Modeling $D_{st}$with Recurrent EM Neural Netwroks

Derrick Takeshi Mirikitani and Lahcen Ouarbya
Department of Computer Science, Goldsmiths College, University of London,
New Cross, London, London SE14 6NW, UK
D.T.Mirikitani@gold.ac.uk

Abstract

Recurrent Neural Networks have been used extensively for space weather forecasts of geomagnetospheric disturbances. One of the major drawbacks for reliable forecasts have been the use of training algorithms that are unable to account for model uncertainty and noise in data. We propose a probabilistic training algorithm based on the Expectation Maximization framework for parameterization of the model, which makes use of a forward filtering and backward smoothing Expectation step, and a Maximization step in which the model uncertainty and measurement noise estimates are computed. The inputs to the network are based on three parameters of the interplanetary magnetic field ($\boldsymbol{IMF}$), $\boldsymbol{b_z}$, $\boldsymbol{b}^2$, and $\boldsymbol{b_y}^2$, along with the $\boldsymbol{D_{st}}$ index. Through numerical experimentation it is shown that the proposed model allows for reliable forecasts and also outperforms other neural time series models trained with the Extended Kalman Filter, and gradient descent learning.

1.1 Introduction

For decades, a number of studies [1, 2, 3] have shown that a change in the solar activities induces a disturbance of the earths magnetic field. This disturbance of the near-earth environment is influenced by the injection of energetic particles through the solar wind[1] into the magnetosphere. The solar wind carries the sun's magnetic field through the solar system, forming what we call the Interplanetary Magnetic Field[2] ($\boldsymbol{IMF}$). A transfer of energy from the solar wind into the magnetosphere takes place when the $\boldsymbol{IMF}$ opposes the Earth's magnetic field. A disturbance of the magnetosphere, known as a magnetic storm, occurs if this transfer of energy persists for several hours [3]. Geomagnetic storms can have many negative effects on technical systems in space and on Earth, such as a change in a spacecraft orientation or power lines on the Earth.

Forecasts of the earth's magnetic field can give vital information about the intensity of future magnetospheric disturbances. At mid-latitudes, magnetic storms are measured in terms of the horizontal component of the Earth's magnetic field [3]. This horizontal component is averaged to form an index known as $\boldsymbol{D_{st}}$. Studies have shown a correlation between the intensity of magnetic storms and the value of of the $\boldsymbol{D_{st}}$ index [4, 5, 6, 7]; where the more negative the $\boldsymbol{D_{st}}$ index the greater the intensity of the magnetic storm. The physical interaction between the $\boldsymbol{IMF}$and the Geo-magnetosphere are not fully understood, and thus previous researchers have built non-parametric predictive models usually based on recurrent neural networks (RNNs) to forecast the $\boldsymbol{D_{st}}$index [8, 9, 10]. In this paper we extend the work in the field by proposing an $\boldsymbol{EM}$ Kalman filtering and smoothing framework for estimation of RNN parameters. The advantage of our approach is the probabilistic representation of model uncertainty and noise in the data, which has been neglected in prior work in the area. This is achieved through the use of the $\boldsymbol{EM}$algorithm for tuning of model hyperparameters representing the process and measurement noise components, leading to improved out of sample performance on $\boldsymbol{D_{st}}$ forecasting tasks. In the following section we provide an overview of the use of neural networks in geomagnetic storm forecasting.

[1] Solar winds are a stream of charged particles, mostly electrons and protons, that are ejected from the upper atmosphere of the sun

[2] The sun's magnetic fi eld carried through the solar system by the solar wind

1.1.1 Use of Neural Networks in Geomagnetic Storm Prediction

Solar wind has a great impact on the magnetosphere. A transfer of mass, energy and momentum through various processes takes place at the magnetopause boundary. The magnetosphere is a complex system, with multi-scale spatio-temporal behaviour. Neural Networks have established themselves as effective tools in the prediction of time series behaviour, especially for noisy data. They have successfully been used in the space weather forecasting. Several models have been developed for the prediction of the ring current index $\boldsymbol{D_{st}}$. In [8, 9, 10] the authors have used Neural Networks to predict geomagnetic storms. Different combinations of solar wind data, such as the density ($\boldsymbol{n}$), the velocity ($\boldsymbol{v}$), And the $\boldsymbol{IMF}$ data ($\boldsymbol{b}$, $\boldsymbol{b_x}$,$\boldsymbol{b_y}$,$\boldsymbol{b_z}$)[3] have been used as an input to various types of neural networks. It was found that forecasts produced by feed-forward Neural Networks gives a good results for the initial and main phase of magnetic storms, but not the recovery phase [3]. This results from the fact that the internal state of the magnetosphere has a great impact on the storm recovery phase. Freeman et al. [8] have used the $\boldsymbol{D_{st}}$ index the magnitude of $\boldsymbol{IMF}(\boldsymbol{b})$, the $\boldsymbol{IMF}$ southward component ($\boldsymbol{b_z}$), and the solar wind dynamic pressure ($\boldsymbol{n}$) to give one hour a head prediction of the $\boldsymbol{D_{st}}$. The use of Elman recurrent networks [11, 12] has shown to give a superior $\boldsymbol{D_{st}}$ prediction than feed forward Neural Networks. Wu and Lundstedt [11] have used different coupling functions of solar wind and $\boldsymbol{IMF}$ parameters as inputs to their Elman Recurrent Network for $\boldsymbol{D_{st}}$ prediction. In [12] Lundstedt et al. have used an optimised recurrent neural network, driven solely by hourly averages of the solar wind parameters, particle density ($\boldsymbol{n}$), and velocity ($\boldsymbol{v}$) and the $\boldsymbol{IMF}$ southward component, $\boldsymbol{b_z}$. They have shown that their recurrent model has smaller errors than previous models. Pallocia et al. [13] argued that the instruments used to measure the plasma parameter, pressure and velocity, can often be affected by enhanced X-Ray and energetic particle flux. This gives rise to wrong or missing gaps in the plasma data. The use of wrong data can have a negative effect on the storm prediction. They used the $\boldsymbol{IMF}$ parameters, $\boldsymbol{b_z}$, $\boldsymbol{b}^2$ and $\boldsymbol{b_y}^2$, only as inputs to predict $\boldsymbol{D_{st}}$. They have highlighted that for quite periods the inputs of Lundstedt and Wu [12] performs better than their algorithm, however their algorithm gives a better prediction in times of severe storms. They concluded that for the period of severe storms the plasma data are not reliable and will deleteriously affect the predictions of magnetic activity.

In this paper we incorporate the three components of the ($\boldsymbol{IMF}$), $\boldsymbol{b_z}$, $\boldsymbol{b}^2$, and $\boldsymbol{b_y}^2$, along with the $\boldsymbol{D_{st}}$ index as exogenous inputs to the RNN. We elaborate on a Maximum Likelihood training algorithm based on the $\boldsymbol{EM}$algorithm, for RNN parameter estimation, which allows for estimation of model uncertainty and noise in the data [16, 17, 18]. The extended Kalman Filter(EKF) [20] and smoother are used for sequential estimation of the network weights. The EKF is a second order estimation algorithm for neural networks [15], which evolves an approximate covariance matrix that encodes second order information about the underlying system during training. Through the use of the EM algorithm, the process and measurement noise hyper-parameters of the filters are estimated via a maximum likelihood approach, which leads to improved out of sample performance on $\boldsymbol{D_{st}}$forecasting. Experimental simulations show that forecasts of $\boldsymbol{D_{st}}$ with the EM-RNN algorithm provides more accurate forecasts that the RNN trained with first-order gradient descent [19], and an RNN trained with the second-order extended Kalman filter(EKF) [20].

This paper is organised as follows. In Section 1.2, the architecture of the RNN network is described. The next section provides the extended Kalman filter and smoother. Section 1.4 offers the EM-RNN training approach for recursive hyperparameter re-estimation. Section 1.5 presents the simulation results, and finally a brief conclusion is provided in Section 1.6.

1.2 Recurrent Neural Networks

RNNs are nonlinear adaptive models with internal states trainable by specialised weight adaptation algorithms. The recurrent architecture chosen for this study is known as the Williams and Zipser fully recurrent network [19], which is the most general RNN architecture. We adopt the following notation to describe the fully recurrent network: $\mathbf{s}_t$ is the input vector for each neuron, which contains the exogenous input to the network x_t, the bias b, and the previous activation of each neuron $\mathbf{s}_t = [c^{(1)}, \ldots, c^{(H)}, x^{(H+1)}, b^{(H+2)}]$, where $\{c^{(1)}, c^{(2)}, \ldots, c^{(H)}\}$ are the activations of the network at the previous time step. The superscript (l) refers to the l^{th} element $s^{(l)} \in \mathbf{s}_t$ for $l = 1, 2, \ldots, H+2$ where H is the number of neurons. The output activation of each neuron is defined as a function $y_t^{(i)} = g(\mathbf{w}_t^{(i)}, \mathbf{s}_t)$ where $\mathbf{w}_t^{(i)} = [w_{i,1}, \ldots, w_{i,H+2}]$ is the weight vector associated with the i^{th} neuron at time t, and the overall network weight vector is defined as $\mathbf{W} = [\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(H)}]$. The functions $g(\cdot)$ and are logistic sigmoidal nonlinearities $g(a) = 1/(1 + \exp(-a))$ which map the input a from $\mathcal{R}$ into a bounded interval $\Omega = (0, 1)$ of length $|\Omega| = 1$ where $\Omega \subset \mathcal{R}$.

[3] $\boldsymbol{b}$ is the magnitude of the $\boldsymbol{IMF}$, where, $\boldsymbol{b_x}$,$\boldsymbol{b_y}$, and $\boldsymbol{b_z}$ are the three component of the $\boldsymbol{IMF}$.

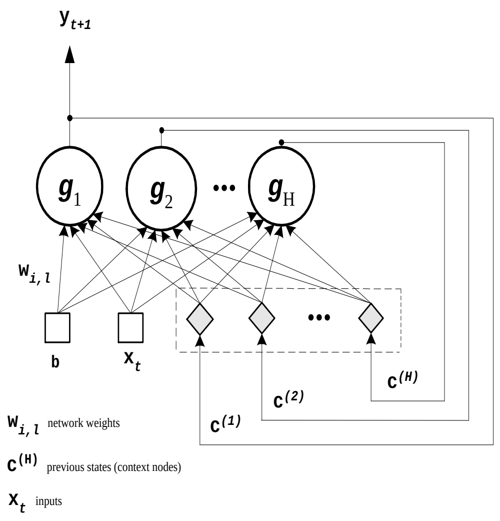

Fig. 1.1. Fully Recurrent Neural Network.

The fully recurrent neural network architecture consists of a single layer of processing neurons which are fully connected to each other. The input layer consists of the exogenous inputs along with the bias, which feed temporal information to the processing layer at each time step. A schematic diagram of the fully connected network is shown in Figure 1.1. Each neuron in the processing layer computes a weighted sum of the previous processing layer activations, along with the exogenous inputs to the network, and the bias given by

$$v_t^{(i)} = \sum_{l=1}^{H+2} w_{i,l} \mathbf{s}_t^{(l)} \tag{1.1}$$

where $w_{i,l}$ is the weight connecting the i^{th} neuron to the l^{th} component of the **s** vector. Each weighted sum v_i is then passed through the nonlinear activation function to produce the activation outputs

$$y_t^{(i)} = g^{(i)}(v_t^{(i)}) \tag{1.2}$$

where the output of the network is y_1. The entire network is referred to as highly nonlinear function h of the weights $\mathbf{W}_t$ and input x_t

$$d_t = h(\mathbf{W}_t, x_t) + \epsilon_t \tag{1.3}$$

where the noise ϵ_t is assumed to be independent zero-mean Gaussian with covariance R: $\epsilon_t \sim \mathcal{N}(0, R)$, and where d_t are the targets from the provided data set $D = \{x_t, d_t\}_{t=1}^N$, where $x_{t+1} = d_t$.

1.2.1 Derivative Computation with RTRL

The gradients of the network can be computed sequentially via the RTRL algorithm [19]. The RTRL algorithm minimizes the instantaneous squared error of the output neuron where the cost function is defined as

$$E_t = \frac{1}{2} \sum_{l=0}^{H} (\epsilon_t^{(l)})^2 = \mathbf{1}(l)_{l=1}(d_t - y_t^{(1)})^2 + \mathbf{1}(l)_{l \neq 1} 0 \tag{1.4}$$

The indicator function is $\mathbf{1}(a)_{a=A} = 1$ if $a = A$ and 0 otherwise, which sets the error to zero if there is no available output and target pattern for that particular neuron. Computing the partial derivative of the cost function with respect to the weights

$$\frac{\partial E}{\partial w_{i,j}} = -\sum_{l=0}^{H+2} \epsilon_t^{(l)} \frac{\partial y_t^{(l)}}{\partial w_{i,l}} \tag{1.5}$$

The computation of the gradient of the l^{th} weight of neuron i at time step t is as follows

$$\frac{\partial y_t^{(1)}}{\partial w_{i,l}} = g'(v_t^{(1)})\Big(\sum_{l=1}^{H} \frac{\partial y_t^{(l)}}{\partial w_{i,l}} w_{1,l+2} + \delta_{i,l} s^{(l)}\Big) \tag{1.6}$$

where $\delta_{i,l} = 1$ if $i = l$ and 0 otherwise. The partial derivatives then form the vector

$$\mathbf{j}_t = [\frac{\partial y_t^{(1)}}{\partial w_{1,1}}, \frac{\partial y_t^{(1)}}{\partial w_{1,2}}, \ldots, \frac{\partial y_t^{(H)}}{\partial w_{H,H+2}}] \tag{1.7}$$

As this paper is concerned with time series processing, only one output is used, however derivatives are easily extended to multiple output systems. In the next section we present Kalman filtering and smoothing algorithms for single output modelling.

1.3 Kalman Filtering and Smoothing

The most widely known recursive Bayesian state estimation algorithm is the Kalman filter. The Kalman filter provides the minimum variance and maximum likelihood estimate, assuming linearity in the process and measurement equations and gaussian noise terms. In situations with nonlinearity in the state space equations, the EKF has been proposed as an extension to the Kalman filter nonlinear systems.

In the Kalman filtering framework, it is assumed that the neural network parameter vector evolves over time in the sense of a first order stochastic process

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \boldsymbol{\nu}_t \tag{1.8}$$

where the process noise is assumed to be normally distributed $\boldsymbol{\nu} \sim \mathcal{N}(\mathbf{w}_{t-1}, \mathbf{Q})$. The RNN provides a nonlinear mapping of the evolving state and the system inputs to the measurements y_t, as defined in equation 1.3.

For neural networks, the EKF requires the computation of the Jacobian matrix $\mathbf{j}_t = \partial h(\cdot)/\partial \mathbf{W}_t$ of partial derivatives of the output y_{t+1} with respect to the weights of the network as in equation 1.7, where the Jacobian $\mathbf{j}_t$ is evaluated at each time step. The EKF has been a popular choice for training RNNs [20, 22, 23, 24, 25, 26, 27], due to the availability of neural network derivatives [20]. The following equations describe the EKF training algorithm:

$$\begin{aligned}
\mathbf{W}_{t+1}^{t} &= \mathbf{W}_{t}^{t} \\
\mathbf{P}_{t+1}^{t} &= \mathbf{P}_{t}^{t} + \mathbf{Q} \\
\mathbf{K}_{t+1}^{t+1} &= \mathbf{P}_{t+1}^{t}\mathbf{j}_{t+1}[\mathbf{j}_{t+1}\mathbf{P}_{t+1}^{t}\mathbf{j}_{t+1}^{T} + R]^{-1} \\
\hat{\mathbf{W}}_{t+1} &= \hat{\mathbf{W}}_{t+1}^{t} + \mathbf{K}_{t+1}(d_{t+1} - h(\hat{\mathbf{W}}_{t+1}^{t}, x_t)) \\
\mathbf{P}_{t+1}^{t+1} &= \mathbf{P}_{t+1}^{t} - \mathbf{K}_{t+1}\mathbf{j}_{t+1}^{T}\mathbf{P}_{t+1}^{t}
\end{aligned} \tag{1.9}$$

The EKF is a suboptimal estimator based on linearization of the nonlinearity of the underlying neural network. It provides an approximation of the state mean $\mathbf{W}_t$ and the state covariance $\mathbf{P}_t$. The matrix $\mathbf{K}_t$ is the Kalman gain.

After computing the estimates $\mathbf{W}_t$ and $\mathbf{P}_t$ by equations 1.9 the Rauch-Tung-Striebel smoother [21] is utilised for recursively computing corrections to the EKF estimates. This is achieved through the following backward recursions:

$$\begin{aligned}
\mathbf{S}_{t-1}^{t-1} &= \mathbf{P}_{t-1}^{t-1}(\mathbf{P}_{t}^{t-1})^{-1} \\
\hat{\mathbf{W}}_{t-1}^{N} &= \hat{\mathbf{W}}_{t-1}^{t-1}\mathbf{S}_{t-1}^{t-1}(\hat{\mathbf{W}}_{t}^{N} - \hat{\mathbf{W}}_{t-1}^{t-1}) \\
\mathbf{P}_{t-1}^{N} &= \mathbf{P}_{t-1}^{t-1} + \mathbf{S}_{t-1}^{t-1}(\mathbf{P}_{t}^{N} - \mathbf{P}_{t}^{t-1})(\mathbf{S}_{t-1}^{t-1})^{T} \\
\mathbf{P}_{t,t-1}^{N} &= \mathbf{P}_{t}^{t}(\mathbf{S}_{t-1}^{t-1})^{T} + \mathbf{S}_{t}^{t}(\mathbf{P}_{t+1,t}^{N} - \mathbf{P}_{t}^{t-1})(\mathbf{S}_{t-1}^{t-1})^{T}
\end{aligned} \tag{1.10}$$

The extended Kalman smoother provides a minimum variance Gaussian approximation to the posterior probability density function $p(\mathbf{W}|x_{1:N})$.

1.4 Expectation Maximisation Learning

The EM algorithm is an iterative method for finding a mode of the likelihood function $p(x_{1:N}|R,\mathbf{Q})$. The algorithm alternates between two steps, the E-step (expectation) and the M-step (maximisation). In the E-step, an estimate of the state **W** given the data $x_{1:N}$ and parameters $\boldsymbol{\theta} = [R, \mathbf{Q}, \boldsymbol{\mu}, \boldsymbol{\Sigma}]$ is produced, and in the M-step, the parameters $\boldsymbol{\theta}$ are then estimated given the new state.

Maximum likelihood estimation of the parameters **W** are found through maximising the complete likelihood of the data, assuming Markovian state evolution and uncorrelated state and measurement noise:

$$p(\mathbf{W}, x_{1:N}|\boldsymbol{\theta}) = p(\mathbf{W}_1|\boldsymbol{\theta}) \prod_{t=2}^{N} p(\mathbf{W}_t|\mathbf{W}_{t-1},\boldsymbol{\theta}) \prod_{t=2}^{N} p(d_t|\mathbf{W}_t,\boldsymbol{\theta}) \tag{1.11}$$

It is assumed that the likelihood of the data given the states, and the evolution of the states is Gaussian.

$$p(\mathbf{W}_1|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{q/2}|\boldsymbol{\Sigma}|^{1/2}} exp[-\frac{1}{2}(\mathbf{W}_1 - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{W}_t - \boldsymbol{\mu})] \tag{1.12}$$

$$p(\mathbf{W}_t|\mathbf{W}_{t-1},\boldsymbol{\theta}) = \frac{1}{(2\pi)^{q/2}|\mathbf{Q}|^{1/2}} exp[-\frac{1}{2}(\mathbf{W}_t - \mathbf{W}_{t-1})\mathbf{Q}^{-1}(\mathbf{W}_t - \mathbf{W}_{t-1})] \tag{1.13}$$

$$p(x_t|\mathbf{W}_t,\boldsymbol{\theta}) = \frac{1}{(2\pi)^{m/2}|R|^{1/2}} exp[-\frac{1}{2}(d_t - h(\mathbf{W}_t,x_t))R^{-1}(d_t - h(\mathbf{W}_t,x_t))] \tag{1.14}$$

By taking the log of the likelihood of the complete data we arrive at

$$\begin{aligned} \ln p(\mathbf{W}, x_{1:N}|\boldsymbol{\theta}) = -\sum_{t=1}^{N} \big[\frac{1}{2}(d_t - h(\mathbf{W}_t,x_t))R^{-1}(d_t - h(\mathbf{W}_t,x_t))\big] - \frac{N}{2}\ln|R| - \sum_{t=2}^{N}[\frac{1}{2}(\mathbf{W}_t - \mathbf{W}_{t-1}) \\ \mathbf{Q}^{-1}(\mathbf{W}_t - \mathbf{W}_{t-1})] - \frac{N-1}{2}\ln|\mathbf{Q}| - \frac{1}{2}(\mathbf{W}_1 - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{W}_t - \boldsymbol{\mu}) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| - \frac{N(m+q)}{2}\ln(2\pi) \end{aligned} \tag{1.15}$$

Taking the Expectation of both sides of the equation and differentiating the expected log-likelihood with respect to R^{-1}

$$\begin{aligned} \frac{\partial}{\partial R^{-1}}\mathbb{E}[\ln p(\mathbf{W}, x_{1:N}|\boldsymbol{\theta})] &\approx \frac{1}{2}\frac{\partial}{\partial R^{-1}}(\frac{N}{2}\ln|R^{-1}| - \sum_{t=1}^{N} tr(R^{-1}[\mathbf{j}^T\mathbf{P}_t^N\mathbf{j} + (d_t - h(\hat{\mathbf{W}}_t,x_t))(d_t - h(\hat{\mathbf{W}}_t,x_t))^T])) \\ &= \frac{N}{2} - \sum_{t=1}^{N}\frac{1}{2}(\mathbf{j}^T\mathbf{P}_t^N\mathbf{j} + (d_t - h(\hat{\mathbf{W}}_t,x_t))(d_t - h(\hat{\mathbf{W}}_t,x_t))^T) \end{aligned} \tag{1.16}$$

and setting the resulting solution equal to zero and solving for R results in

$$R = \frac{1}{N}\sum_{t=1}^{N}(\mathbf{j}^T\mathbf{P}_t^N\mathbf{j} + (d_t - h(\hat{\mathbf{W}}_t,x_t))(d_t - h(\hat{\mathbf{W}}_t,x_t))^T) \tag{1.17}$$

Similarly, differentiating with respect to **Q**

$$\frac{\partial}{\partial \mathbf{Q}^{-1}}\mathbb{E}[\ln p(\mathbf{W}, x_{1:N}|\boldsymbol{\theta})] \approx \frac{N-1}{2}\mathbf{Q} - \frac{1}{2}(\mathbf{C} - 2\mathbf{B}^T + \mathbf{A}^T) \tag{1.18}$$

where **Q** is a diagonal matrix. Equating to zero and solving for **Q** leads to

$$\mathbf{Q} = \frac{1}{N-1}(\mathbf{C} - \mathbf{B}\mathbf{A}^{-1}\mathbf{B}^T) \tag{1.19}$$

where we define the quantities

$$
\begin{aligned}
\mathbf{A} &= \sum_{t=1}^{N}(\mathbf{P}_{t+1}^{N} + \hat{\mathbf{W}}_{t-1}^{N}(\hat{\mathbf{W}}_{t-1}^{N})^{T}) \\
\mathbf{B} &= \sum_{t=1}^{N}(\mathbf{P}_{t,t+1}^{N} + \hat{\mathbf{W}}_{t}^{N}(\hat{\mathbf{W}}_{t-1}^{N})^{T}) \\
\mathbf{C} &= \sum_{t=1}^{N}(\mathbf{P}_{t}^{N} + \hat{\mathbf{W}}_{t}^{N}(\hat{\mathbf{W}}_{t}^{N})^{T})
\end{aligned} \tag{1.20}
$$

It is also possible to solve for the initial conditions in the M step, via taking the derivative of the expected log-likelihood with respect to the initial mean

$$\frac{\partial}{\partial \boldsymbol{\mu}}\mathbb{E}[\ln p(\mathbf{W}, x_{1:N}|\boldsymbol{\theta})] \approx \frac{1}{2}\boldsymbol{\Sigma}^{-1}(-2\hat{\mathbf{W}}_{1}^{N} + 2\boldsymbol{\mu}) \tag{1.21}$$

which leads to the initial value of $\boldsymbol{\mu} = \mathbf{W}_{1}^{N}$ and similarly for the covariance, taking the derivative with respect to $\boldsymbol{\Sigma}$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}^{-1}}\mathbb{E}[\ln p(\mathbf{W}, x_{1:N}|\boldsymbol{\theta})] \approx \frac{1}{2}\boldsymbol{\Sigma} - \frac{1}{2}(\hat{\mathbf{W}}_{1}^{N} - \boldsymbol{\mu})(\hat{\mathbf{W}}_{1}^{N} - \boldsymbol{\mu})^{T} + \mathbf{P}_{1}^{N} \tag{1.22}$$

leads to the initial covariance $\boldsymbol{\Sigma} = \mathbf{P}_{1}^{N}$.

1.4.1 The EM steps

The algorithm starts off with an initial guess for $\boldsymbol{\theta}$. Then, in the E-step, the expected values of $\hat{\mathbf{W}}_{t}^{N}$, $\mathbf{P}_{t}^{N}$ and $\mathbf{P}_{t,t-1}^{N}$ are obtained from their current estimates through extended Kalman filtering and smoothing. In the M-step, the new values of $\boldsymbol{\theta}$ are obtained using the above equations.

1.5 Experimental Results

To asses the performance of the proposed model, we have implemented several training algorithms that are similar to the presented EM-RNN, including the RNN-RTRL algorithm [19], and the Extended Kalman Filter trained recurrent neural network(RNN-EKF). Here we consider hourly observations of the $\boldsymbol{D_{st}}$ index, along with observations of the $\boldsymbol{b_x}$, $\boldsymbol{b_y}$, and $\boldsymbol{b_z}$components of the $\boldsymbol{IMF}$ index. The data considered in this study consisted of 9000 data points ranging from January 1, 1980 to February 5, 1981. Three studies were conducted to asses the model's ability to perform during periods of high geomagnetic instability (storm), which corresponds to the dates of March 31 to August 14 1980 for the mild series of storms, and from December 12, 1980 to December 24, 1980 for the severe storm. The quiet period ranged from December 12, 1980 to February 5, 1981. In building the model, we used the period ranging from January 1 1980 to March 31, 1980.

The forecast errors were measured by the root mean squared error (RMSE) computed by $RMSE = ((1/T)\sum_{t=1}^{T}(d_t - y_t)^2)^{1/2}$, where T is the length of the data set. In all simulations, the weights of the networks were initialised with random uniformly distributed weights in the range of $[-2, 2]$. Each of the recurrent networks were initialised with 3 hidden neurons and 3 input neurons for each factor ($\boldsymbol{b_z}$, $\boldsymbol{b}^2$, $\boldsymbol{b_y}^2$,$\boldsymbol{D_{st}}$), resulting in a total input window of size 12. All RNNs had one output neuron corresponding to the one hour ahead value of the $\boldsymbol{D_{st}}$signal. For the EKF trained network, the initial diagonal elements of the covariance matrix of the $[\mathbf{Q}]_{ii}$ and R were set to $1.0e^{-3}$ and $1.0e^{-2}$ respectively, and for the EM trained models, both $[\mathbf{Q}]_{ii}$ and R were set to 100. The learning rate for the RTRL algorithm was set to $.05$.

1.5.1 Forecasting $\boldsymbol{D_{st}}$

Table 1.1 summarises the experimental results of single step ahead model fitting and prediction of the storms. The in sample performance of EM and the EKF were quite similar, and the gradient descent trained RNN had the worst fit. In both the severe and mild storm forecasting tasks, the EM-RNN outperforms the other recurrent network training algorithms including the recurrent network trained by RTRL and the EKF. However, in the quiet period, the EM-RNN and the RNN-EKF perform similarly. The plots of the experiments are given in Figure 1.2. The use of the EM algorithm has led to improvements in out of sample predictions over the EKF, however, the EM-RNN more closely approximates the target series than the remaining algorithms as shown in the given plots.

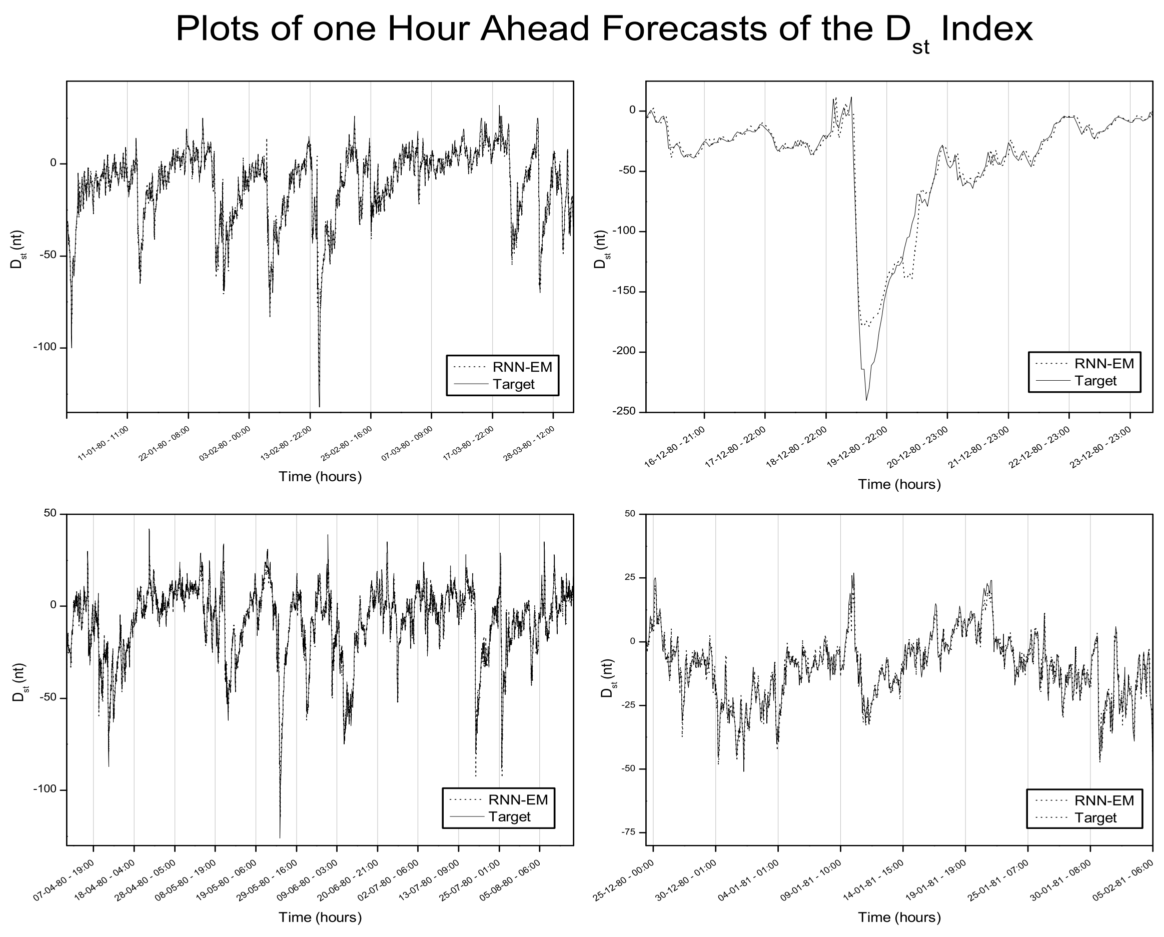

Fig. 1.2. Plots of the EM-RNN and D_{st} measurements. The top left plot is of the in sample fitting of model to the data. The lower left plot shows out of sample forecast performance on a series of mild storms. The lower right hand plot shows the model performance during a quiet period (no storm), and in the top right plot, the model performance on the severe storm is provided.

Table 1.1. Numerical Performance of the Studied Algorithms on One Hour Ahead Forecasts of the D_{st} index

Model	RMSE(*training*)	RMSE(*severe*)	RMSE(*mild*)	RMSE(*quiet*)
RNN-RTRL	9.64	21.17	11.83	12.78
RNN-EKF	3.57	16.625	4.04	3.16
RNN-EM	3.41	11.78	3.65	3.24

1.6 Conclusion

This paper presented a probabilistic training algorithm for dynamic recurrent neural models suitable for modelling the D_{st} index. It has demonstrated that the EM training of the RNN has lead to improvements in prediction accuracy when processing the D_{st} for one hour ahead forecasts. The presented results are encouraging as they show that EM-RNN has the capacity to accurately model complex Geo-magnetic phenomena.

References

1. Dungey, J.W. (2000). Interplanetary magnetic fi eld and the auroral zones.*Phys. Rev. Lett.*, 26 , 47–48
2. Axford, W.I. and Hines, C.O.(1961). A unifying theory of high-latitude geophysical phenomena and geomagnetic storms. *Can. J. Phys.*,39, 1433.
3. Gonzales, W. D., J. A. Joselyn, Y. Kamide, H. W. Kroehl, G. Rostoker, B. T. Tsurutani, and V. M. Vasyliunas (1994). What is a geomagnetic storm?. *Journal of Geophysical Research*,99, 5771–5792.
4. Gosling, J. T. and McComas, D. J. and Phillips, J. L. and Bame, S. J.(1991). Geomagnetic activity associated with earth passage of interplanetary shock disturbances and coronal mass ejections. *Journal of Geophysical Research*, 96,7831-7839.
5. Farrugia, C. J. and Freeman, M. P. and Burlaga, L. F. and Lepping, R. P. and Takahashi, K. (1993). The earth's magnetosphere under continued forcing - Substorm activity during the passage of an interplanetary magnetic cloud. *Journal of Geophysical Research*, 98,7657-7671.
6. Lindsay G. M. ; Russell C. T. ; Luhmann J. G.(1995). Coronal mass ejection and stream interaction region characteristics and their potential geomagnetic effectiveness. *Journal of Geophysical Research*, 100,16999-17013.
7. Tsurutani, B. T., W. D. Gonzalez, F. Tang, S. I. Akasofu and E. J. Smith (1988). Origin of interplanetary southward magnetic fi elds responsible for major geomagnetic storms near solar maximum. *Journal of Geophysical Research*, 93, 8519-8531.
8. Freeman, J., A. Natal, P. Reiff, W. Denig, S. Gussenhoven-Shea, M. Heinemann, F. Rich, and M. Hairston (1993) The use of neural networks to predict magnetospheric parameters for input to a magnetospheric forecast model. *In: Proceedings of Artifi cial Intelligence Applications in Solar-Terrestrial Physics Workshop*, 167-181.
9. Lundstedt, H. (1992). Neural Networks and prediction of solar-terrestrial effects. *Planet Space Science*, 40, 457.
10. Lundstedt, H. and P. Wintoft (1994). Prediction of geomagnetic storms from solar wind data with the use of a neural network. *Planet Space Science, Ann Geophys.*, 12, 19–24.
11. J.-G. Wu, and Lundstedt H. (1997). Geomagnetic storm predictions from solar wind data with the use of dynamic neural networks. *Journal of Geophysical Research*, 102(A7), 14,255 - 14,268.
12. H. Lundstedt, H. Gleisner, P. Wintoft (2002). Operational forecasts of the geomagnetic Dst index. *Geophysical Research Letters*, 29(24), 2181.
13. Pallocchia, G. and Amata, E. and Consolini, G. and Marcucci, M. F. and Bertello, I.(2006). Geomagnetic Dst index forecast based on IMF data only. *Ann Geophys*, 24, 989–999.
14. A. M. Schafer and G. Zimmerman (2007). Recurrent Neural Networks are Universal Approximators. *Int J Neural Syst*, 7(4), 253–263.
15. B. Schottky and D. Saad (1999). Statistical mechanics of EKF learning in neural networks. *J. Phys. A*, 32(9), 1605–1621.
16. R. H. Shumway, D. S. Stoffer (1982). An Approach To Time Series Smoothing and Forecasting Using the EM Algorithm. *J Time Ser Anal*, 3(4), 253–264.
17. J.F.G. de Freitas, M. Niranjan, and A.H. Gee(2000). Dynamic Learning with the EM Algorithm for Neural Networks. *J Vlsi Signal Proc*, 26, 119–131
18. S. Roweis and Z. Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Computation*, 11(2), 305–345.
19. R.J. Williams and D. Zipser (1989). A learning algorithm for continuously running fully connected recurrent neural networks. *Neural Computation*, 1, 270–280.
20. G. V. Puskorius and L. A. Feldkamp (1994). Neurocontrol of nonlinear dynamical systems with Kalman fi lter trained recurrent networks. *IEEE T Neural Networks*, 5(2), 279–297.
21. H.E. Rauch F. Tung, and C.T. Striebel (1965). Maximum Likelihood Estimates of Linear Dynamic Models. *AIAA Journal*, 3(8), 1445–1450.
22. A. Krok, Z. Waszczcyzyn. (2007). Kalman fi ltering for neural prediction of response spectra form mining tremors. *Comput. Struct.*, 85, 1257–1263.
23. W. L. Mao (2008). Novel SREKF-based recurrent neural predictor for narrowband/FM inference rejection in GPS. *International Journal of Electronics and Communications*, 62, 216-222.
24. K. Nouri and R. Dhaouadi and N. B. Braiek (2008). Adaptive control of a nonlinear dc motor drive using recurrent neural networks. *Applied Soft Computing*, 8(1), 371–382.
25. D. V. Prokhorov (2008). Toyota Prius HEV neurocontrol and diagnostics. *Neural Networks*, 21, 458–465.
26. J. d. J. Rubio and W. Yu (2007). Nonlinear system identifi cation with recurrent neural networks and dead-zone Kalman fi lter algorithm. *Neurocomputing*, 70, 2460–2466.
27. Simon s. Haykin (2001) Kalman Filtering and Neural Networks. John Wiley & son, New York.