ESANN'2005 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 27-29 April 2005, d-side publi., ISBN 2-930307-05-6.

Neural Network Classification using Shannon's Entropy

Luis M. Silva[1], J. Marques de Sá[1,2] and Luis A. Alexandre[1,3] *

1- INEB - Instituto de Engenharia Biomédica, Lab. Sinal e Imagem Biomédica
Campus da FEUP,Rua Dr. Roberto Frias, s/n, 4200-465 Porto - Portugal

2- Faculdade de Engenharia da Universidade do Porto - DEEC
Rua Dr. Roberto Frias, s/n, 4200-465 Porto - Portugal

3- IT - Networks and Multimedia Group
Covilhã - Portugal

Abstract. The last years have witnessed an increasing attention to entropy-based criteria in adaptive systems. Several principles were proposed based on the maximization or minimization of entropic cost functions. We propose a new type of neural network classifiers with multilayer perceptron (MLP) architecture, but where the usual mean square error minimization principle is substituted by the minimization of Shannon's entropy of the differences between the MLP's output and the desired target. The backpropagation algorithm is optimized with a variable learning rate and tested in five well known datasets. The results show a very good performance of MLPs trained with Shannon's entropy when compared with the mean square error and cros-entropy criteria.

1 Introduction

The most commonly used cost function for adaptive systems has been the mean square error (MSE). The choice is justified by the assumption that most real-life random processes can be explained by the Gaussian distribution and its first and second order statistics. However, this Gaussianity assumption is very restrictive. This has led to the search for more appropriate criteria taking advantage of high-order statistical behaviours. Entropy was introduced as a measure of average information contained in a distribution. For a continuous random variable X, (differential) entropy is defined as

$$H(X) = -\int_{-\infty}^{\infty} f(x) \log f(x) dx \tag{1}$$

where f is the probability density function of X. Entropy and the related concepts of mutual information and Kulback-Leibler divergence have been used in learning systems (supervised or unsupervised) in several ways. Recently, Príncipe and co-workers, proposed new approaches to using entropic criteria

*This work was supported by the Portuguese FCT-Fundação para a Ciência e a Tecnologia (project POSI/EIA/56918/2004). First author is also supported by FCT's grant SFRH/BD/16916/2004.

[1, 2], in particular, the minimization of the Rényi's second order entropy of the difference between the MLP output and the desired target (the error). The minimization of error entropy implies a reduction of the expected information contained in the error, which leads to a maximization of the mutual information between the desired target and the model output [1]. This means that the network is learning the target variable. This procedure was applied sucessfuly in regression, time series prediction and feature extraction by Príncipe and coworkers. Santos *et al.* [3, 4], proposed the use of this idea to neural network classifiers. Their results showed that entropy generally performs better than MSE in terms of minimum classification test error. In this paper we propose the use of Shannon's entropy in the same setting as the work of Santos *et al.*. The benefits of using Shannon's entropy arise from the well-known information meaning as well as its unique information measure properties.

2 Shannon entropy for multi-layer perceptrons

Consider an MLP with one hidden layer with output y and a target variable (class membership for each example in the dataset), t. For each example we measure the error using $e(n) = t(n) - y(n),\ n = 1, \ldots, N$ where N is the total number of examples. We only consider the two-class problem; thus, as in [3] we set $t \in \{-1, 1\}$ and a single output unit with $y \in [-1, 1]$. The proposed backpropagation algorithm does not use expression (1) directly as a cost function, but, instead it uses a Shannon's entropy estimator with mean square consistency [5], given by

$$\hat{H}(E) = -\frac{1}{N}\sum_{n=1}^{N} \log \hat{f}(e(n)) \tag{2}$$

where E is the error (difference) random variable. Note that expression (1) can be seen as the expected value of $\log f(x)$, thus the approximation of the integral by the mean value over a sample. Also, as we don't know the distribution of the error variable, we must rely on nonparametric estimates. For the estimation of $f(x)$ we use the nonparametric kernel estimator

$$\hat{f}(e(n)) = \frac{1}{Nh}\sum_{l=1}^{N} K\left(\frac{e(n) - e(l)}{h}\right) \tag{3}$$

where h is the smoothing parameter of the standard Gaussian kernel K given by

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) \tag{4}$$

In order to use the steepest descent training rule and the backpropagation algorithm, we need to derive an analytic expression for the gradient. Using the usual notation where $\frac{\partial \hat{H}}{\partial w_{kj}}$ is the partial derivative of $\hat{H}$ related to the weight

connecting neuron j in a previous layer to neuron k in the next layer, we have

$$\frac{\partial \hat{H}_S}{\partial w_{kj}} = -\frac{1}{N}\sum_n \frac{\partial}{\partial w_{kj}} \log(\hat{f}(e(n))) = -\frac{1}{N}\sum_n \frac{1}{\hat{f}(e(n))}\frac{\partial \hat{f}(e(n))}{\partial w_{kj}}$$

Now,

$$\begin{aligned}\frac{\partial \hat{f}(e(n))}{\partial w_{kj}} &= \sum_{l=1}^{N} \frac{1}{\sqrt{2\pi}}\frac{\partial}{\partial w_{kj}} \exp\left(-\frac{1}{2}\left(\frac{e(n)-e(l)}{h}\right)^2\right) \\ &= \sum_{l=1}^{N} \frac{1}{h^2} K\left(\frac{e(n)-e(l)}{h}\right)(e(n)-e(l))\left[\frac{\partial e(n)}{\partial w_{kj}} - \frac{\partial e(l)}{\partial w_{kj}}\right]\end{aligned}$$

Thus

$$\frac{\partial \hat{H}}{\partial w_{kj}} = \frac{1}{N^2 h^2}\sum_{n=1}^{N}\sum_{l=1}^{N} \frac{\frac{1}{h}K\left(\frac{e(n)-e(l)}{h}\right)}{\hat{f}(e(n))}(e(n)-e(l))\left[\frac{\partial e(n)}{\partial w_{kj}} - \frac{\partial e(l)}{\partial w_{kj}}\right] \qquad (5)$$

The computation of $\frac{\partial e(n)}{\partial w_{kj}}$ is as usual for the backpropagation algorithm. We just have to take care if w_{kj} is an input-hidden or an hidden-output neuron. Having determined (5) for all network weights, the weight update is given, for the m-th iteration, by the gradient descent rule

$$w_{kj}^{(m)} = w_{kj}^{(m-1)} - \eta \frac{\partial \hat{H}_E}{\partial w_{kj}}$$

2.1 Optimization

The algorithm has two parameters that one should optimaly set: the smoothing parameter, h, of the kernel density estimator (4) and the learning rate, η.
As the training process evolves, it is expected that the errors get closer, which means that one should need a decreasing smoothing parameter h. Experimental results led to the conclusion that this is a highly sensitive parameter. Indeed, decreasing h as the training evolves (for example, proportional to the variance of the errors in each epoch), leads to an unstable behaviour of the algorithm. In order to cope with this unstability, one should run experiments with several h values and determine the best for each dataset and network configuration.
We also investigated the benefits of adjusting η along the training process. Figure 1 shows the training curves for variable and fixed learning rate, where each curve is a mean over 25 repetitions of the corresponding experiment (for the SONAR dataset mentioned later). As we can see, with a typical value of $\eta = 0.1$ (dashed-dot line) the convergence is very slow when compared with the variable learning rate curve (dotted line). The solid line, for $\eta = 2$ shows a fast convergence but an unstable behaviour during the training process[1]. Thus, the procedure of variable learning rate not only solves the problem of choosing η, but also ensures a stable training.

[1]Remember that each curve is an average curve.

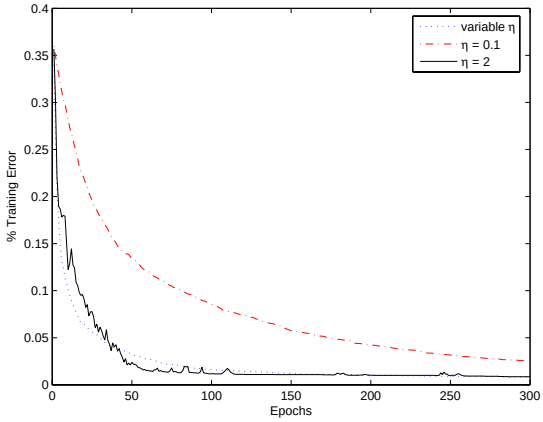

Fig. 1: Training error curves for variable and fixed learning rates.

We used the update rule and values $u = 1.2$ and $d = 0.2$ suggested in [4]

$$\eta^{(m)} = \begin{cases} u\,\eta^{(m-1)} & \hat{H}^{(m)} \leq \hat{H}^{(m-1)} \\ d\,\eta^{(m-1)} \wedge restart & otherwise \end{cases} \quad ,0 < d < 1 \leq u$$

If entropy does not increase from one epoch to another, the algorithm is in the right direction, so η is increased by a factor u in order to speedup convergence. However, if η is large enough to increase entropy, then the algorithm makes a *restart* step and decreases η by a factor d to ensure that entropy is being minimized. This *restart* step is just a return to the weights of the previous epoch.

3 Simulations

We performed experiments with five well known two-class datasets obtained from the UCI repository [6]: SONAR, LIVER, IONOSPHERE, WDBC and PIMA.
Table 1 shows a brief description of each dataset.

Datasets	#Instances	#Features
SONAR	208	60
LIVER	345	6
WDBC	569	30
IONOSPHERE	351	34
PIMA	768	8

Table 1: Description of the five UCI datasets used in the experiments.

For each dataset we trained several one hidden layer MLP configurations, varying the number of hidden units. The following procedure (*holdout*) was performed 20 times: divide the data in two subsets, half for training and half for testing; train the network during 150 epochs and compute the test set error; interchange the roles of the training and test sets; perform training and test again. This procedure was applied for the Shannon entropy (SE), MSE and cross-entropy (CE) [7] cost functions. The results obtained are shown in Table 2.

	SONAR			LIVER			IONOSPHERE		
hid	SE	MSE	CE	SE	MSE	CE	SE	MSE	CE
2	23.3(2.1)	22.0(2.8)	24.9(2.6)	30.6(1.5)	30.3(1.9)	32.3(2.3)	12.4(1.7)	13.7(1.9)	11.7(2.0)
3	23.2(2.6)	21.4(3.4)	24.1(2.8)	31.3(1.9)	29.7(2.0)	32.8(1.9)	12.2(1.4)	13.1(1.6)	11.7(1.7)
4	22.1(3.0)	21.4(2.5)	23.7(3.1)	30.4(2.1)	30.4(1.8)	30.7(2.2)	12.0(1.2)	13.2(1.6)	12.7(2.2)
5	22.3(2.3)	21.7(2.4)	23.5(2.8)	30.4(2.4)	30.3(3.0)	30.4(1.9)	12.2(1.2)	13.5(1.8)	12.5(1.8)
6	21.1(1.9)	21.9(2.7)	22.8(2.8)	30.4(1.7)	31.1(2.1)	30.4(1.8)	12.2(1.4)	13.3(1.2)	12.5(1.7)
7	21.6(2.6)	22.2(2.4)	21.9(3.1)	30.9(2.2)	30.9(2.4)	30.0(1.4)	12.1(1.3)	12.7(1.7)	12.1(1.5)
8	21.1(2.2)	20.9(3.2)	22.0(2.8)	30.4(1.6)	30.9(2.3)	29.7(1.8)	12.4(1.3)	13.4(2.0)	12.1(1.6)
9	21.6(2.2)	22.1(2.2)	23.4(2.9)	29.7(1.7)	29.9(2.0)	30.0(1.8)	12.1(1.3)	13.1(1.5)	12.1(1.4)
10	20.9(2.5)	20.8(2.6)	22.4(2.4)	30.3(2.1)	29.9(2.1)	30.3(1.6)	12.4(1.2)	13.9(3.1)	11.4(1.4)
11	20.8(2.3)	21.1(2.9)	23.7(2.4)	30.0(2.1)	31.1(1.9)	30.8(2.0)	12.6(1.7)	13.5(1.9)	12.2(1.3)
12	20.0(2.2)	20.9(2.9)	21.5(2.6)	30.2(1.6)	30.4(1.6)	30.8(2.5)	12.2(1.4)	13.2(1.2)	12.2(1.5)
hid	12	10	12	9	3	8	4	7	10
%	20.0(2.2)	20.8(2.6)	21.5(2.6)	29.7(1.7)	29.7(2.0)	29.7(1.8)	12.0(1.2)	12.7(1.7)	11.4(1.4)

	WDBC			PIMA		
hid	SE	MSE	CE	SE	MSE	CE
2	3.40(0.59)	3.80(0.85)	3.53(0.94)	24.5(1.2)	26.9(1.5)	24.4(1.1)
3	3.49(0.47)	3.34(0.65)	3.86(0.90)	24.9(0.8)	26.5(1.0)	24.8(1.2)
4	3.34(0.88)	3.35(0.63)	3.59(0.54)	25.6(1.5)	26.2(1.2)	24.8(1.3)
5	3.36(0.59)	3.54(0.73)	3.61(0.54)	25.5(0.9)	26.4(1.4)	24.7(0.9)
6	3.34(0.91)	3.52(0.67)	3.67(0.80)	25.7(1.0)	26.6(1.3)	24.8(0.9)
7	3.36(0.53)	3.30(0.70)	3.66(0.74)	25.6(1.0)	26.6(1.2)	24.3(1.0)
8	3.29(0.66)	3.59(0.62)	4.10(0.78)	25.8(1.3)	26.4(1.2)	24.2(0.9)
9	3.08(0.57)	3.26(0.61)	3.71(0.74)	25.8(1.3)	26.0(1.1)	24.3(1.2)
10	3.32(0.77)	3.58(0.72)	3.44(0.62)	25.6(1.1)	26.2(1.4)	24.1(0.9)
11	3.29(0.50)	3.40(0.72)	3.60(0.66)	25.8(1.2)	26.7(1.3)	23.6(1.0)
12	3.29(0.51)	4.36(4.45)	3.44(0.58)	25.9(1.4)	26.5(1.2)	23.4(0.7)
hid	9	9	10	2	9	12
%	3.08(0.57)	3.26(0.61)	3.44(0.62)	24.5(1.2)	26.0(1.1)	23.4(0.7)

Table 2: Test error (%) and standard deviations (in brackets) for the five UCI datasets using Shannon's entropy (SE), MSE and cross-entropy (CE) cost functions. The number of hidden units is denoted *hid*. The last two lines correspond to the best results.

As one can see, SE performs very well when compared to MSE and CE. In IONOSPHERE and PIMA datasets, all the SE results are better than the best MSE result, being the latter obtained with higher *hid* values than the best SE results. CE performed very well here, with lower error but higher *hid*. For SONAR and LIVER, the best MSE results are obtained with smaller *hid* values; still, the smallest misclassification error and/or standard deviation are obtained with SE. Moreover, CE performed poorly in SONAR. WDBC is the only dataset where the best results were achieved with equal number of hidden units, both for SE and MSE, but again SE outperforms MSE. Also, only two MSE results are better for

WDBC (hid=3,7). Again, CE has a worse performance. One can also see that, in general, SE has lower standard deviations which could mean a more stable learning procedure.

4 Conclusion

We proposed neural network classification using as cost function Shannon's entropy of the error. The principle, named EEM in [3], minimizes error entropy in order to maximize the mutual information between the output and the desired target of a neural network. When applied to five real datasets, EEM with Shannon entropy performed very well when compared with MSE and CE. The results show the effectiveness of entropic criteria, in particular Shannon's entropy, as cost functions in classification tasks. It has to be noticed that (2) is a double estimator only consistent in the mean square sense. It is our purpose, in future work, to study other estimators of entropy with stronger consistency properties, with expected improvements on the results.
The problem of choosing the learning rate η was efficiently solved with an adaptive rule, by monitoring the entropy value at each epoch. Furthermore, specific choices for u and d should also be studied.
It is also our purpose to study in more detail the issue of adaptive smoothing parameter h of the density estimator, in particular, to derive a practical and automatic rule for the adaptation of h during the training process.

References

[1] D. Erdogmus and J. Principe. Comparison of entropy and mean square error criteria in adaptive system training using higher order statistics. In *Intl. Conf. on ICA and Signal Separation*, pages 75–80, Helsinki, Finland, 2000.

[2] J. C. Principe, D. Xu, and J. Fisher. Information theoretic learning. In S. Haykin, editor, *Unsupervised Adaptive Filtering, vol. I: Blind Source Separation*, pages 265–319. Wiley, New York, 2000.

[3] J.M. Santos, L.A. Alexandre, and J. Marques de Sá. The Error Entropy Minimization Algorithm for Neural Network Classification. In *Int. Conf. on Recent Advances in Soft Computing*, Nottingham, United Kingdom, 2004.

[4] J.M. Santos, L.A. Alexandre, and J. Marques de Sá. Optimization of the Error Entropy Minimization Algorithm for Neural Network Classification. In C. Dagli, A. Buczak, D. Enke, M. Embrechts, and O. Ersoy, editors, *Intelligent Engineering Systems through Artificial Neural Networks*, volume 14, pages 81–86. ASME Press Series, 2004.

[5] I.A. Ahmad and P.E. Lin. A nonparametric estimation of the entropy for absolutely continuous distributions. *IEEE Trans. Information Theory*, 22:372–375, 1976.

[6] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/∼mlearn/MLRepository.html 1998.

[7] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.