Learning Input and Recurrent Weight Matrices in Echo State Networks

Hamid Palangi
University of British Columbia
Vancouver, BC, Canada
hamidp@ece.ubc.ca

Li Deng
Microsoft Research
Redmond, WA, USA
deng@microsoft.com

Rabab K Ward
University of British Columbia
Vancouver, BC, Canada
rababw@ece.ubc.ca

Abstract

The traditional echo state network (ESN) is a special type of a temporally deep model, the recurrent network (RNN), which carefully designs the recurrent matrix and fixes both the recurrent and input matrices in the RNN. The ESN also adopts the linear output (or readout) units to simplify the leanring of the only output matrix in the RNN. In this paper, we devise a special technique that takes advantage of the linearity in the output units in the ESN to learn the input and recurrent matrices, not carried on earlier ESNs due to the well-known difficulty of their learning. Compared with the technique of BackProp Through Time (BPTT) in learning the general RNNs, our proposed technique makes use of the linearity in the output units to provide constraints among various matrices in the RNN, enabling the computation of the gradients as the learning signal in an analytical form instead of by recursion as in the BPTT. Experimental results on phone state classification show that learning either or both the input and recurrent matrices in the ESN is superior to the traditional ESN without learning them, especially when longer time steps are used in analytically computing the gradients.

1 Introduction

The recurrent Neural Network (RNN) belongs to a general type of deep neural networks that model time sequences and dynamical systems [1, 2, 3, 4, 5, 6]. The echo state network (ESN) further belongs to the general class of the RNN [7, 8, 9, 10]. Four special properties of the ESN make it distinct from other types of RNNs. First, both the recurrent and input matrices in the ESN are fixed and not learned. This is due largely to the well-known difficulty of learning RNN [11, 12]. Second, the hidden neurons in the ESN are typically much larger than that of regular RNNs, enabled by not training most of the very large number of difficult network parameters thus escaping the main challenges of training RNNs described in [11, 12] and also avoiding potential overfitting problems. Third, the output units, also called readout units, in the ESN are linear, instead of typically nonlinear in the regular RNNs. Given the very large number of hidden neurons of the ESN, the output or readout weight matrix is very large as well. The use of linear output units allows the output weight matrix to be learned very efficiently and with very simple regularization mechanism based on ridge regression. Fourth, the learning of the ESN parameters (i.e. output matrix) is much simpler than that for the regular RNNs. The former is linear learning with convex optimization, and the latter, based typically on BackProp Through Time (BPTT), is highly nonlinear and non-convex. As a result, learning the ESN parameters can be effectively carried out via batch training, greatly facilitation parallel implementation, while learning the general RNN parameters typically requires stochastic gradient descent, more difficult for parallelization.

The simplicity in the ESN learning comes with the cost of not learning important part of the parameters including both the input and the recurrent weight matrices and of the use of linear output units. While special design of the recurrent matrices (see [13]) and the use of a large number of hidden neurons have helped reduce the weakness of fixing rather than learning parameters, it is desirable to make these parameters adaptive to data if the required learning is still simpler and more parallelizable than the common learning method of BPTT applied to regular RNNs.

This paper presents just this type of learning for the input and recurrent matrices of the ESN. We propose a technique that makes full use of the linearity in the output units in constructing constraints among all three input, output, and recurrent matrices in the ESN. The constraints enable the computation of the gradients as the learning signal in an analytical form, and thus making the gradient estimate more accurate than that computed by recursion as in BPTT. Our preliminary experimental results on phone classificaiton are highly positive. It is demonstrated that learning either or both the input and recurrent matrices in the ESN give better phone classification accuracy than the traditional ESN without learning them. Further, when longer time steps are used in analytically computing the gradients, the better classification results are obtained.

2 General Recurrent Networks and Specific Echo State Networks

A general RNN has temporal connections as well as input-to-hidden, hidden-to-output connections. These connections are mathematically represented by the recurrent weight matrix $\mathbf{W}_{rec}$, the input weight matrix $\mathbf{W}$, and the output weight matrix $\mathbf{U}$, respectively. The RNN architecture, in terms of the signal flow, is illustrated in Fig. 1, which also includes input-to-output and output-to-hidden (feedback) connections, with the latter denoted by $\mathbf{W}_{fd}$. The sequential sections of Fig. 1(a), 1(b), 1(c), $\ldots$, denote the RNN unfolding in time. Note all the weight matrices are constained to be the same (i.e. they are tied) in any discrete point in time.

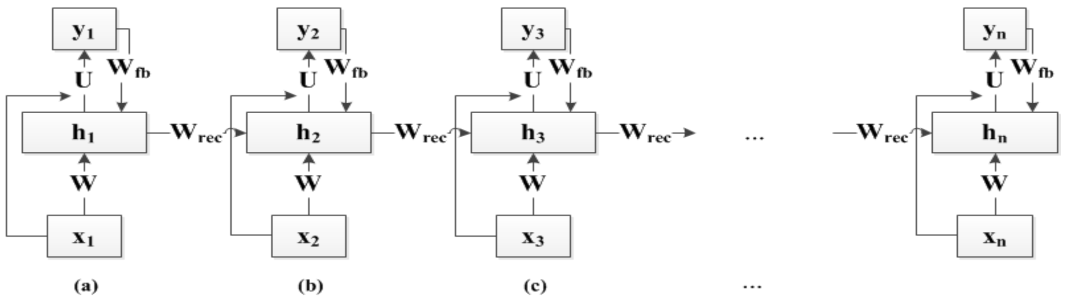

Figure 1: Illustration of a general RNN unfolding over time

In Figure 1, $\mathbf{x}_i$, $\mathbf{h}_i$ and $\mathbf{y}_i$ represent the input, hidden and output vectors at discreate time $t = i$. Again, the connections between the input ($\mathbf{x}_i$) and hidden ($\mathbf{h}_i$) layers, the hidden and output ($\mathbf{y}_i$) layers, and the output and hidden layers are represented by $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{W}_{fb}$ respectively. The temporal connection between $\mathbf{h}_i$ and $\mathbf{h}_{i+1}$ is represented by the matrix $\mathbf{W}_{rec}$. Note that the direct connections from the input to output layers are represented as part of the matrix $\mathbf{U}$; i.e., it is equivalent to concatenating the input layer with the hidden layer.

There are two standard methods to train RNNs, namely, the BPTT method and the method based on Extended Kalman Filtering (EKF) [13]. BPTT is a first order method, which is an extension of error backpropagation for feed-forward networks by treating each time step as a new hidden layer but tying all the weight matrices across time. Generally, BPTT has slow convergence. And the difficulty in capturing long-term memory due to vanishing gradient and exploding gradient problems has been well known for many years [11]. It is often non-trivial to obtain good results with BPTT; see the tremendous amount of engeering required to make BPTT work [2, 14, 15]. The EKF-based method, on the other hand, has fast convergence properties and belongs to a second-order method. However, the computation required of the EKF method is very expensive and its implementation is non-trivial [13], especially for large-scale problems.

One prominent approach in the literature that has been proposed to overcome the difficulty in training RNNs is the ESN [9],[13]. As explained in the introduction section, an ESN is a special type of RNNs where recurrent weights ($\mathbf{W}_{rec}$) and input to hidden layer weights ($\mathbf{W}$) are not trained and only hidden layer to output and input to output weights ($\mathbf{U}$) are trained. The recurrent connections in $\mathbf{W}_{rec}$ are sparse and their value are carefully fixed in a way that the echo-state property is preserved. ESNs can be trained very fast because the only connections that are trained are the output connections. With good initialization, ESNs have been shown to have yielded good performance for one dimensional sequences but for high dimensional data such as speech the studies have been relatively limited; but see [10].

Since the output units have linear activation function in the ESN and assuming the hidden units have sigmoid activation function, the formulation of the ESN as a special type of the RNN shown in Fig. 1 can be succeinctly described by

$$\mathbf{h}_{i+1} = \sigma(\mathbf{W}^T\mathbf{x}_{i+1} + \mathbf{W}_{rec}\mathbf{h}_i + \mathbf{W}_{fb}\mathbf{y}_i) \tag{1}$$

$$\mathbf{y}_{i+1} = \mathbf{U}^T\mathbf{h}_{i+1} \tag{2}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$. As discussed earlier, the ESN has its special designed recurrent matrix, which, in most existing versions [13, 9, 10] , is endowed with the echo state property. The echo state property implies that the state, or hidden units' activities, of the network can be determined uniquely based on the current and previous inputs and outputs provided the network has been running for a sufficiently long time. The formal definition of echo states from [13] is as follows:

Definition 1. ***Echo States****: Assume that $\mathbf{x}_i$ and $\mathbf{y}_i$ belong to compact intervals R_{in} and R_{out} respectively. Assume that for every left-infinite input and output sequence and for all hidden states $\mathbf{h}_i$ and $\mathbf{h}'_i$ such that*

$$\begin{aligned}\mathbf{h}_{i+1} &= \sigma(\mathbf{W}^T\mathbf{x}_{i+1} + \mathbf{W}_{rec}\mathbf{h}_i + \mathbf{W}_{fb}\mathbf{y}_i)\\ \mathbf{h}'_{i+1} &= \sigma(\mathbf{W}^T\mathbf{x}_{i+1} + \mathbf{W}_{rec}\mathbf{h}'_i + \mathbf{W}_{fb}\mathbf{y}_i)\end{aligned} \tag{3}$$

we have $\mathbf{h}_i = \mathbf{h}'_i$ for all $i \leq 0$. Then the network in Fig. 1 is said to have the echo states with respect to R_{in} and R_{out}.

Assume that the maximum eigenvalue of $\mathbf{W}_{rec}$ is λ_{max}, the activation function for hidden units is sigmoid and $\mid \lambda_{max} \mid > 4$, then the network does not have echo states. This is a sufficient condition for the echo states to not exist[1] [13]. However, as emphasized in [13], in practice, when $\mid \lambda_{max} \mid < 4$ the network has echo states. There is a similar sufficient condition under which the exploding gradient problem for recurrent weights would not happen [16].

It is shown in [17] that the connection between the hidden and output layers are important ones for mapping inputs to outputs. Therefore, in ESN training, only the output weight matrix, $\mathbf{U}$, are trained. Input recurrent weight matrices are carefully fixed. There are three main steps in training ESN: constructing a network with echo state property, computing the network states, and estimating the output weights.

To construct a network with the echo state property, the input weight matrix $\mathbf{W}$ and the sparse recurrent weight matrix $\mathbf{W}_{rec}$ are randomly generated. Then, the maximum eigenvalue of $\mathbf{W}_{rec}$ is calculated and all entries of $\mathbf{W}_{rec}$ are renormalized as follows:

$$\mathbf{W}_{rec} = \lambda\frac{\mathbf{W}_{rec}}{\lambda_{max}} \tag{4}$$

where $\lambda < 4$ for sigmoid activation function. λ is also an important parameter which affects the memory length of the network and should be determined based on the required memory size for the specific task. As emphasized in [13], the entries of the input weight matrix is also of great importance, large entries cause most hidden units to saturate while small entries will lead hidden units to stay in the linear region of the sigmoid function. The value of λ should be predetermined and fixed based on the desired task and before the second step (calculating the network states).

To find outputs of the hidden layer units, hidden states are initialized to zero or an initial state. Then network runs freely for i_{trans} time steps where hidden states for each time step are calculated using (1) with $\mathbf{W}_{fb} = 0$. After i_{trans} time steps, the hidden state vectors are stacked in matrix $\mathbf{H}$, i.e.

$$\mathbf{H} = [\mathbf{h}_{i_{trans}}\mathbf{h}_{i_{trans}+1}\ldots\mathbf{h}_N] \tag{5}$$

[1]Note that in [13] the condition is stated as $\mid \lambda_{max} \mid > 1$ because the activation function considered is hyperbolic tangent.

where N is the number of time steps.

To calculate the output weights $\mathbf{U}$, we stack the desired outputs or targets corresponding to input signal $\mathbf{x}_i$ as a matrix $\mathbf{T}$; i.e.,

$$\mathbf{T} = [\mathbf{t}_{i_{trans}} \mathbf{t}_{i_{trans}+1} \ldots \mathbf{t}_N] \tag{6}$$

Since $\mathbf{H}$ is computed using all known quantities (incluidng the fixed input and recurrent matrices) using (1) and hence it is known also, $\mathbf{U}$ can obtained by minimizing the following mean-square-error cost function:

$$E = \| \mathbf{U}^T \mathbf{H}_c - \mathbf{T} \|_F^2 = tr[(\mathbf{U}^T \mathbf{H}_c - \mathbf{T})(\mathbf{U}^T \mathbf{H}_c - \mathbf{T})^T] \tag{7}$$

where F stands for the Frobenius norm of a matrix, $tr(.)$ is the trace of a matrix and

$$\begin{aligned} \mathbf{H}_c &= [\mathbf{H}\ \mathbf{X}] \\ \mathbf{X} &= [\mathbf{x}_{i_{trans}} \mathbf{x}_{i_{trans}+1} \ldots \mathbf{x}_N] \end{aligned} \tag{8}$$

Minimizing (7), we have the globally optimal estimate of $\mathbf{U}$ determined by setting the gradient of the above cost function to zero and solving it; i.e.,

$$\begin{aligned} \frac{\partial E}{\partial U} &= 2\mathbf{H}_c(\mathbf{U}^T \mathbf{H}_c - \mathbf{T})^T = 0 \\ \mathbf{U} &= (\mathbf{H}_c \mathbf{H}_c^T)^{-1} \mathbf{H}_c \mathbf{T}^T \end{aligned} \tag{9}$$

In practical implementation, to prevent inaccurate results when $\mathbf{H}\mathbf{H}^T$ is or is close to be singular, the following solution of "ridge regression" is used for the estimate of $\mathbf{U}$:

$$\mathbf{U} = (\mathbf{H}_c \mathbf{H}_c^T + \mu \mathbf{I})^{-1} \mathbf{H}_c \mathbf{T}^T \tag{10}$$

where $\mathbf{I}$ is the identity matrix and μ is a fixed positive number.

3 Learning the Input Weight Matrix in ESN

Assuming the memory of the network extends back to m time steps, we use the following notation to facilitate the development of the learning method for the input weight matrix $\mathbf{W}$:

$$\begin{aligned} &\mathbf{X}_1 = [\mathbf{x}_1\ \mathbf{x}_{m+1}\ \mathbf{x}_{2m+1} \ldots],\ \mathbf{X}_2 = [\mathbf{x}_2\ \mathbf{x}_{m+2}\ \mathbf{x}_{2m+2} \ldots],\ \ldots \\ &\mathbf{H}_1 = [\mathbf{h}_1\ \mathbf{h}_{m+1}\ \mathbf{h}_{2m+1} \ldots],\ \mathbf{H}_2 = [\mathbf{h}_2\ \mathbf{h}_{m+2}\ \mathbf{h}_{2m+2} \ldots],\ \ldots \\ &\mathbf{T}_1 = [\mathbf{t}_1\ \mathbf{t}_{m+1}\ \mathbf{t}_{2m+1} \ldots],\ \mathbf{T}_2 = [\mathbf{t}_2\ \mathbf{t}_{m+2}\ \mathbf{t}_{2m+2} \ldots],\ \ldots \end{aligned} \tag{11}$$

Therefore, equations (1) and (2) can be written as:

$$\mathbf{H}_{i+1} = \sigma(\mathbf{W}^T \mathbf{X}_{i+1} + \mathbf{W}_{rec} \mathbf{H}_i) \tag{12}$$

$$\mathbf{Y}_{i+1} = \mathbf{U}^T \mathbf{H}_{i+1} \tag{13}$$

To find the gradient of the cost function E with respect to $\mathbf{W}$ and learn input weights $\mathbf{W}$ we consider two cases in the remainder of this section. In Case 1, we assume that $\mathbf{U}$ does not depend on $\mathbf{W}$ and in Case 2 we take into account the dependency between $\mathbf{U}$ and $\mathbf{W}$. Note that in both cases we take into account the time dependency among the hidden state vectors in $\mathbf{H}$, i.e., the dependency of $\mathbf{h}_{i+1}$ on $\mathbf{h}_i$ at every time step i [2]. Since Case 2 is a more realistic formulation of the gradient, it is used for learning the input weight matrix in presenting our experimental results in Section 5. We derive the gradient for one time step dependency and then generalize it to n time step dependency, i.e., $\mathbf{H}_i$ depends on $\mathbf{H}_{i-1}$, $\mathbf{H}_{i-2}$ and so on up to n time steps.

[2]Note that this is one of the main differences with the work presented in [18, 19] where there is no temporal connection in the single layer network and hence no time dependency is considered.

3.1 Case 1

The gradient of the cost function with respect to $\mathbf{W}$ can be written as

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{W}} &= \frac{\partial}{\partial \mathbf{W}} tr[(\mathbf{U}^T\mathbf{H}_2 - \mathbf{T}_2)(\mathbf{U}^T\mathbf{H}_2 - \mathbf{T}_2)^T] \\
&= \frac{\partial}{\partial \mathbf{W}} tr[(\mathbf{U}^T\sigma(\mathbf{W}_{rec}\mathbf{H}_1 + \mathbf{W}^T\mathbf{X}_2) - \mathbf{T}_2)(\mathbf{U}^T\sigma(\mathbf{W}_{rec}\mathbf{H}_1 + \mathbf{W}^T\mathbf{X}_2) - \mathbf{T}_2)^T] \\
&= [\frac{\partial}{\partial \mathbf{W}}\sigma(\mathbf{W}_{rec}\mathbf{H}_1 + \mathbf{W}^T\mathbf{X}_2)][2\mathbf{U}^T(\mathbf{U}^T\mathbf{H}_2 - \mathbf{T}_2)^T]
\end{aligned} \tag{14}$$

Assuming that $\mathbf{H}_1$ depends on $\mathbf{W}$, i.e., $\mathbf{H}_1 = \sigma(\mathbf{W}_{rec}\mathbf{H}_0 + \mathbf{W}^T\mathbf{X}_1)$, and denoting the term independent of $\mathbf{W}$ to be estimated:

$$\mathbf{S} = 2\mathbf{U}^T(\mathbf{U}^T\mathbf{H}_2 - \mathbf{T}_2)^T \tag{15}$$

then using chain rule of calculus we have:

$$\frac{\partial E}{\partial \mathbf{W}} = [\frac{\partial}{\partial \mathbf{W}}[\mathbf{W}_{rec}\sigma(\mathbf{W}_{rec}\mathbf{H}_0 + \mathbf{W}^T\mathbf{X}_1) + \mathbf{W}^T\mathbf{X}_2]]\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \circ \mathbf{S} \tag{16}$$

and therefore

$$\frac{\partial E}{\partial \mathbf{W}} = \mathbf{X}_1[\mathbf{H}_1^T \circ (\mathbf{1} - \mathbf{H}_1^T)\mathbf{W}_{rec}^T \circ \mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \circ \mathbf{S}] + \mathbf{X}_2[\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \circ \mathbf{S}] \tag{17}$$

where $\circ$ is element-wise multiplication.

3.2 Case 2

The gradient calculated in Case 1 is not accurate because the dependency between $\mathbf{U}$ and $\mathbf{W}$ is ignored. To take this dependency into consideration, the first line of equation (14) is rewritten as:

$$\frac{\partial E}{\partial \mathbf{W}} = \frac{\partial}{\partial \mathbf{W}} tr(\underbrace{\mathbf{U}^T\mathbf{H}_2\mathbf{H}_2^T\mathbf{U}}_{a} - \underbrace{\mathbf{U}^T\mathbf{H}_2\mathbf{T}^T}_{b} - \mathbf{T}\mathbf{H}_2^T\mathbf{U} + \mathbf{T}\mathbf{T}^T) \tag{18}$$

Substituting $\mathbf{U} = (\mathbf{H}_2\mathbf{H}_2^T)^{-1}\mathbf{H}_2\mathbf{T}_2^T$ we have

$$\begin{aligned}
a &= \mathbf{T}\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-T}\underbrace{\mathbf{H}_2\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-1}}_{\mathbf{I}}\mathbf{H}_2\mathbf{T}^T \\
&= \mathbf{T}\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-T}\mathbf{H}_2\mathbf{T}^T
\end{aligned} \tag{19}$$

and

$$b = \mathbf{T}\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-T}\mathbf{H}_2\mathbf{T}^T \tag{20}$$

therefore

$$\frac{\partial E}{\partial \mathbf{W}} = -\frac{\partial}{\partial \mathbf{W}} tr(\mathbf{T}_2\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-1}\mathbf{H}_2\mathbf{T}_2^T) \tag{21}$$

Since $tr(\mathbf{AB}) = tr(\mathbf{BA})$ and $tr(\mathbf{A}) = tr(\mathbf{A}^T)$ we have

$$\frac{\partial E}{\partial \mathbf{W}} = -\frac{\partial}{\partial \mathbf{W}} tr((\mathbf{H}_2\mathbf{H}_2^T)^{-1}\mathbf{H}_2\mathbf{T}_2^T\mathbf{T}_2\mathbf{H}_2^T) \tag{22}$$

Using the chain rule presented in equation (126) of [20], the gradient can be written as:

$$\frac{\partial E}{\partial \mathbf{W}} = -\underbrace{[\frac{\partial}{\partial \mathbf{W}}\mathbf{H}_2]}_{\mathbf{B}}\underbrace{[\frac{\partial}{\partial \mathbf{H}_2^T} tr((\mathbf{H}_2\mathbf{H}_2^T)^{-1}\mathbf{H}_2\mathbf{T}_2^T\mathbf{T}_2\mathbf{H}_2^T)]}_{\mathbf{A}} \tag{23}$$

Below we first calculate matrix $\mathbf{A}$ and then matrix $\mathbf{B}$.

For constant values of hidden states $\mathbf{H}_2$ we define $\mathbf{F}$, $\mathbf{M}$ and $\mathbf{S}$ as follows:

$$\begin{aligned}
\mathbf{F} &= (\mathbf{H}_2\mathbf{H}_2^T)^{-1} \\
\mathbf{M} &= \mathbf{T}_2^T\mathbf{T}_2 \\
\mathbf{S} &= \mathbf{H}_2\mathbf{M}\mathbf{H}_2^T
\end{aligned} \tag{24}$$

Then, using the chain rule again, we obtain

$$\mathbf{A} = \underbrace{\frac{\partial}{\partial \mathbf{H}_2^T} tr(\mathbf{F}\mathbf{H}_2\mathbf{M}\mathbf{H}_2^T)}_{\mathbf{A}_1} + \underbrace{\frac{\partial}{\partial \mathbf{H}_2^T} tr((\mathbf{H}_2\mathbf{H}_2^T)^{-1}\mathbf{S})}_{\mathbf{A}_2} \tag{25}$$

considering the fact that $tr(\mathbf{AB}) = tr(\mathbf{BA})$, $\mathbf{A_1}$ can be written as

$$\mathbf{A}_1 = \frac{\partial}{\partial \mathbf{H}_2^T} tr(\mathbf{M}\mathbf{H}_2^T\mathbf{F}\mathbf{H}_2) \tag{26}$$

From equation (107) of [20] we have

$$\mathbf{A}_1 = \mathbf{M}^T\mathbf{H}_2^T\mathbf{F}^T + \mathbf{M}\mathbf{H}_2^T\mathbf{F} \tag{27}$$

Since $\mathbf{F}^T = \mathbf{F}$ and $\mathbf{M}^T = \mathbf{M}$, $\mathbf{A}_1$ can be written as

$$\mathbf{A}_1 = 2\mathbf{M}\mathbf{H}_2^T\mathbf{F} \tag{28}$$

Considering equation (114) of [20], $\mathbf{A}_2$ will be as follows

$$\mathbf{A}_2 = -(\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-1})(\mathbf{S} + \mathbf{S}^T)(\mathbf{H}_2\mathbf{H}_2^T)^{-1} \tag{29}$$

substituting $\mathbf{S}$ and using $\mathbf{M} = \mathbf{M}^T$ results in

$$\mathbf{A}_2 = -2\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-1}(\mathbf{H}_2\mathbf{M}\mathbf{H}_2^T)(\mathbf{H}_2\mathbf{H}_2^T)^{-1} \tag{30}$$

Now substituting $\mathbf{M}$ and $\mathbf{F}$ in (24) results in the final formulation for $\mathbf{A}$ as follows

$$\mathbf{A} = 2\mathbf{T}_2^T\mathbf{T}_2\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-1} - 2\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-1}\mathbf{H}_2\mathbf{T}_2^T\mathbf{T}_2\mathbf{H}_2^T(\mathbf{H}_2\mathbf{H}_2^T)^{-1} \tag{31}$$

To calculate $\mathbf{B}$ we assume that there is just one time step dependency, i.e., $\mathbf{H}_2$ depends on $\mathbf{H}_1$ and $\mathbf{W}$ and $\mathbf{H}_1$ does not depend on $\mathbf{H}_0$ but depends on $\mathbf{W}$. The generalization to an arbitrary number of time steps is straightforward and is presented at the end of this section. From (23) and (12) we write $\mathbf{B}$ as:

$$\mathbf{B} = \frac{\partial}{\partial \mathbf{W}}[\sigma(\mathbf{W}_{rec}\sigma(\mathbf{W}_{rec}\mathbf{H}_0 + \mathbf{W}^T\mathbf{X}_1) + \mathbf{W}^T\mathbf{X}_2)] \tag{32}$$

which is similar to the term calculated in (17) and therefore

$$\mathbf{B} = \mathbf{X}_1[\mathbf{H}_1^T \circ (\mathbf{1} - \mathbf{H}_1^T)\mathbf{W}_{rec}^T \circ \mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T)] + \mathbf{X}_2[\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T)] \tag{33}$$

By substituting (31) and (33) in (23) we get the gradient formulation for one time step dependency as follows:

$$\frac{\partial E}{\partial \mathbf{W}} = -[\mathbf{X}_1[\mathbf{H}_1^T \circ (\mathbf{1} - \mathbf{H}_1^T)\mathbf{W}_{rec}^T \circ \mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \circ \mathbf{A}] + \mathbf{X}_2[\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \circ \mathbf{A}]] \tag{34}$$

This gradient formulation can be generalized for an arbitrary number of time steps as follows:

$$\frac{\partial E}{\partial \mathbf{W}} = -[\sum_{i=1}^{n} \mathbf{X}_i\mathbf{C}_i] \tag{35}$$

where n is the number of time steps and

$$\begin{aligned} \mathbf{C}_i &= [\mathbf{H}_i^T \circ (\mathbf{1} - \mathbf{H}_i^T)\mathbf{W}_{rec}^T] \circ \mathbf{C}_{i+1} \quad , \; for \;\; i = 1, \ldots, n-1 \\ \mathbf{C}_n &= \mathbf{H}_n^T \circ (\mathbf{1} - \mathbf{H}_n^T) \circ \mathbf{A} \end{aligned} \tag{36}$$

To calculate $\mathbf{A}$ using (31), $\mathbf{H}_n$ and $\mathbf{T}_n$ are used.

After calculating the gradient of the cost function with respect to $\mathbf{W}$, the input weights $\mathbf{W}$ are updated using the following update equation

$$\mathbf{W}_{i+1} = \mathbf{W}_i - \alpha\frac{\partial E}{\partial \mathbf{W}_i} + \beta(\mathbf{W}_i - \mathbf{W}_{i-1}) \tag{37}$$

where α is the step size and

$$\begin{aligned} \beta &= \frac{m_{old}}{m_{new}} \\ m_{new} &= \frac{1 + \sqrt{1 + 4m_{old}^2}}{2} \end{aligned} \tag{38}$$

and where the initial value for m_{old} and m_{new} is 1. The third term in (37) helps the algorithm to converge faster and is based on the FISTA algorithm proposed in [21] and used in [19]. To get a better insight of the role of the third term in (37), different values of β over the number of epochs used in updating $\mathbf{W}$ and β is presented in Fig. 2. As observed in this figure, for the first epochs the effect of the third term in (37) is small but gradually increases as the value of β is further updated.

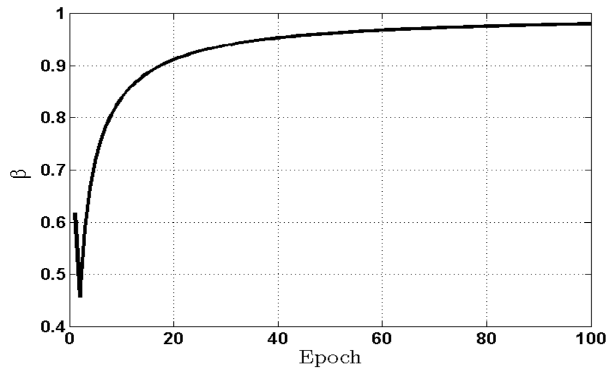

Figure 2: Different values of β over epochs as in Equation (37)

4 Learning the Recurrent Weight Matrix ($\mathbf{W}_{rec}$) in the ESN

To learn the recurrent weights, the gradient of the cost function with respect to $\mathbf{W}_{rec}$ should be calculated. We first derive the formulation for two time steps dependency, i.e., $\mathbf{H}_2$ depends on $\mathbf{H}_1$ and $\mathbf{H}_1$ depends on $\mathbf{H}_0$ and no more time dependency. Then it will be generalized to the arbitrary number of time steps. The same method that is used in section 3 can be used to get the following formulation for gradient:

$$\frac{\partial E}{\partial \mathbf{W}_{rec}} = -\underbrace{[\frac{\partial}{\partial \mathbf{W}_{rec}}\mathbf{H}_2]}_{\mathbf{B}}\underbrace{[\frac{\partial}{\partial \mathbf{H}_2^T} tr((\mathbf{H}_2\mathbf{H}_2^T)^{-1}\mathbf{H}_2\mathbf{T}_2^T\mathbf{T}_2\mathbf{H}_2^T)]}_{\mathbf{A}} \tag{39}$$

$\mathbf{A}$ will be the same as (31). To calculate $\mathbf{B}$ we have

$$\begin{aligned}\mathbf{B} &= \frac{\partial}{\partial \mathbf{W}_{rec}}[\sigma(\mathbf{W}_{rec}\sigma(\mathbf{W}_{rec}\mathbf{H}_0 + \mathbf{W}^T\mathbf{X}_1) + \mathbf{W}^T\mathbf{X}_2)] \\ &= [\frac{\partial}{\partial \mathbf{W}_{rec}}[\mathbf{W}_{rec}\underbrace{\sigma(\mathbf{W}_{rec}\mathbf{H}_0 + \mathbf{W}^T\mathbf{X}_1)}_{\mathbf{H}_1} + \mathbf{W}^T\mathbf{X}_2]]\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \\ &= [\mathbf{H}_1 + \mathbf{W}_{rec}\underbrace{\mathbf{H}_0[\mathbf{H}_1^T \circ (\mathbf{1} - \mathbf{H}_1^T)]}_{\frac{\partial \mathbf{H}_1}{\partial \mathbf{W}_{rec}}}]\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T)\end{aligned} \tag{40}$$

and therefore the gradient will be:

$$\frac{\partial E}{\partial \mathbf{W}_{rec}} = \mathbf{H}_1[\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \circ \mathbf{A}] + \mathbf{W}_{rec}\mathbf{H}_0[\mathbf{H}_1^T \circ (\mathbf{1} - \mathbf{H}_1^T) \circ (\mathbf{H}_2^T \circ (\mathbf{1} - \mathbf{H}_2^T) \circ \mathbf{A})] \tag{41}$$

It can be generalized for arbitrary number of time steps as follows:

$$\frac{\partial E}{\partial \mathbf{W}_{rec}} = \sum_{i=1}^{n} \mathbf{W}_{rec}^{n-i}\mathbf{H}_{i-1}\mathbf{C}_i \tag{42}$$

where $\mathbf{H}_0$ includes the initial hidden states and

$$\begin{aligned}\mathbf{C}_n &= \mathbf{H}_n^T \circ (\mathbf{1} - \mathbf{H}_n^T) \circ \mathbf{A} \\ \mathbf{C}_i &= \mathbf{H}_i^T \circ (\mathbf{1} - \mathbf{H}_i^T) \circ \mathbf{C}_{i+1}\end{aligned} \tag{43}$$

and $\mathbf{A}$ is calculated using (31) based on $\mathbf{H}_n$ and $\mathbf{T}_n$.

Only the non-zero entries of the sparse matrix $\mathbf{W}_{rec}$ are updated using (37) with the gradient calculated in (42). To make sure that the network has the echo state property after each epoch, the entries of $\mathbf{W}_{rec}$ are renormalized such that the maximum eigenvalue of $\mathbf{W}_{rec}$ is λ that is predetermined in (4). This renormalization also prevents the gradient explosion problem for recurrent weights from happening.

A summary of the learning method is as follows:

- The echo state network with predetermined maximum eigenvalue of $\mathbf{W}_{rec}$ is constructed based on the explanations presented in section 2.
- Input weights matrix $\mathbf{W}$ is updated based on (35), (36), (37) and (38).
- Non-zero entries of the sparse recurrent weights matrix $\mathbf{W}_{rec}$ are updated based on (42), (43), (37) for $\mathbf{W}_{rec}$ and (38).
- Updated $\mathbf{W}_{rec}$ is renormalized to have the predetermined maximum eigenvalue λ.
- The forward pass is repeated with the updated input and recurrent weights to find the hidden states. The network runs freely for i_{trans} time steps and then the hidden states are recorded as matrix $\mathbf{H}$.
- Taking into account the direct connections from the input to output in the network, the output weight matrix $\mathbf{U}$ is calculated using (10).

To prevent the value of the gradient w.r.t $\mathbf{W}$ from explosion we have used a similar approach proposed in [16] where the gradient value is renormalized when it is greater than a threshold.

5 Experiments

We have carried out the experiments for frame-level classification of phone states on the TIMIT dataset using the ESN with all parameters learned as discussed so far. The training data includes 1,124,589 frames. The validation set has 122,488 frames from 50 speakers. The results are reported using the core test set consisting of 192 sentences and 57,920 frames. The speech is analysed using the standard Mel Frequency Cepstral Coefficients (MFCC). Each feature vector has 39 entries, the first 13 entries are 12 MFCC coefficients and the energy, the second 13 entries are their first order derivative ($\triangle$) and the energy in the derivative and the third 13 entries are their second order derivative ($\triangle\triangle$) and the energy in the derivative. We have used 3 states for each of 61 phones resulting in a target class vector with 183 entries. Phone state labels are extracted using a GMM-HMM system which aligns the frames with their corresponding states.

We have used a context window of 3 frames for all experiments resulting in the input vectors with $3 \times 39 = 117$ entries. The regularization parameter μ used in (10) is set to 10^{-8}. The maximum eigenvalue of $\mathbf{W}_{rec}$ is set to be 3.9. We have used a step size (α in (37)) of 0.07. The task is classification of each frame in the TIMIT core test set into one of 183 phone states. The results are presented in Table 1 for different hidden layer sizes in the ESN, one for each row in the table. The results are also arranged by for four different ways of learning the input and recurrent weigh matrices $\mathbf{W}$ and $\mathbf{W}_{rec}$, where m is the number of time steps in incorporating matrices' dependencies in the learning. The column of "ESN" refers to the traditional ESN as in [7, 8, 9, 10] where $\mathbf{W}$ and $\mathbf{W}_{rec}$ are not learned.

Table 1: Frame-level phone-state classification error rates for the TIMIT core test set

Hidden units	ESN	Learning $\mathbf{W}$ with $m = 1$	Learning $\mathbf{W}$ and $\mathbf{W}_{rec}$ with $m = 1$	Learning $\mathbf{W}$ and $\mathbf{W}_{rec}$ with $m = 3$
100	75.5%	66.7%	64.0 %	63.2%
300	71.9%			
500	70.1%	59.8%	57.5%	56.8%
1000	66.9%			
2000	63.8%	54.2%	52.7%	52.1%
4000	60.9%			
10000	57.1%	49.5%	48.0%	46.8%
20000	54.8%			
30000	53.3%	45.9%	44.5%	43.0%
40000	52.7%			

The preliminary experimental results shown in Table verify that learning input and recurrent weight matrices in the ESN is superior to the ESN with the same structure but without learning the two matrices. Further, the longer time steps are incorporated in the learning, the lower error rates are obtained. On the column of "ESN", we also observe that the traditional ESN improves its performance as the number of hidden units increases, consistent with the findings reported in [10]. Finally, we would like to remark that the results obtained so far are very preliminary, and the task in on frame-level classification of 183 phone states. Our first step of research is focused on this easiest task since it is a pure and simple machine learning problem and it requires no expertise in speech recognition. The next steps are to move 1) from 183-state classification to 39-phone classification; 2) from frame level to segment level (which requires dynamic programing over three states of each phone); and 3) from classification (with no phone insertion and deletion errors) to recognition (with phone insertion and deletion errors). Then the results will be able to be meaningfully compared with other apporaches in the literature on the TIMIT phone recognition task.

6 Discussion and Conclusion

The main idea of this paper is straightforward: the traditional ESN learns only one of three important sets of weight matrices, and we want to learn them all. The key property that characterizes the ESN is the use of linear output units so that the learning is simple, convex, least-square ridge regression problem with a global optimum (in learning the output weights). In extending learning the output weights only to learning input and recurrent weights, we make use the same property of linear output units to develop and formulate constraints among various sets of ESN weight matrices. Such constraints are then used to derive analystic forms of the error gradients with respect to the input and recurrent weights to be learned. The standard learning method of BPTT for the general RNN (with typically nonlinear output units) does not admit to analytical forms of gradient computation. BPTT requires recursively propagating the error signal backward through time, a very different style of computation and learning than what we have developed in this work for ESN.

In this paper we focus on ESNs with one layer and with no feedback connection, i.e., $\mathbf{W}_{fb} = 0$, without loss of generality. As our future work, oo build more layers of ESN, we can simply stack the output of one-layer ESN on top of another, or we can combine the output with the original data input and/or with hidden units. Also, in the current work with one-layer ESN, we take into account the dependency between $\mathbf{U}$ and $\mathbf{W}$, $\mathbf{h}_i$ and $\mathbf{W}$, $\mathbf{h}_i$ and $\mathbf{h}_{i-1}$, etc. When more layers of ESN are built, richer dependency becomes available to exploit but the same principle used in this work in deriving the analysitc forms of the gradient computation applies to the multiple-layer ESN in our future work.

References

[1] Jeffrey L. Elman, “Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[2] A. J. Robinson, “An application of recurrent nets to phone probability estimation,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 298–305, August 1994.

[3] L. Deng, K. Hassanein, and M. Elmasry, “Analysis of the correlation structure for a neural predictive model with application to speech recognition,” *Neural Networks*, vol. 7, no. 2, pp. 331–339, 1994.

[4] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, “Recurrent neural network based language model.,” in *Proc. INTERSPEECH*, Makuhari, Japan, September 2010, pp. 1045–1048.

[5] A. Graves, “Sequence transduction with recurrent neural networks,” in *Representation Learning Workshp, ICML*, 2012.

[6] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, “Advances in optimizing recurrent networks,” in *Proc. ICASSP*, Vancouver, Canada, May 2013.

[7] H. Jaeger, *The “echo state” approach to analysing and training recurrent neural networks*, GMD Report 148, GMD - German National Research Institute for Computer Science, 2001.

[8] H. Jaeger, *Short term memory in echo state networks*, GMD Report 152, GMD - German National Research Institute for Computer Science, 2001.

[9] Herbert Jaeger and Harald Haas, “Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication,” *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[10] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, “Phoneme recognition with large hierarchical reservoirs,” *Advances in Neural Information Processing Systems*, 2009.

[11] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[12] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *Proc. ICML*, Atlanta, GA, June 2013.

[13] Herbert Jaeger, “A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach,” Tech. Rep., Fraunhofer Institute for Autonomous Intelligent Systems (AIS) since 2003: International University Bremen, 2005.

[14] Tomas Mikolov, Stefan Kombrink, Lukas Burget, JH Cernocky, and Sanjeev Khudanpur, “Extensions of recurrent neural network language model,” in *Proc. IEEE ICASSP*, Prague, Czech, May 2011, pp. 5528–5531.

[15] I. Sutskever, *Training Recurrent Neural Networks*, Ph.D. thesis, Ph. D. thesis, University of Toronto, 2013.

[16] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, “On the difficulty of training recurrent neural networks,” *http://arxiv.org/abs/1211.5063v2*, 2013.

[17] Ulf D. Schiller and Jochen J. Steil, “Analyzing the weight dynamics of recurrent learning algorithms,” *Neurocomputing*, vol. 63, no. 0, pp. 5 – 23, 2005.

[18] Li Deng and D. Yu, “Deep convex networks for speech pattern classification,” *Proc. Interspeech*, 2011.

[19] D. Yu and L. Deng, “Efficient and effective algorithms for training single-hidden-layer neural networks,” *Pattern Recognition Letters*, vol. 33, no. 5, pp. 554–558, 2012.

[20] Kaare Brandt Petersen and Michael Syskind Pedersen, “The matrix cookbook,” Tech. Rep., 2008.

[21] Amir Beck and Marc Teboulle., *Gradient-based algorithms with applications to signal-recovery problems*, Cambridge University Press, 2009.