A Neural Network Algorithm with Weight Decay for 2-6 Hour Ahead Wind Speed Forecasting

Colin Rickert
University of Vermont
Dept of Computer Science
crickert@uvm.edu

1 Abstract
A single layer feed forward neural network algorithm using back propagation, gradient descent and weight decay is proposed for the purpose of wind speed forecasting using only the observed hourly wind speeds, directions, temperatures and pressures observed at a single site. The site data used for this experiment was 10 years worth of hourly ASOS data from the Bismarck North Dakota Regional Airport gathered from the NOAA ftp site which is available for users upon subscription. The experiment was coded in the R programming language using the package implementation nnet which implements a single layer feed forward neural network with an optional weight decay factor.

2 Introduction
Bismarck North Dakota was chosen due to the persistence and intensity of the wind in that region of the United States. The data comes in at an hourly interval and contains the current wind speed in mph, direction (in degrees from true north), pressure (psi), temperature (fahrenheit) and the current time among others. A training set containing 5 vectors for each of the previous 5 hours of the features mentioned above was used for the training of the neural network. The network used a gradient descent energy minimization function designed to minimize the training error between the correct bin of the true wind speed in the lookahead interval and the predicted bin for that interval across all observations. This minimization process involves the updating of weights between the nodes in the network by incrementing those weights in the direction of the steepest gradient of the energy function. The network was trained using a 3-5 hour window of observations containing the aforementioned features which was then used to predict a discrete output value representing the wind speed in the 2, 4, 6 and 8 hour lookahead periods.
3 Related Work

Forecasting wind is challenging due to the numerous causal parameters such as surface roughness, pressure, temperature, direction and the interactions between them [1]. Wind forecasting at the mesoscale (i.e. upper atmosphere) generally requires numerical weather prediction techniques (NWP) which in turn requires extensive atmospheric modeling and a large amount of computational resources. According to some estimates this process at best takes roughly 5 hours of data gathering and computation [2]. However, when predicting wind at or near the surface of the earth, numerous interactions take place between the boundary layer and upper atmosphere wind which can transform the wind into a variety of substreams as it approaches the surface of the earth and can result in a large amount error in the NWP model [3]. It is well known that the wind resource contains definite patterns particularly with respect to diurnal and synoptic variations as well as coriolis effects, all of which are dependent on temperature, pressure and time [3]. In addition there is turbulence which is essentially the randomness or deviation from a base wind speed [3]. These fluctuations typically have a zero mean when averaged over 10 minutes [3]. Turbulence has two main causes: the first being the effects of hills and mountains over which the upper atmospheric air must flow and the second being mixing of warmer and cooler air flows [3]. Such complex interactions may be too subtle for discernment in a classical numerical weather prediction model [5] but may be detectable via an artificial neural network. Furthermore, a distinct spectrum of frequencies exist at many locations with detectable frequencies ranging from 4 to 24 hours [3] which suggests that the models used for predicting short term wind speeds may have different parameters from those that predict over a longer range. In fact a wide range of AI based techniques have been employed to model this problem including Feed Forward Neural Networks [2], Markov Models [2], Kalman Filters [2], Paricle Swarm Optimization [4], Recurrent Neural Networks [5] as well as Fuzzy Logic [5] and the combination of a Fuzzy Classifier with a Temporal Neural Network [1].

Artificial Neural Networks have been shown to have great promise in the short term forecasting of wind in the 1-8 hour lookahead time predictions due in part to the fact that wind speeds have been shown to be highly auto-correlated (i.e. cyclic) in this time range [6]. Both feed forward and recurrent architectures have been shown to have great merit. Most recently Goh, Chen, Popovic, Aihara, Obradovic and Mandic demonstrated the efficacy of Fully Complex Recurrent Neural Networks (FCRNN’s) and Complex Pipelined Recurrent Networks (CPRN’s) for the 2-6 hour ahead prediction of wind speeds [6]. Of particular interest is the FCRNN where the recurrences are built between the output/prediction of the input vector \( i \) for time \( t+1 \) of the network and the input vector \( i+1 \) which is used to predict time \( t+2 \). In this manner, one can predict well into the future (albeit with less and less precision) using a single trained network [6]. Additionally they also discovered that treating speed and direction as a complex number rather than two component vector was advantageous as it seemed to encapsulate more information for the network [6]. However neither
of these approaches utilized the full set of atmospheric variables (i.e. they did not include pressure, temperature and time of day) which have an influence on the wind speed from a meteorological perspective [3].

Other work has been done using artificial neural networks to determine spatial correlation between multiple sites [7]. Such efforts revolve around training a neural network to infer the delay or lag times between multiple sites and using this information to fine tune a prediction. The determination of the lag time however is a non-trivial process due to surface roughness factors between sites that would preclude simple linear calculation (which is why neural networks were recruited for this task). Furthermore, finding appropriate sites that have enough upstream data that is readily available and whose primary wind direction is pointed in the general direction of the site of interest is another drawback to this approach: in other words such data may simply not exist. Therefore, what remains to be determined is whether a general purpose time series prediction model can be created that takes in all of the locally available standard atmospheric data (speed, direction, pressure, temperature, time of day) into one accurate and coherent model for a single site. The focus of this paper is to construct such a model.

4 Theory

Consider a training set \( X_m = \{ (x^{(1)}, t^{(1)}), ..., (x^{(m)}, t^{(m)}) \} \) of \( p \) patterns with each \( x^{(k)} \in \mathbb{R}^{n_0} \) and \( t^{(k)} \in \mathbb{R}^{n_L} \). The neural network will then attempt to find a set of weights \( W = \{ w^{(1)}_{1,0}, ..., w^{(L)}_{n_L, n_L-1} \} \) that optimally reduces the least mean square (LMS) objective energy function:

\[
E(W) = \sum_{p=1}^{n_L} E_p(W) \quad (1)
\]

where

\[
E_p(W) = \frac{1}{2} \sum_{r=1}^{n_L} (t^{(p)}_r - y^{(L)}_r(x^{(p)}, W))^2 \quad (2)
\]

The formula \( y^{(L)}_r(x^{(p)}, W) \) represents the output of the \( r \)’th unit in the final layer \( L \). A general formula for the output of node \( i \) in layer \( l \) in a feed forward neural network with \( L \) layers that maps from \( \mathbb{R}^{n_0} \rightarrow \mathbb{R}^{n_L} \) with \( n_0 \) real inputs and \( n_L \) output units is:

\[
y^{(l)}_i = \sigma \left( \sum_{j=1}^{n_{l-1}} w^{(l)}_{i,j} y^{(l-1)}_j + w^{(l)}_{i,0} \right) \quad (3)
\]

for \( i = 1, 2, ..., n_L \), where

- \( y^{(0)}_i = x_i \), the \( i \)'th element of the input vector \( x \)
The number of units in layer L

The output value of unit i in layer l

The synaptic weight of unit i in layer l that is applied to the output of unit j from layer l-1

The internal bias of unit i in layer l

A sigmoidal function that is required to have a definite integral on the range \([-\infty, +\infty]\).

A feed forward neural network with backpropagation will attempt to reduce the LMS error above using a gradient descent strategy where the weights in the neural network are updated at each iteration in the opposite direction of the gradient of the energy function with respect to the weights:

\[ w_{i,j}^{(l)}(t+1) = w_{i,j}^{(l)}(t) - \eta \frac{\partial E(W)}{\partial w_{i,j}^{(l)}} \]  

(4)

Where \( \eta \) is known as the step size and is often set to 1 though may need to be adjusted in case the algorithm converges too slowly or is overshooting the minimum. The partial derivative \( \frac{\partial E(W)}{\partial w_{i,j}^{(l)}} \) is known to have the following recursive relationship [9]:

\[ \frac{\partial E(W)}{\partial w_{r,s}^{(l)}} = -\Delta_r^{(l)} y_s^{(l-1)} \]  

(5)

where

\[ \Delta_r^{(l)} = \sigma'(S_r^{(l)}) \sum_{i=1}^{n_{i+1}} \Delta_i^{(l+1)} w_{i,r}^{(l+1)} \]  

(6)

if \( l > L \) else

\[ \Delta_r^{(L)} = [(t_r^{(p)} - y_r^{(L)}) \sigma'(S_r^{L})] \]  

(7)

and

\[ S_r^{(l)} = \sum_{s=1}^{n_{i-1}} w_{r,s} y_s^{(l-1)} + w_{s,0}^{(0)} \]  

(8)

In addition to the general framework outlined above a weight decay function was used with the nnet package in R. The use of a weight decay function changes the general form of the error function to a “ridge regression” equation:

\[ y_i^{(l)} = \frac{1}{2} \sum_{r=1}^{n_L} (t_r^{(p)} - y_r^{(L)}(x^{(p)}, W))^2 + \gamma \sum_{r,t,j,l} (w_{i,j,r}^{(l)})^2 \]  

(9)

This new definition of the “energy” of the system is a justifiable measure against overfitting and bias in a neural network. Unnecessarily large weights in
the hidden layers will often cause the output function to be jagged and possibly discontinuous especially when the data itself is noisy [8]. Because wind data is exceptionally noisy it is prone to overfitting which must be accounted for in the model. Thus, for this experiment the weight decay constant $\gamma$ was set to 0.1 after some trial and error. It should be noted that when this parameter was left out of the model it would inevitably overfit and converge prematurely.

5 Methods

The nnet package (available with all R installations) provides a generic API for a single layer feed forward neural network algorithm with several input parameters including:

- **X**: Matrix or data frame of x values for training examples.
- **Y**: Matrix or data frame of target values.
- **Size**: Number of units in the hidden layer.
- **Maxit**: Maximum number of iterations (if the network does not converge).
- **Decay**: Parameter for weight decay

For the purposes of this experiment, the input vectors of wind speed, direction, temperature, pressure and time of the day for the previous 3, 5 and 7 hours were used for training and classification of the feed forward neural network for the 2 hour, 4 hour and 6 hour lookahead models respectively. It was assumed that larger lookahead periods required larger windows of recent observations since the periodicity of wind speeds may increase as you predict further into the future and thus may have a greater dependence on the the more distant past than for shorter lookahead periods. For the target vectors, a binary string representing the correct bin of 45 possible wind speeds bins for the 2, 4, 6 and 8 hour lookahead periods was used as the target class for the training step. The bins were divided evenly between 0 and 60 mph. The justification for this was to reduce overfitting of the neural network since lumping various wind speeds into the same bins results in a reduction of the problem space without a large loss in predictive accuracy. Hourly ASOS data from 01-01-1994 00:00 EST, to 04-18-2006 18:51:00 EST and corresponding 2-6 hour target speeds were loaded into the data matrix and target variables respectively. After a few trial runs the number of hidden units in the single hidden layer was set to 50 for each model though further research could be done to optimize this number. Lastly, the models were run on a Toshiba Satellite laptop computer with an Intel Celeron Processor running at 1.3 GHz with 240 MB of RAM and took between 6-14 hours to complete. It should be noted however that once a model was constructed its predictive accuracy was valid throughout the predictive range (i.e. the models did not need to be reconstructed after each new observation for testing).
6 Results

Tables 1 and 2 contain the Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) values of the predictions of a neural network trained on the Bismarck ND ASOS data between 03-18-2006 to 04-18-2006. The predictions were for the period of data from 04-18-2006 to 05-04-2006. The metrics used to measure the quality of the model were Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Additionally, for comparison the RMSE and AbsME measurements for the well-known persistence model was calculated on the same region of test data as the neural network and the results of these comparisons are displayed in tables 1 & 2. The persistence model simply takes the last observed wind speed and projects it into the future at the desired time interval. The persistence model may seem trivial but it is surprisingly accurate especially during short look ahead periods. In all cases the ANN significantly outperformed the persistence method (figures 1-3). All of the models were constructed from the data mined between 04-18-2006 to 05-04-2006 and there was a distinct partition created between the training and test data. Each prediction (for a given 2, 4 or 6 hour model) was based on the same model constructed from the same data in the training period. Therefore the model was not updated with each new observation after each new prediction which indicates that the model generalized well and did not decay rapidly with time.

<table>
<thead>
<tr>
<th>Lookahead</th>
<th>ANN RMSE</th>
<th>Persistence RMSE</th>
<th>% improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>2hr</td>
<td>2.23</td>
<td>4.51</td>
<td>50.5%</td>
</tr>
<tr>
<td>4hr</td>
<td>3.92</td>
<td>5.62</td>
<td>30.2%</td>
</tr>
<tr>
<td>6hr</td>
<td>5.21</td>
<td>6.36</td>
<td>18.1%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lookahead</th>
<th>ANN MAE</th>
<th>Persistence MAE</th>
<th>% improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>2hr</td>
<td>1.68</td>
<td>3.34</td>
<td>49.7%</td>
</tr>
<tr>
<td>4hr</td>
<td>3.03</td>
<td>4.28</td>
<td>29.2%</td>
</tr>
<tr>
<td>6hr</td>
<td>3.98</td>
<td>4.95</td>
<td>19.6%</td>
</tr>
</tbody>
</table>

7 Conclusion

The results indicate that a robust feature set containing wind speed, direction, pressure, temperature and time of day can be used to construct an accurate single layer, gradient descent based neural network for timely forecasting of future wind speeds. No external data from either upstream data sources or mesoscale atmospheric data was used for these calculations which suggests that a simple
model framework can still give fairly accurate results for wind speed forecasting in the 2-6 hour range. A similar study was performed by Alexiadis [7] where a neural network was used in conjunction with spatial correlation for a wind farm in Greece and they achieved 20% forecast improvement over the persistence method for a 1 hour lookahead period. Even though the study done by Alexiadis involved a different data set and was aimed at the forecasting of wind power derived from wind speed (rather than wind speed alone), the methods proposed in this paper suggest that they may have a significant advantage since a nearly 50% improvement over the persistence method was achieved for the 2 hour lookahead period. This would also suggest that there is enough signal in the locally available atmospheric data to offset the lack of spatially correlated data. Wind farm developers and utility companies who are interested in power forecasting for the utility grid may not have access to external data sources on which to base their forecasts, however if they have gathered a sufficient amount of local data over time they can still construct a fairly accurate forecast model using the techniques outlined in this paper. Future work in this area may include further fine tuning of the algorithm to increase the model accuracy and extend the forecast period.

References


©2008, Colin Rickert