The use of sequential recurrent neural filters in forecasting the $D_{st}$ index for the strong magnetic storm of autumn 2003

Lahcen Ouarbya *, Derrick Takeshi Mirikitani, Eamonn Martin

Department of Computer Science, Goldsmiths College, University of London, New Cross, London, London SE14 6NW, UK

**A R T I C L E  I N F O**

Article history:
Received 23 June 2010
Received in revised form 1 December 2011
Accepted 5 December 2011

Keywords:
Forecasting magnetospheric disturbances $D_{st}$ Recurrent neural networks

**A B S T R A C T**

Neural based geomagnetic forecasting literature has heavily relied upon non-sequential algorithms for estimation of model parameters. This paper proposes sequential Bayesian recurrent neural filters for online forecasting of the $D_{st}$ index. Online updating of the RNN parameters allows for newly arrived observations to be included into the model. The online RNN filters are compared to two (non-sequentially trained) models on a severe double storm that has so far been difficult to forecast. It is shown that the proposed models can significantly reduce forecast errors over non-sequentially trained recurrent neural models.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

It has been well established that changes in the Sun’s magnetic field influence the structure of the magnetic field surrounding the earth (magnetosphere) [1–3]. The solar wind expands the reach of the Sun’s magnetic field to form what is known as the Interplanetary Magnetic Field (IMF). The IMF can cause energetic particles to be injected into the Earth’s magnetic field, resulting in magnetospheric disturbances. When a sufficient and prolonged transfer of energy from the solar wind opposes the Earth’s magnetic field, a magnetospheric storm occurs [3]. Geomagnetic storms can have many negative effects on technical systems on Earth, such as disruptions in power generation, transmission, and distribution.

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 the $D_{st}$ index. Numerous studies have shown a correlation between the value of the $D_{st}$ index and the magnetic storm’s intensity [4,5]. The physical interaction (transfer of mass, energy and momentum) between the IMF and the magnetosphere takes place at the magnetopause boundary. Neural networks have often been used for $D_{st}$ forecasting due their ability to learn from examples, and that they do not need an exact mathematical model. The most successful neural models to forecast the $D_{st}$ index have been based on recurrent neural networks (RNNs) [6,7]. Recurrent neural networks were found to uncover some of the relationships between the IMF and $D_{st}$. Recent research has focused on exploring the effects of variations or transformations on the input factors to the RNN [8] to reduce forecast errors.

Most previous RNN based $D_{st}$ forecasting research [6,8] has relied on non-sequential algorithms for training the RNN. Non-sequential training algorithms are unable to incorporate newly arrived information into the model parameters without re-processing the entire data set (which is a costly operation). Furthermore, the training algorithms used thus far for training the RNNs on $D_{st}$ forecasting (all of which have been non-sequential first order gradient descent) are known to be susceptible to local minima, and uncertain convergence. This has been a bottleneck in the area (and may stifle future progress in neural...
based forecasting of geomagnetic phenomena). Also performing models are difficult to obtain, and new events can not readily be incorporated into the model for improved forecasts.

In this paper we investigate solutions to this problem through the use of the sequential Bayesian framework of which nonlinear Kalman filters are utilized for RNN training [9]. The advantage of our approach is a framework based on second-order [10] online estimation of model parameters, resulting in fast convergence and accurate forecasts. The main results of the paper are as follows: (1) an efficient framework to reliably obtain RNN parameters for \( D_{st} \) forecasts, (2) the ability to sequentially incorporate new measurements into the model, (3) improved forecast accuracy over previously demonstrated results.

2. Recurrent neural networks

The recurrent architecture chosen for this study is known as the Williams and Zipser fully recurrent network [11] as shown in Fig. 1. The Williams and Zipser fully recurrent network is the most general recurrent network in that each neuron is fully connected to every other neuron, which has the most interconnections allowing for greater memory depth and modeling potential. We adopt the following notation to describe the fully recurrent network: \( s_t = [c_t, x_t, b] \) is the input vector for each neuron, which contains a vector of previous activations of each neuron \( c_t \), the exogenous inputs to the network \( x_t \) (defined in Section 5), and the bias \( b \). The context vector \( c_t = [c^{(1)}, c^{(2)}, \ldots, c^{(H)}] \) are the activations of the network at the previous time step. The \( i \)th neuron weight vector is \( v^{(i)}_t = [w^{(i,1)}, \ldots, w^{(i,H+I+1)}] \), and the overall network weight vector is

\[
\mathbf{w}_t = [v^{(1)}, \ldots, v^{(H)}],
\]

where \( H \) is the total number of neurons and the total number of weights is \( L = H^2 + (I + 1) \times H \), and \( I \) is the length of \( x_t \). The activation function \( g(\cdot) \) is a logistic sigmoidal nonlinearity \( g(a) = 1/(1 + \exp(-a)) \) which maps the input \( a \in \mathbb{R} \) into a bounded interval \( \Omega = (0, 1) \) of length \( |\Omega| = 1 \) where \( \Omega \subset \mathbb{R} \).

The fully recurrent neural network architecture consists of a single layer of processing neurons (known as the processing layer) which are fully connected to each other. Each neuron computes a weighted sum of the previous processing layer activations, along with the exogenous input to the network, and the bias given by

\[
u^{(i)}_t = \mathbf{v}^{(i)}_t \mathbf{s}^T_t,
\]

where \( T \) denotes the transpose operator. Each weighted sum \( u^{(i)}_t \) is then passed through the nonlinear activation function to produce the activation outputs

\[
y^{(i)}_t = g(u^{(i)}_t),
\]

where the output of the network is \( y^{(1)} \). The overall functionality of the RNN is represented as a function \( h(t, \mathbf{w}_t, \mathbf{x}_t) \) and the output of the RNN is related to the overall functional description of the RNN as follows \( y^{(1)}_t = h(t, \mathbf{w}_t, \mathbf{x}_t) \). To simplify the notation, we drop the superscript when referring to the outputs of the RNN, i.e. \( y^{(1)}_t = y_t \).
2.1. State space modeling with RNNs

Weight estimation of the RNN can be formulated in a sequential Bayesian filtering framework: given a hidden state represented by RNN weights and a noise contaminated measurement, the task is to re-estimate the weights so as to factor in the newly arrived information. The weights in the recurrent neural network, \( \mathbf{w} \in \mathbb{R}^d \), are considered as the discretized state of the system. The RNN weights \( \mathbf{w} \) are treated as a random vector whose time evolution is specified by the following nonlinear discrete time state space model

\[
\begin{align*}
\mathbf{w}_t & = \mathbf{w}_{t-1} + \omega_{t-1} \quad (a) \\
\mathbf{d}_t & = h(t, \mathbf{w}_t, \mathbf{x}_t) + \nu_t \quad (b),
\end{align*}
\]

where (a) and (b) are the process and measurement equations respectively. \( \omega_t \in \mathbb{R}^d \) represents a stochastic perturbation assumed to be an i.i.d. (independent and identically distributed) Gaussian process with zero mean and covariance \( Q \). Where (a) and (b) are the process and measurement equations respectively.

In the sequential filtering framework, it is assumed that past information \( p(\mathbf{w}_{t-1} | \mathbf{d}_{t-1}) \) is available and can be used to find two quantities of interest: \( p(\mathbf{w}_t | \mathbf{d}_{t-1}) \), the forecast (prior) distribution and \( p(\mathbf{w}_t | \mathbf{d}_t) \), the analysis (posterior) distribution, where \( \mathbf{d}_t = [d_1, \ldots, d_t] \). The forecast distribution is specified via the integral

\[
p(\mathbf{w}_t | \mathbf{d}_{t-1}) = \int p(\mathbf{w}_t | \mathbf{w}_{t-1}) p(\mathbf{w}_{t-1} | \mathbf{d}_{t-1}) d\mathbf{w}_{t-1}.
\]

The posterior distribution is filtered using the Bayes rule, which combines the prior information \( p(\mathbf{w}_t | \mathbf{d}_{t-1}) \) with the most recently observed information \( p(\mathbf{d}_t | \mathbf{w}_t) \) to compute the analysis distribution

\[
p(\mathbf{w}_t | \mathbf{d}_t) \approx p(\mathbf{d}_t | \mathbf{w}_t) p(\mathbf{w}_t | \mathbf{d}_{t-1}).
\]

A sequential estimation of the two distributions is achieved through iteration of this cycle at each time step.

3. EKF Training of the RNN

For RNN training, the state space equations are highly nonlinear. The Extended Kalman filter (EKF) applies the Kalman filter framework to nonlinear Gaussian systems by first linearizing the dynamic state space equation using a first-order truncated Taylor series expansion around the current estimates [12]. The real time recurrent learning algorithm [11] was used for the linearization of the RNN through computation of the Jacobian matrix \( \mathbf{j}_t = \frac{\partial h(t, \cdot, \cdot)}{\partial \mathbf{w}_t} \) consisting of partial derivatives of the output \( y_t = h(\cdot, \cdot, \cdot) \) with respect to the weights of the network [13]. The Jacobian \( \mathbf{j}_t \) is evaluated at each time step. EKF filtering for RNN weight estimation leads to faster convergence [10] than gradient based algorithms, and also may resolve issues with vanishing gradients [12]. For neural networks, the EKF solution to the parameter estimation problem is given by the following recursion

\[
\begin{align*}
\mathbf{g}_t & = \mathbf{P}_t \mathbf{j}_t [R + \mathbf{j}_t^T \mathbf{P}_t \mathbf{j}_t]^{-1} \\
\hat{\mathbf{w}}_{t+1} & = \hat{\mathbf{w}}_t + \mathbf{g}_t (d_t - h(t, \hat{\mathbf{w}}_t, \mathbf{x}_t)) \\
\mathbf{P}_{t+1} & = \mathbf{P}_t - \mathbf{g}_t \mathbf{j}_t^T \mathbf{P}_t + \mathbf{Q}.
\end{align*}
\]

where \( R \) and \( Q \) are the measurement and process noise covariance matrices. Since the EKF is a suboptimal estimator based on linearization of a nonlinear mapping; \( \mathbf{w} \) is only an approximation of the expectation, \( \mathbf{P} \) is an approximation of the state covariance, and the matrix \( \mathbf{g} \) is the Kalman gain. It is well known that the EKF may experience instabilities as a result of this approximation, especially in situations of high nonlinearity [12].

4. Unscented Kalman filter (UKF training of the RNN)

The shortcomings of EKF [12] have led many researchers to develop a number of closely related Gaussian approximate filters based on novel deterministic sampling methods which propagate Gaussian random variables through nonlinear transforms. The authors in [14] have introduced the Unscented Kalman filter (UKF). Unlike the EKF, the UKF estimates the mean and covariance through a weighted sum called the unscented transform (UT).

The weights used for the calculation of the posterior mean and covariance are defined as follows

\[
\begin{align*}
\alpha^{-} & = \frac{\lambda}{L + \lambda} + (1 - \alpha^2 + \beta) \\
\alpha^{+} & = \frac{1}{2(L + \lambda)} \quad i = 1, \ldots, 2L,
\end{align*}
\]
where the superscript operator $a^{(i)}$ represents the $i$th element in the vector $a$. The parameter $\lambda = \alpha^2 (L + k) - L$ represents a scaling parameter where $L$ is the length of the state vector and the value of $\alpha = 1$. The parameter $k$ represents a secondary scaling parameter, which was set to $3 - L$. The parameter $\beta = 1$ represents information about prior knowledge of the distribution of $w$.

After each iteration of the UKF, the sigma points are calculated as follows

$$ I_t = (L + \lambda)(P_t + Q) \quad (8) $$

$$ \phi^{(i)}_t = \left[ \mathbf{w}^{(0)}_t, (\mathbf{w}_t + \sqrt{T_t} \phi^{(1)}_t, (\mathbf{w}_t - \sqrt{T_t} \phi^{(2L)}_t, \right] \quad (9) $$

$$ \delta^{(i)}_t = h(t, \phi^{(i)}_t, x_t) \quad y_t = h(t, \mathbf{w}_t, x_t). \quad (10) $$

where $P_t$ and $Q$ represent an $L$-by-$L$ approximate error covariance and process noise covariance matrices, respectively.

The RNN–UKF weight vector is then updated online as follows

- The filtered measurement variance and the cross covariance between the state and measurement are computed as follows

$$ p_{yy}^t = \sum_{i=0}^{2L} \phi^{(i)}_t (\delta^{(i)}_t - y_t)(\delta^{(i)}_t - y_t)^T + R \quad (11) $$

$$ p_{vy}^t = \sum_{i=0}^{2L} \phi^{(i)}_t (\delta^{(i)}_t - \mathbf{w}_t)(\delta^{(i)}_t - y_t)^T. $$

- The gain matrix, the filtered state estimate and the error covariance are computed as follows

$$ g_t = p_{vy}^t (p_{yy}^t + R)^{−1} $$

$$ \mathbf{w}_{t+1} = \mathbf{w}_t + g_t (d_t - h(t, \mathbf{w}_t, x_t)) $$

$$ P_{t+1} = P_t - g_t p_{vy}^t g_t^T. \quad (12) $$

5. Experimental results

There have been two main studies on RNN forecasting of the $D_s$ index [15,8]. The two papers differ mainly on the inputs to the model. In [8] the authors have used IMF components only whereas in [15] the authors used both IMF and solar wind data to predict the $D_s$ index. Although the inputs to the models differ, the training algorithms for parameter estimation were essentially the same (i.e. both authors used non-sequential gradient descent to train the RNN). As this paper proposes online trained RNN models, a comparison is made between these “batch” trained models and the proposed online algorithms.

Here Edda refers to the algorithm proposed in [8]. The input vector for the models with Edda based inputs are defined as $x_t = [x_t^{(1)}, x_t^{(2)}, x_t^{(3)}]$ where the input variables were scaled as follows $x_t^{(1)} = (b_t)/44.7$, $x_t^{(2)} = b_t/2323.24$, and $x_t^{(3)} = (b_t/1176.49$. The output of the normalized $R$ was as follows $y_t = [87, \ldots, 100]$. The variables $a$ and $w$ represent the density and velocity of the solar wind. The outputs of the Lund input based RNN algorithms were scaled by $150y_t - 100$.

As the proposed models operate online, multiple passes over a large training set are not necessary. Only a small set of data consisting of the last 240 hourly observations before 25/10/2003 at 7:00 h were used to “prime” the networks before measuring the errors of the predictions. In the “priming” phase, only one pass over the data set was performed. Forecast errors were then measured from the dates of 25/10/2003 at 7:00 h to 4/11/2003 at 11:00 h.

Most of the previous algorithms proposed for $D_s$ forecasting [8] have had difficulty in modeling double storms [16]. To evaluate the performance of the proposed algorithms, we compare the forecast results to two widely known forecast models [15,8] on one hour ahead forecasting of a severe double storm. In all simulations, the weights of the networks were initialized with random uniformly distributed weights in the range of [−0.5, 0.5]. Each of the Kalman trained recurrent networks were initialized with 3 hidden neurons. All RNNs had one output neuron corresponding to the one hour ahead value of the $D_s$ signal. In all filters, the initial diagonal elements of the covariance matrix $\{Q\}_t$ for the RNN–EKF and the RNN–UKF were set to $1.0e^{−4}$, and the hyper-parameter $R$ for all models were set to $1.0e^{−4}$. Fig. 2 illustrates the performance of the online RNN models, the RNN–EKF and the RNN–UKF, using the inputs proposed in [15] for the period starting from October the 25th at 7:00 h to the 4th of November at 11:00 h, 2003. Also included are the predictions of the Fig. 3, providing the predictions from the non-sequentially trained RNN model of Lundstedt [15]. Similarly, Fig. 3 provides the predictions from the online trained RNN models (RNN–EKF and the RNN–UKF) using the inputs proposed in [8]. The predictions of the non-sequentially trained RNN model of [8] are included for comparison. The figures show that the proposed online algorithms outperform the non-sequential algorithms proposed in previous work [15,8]. The non-sequential algorithms are not dynamically updated and this has resulted in poor forecast performance. From these simulations it is clear that online
A comparison of one-hour ahead $D_e$ predictions of the online RNNs trained with the EKF and UKF, using Lund inputs. The predictions of the proposed models are shown in comparison with the outputs of the non-sequential trained Lund model from 25-Oct-2003 to 4-Nov-2004.

Fig. 3. A comparison of one-hour ahead $D_e$ predictions of the online RNNs trained with the EKF and UKF, using Edda inputs. The predictions of the proposed models are shown in comparison with the outputs of the non-sequential trained Edda model from 25-Oct-2003 to 4-Nov-2004.

Table 1 presents the model residuals reported in RMSE over the out-of-sample data set. For both the EDDA and Lund input vectors, the RNN–EKF had the larger forecast errors than the RNN–UKF. The higher errors from RNN–EKF modeling are mostly likely due to filter divergence during periods of high non-linearity. The RNN–UKF does not rely on approximate linearization schemes which seem to have allowed the filter to model the $D_e$ index more closely. However, the performances of both online models (in terms of RMSE) outperform the non-sequentially trained models put forth by [15] and by [8].

Table 1

<table>
<thead>
<tr>
<th></th>
<th>Non-sequential</th>
<th>RNN–EKF</th>
<th>RNN–UKF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edda Input</td>
<td>34.04</td>
<td>20.69</td>
<td>2.86</td>
</tr>
<tr>
<td>Lund Input</td>
<td>58.59</td>
<td>31.43</td>
<td>6.42</td>
</tr>
</tbody>
</table>
6. Concluding remarks

This paper introduced a framework for recursive estimation of RNN parameters for modeling geomagnetic activity. Through a numerical comparison on a severe double storm between online trained models proposed in this paper and existing non-sequentially trained models [17,8], we have observed a significant increase in prediction accuracy through online training of RNNs. This can be attributed to the recursive updating of the RNN weights which incorporates new information into the model each time an observation arrives. Other benefits of the sequential approach include reduced effort in training the RNN before forecasting begins (only one pass over a small data set was needed for training). Future work will focus on finding parsimonious recurrent neural models for possibly further improved forecasts.

References