IDIAP RESEARCH REPORT

Bagging using the VMSE Cost Function

Vincent Lemaire [a]

IDIAP–RR 02-27

June 2002

SUBMITTED FOR PUBLICATION

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

[a] Telecommunication and Neural Techniques Group - France Telecom Research and Development - FTR&D/DTL/TIC - 2 Avenue Pierre Marzin - 22307 Lannion cedex FRANCE - *email: vincent.lemaire@francetelecom.com*

IDIAP Research Report 02-27

Bagging using the VMSE Cost Function

Vincent Lemaire

June 2002

submitted for publication

Abstract. A novel method is presented to enhance the generalization performances of multi-layer perceptrons used as discriminant networks. We clearly show how to modify the learning criterion in order to control the error distribution. We show that using this cost function, in a Bagging framework, the generalization performance increases. This method is tested and compared with the standard mean squared error criterion and is applied here to a benchmark problem.

1 Introduction

The choice of the capacity of a neural network is very important for its learning abilities and generalization. If the model is too simple, it cannot learn the desired function, if it is too complex, it cannot correctly generalize. In this paper, we argue that the cost function is also an important factor to control performances of a neural network.

The classical cost function used in classification and regression problems, is the mean squared error function [15]. We will name this method "MSE" in the following.

Numerous algorithms were proposed to speed up training, to find "good" learning rates, or to find a good early stopping method. We can note however that although one aims at reducing the errors made by the neural network, one has no control on the distribution of these during training. In fact, controlling the "shape" of the error distribution during training seems an interesting idea, because it is connected with the confidence on the predictions.

In other words, is it possible to control the convergence of the learning algorithm and in the same time control the shape of the error distribution such that it improves the robustness of learning? Does a control of the shape of the error distribution allow a control of the capacity of the network and thus a better generalization?

In [9] we gave a part of the answer but in a limited case which was used notably in [10, 1]. In this paper, we extent the cost function proposed in the framework of Bagging. The organization of the paper is as follows: in the next section, we briefly introduce why controlling the distribution during training can be interesting. In section 3 we present our cost function in a general setting, followed in section 4 by a comparative study on our cost function and the MSE cost function within the framework of Bagging, showing some experimental results on a benchmark database. A short conclusion then follows.

2 Classification Errors

When using MLP for classification, two kinds of errors can be computed: the squared error which is a function of the difference between the desired output and the obtained output and the classification error when an example is misclassified.

Bounds on the generalization error of composite classifier systems have been previously formulated [6, 16] and are based on the notion of the margin of classification (the distance of the estimation to the threshold decision). The size of the margin depends on the mean squared error but more generally on the distribution of the squared error and therefore on its variance (Figure 1). Consequently, the performance in terms of correct classification depends on the particular shape of the distribution of the squared error. Therefore, the choice of an appropriate cost function to control the shape of the distribution can be crucial to obtain an better solution to the classification problem.

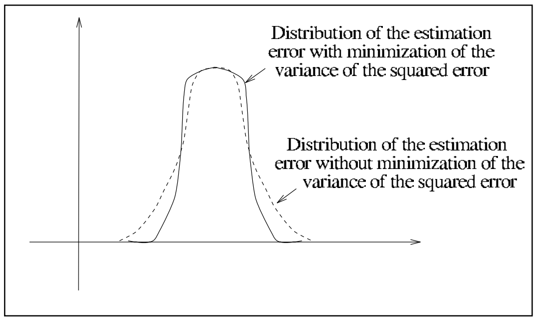
Distribution of the estimation error with minimization of the variance of the squared error
Distribution of the estimation error without minimization of the variance of the squared error

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 control the shape of the distribution of the squared error. This method takes into account a fourth order moment, the variance of the squared error, in the used cost function during training in order to improve generalization.

3 The VMSE Cost Function

3.1 Description

There are several ways to take into account the shape of the distribution of the errors without degradation of the global mean squared error. One of them is to add to the standard cost function a term related to the variance of the squared error.

The expression of the cost function hence becomes:

$$C_i^x = \alpha_{SE}\left((d_i^x - o_i^x)^2\right) + \alpha_{VSE}\left(\frac{1}{P}\sum_{a=1}^{P}\left((d_i^a - o_i^a)^2 - \frac{1}{P}\sum_{b=1}^{P}(d_i^b - o_i^b)^2\right)^2\right) \tag{1}$$

which can be written as the sum of two costs

$$(C_i^x)_{VMSE} = \alpha_{SE}\,(C_i^x)_{SE} + \alpha_{VSE}\,(C_i^x)_{VSE} \tag{2}$$

where P is the number of examples in the training set, o_i^x the output value of the neuron i after the presentation of the example x and d_i^x the desired output for the corresponding neuron. The first part of this expression is the cost on the squared error (SE) and the second part is the cost on the variance of the squared error (VSE).

As usual, the gradient of the cost function with respect to the parameters can be written as:

$$\frac{\partial C_i^x}{\partial W} = \frac{\partial C_i^x}{\partial o_i^x}\frac{\partial o_i^x}{\partial W} \tag{3}$$

with W the parameters of the neural networks. The computation of $\frac{\partial o_i^x}{\partial W}$ is the same than for the other cost functions.

The first term can be written as:

$$\begin{aligned}\frac{\partial C_i^x}{\partial o_i^x} &= \alpha_{SE}\left[\frac{\partial}{\partial o_i^x}\left((d_i^x - o_i^x)^2\right)\right] + \\ &\alpha_{VSE}\left[\frac{\partial}{\partial o_i^x}\left(\frac{1}{P}\sum_{a=1}^{P}\left((d_i^a - o_i^a)^2 - \frac{1}{P}\sum_{b=1}^{P}(d_i^b - o_i^b)^2\right)^2\right)\right]\end{aligned} \tag{4}$$

which is equal to:

$$\begin{aligned}\frac{\partial C_i^x}{\partial o_i^x} &= \alpha_{SE}\left[-2(d_i^x - o_i^x)\right] + \\ &\alpha_{VSE}\left[2\left(-2(d_i^x - o_i^x)\right)\left(\frac{1}{P}\left((d_i^x - o_i^x)^2 - \frac{1}{P}\sum_{b=1}^{P}(d_i^b - o_i^b)^2\right)\right)\right]\end{aligned} \tag{5}$$

which can also be written as:

$$\frac{\partial C_i^x}{\partial o_i^x} = \alpha_{SE}\left[-2(d_i^x - o_i^x)\right] + \alpha_{VSE}\left[-4(d_i^x - o_i^x)\left(\frac{(d_i^x - o_i^x)^2 - MSE}{P}\right)\right] \tag{6}$$

where MSE represents the mean squared error obtained on all the examples. We realize that this gradient can be seen as the usual gradient obtained, from the mean squared error, but corrected by a measure of the distance between the error on the example x and the error on all the examples. We named this training method "VMSE" for *Variance and Mean Squared Error*.

Since the MSE should be calculated after each modification of the weights, the computation is very expansive and could be very long. To circumvent this problem we propose to approximate it with the MSE computed at the previous step, hence:

$$\frac{\partial C_i^x}{\partial o_i^x} \approx \alpha_{SE}\left[\frac{\partial}{\partial o_i^x}\left((d_i^x - o_i^x)^2\right)\right] + \alpha_{VSE}\left[\frac{\partial}{\partial o_i^x}\left(\frac{1}{P}\sum_{a=1}^{P}\left((d_i^a - o_i^a)^2 - \frac{1}{P}\sum_{b=1}^{P}(d_{i_{I-1}}^b - o_{i_{I-1}}^b)^2\right)^2\right)\right] \tag{7}$$

The computation using this approximation give a similar result to the previous one (equation (6)):

$$\frac{\partial C_i^x}{\partial o_i^x} \approx \alpha_{SE}\left[-2(d_i^x - o_i^x)\right] + \alpha_{VSE}\left[-4(d_i^x - o_i^x)\left(\frac{(d_i^x - o_i^x)^2 - MSE_{I-1}}{P}\right)\right] \tag{8}$$

where $I-1$ represents the previous iteration. This algorithm can be viewed as a batch - stochastic algorithm. The only difference between equation (6) and equation (8) is the value of the MSE. In the comparative studies presented in this paper and to compare the influence of the added term we normalize the term on the variance of the squared error and we define the variable ν:

$$\nu = \frac{\alpha_{VSE}P}{\alpha_{SE}} \tag{9}$$

From equation (2) the learning rule can then be expressed, as for the other cost functions, as:

$$\Delta W = \beta\left(\alpha_{SE}\left(\frac{\partial (C_i^x)_{SE}}{\partial W}\right) + \alpha_{VSE}\left(\frac{\partial (C_i^x)_{VSE}}{\partial W}\right)\right) \tag{10}$$

with β the learning rate.

4 Comparative Study : Classification with Bagging Method

4.1 Bagging and VMSE Cost Function

Bagging [2] is a method for generating multiple versions of a predictor and using these to get an aggregated predictor. The aggregation averages over the versions when predicting a numerical outcome and does a plurality vote or an average when predicting a class. The multiple versions are formed by making bootstrap replicates of the learning set and using these as new learning sets. This method, among some others [6, 7, 13, 16, 17], has shown the interest to combine multiple versions of a predictor to improve the precision of the results. These methods are motivated by the bias-variance dilemma [2, 4, 5, 11, 14, 19].

Here the novel cost function do not minimize this kind of variance. We showed in [9] that, with the cost function described in this paper, the performance are improved for a single neural network. In the comparative study below the objective is to see if this improvement is preserved in a Bagging setup.

4.2 Experimental Conditions

The problem is a two class classification problem on credit[1]. The available database contains 690 examples each made of fifteen attributes and a target which indicates the class 1 or 2. The database is well balanced: 307 examples belong to class 1, 44.5 % and 383 belong to class 2, 55.5 %.

We used a 10-fold cross-validation method, on the original database, to find the good parameters of the neural network and the learning rate. Each neural network, in this section, is a multilayer perceptron, with standard sigmoidal functions, 15 input neurons, one hidden layer with 6 neurons and one output. The number of hidden units was selected for the MSE cost function, between 2 and 20. The learning rate is $\beta = 0.01$.

We then used exactly the same training process than the one mentioned in [12] : we created f bootstraps of the dataset. For each bootstrap we trained a neural network on the complete bootstrap. Every time, this process was done twice: one using classical MSE cost function and one using the VMSE cost function. We used the stochastic version on each cost function. The training process on a bootstrap is stopped when the cost does not decrease a lot compared to previous iteration. This early stopping mechanism is based only on the MSE criterion. The added term (V) in the VMSE cost function is hence used only as a regularization term.

The MSE, the VSE and the class error are computed on the complete original database with respect to the number of bootstraps. As the database contains missing values, they were simply replaced by the minimum of the corresponding variable on the training set [3, 8, 18].

4.3 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 with the added term, the variance of the squared error is decreased which increases the classification performance.

In this experiment we examined the influence of ν on two values to estimate how the added gradient interacts with the gradient of the squared error. Comparisons are performed for the global MSE, for the VSE and for the classification error rate.

We tried 5 values for ν: 0.01, 0.1, 1.0, 10.0 and 100.0. For $\nu = 0.1$ and $\nu = 0.01$ the results were the same for the two cost functions and for $\nu = 100.0$ the results deteriorate. The tested values of ν have been chosen with the experience acquired on this cost function on other databases. We recommend to try these five characteristic values of ν. The reader can read [9] for more details. In the following figures the results are labeled MSE for the MSE cost function which use only the squared error and VMSE1 and VMSE10 respectively for the VMSE cost function for $\nu = 1$ and $\nu = 10$.

Figure 2 shows the results obtained on the global MSE with the two cost functions. Figure 3 shows the results on the VSE. These results show that the added variance term interacts with the squared error term. A well chosen value of ν improves the minimization of the VSE. The minimization of the VSE, without degradation of the MSE, improves the performances. Figures 4 shows the effect of the added term in the cost function on the classification error rate.

[1]One can find this database and other informations on comparison conditions on: ftp://ftp.ics.uci.edu/pub/machine-learning-databases/credit-screening/

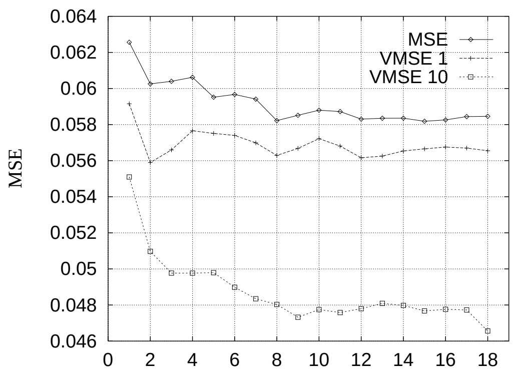

Figure 2: The mean squared error (MSE) obtained when optimizing the two cost functions with respect to the number of bootstraps. The results with the VMSE cost function are better than with the MSE cost function. The three curves follow the same improvement as the number of bootstraps increases. The improvement obtained with only one neural network is preserved as the number of bootstraps increases. The performance, with $\nu = 10.0$ and 15 bootstraps, compared to the MSE cost function results, are improved by 18 % (0.48 against 0.58).

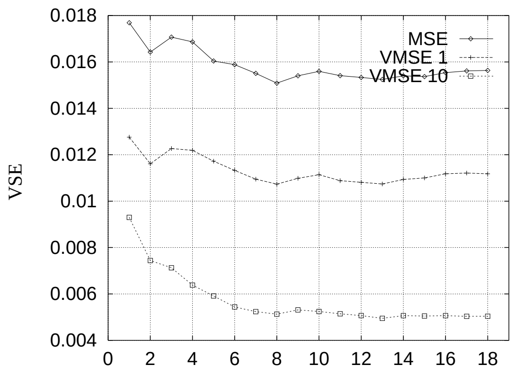

Figure 3: The variance of the squared error (VSE) obtained when optimizing the two cost functions with respect to the number of bootstraps. The results with the VMSE cost function, as for the MSE (Figure2) are better than with the MSE cost function. The three curves follow the same improvement as the number of bootstraps increases. The improvement obtained with only one neural network is preserved as the number of bootstraps increases. The performance, with $\nu = 10.0$ and 15 bootstraps, compared to the MSE cost function results, are improved by 54 % (0.05 against 0.11).

We notice that the obtained results with the VMSE method are better than with the MSE function both on the MSE, on the VSE and on the classification error rate. The control of the shape of the error distribution, realized with the minimization of the variance of the squared error, allows a better generalization. This improvement already observed on a first comparative study [9] which used one neural network is preserved as the number of neural networks increases. We see, on the different curves, that the improvement is preserved when the number of bootstraps increases. Each curve decreases in the same way.

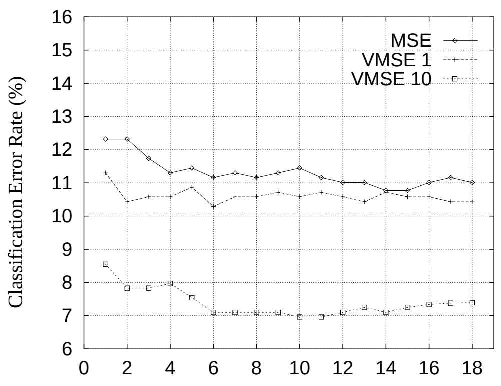

Figure 4: The classification error rate (%) obtained when optimizing the two cost functions with respect to the number of bootstraps. The results with the VMSE cost function, as for the MSE and the VSE (Figures 2 and 3), are better than with the MSE cost function. The three curves follow the same improvement as the number of bootstraps increases. The improvement obtained with only one neural network is preserved as the number of bootstraps increases. The performance, with $\nu = 10.0$ and 15 bootstraps, compared to the MSE cost function results, are improved by 36 % (7 against 10.8).

5 Conclusion

In this paper we have presented a new cost function to train ensemble of neural network with Bagging that gave very good results compared to classical MSE either in terms of mean squared error or classification performance.

We showed how to modify the training criterion to control the shape of the error distribution during training.

Moreover, the method requires very small changes. The training time does not change a lot because the MSE required is already computed in the MSE cost function (at the previous iteration). We just add 2 multiplications and a substraction for each example trained. The results can not be degraded if the parameter ν is small enough.

These results are extremely encouraging and suggest that the proposed method could allow training other ensemble method. Future work will address at least one question: Does the approach work well for other types of ensemble methods, as adaboost, or mixture of experts?

Acknowledgments

The author of this paper works for France Telecom Research and Development but was invited at IDIAP (Dalle Molle Institute for Perceptual Artificial Intelligence, www.idiap.ch) for six months in the framework of a collaboration. The author would like to thank Samy Bengio for discussions. The VMSE cost function is now implemented in the current development version of the machine learning library TORCH (www.torch.ch) and will be available in a future release of this library.

References

[1] O. Bernier, M. Collobert, R. Fraud, 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] L. Breiman. Bagging predictors. Technical Report TR-421, University of California, Berkley, 1994.

[3] F. Fessant and S. Midenet. A knowledge-based parser: Neural network bases approaches .development of a neural network based imputation system for travel diqry data. Technical report, Test: Technologies for European Surveys of Travel Behaviour, 1999. Deliverable D5-B.

[4] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias-variance dilemma. *Neural Computation*, 4:1–58, 1992.

[5] L. K. Hansen and P. Salamon. Neural networks ensembles. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 12, pages 993–1001, 1990.

[6] S. Holger and Y Bengio. Training method for adaptative boosting of neural networks. In *Neural Information Processing System*, 1998.

[7] R. Kohavi and G. H. John. Automatic parameter selection by minimizing estimated error. In Morgan Kauffman, editor, *International Conference on Machine Learning*, volume 12, pages 304–311, San Francisco, 1995.

[8] K Lakshminarayan, S. Harp, R. Goldman, and T. Samad. Imputation of missing data using machine learning techniques. In *KDD*, 1996.

[9] V. Lemaire, O. Bernier, D. Collobert, and F. Clrot. A new method to increase the margin of multilayer perceptrons. *Neural Processing Letter*, 11(1):7–15, 2000.

[10] V. Lemaire and F. Clérot. Estimation of the blocking probabilities in an ATM network node using Artificial Neural Networks for Connection Admission Control. In *International Teletraffic Congress*, volume 16, Edinburgh, 1999.

[11] M. P. Perrone. *Improving Regression Estimation: Averaging Methods for Variance Reduction with Extensions to General Convex Measure Optimization.* PhD thesis, Brown University, Institute - Brain and Neural Systems, 1993.

[12] J. R. Quinlan. Bagging, Boosting and C4.5. Technical report, University of Sydney, 1998.

[13] H. Ragavan and L. Rendell. Lokkahead feature construction for learning hard concepts. In Morgan Kauffman, editor, *International Conference on Machine Learning*, volume 10, pages 252–259, San Francisco, 1993.

[14] Raviv, Y. and Intrator, N. Bootstrapping with noise: An effective regularization technique. *Connection Science*, 8:355–372, 1996.

[15] D. E. Rumelhart, G. E. Hinton, 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.

[16] 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/.

[17] P. E. Utgogg and C. Brodley. An incremental method for finding multivariate splits for decision trees. In Morgan Kauffman, editor, *International Conference on Machine Learning*, volume 7, pages 58–65, San Francisco, 1990.

[18] P. Vamplew, D. Clark, A. Adams, and J. Muench. Techniques for dealing with missing values in feedforward networks. In *ACNN*, pages 250–254, 1996.

[19] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.