An EM based training algorithm for recurrent neural networks

Jan Unkelbach, Sun Yi, and Jürgen Schmidhuber

IDSIA,Galleria 2, 6928 Manno, Switzerland
{jan.unkelbach,yi,juergen}@idsia.ch
http://www.idsia.ch

Abstract. Recurrent neural networks serve as black-box models for nonlinear dynamical systems identification and time series prediction. Training of recurrent networks typically minimizes the quadratic difference of the network output and an observed time series. This implicitely assumes that the dynamics of the underlying system is deterministic, which is not a realistic assumption in many cases. In contrast, state-space models allow for noise in both the internal state transitions and the mapping from internal states to observations. Here, we consider recurrent networks as nonlinear state space models and suggest a training algorithm based on Expectation-Maximization. A nonlinear transfer function for the hidden neurons leads to an intractable inference problem. We investigate the use of a Particle Smoother to approximate the E-step and simultaneously estimate the expectations required in the M-step. The method is demonstrated for a sythetic data set and a time series prediction task arising in radiation therapy where it is the goal to predict the motion of a lung tumor during respiration.

Key words: Recurrent neural networks, Dynamical System identification, EM, Particle Smoother

1 Introduction

Recurrent neural networks (RNNs) represent dynamical systems. With a large enough number of hidden neurons, a given dynamical system can in principle be approximated to arbitrary precision. Hence, recurrent networks have been used as black-box models for dynamical systems, e.g. in the context of time series prediction [1].

When using recurrent neural networks for dynamical system identification, it is typically assumed that the dynamics of the system to be modeled is deterministic. RNNs do not model *process noise*, i.e. uncertainty in the internal state transition between two time steps. Hence, RNNs may not be suitable models for systems exhibiting process noise as a characterizing feature.

In contrast, state space models allow for noise in both the internal state transitions and the mapping from internal states to observations, and therefore represent a richer class of probabilistic models for sequential data. In the special

case of linear-gaussian state space models [6, chapter 13.3], the parameters of the model can be learned using an instantiation of the Expectation-Maximization (EM) algorithm [2]. However, computationally tractable algorithms for exact inference in the E-step are limited to the linear-gaussian case. Nonlinear, non-gaussian models require approximation.

Here, we consider RNNs as nonlinear state space models and suggest a training algorithm based on Expectation-Maximization. We investigate the use of a sequential sampling method, the Particle Smoother [3], to approximate the E-step and simultaneously estimate the expectations required in the M-step. In the M-step, the similarity of linear-gaussian state space models and simple recurrent networks can be exploited.

The proposed method has the advantage that the RNN can model dynamical systems characterized by process noise, and that it represents a *generative* model of the data. This is not the case for conventionally trained RNNs, e.g. using a quadratic objective function minimized through gradient descent.

Stochasticity in recurrent networks has been introduced before in the context of training algorithms based on Extended Kalman Filtering [4]. However, this work did not aim at training a recurrent network as a generative model for a stochastic dynamical system, but at further developing training algorithms for conventional recurrent networks.

Related work includes inference and learning in nonlinear state space models in general. For example, Ghahramani [5] addresses learning in nonlinear state space models where the nonlinearity is represented by radial basis functions. The E-step is approximated via Extended Kalman Filtering. The M-step can be solved in closed form for this model.

Our algorithm is demonstrated for a synthetic data set and a time series prediction task arising in radiation therapy. Lung tumors move during respiration due to expansion and contraction of the lungs. Online adjustment of the radiation beam for tumor tracking during treatment requires the prediction of the tumor motion for about half a second. The variability of the breathing pattern is characterized by process noise rather than measurement noise.

The remainder of this paper is organized as follows: Section 2 reviews Linear Dynamical Systems[1] and Recurrent neural networks as black-box models for dynamical systems. Section 3 describes the EM-based training algorithm for recurrent networks, where subsection 3.1 details the sampling method used in the E-step. Section 4 discusses two applications of the method.

2 Dynamical system identification

Given is a sequence $Y = \{y_t\}_{t=1}^{T}$, where y_t is a vector of observations at time step t. We wish to find a model of the underlying dynamical system that generated the sequence, e.g. for the purpose of time series prediction. Below, we introduce

[1] Linear Dynamical System and Linear-gaussian state space models refer to the same model.

notation, review linear dynamical systems (LDS) and recurrent neural networks, and address the parameter learning problem in these models.

2.1 Linear Dynamical Systems

A stochastic linear dynamical system is described by

$$x_t = Ax_{t-1} + \eta_t^p \tag{1}$$
$$y_t = Bx_t + \eta_t^m \tag{2}$$

where x is an internal state vector, y is an observation vector, A and B are matrices with constant coefficients, η^p is zero mean gaussian process noise with covariance matrix Γ, and η^m is zero mean gaussian measurement noise with covariance matrix Σ.

The stochastic LDS can be formulated as a probabilistic model with observed variables y_t and latent variables x_t. The model is defined via the conditional probabilities

$$P(x_t|x_{t-1}) = \mathcal{N}(x_t|Ax_{t-1}, \Gamma) \tag{3}$$
$$P(y_t|x_t) = \mathcal{N}(y_t|Bx_t, \Sigma) \tag{4}$$

Hence, the hidden variables x_t form a markov chain.

2.2 Learning in linear dynamical systems

In LDS the model parameters $\theta = (A, B, \Gamma, \Sigma)$ can be learned using maximum likelihood through an instantiation of the EM algorithm [6, chapter 13.3.2]. In the E-step, we need to solve the inference problem and calculate the marginal probabilities of the hidden variables conditioned on the observation sequence:

$$P(X|Y;\theta) \tag{5}$$

where $X = \{x_t\}_{t=1}^T$ and $Y = \{y_t\}_{t=1}^T$ denote the entire sequence of hidden states and observations, respectively. For LDS, the inference problem can be solved exactly as $P(X|Y;\theta)$ remains a Gaussian. It's mean and covariance matrix can be calculated with a forward-backward algorithm. Given the posterior distributions over the latent variables, the M-step can also be performed analytically[2].

2.3 Recurrent neural networks

If we model the sequence of observations with a recurrent neural network[3], equations 1 and 2 are replaced by

$$x_t = Af(x_{t-1}) + \eta_t^p \tag{6}$$
$$y_t = Bf(x_t) + \eta_t^m \tag{7}$$

[2] Here, we assume for simplicity that $x_0 = 0$. However, it is straight forward to learn the mean and covariance matrix of a Gaussian distribution over the initial internal state.

[3] This type of network is also referred to as an Elman network.

where x is now interpreted as the vector of net inputs of the hidden neurons, $f = \tanh$ is the transfer function of the hidden neurons so that $f(x)$ is the vector of hidden neuron activations, y is the network output, A is the weight matrix for the recurrent connections, and B is the output weight matrix. Hence, the structure of the RNN model is very similar to LDS. The measurement equation 7 represents a linear output layer. The network has no external inputs in this formulation (except the process noise which can be interpreted as an unknown external influence).

2.4 Conventional learning in recurrent neural networks

When training recurrent networks, the process noise η^p is typically neglected. In addition, the measurement noise η^m is assumed to be uncorrelated and of the same variance in all components of y. In this case, maximizing the likelihood of the data corresponds to minimizing a quadratic cost function:

$$\underset{A,B}{\text{minimize}} \quad \frac{1}{2}\sum_{t=1}^{T}\left[y_t - Bf\left(x_t\right)\right]^2 \tag{8}$$

Minimization of the cost function can be performed via gradient descent. Backpropagation through time (BPTT) and Real time recurrent learning (RTRL) are well-known algorithms to calculate the gradient [7]. Alternative methods include Evolino [8], where the matrix A is determined via evolutionary methods whereas B is determined through linear regression. In Echo-State networks [9], A is chosen to be a fixed sparse random matrix and B is determined via linear regression.

3 EM based learning in recurrent neural networks

Here, we wish to train recurrent networks without neglecting the process noise term. The RNN is considered as a nonlinear state-space model. We exploit the similarities of RNNs and LDS to obtain an EM-based training algorithm.

We wish to maximize the following likelihood function with respect to the model parameters $\theta = (A, B, \Gamma, \Sigma)$:

$$\underset{\theta}{\text{maximize}} \quad L = \int P\left(X, Y|\theta\right)\, dX \tag{9}$$

where $\int dX$ denotes the integration over all latent variables $X = \{x_t\}_{t=1}^{T}$. The complete data Likelihood is given by

$$P\left(X, Y|\theta\right) = \prod_{t=1}^{T} P\left(x_t|x_{t-1}; A, \Gamma\right) P\left(y_t|x_t; B, \Sigma\right) \tag{10}$$

so that the complete data log-Likelihood is given by

$$\ln P\left(X,Y|\theta\right)=\sum_{t=1}^{T}\ln\mathcal{N}\left(x_t|Af(x_{t-1}),\Gamma\right)+\sum_{t=1}^{T}\ln\mathcal{N}\left(y_t|Bf(x_t),\Sigma\right) \quad (11)$$

In the M-step, we maximize the following Q-function with respect to the model parameters θ:

$$\underset{\theta}{\text{maximize}} \quad Q(\theta|\theta^{old})=\int P\left(X|Y;\theta^{old}\right)\ln P\left(X,Y|\theta\right)\,dX \quad (12)$$

The structure of the Q-function is, apart from the non-linear transfer function f, identical to the one for linear dynamical systems [6, chapter 13.3.2]. This similarity can be exploited in the M-step as detailed in section 3.2.

In the E-step, we need to determine the joint probability distribution over the latent variables X conditioned on the observation sequence Y, given the current parameter values θ^{old}:

$$P\left(X|Y;\theta^{old}\right) \quad (13)$$

Since exact inference is intractable, we employ a sampling method, the Particle Smoother, to approximate the distribution 13 as detailed in section 3.1.

3.1 The Particle Smoother for approximate inference

In the E-step of the Expectation-Maximization algorithm, we wish to approximate the probability density of the latent variables X conditioned on the observed sequence Y. However, inspecting the structure of the log-Likelihood function 11 shows that we only need the marginals $P\left(x_t|Y;\theta\right)$ and $P\left(x_t,x_{t-1}|Y;\theta\right)$.

To start, we consider $P\left(x_t|Y_t\right)$, i.e. the probability of x_t given the observations $Y_t=\{y(t')\}_{t'=1}^{t}$ until time step t. This can be approximated via a sequential Monte Carlo method, the particle filter. At each time step t, the distribution is approximated by

$$\hat{P}\left(x_t|Y_t\right)=\sum_{i=1}^{N}w_t^i\delta\left(x_t-x_t^i\right) \quad (14)$$

where x_t^i is the position of particle i at time t, w_t^i is the particle weight, δ denotes the delta-function, and N is the number of particles. To proceed to the next time step, we sample new particles x_{t+1}^j from a mixture of Gaussians:

$$x_{t+1}^j\sim\sum_{i=1}^{N}w_t^i\mathcal{N}\left(x_{t+1}^j|Af(x_t^i),\Gamma\right) \quad (15)$$

The new particles are weighted according to the probability of generating the next obsevation y_{t+1}:

$$w_{t+1}^j=\frac{P\left(y_{t+1}|x_{t+1}^j\right)}{\sum_{k=1}^{N}P\left(y_{t+1}|x_{t+1}^k\right)} \quad (16)$$

An approximation of the probability density $P(x_t|Y)$, now conditioned on the entire observation sequence Y, can be obtained using a forward-backward smoother [3]. Following this method, we first approximate $P(x_t|Y_t)$ for every time step using a particle filter, and in a second step, correct the particle weights via a backwards recursion. The smoothed distribution $P(x_t|Y)$ is then approximated via

$$\hat{P}(x_t|Y) = \sum_{i=1}^{N} v_t^i \delta\left(x_t - x_t^i\right) \tag{17}$$

with modified weights v_t^i. To derive the backward recursion, we consider the following identity:

$$\begin{aligned} P(x_t|Y) &= \int P(x_{t+1}|Y)\, P(x_{t+1}|x_t, Y)\; dx_{t+1} \\ &= \int P(x_t|Y_t)\, \frac{P(x_{t+1}|Y)\, P(x_{t+1}|x_t)}{\int P(x_{t+1}|x_t)\, P(x_t|Y_t)\; dx_t}\, dx_{t+1} \end{aligned} \tag{18}$$

By inserting equations 14 and 17 into 18, we obtain the backward recursion for the corrected weights v_t^i:

$$v_t^i = w_t^i \left[\sum_{j=1}^{N} v_{t+1}^j \frac{P_t^{ji}}{\sum_{k=1}^{N} w_t^k P_t^{jk}} \right] \tag{19}$$

where $P_t^{ji} = P(x_{t+i}^j | x_t^i)$ is the transition probability from particle i at time t to particle j at time $t+1$. The backward recursion is initialized as $v_T^i = w_T^i$. After obtaining particle weights w_t^i in the forward pass and weights v_t^i in the backward pass, equation 18 gives also rise to an approximation of the joint probability for x_t and x_{t+1}:

$$\hat{P}(x_t, x_{t+1}|Y) = \sum_{i=1}^{N} \sum_{j=1}^{N} \left[\frac{w_t^i v_{t+1}^j P_t^{ji}}{\sum_{k=1}^{N} w_t^k P_t^{jk}} \delta(x_t - x_t^i) \delta(x_{t+1} - x_{t+1}^j) \right] \tag{20}$$

3.2 The M-step

In the M-step, we need to maximize the Q-function 12 with respect to the model parameters $\theta = (A, B, \Gamma, \Sigma)$. Here, we can make use of the fact that the structure of the dynamic equation describing the RNN is similar to those for LDS. This leads to similar update equations for the parameters. Exemplarily, we consider the update equation for the recurrent weight matrix A which we obtain by setting the Q-function derivative with respect to A to zero:

$$A^{new} = \left[\sum_{t=1}^{T} \mathbb{E}\left[x_t f(x_{t-1})^\top \right] \right] \left[\sum_{t=1}^{T} \mathbb{E}\left[f(x_{t-1}) f(x_{t-1})^\top \right] \right]^{-1} \tag{21}$$

where $\mathbb{E}$ denotes the expectation with respect to the distribution $P(X|Y;\theta^{old})$. The expectations in equation 21 are approximated through the particle smoother as

$$\mathbb{E}\left[f(x_{t-1})f(x_{t-1})^\top\right] = \sum_{i=1}^{N} v_{t-1}^i f(x_{t-1}^i) f(x_{t-1}^i)^\top \tag{22}$$

$$\mathbb{E}\left[x_t f(x_{t-1})^\top\right] = \sum_{i=1}^{N}\sum_{j=1}^{N} \frac{w_{t-1}^i v_t^j P_{t-1}^{ji}}{\sum_{k=1}^{N} w_t^k P_{t-1}^{jk}} x_t^i f(x_{t-1}^i)^\top \tag{23}$$

The update equations for the parameters B, Γ and Σ are obtained in analogy to the update equations for linear-gaussian models as described in [6, chapter 13.3.2]. They are obmitted here due to space limitations.

4 Applications

The method is demonstrated for a synthetic data set and a real data set describing irregular breathing motion of a lung cancer patient.

4.1 Synthetic data

We consider synthetic data generated by a recurrent network with two hidden neurons with parameters

$$A = \begin{pmatrix} 1 & 1 \\ -0.05 & 0.98 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad \Gamma = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.001 \end{pmatrix} \quad \Sigma = (0.01)$$

These parameters were chosen so that the system outputs an oscillation which is perturbed by both process and measurement noise. Without noise, the system generates an (almost harmonic) oscillating output signal with a period of 28 time steps. Figure 1 shows a sample sequence generated by the system with noise. A recurrent network with 5 hidden neurons[4] was trained on a data set containing 1000 time steps. Figure 1 shows parts of a test data set, together with predictions of the trained RNN (blue circles). Every 20 time steps, the network prediction for the next 10 time steps is shown. For prediction, a particle filter is used to estimate the mean of the posterior distribution over the hidden state. Then the deterministic dynamics of the network (without noise) is used for prediction, starting from the estimated hidden state[5]. In order to compare the predictions of the RNN, figure 1 also shows the predictions based on the true dynamical system (green dots). In this example, the prediction performance of the RNN is similar to the best possible prediction where the true generative process of the data is known[6].

[4] Similar results are obtained with 2 or more hidden neurons.

[5] A prediction method where all particles were propagated forward in time and averaged at each time step, yields almost identical results.

[6] The mean squared errors of the trained network are 0.145/0.259/0.327 for the 1/5/10-step prediction. The corresponding values for the true system are 0.142/0.240/0.307. The standard deviation of the test data is 0.380.

Figure 2 shows the internal, noise free dynamics of the network. The network output is shown as the thick red line, whereas the blue thin lines show the dynamics of the internal state. The network was able to learn the underlying harmonic oscillation of the dynamical system and estimate the amount of process and measurement noise.

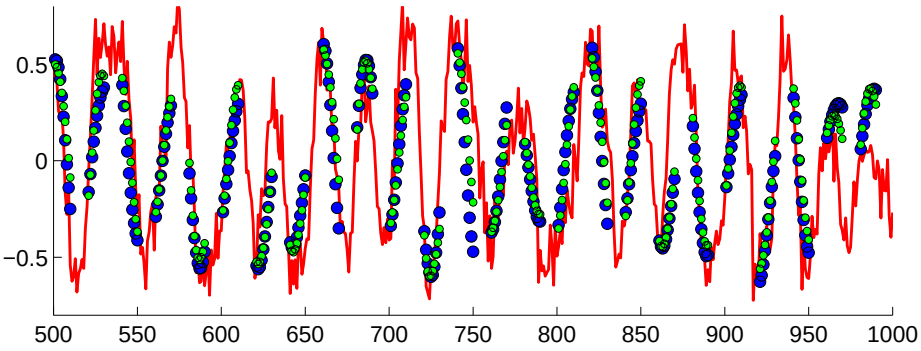


Fig. 1. Synthetic time series (red line). Every 20 time steps, the network predictions for the next 10 time steps are shown (blue circles), together with the predictions obtained using the true dynamical system (green dots).

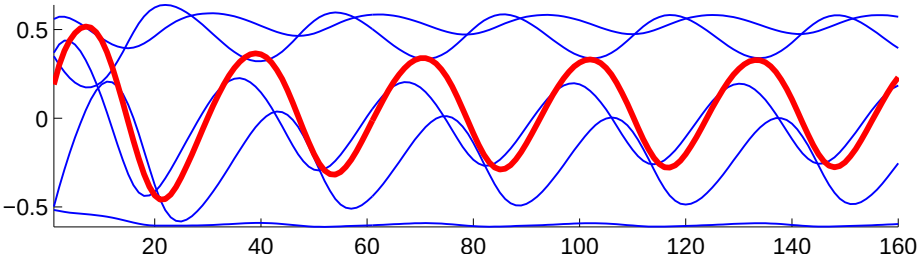


Fig. 2. Deterministic dynamics of the network trained on the noisy data set. Red thick line: network output; Blue thin lines: internal states

4.2 Breathing motion data

Figure 3 (red line) shows a surrogate for breathing motion for a lung cancer patient[7]. The signal is roughly periodic, but has substantial variations in amplitude and period. In radiation therapy practice, the goal is to predict the breathing pattern for about 500 milliseconds (5 time steps), where the length of one breathing cycle is around 3 seconds. It is also of interest to have a generative model of the breathing signal in order to simulate the effect of breathing on the accuracy of a radiation treatment.

A recurrent network with 10 hidden neurons was trained on a sequence of 1000 time steps. Figure 4 shows the intrinsic dynamics of the trained network, i.e.

[7] The expansion of the abdomen was measured as a function of time.

the dynamics without noise. Figure 3 shows samples of the network prediction on the test data. Every 20 time steps, the network predictions for the next 10 time steps are shown. Like in the synthetic data example above, the training algorithm is able to find solutions that model the oscillatory behaviour of the system. However, it failed to model the dynamics very precisely. Consequently, in order to explain the deviation of data and network output, the estimated amount of measurement noise Σ is too large, and the trained network does not represent an adequate generative model.

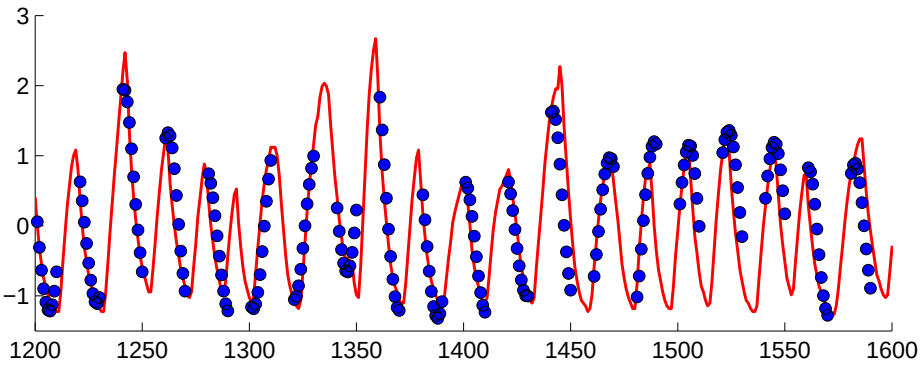


Fig. 3. Breathing signal time series (red line). Every 20 time steps, the next 10 time steps predicted by a trained network are shown (blue circles).

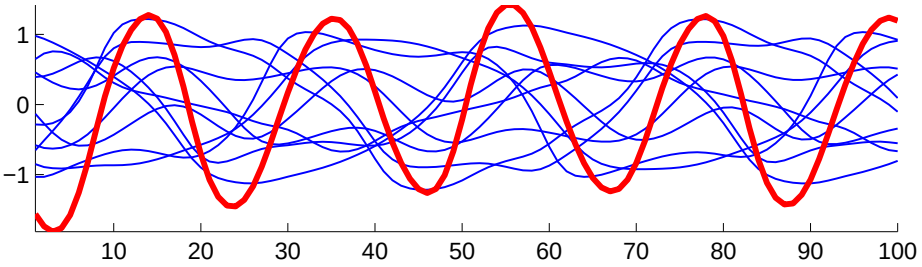


Fig. 4. Deterministic, noise free dynamics of a trained network. Red thick line: network output; Blue thin lines: internal states

4.3 Remarks

Initialization of parameters: The weight matrix A was initialized to a diagonal matrix plus random values with a standard deviation of 0.1. The components of B were initialized to random values with mean one and standard deviation 0.1. These values led to solutions qualitatively similar to those shown above.

Local minima: Dynamical system identification tends to be a difficult task. For the breathing motion data set, the network did not learn details of the dynamics. However, the same problem applies to linear-gaussian state space models and RNNs trained with gradient descent.

Number of particles: The results presented above were generated with $N = 100$ particles. However, similar results were obtained for only 10 particles – although particle numbers that low are not expected to provide an adequate characterization of the posterior distribution at a given time step. This aspect will be investigated in future work.

5 Conclusion

We address the problem of training recurrent neural networks as black box models for stochastic nonlinear dynamical systems. In conventional training algorithms based on a quadratic objective function, it is assumed that the underlying dynamics to be modelled is deterministic, i.e. process noise is neglected. In this paper, we generalize RNNs to model stochastic dynamical systems characterized by process noise. We suggest a training algorithm based on Expectation-Maximization, exploiting the similarities between RNNs and linear dynamical systems. We apply a particle smoother for approximate inference in the E-step and simultaneously estimate expectations required in the M-step. The algorithm is sucessfully demonstrated for one synthetic and one realistic data set. Further characterization of the performance of the training algorithm, the influence of the number of particles, and comparison to standard training methods for RNNs are subject to current studies.

References

1. D. P. Mandic and J. A. Chambers. *Recurrent neural networks for prediction.* John Wiley & Sons, Inc., 2001.
2. G. J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions.* John Wiley & Sons, Inc., 1997.
3. Mike Klaas, Mark Briers, Arnaud Doucet, and Simon Maskell. Fast particle smoothing: If i had a million particles. In *In International Conference on Machine Learning (ICML*, pages 25–29, 2006.
4. Ronald J. Williams. Training recurrent networks using the extended kalman filter. In *In Proceedings International Joint Conference on Neural Networks*, pages 241–246. Online]. Available: citeseer.nj.nec.com/williams92training.html, 1992.
5. Zoubin Ghahramani and Sam T. Roweis. Learning nonlinear dynamical systems using an EM algorithm. In *Advances in Neural Information Processing Systems 11*, pages 599–605. MIT Press, 1999.
6. C. M. Bishop. *Pattern recognition and Machine learning.* Springer, 2006.
7. R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Back-propagation: Theory, Architectures and Applications.* Hillsdale, NJ: Erlbaum, 1994.
8. J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. Training recurrent networks by evolino. *Neural Computation*, 19(3):757–779, 2007.
9. 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.