Batch-Sequential Algorithm for Neural Networks Trained with Entropic Criteria*

Jorge M. Santos[1,3], Joaquim Marques de Sá[1], and Luís A. Alexandre[2]

[1] INEB - Instituto de Engenharia Biomédica
[2] IT - Networks and Multimedia Group, Covilhã
[3] Instituto Superior de Engenharia do Porto, Portugal
jms@isep.ipp.pt, jmsa@fe.up.pt, lfbaa@di.ubi.pt

Abstract. The use of entropy as a cost function in the neural network learning phase usually implies that, in the back-propagation algorithm, the training is done in batch mode. Apart from the higher complexity of the algorithm in batch mode, we know that this approach has some limitations over the sequential mode. In this paper we present a way of combining both modes when using entropic criteria. We present some experiments that validates the proposed method and we also show some comparisons of this proposed method with the single batch mode algorithm.

1 Introduction

In our previous work we introduced the use of Entropy as cost function in the learning process of Multi Layer Perceptrons (MLP) for classification [1]. This method computes the entropy of the error between the output of the neural network and the desired targets as the function to be minimized. The entropy is obtained using probability density estimation with the Parzen window method which implies the use of all available samples to estimate its value. This fact forces the use of the batch mode in the Back-propagation algorithm limiting the use of, in some cases most appropriate, sequential mode. To overcome this limitation we propose a new approach that combines these to modes (the batch and the sequential) to try to use their mutual advantages. What we call the batch-sequential mode divides, in each epoch, the training set in several groups and sequentially presents each one to the learning algorithm to perform the appropriate weight updating.

The next section of this work introduces the Error Entropy Minimization Algorithm and several optimizations to achieve a faster convergence by manipulating the smoothing parameter and the learning rates. Section 3 presents the new batch-sequential algorithm and section 4 several experiments that show the applicability of the proposed method. In the final section we conclude with some discussion of the paper.

* This work was supported by the Portuguese Fundação para a Ciência e Tecnologia(project POSI/EIA/56918/2004).

W. Duch et al. (Eds.): ICANN 2005, LNCS 3697, pp. 91–96, 2005.

2 The EEM Algorithm

The use of entropy and related concepts in learning systems is well known. The Error Entropy Minimization concept was introduced in [2] and, using the same approach, we introduced in [1] the Error Entropy Minimization Algorithm for Neural Network Classification. This algorithm uses the entropy of the error between the output of the neural network and the desired targets as the function to be minimized in the training phase of the neural network. Despite the fact that we use Renyi's Quadratic Entropy we may as well use other kinds of entropy measures, like Shannon's entropy, as in [4].

Let $y = \{y_i\} \in \mathbb{R}^m$, $i = 1, ..., N$, be a set of samples from the output vector $Y \in \mathbb{R}^m$ of a mapping $\mathbb{R}^n \mapsto \mathbb{R}^m : Y = g(w, X)$, where w is a set of neural network weights, X is the input vector and m is the dimensionality of the output vector. Let $d = d_i \in \{-1, 1\}^m$ be the desired targets and $e_i = d_i - y_i$ the error for each data sample. In order to compute the Renyi's Quadratic Entropy of e we use the Parzen window probability density function (pdf) estimation using Gaussian kernel with zero mean and unitary covariance matrix, $G(e, I) = \frac{1}{(2\pi)^{\frac{m}{2}}} exp\left(-\frac{1}{2}e^T e\right)$. This method estimates the pdf as

$$f(e) = \frac{1}{Nh^m} \sum_{i=1}^{N} G(\frac{e - e_i}{h}) \tag{1}$$

where h is the bandwidth or smoothing parameter.

Renyi's Quadratic Entropy of the error can be estimated, applying the integration of gaussian kernels [5], by

$$\begin{aligned} \hat{H}_{R2}(e) &= -\log \left[\frac{1}{N^2 h^{2m-1}} \sum_{i=1}^{N} \sum_{j=1}^{N} G(\frac{e_i - e_j}{h}, 2I) \right] \\ &= -\log V(e) \end{aligned} \tag{2}$$

The gradient of $V(e)$ for each sample i is:

$$F_i = -\frac{1}{2Nh^{2m+1}} \sum_{j=1}^{N} G(\frac{e_i - e_j}{h}, 2I)(e_i - e_j) \tag{3}$$

The update of the neural network weights is performed using the back-propagation algorithm with $\Delta w = -\eta \frac{\partial V}{\partial w}$.

One of the first difficulties in estimating the entropy is to find the best value for h in pdf estimation. In our first experiments with this entropic approach the value of the smoothing parameter was experimentally selected. Latter, we have developed a formula to obtain an appropriate value for h, as a function of the number of samples and the dimensionality of the neural network output (related

with the number of classes in the classification problem). This formula, proposed in [6]

$$h_{op} = 25\sqrt{\frac{m}{N}} \tag{4}$$

gives much higher values than those formulas usually proposed for probability density function estimation and gives very good results for the EEM algorithm.

3 Batch-Sequential Algorithm

The Batch-Sequential algorithm tries to combine the two methods applied in the back propagation learning algorithm: the *sequential* mode, also referenced as on-line or stochastic, where the update is made for each sample of the training set, and the *batch* mode, where the update is performed after the presentation of all samples of the training set.

We know that, the estimated pdf approximates the true pdf as $N \rightarrow \infty$ but, in the EEM algorithm, we only need to compute the entropy and its gradient; we do not need to estimate the probability density function of e. This is a relevant fact because, in the gradient descent method, more important than computing with extreme precision the gradient is to get accurately its direction. Also, the computation with extreme accuracy of the probability density function causes the entropy to have high variability. This fact could lead to the occurrence of local minima. The sequential mode updating of the weights leads to a sample by sample *stochastic* search in the weight space implying that becomes less likely for the back-propagation algorithm to be trapped in local minima [7]. However, we still need some samples to estimate the entropy what limits the use of the sequential mode. Other advantage of the sequential mode occurs when there are some redundancy in the training set. The batch mode also presents some advantages: the gradient vector is estimated with more accuracy guarantying the convergence to, at least, a local minima and the algorithm is more easily parallelized than using sequential mode.

In order to make use of the advantages of both modes and also to speedup the algorithm, we developed a batch-sequential algorithm consisting of the splitting of the training set in several groups that are presented to the algorithm in a sequential way. In each group we apply the batch mode.

Let $\{Ts\}$ be the training set of a given data set and $\{Ts_j\}$ the subsets obtained by randomly dividing Ts in several groups with an equal number of samples, such as

$$\#Ts = n + \sum_{j=1}^{L} \#Ts_j \tag{5}$$

where L is the number of subsets and n the remainder. Leaving, in each epoch, some samples out of the learning process (when $n \neq 0$) is not significant because those samples will most likely be included in the next epoch. The partition of the

training set in subsets being performed in a random way reduces the probability of the algorithm getting trapped in local minima. The subsets are sequentially presented to the learning algorithm, that applies to each one, in batch mode, the respective computation and subsequent weight update. The pseudo code for the Error Entropy Minimization Batch-Sequential algorithm (EEM-BS) is presented in Table 1.

Table 1. Pseudo-code for the EEM-BS Algorithm

```
For k:=1 to number of epochs
    Create L subsets of Ts
    For j:=1 to L
        - Compute the error entropy gradient of Ts_j applying formula 3
        - Perform weight update
    End For
End For
```

One of the advantages in using the batch-sequential algorithm is the decrease of the algorithm complexity. The complexity of the original EEM algorithm, due to formulas 2 and 3, is $O(Ts^2)$. We clearly see that, for large training sets, the algorithm is highly time consuming. With the EEM-BS algorithm the complexity is proportional to:

$$L\left(\frac{Ts}{L}\right)^2 \tag{6}$$

Therefore, the complexity ratio of both algorithms is:

$$\frac{Ts^2}{L(\frac{Ts}{L})^2} = L \tag{7}$$

which means that, in terms of computational processing time, we achieve a reduction proportional to L. For a complete experiment, similar to the one presented in the next section with the data set "Olive", we reduce the processing time from about 30 to 6 minutes in our machine.

The number of subsets, L, is determined by the size of the data set. If, in a given problem, the training set has a large number of data samples, we can use a higher number of subsets than if we have a small training set. We recommend the division of the training set in a number of subsets with a number of samples not less than 40, even though we had some good results with less elements.

In order to perform the experiments with the batch-sequential algorithm, we tried to use the optimization proposed in [3], the EEM-VLR. This optimization is based on the use of a global variable learning rate (VLR) during the training phase, as a function of the entropy value in consecutive iterations. Since this optimization compares H_{R2} of a certain epoch with the same value in the previous one, we could not use it because, in each epoch, we use different sets of samples and, by this simple fact, we would have different values of H_{R2}. To overcome this limitation we implemented a similar process, also using variable learning

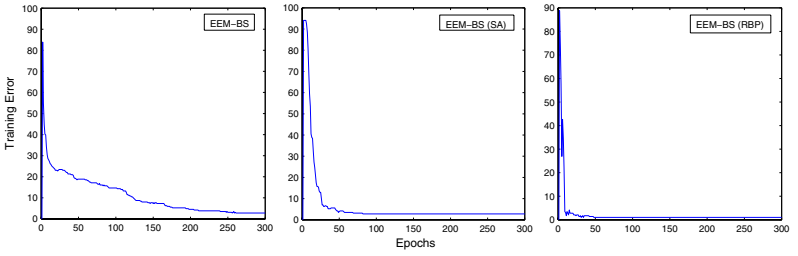

Fig. 1. Training Error for the EEM-BS and the two optimizations

rate, but this time, the variation of the learning rate is done for each neural network weight by comparing the respective gradient in consecutive iterations (EEM-BS(SA)). This approach was already used in back-propagation with MSE [8]. We also used, for the same purpose of speeding up the convergence, the combination of this implementation with the resilient back-propagation, achieving very good results (EEM-BS(RBP)). Examples of the training phase for the three different methods, with data set "Olive", are depicted in Fig.1.

4 Experiments

In order to establish the validity of the proposed algorithm we performed several classification experiments, comparing the results obtained with the EEM-BS algorithm and with the simple EEM-VLR algorithm. The characteristics of the data sets used in the experiments are summarized in Table 2.

In all experiments we used (I, n_h, m) MLP's, where I is the number of input neurons, n_h is the number of neurons in the hidden layer and m is the number of output neurons. We applied the cross-validation method using half of the data for training and half for testing. The experiments for each data set were performed varying the number of neurons in the hidden layer, the number of subsets used and the number of epochs. Each result is the mean error of 20 repetitions. In Table 3 we only present the best results for each experiment with 4 and 8 subsets for the EEM-VLR and the EEM-BS algorithms.

The results of EEM-BS algorithm are, in some cases, even better than those of EEM-VLR. The complexity of the neural networks for each experiment is very

Table 2. Data sets used for the experiments.

Data set	*# Samples*	*# Features*	*# Classes*
Ionosphere	351	33	2
Olive	572	8	9
Wdbc	569	30	2
Wine	178	13	3

Table 3. Results for EEM-VLR and EEM-BS (Tpe: Time per epoch $\times 10^{-3}$ sec.)

Ionosphere	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	12.06 (1.11)	-	12	40	16.7
EEM-BS	12.00 (1.22)	4	16	80	6.4
EEM-BS	12.22 (1.14)	8	16	60	4.8

Olive	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	5.04 (0.53)	-	25	200	77.7
EEM-BS	5.17 (0.51)	4	30	140	17.6
EEM-BS	5.24 (0.70)	8	20	180	12.8

Wdbc	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	2.33 (0.37)	-	4	40	38.7
EEM-BS	2.31 (0.35)	5	10	60	13.6
EEM-BS	2.35 (0.48)	8	10	40	9.6

Wine	*Error (Std)*	*L*	*n_h*	*Epochs*	*Tpe*
EEM-VLR	1.83 (0.83)	-	14	40	5.8
EEM-BS	1.88 (0.80)	4	16	60	3.2
EEM-BS	1.88 (0.86)	8	16	60	2.5

similar for both algorithms what reenforces the validity of the proposed method. Since the best results were obtained with different neural network complexity we present in column *Tpe* the processing time per epoch for each algorithm.

5 Conclusions

We presented, in this paper, a way of combining the sequential and batch modes when using entropic criteria in the learning phase, taking profit of the advantages of both methods. We show, using experiments, that this is a valid approach that can be used to speed-up the training phase, maintaining a good performance.

References

1. Jorge M. Santos, Luis A. Alexandre, and Joaquim Marques de Sá. The Error Entropy Minimization Algorithm for Neural Network Classification. *Int. Conf. on Recent Advances in Soft Computing*, pages 92–97, 2004.
2. D. Erdogmus and J. Prncipe. An error-entropy minimization algorithm for supervised training of nonlinear adaptive systems. *Trans. On Signal Processing*, 50(7):1780–1786, 2002.
3. Jorge M. Santos, Joaquim Marques de Sá, Luis A. Alexandre, and Fernando Sereno. Optimization of the Error Entropy Minimization Algorithm for Neural Network Classification. In C.H.Dagli, A. L. Buczak, D. L. Enke, M. J. Embrechts, and O. Ersoy, editors, *Intelligent Engineering Systems Through Artificial Neural Networks*, volume 14, pages 81–86. ASME Press, 2004.
4. Luis M. Silva, Joaquim Marques de Sá, and Luis A. Alexandre. Neural Network Classification using Shannon's Entropy. In *European Symposium on Artificial Neural Networks (Accepted for publication)*, 2005.
5. D. Xu and J. Princpe. Training mlps layer-by-layer with the information potential. In *Intl. Joint Conf. on Neural Networks*, pages 1716–1720, 1999.
6. Jorge M. Santos, Joaquim Marques de Sá, and Luis A. Alexandre. Neural Networks Trained with the EEM Algorithm: Tuning the Smoothing Parameter. In *6th WSEAS Int. Conf. on Neural Networks, (accepted)*, 2005.
7. Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, 2 edition, 1999.
8. F. Silva and L. Almeida. Speeding up backpropagation. In Eckmiller R., editor, *Advanced Neural Computers*, pages 151–158, 1990.