OPTIMIZATION OF THE ERROR ENTROPY MINIMIZATION ALGORITHM FOR NEURAL NETWORK CLASSIFICATION

JORGE M. SANTOS
Instituto de Engenharia Biomédica,
Porto, Portugal.
Instituto Superior de Engenharia do
Porto, Dep. Matemática, Porto,
Portugal; jms@isep.ipp.pt

JOAQUIM MARQUES DE SÁ
Instituto de Engenharia Biomédica,
Porto, Portugal.
Faculdade de Engenharia da
Universidade do Porto, DEEC,
Porto, Portugal

LUÍS A. ALEXANDRE
Instituto de Engenharia Biomédica,
Porto, Portugal.
IT - Networks and Multimedia
Group, Covilhã, Portugal

FERNANDO SERENO
Escola Superior de Educação,
Instituto Politécnico do Porto, Porto,
Portugal

ABSTRACT

One way of using entropy criteria in learning systems is to minimize the entropy of the error between the output of the learning system and the desired targets. In our last work, we introduced the Error Entropy Minimization (EEM) algorithm for neural network classification. There are some sensible aspects in the optimization of the EEM algorithm: the size of the Parzen Window (smoothing parameter) and the value of the learning rate parameter are the two most important. In this paper, we show how to optimize the EEM algorithm by using a variable learning rate during the training phase.

INTRODUCTION

The Error Entropy Minimization (EEM) algorithm minimizes the Reniy's Quadratic Entropy of the error between the output of the neural network and the desired targets. Reniy's Quadratic Entropy is used because, when using a non-parametric estimate of the probability density function (pdf) by the Parzen Window method, a simplified expression for the entropy is obtained.

The size of the Kernel window, *h*, used in Parzen Window method is one of the two most sensible aspects in the implementation of the EEM algorithm. The other important aspect is the backpropagation learning rate. We implemented a variable learning rate, similar to the one used in the MSE algorithm, and the results were better than the ones obtained with the previous algorithm.

RENYI'S QUADRATIC ENTROPY AND BACK-PROPAGATION ALGORITHM

Renyi extended the concept of entropy introduced by Shanon and defined the Renyi's α entropy, applied to continuos random variables, as (Renyi 1976):

$$H_{R\alpha} = -\log \int_C \left[f(x) \right]^{\alpha} dx \qquad (1)$$

Renyi's Quadratic Entropy, $\alpha = 2$, in conjunction with the Parzen Window pdf estimation with gaussian kernel allows, the determination of entropy in a non-parametric, practical and computationally efficient way. The Parzen window method estimates the pdf $f(x)$ as

$$f(x) = \frac{1}{Nh^m} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right) \tag{2}$$

where N is the number of data points, K is a kernel function, m is the dimensionality of vectors x ($x \in \Re^m$) and h the bandwidth or smoothing parameter. We usualy use the simplest Gaussian kernel with zero mean and Σ equal to $\boldsymbol{I}$ (the identity matrix)

$$G(x, \boldsymbol{I}) = \frac{1}{(2\pi)^{\frac{m}{2}}} \exp\left(-\frac{1}{2} x^T x\right) \tag{3}$$

Let $a = a_i \in \Re^m$, be a set of samples from x. Reniy's Quadratic Entropy can be estimated, applying the integration of Gaussian kernels (Xu and Principe, 1999), by

$$\hat{H}_{R2} = -\log\left[\frac{1}{N^2 h^{2m-1}} \sum_{i=1}^{N} \sum_{j=1}^{N} G(\frac{a_i - a_j}{h}, 2\boldsymbol{I})\right] = -\log V(a) \tag{4}$$

Principe (1998) calls $V(a)$ the *information potential* in analogy with the potential field in physics. For the same reason he also calls the derivative of $V(a)$ the *information force F*. Therefore

$$F_i = -\frac{1}{2N^2 h^{2m+1}} \sum_{j=1}^{N} G(\frac{a_i - a_j}{h}, 2\boldsymbol{I})(a_i - a_j) \tag{2}$$

This *information force* at each point is backpropagated into the MLP in the same way as with the MSE algorithm.

THE EEM ALGORITHM AND ITS OPTIMIZATION

In order to make neural network classification we use the information-theoretic concepts, applying an entropy approach to the classification task, using, as cost function, the entropy minimization of the error between the output of the network and the desired targets: the EEM algorithm.

The error entropy minimization approach, introduced by Erdogmus and Principe (2002) in time series prediction, states that Renyi's Quadratic Entropy of the error, with pdf approximated by Parzen window with Gaussian kernel, has minima along the line where the error is constant over the whole data set. Also the global minimum of this entropy is achieved when the pdf of the error is a Dirac delta function.

In classification problems we proved (Santos et al., 2004) a more strict result than simply the equality of the errors. As a matter of fact, we proved that by imposing some conditions to the output range and target values, the EEM algorithm drives all errors towards zero. Those conditions consist in a relation between the set of the output of the neural network and the set of the desired targets. In an unidimensional case the targets should be defined as $t \in \{-p, p\}$ and the output of the network as $y \in [-p, p]$.

The gradient of Renyi's Quadratic Entropy of the Error is back-propagated into the MLP in the same way as with the MSE algorithm. The updating of the neural network weights is performed using $\Delta w = \pm\eta \partial V / \partial w$. The choice of the learning rate η is one of the most important aspects in the implementation of the EEM algorithm. We will see in the next section how the variation of the learning rate along the training process can yield good results.

We started the optimization of EEM algorithm by trying to make the kernel smoothing parameter (kernel window size) *h* variable along the training process, namely proportional to the error variance. This strategy was based on the fact that, as we approach the optimal solution, the errors tend to zero (m-tuples of zeros) and so, it makes sense to decrease *h* since the points are all near. The entropy and the subsequent *information force* depends on the values of *h* (smaller *h* originates higher entropies and information forces), and that could be opposite to our objective of minimization of the entropy of the error. However, if, in each instance of the algorithm, we minimize the entropy function we can, at least theoreticaly, get an optimal solution. The problem in the variablility of *h* is that, in the proximity of the minimum training error the algorithm is very unstable and looses the capability of convergence.

In the next optimization phase of the EEM algorithm we used a variable learning rate and a fixed smoothing parameter *h*.

The variability of the learning rate follows a simple but efective rule. If the entropy of the error decreases between two consecutive epochs of the training process then the algorithm produces an increase in the learning rate parameter. Similarly, if the entropy of the error increases between two consecutive epochs then the algorithm produces a decrease in the learning rate parameter and, furthermore, it restarts the update step.

The rule for learning rate variability is:

$$\eta^{(n)} = \begin{cases} \eta^{(n-1)}.u & \text{if } H_{R2}^{(n)} < H_{R2}^{(n-1)} \\ \eta^{(n-1)}.d & \text{if } H_{R2}^{(n)} > H_{R2}^{(n-1)} \end{cases} \quad (3)$$

where $\eta^{(n)}$ and $H_{R2}^{(n)}$ are the learning rate and the Renyi's Quadratic entropy of the error in *n*th iteration, respectively, and *u* and *d* are the increasing and decreasing factors.

We performed several experiments in order to find good values for *u* and *d*. We used a bi-dimentional data set (Jacobs(1991)), with 608 data points, 4 classes, consisting of two separable groups of two classes each. The two classes in each group are non separable.

In Fig. 1 we show the training phases with fixed learning rate, FLR, (doted lines) and with variable learning rate, VLR, (solid lines), of two experiments that have produced the least classification errors. The use of VLR produces a continuos decreasing entropy curve and a minimum training error is achieved.

In Table 1 we present the results of the experiments made with diferent values for *u* and *d*. The column (*restart*) indicates the number of times that the algorithm restarts the update step. These experiments suggest that, if the algorithm produces an increase on the entropy then the learning rate should be decreased by a considerable factor. Based in the several tests that we performed and in the fact that our errors are allways limited to a restrict set, due to the conditions mentioned in (Santos et al., 2004), we recommend a value around 0.2 for *d* and around 1.2 for *u*. However this issue deserves further extensive tests with multidimentional errors to support this recommendation. The solid line in Fig. 1 represents a case with $d = 0.2$ and $u = 1.2$.

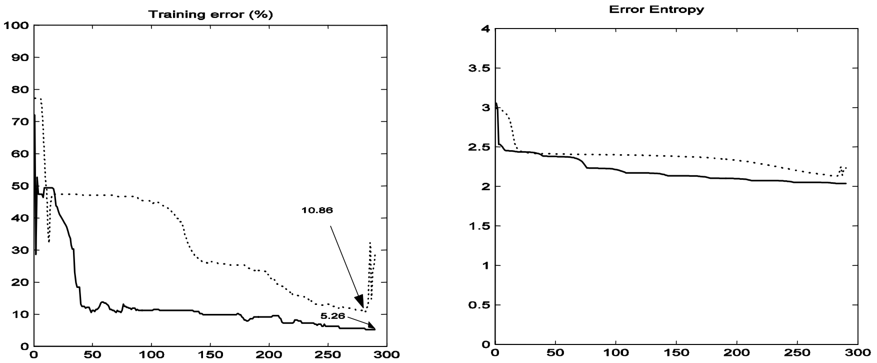

Fig. 1 - Two best results for FLR (doted) and VLR (solid)

Table 1 - Results for the variable learning rate

u	*d*	*restart*	Training Error	*u*	*d*	*restart*	Training Error
1.2	0.2	36	5.26	1.6	0.2	90	5.59
1.2	0.4	65	5.26	1.6	0.4	154	5.26
1.2	0.6	112	5.59	1.6	0.6	279	5.59
1.2	0.8	256	5.92	1.6	0.8	640	5.26
1.4	0.2	64	5.59	1.8	0.2	112	5.59
1.4	0.4	115	5.26	1.8	0.4	193	5.59
1.4	0.6	197	24.34	1.8	0.6	352	5.59
1.4	0.8	465	5.59	1.8	0.8	801	5.26

EEM-VLR VERSUS MSE-VLR

We made a first experiment, using multilayer perceptrons (MLP), to show the application of the Error Entropy Minimization with Variable Learning Rate (EEM-VLR) algorithm to data classification and compare it with Mean Square Error with Variable Learning Rate (MSE-VLR) algorithm.

In this experiment we used the data set described in the previous section. Several MLP's with 2 inputs, *n* neurons in the hidden layer and 2 outputs (2:n:2), were trained and tested 40 times, 300 epochs, using the EEM-VLR and also the MSE-VLR. We made *n* vary from 3 to 18. Each time, half of the data set was randomly chosen for training and the other half for testing. Then the data sets were used with inverted roles (the original training set became the test set and the original test set became the training set). The results of this experiment are shown in Table 2.

Table 2 – Classification errors for EEM-VLR and MSE-VLR

	Number of neurons in hidden layer																
	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	
EEM-VLR	9,68	**8,18**	8,54	8,73	9,22	8,67	9,03	8,77	9,90	9,30	10,16	10,01	10,14	11,50	10,72	12,68	Mean
	3,31	2,30	2,81	2,51	4,60	2,60	2,51	1,88	3,55	2,56	2,83	2,50	2,16	5,54	1,94	4,47	Std
MSE-VLR	24,46	17,49	15,86	13,80	14,35	12,29	12,46	11,95	11,34	10,67	9,44	9,46	8,61	9,22	9,77	10,61	Mean
	9,84	9,10	8,61	7,61	8,26	6,58	7,48	6,64	6,66	6,01	3,84	3,63	1,30	3,48	4,97	6,18	Std

We see in Table 2 that EEM-VLR algorithm produces better results comparing to MSE-VLR algorithm. Only for larger values of the number of neurons in the hidden layer the MSE-VLR algorithm produces better results. However this could be due to overfitting. We also see that similar results are achieved with less complex MLP's in the EEM-VLR algorithm (Fig. 2). This sugests that, with EEM-VLR, we need less complex neural nerworks, compared to MSE, in order to solve a particular classification problem.

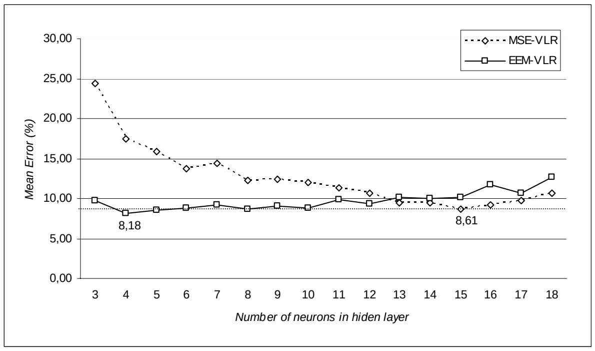

Fig. 2 - Classification mean error between EEM-VLR and MSE-VLR algorithms

Two more experiments were made applying the two algorithms in real data sets. These data sets are publicly available (*diabetes* can be found in Marques de Sá (2003) and *wine* can be found in the UCI repository of machine learning databases, http://www.ics.uci.edu/˜mlearn/MLRepository.html. Table 3 contains a summary of the characteristics of these data sets.

Table 3 - The data sets used in second experiments

Data set	N. points	N. Features	N. Classes
Diabetes	768	8	2
Wine	178	13	3

Several MLP's were trained and tested 10 times, 120 epochs, with d=0.2 and u=1.2. Each time, half of the data sets were randomly chosen for training and the other half for testing. Then the data sets were used with inverted roles. The results of this experiments are shown in Table 4. Again, the best results (bold), in this three classification problems, were achieved with the EEM-VLR algorithm.

Table 4 – Classification errors for the EEM-VLR and the MSE-VLR algorithms.

		Number of neurons in hidden layer							
		4	5	6	7	8	9	10	
Diabetes	EEM-VLR		23,8	24,1	24,1	23,9	*24,3*	**23,6**	Mean
			1,04	1,33	0,9	0,71	1,42	0,86	Std
	MSE-VLR		25,1	24,7	24,4	23,9	24	24,1	Mean
			1,8	1,8	1,06	1,18	0,95	1,2	Std
Wine	EEM-VLR	**1,94**	2,5	2,47	*2,44*	2,16	2,22	2,31	Mean
		0,72	1,01	1,2	1	0,92	0,83	0,51	Std
	MSE-VLR	3,03	3,2	3,06	2,39	2,92	2,5	2,95	Mean
		1,08	1,83	1,43	1,5	1,07	1,35	1,29	Std

CONCLUSIONS

In this paper we present an improved version of the EEM algorithm, using a variable learning rate parameter, which we call the EEM-VLR algorithm. This algorithm shows a very good performance achieving good results in classification problems when compared to MSE-VLR algorithm.

Despite the fact that the results using variable smoothing parameter h were not satisfactory we think that, this issue deserves further investigation, namely on how to combine the two variable factores, η and h in order to produce an even more efficient algorithm.

REFERENCES

Erdogmus D. and Principe J., 2002, "An Error-Entropy Minimization Algorithm for Supervised Training of Nonlinear Adaptive Systems", Trans. On Signal Processing, Vol. 50, No. 7, pp. 1780-1786.

Jacobs, R., Jordan, M., Nowlan, S. and Hinton, G., 1991, "Adaptive mixtures of local experts", Neural Computation, pp.79-87.

Marques de Sá, J., 2003, Applied statistics using SPSS, STATISTICA and MATLAB, Springer.

Principe J., Fisher J. and Xu D., 1998, "Information-Theoretic Learning", Computational NeuroEngineering Laboratory, University of Florida, Florida.

Renyi A., 1976, "Some Fundamental Questions of Information Theory", Selected Papers of Alfred Renyi, Vol. 2, pp. 526-552.

Santos J., Alexandre L. and Marques de Sá J.,2004, "Neural network Classification using Error Entropy Minimization", submitted to the Int. Conf. on Recent Advances in Soft Computing, Nottingham, United Kingdom.

Silva, F. and Almeida, L., 1990, "Speeding up Backpropagation", Advanced Neural Computers, Eckmiller R. (Editor), pp. 151-158.

Xu D. and Principe J., 1999, "Training MLPs layer-by-layer with the information potential", Intl. Joint Conf. on Neural Networks, pp.1716-1720.