ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

Pruning and Regularisation in Reservoir Computing: a First Insight

Xavier Dutoit[1], Benjamin Schrauwen[2], Jan Van Campenhout[2],
Dirk Stroobandt[2], Hendrik Van Brussel[1], Marnix Nuttin[1] *

1- Katholieke Universiteit Leuven - Mobile Learning Robots Research Group
Celestijnenlaan 300b, 3000 Leuven - Belgium

2- Universiteit Gent - Dept of Electronics and Information Systems
Sint Pietersnieuwstraat 41, 9000 Gent - Belgium

Abstract. Reservoir Computing is a new paradigm for using Recurrent Neural Networks which shows promising results. However, as the recurrent part is created randomly, it typically needs to be large enough to be able to capture the dynamic features of the data considered. Moreover, this random creation is still lacking a strong methodology. We propose to study how pruning some connections from the reservoir to the readout can help on the one hand to increase the generalisation ability, in much the same way as regularisation techniques do, and on the other hand to improve the implementability of reservoirs in hardware. Furthermore we study the actual sub-reservoir which is kept after pruning which leads to important insights in what we have to expect from a good reservoir.

1 Introduction

Reservoir Computing (RC) is a relatively new technique for using Recurrent Neural Networks (RNNs), which has been introduced independently in [1, 2, 3].

This technique has been shown to achieve performance comparable to the state-of-the-art on various tasks with a much simpler training algorithm than classical RNNs. The basic idea is to use a large RNN as a reservoir of functions and to train a linear readout layer to extract the relevant information from the reservoir. The recurrent part itself is randomly created and left untrained.

This has raised the question on how to create good reservoirs for a given task. Several solutions have been proposed: general guidelines [4], gradient-descent algorithms [5], reservoir adaptation [6], and, in a previous work from the authors, reservoir pruning [7]. The latter idea, that we extend here, is to improve the reservoir by pruning out the least useful neurons. Actually, in order to disturb the dynamics of the reservoir less, only the readout connections, i.e. the least useful connections from the neurons to the readout, are pruned.

We will show that this readout pruning improves the generalisation ability of the reservoir. Moreover, as the main bottleneck of most hardware implementations of reservoir [8, 9] is the number of connections from the reservoir to the readout, it also decreases the hardware requirements.

In the last part of this work we will study what the properties of the resulting optimally pruned sub-reservoirs (i.e. the part of the reservoir still connected

*This work was funded by the FWO Flanders project G.0317.05.

to the readout) are. This enables us to have a better understanding of what important properties of reservoirs are and how we can build better reservoirs.

2 Approach

We consider here only the ESN model [2], although the same pruning method can be applied to LSM. The reservoir state $\mathbf{s}[t]$ and output $\mathbf{o}[t]$ are described by

$$\mathbf{s}[t] = f\Big(\mathbf{W}_{\mathtt{r}}^{\mathtt{r}} \cdot \mathbf{s}[t-1] + \mathbf{W}_{\mathtt{o}}^{\mathtt{r}} \cdot \mathbf{o}[t-1]\Big) \tag{1}$$

$$\text{and } \mathbf{o}[t] = \mathbf{W}_{\mathtt{r}}^{\mathtt{o}} \cdot \mathbf{s}[t], \tag{2}$$

where f is a non-linear function (here an hyperbolic tangent) and $\mathbf{W}_{\mathtt{r}}^{\mathtt{r}}$, $\mathbf{W}_{\mathtt{o}}^{\mathtt{r}}$ and $\mathbf{W}_{\mathtt{r}}^{\mathtt{o}}$ are the *reservoir*, *feedback* and *readout* matrices, resp.

The main advantage of the ESN approach (and of Reservoir Computing in general) is that only the readout matrix $\mathbf{W}_{\mathtt{r}}^{\mathtt{o}}$ needs to be trained (the other matrices are created randomly beforehand). In order to generate the states used for training, the network is first run in a *teacher-forced* mode, by feeding back the desired output to the reservoir. The states are collected in a matrix $\mathbf{S}$ and the desired output in $\hat{\mathbf{O}}$. The readout matrix is then found, originally, by solving $\mathbf{W}_{\mathtt{r}}^{\mathtt{o}} \cdot \mathbf{S} = \hat{\mathbf{O}}$ in the least square sense. However, in order to improve the generalisation of the solution, we can apply a ridge regression technique [10] and instead solve

$$\mathbf{W}_{\mathtt{r}}^{\mathtt{o}} = \arg\min_{\mathbf{W}_{\mathtt{r}}^{\mathtt{o}}} \big(\|\mathbf{W}_{\mathtt{r}}^{\mathtt{o}} \cdot \mathbf{S} - \hat{\mathbf{O}}\|^2 + \lambda \cdot \|\mathbf{W}_{\mathtt{r}}^{\mathtt{o}}\|^2\big). \tag{3}$$

Once the readout matrix has been defined, the network can run in *free-run* mode, feeding back the actual output instead of the desired one. The optimal regularisation parameter λ is found by grid-search, by optimising a validation error computed on the first 3000 time steps of the free-run mode.

From equations (1) and (2), the state can be described recursively by

$$\mathbf{s}[t] = f\left(\left(\mathbf{W}_{\mathtt{r}}^{\mathtt{r}} + \mathbf{W}_{\mathtt{o}}^{\mathtt{r}} \cdot \mathbf{W}_{\mathtt{r}}^{\mathtt{o}}\right) \cdot \mathbf{s}[t-1]\right) \triangleq f\Big(\mathbf{W}_{\mathtt{loop}} \cdot \mathbf{s}[t-1]\Big) \tag{4}$$

The hyperbolic tangent non-linearity inside the reservoir ensures that the state will always be bounded. However, as the feedback matrix $\mathbf{W}_{\mathtt{o}}^{\mathtt{r}}$ has very low values (by construction, see later), the states used to train the readout matrix are very small, and the elements of the readout matrix $\mathbf{W}_{\mathtt{r}}^{\mathtt{o}}$ are thus large in order to project them back to the desired output signal. This means that, despite the output can not explode to unbounded values, it can drive the reservoir to saturation and reach a very high value. It is then stable around this large value.

There are actually 3 possible cases in the long run: (1) the system "explodes" and the reservoir is driven to saturation, (2) the activity fades out towards zero or (3) the output keeps the desired magnitude (at least for the length of the experiment - 16'000 time steps in free-run in this case).

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

2.1 Error measure and pruning

We evaluate the reservoir as follows: first, the reservoir runs in teacher-forced mode for 4'000 time steps. The first 1'000 time steps are discarded to get rid of the transients states and the readout matrix is trained on the remaining 3'000 steps. The reservoir runs then for 16'000 time steps in free-run mode. The mean square error (MSE) between the desired output and the actual output on the first 3'000 steps of free-run mode is the *regularisation validation error*, the MSE on the next 3'000 steps is the *pruning validation error* and finally the MSE on the remaining 10'000 steps is the *test error*.

We apply in the present contribution *exhaustive* pruning, i.e. we try to prune each neuron at each stage. When pruning neuron i, we remove its connection to the readout before training (i.e. we remove the i-th row of $\mathbf{S}$ used for training). Once trained, the network runs in free-run mode in order to compute the pruning validation error corresponding to pruning neuron i.

The pruning is applied as follows to a reservoir of size N:

1. The validation error of neuron i is computed for $i = 1, \ldots, N$.
2. The readout connection k corresponding to the neuron k with the minimal pruning validation error is pruned.
3. $N := N - 1$. If $N = 1$, stop; otherwise, go back to step 1.

Of course, the algorithm can be stopped earlier, if for instance the error increases significantly with the pruning.

One has to note that this requires to re-train and then re-run (in free-run) the whole network at each step and for each neuron (in order to compute the pruning validation error of each neuron). This algorithm is thus computationally demanding.

A possible optimisation comes from the solution of equation (3). The bottleneck when computing this solution is the product $\mathbf{S} \cdot \mathbf{S}^T$ [1]. Removing neuron i, i.e. removing the i-th row of $\mathbf{S}$ before computing the product is equivalent to computing the product and then removing the i-th row and i-th column of the resulting matrix. It is thus possible to only once compute the original product and then to apply the pruning directly on the result.

Another suboptimal approach is a heuristic based on ridge regression: one can remove the n readout connections with smallest weights after training with ridge regression. Another heuristic can use the Rayleigh criterion (like Fisher discriminant), however this only applies to classification tasks with a finite number of classes [7].

3 Experimental Results

We now present the results obtained by applying pruning to the Multiple Superimposed Oscillator (MSO) task [2, 11], where the goal is to generate a su-

[1] The number of neurons (i.e. the height of $\mathbf{S}$) is typically much smaller than the number of data samples (i.e. the length of $\mathbf{S}$)

perimposition of two sine waves, $\sin(0.2t) + \sin(0.311t)$. We present the results with and without regularisation. When there is regularisation, the optimal λ is found by grid-search, by optimising the regularisation validation error.

As a reservoir is created randomly, the results are averaged over 25 different random reservoirs[2]. In order to account for the different possible cases described sooner (i.e. "explosion", "fading out" and "stable"), we plot here the base-10 logarithm of the test error, i.e. of the mean square error between the desired output and the ESN output over the last 10'000 time steps. For reference, a "fading out" effect will lead to an error around 0.

The results are shown in Figure 1. In Figure 1(a), we can see that the regularisation helps to drastically decrease the error (the two solid lines), even without pruning. Then, when pruning is applied, the error is further decreased by 3 orders of magnitude (solid thick line). It is interesting to see that when there is no regularisation, pruning out connections (dotted thick line) can achieve the same performance as when there are both regularisation and pruning. Note that when using classical approach, the number in the abscissa is the numbers of neurons in the reservoir, but when using pruning it is the number of readout connections left unpruned while the reservoir has always 50 neurons. However, pruning can still decrease the error with respect to an unpruned reservoir of 50 neurons.

Figure 1(b) shows a comparison of different pruning techniques (with regularisation): exhaustive, ridge-based and random pruning as a baseline. We see that ridge-based pruning performs better than random pruning. It could thus be used in practice, when training time is an issue. However, exhaustive pruning is still significantly better than ridge-based pruning.

Figure 1(c) shows the average pair-wise correlation coefficient between the activation of the different neurons connected to the readout. The results with (thick line) and without (thin line) regularisation are shown. We observe that when there is no regularisation, the correlation decreases with pruning. This is because basic least square methods perform optimally if the input is decorrelated [10], and thus the pruning picks the most orthogonal neurons.

Finally, Figure 1(d) shows the average spectral radius of the loop matrix $\mathbf{W}_{\text{loop}}$. A stable ESN is characterised by a spectral radius close to 1. When there is no regularisation (thin line), the spectral radius is larger than one, as expected intuitively (as there is no weight decay). Pruning out connections can then help to decrease the spectral radius towards 1 and stabilise the ESN.

4 Discussion and conclusions

We applied a method to prune the connections from a reservoir to the readout in order to increase the generalisation ability while decreasing the hardware requirements. The results show that the pruning can be used as a way to further

[2] The reservoir is created as follows: each element of $\mathbf{W}_{\text{r}}^{\text{r}}$ is sampled from a normal distribution, and the whole matrix is then re-scaled to have a spectral radius of 0.9, and each element of $\mathbf{W}_{\text{o}}^{\text{r}}$ is sampled from a normal distribution and then divided by 100.

ESANN'2008 proceedings, European Symposium on
Artificial Neural Networks - Advances in Computational Intelligence and Learning
Bruges (Belgium), 23-25 April 2008, d-side publi., ISBN 2-930307-08-0.

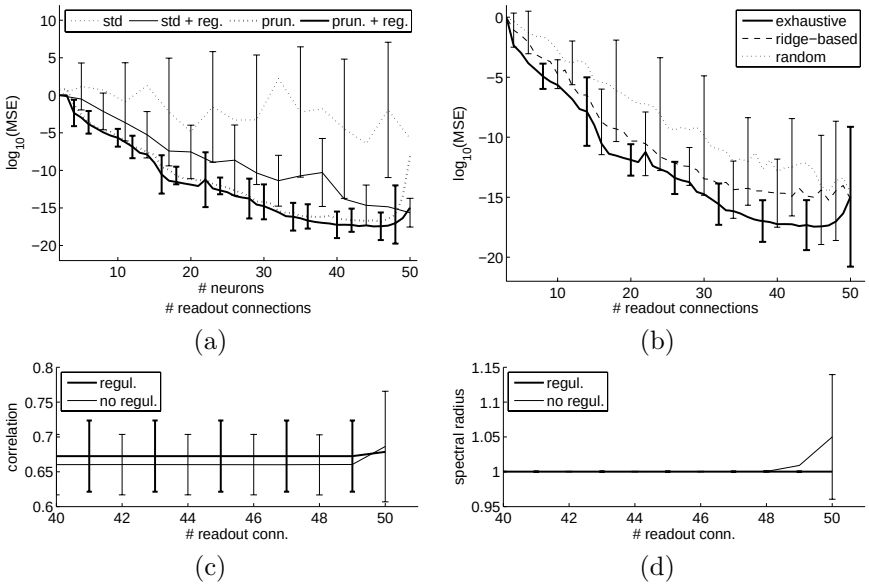

Fig. 1: Experimental results for the MSO task. The bars represent the standard deviation. (a) comparison of results with (thick lines) and without (thin lines) pruning, and with (solid lines) and without (dashed lines) regularisation; note that the abscissa gives the number of neurons in the reservoir for the results without pruning and the number of readout connections for the pruning case. (b) comparison of the different pruning methods: exhaustive, ridge-based and random. (c) average pairwise correlation coefficient for exhaustive pruning with (thick line) and without (thin line) regularisation. (d) spectral radius of the loop matrix with (thick) and without (thin) regularisation. Note that for (c) and (d), only the first pruning steps are shown and the vertical axis does not start at the origin.

regularise the readout and that it is best used along with another regularisation technique like ridge regression. For comparison purposes, random pruning has been tried, and it appeared that it is significantly different from the results obtained with the exhaustive pruning.

The main drawback of this technique is its computational requirement. However, it should be pointed out that this is only at training time. Once the readout is trained, the pruning only decreases the computational requirement in both software and hardware.

We have shown that the basic mechanism behind pruning in the case of basic linear readout is orthogonalisation of the sub-reservoir. This shows that reservoirs which are used with for example Least Mean Square regression need to be as decorrelated as possible.

As future work we plan to apply the above pruning rules to more complex real-world tasks to show that the better generalisation via pruning extends to real applications. We also envisage to further study the properties of optimally pruned sub-reservoirs and try do derive construction criteria or local learning rules which are able to construct reservoirs with the good properties found in these sub-reservoirs. This study will be a first step towards understanding what the important underlying properties of good reservoirs are.

References

[1] W. Maass, T. Natschläger, and H. Markram. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation*, 14:2531–2560, 2002.

[2] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical report, 2001.

[3] J. J. Steil. Backpropagation-Decorrelation: online reccurent learning with O(N) complexity. *Proc.IJCNN*, 1:843–848, 2004.

[4] M Lukoševičius and H. Jaeger. Overview of Reservoir Recipes. Technical report, 2007.

[5] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert. Optimization and applications of Echo State Networks with leaky integrator neurons. *Neural Networks*, 2007.

[6] J. J. Steil. Online Reservoir Adaptation by Intrinsic Plasticity for Backpropagation-Decorrelation and Echo State Learning. 2006.

[7] X. Dutoit, H. Van Brussel, and M. Nuttin. A First Attempt of Reservoir Pruning for Classification Problems. In *Proceedings of the 15th European Symposium on Artificial Neural Networks*, pages 507–512, Bruges, Belgium, 2007.

[8] D. Verstraeten, B. Schrauwen, and D. Stroobandt. Reservoir Computing with Stochastic Bitstream Neurons. In *Proceedings of the 16th Annual ProRISC Workshop*, pages 454–459, 2005.

[9] B. Schrauwen, M. D'Haene, D. Verstraeten, and J. Van Campenhout. Compact hardware for real-time speech recognition using a Liquid State Machine. In *Proceedings of the 20th International Joint Conference on Neural Networks*, page on CD, 2007.

[10] A. Hoerl and R. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 42:55–67, 1970.

[11] Y. Xue, L. Yang, and S. Haykin. Decoupled Echo State Networks With Lateral Inhibition. *Neural Networks*, 20:365–376, 2007.