ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

A First Attempt of Reservoir Pruning for Classification Problems

Xavier Dutoit, Hendrik Van Brussel, Marnix Nuttin *

Katholieke Universiteit Leuven - P.M.A.
Celestijnenlaan 300b, 3000 Leuven - Belgium

Abstract. Reservoir Computing is a new paradigm to use artificial neural networks. Despite its promising performances, it has still some drawbacks: as the reservoir is created randomly, it needs to be large enough to be able to capture all the features of the data. We propose here a method to start with a large reservoir and then reduce its size by pruning out neurons. We then apply this method on a prototypical and a real problem. Both applications show that it allows to improve the performance for a given number of neurons.

1 Introduction

Reservoir Computing (RC) [1, 2, 3] is a new paradigm with promising results [4, 5, 6, 7, 8]. However, as the reservoir is created randomly, it is still difficult to say a priori which characteristics the reservoir should have in order to perform the desired task successfully, so it typically needs to be large enough to ensure that all the necessary features of the data are present in the reservoir. Here we propose a way to prune out some neurons in the reservoir to reduce its size while trying to keep the performance steady.

In section 2, we will describe the RC model used here, and present a way to prune the reservoir. In section 3 we will apply the derived algorithm to two tasks to validate it. Finally, a conclusion is formulated in section 4.

2 Approach

2.1 RC model and dynamics

Amongst the different models of RC, we will use here the Echo State Network (ESN) as it is simpler to analyze from a theoretical point of view, and we will consider only classification tasks. The reservoir is described by an input matrix $\mathbf{I}$ and a connection matrix $\mathbf{C}$ and obeys the following equation:

$$\mathbf{s}(t+1) = f\big(\mathbf{I}\cdot\mathbf{i}(t) + \mathbf{C}\cdot\mathbf{s}(t)\big) \ ,$$

*This work was partially funded by the FWO Flanders project G.0317.05.,partially by the European IST Programme FET Project FP6-003758, and by the Belgian government under the Inter-University Attraction Poles, Office of the Prime Minister, IUAP-AMS.
The authors are grateful to the Mobile Learning Robots research group for providing the dataset and to Dirk Stroobandt, Benjamin Schrauwen, Michiel D'Haene and David Verstraeten for useful discussions and for providing the RC framework.

where $\mathbf{i}(t)$ is the input at time t, $\mathbf{s}(t)$ the current state (with $\mathbf{s}(0) = \mathbf{0}$), and $f(.)$ is a non-linear function (e.g. a hyperbolic tangent).

At each time step, the desired output is $\hat{o}(t) \in \{+1; -1\}$ and the actual output is given by a linear discriminant as $o(t) = \mathbf{W} \cdot \bar{\mathbf{s}}(t)$, where $\bar{\mathbf{s}}(t)$ is the augmented state vector, i.e. the state vector with one more bias element set to 1.

If the reservoir consists of n neurons, the state is then a vector in n dimensions. In this state space, we can collect the states obtained while running the network in two sets S^+ and S^- corresponding to all the states associated to a positive or negative desired output, resp (i.e. $S^\pm = \{\mathbf{s}(t)|\hat{o}(t) = \pm 1\}$).

Now we would like to prune out some neurons. In order to keep the performance as steady as possible, we should prune out "useless" neurons while keeping the "useful" ones. In order to determine which are the most useful neurons, we will use the Fisher criterion.

2.2 Fisher Discriminant

The problem is to classify two classes in n dimensions. The Fisher Linear Discriminant (FLD) is the vector $\mathbf{w}$ maximizing the following criterion:

$$J(\mathbf{w}) = \frac{\mathbf{w} S_B \mathbf{w}^T}{\mathbf{w} S_W \mathbf{w}^T} \ , \tag{1}$$

where S_B is the *between classes scatter matrix* and S_W is the *within classes scatter matrix*, defined as:

$$S_B = (\mu^+ - \mu^-)(\mu^+ - \mu^-)^T$$

$$S_W = \Sigma^+ + \Sigma^- \ ,$$

with $\mu^\pm$ the mean and $\Sigma^\pm$ the covariance matrix of the sets $S^\pm$.

2.3 Dimensionality reduction

Each neuron corresponds to a dimension in the state space (the i-th element of $\mathbf{s}(t)$ is the state of the neuron i at time t). For the purpose of classification, some dimensions of $\mathbf{s}(t)$ are more important than some other ones.

To determine the importance of a given dimension, we will consider the FLD criterion. In eq. (1), $J(\mathbf{w})$ is the discrimination quality along a vector $\mathbf{w}$. If we consider only the FLD of the neurons, i.e. the FLD only along a basis vector $\mathbf{e}_k$ (i.e. $e_k(i) = \delta_{ki}$, where δ is the Kronecker delta), it reduces to:

$$J(\mathbf{e}_k) = \frac{|\mu_k^+ - \mu_k^-|^2}{(\sigma_k^+)^2 + (\sigma_k^-)^2} \ , \tag{2}$$

with μ_k the mean and σ_k the standard deviation in dimension k.

So we can assign a discriminating power $J(\mathbf{e}_k)$ to each neuron k. Neurons with a high $J(\mathbf{e}_k)$ are more useful to classify correctly the input than neurons

with a low one. If we want to reduce the size of the reservoir while keeping its classifying power as steady as possible, we should thus prune out the neurons with lowest $J(\mathbf{e}_k)$. We propose the following pruning algorithm:

1. Run the reservoir with the input data $\mathbf{i}(t)$, record the states $\mathbf{s}(t)$.
2. Gather the states in two sets S^+ and S^-: $S^\pm = \{\mathbf{s}(t)|\hat{o}(t) = \pm 1\}$.
3. In each dimension i of $S^\pm$, compute the mean $\mu_i^\pm = E(S_i^\pm)$ and the standard deviation $\sigma_i^\pm = \sqrt{V(S_i^\pm)}$.
4. Compute the different discriminating powers according to eq. (2).
5. Determine the least discriminating neuron k: $k = \arg\min_i J(\mathbf{e}_i)$.
6. Remove neuron k from the reservoir
7. Go back to 1, or stop if the size of the reservoir is the desired size.

Alternatively, one can also remove at each iteration the K neurons minimizing $J(.)$ before going back to step 1 to run the reservoir again.

To know the desired size where a reservoir is small enough while the performance is still good enough, one can for instance look at the $J(\mathbf{e}_k)$ distribution and see how many are the most relevant ones when compared to the others.

3 Experimental Results

We now apply the derived idea to several cases: first, a prototypical case of frequency classification and then a real robotic problem of shared autonomy.

In each case, we apply the algorithm as follows: we evaluate the network with 10-fold cross-validation; for each fold, we assign a discriminating power to each neuron, according to eq. (2) (where we consider all the states obtained when running the reservoir with the current fold); we then remove the 10 neurons with least discriminating powers, averaged over the 10 folds, are minimal. We then run again a cross-validation with the reduced reservoir, and iterate until we reach a limit size (set here to 10 neurons).

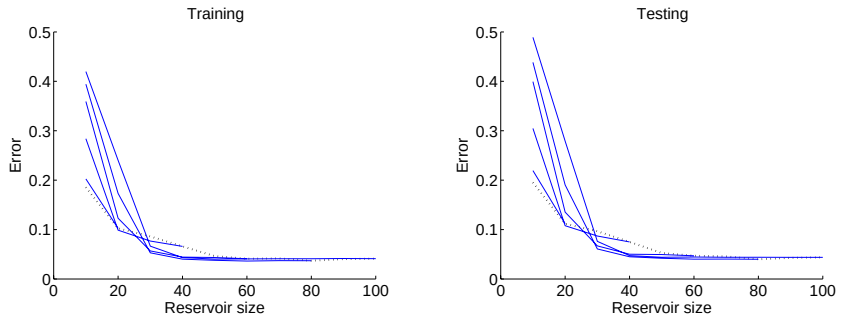
3.1 Frequency classification

We first consider a simple problem of classification of sines according to their frequencies. The data set consists of 50 samples of 20 steps. Each signal is a sine with an amplitude equal to 1 and with a period of either 10 or 20 time steps, and we add a Gaussian noise of 5% of the amplitude.

The network has to produce a positive output when the input is a sine of period 10, and a negative output when it is a sine of period 20. The error is the fraction of wrong output.

The reservoir is created with the following parameters: each element of $\mathbf{I}$ is $0, -0.1, +0.1$ with probability $0.2, 0.4, 0.4$ respectively. Each element of $\mathbf{C}$ is 0 with probability 0.9 and is drawn from a normal distribution otherwise. $\mathbf{C}$ is then divided element-wise by its spectral radius to make it equal to 1.

The results are shown in Fig. 1. The thick dotted line represents the error for unpruned reservoirs of size $n = 10, 20, \ldots, 100$. The thin lines, starting



(a) (b)

Fig. 1: Frequency classification: (a) training error, (b) testing error. The dotted thick line represents the original error, the thin lines represent the error after dimension reduction, starting from different sizes.

from the original sizes and going to the left, show the average error of the pruned reservoirs (where the reduction was made by deleting 10 neurons at each iteration), averaged over 50 simulations.

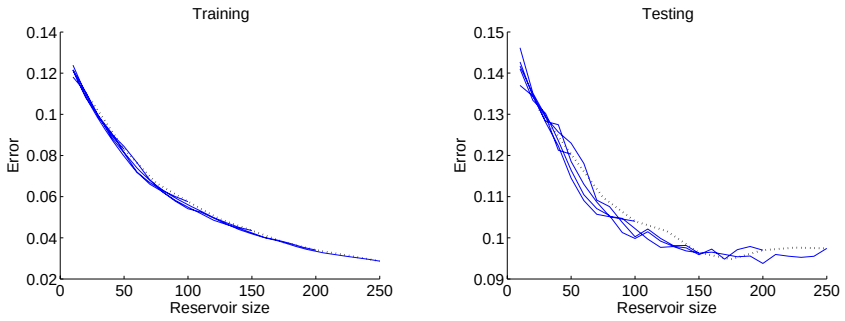
We see that the algorithm allows reducing significantly the size of the reservoir while keeping the performance steady, which increases the performance for a given size. For instance, the fraction of wrong output on the testing set for a reservoir of 40 neurons is around 7.5% (with a standard deviation $\sigma = 2.2\%$), but if we start with 80 neurons and prune out half of them, the error with 40 neurons decreases down to around 4.5% (with $\sigma = 1.3\%$).

On the other hand, we can see that there is a limit to this algorithm: if we prune out too many neurons, a pruned reservoir performs worse than an unpruned reservoir of the same size.

3.2 Real problem: adaptive shared autonomy

We now consider a real robotic problem of shared autonomy [9, 10, 11]. This kind of problems arises when both a human and an intelligent system are in control of a robot. Here the robot is an intelligent wheelchair. The human has a joystick to control the wheelchair, but the input is modified in order to simulate a disability. The wheelchair is able to perform obstacle avoidance, but the question is to know when this is actually required by the human. When the joystick command tends to drive the wheelchair to an obstacle (a wall, a table, etc.), there can be two causes: either it is the actual intention of the user to go to this object (e.g. to dock at a table), either the user wants to avoid it but is unable to give the correct joystick input because of the disability. The goal is to estimate, given the modified joystick input and some sensory inputs from the environment, the intention of the user.

For the experiment, the reservoir is provided with a PCA reduction of the



(a) (b)

Fig. 2: Shared autonomy task: (a) training error, (b) testing error. The dotted thick line represents the original error, the thin lines represent the error after dimension reduction, starting from different sizes. (mind the y axis)

joystick inputs and the sensors (lasers and sonars) of the wheelchair, and it has to produce a positive output when assistance is required (i.e. when the wheelchair should switch in obstacle avoidance behaviour), and a negative output otherwise.

The whole data set consists of 494 recorded runs along with the user intention for each run. The error is determined by the fraction of wrong output.

The reservoir is created with the same parameters as for the precedent experiment.

The results are shown in Fig. 2. The thick dotted line represent the error for unpruned reservoirs of size $n = 25, 50, \ldots, 200$. The thin lines, starting from the original sizes and going to the left, shows the average error of the pruned reservoirs (where the reduction was made by deleting 10 neurons at each iteration), averaged over 50 simulations.

Concerning the training error (Fig. 2(a)), the pruning only allows to slightly reduce the error for a given reservoir size. But for the testing error (Fig. 2(b)), the pruning allows to decrease the error more significantly. It sometimes even allows to decrease the error along with the size.

For instance, the average testing error obtained by creating randomly reservoirs of 125 neurons is 0.101 (with a standard deviation $\sigma = 0.011$). If now we start with reservoir of 150 neurons and prune it down to 120 neurons, the average error of the pruned reservoirs becomes 0.097 (with $\sigma = 0.11$).

For a reservoir of size 250, which seems to be over-fitted, the test error is 0.097. If we prune out some neurons to reduce the size to 200 neurons, the error also reduces to 0.094.

4 Conclusion

We introduced a new algorithm to prune neurons in a reservoir. Such an algorithm could help to simplify the search complexity when looking for an efficient

reservoir. Rather than starting with numerous small reservoirs, one could start with a small number of large reservoirs (large enough to ensure that the important features of the input data do appear in the reservoir).

We derived the pruning procedure from the ideas underlying the Fisher's Linear Discriminant. We then applied it to a prototypical and a real robotic problem, which both validated the proposed algorithm.

Although the error reduction is not large in this first implementation, we expect that the pruning algorithm can still be improved.

An interesting feature which appears is that for the testing error, when the reservoir is very large and seems to be over-fitted, pruning out some neurons can sometimes help to decrease the error. This should be further studied as pruning might help to regularize reservoirs.

An important point which is still to be studied is the problem of the reservoir dynamics: when neurons are removed, the dynamics of the resulting reservoir is changed. This point has not been addressed in the current paper, but should be further considered.

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(11):2531–2560, 2002.

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

[3] U.D. Schiller and J.J. Steil. On the weights dynamics of reccurent learning. In *ESANN'2003 Proceedings*, pages 73–78, 2003.

[4] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.

[5] H. Burgsteiner, M. Kröll, A. Leopold, and G. Steinbauer. Movement prediction from real-world images using a liquid state machine. In *Proceedings of the 18th Internal Conference IEA/AIE*. Springer-Verlag, Berlin, Germany, 2005.

[6] C. Fernando and S. Sojakka. Pattern recognition in a bucket. In *Proc. of the ECAL*, 2003.

[7] H. Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In *Proceedings of the ICEANN*, pages 129–136, 2005.

[8] P. Joshi and W. Maass. Movement generation and control with generic neural microcircuits. In *Proceedings of BIO-ADIT*, 2004.

[9] M. Nuttin, E. Demeester, D. Vanhooydonck, and H. Van Brussel. Obstacle avoidance and shared autonomy for electric wheel chairs: An evaluation of preliminary experiments. *Robotik 2002*.

[10] D. Vanhooydonck, E. Demeester, M. Nuttin, and H. Van Brussel. Shared control for intelligent wheelchairs: an implicit estimation of the user intention. In *Proc. of ASER 2003*, pages 176–182, 2003.

[11] E. Demeester, A. Hüntemann, D. Vanhooydonck, G. Vanacker, A. Degeest, H. Van Brussel, and M. Nuttin. Bayesian estimation of wheelchair driver intents: Modeling intents as geometric paths tracked by the driver. In *IROS 2006*.