Maximizing the Zero-Error Density for RTRL *

Luís A. Alexandre
Department of Informatics, University of Beira Interior and
IT-Networks and Multimedia Group
Covilhã, Portugal
email: luis.alexandre@ubi.pt

Abstract

A new learning principle was introduced recently called the Zero-Error Density Maximization (Z-EDM) and was proposed in the framework of MLP backpropagation. In this paper we present the adaptation of this principle to online learning in recurrent neural networks, more precisely, to the Real Time Recurrent Learning (RTRL) approach. We show how to modify the RTRL learning algorithm in order to make it learn using Z-EDM criteria by using a sliding time window of previous error values. We present experiments showing that this new approach improves the convergence rate of the RNNs and improves the prediction performance in time series forecast.

1 Introduction

Recently [3] a new principle for learning in neural networks was proposed: since the network learns as the errors converge to zero, the learning is made such that the zero-error density is maximized. Hence the name Zero-Error Density Maximization (Z-EDM). It has been shown to be a good alternative to learning according to the minimization of the mean squared error and cross entropy [3, 4].

In this paper we adapt the idea of Z-EDM to learning in recurrent neural networks. We show how an online learning algorithm for RNN, the Real Time Recurrent Learning (RTRL) [5] can make use of the Z-EDM principle.

In the experiments section we present two applications: a symbolic prediction problem and a time series forecast problem (Mackey-Glass chaotic time series). In both cases the new approach brings advantages when compared to the original RTRL.

The rest of the paper is organized as follows: the next section presents Z-EDM, the following section discusses RTRL and section 4 shows how to incorporated Z-EDM into the RTRL algorithm. Section 5 presents the experiments and the last section contains the conclusions.

*This work was supported by the Portuguese FCT-Fundação para a Ciência e a Tecnologia, POS_Conhecimento and FEDER (project POSC/EIA/56918/2004).

2 Z-EDM

The Z-EDM was proposed in [3]. Minor modifications were introduced in [4].

The idea is that when training a learning machine, in our case a NN, the random variable error, **e**, will tend to increase its density at the origin, as more and more patterns are correctly predicted. So, instead of taking the usual approach of defining an error surface (usually based on MSE error) and use gradient descend to search for the weights that minimize that error, in the Z-EDM approach, we define the density of the error variable and, also using the gradient descent, look for the weights that maximize the error density at the origin. We will now formalize these ideas. We consider here the original Z-EDM. The online version proposed in this paper is described below.

For a training set of size N, the error r.v. $\mathbf{e}(i) = \mathbf{d}(i) - \mathbf{y}(i)$ represents the difference between the desired output vector $\mathbf{d}(i)$ and the actual output $\mathbf{y}(i)$, for a given pattern i. Since as the training proceeds we expect that the values of $\mathbf{e}(i) = 0$ for most i, we will in fact use the following rule to search for the desired weights:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} f(\mathbf{0}; \mathbf{w}) \tag{1}$$

where $f(\mathbf{0}; \cdot)$ stands for the density of the errors at the origin, and $\mathbf{w}$ represents the network weights. The weight dependency of the error density was made explicit in the above expression. To apply this expression we need the error density, which is normally unknown. So we estimate it using the Parzen window non-parametric estimator:

$$\hat{f}(\mathbf{0}; \mathbf{w}) = \frac{1}{N h^p} \sum_{i=1}^{N} K\left(\frac{\mathbf{0} - \mathbf{e}(i)}{h}\right) \tag{2}$$

978-1-4244-3555-5/08/$25.00 ©2008 IEEE

where h represents the bandwidth of the kernel K and p is the dimension of $\mathbf{e}$ (the number of network outputs).

The kernel used is the Gaussian kernel with zero mean and unit covariance given by

$$K(\mathbf{x}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\mathbf{x}^T\mathbf{x}}{2}\right) \tag{3}$$

By replacing this in expression (2) we get the our estimator for the error density

$$\hat{f}(\mathbf{0};\mathbf{w}) = \frac{1}{\sqrt{2\pi}Nh^p} \sum_{i=1}^{N} \exp\left(-\frac{\mathbf{e}(i)^T\mathbf{e}(i)}{2h^2}\right) \tag{4}$$

Due to reasons discussed in [4] related to the speed of convergence of the Z-EDM, instead of using expression (4) we shall use the following simplification

$$\hat{f}(\mathbf{0};\mathbf{w}) = h^2 \sum_{i=1}^{N} \exp\left(-\frac{\mathbf{e}(i)^T\mathbf{e}(i)}{2h^2}\right) \tag{5}$$

and given that the difference relies only on constant terms, we know that the same extrema will be found.

We are now interested in the gradient of (5):

$$\frac{\partial \hat{f}(\mathbf{0};\mathbf{w})}{\partial \mathbf{w}} = -\sum_{i=1}^{N} \exp\left(-\frac{\mathbf{e}(i)^T\mathbf{e}(i)}{2h^2}\right) \mathbf{e}(i) \left(\frac{\partial \mathbf{e}(i)}{\partial \mathbf{w}}\right) \tag{6}$$

Since we are searching for the weights yielding the maximum of the error density at $\mathbf{0}$, the network weight update shall be made by

$$\Delta\mathbf{w} = \eta \frac{\partial \hat{f}(\mathbf{0};\mathbf{w})}{\partial \mathbf{w}} \tag{7}$$

where η stands for the learning rate.

3 RTRL

The RTRL algorithm for training fully recurrent neural networks (NNs) was originally proposed in [5]. Many modifications to this original proposal have been made.

In this section we will follow the notation of [1]. Consider the fully recurrent neural network of figure 1. It contains q neurons, m inputs and p outputs.

The state-space description is given by the following equations

$$\mathbf{x}(t+1) = \left[\varphi(\mathbf{w}_1^T\boldsymbol{\xi}(t)), \ldots, \varphi(\mathbf{w}_q^T\boldsymbol{\xi}(t))\right]^T \tag{8}$$

where $\mathbf{x}$ represents the state vector, t stands for the time, and φ is the activation function. The $(q+m+1)$-by-1 vector $\mathbf{w}_j$ contains the weights of neuron j and $\boldsymbol{\xi}(t)$ is another $(q+m+1)$-by-1 vector defined by $[\mathbf{x}(t), \mathbf{u}(t)]^T$. The

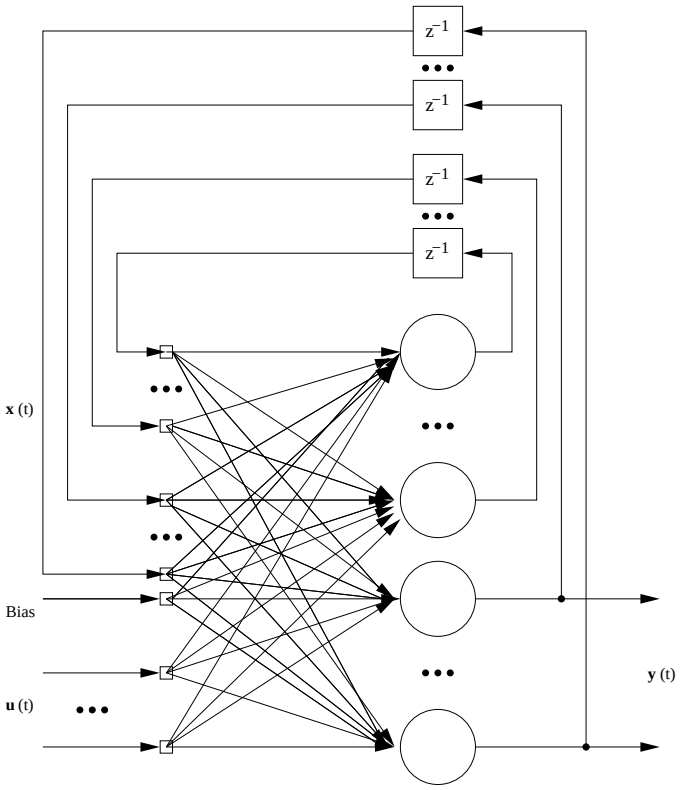

Figure 1. A fully recurrent one hidden layer neural network. The notation is explained in the text.

$(1+m)$-by-1 vector $\mathbf{u}(t)$ contains in the first position 1 (the bias fixed input value) and in the remaining positions the m network inputs. The equation that gives the p-by-1 vector of network outputs, $\mathbf{y}$, is

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \tag{9}$$

where $\mathbf{C}$ is a p-by-q matrix that is used to select which neurons produce the network output.

The idea is to use the instantaneous gradient of the error to guide the search for the optimal weights that minimize this error. The algorithm works by computing the following for each time t:

$$\boldsymbol{\Lambda}_j(t+1) = \boldsymbol{\Phi}(t)\left(\mathbf{W}_a(t)\boldsymbol{\Lambda}_j(t) + \mathbf{U}_j(t)\right) \tag{10}$$

$$\mathbf{e}(t) = \mathbf{d}(t) - \mathbf{C}\mathbf{x}(t) \tag{11}$$

$$\Delta\mathbf{w}_j = \eta\left(\mathbf{e}(t)^T\mathbf{C}\boldsymbol{\Lambda}_j(t)\right)^T \tag{12}$$

where $\boldsymbol{\Lambda}_j$ contains the partial derivatives of $\mathbf{x}$ w.r.t. the weight vector $\mathbf{w}_j$, $\boldsymbol{\Phi}$ is a diagonal matrix with the partial derivatives of the activation function w.r.t. its arguments, $\mathbf{W}_a$ contains part of the network weights and $\mathbf{U}_j$ is a zero

978-1-4244-3555-5/08/$25.00 ©2008 IEEE

matrix with the transpose of vector $\boldsymbol{\xi}$ in its jth row (please see [1] for details). $\mathbf{e}$ is the error and $\mathbf{d}$ the desired output.

4 Application of Z-EDM to RTRL

To use the Z-EDM approach in a RNN, we chose to adapt the RTRL algorithm. The idea of finding the maximum of the density now can't be done using a static set of N error values.

We propose to use the previous L values of the error to build a dynamic approximation to the error-density. This will use a time sliding window that will allow us to define the density in an online formulation of the learning problem.

This adaptation of the Z-EDM will change expression (6) above to

$$\frac{\partial \hat{f}(\mathbf{0}, t; \mathbf{w})}{\partial \mathbf{w}} = -\sum_{i=1}^{L} \exp\left(-\frac{\mathbf{e}(t-i)^T \mathbf{e}(t-i)}{2h^2}\right) \mathbf{e}(t-i) \left(\frac{\partial \mathbf{e}(t-i)}{\partial \mathbf{w}}\right) \tag{13}$$

where instead of computing the density over the N data points of a training set, we are computing it over the last L errors of the RNN. Notice that the dependency on time of the gradient of the density is now explicit.

This approach is an approximation to the real gradient of the density since it uses error values from different time steps to create an estimate of the error density. Given that learning is online and the weights are adjusted on each time step, the construction of a density from errors at different time steps is valid if L is not too large since then it would include error information from a very different weight state and L can't be too small otherwise the density estimation will suffer from the lack of samples.

The modification to the RTRL will be on equation (12) which contains the weight update rule:

$$\Delta \mathbf{w}_j = \eta \sum_{i=1}^{L} \exp\left(-\frac{\mathbf{e}(t-i)^T \mathbf{e}(t-i)}{2h^2}\right) \left(\mathbf{e}(t-i)^T \mathbf{C} \boldsymbol{\Lambda}_j(t-i)\right)^T \tag{14}$$

The negative sign on expression 13 cancels with the negative sign we would insert on the $\Delta \mathbf{w}_j$ given that we now want to maximize instead of minimize.

5 Experiments

In this section we present experiments to evaluate the performance of the proposed method. We compare the obtained results against the original RTRL on the same RNNs. First we should refer that the activation function we used in the RNNs is the standard sigmoid. Also worth mentioning is the initialization values of the following matrices: $\mathbf{x}(0) = \mathbf{0}, \boldsymbol{\Lambda}_j(0) = \mathbf{0},\ j = 1, \ldots, q$.

5.1 First experiment

The first problem consists in predicting the next symbol of the sequence: `0 1 0 0 1 0 0 0 1 0 0 0 0 1` ..., up to twenty zeros, always followed by a one. We record how many symbols did the network need to see to correctly make the remaining predictions until the end of the sequence. The sequence is composed of 230 symbols.

We made 100 repetitions starting with random initialization of the weights, varied the learning rate, η, in $\{2, 3, 4, 5, 6\}$, varied the kernel bandwidth, h, in $\{1, 2, 3\}$ and the size of the sliding window for the temporal estimation of the density, L, in $\{8, 10, 12\}$. The results are in figures 2 (for a network with 4 neurons) and figure 3 (for a network with 6 neurons).

Each point in these figures represents the percentage of convergence on 100 experiments versus the correspondent average number of symbols necessary for learning the problem (NS), for the standard RTRL (circle) and the RTRL with Z-EDM (star). The different points were obtained by changing the parameters: η, L, and h (in the case of standard RTRL only η is used).

We only plotted the cases where at least one of the 100 repetitions converged.

The figures show that standard RTRL is not able to obtain more than 30% convergence, but the RTRL with Z-EDM can reach 100% convergence.

We can also observe that for a given value of NS, the RTRL with Z-EDM is able to obtain higher percentages of convergence than the original RTRL.

An advantage of the original RTRL over the new proposal is that it is able to learn the problem with fewer symbols: around 20 (although with very small percentage of convergence) while the new method needs around 50 symbols to learn.

5.2 Second experiment

In this experiment 3000 points of the Mackey-Glass time series [2] are used. The first 2000 are ignored (we consider that the network is still adapting to the signal) and the last 1000 are used for evaluation. We use the mean squared error (MSE) between the true value of the series and the prediction as the measure for prediction quality. The predictions are 1-step ahead.

To see how good these predictions are, we found the MSE for the naïf predictor: next value is equal to the previous. We call this the MSE_0 and its value is 0.00109. So,

978-1-4244-3555-5/08/$25.00 ©2008 IEEE

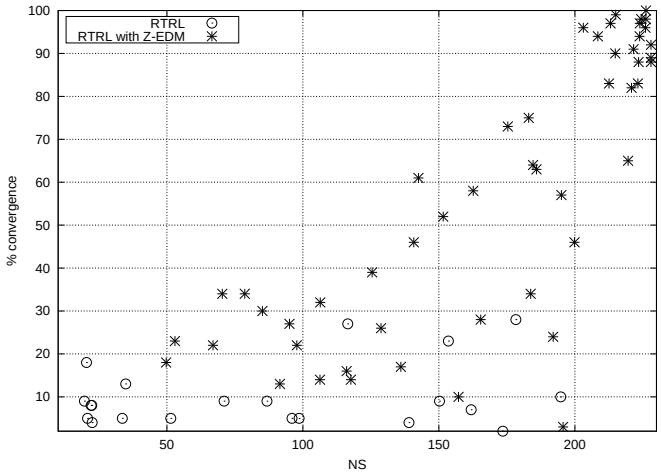

Figure 2. Percentage of convergence on 100 experiments versus the correspondent average number of symbols necessary for learning the problem (NS), for the standard RTRL and the RTRL with Z-EDM. Network with 4 neurons. First experiment.

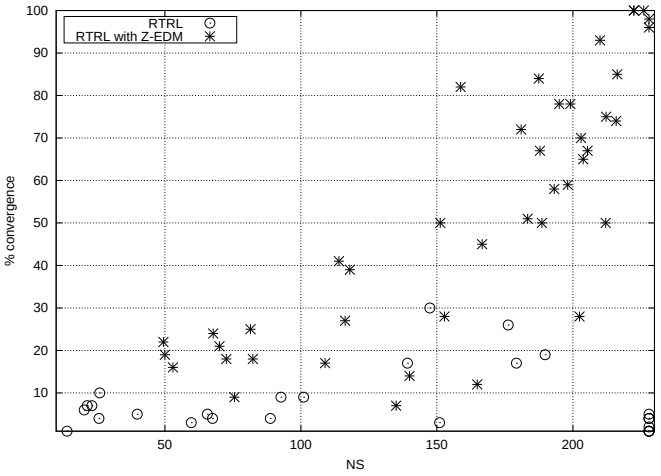

Figure 3. Percentage of convergence on 100 experiments versus the correspondent average number of symbols necessary for learning the problem (NS), for the standard RTRL and the RTRL with Z-EDM. Network with 6 neurons. First experiment.

good predictors should have smaller MSE than MSE_0.

Each point in figures 4 and 5 represents the percentage of convergence on 100 experiments versus the correspondent average MSE, for the standard RTRL (circle) and the RTRL with Z-EDM (star). The different points were obtained by changing the parameters: η, L, and h (in the case of standard RTRL only η is used). Both figures have the same scale to facilitate comparisons.

Figure 4 contains the results for a network with 4 neurons and figure 5 for a network with 6 neurons. We did not continue to expand the number of neurons since there was not a considerable difference in performance by increasing the number of neurons in either approach.

Note that we only represented the results when the average MSE for the 100 repetitions was smaller than MSE_0 (the total number of experiments conducted was over 500, but only a small portion had the required MSE value). This means that the number of points from RTRL is not the same as the number of points from RTRL with Z-EDM. Notice also that since the approach with Z-EDM has 3 parameters (excluding the number of neurons) versus only one parameter for the standard RTRL, we had to make many more experiments for the approach with Z-EDM to search for appropriate parameters than the number of experiments done for the standard RTRL.

By looking at the figures we find that the RTRL with Z-EDM is always able to obtain smaller values of MSE than the standard RTRL and that the difference is significative. Another important aspect is that only the RTRL with Z-EDM was able to obtain MSE smaller than MSE_0 and also have 100% convergence. This is more significative for the case of 6 neurons since the best convergence here for the standard RTRL is 4%. In the case of 4 neurons, the standard RTRL is able to reach 82% convergence (although with not as good MSE as the best 100% convergence RTRL with Z-EDM).

6 Conclusions

In this paper a new learning algorithm for RTRL was introduced: RTRL with Z-EDM.

We presented experiments on symbol prediction and time series prediction.

This shows that it is possible to use the concept of Z-EDM in recurrent neural network training; that this approach is beneficial since the percentage of the time that the RNN converges was increased in the experiments we presented; and in the time series prediction experiment, the proposed method has much better prediction capabilities than the original.

The possible drawbacks are a slight increase in computational time and the need to set two more parameters: the kernel bandwidth h and the size of the window used for keeping previous error values, L. In our experiments we did not find it difficult to find adequate parameters with a standard grid search.

978-1-4244-3555-5/08/$25.00 ©2008 IEEE

References

[1] S. Haykin. *Neural Networks: A Comprehensive Foundation, 2nd edition*. Prentice Hall, 1999.

[2] M.C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287–289, 1977.

[3] L.M. Silva, L.A. Alexandre, and J. Marques de Sá. Neural network classification: Maximizing zero-error density. In *Third International Conference on Advances in Pattern Recognition - ICAPR 2005*, volume LNCS 3686, pages 127–135, Bath, United Kingdom, August 2005. Springer.

[4] L.M. Silva, L.A. Alexandre, and J. Marques de Sá. New developments of the Z-EDM algorithm. In *6th International Conference on Intelligent Systems Design and Applications - ISDA 2006*, volume 1, pages 1067–1072, Jinan, China, October 2006. IEEE Computer Society Press.

[5] R.J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280, 1989.

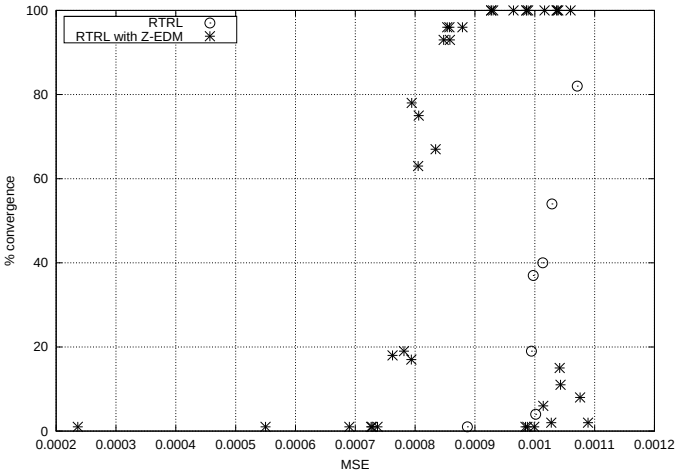

Figure 4. Percentage of convergence on 100 experiments versus the correspondent average MSE, for the standard RTRL and the RTRL with Z-EDM. Network with 4 neurons. Second experiment.

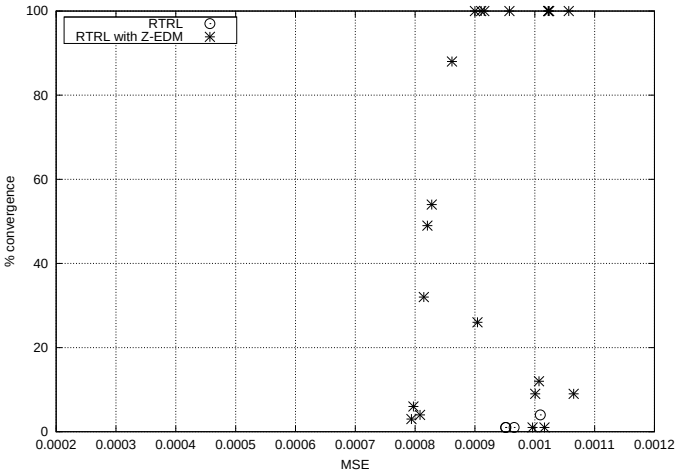

Figure 5. Percentage of convergence on 100 experiments versus the correspondent average MSE, for the standard RTRL and the RTRL with Z-EDM. Network with 6 neurons. Second experiment.

978-1-4244-3555-5/08/$25.00 ©2008 IEEE