BAYESIAN LEARNING OF ECHO STATE NETWORKS WITH TUNABLE FILTERS AND DELAY&SUM READOUTS

Christoph Zechner, Dmitriy Shutin

Graz University of Technology
Signal Processing and Speech Communication Laboratory
Inffeldgasse 12, A-8010 Graz, Austria
christoph.zechner@student.tugraz.at, dshutin@tugraz.at

ABSTRACT

In this paper we investigate the problem of learning Echo State Networks (ESN) with adaptable filter neurons and delay&sum readouts. A brute-force solution to this learning problem is often impractical due to nonlinearity and high dimensionality of the resulting optimization problem. In this work we propose an approximate solution to the ESN learning by appealing to the variational Bayesian EM-type of estimation algorithm. We show that such approach allows to significantly reduce the dimensionality of the resulting objective functions. Furthermore, it allows to implement ESN learning and adapt filter neurons and delays jointly within the variational framework. Simulations are performed for learning randomly generated target ESNs, as well as other synthetic nonlinear dynamic systems. The results demonstrate that the proposed learning algorithm can improve ESN learning for a wide class of problems.

Index Terms— Echo State Network, Variational Bayesian Inference

1. INTRODUCTION

Echo state networks (ESN) and reservoir computing represent a powerful mechanism for training recurrent neural network with an advantage of very simple learning [1, 2]. They are used to approximate nonlinear dynamical systems, and represent a powerful improvement over classical non-temporal blackbox nonlinear modeling. ESNs have found their applications in many areas of signal processing, such as speech recognition and audio processing, system modeling and prediction, and filtering [3–5], to name just a few. ESN allows to learn/reconstruct a nonlinear relationship between an input \( u[n] \) and an output \( y[n] \) of a nonlinear dynamical system

\[
\begin{align*}
    x_l[n] & = f(c_{ul} u[n] + c_{x_l}^T x[n-1]), \quad l = 0, \ldots, L-1; \\
\end{align*}
\]

where \( f(\cdot) \) is a neuron activation function, e.g., a hyperbolic tangent, \( u[n] \) are the network inputs, and \( x[n] \) is a vector of neuron states. Coefficients \( c_{ul} = [c_{ul,0}, \ldots, c_{ul,L-1}]^T \) and \( c_{x_l} = [c_{x_l,0}, \ldots, c_{x_l,L-1}]^T \) are the weights of the internal network connections, and input signal weights, respectively. They are generated and fixed at the network initialization stage. When properly designed, the sequence of network states \( x[n] \) forms echo functions – a temporal basis used to reconstruct the output signal \( y[n] \). The dynamics of the observed data is thus incorporated in the echoes generated by the reservoir. Typically, a superposition of generated echoes and network inputs (eq. (2)) can be found such as to closely follow the desired response \( y[n] \). Since \( W \) enters (2) linearly, the optimal solution can be found by solving a system of linear equations, which significantly simplifies ESN learning.

However, it has been shown that a single ESN cannot learn multiple attractors at the same time [6]. The learning power of ESNs can be significantly boosted by introducing filter neurons and delay&sum readouts in the ESN structure [6, 7]. Equipping neurons with additional filters will result in neurons that are “specialized” to more relevant frequency bands; introducing delays makes it possible to shift the reservoir signals in time and provides a computationally cheap method to vastly improve the memory capacity of the network [8]. The parameters of such filter neurons and corresponding readout delays can be chosen randomly during the network initialization, or heuristically chosen through trial and error [6]. Alternatively, these parameters can be estimated.

Estimating delays, as well as filter neuron parameters turns a simple linear learning problem into a nonlinear one. Nonetheless the desired output is still formed as a linear combination of inputs and echo states that now depends nonlinearly on the parameters of the extended ESN. Thus, we can make use of the algorithms developed for estimating parameters of superimposed signals (e.g., [9, 10]) for learning ESNs and estimating internal network parameters. We restrict ourselves to a class of EM-type of algorithms, known as Space-Alternating Generalized EM [9, 11] that, by means of latent variables, can significantly simplify the parameter estimation even in nonlinear cases. The SAGE algorithm allows to optimize only a subset of parameters at each iteration, while keeping the other parameters fixed. In our setting this will allow to optimize parameters of one neuron at a time. By casting the SAGE algorithm in the variational Bayesian inference framework [12, 13], we can simultaneously estimate parameters of extended neurons, as well as learn the ESN. The obtained solution to the learning problem can be seen as an iterative alternative to the standard Wiener-Hopf solution. We will show that by using the EM-type of estimation, the final update expressions for \( W \) require inverting significantly smaller matrices. Second, the ensuing estimation of delays and filter parameter is significantly simplified since the dimensionality of the objective function is reduced. As the consequence, both \( W \) and neuron parameters can be jointly estimated.
2. EXTENDED ESN STRUCTURE

The learning ability of ESNs can be boosted by extending the neurons with bandpass filters, and by introducing delay&sum readouts [7], such that each of the L neurons produces its state as the output of a Pth order bandpass FIR filter, and delayed network outputs are used to reconstruct \( y[n] \). We suggest to construct a tunable FIR filter for each neuron by modulating a fixed low-pass impulse response \( e \) with the frequency \( \theta_l \); the impulse response of the resulting filter is given as \( M(\theta_l) e \), where \( M(\theta_l) = \text{diag}[1, \cos(\pi \theta_l), \ldots, \cos(\pi \theta_l (P - 1))] \) is the modulation matrix. The signal graph of the resulting filter neuron is shown in Fig. 1. The state of a single neuron is then a function of the modulation frequency \( \theta_l \), and is computed as

\[
x_l[n; \theta_l] = c_0[n]M(\theta_l)e_c[n+1]
\]

where \( c_0[n] = [c_0, \ldots, c_0 (n - P + 1)]^T \) is a vector of delayed nonlinearity outputs (Fig. 1). Let us now assume that the network output \( y[n] \) in (2) is a scalar, i.e., \( y[n] \equiv y[n] \). For learning the ESN, we collect \( N \) samples of input-output data. Let us now define \( u_l = [u_l[0], \ldots, u_l[N - 1]]^T \) as the input signal for the \( l \)-th neuron, \( y = [y[0], \ldots, y[N - 1]]^T, \) and \( x_l[\tau; \theta_l] = [x_l[\tau + \tau; \theta_l], \ldots, x_l[\tau - 1; \theta_l]]^T \) as the neuron echo state. Notice that the latter depends on the delay \( \tau_l \). The overall delay&sum network response is then constructed simply as

\[
y = \sum_{i=0}^{L-1} (w_{ui}x_l[\tau; \theta_l] + w_{ui}u_l)
\]

Define \( \Omega_l = \{\tau_l, \theta_l, w_{ui}, w_{ui}\} \) as a set of parameters associated with the \( l \)-th neuron. Clearly, obtaining joint estimates of all network parameters \( \Omega = \{\Omega_0, \ldots, \Omega_{L-1}\} \) is a nontrivial task. However, from (4) we notice that \( y \) can be interpreted as a linear combination of nonlinearly parametrized functions \( x_l[\tau; \theta_l] \) and \( u_l \). Thus we can exploit algorithms developed for estimating parameters of superimposed signals in order to find the solution to our estimation problem.

3. PARAMETER ESTIMATION

Let us first cast (4) in a probabilistic setting. For that we will assume that \( y \) is perturbed by \( \xi \), which can be interpreted as a random learning error. We will assume \( \xi \sim \mathcal{N}(0, \Sigma) \), i.e., it is a zero-mean Gaussian random variable with the covariance matrix \( \Sigma \). We will now formulate the estimation algorithm within the variational inference framework. When estimating parameters using the SAGE algorithm [9,11], the concept of complete data in the EM algorithm is replaced by that of admissible hidden data. Hidden data allows to optimize only a subset of parameters at each iteration, while monotonically increasing the parameter likelihood. The concept of admissible hidden data can also be exploited within the variational framework.

Consider a new variable \( h_l = \{x_l[\tau_l; \theta_l]u_l + u_l w_{ui}\} + \xi_l \), which can be interpreted as the (unobserved) contribution of the \( l \)-th neuron to the network response \( y \). The noise \( \xi_l \) is obtained by arbitrarily decomposing the total perturbation \( \xi \), such that \( \Sigma_l = E[\xi_l \xi_l^T] = \delta_l \Sigma, \) and \( 0 \leq \delta_l \leq 1 \). Under the Gaussian noise assumption, \( h_l|\Omega_l \sim \mathcal{N}(x_l[\tau_l; \theta_l]u_l + u_l w_{ui}, \Sigma_l) \). The role of the \( h_l \) is to enforce conditional independence between parameters of the \( l \)-th neuron and the rest of the neurons in the network, and thus simplify the estimation algorithm. Now, the joint probability density function \( p(\Omega, h_l, y) \) can be factorized as

\[
p(\Omega, h_l, y) = p(y|h_l, \Omega_{k\neq l}) \prod_{k} p(\Omega_k) \times p(h_l|\Omega_l)p(h_l), \quad (5)
\]

where \( \Omega_l, \Omega_k, k \neq l \), are conditionally independent given \( h_l \). As the consequence, the estimation algorithm can be formulated as sequence of \( L \) “smaller” optimizations, which can be run for each neuron independently, assuming the parameters of the other neurons are fixed. We will assume that \( y|h_l, \Omega_{k\neq l} \sim \mathcal{N}(h_l + \sum_{k} (x_k[\tau_k; \theta_k]u_k w_{uk} + u_k w_{uk}), \Delta_l) \), where \( \Delta_l = (1 - \delta_l)\Sigma \) is the part of the total disturbance covariance \( \Sigma \) that is not associated with the \( l \)-th neuron, i.e., \( \Delta_l + \Sigma_l = \Sigma_l \).

Variables \( h_l \) can be shown to be admissible for estimating \( \Omega_l \). Just like in the EM framework, statistics of \( h_l \) can be estimated using the variational calculus, similar to the variational Bayesian EM algorithm [13]. Variational inference approximates (5) with a “simpler” proxy distribution

\[
q(\Omega, h_l) = q(h_l) \prod_{i=1}^{L} q(w_{ui}, w_{ui})q(\theta_l)q(\tau_l), \quad (6)
\]

The choice of factors in (6) is arbitrary, as long as they are valid density functions. In our work we choose \( q(h_l) = \delta(h_l - h_l) \), \( q(\tau_l) = \delta(\tau_l - \tau_l) \), \( q(\theta_l) = \delta(\theta_l - \theta_l) \), \( q(w_{ui}, w_{ui}) = \delta(w_{ui} - w_{ui}, w_{ui} - w_{ui}) \) where \( \delta(\cdot) \) is a multivariate Dirac distribution. By choosing Dirac distributions we restrict ourselves to the point estimates of these variables.2 Variational inference aims to find the parameters \( h_l, \tau_l, \theta_l, w_{ui}, w_{ui} \), such as to minimize the Kullback-Leibler divergence \( D_{KL}[q(h_l)|p(h_l)] \) between (5) and (6). Due to the sparse limitations we skip the detailed description of the variational inference algorithm, which can be found elsewhere (see e.g., [13]), and provide the final estimation expressions for our specific problem. In what follows, we consider estimating variational parameters related to \( l \)-th neuron only; the corresponding expressions provide updates at the iteration \( j + 1 \), assuming that variational parameters at the iteration \( j \) are known. The corresponding expressions are given by:

\[
h_l^{[j+1]} = y - \sum_{k \neq l} (x_k[\tau_k; \theta_k]u_k + u_k w_{uk}), \quad (7)
\]

\[
\begin{bmatrix}
w_{ui}^{[j+1]} \\
w_{ui}^{[j+1]}^{[j+1]}
\end{bmatrix} = \Phi_l \begin{bmatrix}
x_l[\tau_l; \theta_l]u_l \end{bmatrix} \Sigma_l^{-1} h_l^{[j+1]}, \quad (8)
\]

1Later in the text we will consider solutions when \( \beta_l \rightarrow 0 \). This corresponds to assuming that \( h_l \sim y - \sum_{k \neq l} (x_k[\tau_k; \theta_k]u_k + u_k w_{uk}) \), which was shown to accelerate the convergence rate of the SAGE algorithm [9].

2Although this choice leads to the classical EM/SAGE algorithm, variational approach is quite generic and allows to use more complex densities. We, however, leave this case outside the scope of this paper.
\[ \Phi_l = \left( \frac{\|x_l(\hat{\tau}[j]_l, \hat{\theta}[j]_l)\|^2}{\Sigma_l} x_l(\hat{\tau}[j]_l, \hat{\theta}[j]_l)^T \Sigma^{-1}_l u_l \right) u_l^T \Sigma^{-1}_l x_l(\hat{\tau}[j]_l, \hat{\theta}[j]_l), \tag{9} \]

where \(\|x\|_2^2 = x^T \Sigma^{-1} x\) denotes a weighted vector norm. Expression (7) is obtained as \(\beta_l \rightarrow 1\). Notice also that (10) has to be solved numerically, since \(\theta_l \) and \(\tau_l \) enter \(x_l(\tau_l, \theta_l)\) in a nonlinear way. However, this is only a two-dimensional optimization performed \(L\) times, as compared to the \(2L\)-dimensional problem for the original extended ESN.

3.1. Implementation Aspects

Basically, our learning algorithm covers the joint estimation of \(W\), \(\theta = [\theta_0, \ldots, \theta_L-1]\) and \(\tau = [\tau_0, \ldots, \tau_{L-1}]\), which is summarized in Algorithm 1.

Algorithm 1 Learning \(W\) and \((\theta, \tau)\)

Require: \(u_0, \ldots, u_{L-1}, y\)
while continue SAGE iterations do
for all \(l\)-th neuron do
\(\{x_l, o_l[n]\} \leftarrow \) Simulate ESN \{recompute neuron outputs\}
\(h_l \leftarrow \) eq. (7)
\((\hat{w}_{\tau[l]}, \hat{w}_{\tau[l] + 1}) \leftarrow \) eq. (8), (9)
\((\hat{\tau}_{l}[j+1], \hat{\theta}_{l}[j+1]) \leftarrow \) eq. (10)
end for
end while

Note that the echo states \(x_l(\tau_l, \theta_l)\) depend on the optimization parameters \(\theta_l\) and \(\tau_l\). Furthermore, remember that \(x_l[n; \theta_l]\) from eq. (3) shows a recurrent dependency on the states of other neurons, which would require recomputing all the echo states anew for each values of the recurrent parameter. As this would be quite computationally expensive, we instead compute \(x_l(\tau_l, \theta_l)\) using the current neuron output \(o_l[n]\) (eq. (3)) by ignoring the recursive dependency on other neurons. This heuristic approximation reduces the computational effort for optimizing one neuron by \(O(L \cdot N)\) for each possible value of \(\tau_l, \theta_l\). Although the initial algorithm is optimal for the joint estimation of \((\tau, \theta)\) and \(W\), with this approximation we found that better learning results are achieved by first performing the update iteration over \((\tau, \theta)\) with \(W\) fixed, followed by an update iteration of \(W\) with fixed \((\tau, \theta)\) and so on.

4. SIMULATIONS

In this section we will analyze the learning performance of the ESN with extended neuron structures using the proposed algorithms. For comparison three algorithms are used:
- **Standard ESN**: An ESN without filter neurons and delay & sum readouts. The output weights are estimated using ridge regression with the regularization constant \(\alpha = 1e-5\).
- **Algorithm A**: Only the output weights \(W\) are estimated using equations (8), (9) with \(\theta_l\) and \(\tau_l\) being randomly generated.
- **Algorithm B**: \((\tau, \theta)\) and \(W\) are estimated consecutively using the proposed algorithm (eq. (10), and (8), (9)).

Remarks: Eq. (10) was numerically solved using a grid search. As an input signal for all our simulations we used i.i.d. samples drawn from a zero-mean Gaussian distributions with the variance 0.36, and \(u_l[n] = u_l, l = 0, \ldots, L - 1\).

4.1. Neuron parameter learning

First of all, we will evaluate the adaptation properties of Algorithm A and B. To do so, it is useful to analyze the training error as a function of the number of update iterations.

![Fig. 2: The training error as a function of the number of update iterations. All errors are given in dB related to the output variance \(\sigma_y^2\).](image)

Figure 2 illustrates the learning process of Algorithm A and B using an arbitrary nonlinear fading memory target system. In both cases 40 neurons were adapted during six SAGE iterations\(^3\). We can see that Algorithm B can drastically reduce the training error compared to Algorithm A. The Standard ESN algorithm seems to completely fail for this task, as the resulting training error is in the range of the output variance \(\sigma_y^2\). Note that as we made an approximation to find the neuron parameters of the ESN, the training error can also increase during the learning process. In our simulations it seemed that more frequent ascends of the error indicate overfitting and lead to a worse performance on the test data.

4.2. System Identification

Figure 3 compares the test error between Algorithm A and B for identifying two different target systems. In figure 3(a) an ESN (250 neurons) with \((\tau, \theta)\) randomly generated was used. Figure 3(b) shows the results for learning the following target system:

\[ g[n] = median\left(\{g[n], g[n-1], \ldots, g[n-19]\}\right) \tag{11} \]

with \(g[n]\) being the output of an FIR filter with the impulse response \(v[n]\); the latter is a Gaussian filter of length \(N = 61\) with the width parameter \(\sigma_z^2 = 9\). We can see from Fig. 3 that Algorithm B outperforms Algorithm A in both cases. Even when estimating the output weights \(W\) with randomly chosen neuron parameters (Algorithm A) already leads to an acceptable performance, the results can be further improved by estimating \((\tau, \theta)\).

In general the improvement of Algorithm B compared to Algorithm A strongly depends on the properties of the target system. There are also systems for which the learning performance cannot be significantly improved by adapting \((\tau, \theta)\) so that the computational effort is large compared to the improvement. It seems that tunable

\(^3\)For Algorithm B this means performing three SAGE iterations for \((\tau, \theta)\) and three SAGE iterations for \(W\) consecutively
neurons with delay&sum readouts can model the signals with rich time-varying frequency content better, as compared to the standard ESN or ESNs with frozen filter parameters. This can be explained by the fact that neurons with tunable filters can adapt better to the specific frequency bands. In contrast, ESNs with fixed filters and delays would require more neurons in order to capture the required frequency information.

5. CONCLUSION

An algorithm that combines variational Bayesian inference with the SAGE algorithm for efficient estimation of Echo State Network parameters was presented and evaluated using extended neuron models (filter neurons, delay&sum readouts). Using the concept of admissible hidden data, presented within the SAGE framework, it became possible to significantly reduce the dimensionality of the objective function, and thus the complexity of the estimation algorithm. The presented approach provides an efficient iterative scheme for learning ESNs with filter neurons and delay&sum readouts, as well as standard ESNs.

6. REFERENCES