One Step Backpropagation through time for learning input mapping in reservoir computing applied to speech recognition

Michiel Hermans and Benjamin Schrauwen
ELIS department
Ghent University
Sint Pietersnieuwstraat 42 Ghent, Belgium
Email: Michiel.Hermans@ugent.be, Benjamin.Schrauwen@ugent.be

***Abstract*— Recurrent neural networks are very powerful engines for processing information that is coded in time. Many problems with common training algorithms, such as Backpropagation Through Time, remain however. Because of this, another important learning setup known as Reservoir Computing has appeared in recent years, where one uses an essentially untrained network to perform computations. Though very successful in many applications, using a random network can be quite inefficient when considering the required number of neurons and the associated computational costs. In this paper we introduce a highly simplified version of Backpropagation Through Time by basically truncating the error backpropagation to one step back in time, and we combine this with the classic Reservoir Computing setup using an instantaneous linear readout. We apply this setup to a spoken digit recognition task and show it to give very good results for small networks.**

I. Introduction

A significant body of research on neural networks focuses on recurrent neural networks. These networks have for instance been studied for their ability to store patterns (the so-called Hopfield networks [1]). More recently however, recurrent networks are being used to process temporal information; due to internal feedback, these networks have an intrinsic ability to retain information about past input for a certain time, which allows for the processing of signals which are explicitly coded in time.

Two important methods for using recurrent networks currently exist. First of all one can define an error gradient, which can be used in a classical gradient descent algorithm to adapt the connection weights. This method is known as *Backpropagation Through Time* (BPTT) [2]. Though this algorithm can in many cases be very powerful, some problems like slow convergence, high complexity, bifurcations, unstability, and limited applicability due to high computational costs remain [3]. The gradient needed in BPTT typically depends on the entire history of the network states, which greatly complicates its implementation. An equally important criticism is the fact that - even though BPTT is in many cases quite a powerful technique to solve engineering tasks - such a form of error backpropagation is highly unlikely to occur in the brain, mostly due to its nonlocal character.

A second technique circumvents these difficulties by not training the network at all, but rather train an external layer of linear readout nodes which only get input from the network. Training single linear nodes can be done very efficiently offline by solving a system of linear equations or online by e.g. recursive least squares (RLS) [4]. This technique is called *Reservoir Computing* (RC), and is discovered independently by Jaeger [5] and Maass [6]. Despite its recent introduction, RC has already proved to be quite powerful: many applications for RC have already been successfully implemented [7]–[9].

RC uses random networks, and even though these are already very useful, one can also state that the performance of a specific network will depend more on luck than anything else. For this reason we shall look at a hybrid between RC and BPTT, where we use the basic setup of RC (i.e. instantaneous linear projections from input to network and from network to output) and only propagate the error back one timestep, which we shall call One Step Backpropagation (OSBP). We apply this setup to a spoken digit classification task, where we show that this approach is superior over using random networks and causes an inherent representation of individual digits within the network state. Furthermore we show that only training the input connections already gives very good performance, which has advantages in computational cost and more importantly avoids problems as bifurcation and networks that become unstable.

This paper is structured as follows: first we describe the general network and training setup, next we measure performance and analyze results. We conclude with a discussion of the results and future work.

II. Basic setup

A. Network setup

We use a network setup as used in [10], where each neuron performs an additional low-pass filter on its output. The evolution of the network states is then defined by the following equation:

$$\mathbf{a}(t+1) = \left(1 - \frac{1}{\tau}\right)\mathbf{a}(t) + \frac{1}{\tau} f\left(\mathbf{W}\mathbf{a}(t) + \mathbf{V}\mathbf{s}(t)\right), \quad (1)$$

with $\mathbf{a}(t)$ the network states, τ a time constant, (which we choose at 5 for all experiments in this paper) consistent with the time scale of the low-pass filtering in each neuron, $\mathbf{s}(t)$ are the input signals and $\mathbf{W}$ and $\mathbf{V}$ internal and input connection weights respectively. The function $f(x)$ is the fermi function, defined as $f(x) = 1 + \frac{1}{2}\tanh(x)$. The linear output is then defined as

$$\mathbf{o}(t) = \mathbf{U}\mathbf{a}(t), \tag{2}$$

with $\mathbf{U}$ readout weights, making the output an instantaneous linear projection of the network states. For all experiments in this paper we used networks of 50 neurons. All weights are initially drawn from a uniform distribution between -1 and 1, and then divided by the spectral radius of $\mathbf{W}$. Notice that the spectral radius has no specific meaning in a network with a fermi function nonlinearity. We simply do this for having a convenient scaling measure.

B. One Step Backpropagation

BPTT can be understood from an error gradient point of view. Weights change according to the differential equation $W_{ij}(t+1) = W_{ij}(t) - \eta\frac{\mathrm{d}E(t)}{\mathrm{d}W_{ij}(t)}$, with $E(t) = ||\mathbf{e}(t)||^2$, the squared error $\mathbf{e}(t) = \mathbf{o}(t) - \mathbf{o}^*(t)$ between actual and desired outputs, and η a small learning rate which we choose at $\eta = 2\times10^{-4}$ for all experiments in this paper. With the chain rule, the derivative is then expanded to

$$\frac{\mathrm{d}E(t)}{\mathrm{d}W_{ij}(t)} = \sum_k \frac{\partial E(t)}{\partial a_k(t)}\frac{\mathrm{d}a_k(t)}{\mathrm{d}W_{ij}(t)}. \tag{3}$$

The total derivative $\mathrm{d}a_k(t)/\mathrm{d}W_{ij}(t)$ then depends on the previous states $\mathbf{a}(t-1)$ which on their turn depend on the previous states and so forth. Put shortly, the gradient will depend on the entire history of the network states, making BPTT quite complicated. We apply the following simplification:

$$\frac{\mathrm{d}a_k(t)}{\mathrm{d}W_{ij}(t)} \approx \frac{\partial a_k(t)}{\partial W_{ij}(t)}, \tag{4}$$

ignoring all dependency on the history of states. Using this definition, and using similar expressions for the input and output weights, we get the update rules for OSBP:

$$\mathbf{U}(t+1) = \mathbf{U}(t) + \eta\mathbf{e}(t)\mathbf{a}^\mathsf{T}(t) \tag{5}$$

$$\mathbf{V}(t+1) = \mathbf{V}(t) + \eta\mathbf{D}(t)\mathbf{U}^\mathsf{T}\mathbf{e}(t)\mathbf{s}^\mathsf{T}(t-1) \tag{6}$$

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta\mathbf{D}(t)\mathbf{U}^\mathsf{T}\mathbf{e}(t)\mathbf{a}^\mathsf{T}(t-1), \tag{7}$$

where the elements of $\mathbf{D}$ are given by

$$\mathbf{D}_{ij}(t) = \delta_{ij}\frac{1}{2\tau}\left(1 - \tanh^2([\mathbf{W}\mathbf{a}(t-1) + \mathbf{V}\mathbf{s}(t-1)]_i)\right). \tag{8}$$

Notice that the update rule for the readout weights is that of the classic perceptron rule. This means that, without OSBP, we basically end up with the classical reservoir setup where the readout weights are trained with the perceptron rule. This means that the readout weights are constantly trying to get as close as possible to the optimal readout values.

The update rules for $\mathbf{V}$ and $\mathbf{W}$ are in fact equivalent with the setup of training an Elman network [11]. An Elman network is basically a feedforward network which has the hidden states of the previous timestep as extra inputs. It is easy to see that an Elman network always has an equivalent recurrent network, and that propagating the error back one timestep is equivalent with classic backpropagation in an Elman network.

Even though OSBP is conceptually easier and less computationally demanding as BPTT, it also has the disadvantage of bifurcations due to the fact that internal weights are trained. Also, in general, the number of internal connections is quite high compared to the number of input and output connections. This is especially true for RC, where the rule of thumb is that the dimensionality of state space should be much higher than the input dimensionality. For this reason, we consider a further simplification of OSBP by only training the input and output weights $\mathbf{V}$ and $\mathbf{U}$, which is clearly more closely linked to the RC setup.

III. Experimental validation

A. Spoken digit recognition

To investigate the performance of our learning algorithm, we applied it to a spoken digit classification task. We used a subset of the TI46 isolated digit corpus where the digits from "zero" to "nine" are spoken 10 times by 5 different females (a total of 500 words). The resulting data was preprocessed using the Lyon passive ear model [12], which produces initially 88 frequency channels. We downsampled the data to 20 input channels and a sample rate of approximately 35 timesteps per word. Next, we randomly selected half the words to be part of the test set, and the other half as the training set. The training datasets were then constructed by randomly sampling digits from the training set and separating them with intervals of 15 timesteps. Testing was performed by presenting all 250 words of the testset in a random order and measuring the classification error, which we call the Word Error Rate (WER). For each training and testing cycle, a new training and testing set was drawn.

The readout layer consists of 10 classifiers where initially each classifier had a target output of 1 when their corresponding digit is uttered, -0.1 for the other classifiers, and zero for all classifiers during the 15 timestep intervals. However, it appears the networks have some trouble with this signal as a desired output signal, probably since it is virtually impossible to recognize a word at the very start of its utterance. This results in slightly worse performance and we suspect this is due to the fact that the network will make wrong associations at the beginnings of the words. To reduce this effect, we low-pass filtered the target output with the time scale of the network, effectively slightly softening and delaying the desired output. Put in a formula: if $\hat{\mathbf{o}}^*$ is the original desired output $\mathbf{o}^*$we produce the actual desired output by calculating $\mathbf{o}^*(t+1) = \left(1-\tau^{-1}\right)\mathbf{o}^*(t) + \tau^{-1}\hat{\mathbf{o}}^*(t)$.

Classification is finally performed by taking the mean of the output signals for the duration of the utterance, and selecting the digit which corresponds to the channel with the highest mean output.

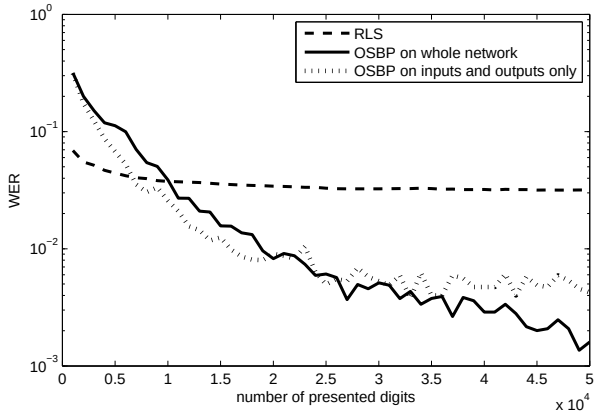

Fig. 1. WER of the three training setups in function of the number of presented digits. Results are averaged over 50 network initializations

B. Experimental setup

To compare OSBP to classic reservoir computing, i.e. with random input mapping and random internal connections, we compared its performance with training the readout layer with RLS, which is an efficient way of determining optimal linear readout weights in an online fashion[1]. To monitor the progress of both learning rules, we freeze all the weights every 1000 digits and measure the WER on the test set. Second, we investigated whether or not the network internally adapts specifically to the task, i.e., whether or not the internal representation of the digits depends more on the digit itself than its specific utterance. For this purpose we measured the centroids of the reservoir states during each word (the mean over time of the network activation during the utterance of a word), which we denote $\mathbf{q}^k$ for the k-th word. We investigated the clustering of these before and after OSBP. As a distance measure[2], we used $D = 1-r$, in which r is overall correlation between the centroids, i.e., the distance between the k-th and l-th word is then given by

$$D_{kl} = 1 - \frac{\sum_{i=1}^{N} (q_i^k - \bar{q}^k)(q_i^l - \bar{q}^l)}{\sqrt{\sum_{i=1}^{N} (q_i^k - \bar{q}^k)^2 \sum_{i=1}^{N} (q_i^l - \bar{q}^l)^2}}, \quad (9)$$

where $\bar{q}^k = N^{-1} \sum_{i=1}^{N} q_i^k$.

IV. Results

Basic results of performance are shown in Figure 1. It appears that the RLS algorithm very rapidly converges to optimal readout weights and gives a final performance of about 3% WER. However, it is clear that OSBP, though slower in convergence gives superior final results. Remarkably, it seems that only training the input weights gives already a great improvement over random networks but with obviously less computational demands than training the full network.

Figure 2 gives an example of the dendrograms of the centroids of the network dynamics for the different digits before and after adaptation. Untrained networks seem to have very little to no tendency to cluster the different digits, suggesting that the dynamics of random networks depend more strongly on the specific utterance of the digit than the actual digit. The learning rule automatically seems to cluster the data, trying to enforce similar trajectories for all digits within a single class while enlarging the separation between the classes.

It is interesting to note that the clustering of the centroids is also very good when only the input weights are trained. This can be due to two factors. First of all, the input mapping can already improve the clustering of the digits itself, i.e. the input mapping can try to find a projection which performs optimal spatial clustering of the input data (where each datapoint corresponds to the mean over time of the input data of a digit, i.e. the centroids of the digits). On the other hand, the input mapping could also take into account the dynamics of the network itself, where the temporal structure of the data will play a crucial role.

To investigate this we consider the clustering of the centroids of the digits and the centroids of the digits after the linear input mapping (i.e. the centroids of $\mathbf{V}\mathbf{s}(t)$). Examples of the resulting dendrograms are shown in Figure 3. Though some improvement is apparent, the linear mapping alone cannot fully account for the nearly perfect clustering of the centroids of the network states. This means that the learning rule finds an optimal mapping from the input data to the network which also accounts for temporal information.

Apparently, finding an optimal projection into the state space of a random dynamic network - at least for digit recognition - already offers a very significant improvement over a random projection. Most importantly, training only the input and output weights of a network has the great advantage that problems as bifurcation or loss of stability can no longer occur, since none of the recurrent weights are trained. The network itself can still be considered as a separate, stable-by-construction dynamic system which is left unchanged. This allows OSBP on the input weights to be interpreted as a powerful extension to the common RC setup.

V. Conclusions and future work

In this paper we have investigated a highly simplified version of Backpropagation Through Time by simply truncating the backpropagation process at one step back in time. We furthermore used a central paradigm of Reservoir Computing, i.e. using a single layer of instantaneous linear nodes as readout units. We apply our learning algorithm to a spoken digit classification task.

The networks trained by OSBP outperformed random networks, giving nearly perfect classification with a relatively small number of neurons. Furthermore, it appears that the network adapts functionally to its task by improving the clustering of the different digits. Remarkably, it seems that only training the input mapping gives already much improvement in performance.

It appears that our learning method offers a possible way

[1]Note that the perceptron rule will eventually also converge to the optimal readout weights. Its convergence however is very slow compared to RLS.

[2]Euclidian distance is less meaningfull in high dimensional spaces [13].

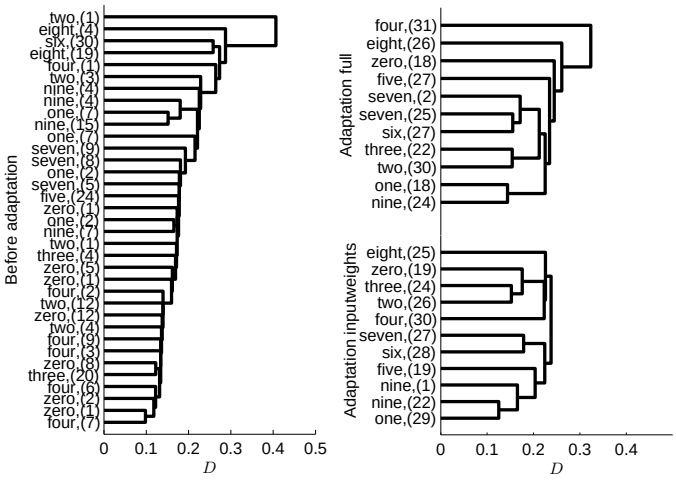

Fig. 2. Examples of typical dendrograms of the centroids of the dynamics for different digits before and after adaptation. The written words are the digits, the number between brackets are the number of digits in each leaf.

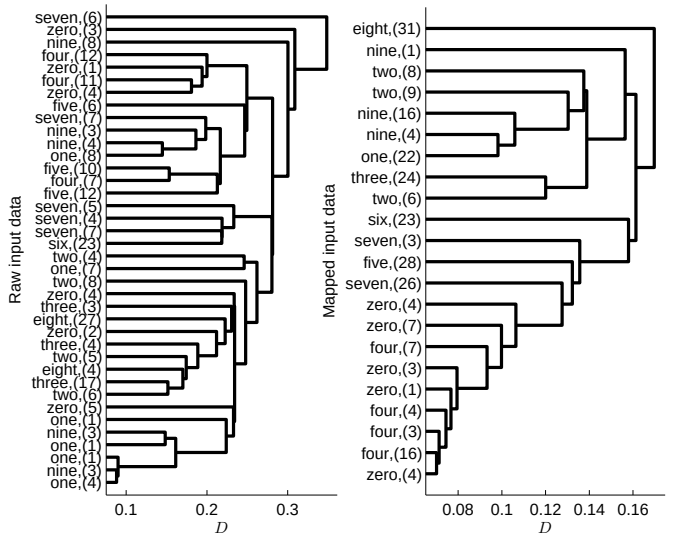

Fig. 3. Examples of typical dendrograms of the centroids of the input data and the projected input data, the number between brackets are the number of digits in each leaf.

to functionally improve performance of random recurrent neural networks, while at the same time remaining relatively uncomplicated and being "safe", in the sense that the common problems training recurrent weights - bifurcations and unstability - are avoided.

Many clear paths for future research remain available; most obviously, trying to extend the above paradigm to more difficult temporal tasks which depend on a longer history of the input signal. This poses extra challenges as it is a well known fact that these problems are hard to solve even with classic BPTT. Also, it is not very clear how much the success of OSBP depends on the low pass filtering operation, the shape of the nonlinearity or the distribution of the initial weights. Most importantly, it is unclear whether the fact that we use a separate linear readout layer to generate output is a critical factor in our setup.

Another fairly natural extension is to try to speed up the convergence by using recursive least squares on the readout weights instead of the slow perceptron learning rule. This would basically rapidly force output weights to optimal values such that the internal weight updates would never be very far away from a local optimum. This idea would also draw an interesting parallel between OSBP and the recently introduced FORCE learning rule [14], which uses RLS on the outputs, combined with feedback to generate signals.

ACKNOWLEDGMENT

The research leading to the results presented here has received funding from the European Community's Seventh Framework Programme (EU FP7) under grant agreement n. 231267 " Self-organized recurrent neural learning for language processing (ORGANIC)". This work was partially funded by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

REFERENCES

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities." *Proc Natl Acad Sci U S A*, vol. 79, no. 8, pp. 2554–2558, Apr 1982.

[2] D. Rumelhart, G. Hinton, and R. Williams, *Learning internal representations by error propagation.* MIT Press, Cambridge, MA, 1986.

[3] B. Hammer and J. J. Steil, "Perspectives on learning with recurrent neural networks," in *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, 2002.

[4] M. H. Hayes, *Statistical Digital Signal Processing and Modeling*. Wiley, 1996, ch. Recursive Least Squares, p. 541.

[5] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 148, 2001.

[6] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[7] D. Verstraeten, B. Schrauwen, and D. Stroobandt, "Reservoir-based techniques for speech recognition," in *Proceedings of the World Conference on Computational Intelligence*, 2006, pp. 1050–1053.

[8] E. A. Antonelo, B. Schrauwen, and D. Stroobandt, "Event detection and localization for small mobile robots using reservoir computing," *Neural Networks*, vol. 21, pp. 862–871, 2008.

[9] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication," *Science*, vol. 308, pp. 78–80, April 2 2004.

[10] H. Jaeger, "Short term memory in echo state networks," German National Research Center for Information Technology, Tech. Rep. GMD Report 152, 2001.

[11] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[12] R. Lyon, "A computational model of filtering, detection and compression in the cochlea," in *Proceedings of the IEEE ICASSP*, May 1982, pp. 1282–1285.

[13] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Lecture Notes in Computer Science*. Springer, 2001, pp. 420–434.

[14] D. Sussillo and L. F. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, no. 4, pp. 544–557, August 2009. [Online]. Available: http://dx.doi.org/10.1016/j.neuron.2009.07.018