A new method to increase the margin of multilayer perceptrons

Vincent Lemaire , Olivier Bernier , Daniel Collobert and
Fabrice Clérot
France-Télécom CNET DTL/DLI/TNT
Technopole Anticipa, 2 Avenue Pierre Marzin 22307 Lannion cedex FRANCE
tel : +33 (0) 2 96 05 31 07 — fax : + 33 (0)2 96 05 23 58
email: vins.lemaire@cnet.francetelecom.fr

(Received ..... ; Accepted in final form .....)

Key words: classification, face detection, margin, neural network

Abstract.
A new method to maximize the margin of MLP classifier in classification problems is described. This method is based on a new cost function which minimizes the variance of the mean squared error. We show that with this cost function the generalization performance increase. This method is tested and compared with the standard mean square error and is applied to a face detection problem.

1. Introduction

The multilayer perceptron is one of the most widely used network paradigm [9] [6] and is usually trained using conventional techniques such as backpropagation [5] with the mean squared error as cost function. In classification problems, the decision boundaries can be very complex and difficult to learn. In order to solve this problem more advanced methods have been developed and applied for training multilayer perceptrons.

The training phase, using backpropagation, is an algorithmic process during which the network parameters are adjusted to minimize a cost function in order to find appropriate boundaries between the classes. Most of the technics discussed in the litterature about feedforward neural networks refer to network trained by minimizing a quadratic function such as the mean squared error [7].

For multilayer perceptrons with continuous functions, several particular values of the output are used as desired values for the different classes of a classification problem. Therefore two kinds of errors exist : the estimation error which is the difference between the desired output and the obtained output and the classification error leading to a missclassified example when its estimation error is greater than a given threshold.

Bounds of generalization error of composite classifier systems have been previously formulated [8][3] and are based on the notion of the margin of classification. The size of the margin depends both on the mean squared error and on the distribution of the estimation errors and therefore on the variance of the squared error on each class (fig 1). Consequently, the performance in terms of correct classification depends on the particular shape of the distribution of the estimation error. Therefore, the choice of an appropriate cost function to control the shape of the distribution can be crucial to obtain a reasonable solution to the problem.

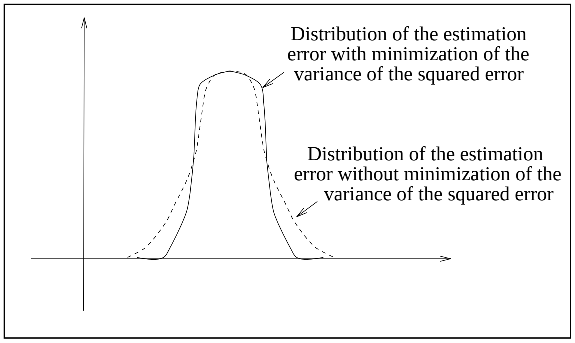

Figure 1. Influence of the minimization of the variance of the squared error on the distribution of the estimation error.

The present paper approaches the problem by introducing a new method to maximize the margin. This method takes into account a fourth order momentum , the variance of the squared error, through the cost function to improve the generalization. This method is studied where previous methods [8] [4] (involving several neural networks) to increase the margin would lead to a too large computation time for industrial application. We show that the margin can be increased with a single multilayer perceptron, using this new cost function.

2. Increasing the margin by decreasing the variance of the error

Let us consider a classification problem consisting of two classes C_1, C_2 classified by a discriminant neural network with one output. The goal of the training phase is to obtain the following outputs for the network:

- if $x \in C_1$ then $f_w(x) = d_1$
- if $x \in C_2$ then $f_w(x) = d_2$

with x the input vector, d_1, d_2 the desired outputs respectively for an example of the class C_1, C_2 and $f_w(x)$ the answer given by the neural network.

At the end of the learning phase, this neural network has a mean squared error m_1 on the class C_1 with a variance σ_1^2 and a mean squared error m_2 on the class C_2 with a variance σ_2^2 such as:

$$\begin{aligned}
\sigma_1^2 &= \frac{1}{n_1} \sum_{\substack{a=1 \\ \mathrm{a} \in \mathrm{C}_1}}^{n_1} \left[(d^a - s^a)^2 - \frac{1}{n_1} \sum_{\substack{b=1 \\ \mathrm{b} \in \mathrm{C}_1}}^{n_1} (d^b - s^b)^2 \right]^2 \\
\sigma_2^2 &= \frac{1}{n_2} \sum_{\substack{c=1 \\ \mathrm{c} \in \mathrm{C}_2}}^{n_2} \left[(d^c - s^c)^2 - \frac{1}{n_2} \sum_{\substack{d=1 \\ \mathrm{d} \in \mathrm{C}_2}}^{n_2} (d^d - s^d)^2 \right]^2
\end{aligned} \tag{1}$$

where :

- n_p is the number of examples of the class C_p;
- s^k is the output of the neural network for the input k;
- d^k is the desired output for the input k;

There are several ways to take into account the minimization of the variances σ_1^2 and σ_2^2 without degradation of the global mean squared error m to increase the margin. One of them is to add to the standard cost function a term associated to the variance of the squared error for each class.

The expression of the cost function becomes in the case where there are several output neurons :

$$\begin{aligned}
C^k = \Big[\quad & (d^k - s^k)^2 \\
& + \tfrac{1}{n_1} \sum_{\substack{a=1 \\ a \in C_1}}^{n_1} \left[(d^a - s^a)^2 - \tfrac{1}{n_1} \sum_{\substack{b=1 \\ b \in C_1}}^{n_1} (d^b - s^b)^2 \right]^2 \\
& + \tfrac{1}{n_2} \sum_{\substack{c=1 \\ c \in C_2}}^{n_2} \left[(d^c - s^c)^2 - \tfrac{1}{n_2} \sum_{\substack{d=1 \\ d \in C_2}}^{n_2} (d^d - s^d)^2 \right]^2 \ \Big]
\end{aligned} \tag{2}$$

This cost function is the sum of the standard squared error and of the variance of the squared error of both classes.

The expression of the gradient of this cost function C for to the output neuron i and for an example $k \in C_1$ is :

$$\begin{aligned}
\left.\frac{\partial C^k}{\partial w_{ij}}\right|_{k \in C1} &= \frac{\partial}{\partial w_{ij}}\left[(d^k - s^k)^2\right] \\
&+ \frac{\partial}{\partial w_{ij}}\left[\frac{1}{n_1}\sum_{\substack{a=1 \\ a \in C_1}}^{n_1}\left[(d^a - s^a)^2 - \frac{1}{n_1}\sum_{\substack{b=1 \\ b \in C_1}}^{n_1}(d^b - s^b)^2\right]^2\right] \\
&+ \frac{\partial}{\partial w_{ij}}\left[\frac{1}{n_2}\sum_{\substack{c=1 \\ c \in C_2}}^{n_2}\left[(d^c - s^c)^2 - \frac{1}{n_2}\sum_{\substack{d=1 \\ d \in C_2}}^{n_2}(d^d - s^d)^2\right]^2\right]
\end{aligned} \tag{3}$$

The gradient, which does not depend on the variance of the class C_2, is consequently :

$$\begin{aligned}
\left.\frac{\partial C^k}{\partial w_{ij}}\right|_{k \in C1} &= s_j^k(-2f'(a^k)(d^k - s^k) \\
&- s_j^k \frac{4}{n_1} f'(a^k)(d^k - s^k)\left[(d^k - s^k)^2 - \frac{1}{n_1}\sum_{\substack{e=1 \\ e \in C_1}}^{n_1}(d^e - s^e)^2\right]
\end{aligned} \tag{4}$$

where a^k is the weighted input of the output neuron i for the example k.

This gradient can be expressed as the sum of the standard gradient (Y^k_{quad}) and of the gradient bound to the variance of the squared error (Y^k_{var}).

$$\frac{\partial C^k}{\partial w_{ij}} = Y^k_{quad} + Y^k_{var}$$

This formula is the same for an example belonging to the class C_2. The derivation of the gradient of the hidden neurons is done using the standard backpropagation with these two gradients on each hidden neurons. The variation of the weights can then be expressed as:

$$\Delta w_{ij}^{t+1} = \alpha_{quad} Y^k_{quad} + \alpha_{var} Y^k_{var} + \beta \Delta w_{ij}^t$$

with α_{quad} the learning rate on the squared error, α_{var} the learning rate on the variance of the squared error and β the momentum.

Since the mean squared error of each class and their variances should be calculated after each modification of the weights, the computation could be very long. To circumvent this problem, without degradation of the performances, the following algorithm is used :

```
⋆ for all iterations
    • for all examples $k$
        ∗ compute $f_w(k)$
        ∗ compute $Y^k_{quad}$ and $Y^k_{var}$ for the output neurons
        ∗ compute $Y^k_{quad}$ and $Y^k_{var}$ for the hidden neurons
        ∗ update the weights
    • compute the mean squared error for each class and their variances
```

3. Application

This new cost function is tested in the face detection pre-network of the MULTRAK [1] application, which is a real time system for automatic detection and tracking of multiple persons in a video conference. This system is able to continously detect and track the position of faces in its field of view. The heart of the system is a modular neural network based face detector [2] giving accurate and fast face detection. The pre-network is used as a filter which must be much faster than the modular neural network without degradation of the face detection rate. For real time performance of the system, the speed of the pre-network is critical and imposes to use only one neural network.

We train two pre-networks as face detectors : one using the previously described cost function, and the other using the standard squared error. Each neural network is a multilayer perceptron, with standard sigmoidal functions, 300 input neurons (corresponding to the window 15x20 pixels), one hidden layer with 8 neurons and one output.

The database consisted of three set of examples :

- Learning set : 7000 front view and turned faces and 7000 non faces;
- Validation set : 7000 front view and turned faces and 7000 non faces;
- Test set : 7000 front view and turned faces and 7000 non faces;

In order to compare the two cost functions differents experiments are made. For each experiment, 50 trainings are performed with different initialization of the weigths. This allow us to obtain for each experimental conditions, the mean and the confidence interval of each value. Each training is stopped when the cost on the validation set does not decrease since 200 iterations. At the end of each training, the global mean squared error, the variance of the squared error of each class, the margin and the detection rate are computed for the best configuration of the weights for this training on each one of the three subsets (training, validation and test set) and are discussed on the following section.

4. Comparison and Results

The new cost function, which includes the variance term, is compared to the standard cost function. In the following experiments we study the influence of the variance term. We show that if the added term is well chosen, the variance on the training set is decreased which increases both the margin on the training set and the classification performance on the test set. In the following figures the results based on the new cost function are labelled 'VMSE' and the results based on the standard cost function 'MSE'.

4.1. The influence of the variance term

In this section the α_{quad} parameter related to the squared error has a constant value of 10^{-2}. The influence of $\eta = (\alpha_{var}{}' / \alpha_{quad}) = (\alpha_{var}\ n_p / \alpha_{quad})$ (where n_p is the number of the examples of the class C_p) is examined in the $[10^{-4} : 10^2]$ range to estimate how the added gradient interacts with the gradient of the squared error. Comparisons are performed for the global mean squared error and for the variance of the squared error of each class. The results for the standard cost function are constant since α_{quad} is constant.

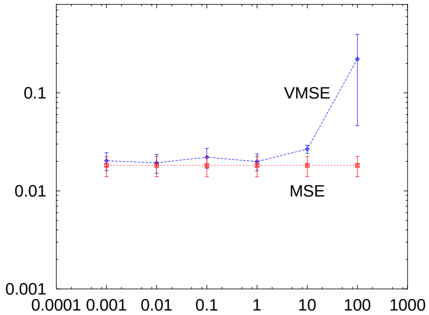

Figure 2. The mean squared error for the learning set with the two cost functions versus η.

Figure 2 shows the results obtained for the global mean squared error with the two cost functions on the learning set. For $\eta \in [10^{-4} : 10]$, the two cost functions provide approximatively the same results with the same confidence interval. On the other hand, for $\eta = 100$, the global mean squared error strongly increases with the new cost function. In this case, α'_{var} is so great, compared to α_{quad}, that the minimization of the variance prevents the minimization of the mean squared error and the neural network always gives the same result.

Figures 3 and 4 show the results obtained for the variance of the squared error of each class on the training set. For $\eta \in [10^{-4} : 10^{-1}]$ the two cost functions exhibit similar performances. For $\eta \in [10^{-1} : 10]$ the new cost function reduces the variance, which is much as 37 % smaller than with the standard cost function, together with a similar confidence interval. On the other hand, for $\eta = 100$, the variance is smaller but the confidence interval strongly increases.

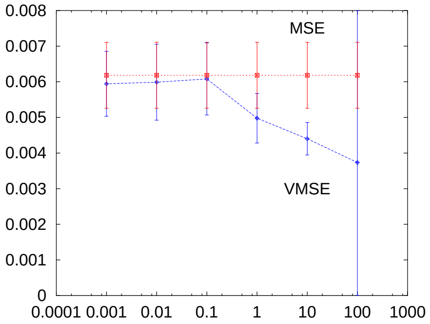

Figure 3. The variance of the first class (faces) for the learning set with the two cost functions versus η.

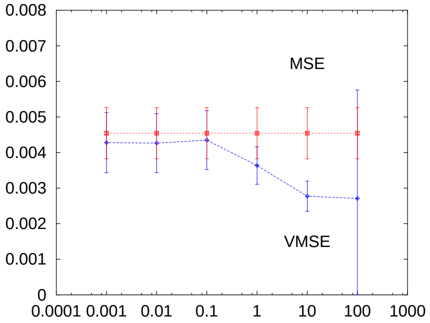

Figure 4. The variance of the second class (non faces) for the learning set with the two cost functions versus η.

These results show that the added variance term interacts with the squared error term. If it is comparable to the squared error term, it allows to improve the variance of the mean squared error on both classes. A well chosen value of the α'_{var} learning step of the variance term improves the variance of the squared error.

4.2. Margin maximization

A second experiment shows the relation between the minimization of the variance of the squared error and the maximization of the margin (α_{quad}=0.01; α'_{var}=0.01).

To quantify the percentage of the population within the margin, near the boundary and therefore the correctly classified rate outside the margin we determine a threshold θ (see figure 5), for different values of the margin. This threshold is tuned to obtain the best detection rate for the learning set, such that an example is considered well classified if :

- $f_w(k) \leq \theta$ and k $\in C_1$
- $f_w(k) \geq \theta$ + Margin and k $\in C_2$

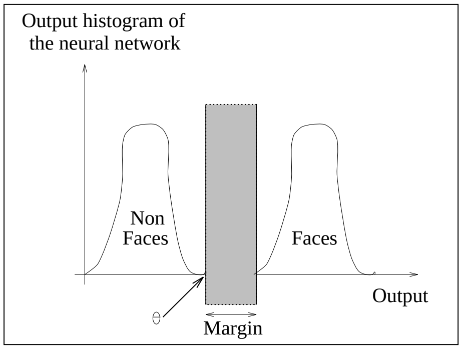

Figure 5. Explanation of the correctly classified rate versus the margin

The figure 6 shows that for a given margin, the detection rate with the new cost function is better than the standard cost function. Therefore for a given detection rate, the margin is increased.

Considering a detection rate of 98.5 % (fig 6) the new cost function has a margin of 0.17 ± 0.01 and the standard cost function has a margin of 0.12 ± 0.04. This difference, although small, represents an improvement of 29.4 % which is important. The effect of the minimization of the variances is to push the face distribution towards the right and the non face distribution towards the left on figure 5.

Our second goal is achieved : the minimization of the variance of the squared error of both classes indeed maximizes the margin.

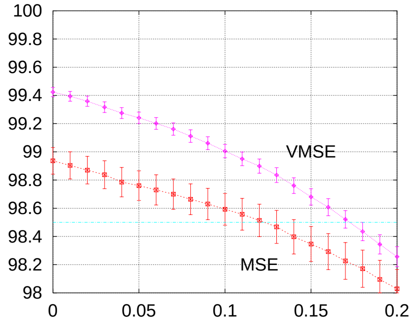

Figure 6. The correctly classified rate for the learning set versus the margin

4.3. A better margin increases the generalization

This section shows that the maximization of the margin on the learning set improves the performances on the test set. Figure 6 shows the effect on an improved margin : the difference between the two curves represents the improvement of the margin.

With the standard mean squared error and for a detection rate of the faces of 99.5, % the false alarm rate is 8 % while, with the new cost function, this false alarm rate is only 5 % which represents an improvement of 37 %.

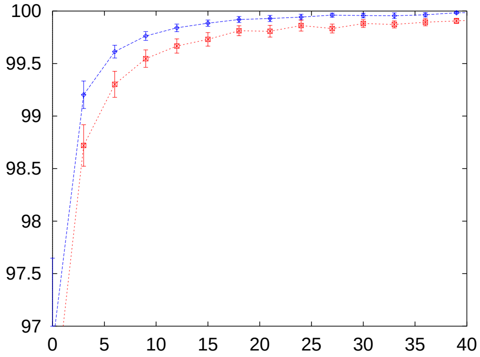

Figure 7. Detection rate of the faces on the test set versus the false alarm rate (missclassified faces and non faces) for the two cost functions.

5. Conclusion

A new direction is proposed to increase the generalization performance in classification problems of multilayer perceptrons. This new cost function is used in the face detection network called pre-network of the MULTRAK [1] application. The use of our new cost function has allowed to increase the performances of the pre-network as compared to the standard mean squared error : the false alarm is reduced by 20 % (on the test set A of the CMU face database) for the same detection rate.

This new method has been applied to a classification problem with two classes but could be extended to classification problems with more classes and in methods involving several neural networks.

References

1. O. Bernier, M. Collobert, R. Féraud, V. Lemaire, J.E. Viallet, and D. Collobert. MULTRAK : a system for Automatic Multiperson Localization and tracking in real-time. In *International Conference on Image Processing*, 1998.
2. Féraud, R. and Bernier, O. Ensemble and modular approaches for face detection: a comparison. In *Neural Information Processing System*, volume 10, pages 472–478, december 1997.
3. S. Holger and Y Bengio. Training method for adaptative boosting of neural networks. In *Neural Information Processing System*, 1998.
4. Breiman L. Bagging predictors. In *Technical report TR-421, University of california, Berkley*, 1994.
5. Le Cun, Y., Boser, B., and Denker, J. S. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 1989.
6. M. Rosenblatt. *Principles of neurodynamics: Perceptron and theory of Brains mechanisms*. Spartan Books, Washington D.C, 1962.
7. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In *Parallel Distribued Processing: Explorations in the Microstructures of Cognition*, volume 1, pages 318–362, 1986.
8. R. E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin : A new explanation for the effectiveness of voting methods. In *Machines That Learn*, 1997. http://www.research.att.com/ schapire/.
9. White, H. and Hornik, K. Mutilayer feedforward networks are universal approximators. *Neural Network*, 2:359–366, 1989.