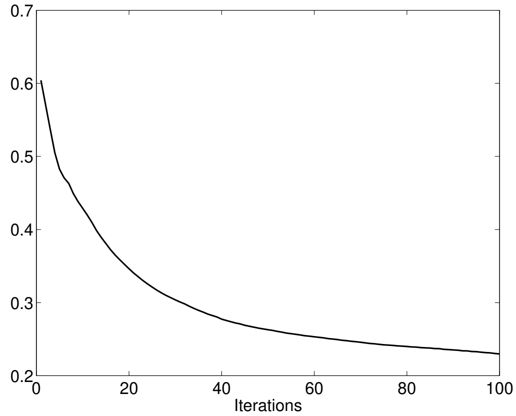
$$\frac{\partial^2 E_{x,y}}{\partial w_i\, \partial w_j} = (1 - z^2)(1 + 2zy - 3z^2)\,\alpha_i \alpha_j\, h'(w_i\, x)\, h'(w_j\, x)\, x x^{\mathbf{T}} \quad (i \neq j)\;.$$

It is interesting to note that the difference with an MLP without a hyperbolic tangent output is simply the coefficient $c(z) = (1 - z^2)(1 + 2zy - 3z^2)$. To study this coefficient, we plotted it in Figure 4(a) with $y = 1$, for a range of z values between -1 and 1 (which corresponds to the values the hyperbolic tangent can take). It is interesting to note that this coefficient tends to zero when $z = \tanh(f(x))$ tends to ± 1. As we are training the MLP with this aim, $c(z)$ will tend to small values during training, as shown in Figure 4(b). Once again, the Hessian will tend to have small values outside the block-diagonal. Note however that there is a drawback when z tends to ± 1: the first derivative

$$\frac{\partial E_{x,y}}{\partial w_i} = -(y - z)\,(1 - z^2)\,\alpha_i\, h'(w_i\, x)\, x \tag{12}$$



(a) Plot of the function $c(z) = (1 - z^2)(1 + 2zy - 3z^2)$.



(b) Average over the examples of the coefficient $|c(z)|$ during training.

Figure 4: Block-diagonality of the Hessian for an MLP trained with the MSE criterion, with hyperbolic tangent outputs. (a) shows that the coefficient $c(z)$ involved in the Hessian tends to zero when the output $z = \tanh(f(x))$ tends to ± 1. (b) shows that the average over the training examples of the absolute value of this coefficient $c(z)$ decreases during training. (a) and (b) have been computed on the Forest database with 200 hidden units and 100,000 training examples.

will decrease, even for misclassified examples, due to the coefficient $1 - z^2$, whereas for an MLP with a CE criterion, the first derivative is (in general) small only for well-classified examples. Even if the hyperbolic tangent output improves training results for an MLP with the MSE criterion, we could expect better results in general with the CE criterion. This is shown in Table 2: on one hand, MLPs

Forest			
Model	Cost Function	Train Err. (%)	Test Err. (%)
MLP (200HU)	MSE	13.7	13.9
MLP+tanh (200HU)	MSE	10.9	12.6
MLP (200HU)	CE	10.5	12.0
MLP (500HU)	CE	10.7	11.1
Letter			
Model	Cost Function	Train Err. (%)	Test Err. (%)
MLP (500HU)	MSE	1.7	3.6
MLP+tanh (500HU)	MSE	1.1	3.9
MLP (500HU)	CE	0.4	3.6

Table 2: Improvement obtained with an hyperbolic tangent output for an MLP trained with the MSE criterion. Results with MLPs trained with the CE criterion are given for comparison.

trained either with the MSE criterion with a hyperbolic tangent output and the CE criterion obtain similar training performance on the Forest data set. On the other hand, on the Letter data set, the MLP with hyperbolic tangent is still not able to reach a training error similar to the one using the CE criterion. As suggested previously, this could be explained by the fact that the coefficient $1 - z^2$ in (12) is very small on the Letter data set (less than 1/25 after 10 iterations, decreasing to 1/100) leading to a small gradient, whereas it remains quite reasonable on the Forest data set (around 1/4).

3.4 Conclusion

To conclude this section, let us wrap up the current findings. First of all, we showed that both an MLP trained with a CE criterion and a ME trained with an MSE criterion lead to local optimization problems where the hidden units or the experts are solution of independent minimization problems, mainly because the Hessian becomes almost block-diagonal. This is in contrast to an MLP trained with an MSE criterion, where all units depend on the others, which lead to a more complicated optimization problem. We conjectured that during training, the independence between units of a model (such as hidden units for an MLP or experts for a ME) could improve training results in general. Using this argument, we showed that one could explain improvements obtained with an MLP with a hyperbolic tangent output. However the addition of this hyperbolic tangent has a drawback: the gradient could tend to be very small, which in some cases could reduce the expected improvements.

Finally, note that while optimization is a crucial problem, it doesn't solve everything: even when an MLP is trained with the cross-entropy criterion, its performance is still worse in generalization than the one obtained with a mixture of experts (see Table 1). Moreover, on the Letter data set, even if we improve the training performance of an MLP by adding an hyperbolic tangent output, the results in test are in fact worse. Thus, let us now focus on another interesting viewpoint: the margin introduced in SVMs algorithms.

4 SVM Margin for MLPs

Support Vector Machines (SVMs) were introduced by Vapnik (1995). They have been applied to many classification problems, generally yielding good performance compared to other algorithms. Given the two-class classification problem presented in Section 2, a linear SVM finds a separating hyperplane which maximizes the margin between this hyperplane and the two classes, as suggested in Figure 5. More formally, given a hyperplane $f_{\alpha,b}(x) = 0$ with $f_{\alpha,b}(x) = \alpha\, x + b$ ($\alpha \in \mathbb{R}^d$, $b \in \mathbb{R}$), the SVM

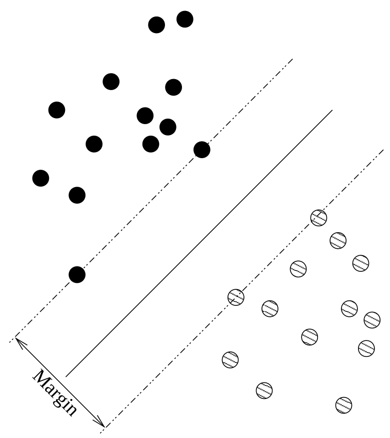


Figure 5: Concept of the margin in SVMs: we want to separate two classes (black and white circles). Many separating hyperplanes could exist, but we are searching for the one that maximizes the margin.

problem is equivalent to minimizing

$$L(\alpha, b) = \frac{\mu}{2}\, \|\alpha\|_2^2 + \sum_{t=1}^{T} |1 - y_t f_{\alpha,b}(x_t)|_+ \tag{13}$$

where $|x|_+ = \max(0, x)$, and μ is a constant which acts as a trade-off between the first term which corresponds to the margin maximization, and the second term which tries to force the two classes to be separated. This problem is equivalent to a simple quadratic optimization problem under constraints, and it can be shown that the optimal value of α is a linear combination of input training vectors x_t. Moreover, the optimal separating hyperplane can be expressed as a weighted sum of inner-products $x_t\, x$. It is then straightforward to obtain non-linear SVMs by projecting the input vectors in a higher-dimensional space using traditional kernel methods: basically, we just have to replace all inner-products by a kernel evaluation. Let us just point out here that even with non-linear SVMs, the idea remains the same: trying to maximize the margin, but *in a higher dimensional space*.

4.1 A First SVM Criterion for MLPs

We propose to apply directly the margin concept of SVMs to MLPs. This idea is not new (see for example the paper written by Zhong and Ghosh, 2000), but we think it has not been studied deeply enough, especially from an optimization view point. Note that the one-hidden layer MLP that we proposed in (2) first sends input vectors in a non-linear space using the hidden layer, and then separates the data in this space using the output layer. From a structural point of view, SVMs and MLPs are very close: with SVMs, we select the non-linear space by choosing the kernel, while with MLPs we attempt to optimize the non-linear space by optimizing the hidden layer. Thus, we could try to maximize the margin in the non-linear space of the MLPs, using an adapted stochastic version of the cost function (13):

$$C(f(x),\, y) = \frac{\mu}{2\,T}\, \|\alpha\|_2^2 + |1 - y\, f(x)|_+ \; . \tag{14}$$

The results obtained with this criterion are given in Table 3 for $\mu = 0$. Note that no improvement is obtained when compared to the use of the CE criterion.

Forest			
Model	Cost Function	Train Err. (%)	Test Err. (%)
MLP (500HU)	CE	10.7	11.1
MLP (500HU)	SVM-like	10.2	11.9

Letter			
Model	Cost Function	Train Err. (%)	Test Err. (%)
MLP (500HU)	CE	0.4	3.6
MLP (500HU)	SVM-like	1.4	3.9

Table 3: Comparison between MLPs trained with the CE criterion and an SVM criterion.

4.2 Local Analysis of the SVM Criterion

Let us now analyze locally the cost function (14) for $\mu = 0$, following the ideas given in Section 3. We will not consider the case $y f(x) = 1$ where the cost function is not differentiable, simply because it is quite rare in practice, at least from a computational point of view. Thus, we can notice that in general we can compute the first derivative as follows ($\mu = 0$):

$$\begin{aligned} \frac{\partial E_{x,y}}{\partial w_i} &= -y\, \alpha_i\, h'(w_i\, x)\, x \quad \text{if} \quad y f(x) < 1 \\ &= 0 \quad \text{otherwise.} \end{aligned} \tag{15}$$

In general, the derivative is non-zero when the example is inside the margin or misclassified, (that is if $y f(x) < 1$), and behaves similarly to the derivative obtained with the CE criterion (9), where the coefficient $p(-y|x)$ tends to be non-zero only if the example is misclassified. From (15), we can derive the Hessian easily:

$$\frac{\partial^2 E_{x,y}}{\partial w_i\, \partial w_j} = 0 \quad (i \neq j) \; .$$

The Hessian of the cost function is thus *completely diagonal*, and according to Section 3.3.3 we can expect an easier optimization for the MLP, as locally the cost function is simpler than those obtained with the MSE and the CE criteria. However, the results in Table 3 (which show that training and testing errors are not better with this criterion), suggest (as already highlighted in Section 3.4) that the optimization is not the only crucial point to consider. Let us consider now the point of view of margin maximization adopted in SVM-type algorithms.

4.3 Improving the Margin of MLPs

Going back to equation (14), let us recall that the first part of this criterion represents the maximization of the margin which is controlled by the hyper-parameter μ. Unfortunately, when we try to vary μ in order to force larger margins, the corresponding results are similar or even worse than those obtained with $\mu = 0$ in Table 3. This seems to be in contradiction with the largely accepted fact in the SVM community that margin maximization improves generalization. We suspect some numerical problems with the coefficient $\frac{\mu}{2T}$ which is probably too small during stochastic gradient descent. In order to fix this problem, we propose to instead use the following cost function:

$$C(f(x),\, y) = |\beta - y f(x)|_+ \ . \tag{16}$$

By definition, the margin in the space corresponding to projected examples $\tilde{x} = (\tanh(w_1\, x), \tanh(w_2\, x), \ldots, \tanh(w_N\, x))$ generated by the hidden layer of the MLP is the distance between hyperplanes $\alpha\tilde{x} + b = \beta$ and $\alpha\tilde{x} + b = -\beta$. After some arithmetics, we obtain the following margin:

$$\frac{2\beta}{\|\alpha\|_2} \ .$$

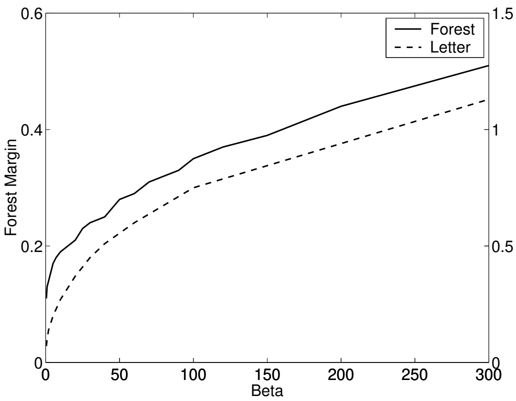
Thus, in order to increase this margin we could increase β (instead of μ in (14)) with the hope that the norm of α obtained after training with a larger β would not increase as quickly as β. In practice this seems to be the case, as shown in Figure 6(a). It is also interesting to notice that there is an optimal size of the margin as suggested in Figure 6(b). The best generalization performances (shown in Table 4), obtained after tuning β according to the validation set, were 8.3% error on the test set of the Forest data set and 2.3% error on the test set of the Letter data set. These are great improvements (statistically significant with 99% confidence) compared to a standard CE criterion which obtained respectively 11.1% and 3.6% of testing error rates.

4.4 Criteria Comparison

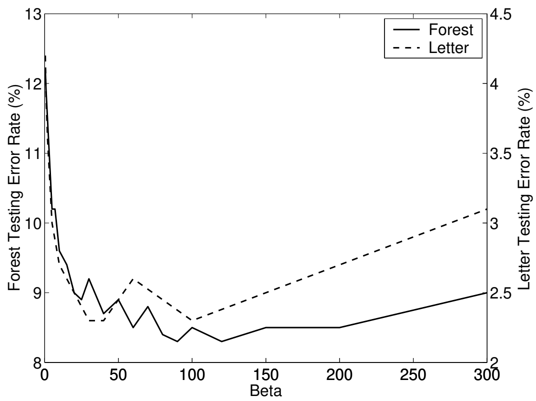
Let us conclude this section by noting that function $z \mapsto |\beta - z|_+$ can be seen as a “hard” version of function $z \mapsto \log(1 + \exp(\beta - z))$. This is depicted in Figure 7 for $\beta = 1$. Moreover, we observed in practice that results obtained with the “hard” and the “soft” versions of the cost function are similar: if the best results with the “soft” version were achieved for a certain β_s, we always found a $\beta_h \geq \beta_s$ which led to similar results with the “hard” version. In fact, *the SVM criterion and the cross-entropy criterion used with MLP are very close.* From a structural point of view, the main difference between SVMs and MLPs using this criterion seems to be the feature space: in SVMs the feature space is fixed and generated using a kernel, whereas in MLPs it is generated by the hidden layer and adapted by training the weights of this hidden layer. Of course, bear in mind that there are other differences between SVMs and MLPs, such as the training algorithm itself, which might influence your selection of one or the other for specific applications.

5 Conclusion

In this paper we analyzed gradient descent algorithms with respect to several important aspects. As already known, the Hessian matrix plays a major role in the effectiveness of any gradient descent



(a) Margin with respect to β.



(b) Test Error Rate with respect to β.

Figure 6: Importance of the margin in MLPs. (a) shows the evolution of the margin with respect to the parameter β introduced in the criterion (16). (b) shows the evolution of the testing error with respect to β. Results in (a) and (b) have been both evaluated on the Forest (left scale of both figures) and Letter (right scale of both figures) data sets with MLPs containing 500 hidden units, using 100,000 training examples for Forest and 14,000 examples for Letter.

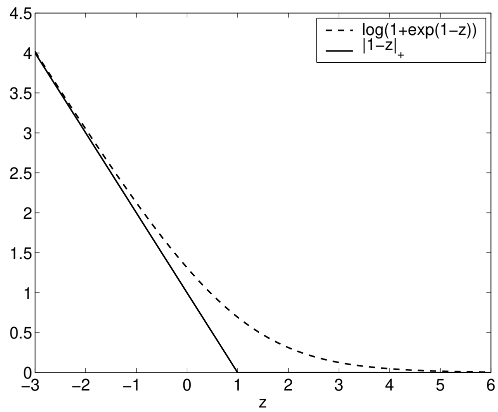


Figure 7: Comparison between the function $z \mapsto |1 - z|_+$ involved in the SVM criterion, and its smooth approximation $z \mapsto \log(1 + \exp(1 - z))$ used in the CE criterion.

algorithm. We explained, both empirically and theoretically, why a block-diagonal Hessian should yield more efficient training algorithms. We showed how the choice of the training criterion influences the Hessian matrix, and hence how to select an efficient training criterion. More specifically, we explained why the Cross Entropy criterion should be selected for classification problems instead of the more classical Mean Squared Error criterion, and why Mixtures of Experts also generate simple Hessian matrices. Finally, while efficient optimization is important, bearing in mind that the ultimate goal in machine learning is good generalization performance, we introduce a new cost function inspired by the SVM algorithm which yields a block-diagonal Hessian and enables the control of the margin in the hidden layer space. This cost function yielded (statistically significantly) better generalization performance on the two tested benchmark data sets. Future work could involve comparing the feature space generated by kernels in SVMs with the feature space generated by the hidden layer in MLPs,

Forest		
Model	Cost Function	Test Err. (%)
MLP (200HU)	MSE	13.9
MLP (500HU)	CE	11.1
ME (25HU per expert, 10 experts, 100HU for the gater)	MSE	9.8
MLP (500HU)	**Margin**	**8.3**

Letter		
Model	Cost Function	Test Err. (%)
MLP (500HU)	MSE	3.6
MLP (500HU)	CE	3.6
ME (50HU per expert, 20 experts, 25HU for the gater)	MSE	3.0
MLP (500HU)	**Margin**	**2.3**

Table 4: Comparison of the generalization performances of models trained with standard criteria (MSE and CE) and the "margin" criterion proposed in (16).

which remains the main structural difference between SVMs and MLPs trained with the proposed criterion. Moreover, we could exploit the block-diagonality of the Hessian to apply well-known second order techniques in an efficient way: this could eventually improve the quality of the optimization.

Acknowledgments

The authors acknowledge financial support provided by the Swiss National Center of Competence in Research (NCCR) on Interactive Multimodal Information Management (IM)2. The NCCR are managed by the Swiss National Science Foundation on behalf of the Federal Authorities. Moreover, the authors would like to thank Conrad Sanderson for fruitful discussions.

References

S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 14(2):251–276, 1998.

T. Battiti. First and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4(2):141–166, 1992.

C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, London, UK, 1995.

R. Fletcher. *Practical Methods of Optimization*. Wiley, New York, second edition, 1987.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3:79–87, 1991.

Y. LeCun, L. Bottou, G.B. Orr, and K.-R. Müller. Efficient backprop. In G.B. Orr and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 1998.

Y. LeCun, I. Kanter, and S. Solla. Second order properties of error surfaces: learning time, generalization. In R. P. Lippman, J. M. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 918–924, Denver, CO, 1991. Morgan Kaufmann.

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipies in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002.

V.N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

S. Zhong and J. Ghosh. Decision boundary focused neural network classifier. In *Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE)*. ASME, 2000.