Unsupervised Learning of Echo State Networks: Balancing the Double Pole

Fei Jiang[1,2], Hugues Berry[2], Marc Schoenauer[1]

[1]Project-Team Alchemy,
INRIA Saclay, Parc Orsay Université
28, rue Jean Rostand
91893 Orsay Cedex, France

[2] Project-Team TAO
INRIA Saclay & LRI (UMR CNRS 8623)
Bât 490, Université Paris-Sud
91405 Orsay Cedex, France

FirstName.LastName@inria.fr

ABSTRACT

A possible alternative to fine topology tuning for Neural Network (NN) optimization is to use Echo State Networks (ESNs), recurrent NNs built upon a large reservoir of sparsely randomly connected neurons. The promises of ESNs have been fulfilled for supervised learning tasks, but unsupervised learning tasks, such as control problems, require more flexible optimization methods. We propose here to apply state-of-the-art methods in evolutionary continuous parameter optimization, to the evolutionary learning of ESN. First, a standard supervised learning problem is used to validate our approach and compare it to the standard quadratic one. The classical double pole balancing control problem is then used to demonstrate that unsupervised evolutionary learning of ESNs yields results that compete with the best topology-learning methods.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and neural nets*

General Terms

Algorithms, Design

Keywords

Echo State Networks, Evolution Strategies, CMA-ES

1. SUPERVISED LEARNING OF ESN

Echo state networks (ESNs) [4] are discrete time, continuous state, recurrent neural networks in which the input layer is totally connected to a hidden layer called *reservoir*, that is itself totally connected to the output layer. The connectivity and weights of the connections within the reservoir are randomly generated during ESN building and remain constant during learning. The main point in ESN is that only the weights going from reservoir (hidden) nodes to the output ones are to be learned. Any supervised learning problem using some mean-squared error objective is thus reduced to a quadratic optimization that can be quickly solved by any deterministic optimization procedure, even for very large reservoirs. ESNs have been shown to perform surprisingly well in such context of supervised learning, in particular for problems of prediction of time series [4].

The idea of Evolutionary Learning for Echo State Networks amounts to replace the gradient descent that is used to optimize the outgoing weights in the classical ESN approach by an Evolutionary Algorithm (EA). As a first toy example, we reproduce one of the initial settings of the seminal ESN paper [4] in which the network is trained to produce an univariate time-series output, $y_{teach}(n) = \frac{1}{2}u^7(n)$ (where n denotes time) from an univariate sinusoidal input, $u(n) = \sin(n/5)$. The actual network output $y(n)$ is computed from the reservoir neurons according to $y(n) = f(\sum_{i=1}^{N} w_i^{out} \times x_i(n))$ where w_i^{out} denotes the weight of the i-th output connection, $x_i(n)$ is the state of i-th neuron at time step n, $f(x) = (1 - e^{-ax})/(1 + e^{-ax})$ and a is the half-slope of f at the origin. As in [4], the fitness to minimize is the Mean Square Error of the network that is computed between time steps 101 and 300 according to $MSE_{train} = 1/200 \sum_{n=101}^{300} (y(n) - \text{atanh}(y_{teach}(n)))^2$.

We have compared three variants of the ESN evolutionary optimization: (i) optimizing the output weights only, denoted *Std* in the following; (ii) optimizing the output weights *and* the spectral radius ρ of the reservoir, denoted *Rho*; and (iii) optimizing the sigmoidal slopes a of the function f, denoted *Slopes*. With a reservoir size of $N = 100$ neurons, we observed (not shown) that CMA-ES (*Std*) can be as precise as the gradient method reported in [4] (i.e. with a MSE of the order of 10^{-15}), though undoubtedly requiring a much greater computational effort. Interestingly, the results also show that optimizing only the reservoir slopes (*Slopes*) yields precisions that are only similar to the original ESN learning method. Note however that with smaller reservoir sizes (e.g. $N = 30$), optimizing the reservoir neuron slopes (*Slopes* variant) yielded better fitness than the standard procedure (not shown). Finally, our results also evidence that increasing the search space fails to improve precision: the *Rho* variant yields the worst precision in this supervised task.

Table 1: Experimental results for the double pole balancing. Reservoir size was $N = 20$ neurons

	Cheap	Fitness			New	Fitness		
Variant	Avg. Eval.	Std. Dev.	Genera-lization	% success	Avg. Eval.	Std. Dev.	Genera-lization	% success
Std	14960	6291	234	6.8%	16303	11511	209	82.3%
Rho	23571	10175	241	52.7%	19796	6770	224	91.4%
Std - Opt	19168	21782	232	9.5%	15965	11813	208	86.8%

2. UNSUPERVISED LEARNING OF ESN

The double pole balancing problem without velocity information is a challenging task that is commonly used to compare different neuroevolution methods that evolve both the topology and the weights of neural networks [5, 3, 1]. The system consists of a cart moving along the x axis, and two poles of different lengths and masses that are connected to the car by an hinge. The poles have one degree of freedom (their angle w.r.t. the vertical). The challenge is to keep both poles up (i.e. within given bounds for the angles) as long as possible using the ESN output, which is interpreted as a force applied to the cart. To avoid heavy computational cost, many, if not all, previous works have used a simplified fitness (referred to as F_{cheap} below), where the controller performance only depends on its ability to maintain the poles up during a single trial of $1,000$ time steps. Only the best individual in the population (for this fitness) is then picked up and evaluated through two generalization tests: a first test is passed if the individual keeps the system within the success domain during $100,000$ further time steps. The second test is passed if the controller as well succeeds in balancing the system for $1,000$ time steps starting from 625 different initial positions. When the best individual in the population succeeds for at least 200 of those 625 trials, the run is stopped and this individual is returned as the solution. However, some individuals commonly obtain a very high fitness but fail on the second test, while some others pass all 200 generalization tests with a rather low F_{cheap}. Hence we have also used a new fitness ($F_{gen.}$) that takes into account all 3 tests describe above: $F_{gen.} = F_{cheap} + 10^{-5}n_I + 30n_S/625$ where n_I is the number of iterations where the system was maintained within the success domain during the first generalization test, and n_S is the number of generalization trials passed by the controller during the second one. The constant 30 was chosen by trial and error.

The results are summarized in Table 1. For each variant, the columns show the number of needed evaluations averaged over the successful runs (column *Avg Eval.*), its standard deviation (*Std Dev.*), the number of tests (out of 625) passed during the third generalization test (*Generalization*), and, most importantly, the percentage of success (*% success*), i.e. of runs where the best individual did pass the 3 tests. Using the *Cheap Fitness*, table 1 shows that the *Std* variant (output weight minimization) obtains poor performance results. The only variant that can compete with other published results is the *Rho* one: more than half of the runs succeeded, with an average cost of $23,571$ evaluations, which is still worse than NEAT [5] and AGE [1], but within the same order of magnitude.

As expected, the results really improve with the new fitness ($F_{gen.}$), that takes into account the generalization ability of the network: the *Rho* variant almost always finds a solution (except for one run out of 220), and the *Std* variant improves a lot over its results with the cheap fitness. More importantly, all variants reach performances that are at the level of those of NEAT [5], AGE [1] or a totally recurrent network with 9 neurons [3].

It has always been advocated by ESN pioneers that the upper bound on the spectral radius was important for successful ESN use. However, the most remarkable fact here is that for all settings, the *Rho* variant, that explicitly optimizes the spectral radius, almost always gives the best results. This is surprising when compared to the situation in the supervised context (see above), where the *Rho* variant performed the worst of all. Further experiments were run, using the *Std* variant but fixing the Spectral Radius to the final value found by the *Rho* method (see the lines "*Std – Opt*" in Table 1). Though it generally slightly improves the results over the *Std* variant, it does not allow to reach the same level of performance than the *Rho* method itself. The important feature is thus that ρ is allowed to vary during the optimization, and not the final value it reaches.

Finally, the influence of the initialization of the topology of the reservoir is clearly large. Indeed, in the case of methods with low performance, all the successful runs often stem from a small number of initial reservoir topologies, while a majority the initial reservoir topologies fail to generate even a single success. Together with the differences noted in the supervised learning context, this makes a clear picture that the topology of the reservoir matters. Why, and how to take advantage of this fact, is left to further work.

3. REFERENCES

[1] P. Dürr, C. Mattiussi, and D. Floreano. Neuroevolution with Analog Genetic Encoding. In Th. Runarsson et al., editor, *PPSN IX*, pages 671–680, 2006.

[2] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao et al., editors, *PPSN VIII*, pages 282–291, 2004.

[3] C. Igel. Neuroevolution for reinforcement learning using evolution strategies. In *Proc. CEC'03*, pages 2588–2595. IEEE Press, 2003.

[4] 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.

[5] K. O. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In W. B. Langdon et al., editor, *Proc. GECCO'02*, pages 569–577. Morgan Kaufmann, 2002.