Supervised and Evolutionary Learning of Echo State Networks

Fei Jiang, Hugues Berry, Marc Schoenauer

Supervised and Evolutionary Learning of Echo State Networks

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

[1] Alchemy, INRIA Saclay, 28, rue Jean Rostand, 91893 Orsay Cedex, France
[2] TAO,INRIA Saclay & LRI (UMR CNRS 8623),Bât 490, Université Paris-Sud, 91405 Orsay Cedex, France

Abstract. A possible alternative to topology fine-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 ones, e.g. control problems, require more flexible optimization methods – such as Evolutionary Algorithms. This paper proposes to apply CMA-ES, the state-of-the-art method in evolutionary continuous parameter optimization, to the evolutionary learning of ESN parameters. First, a standard supervised learning problem is used to validate the approach and compare it to the standard one. But the flexibility of Evolutionary optimization allows us to optimize not only the outgoing weights but also, or alternatively, other ESN parameters, sometimes leading to improved results. The classical double pole balancing control problem is then used to demonstrate the feasibility of evolutionary (i.e. reinforcement) learning of ESNs. We show that the evolutionary ESN obtain results that are comparable with those of the best topology-learning methods.
Keywords: Neural networks, Evolutionary algorithms, Control

1 Introduction

It has long been known to Neural Networks practitioners that a good design for the topology of the network is an essential ingredient for a successful application of Neural Networks to a given learning task. The critical issue then becomes that of learning the appropriate weights. Echo State Networks (ESNs) [12], that were recently proposed for supervised learning of time series, can be seen as an alternative approach based on a large *reservoir* of neurons with random, constant (non-learned) and sparse connectivity. Learning is thus restricted to the outgoing connections only. In the supervised learning case, this efficiently transforms the learning process into a simple quadratic optimization problem. The situation changes dramatically with unsupervised learning tasks, such as control ones: no input-output example being available, the learning problem can no longer be set as quadratic. Evolutionary Computation provides a possible solution for such situations, as long as some fitness is available. This paper addresses the following issues: are Evolutionary Algorithms (EAs) a viable method to train ESNs in

general and on reinforcement learning tasks in particular? Furthermore, are ESNs an alternative to topology learning in the framework of control problems? Finally, Evolutionary Algorithms can learn to adjust more than just the weights of the outgoing connections of the ESN. Does this improve the learning power of ESNs?

The paper is organized as follows. Section 2 introduces ESNs and our evolutionary algorithm. The supervised task case (time-series prediction) is addressed in Section 3. Moreover, Evolutionary Learning opens up the field of reinforcement learning to ESNs. We address this issue with a canonical example, the double pole balancing problem [17, 10, 4], in Section 4. Finally, Section 5 sums up the paper and sketches directions for on-going and further researches.

2 Background

2.1 Echo State Networks

Echo state networks (ESN) are discrete time, continuous state, recurrent neural networks using a sigmoidal activation function for all neurons [12]. A typical ESN is shown in figure 1: the input layer is totally connected to the hidden layer (the *reservoir*) whose neurons are themselves totally connected to the output layer. Note that the output layer can also be connected backward to the reservoir. To generate the reservoir, one connects N neurons randomly (with independent uniform distribution) up to a user-defined connection density α. The weight of these connections are randomly chosen, then scaled so that the spectral radius of the reservoir, ρ (i.e. the largest modulus among the eigenvalues of the reservoir weight matrix) is less than a prescribed value < 1 (see e.g. [13]). The main point in ESN is that only the weights from the reservoir nodes to the output ones are to be learned. Any supervised learning problem using some mean-square error objective thus reduces to a quadratic optimization problem that can be quickly solved by any deterministic optimization procedure, even for very large values of N. ESNs have been shown to perform surprisingly well in the context of supervised learning, in particular for time series prediction. They have also been successfully used in the context of (supervised) robot control learning [14]. The idea beyond Evolutionary Learning for Echo State Networks is to replace the gradient descent used to optimize the outgoing weights in Jaeger's approach [13] by an Evolutionary Algorithm (EA). We first present the Evolutionary Algorithm that will be used throughout the paper, the Covariance Matrix Adaptation Evolution Strategy, aka CMA-ES.

2.2 CMA-ES

The CMA-ES is a well-established and state-of-the-art Evolutionary Algorithm in continuous domain evolutionary computation [8, 9, 7]. At iteration step t, $\lambda > 1$ offspring individuals $\boldsymbol{x} \in \mathbb{R}^n$ are generated by sampling a multi-variate normal distribution: $\boldsymbol{x} = \boldsymbol{m}^t + \sigma^t \times \mathcal{N}(\mathbf{0}, C^t)$, where $\boldsymbol{m}^t$ is the average of the best individuals of the previous generation, $\mathcal{N}(\mathbf{0}, C^t)$ is a normally distributed

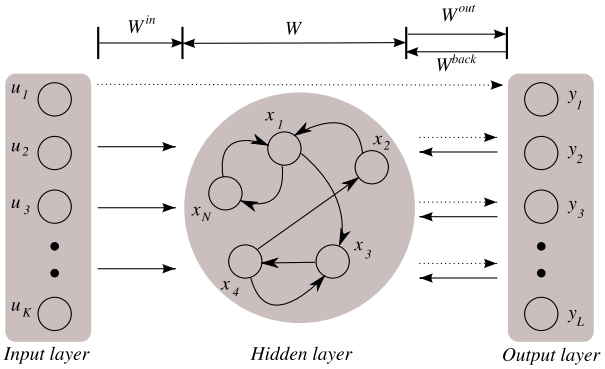

Fig. 1. Schematic view of an Echo State Network. Plain arrows stand for weights that are randomly chosen and remain fixed, while dashed arrows represent the weights to be optimized.

variable with mean $\mathbf{0}$ and $n \times n$ covariance matrix C^t, and $\sigma^t > 0$ is a scaling parameter, the step-size. After those λ individuals have been sampled, evaluated on f, and sorted according to their objective function values, the distribution parameters $\boldsymbol{m}^t$, σ^t, and C^t are updated for a new iteration step using the sorted population and cumulated information about the whole optimization path. It has been shown experimentally that the covariance matrix C^t approximates the inverse of the Hessian matrix of the problem at hand near the optimum, and CMA-ES can hence be considered a quasi-second order optimization method. Importantly, CMA-ES is almost a parameter-free algorithm. Only the number of offsprings λ is crucial to the evolution success and must possibly be adapted to account for the ruggedness of the fitness landscape at hand. In our case, the default value [9], that increases logarithmically with the dimension n of the problem (number of unknown parameters): $\lambda = 4 + 3\ln(n)$, was found to be well adapted.

3 Supervised Learning of ESN

In order to validate the Evolutionary approach to ESN learning, we first replicate Jaeger's initial setting [12], but using an Evolutionary Algorithm in lieu of its gradient-based quadratic optimization procedure.

3.1 The Original Settings

In this toy example, the aim is to train the network to produce a univariate time-series output, $y_{teach}(n) = \frac{1}{2}u^7(n)$ (where n is time) from a univariate input given by $u(n) = \sin(n/5)$. The network output is given by $y(n) = f(\sum_{i=1}^{N} w_i^{out} \times x_i(n))$, where w_i^{out} denotes the weight of the i-th output connection, $x_i(n)$ is the state of the i-th neuron, $f(x) = (1 - e^{-ax})/(1 + e^{-ax})$

and a is the half-slope of f at zero activation. Like in Jaeger's original paper, the reservoir consists of $N = 100$ randomly connected neurons (independent uniform distribution). Its weights are set to 0, +0.4 or -0.4 with probabilities 0.95, 0.025, 0.025 respectively (sparse connectivity of 5%). They are then scaled so that the spectral radius of the reservoir is $\rho \approx 0.88$. The input weights (from the input to all neurons in the reservoir) are set to $+1$ or -1 with equal probability. Direct links from inputs to outputs or backward links from outputs to the reservoir are not used here. The fitness to minimize is the Mean Square Error of the network, computed between time steps 101 and 300: $\text{mse}_{train} = 1/200 \sum_{n=101}^{300} (y(n) - \text{atanh}(y_{teach}(n)))^2$.

3.2 Which parameter to optimize?

In Jaeger's original paper [12], the output weights were optimized with a gradient method, resulting in a reported error mse $\approx 3.5 \times 10^{-15}$ [12]. But a critical parameter in ESN tuning seems to be the spectral radius, that is usually advised to be < 1 [12] though different values have been proposed in the literature for different problems. Hence it seems a good idea to use the spectral radius as a free parameter to be optimized by CMA-ES: it only adds one dimension to the problem. The procedure goes as follows: the weights of the recurrent connections within the reservoir are first scaled so that the spectral radius of the connection matrix takes the value prescribed by the additional optimized parameter. The weights are of course set back to their original values before the evaluation of next individual. Jaeger's original sigmoidal function was tanh, corresponding to the case $a = 2$ for the transfer function f above. However, if both the output weights and the sigmoid slopes a are optimized, the dimension of the optimization problem is twofold. Hence, we examined the case where only the slopes a are optimized.

3.3 Comparative Measures

Because CMA-ES, like all EAs, is a stochastic optimization procedure, no strong conclusion can be drawn in absence of a thorough statistical analysis of the performances. Here 15 different networks have been used, and for each network, 5 runs of CMA-ES were launched with different random seeds (and hence starting points). To have a global performance measure, we used an estimator of the success performance called the "SP1 measure" [7, 1], which is the number of evaluation of the fitness function that is needed to reach a given fitness level, divided by the fraction of runs that did reach that fitness value. SP1 can thus be viewed as the computational effort required to reach a given performance level.

3.4 Results

Three variants of the ESN evolutionary optimization have thus been compared: *(i)* optimizing the output weights only, denoted *Std* in the following; *(ii)* optimizing the output weights *plus* the spectral radius, denoted *Rho*; and *(iii)*

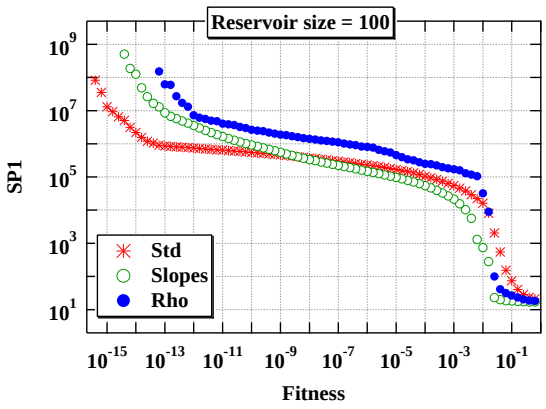

Fig. 2. Comparative SP1 measures for the case $N = 100$, in log-log scale.

optimizing the sigmoidal slopes only, denoted *Slopes*. Figure 2 shows the SP1 plots for a 100 neuron reservoir and confirms that CMA-ES (*Std*) can be as precise as the gradient method reported in [12] (i.e. with a *mse* of the order of 10^{-15}), though undoubtedly requiring a much greater computational effort. Interestingly, the results show that optimizing only the reservoir slopes (*Slopes*) yields precisions that are also 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 even better fitness than the standard procedure (not shown). This, however, has a cost and requires almost 100-folds more evaluations, due to the fact that very few runs do find such low fitness values. Finally, Figure 2 also evidences that increasing the search space fails to improve precision: the *Rho* variant yields the worst precision in this supervised task. Taken together, these results validate the use of Evolutionary Learning for supervised tasks. We now turn to the study of a reinforcement learning task.

4 Reinforcement Learning of ESN

The double pole balancing problem without velocity information is a benchmark learning task for the evaluation of neuroevolution methods - i.e. methods that evolve both the topology and the weights of neural networks [17, 6, 5, 10, 4]. Albeit they don't belong to supervised learning methods, evolutionary methods are based on the evaluation of some individual fitness. This fitness can be considered as a feedback or a kind of reward emitted by the environment, so that such neuroevolution methods are considered as reinforcement learning methods. The system consists of a cart (mass =1 kg) moving along the x axis, and two poles of different lengths ($l_1 = 1\,\mathrm{m}$, $l_2 = 0.1\,\mathrm{m}$) and masses ($m_1 = 0.1\,\mathrm{kg}$, $m_2 = 0.01\,\mathrm{kg}$) that are connected to the cart by a hinge. The poles have a single degree of freedom (their angle θ_1 and θ_2 w.r.t. the vertical). The challenge is to keep

both poles up (i.e. within given bounds for their angles) as long as possible using the ESN output, which is interpreted as a force F_x applied to the cart ($F_x \in [-10\,\mathrm{N}, 10\,\mathrm{N}]$). In all experiments (in this paper as well as in previous works), the dynamics of this mechanical system was solved using fourth-order Runge-Kutta method with a step size of 0.01 s.

4.1 Fitness(es)

To avoid heavy computational cost many, if not all, previous works in the evolutionary literature addressing the double pole balancing problem [6, 5, 17, 10, 4] have used a simplified fitness (thereafter referred to as F_{cheap}): a single trial is run for every individual in the population, starting from the same state ($\theta_1(0) = 4.5^o, \dot{\theta}_1(0) = \theta_2(0) = \dot{\theta}_2(0) = x(0) = \dot{x}(0) = 0$). The simulation stops if one of the poles falls, i.e. the system leaves the success domain $x \in [-2.4\,\mathrm{m}, 2.4\,\mathrm{m}]$ and $\theta_1, \theta_2 \in [-36^o, 36^o]$ (no solution found) or if the poles remain up for $1,000$ time steps (successful individual). The fitness function F_{cheap} is then:

$$F_{cheap} = 10^{-4}t + 0.9 f_{stable}, \text{ with}$$

$$f_{stable} = \begin{cases} 0 & \text{if } t < 100 \\ \dfrac{0.75}{\sum_{i=t-100}^{t} \left(|x(i)| + |\dot{x}(i)| + |\theta_1(i)| + |\dot{\theta}_1(i)|\right)} & \text{otherwise} \end{cases}$$

where t denotes the number of time steps during which the system remained inside the success domain and f_{stable} quantifies the cart stability during the last 100 time steps. At every generation, the best individual for fitness F_{cheap} undergoes two generalization tests. The 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 individual as well succeeds in balancing the system for $1,000$ time steps starting from 625 different initial positions. When one individual succeeds for at least 200 of those 625 trials, the run is stopped and this individual is returned as the solution.

However, though this simplified fitness does save a lot of computational resources, it is a poor fitness with respect to the overall goal of the optimization. For instance, individuals are commonly obtained that have a very high fitness but never pass the first generalization test, while some others pass all generalization tests but with a rather low F_{cheap}. Hence we propose here a new fitness ($F_{gen.}$) that takes into account all 3 tests described above: $F_{gen.} = F_{cheap} + 10^{-5} n_I + 30 n_S/625$ 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 constants 10^{-5} and 30 were chosen by trial and error.

4.2 Experimental conditions

The size of the reservoir was fixed here to $N = 20$: initial experiments indicated that larger reservoirs did not improve the results. To study the variability with

respect to the reservoir topology, 20 different reservoirs were generated and 11 independent runs of CMA-ES were made for each reservoir. Each reservoir was initialized as described in section 3.1, except for the fixed weights: here, the reservoir connectivity was 10% and non-zero weights were randomly initialized between $[-1, 1]$. At the beginning of each run, the activity of all neurons in the reservoir was zeroed, and the network was run for 20 iterations before control actually began and the fitness started to accumulate. As mentioned in Section 2.2, CMA-ES is almost a parameter-free algorithm. However, Igel advises in his paper [10] to impose a lower bound on the actual lower eigenvalue of the covariance matrix during CMA-ES runs. Indeed, our preliminary results confirmed that, without this constraint, the solutions systematically evolves toward "Bang-Bang" types of motor control, that do not seem very efficient for the task at hand. We were able to solve this issue by imposing a lower bound of 0.05 on the step-size σ. Finally, the presented results were obtained with the *Std* and *Rho* variants of the evolutionary ESN learning (Section 3.2).

4.3 Results and Discussion

All results are summarized in Table 1. Every line in the table gives the results of one variant of the algorithm (two spectral radii, 0.6 and 0.95 were tried for the *Std* variant, and variant *Std - Opt* will be discussed later). For each variant, the 220 runs (11 runs for each of the 20 different reservoir initializations) are here grouped together. Each sub-table shows the average 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 F_{cheap}, a first striking result is the very low performance of the *Std* variants (whatever the spectral radius): less than 7% of the runs did pass the 3 generalization tests in these cases. Things are better for the *Rho* variant: more than half of the runs succeeded, with an average cost of $23,571$ evaluations, which amounts to an SP1 value of about $45,300$. This value is still worse than NEAT ($\approx 33,000$ evaluations [17]) and AGE ($\approx 25,000$ evaluations [4]), but within the same order of magnitude.

As expected, the results really improve when using the new fitness, that takes into account the generalization ability of the network: the *Rho* variant almost always find a solution (except for one run out of 220). Even the *Std* ones improve a lot over their results with the cheap fitness. More importantly, using the new fitness allows all variants to reach performances that are comparable to those of NEAT ($\approx 33,000$ evaluations [17]) and AGE ($\approx 25,000$ evaluations [4]), though of course those results can hardly be compared, as they were obtained using a different fitness. Indeed, the found SP1 values for *Std-0.60*, *Std-0.95*, and *Rho* variants respectively were $19,342$, $19,808$ and $21,658$.

Spectral Radius. It has always been advocated by ESN pioneers that the upper bound on the spectral radius was important for successful ESN use, and the results for both *Std* variants with different spectral radius seem to confirm

Table 1. Experimental results for the double pole balancing.

	Cheap Fitness				New Fitness			
Method	Avg. Eval.	Std. Dev.	Genera-lization	% success	Avg. Eval.	Std. Dev.	Genera-lization	% success
Std - 0.95	14960	6291	234	6.8%	16303	11511	209	82.3%
Std - 0.60	16639	17037	225	6.8%	16886	11073	211	87.3%
Rho	23571	10175	241	52.7%	19796	6770	224	91.4%
Std - Opt	19168	21782	232	9.5%	15965	11813	208	86.8%

this. 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 (Section 3.4), 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 an arbitrary value like 0.6 or 0.95, it does not allow to reach the same level of performance as 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. A final advantage about the *Rho* variant, is that it seems to be able to provide controllers that generalize very well, if evolution (using the new fitness) is continued after the first network has passed the 200-tests of the last generalization test: all resulting networks are able to successfully solve more than 500 out of the 625 test cases, with a peak at 555 for one network. Unfortunately, the previously published studies do not report this kind of result, except for one sentence in [4] that mentions that one network successfully solved 525 test cases.

Reservoir topologies. The results obtained with the the double pole problem were found to vary a lot among the different (random) realizations of the connections (i.e. the non-zero weights) in the reservoir, for the same value of the density of connection. 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.

The question is now open: whereas reservoir computing has been proposed as a possible alternative to fine tuning of the weights in Neural Networks, it might be the case that tuning the topology of the reservoir allows to obtain more efficient ESNs. Further work will address this research question, and two main directions can be imagined. The network can be built using different topological classes (e.g. small world, scale free, ...); identifying classes of networks that are efficient for a given type of problem (i.e. such that randomly built networks from this class have a very high probability to solve the problem at hand) would indeed relieve the programmer from the task of optimizing the topology, restricting the

search space to a fraction of the parameter space, where CMA-ES proved to be an efficient tool. It might be the case, however, that for reservoir computing, problem-specific topology tuning is nevertheless required anew for each problem. The main difficulty will then be to design efficient techniques for tuning the topology of large networks, as most existing methods do not really scale up to hundreds of neurons or more. Some hints have been recently given with HyperNEAT [16] on the one hand, and with the different approaches based on Genetic Regulatory Networks, starting with AGE, though other GRN approaches can be envisioned, too (see e.g. [2]).

5 Conclusion

Several recent studies have attempted to couple EAs with ESNs, mostly using supervised learning [15]. A limited number of works have used reinforcement learning to optimize ESNs with EAs, in which the network tasks were time series predictions [18, 11] or robust spatial pattern formation ("flag" problems [3]). To our knowledge, our study is the first one to show the feasibility of the EA-ESN couple for motor control tasks. On addition, previous articles restricted evolutionary optimization to the reservoir weights [15] or more frequently the outgoing weights of the ESN. Here we show that optimizing additional ESN parameters could indeed be efficient.

In a supervised context, the results on a standard time series prediction problem reach the same precision when optimizing the output weights than the original results obtained using quadratic optimization, and further optimizations fail to improve this precision. For reinforcement learning tasks, the good news is that the Evolutionary Learning of ESNs works. Moreover, optimizing more than just the outgoing weights does improve the results. Furthermore, there seems to be a high dependency of the results on the topology of the reservoir, at least for the small sizes experimented with here. Hence, the results presented here do not satisfactorily answer the question of where ESNs stand between the two extremes of neuroevolution today: evolutionary optimization of the weights of a fully recurrent neural network (as proposed in [10]) and carefully crafted developmental systems that evolve the topology of highly efficient NNs for a given task [17, 4]. Further experiments using more reliable test problems, and larger reservoir sizes, are needed to definitely address this issue. Additionally, a side take-home lesson from this paper concerns the usefulness of the double pole balancing problem as a benchmark for evolutionary control in general: the answer is clearly negative for us now (but had been claimed by others before), at last with the kind of fitness used up to now to tackle the problem.

6 Acknowledgments

The simulator of the double pole balancing problem relies on the source code kindly provided by Kenneth O. Stanley.

References

1. A. Auger and N. Hansen. Performance evaluation of an advanced local search evolutionary algorithm. In *Proc. CEC'05*, 2005.
2. W. Banzhaf. Artificial Regulatory Networks and Genetic Programming. In R. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, chapter 4, pages 43–62. Kluwer, 2003.
3. A. Devert, N. Bredeche, and M. Schoenauer. Robust multi-cellular developmental design. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 982–989, New York, NY, USA, 2007. ACM.
4. P. Dürr, C. Mattiussi, and D. Floreano. Neuroevolution with Analog Genetic Encoding. In Th. Runarsson et al., editor, *PPSN IX*, pages 671–680. LNCS 4193, Springer Verlag, 2006.
5. F. J. Gomez and R. Miikkulainen. Solving non-markovian control tasks with neuroevolution. In *IJCAI*, pages 1356–1361, 1999.
6. F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In J.R. Koza et al., editor, *Proc. GP'96*, pages 81–89. MIT Press, 28–31 1996.
7. 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. LNCS 3242, Springer Verlag, 2004.
8. N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolutionstrategies: the covariance matrix adaptation. In *Proc. CEC'96*, pages 312–317. IEEE Press, 1996.
9. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
10. C. Igel. Neuroevolution for reinforcement learning using evolution strategies. In *Proc. CEC'03*, pages 2588–2595. IEEE Press, 2003.
11. K. Ishu, T. van der Zant, V. Becanovic, and P. Ploger. Identification of motion with echo state network. In *Proc. OCEANS '04. MTTS/IEEE TECHNO-OCEAN '04*, volume 3, pages 1205–1210, 2004.
12. H. Jaeger. The Echo State Approach to Analysing and Training Recurrent Neural Networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.
13. H. Jaeger. Tutorial on training recurrent neural networks. Technical report, GMD Report 159, Fraunhofer Institute AIS, 2002.
14. H. Jaeger, H. Haas, and J. C. Principe, editors. *NIPS 2006 Workshop on Echo State Networks and Liquid State Machines*, 2006.
15. J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. Training recurrent networks by evolino. *Neural Comput.*, 19(3):757–779, Mar 2007.
16. K. Stanley. Compositional Pattern Producing Networks: A Novel Abstraction of Development. *Genetic Programming and Evolvable Machines*, 8(2):131–162, 2007.
17. 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.
18. D. Xu, J. Lan, and J. Principe. Direct adaptive control: an echo state network and genetic algorithm approach. In *Proc. IEEE International Joint Conference on Neural Networks IJCNN '05*, volume 3, pages 1483–1486, 31 July–4 Aug. 2005.