

# Reservoir Size Reduction in Echo State Networks for Classification Problems

Xavier Dutoit, Hendrik Van Brussel, Marnix Nuttin

15th November 2006

## Abstract

Reservoir Computing is a new paradigm to use artificial neural networks. It has interesting performances, however, as the reservoir is created randomly, methods are still lacking to find a good reservoir. Here we propose a way to avoid this problem: starting with a reservoir whose size ensures a good performance, we propose a method inspired from Fisher's Linear Discriminant to reduce its size while keeping its performance steady. We then validate this algorithm by prototypical and real world examples.

## 1 Introduction

Reservoir Computing (RC) is a relatively new paradigm to use recurrent networks of neurons. The basic idea can be seen as a kernel method: rather than considering the input data, this data is projected into a high-dimensional space, and then a set of simple readouts functions are trained to extract the desired features.

RC has been introduced independently by [7] with the Liquid State Machine (LSM) where the reservoir is made of spiking neurons and synapses and by [5] with the Echo State Network (ESN) where the reservoir consist of sigmoidal neurons. It can also be linked with the results from [8] when they studied the weight dynamics of a recurrent neural network and to the *Backpropagation-Decorrelation* algorithm [9].

The main advantage of RC is that the reservoir itself is not trained, but instead we apply simple memoryless readouts on the reservoir to produce the desired output. Thus the training is dramatically simplified. It has been applied in various task [10, 2, 4, 1, 6].

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. We present here a way to avoid this problem: rather than creating and testing numerous small reservoirs, we propose instead to create a big one and then to prune it in order to reduce its size, while keeping the performance as steady as possible.

## 2 Approach

### 2.1 RC model and dynamics

We will consider the ESN, as it is more simple to analyze from a theoretical point of view, for a classification task. The reservoir is described by an input matrix  $I$  and a connection matrix  $C$ , and it obeys the following equation:

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

where  $\mathbf{i}(t)$  is the input at time  $t$ ,  $\mathbf{s}(t)$  the current state (with  $\mathbf{s}(0) = \mathbf{0}$ , and  $f(\cdot)$  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 space, we can group the states in two groups  $S^+$  and  $S^-$  corresponding to the ensemble of states associated to a positive or negative output, resp (i.e.  $S^\pm = \{\mathbf{s}(t) | \hat{o}(t) = \pm 1\}$ ). We can reasonably assume that the distributions are Gaussian:

**Assumption 1** *The states groups follow a normal distribution:*

$$S^+ \sim N(\mu^+, \Sigma^+)$$

$$S^- \sim N(\mu^-, \Sigma^-)$$

where  $N(\mu, \Sigma)$  denotes a Gaussian distribution with mean  $\mu$  and variance matrix  $\Sigma$ . We will restrain ourselves to the case where the connectivity is low, so the influence of a neuron on the other neurons can be neglected. That means we can remove a neuron without affecting the behaviour of the whole reservoir.

### 2.2 Fisher Discriminant

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

$$J(\mathbf{w}) = \frac{\mathbf{w} S_B \mathbf{w}^T}{\mathbf{w} S_W \mathbf{w}^T} \quad (2)$$

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

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

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

### 2.3 Dimensionality reduction

At each time step, the state of the network  $\mathbf{s}(t)$  is a point in  $n$  dimensions. Each dimension correspond to a given neuron (the  $i$ -th component of  $\mathbf{s}(t)$  is the state of the neuron  $i$  at time  $t$ ). For classification purpose, some dimensions of  $\mathbf{s}(t)$  are more important than some other ones. So the idea is to reduce the number

of dimensions, i.e. the number of neurons, by pruning out the most useless ones. To determine the importance of a given dimension, we will consider the FLD criterion. In eq. 2,  $J(\mathbf{w})$  is the discrimination quality along a vector  $\mathbf{w}$ . If we consider only basis vector  $\mathbf{e}_k$  (i.e.  $e_k(i) = \delta_{ki}$ , where  $\delta$  is the Kronecker delta), this reduces to:

$$J(\mathbf{e}_k) = \frac{|\mu_i^+ - \mu_i^-|^2}{(\sigma_i^+)^2 + (\sigma_i^-)^2} \quad (3)$$

with  $\mu_i$  being the  $i$ -th component of  $\mu$  and  $\sigma_i$  being the  $i$ -th component of  $\sigma$ . So the discriminating power of a neuron  $k$  is  $J(\mathbf{e}_k)$ . This means that neurons with a high  $J(\mathbf{e}_k)$  are more useful to classify correctly the input than neurons with a low one. Thus if we want to reduce the size of the reservoir while keeping its classifying power as steady as possible, we should prune out the neurons with minimal  $J(\mathbf{e}_k)$ . We propose thus the following pruning algorithm:

1. Run the reservoir with the input data  $\mathbf{i}(t)$ , record the states  $\mathbf{s}(t)$ .
2. Group the states in two groups  $S^+$  and  $S^-$ :

$$S^\pm = \{\mathbf{s}(t) | \hat{o}(t) = \pm 1\}$$

3. In each dimension of  $S^\pm$ , compute the mean and the deviation:

$$\mu_i^\pm = E(S_i^\pm) \quad i = 1, \dots, n$$

$$\sigma_i^\pm = \sqrt{V(S_i^\pm)} \quad i = 1, \dots, n$$

4. Compute the different discriminating power according to 3
5. Determine the less discriminating neuron  $k$ :

$$k = \arg \min_i d(i)$$

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  $d(\cdot)$  before going back to step 1. to run the reservoir again.

To know the desired size where the reservoir is small enough while the performance is still big 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 pattern classification and then a real robotic problem of shared autonomy.

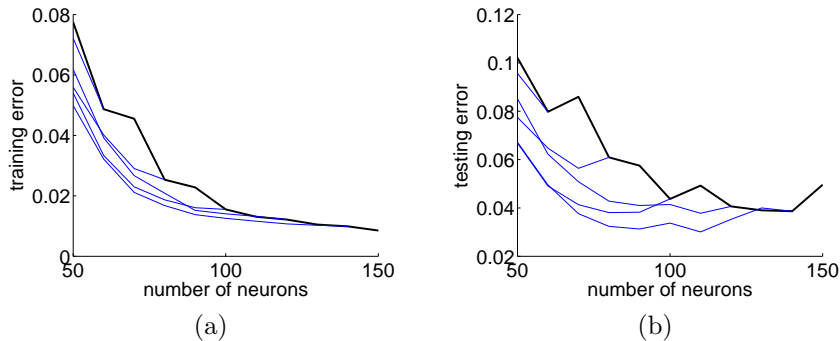


Figure 1: Pattern classification: (a) training error, (b) testing error. The solid thick line represents the original error, the thin lines represent the errors after dimension reduction.

Table 1: mean and standard deviation of the error for some original and reduced sizes - pattern classification

size	training error (%)	testing error (%)
60	<b>4.9</b> (3.4)	<b>8.0</b> (4.2)
100	<b>1.6</b> (1.2)	<b>4.4</b> (4.1)
60 (red. from 100)	<b>3.3</b> (2.7)	<b>4.9</b> (3.6)

**mean error** (standard deviation)

### 3.1 Pattern classification

We consider a simple classification problem: we create two random patterns, in this case two series of 100 random numbers. We then create a training sequence of 90 instances of those patterns (each instance being one of the two patterns with equal probability). We input this sequence into the ESN and train a linear discriminant to output a +1 or a -1 at each step of the sequence, depending on which pattern is the current one. Then, a testing sequence of 10 patterns is presented. The training and testing error are the proportion of misclassified pattern for the training and testing sequence, respectively.

The results, averaged over 50 simulations, are shown in Fig. 1. The thick line shows the original error before applying the algorithm. The thin lines show different dimension reduction with the algorithm. At each iteration,  $K = 10$  neurons were removed. The table 1 shows the original error for a size of 60 neurons and then the results when starting from a reservoir of 100 neurons and reducing it to 60 neurons: So we see that this algorithm can reduce the mean error and also slightly the variance. It performs especially well on the testing set, which suggest that it helps to discard neurons which take part in over-fitting. Sometimes for the testing error, when the starting size is big enough, the first reduction step allows even to have a smaller reservoir which performs better.

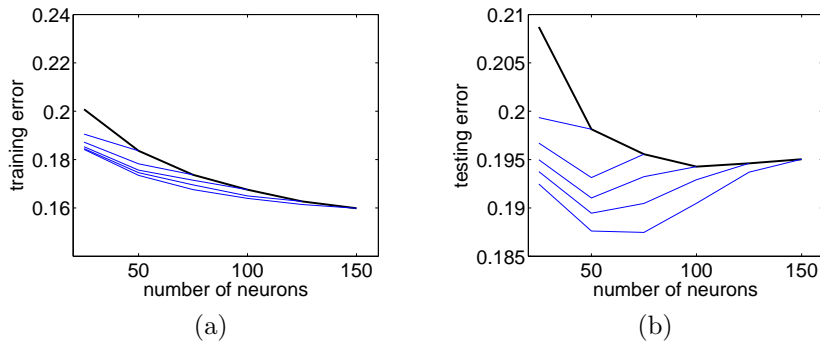


Figure 2: Shared autonomy: (a) training error, (b) testing error. The solid thick line represents the original error, the thin lines represent the error after dimension reduction.

### 3.2 Real problem: adaptive shared autonomy

We will now consider a real robotic problem of adaptive shared autonomy [3]. This kind of problems arises when both a human and an intelligent system are in control of a robot. In our case we will consider 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. So the goal is to estimate, given the scrambled joystick input and some sensory inputs from the environment, whether the intention of the user is to approach a given object or to avoid it.

For the experiment, the reservoir is provided with a PCA reduction of the joystick inputs and the other sensors (lasers and sonars), 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 consist of 494 recorded runs along with the user intention for each run. The reservoir is trained with 10-folds cross-validation. The error is determined by the proportion of wrong output.

The results are shown in Fig. 2. The thick line shows the original error before applying the algorithm. The thin lines show different dimension reduction with the algorithm. At each iteration,  $K = 25$  neurons were removed.

We can see that when we start with a reservoir of 150 neurons, the testing error first increases (at 125 neurons) more than the original error, and then it decreases when we removes neuron. This might comes from the fact that the reservoir is over-fitting the training data and thus performs better on the testing data with a few neurons pruned.

The table 2 shows the original error for a size of 50 neurons and then the results when starting from a reservoir of 100 neurons and reducing it to 60 neurons:

Table 2: mean and standard deviation of the error for some original and reduced sizes - shared autonomy

size	training error (%)	testing error (%)
50	<b>18.4</b> (0.8)	<b>19.8</b> (1.0)
100	<b>16.8</b> (0.6)	<b>19.4</b> (0.9)
50 (red. from 100)	<b>17.6</b> (0.6)	<b>19.1</b> (0.9)

**mean error** (standard deviation)

## 4 Conclusion

We introduced a new algorithm to reduce the search complexity when trying to find an efficient reservoir. To the best of our knowledge, it's the first attempt for such a task. Rather than trying randomly a relatively high number of reservoirs, we proposed here an algorithm to prune a big reservoir in order to reduce its size while keeping the most relevant neurons.

The idea has first been theoretically derived, and then applied to a prototypical and a real problems. Both experimental applications validated the proposed algorithm.

On the testing data, in some cases, this algorithm allowed to reduce the error as well as the size of the reservoir. This might be a way to avoid over-fitting. This question should further be studied.

And the algorithm could also maybe be refined, as there are still a few cases where it does not seem to work efficiently.

Despite those open questions, the algorithm in its current form has been validated for a real problem, and we think it might be interesting to study its applications in more details.

## 5 Acknowledgements

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 thankful to the MLR research group for providing the dataset (<http://www.mech.kuleuven.be/mlr/>).

## References

- [1] H. Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In *Proceedings of the EANN*, 2005.
- [2] 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.
- [3] E. Demeester, M. Nuttin, D. Vanhooydonck, and H. Van Brussel. Fine

motion planning for shared wheelchair control: Requirements and preliminary experiments. In *11th International Conference on Advanced Robotics (ICAR)*, pages 1278–1283, 2003.

- [4] C. Fernando and S. Sojakka. Pattern recognition in a bucket. In *Proc. of the ECAL*, 2003.
- [5] H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD 148, German National Research Center for Information Technology, 2001.
- [6] P. Joshi and W. Maass. Movement generation and control with generic neural microcircuits. In *Proceedings of BIO-ADIT*, 2004.
- [7] 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.
- [8] U.D. Schiller and J.J. Steil. On the weights dynamics of recurrent learning. In *ESANN'2003 Proceedings*, pages 73–78, 2003.
- [9] J.J. Steil. Backpropagation-decorrelation: online recurrent learning with  $o(n)$  complexity. In *Proc. IJCNN*, volume 1, pages 843–848, 2004.
- [10] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *preprint submitted to Elsevier Science*, 2005.