Georgia Southern University
Digital Commons@Georgia Southern

Electronic Theses & Dissertations

Jack N. Averitt College of Graduate Studies (COGS)

Spring 2013

Using Neural Networks to Provide Local Weather Forecasts

Andrew Culclasure
Georgia Southern University

Follow this and additional works at: http://digitalcommons.georgiasouthern.edu/etd

Part of the Computer Sciences Commons, and the Meteorology Commons

Recommended Citation
Culclasure, Andrew, "Using Neural Networks to Provide Local Weather Forecasts" (2013). *Electronic Theses & Dissertations*. 32.
http://digitalcommons.georgiasouthern.edu/etd/32

This thesis (open access) is brought to you for free and open access by the Jack N. Averitt College of Graduate Studies (COGS) at Digital Commons@Georgia Southern. It has been accepted for inclusion in Electronic Theses & Dissertations by an authorized administrator of Digital Commons@Georgia Southern. For more information, please contact digitalcommons@georgiasouthern.edu.

USING NEURAL NETWORKS TO PROVIDE LOCAL WEATHER FORECASTS

by

ANDREW CULCLASURE

(Under the Direction of James Harris)

ABSTRACT

Artificial neural networks (ANNs) have been applied extensively to both regress and classify weather phenomena. While one of the core strengths of neural networks is rendering accurate predictions with noisy datasets, there is currently not a significant amount of research focusing on whether ANNs are capable of producing accurate forecasts of relevant weather variables from small-scale, imperfect datasets. Also, there is not a significant amount of research focusing on the forecasting performance of neural networks applied to weather datasets that have been temporally rolled-up from a base dataset.

In this paper, a survey of existing research on applying ANNs to weather prediction is presented. Also, an experiment in which neural networks are used to regress and classify minimum temperature and maximum gust weather variables is presented.

This experiment used a dataset containing weather variables recorded every 15 minutes over the course of a year by a personal weather collection station in Statesboro, Georgia. Data cleansing and normalization were applied to this dataset to subsequently derive three separate datasets representing 1-hour, 6-hour, and 24-hour time intervals. Three different NN structures were then applied to these datasets in order to generate minimum temperature regressions at 15-minute, 1-hour, 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges. Maximum gust regressions were also generated for each dataset at 1-hour, 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges. Finally, neural networks were applied to these datasets to classify freezing events and gusty events at 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges.

INDEX WORDS: Artificial neural network, Weather forecasting, Multilayer perceptron, Resilient propagation training, Particle swarm optimization training, Radial basis function training, Dataset preprocessing

USING NEURAL NETWORKS TO PROVIDE LOCAL WEATHER FORECASTS

by

ANDREW CULCLASURE

B.S., Clemson University, 2005

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial Fulfillment of the Requirements for the Degree

MASTER OF COMPUTER SCIENCE

STATESBORO, GEORGIA

2013

© 2013

ANDREW CULCLASURE

All Rights Reserved

USING NEURAL NETWORKS TO PROVIDE LOCAL WEATHER FORECASTS

by

ANDREW CULCLASURE

Major Professor: James Harris
Committee: Wen-Ran Zhang
Robert Cook

Electronic Version Approved:
MAY 2013

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to Dr. James Harris for inspiring me at the graduate student orientation in Atlanta to investigate neural network performance on an extremely localized and somewhat imperfect weather dataset. Also, I would like to thank Dr. Harris for allowing me the use of a dataset he built from a personal weather collection station in Statesboro, GA.

Next, I wish to thank Mr. Jeff Heaton, the founder of the Encog neural network software framework. His well-maintained forums, excellent tutorial materials, and prompt responses to questions from curious minds all over the world greatly aided my research.

Finally, I wish to thank Mr. Chris Taylor. Chris was one academic year ahead of me in the undergraduate computer science program at Clemson University. He has been a great mentor and friend ever since and has always helped me find ways of applying common sense approaches to even the most difficult computer science issues. Thank you for your willingness to help me with a topic that is outside of your comfort zone!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Characterization of Current Research

Ultimately, this paper aims to both explore current applications of neural networks to weather variable prediction and also to apply neural networks to a custom weather dataset. However, before diving into these two tasks, it is beneficial to describe the underlying, practical goals behind these efforts.

To better understand how this research stands apart from existing research, it is helpful to note some major themes in existing research. First, most documented experiments have used neural networks to predict weather occurrences in large-scale settings or environments. For example, neural networks have been used to predict quantitative rainfall amounts for the Dallas-Ft. Worth area [1]. Second, even in research focused on employing neural networks to account for local weather differences not capable of being predicted by large scale weather models, the local differences usually still apply to larger regions being monitored at several different points. For instance, neural networks have been used to process output from numerical weather prediction (NWP) models in order to give more accurate and localized rainfall predictions in four separate regions in the mid-Atlantic United States [2].

This research seeks to take the concept of localization even further. The goal is to determine the feasibility of using a rather imperfect dataset obtained from a single collection unit as input to neural networks in order to obtain regression and classification predictions for various weather variables of interest.

Next, there is sparse research focusing on how temporally summarizing a given dataset affects neural network performance. Temporal summarization results in datasets containing a smaller amount of tuples with derived attributes as opposed to the base dataset, which does not contain derived attributes but contains a larger amount of training tuples. One of the basic ideas behind neural network training is that the network will perform better given a large dataset with a variety of examples. One way to understand this concept is to visualize the analogy of a child learning to classify different animals [3]. A child learning the difference between animals needs to see many different examples of animals, and also needs to see the same animal many times for the classification to “sink in” [3]. However, as several points in the experiment will show later, training a neural network can be a time-consuming process, particularly when the underlying function being approximated is very complex.

What would happen if a large dataset was summarized as a series of smaller datasets that contain derived attributes based on time intervals? Does a smaller dataset containing derived attributes from a 24 hour time frame allow neural networks to regress and classify values as well as a base dataset containing weather measurements recorded every 15 minutes? Is it possible for a neural networks trained on time-summarized datasets to still achieve reasonable predictive performance with the added benefit of having much smaller convergence times? Does the ability to use derived attributes from a rolled-up dataset provide useful training information that is not present in the base level dataset? These core questions guided both the survey and the experiment.

Practical Applications

This research is distinguished from existing research primarily through the choice of datasets. Rather than using large datasets built over decades from a network of collection stations, a dirty, real-world dataset obtained from a single, commercially available, solar-powered weather collection station is used. If it can be shown that neural networks trained from this dataset are capable of predicting even a few useful variables with reasonable accuracy, then further research into predicting a wider range of regression and classification variables from the dataset is warranted.

The practical applications of a system that could be built around this model also make this research worthwhile. First, the experiment uses low-cost or free software and hardware, which minimizes the production cost of a system built around the neural network model. Second, let's assume that experimentation reveals that it is possible to develop a regression and classification model with strong predictive capabilities from the source dataset through the use of neural networks. If a collection station is then pre-positioned in a crop field and allowed to gather data over time, then a neural network can be trained to predict weather variables of interest for that small geographical point. This information would be very useful in remote areas, where radio and network connectivity to existing weather services is limited.

An end user, such as a farmer, could access this neural network information from an application interface such as a mobile device and determine extremely localized estimates for useful weather phenomena such as rainfall, freezing temperatures, wind levels, etc. It is likely that just knowing if freezing temperatures will occur in a given forecast range would be extremely useful in an agricultural setting. If the experiment reveals that neural networks running as part of such a low cost implementation can produce reasonable classifications and regressions of a few common weather variables, then there would be sufficient reason to research the predictability of additional weather variables in the dataset.

Research Motivations

There are three main purposes in this paper. First, some background explanation of neural network basics is provided to set the stage for an in-depth discussion of the experiment. Next, how neural networks have already been applied to weather forecasting is reviewed in order to integrate lessons learned from past research into this experiment when possible. Finally, using a weather dataset from a single station containing measurements recorded every 15 minutes, additional datasets are generated by rolling up the attribute values into datasets based on 1-hour, 12-hour, and 24-hour time intervals. For simplicity, the base dataset is referred to as the 15-minute dataset, the dataset rolled up into 1-hour time intervals is referred to as the 1-hour dataset, etc.

Using all of these previously described datasets, the experiment determines how well three different neural network structures perform in predicting numeric minimum temperature values at 15-minute, 1-hour, 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges. Additionally, it explores how well neural networks numerically predict maximum gust values at 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges. Finally, it evaluates how effectively neural networks can perform classification prediction of freezing events and gusty events at 6-hour, 12-hour, and 24-hour look-ahead ranges.

CHAPTER 2

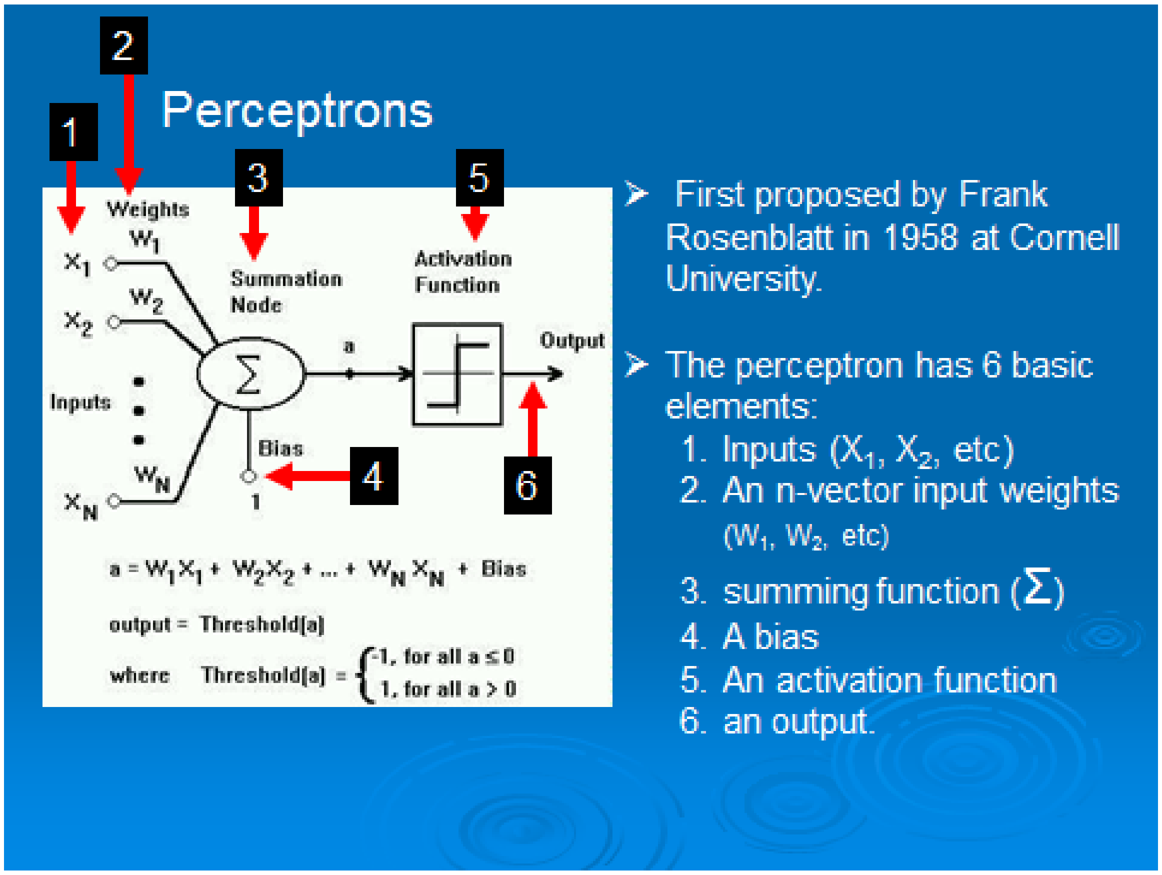
NEURAL NETWORK BASICS

Neuron Components

The basic computational unit in a neural network is the neuron or perceptron. This model, which is based upon the neurons that make up the human brain, was first proposed by Frank Blosenblatt in 1958 at Cornell University [3]. The six basic components of the neuron are shown in figure 1. While this figure sufficiently illustrates the purpose of each component, the input and activation function components are discussed further.

For the purposes of this research, the neuron inputs correspond to the values of chosen weather predictor variables. To better understand this concept, it actually helps to distinguish between the different types of neurons. An input layer neuron has the sole purpose of receiving a single input value. For example, if three predictor variables are chosen to define the input layer of a neural network, then the input layer will have three neurons to accept as input the values of the three predictor variables in a given training or testing tuple.

Hidden layer and output layer neurons can accept an arbitrary number of inputs depending on the synapse type chosen to interconnect the neurons. The most common strategy is to employ weighted synapses, which connect every neuron in the source layer to every neuron in the target layer to form a feed-forward network. For example, in figure 2, neurons in the input layer are connected via synapses to every neuron in the target layer, which also happens to be the output layer.



Perceptrons

$$a = W_1X_1 + W_2X_2 + \dots + W_N X_N + \text{Bias}$$

$$\text{output} = \text{Threshold}[a]$$

$$\text{where} \quad \text{Threshold}[a] = \begin{cases} -1, & \text{for all } a \le 0 \\ 1, & \text{for all } a > 0 \end{cases}$$

- First proposed by Frank Rosenblatt in 1958 at Cornell University.
- The perceptron has 6 basic elements:
  1. Inputs (X_1, X_2, etc)
  2. An n-vector input weights (W_1, W_2, etc)
  3. summing function (Σ)
  4. A bias
  5. An activation function
  6. an output.

Figure 1. The six basic components of the perceptron. This figure shows the basic components of the neuron [3].

The activation function component scales the output from a given layer to a useful value [4]. The desired numerical output for the application tends to drive the choice of activation function. In this experiment, for instance, since all data, including output values, is normalized to fall in the range of 0 to 1, an activation function that produces a value in the range of 0 to 1 is used. This type of activation function is also referred to as a sigmoidal activation function [5]. There are many different types of activation functions suited to different purposes. Perceptron activation functions produce hard threshold values, such as -1, or 1, as indicated in figure 1. Sigmoidal functions, such as Gaussian and tangential functions, offer a range of possible values. Heaton [4] gives in-depth treatment to the various types of activation functions for the interested reader.

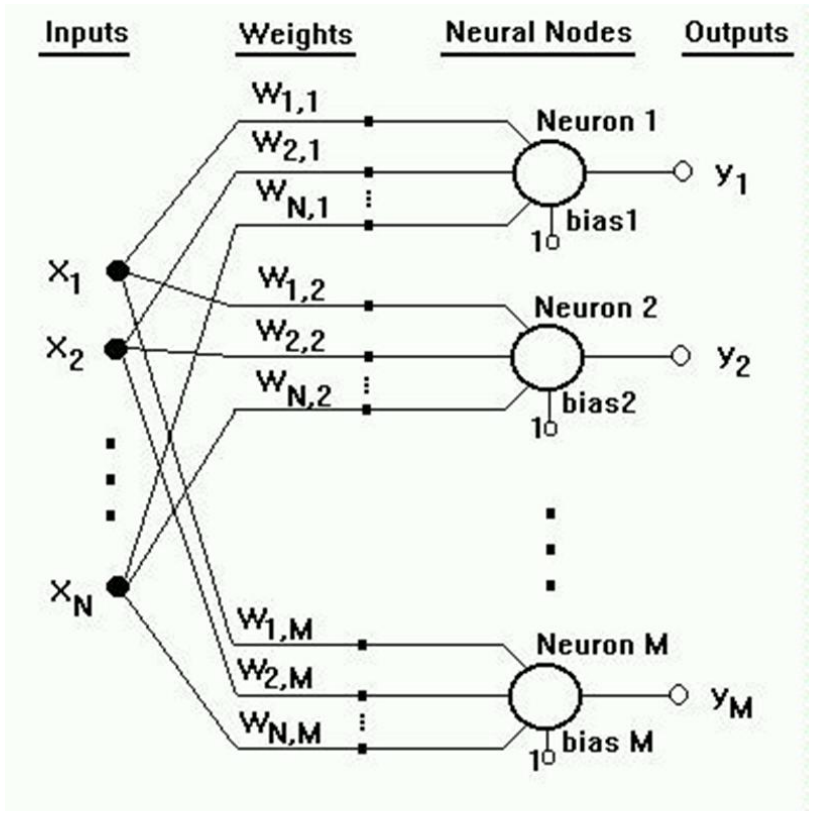


Figure 2. A weighted-synapse, feed-forward neural network [3].

Neural Network Architectures

Since so much research has been devoted to neural networks over the past 40 years, there are abundant resources that explain the nuances of various neural network architectures. However, it is beneficial to give an overview of two major neural network structures that were frequently encountered in the survey of existing research and which were also incorporated into this experiment. Therefore, the feed-forward and radial basis function neural network structures are examined next.

A feed-forward neural network, as explained earlier, consists of an input layer, one or more hidden layers, and an output layer of neurons. These neurons are interconnected via weighted synapses. This type of network is probably the most common neural network structure used in neural network problems [4]. The key point to remember is that a feed-forward network can support an arbitrary number of hidden layers, each of which may contain an arbitrary number of neurons.

Radial basis function (RBF) neural networks are specialized forms of feed-forward networks which only contain a single hidden layer. Each hidden layer neuron represents an RBF function, such as the Gaussian function, which produces a bell-shaped curve that characteristically peaks in the center and then drops off sharply to either side. RBF networks actually learn two different components: the centers and widths of each hidden layer RBF neuron, and the weights that connect these hidden layer neurons to the output layer [6]. The RBF centers and widths can actually be learned via unsupervised training, such as clustering. The weights can then be learned very quickly, which is why RBF neural networks tend to exhibit very quick learning rates.

Neural Network Training

In order to understand the process of neural network training, it helps to think of a neural network in terms of a mathematical function. For instance, suppose a neural network is modeled as $y = O(w,e) + Err$ where y is a vector containing actual outputs from the network, w is a matrix containing the synapse weights, e is a matrix containing training information (inputs and expected outputs), O is the activation function, and Err

is the error between expected and actual outputs. Neural network training is basically an optimization problem that seeks to minimize the error value. Ideally, the goal is to minimize the *Err* to 0, or as close to 0 as possible, but for most applications with large datasets, it is more practical to accept a threshold amount of error between actual and predicted values, such as 1%, in order to reach convergence in a feasible time.

The first form of neural network training to discuss is propagation training. Backpropagation (BPROP), Manhattan update rule, and resilient or dynamic propagation (RPROP) are three commonly used propagation training algorithms. Additionally, they form a class of training algorithms known as gradient descent algorithms, which means that weights are adjusted in a direction to minimize the error between ideal and actual outputs.

Perhaps the key factor for all these algorithms is the varying level of configurability they each entail, which highlights one of the weaknesses of using neural networks to solve problems. Indeed, neural networks are capable of complex learning tasks but "a certain amount of twiddling is needed to get the network structure right and to achieve convergence to something close to a global optimum in weight space" [5]. As discovered during the course of this experiment, this is quite an understatement. In particular, minimizing the amount of twiddling mitigates the random aspect of neural network tuning. Unfortunately, the BPROP and MPROP neural networks both have a random aspect because they require parameters such as learning rate, momentum, and update amounts to be set [4]. In order to better understand the decision to use RPROP training in this experiment, the major features of each propagation training algorithm are described.

BPROP is primarily configured through two settings: learning rate and momentum. The learning rate establishes how much the delta between expected and actual values should be applied to the weights and thresholds and the momentum parameter determines how much change from a previous training iteration is applied to the weight update in a current iteration [4]. Unfortunately, if these parameters are not chosen wisely, the neural network may have a very difficult time converging on a solution. Next, the Manhattan update rule propagation training algorithm is discussed.

The Manhattan update rule ignores the magnitude of change in a gradient descent calculation and focuses on the direction of the change as either positive or negative [4]. In this case, the user supplies a constant that determines the update amount to apply to each weight and threshold. Similar to BPROP parameter selection, if this update amount is not chosen carefully, then the neural network may have a very difficult time converging on an optimal solution of weight and threshold values.

Finally, the RPROP algorithm is examined. This algorithm is very similar to the Manhattan update rule in that it is only concerned with the direction of change from a gradient descent calculation [4]. However, rather than requiring an update amount parameter to be set ahead of time, the delta amount is dynamically calculated for each weight and threshold as training occurs [4]. Since this algorithm minimizes the random aspect present in the BPROP and Manhattan update rule algorithms, it was employed as the propagation training algorithm for this experiment. Next, particle swarm optimization is discussed as an alternate form of neural network training.

The other form of network training to discuss is particle swarm optimization (PSO). This algorithm is based on the migratory actions of birds in a flock [7]. The basic idea is that in a flock of birds looking for food, one bird is located closest to the food. If that bird then communicates his proximity to the food with the other birds in the flock, they will swarm towards the area where the bird closest to the food is and will search for food there.

In terms of neural network optimization, particles are analogous to birds, and represent different weight space solutions for all weights in the neural network. The idea is that over time, the particles will flock towards the solution space containing the particles whose weights contribute most to the error minimization function described earlier. PSO training might be worthwhile in this experiment because the base dataset does contain significant data holes present over a discontinuous time range. If a particle is able to identify the global best early on, then it is possible that convergence times might be less that those achieved by RPROP training.

Local Minima Entrapment

Local minima entrapment is a danger that is present in many types of optimization problems. This occurs when an optimizer, such as a neural network training algorithm, converges on a solution that produces only a locally minimum amount of error, but not the global or lowest amount of error possible. Being stuck in local minima is usually indicated by an unchanging amount of error over the course of many training epochs. Next, some local minima avoidance mechanisms are considered.

There are several ways to mitigate the threat of local minima entrapment discussed in the literature. Genetic algorithms and simulated annealing can be applied to reduce the threat [8]. Since PSO is closely related to genetic algorithms, it was incorporated into this experiment. There are even more sophisticated approaches to avoiding local minima entrapment. A conjugate gradient approach can be used to assign penalty terms during training that constrains the size of weights to help avoid local minima entrapment [9]. Finally, many neural network software frameworks also offer high-level local minima avoidance mechanisms. For instance, Encog, a neural network framework written in Java, provides a way to specify improvement strategies to propagation training algorithms [10]. This feature allows a user to specify weights to be randomized if a certain error threshold is not achieved in a specified number of training cycles.

Choosing the Network Topology

The final neural network topic to discuss before examining the survey of neural network application in weather forecasting is choosing the network topology. Before discussing the issues surrounding hidden layer configuration, it is important to understand why the hidden layer is so important to the neural network, particularly for a weather application. When implementing neural networks "with a single, sufficiently large hidden layer, it is possible to represent any continuous function of the inputs with arbitrary accuracy; with two layers, even discontinuous functions can be represented” [5]. In other words, hidden layers allow neural network to approximate non-linear functions. The capability to approximate non-linear functions is very relevant to weather prediction since the dynamics of meteorology are “inherently nonlinear” [8].

Even with the massive amount of neural network research that has taken place since the 1940s, there is still a large amount of controversy about the best way to determine the number of hidden layers and hidden layer neurons. Existing research suggests the following four options to select hidden layers and hidden layer neurons. The first and most popular approach is the trial and error method, in which suitable hidden layers are “determined for each application using the trial and error method” [11]. Second, hidden neural network layers can be evolved through the use of genetic algorithms [12]. Third, Bayesian modeling can be applied to determine the optimum neural network structure, but this method is computationally very expensive and complicated to implement because it entails computing complicated probabilities of large datasets [13]. Finally, there is an input-output based guideline which suggests that the number of neurons in a hidden layer should be in the range of $[2n^{1/2}, 2n+m]$ where *n* is the number of inputs and *m* is the number of outputs in the neural network [15]. This experiment used a combination of the first and fourth methods, which is discussed later in the paper.

CHAPTER 3

SURVEY OF NEURAL NETWORK APPLICATION TO WEATHER FORECASTING

Goals

Before detailing the trials and results of this experiment, it is beneficial to present a survey revealing how neural networks have been applied to weather forecasting. In this chapter, the relationship of predictor variable selection to a priori domain knowledge is discussed, the difference between stochastic and deterministic approaches to neural network modeling is examined, and the functions approximated by neural networks in previous weather forecasting research are described. Additionally, how and when neural network forecasts have exceeded other traditional forecasting methods are examined.

Predictor Variable Selection

While most neural network-based weather prediction experiments have been conducted by meteorologists or weather researchers, there is a large amount of controversy surrounding the value of a priori knowledge in determining predictor variables. Some research suggests that there are three approaches to predictor-driven forecasting that require descending levels of domain knowledge: physical/numerical modeling of climate, empirical modeling using historical datasets and domain a priori knowledge to pick suitable predictors, and employing statistical techniques to choose suitable predictor variables [15]. It seems that most research experimentation adopts one of the first two approaches. However, in many cases, incorporating a priori weather knowledge is not feasible because it is very difficult to quantify prior knowledge of weather processes as input to a neural network [16]. Furthermore, since choosing predictor variables with minimal a priori domain knowledge is one of the core premises of this experiment, a forward stepwise regression procedure was chosen to select predictor variables. The strengths and weaknesses of this approach are discussed later in the experiment.

The Weather Process

While there is almost universal agreement in the literature that the weather process is a dynamic and nonlinear phenomenon, there seem to be two schools of thought for classifying the nature of a prediction made from a given dataset. On the one hand, there is the belief that if the dataset is sufficiently large, usually spanning many years, then predictions based off of the dataset are deterministic, meaning that every factor needed to determine the next state of the forecast variable is present in the dataset and that a discrete value can be predicted [15]. On the other hand is the belief that given a smaller dataset and a few wisely chosen predictors, a stochastic prediction can be made [15]. A stochastic prediction assumes that there are random variables not present in the dataset which also affect the state of the weather system. In neural network terms, a stochastic prediction assumes that the underlying function that the neural network is supposed to approximate is too complex to be approximated [5]. Therefore, a stochastic prediction computes a probability distribution rather than a discrete value [5]. While it has been proven that both deterministic and stochastic predictions can be combined in an ensemble approach [21], a deterministic approach was adopted for this experiment.

Types of Functions Being Approximated

As described in chapter 2, the objective of using a neural network is to approximate an unknown function. For a deterministic neural network, this function is typically either a classifier, which outputs a hard value such as -1 or 1 to represent some nominal state, or a regression function, which outputs a numerical value [5]. As described in the previous section, if the underlying function is too complex to be approximated, then neural networks can be trained to compute probability distributions [5].

There are experiments in the literature incorporating all three types of function approximations. In one instance, feed-forward neural networks were used to predict quantitative rainfall amounts in the 1 to 3 hour look-ahead range in Bangkok, Thailand, which is equivalent to deterministically approximating a regression function [17]. In another instance, researchers used a radial basis function neural network to predict rain and non-rain days, which is equivalent to using the neural network to deterministically approximate a classification function [18].

Finally, neural networks have been used to compute probabilities of weather phenomena in stochastic circumstances. For instance, researchers in Argentina employed a series of neural networks to approximate cumulative distribution functions for the occurrence of wet and dry spells, which is equivalent to using a neural network to stochastically calculate probability distribution functions [19]. While it would be pointless to recount the details of every experiment, it is worthwhile to discuss some of the active applications of neural networks in the weather forecasting domain and to discuss how success has been measured in these applications.

Applications of Neural Networks in Weather Forecasting

Research illustrates that there is a wide variety of weather forecasting application using neural networks. While it is contrary to the aim of this paper to give an in-depth treatment to every experiment, it is useful to give the reader a brief summary of some practical application areas encountered in the survey of neural network-driven weather forecasting.

First, many experiments have used neural networks to predict quantitative rainfall amounts at various locations and look-ahead ranges. For instance, researchers in Thailand were able to obtain highly accurate forecasts using feed-forward neural networks to predict quantitative rainfall amounts in the one to three hour look-ahead range in order to predict possible flooding dangers [17]. Additionally, neural networks have been used in research to generate probabilities of precipitation and quantitative precipitation forecasts using data from the Eta atmospheric model and upper air soundings [1]. As shown next, neural networks have also been used to predict other less common weather phenomena.

Neural networks have also been used to predict weather phenomena besides the traditional forecast values, such as probability/amount of rainfall, wind speed, barometric pressure, etc. They have been used very successfully to predict tornadoes [8]. Additionally, researchers in Australia successfully used a neural network to identify fog at various forecast ranges ranging from 3 hours to 18 hours around Canberra International Airport [18]. Hopefully, this survey provides the reader with an idea of the depth and variety of neural network-based weather forecasting. The following section discusses the various ways in which success is measured in research.

Measuring Success in a Neural Network Forecast

For the most part, a neural network experiment is deemed successful by the degree to which it exceeds the predictive capabilities of alternate forecasting systems, such as linear or logistic regression systems. For example, in an experiment seeking to enhance local precipitation forecasting unable to be predicted by a large scale numerical weather prediction (NWP) model, both traditional, linear-based model output statistics (MOS) and neural network prediction systems were applied to the NWP output [2]. Comparison of prediction performance revealed that the neural network performed very well compared to the linear model for predicting moderate to high precipitation amounts [2]. Additionally, research revealed that neural networks outperformed logistic regression, discriminant analysis, and rule-based prediction systems in the classification of tornado events [8].

In many cases, various neural network structures are compared to each other to find the optimum neural network configuration for a given forecasting problem. For example, RBF neural networks exhibited superior prediction performance to BPROP neural networks in classifying rain and non-rain days [18]. Similarly ensemble-based neural networks combining the outputs of neural network subcomponents have been shown to outperform each individual neural network subcomponent in the prediction of wind-speed, temperature, and humidity [21]. The following section presents common measurements for objectively identifying predictor performance.

Common Prediction System Measurements

It is appropriate to mention some common measurements used in forecasting applications, both from a numeric standpoint and a classification standpoint. The basic idea is to compare the predicted outputs to the actual outputs. In the previous regression examples cited, RMSE or root mean square error is commonly used to indicate a forecasting system's predictive skill when dealing with numeric prediction. RMSE is defined as $RMSE = \sqrt{\frac{(p_1 - a_1)^2 + \ldots + (p_n - a_n)^2}{n}}$ where *p* is the predicted value, *a* is the actual value, and *n* is the total number of predictions [6]. An RMSE value close to 0 indicates higher predictive skill whereas an RMSE value close to 1 indicates poor predictive skill. While there is a bevy of numeric error measures in the literature, it turns out that in most situations, "the best numeric prediction method is still the best no matter which error measure is used" [6].

Classifier systems also have a wealth of quality measurement metrics. These quality measurements differ from regression quality measurements in that they must provide some sense of both properly classified and misclassified outcomes. These skill scores are usually derived from a 2x2 contingency table, sometimes referred to as a confusion matrix, such as that listed in table 1 below.

Table 1.

An example of a 2x2 contingency matrix

	Event observed	Event not observed
Event Predicted	a	b
Event Not Predicted	c	d

While it is unnecessary to examine every possible skill score that can be derived from this table, the survey revealed that the Heidke Skill Score (HSS) is commonly used to measure classification ability. In terms of the components shown in table 1, the HSS is defined as:

$$\frac{2x(ad-bc)}{[(a+c)(c+d)+(a+b)(b+d)]}$$

HSS scores indicate the forecast skill of a prediction system in comparison to a completely random forecast. A perfect forecasting system has an HSS of 1, whereas a completely unskilled forecasting system has an HSS of 0 [22]. As applied to neural network performance, the HSS indicates the amount of improvement a neural-network based classification system exhibits over a random classification system that knows absolutely nothing about the training data used by the neural network. This skill score is valuable because it provides an intuitive method to measure the success of neural networks by. Additionally, the HSS score is valuable because it makes use of every component of the contingency table. Now that some objective measurement techniques have been discussed, it is appropriate to acknowledge circumstances where neural networks are not a forecasting panacea.

Neural Network Shortcomings

It is only fair to acknowledge some instances where neural network forecasts did not perform better than alternative forecasting systems. For example, while existing research found that neural networks offered significant improvements in predicting moderate to high precipitation amounts when being used to post process output from a numerical weather prediction model, it also found that error measurements were slightly worse for neural networks than for traditional linear based regression models [2]. Furthermore, additional research revealed that for very large datasets, linear regression can often outperform neural networks in the prediction of precipitation amounts from numerical weather model data [23]. These instances reveal that neural networks can be a very useful tool in predicting various weather variables, but that they should not exclude the consideration of other possibilities. Next, some critical lessons learned from this survey are discussed.

CHAPTER 4

LESSONS LEARNED FROM EXISTING RESEARCH

Goals

Perhaps the most important reason to survey existing research is to garner lessons learned that can be applied to current and future research efforts. In this chapter, key lessons learned from existing research regarding data preprocessing and neural network training are discussed.

Data Preprocessing

Data preprocessing is a common step in many disciplines, including data mining, data warehousing, and optimization problems. As this section explains, data preprocessing also plays a very important role in neural networking. First, data normalization is examined. Data normalizing is the process of scaling data “to fall within a smaller range, such as -1.0 to 1.0, or 0.0 to 1.0” [24]. The central idea behind data normalization is to remove the dependence on measurement units, which is directly relevant to weather forecasting since predictor variables are measured using a wide variety of units (miles per hour, degrees Fahrenheit, inches of Mercury, etc.). Data normalization has direct implications to neural network performance. In fact, “normalizing the input values for each attribute measured in the training tuples will help speed up the learning phase” [24]. Furthermore, it is important to normalize neural network training data in order to prevent weights from being overly adjusted due to the possible large magnitudes of measured predictor variable values [23]. Next, the issue of dealing with missing values in training data is explored.

Real world data is usually never perfect. It is often noisy and is missing values. Data mining research has produced several methods of dealing with such dirty data, including interpolation of missing values, binning, clustering for outlier analysis, etc [24]. In the context of neural network-based weather prediction systems however, research has established that removing training tuples with missing values is usually the wisest approach. The forecasting skill of a neural network trained using replacement values for missing values is at the mercy of the quality of the estimated values [16]. In other words, if the estimated values are highly inaccurate, then the predictive capability of a neural network trained with this data will suffer. Furthermore, by comparing neural networks trained using tuples with estimated values and neural networks trained using only complete data, researchers found that there was no significant benefit to using estimated values in training tuples [16]. Next, the issue of class size distributions in classification problems is explored.

As discussed earlier, neural networks are often used to perform classification of variables. The distribution of classes in the training dataset is an important factor that can ultimately affect neural network classifier performance. When dealing with severely disparate class distributions, research has shown that oversampling, or increasing the number of positive classes in an imbalanced dataset, can significantly improve classification accuracy more so than removing negative samples (undersampling) or leaving the training dataset unchanged [25]. Now that significant data preprocessing issues have been examined, factors that significantly affect neural network training are examined next.

Factors that Affect Neural Network Training

Training is perhaps the most important aspect of neural networking since it ultimately determines how the neural network will perform. In terms of weather forecasting, the literature suggests that selecting appropriate training intervals, dealing with inter-seasonal and intra-seasonal variability in datasets, and implementing a validation scheme are key factors to consider in neural network training. All of these issues are addressed in the following paragraphs.

The selection of training or update intervals is highly application and dataset specific. For example, when using neural networks to post-process output from higher-scale NWP models, research suggests that there are two basic approaches: adaptive and non-adaptive [23]. Comparison of both approaches revealed that when dealing with a large dataset containing more than 5 years of measurements, the non-adaptive approach in which neural network retraining is not performed often works very well [23]. However, when analyzing a smaller dataset, the analysis revealed that an adaptive approach in which the neural network is trained frequently to mirror the changes in the NWP model produced accurate localized precipitation forecast results [23].

Unfortunately, there is no clear-cut solution for how often a network should be retrained since the physical processes that drive weather vary so much from region to region and since dataset sizes can vary so much. This experiment adopts a sensible approach that takes the size of the dataset into account. Furthermore, this experiment is based on the concept that a neural network trained from a small dataset can be re-trained more frequently than a neural network trained from a significantly larger dataset.

Next, some research-based approaches to dealing with inter-seasonal and intra-seasonal variability in datasets are discussed. The easiest solution for dealing with inter-seasonal variability is creating a separate neural network for each season [21]. However, this solution is only appropriate for larger datasets that contain multiple seasons of training data. Intra-seasonal variability is also relatively easy to deal with in neural network training. One approach involves randomizing the order of training input to the neural network. Feeding training data to network this way removes intra-seasonal variability, removes any correlation that may exist between consecutively presented inputs and outputs, and perhaps most importantly, mitigates the risk that old data will not be validated correctly with newer test data [16]. Next, how and why a general validation scheme should be implemented in neural network training is considered.

General validation is the final issue covered before describing this experiment. General validation refers to a collection of techniques for dividing a dataset into training and test tuples [5]. While the ideal validation of neural network's predictive capacity comes from generating forecasts on real-world data that is not part of the training dataset, this is not always a possibility. General validation techniques are the next best means of estimating neural network performance. Cross validation, which involves dividing a dataset up into a specified number of folds that each contain training and testing data, allows for the development of a neural network that is optimized for predicting nontraining tuples [23]. Now that significant lessons learned from the survey of existing research have been discussed, the details of this experiment are examined next.

CHAPTER 5

THE DATASET COMPONENTS OF THE NEURAL NETWORK EXPERIMENT

Dataset Description

The base dataset used for this experiment contains 15,893 weather records collected from December 21st, 2011 to January 9th, 2013 via a personal weather collection station at 15 minute intervals. Each tuple contains measurements for the following 14 variables: observation date, indoor humidity, indoor temperature, outdoor humidity, outdoor temperature, absolute pressure, wind, gust, wind direction, relative pressure, dew point, wind chill, wind level, and gust level. While there are certainly much richer datasets available from meteorological databases, this dataset was intentionally chosen because it is imperfect. The specific reasons for using this dataset in the experiment are explained next.

There are many reasons for choosing an imperfect dataset. First, the dataset provides an opportunity to see how neural networks handle datasets with large date gaps. For instance, there is a large gap in collected data from October 12th 2012 to December 29th, 2012. Second, the dataset pertains to a geographic location of interest (Statesboro, Georgia). Finally, the dataset contains many tuples (506 to be exact) with null values. Even high end weather collection systems occasionally fail to collect measurements due to power outages, failed sensors, etc. Since it is useful to assess how neural networks would perform over periods of time in real environments where measurements may occasionally be flawed, this dataset is considered valuable.

Prior to generating roll-up datasets, the dataset was cleansed using Python scripts in order to standardize date/time formats. After this phase was complete, the dataset was loaded into a MySQL database in order to capitalize on its rich set of date and time functions to produce the 1-hour, 6-hour, and 24-hour datasets.

Roll-up Dataset Generation

Before the time-summarized datasets are described, it is helpful to clarify the concept of rolling up as it pertains to this experiment. The traditional data warehousing concept of roll-up usually refers to data aggregation from a lower granularity to a higher granularity [24]. In this experiment, temporally rolling up the base dataset allows for the generation of derived attributes not available in the base dataset that can be used as neural network inputs. For example, a 24-hour roll-up contains a minimum temperature attribute which indicates the minimum temperature observed over the past 24 hours. This information is not available in any tuple in the base 15-minute interval dataset. A complete listing of derived attributes used at each roll-up level is shown in table 3. MySQL date and time functions were applied to the base dataset described above to generate 1-hour, 6-hour, and 24-hour datasets containing 3991, 672, and 173 tuples, respectively.

Forecast Variable Selection

Following roll-up dataset generation, regressed values were generated for minimum temperature and maximum gust predictions. Additionally, positive/negative classifications were generated for freezing events and gusty events, since these phenomena are often predicted in mainstream weather forecasts. In this experiment, a positive freezing event is defined as an occurrence of a minimum temperature less than or equal to 32 degrees Fahrenheit in a given look-ahead period. Also, a positive gusty event is defined as an occurrence of maximum gusts greater than or equal to 10 miles-per-hour in a given look-ahead period. From a meteorological standpoint, maximum observed gusts of 10 miles-per-hour may not really constitute gusty conditions, but defining the threshold as such helps to maintain a somewhat reasonable class distribution of gusty to non-gusty events over the various look-ahead ranges in each dataset.

In this experiment, quantitative precipitation forecasts and rain/non-rain event classifications were not generated for several reasons. First, the original dataset does not contain enough rain events to allow a neural network to train effectively. Second, rainfall is traditionally a very difficult weather phenomenon to predict, even with rich datasets that span decades [20].

Look-Ahead Ranges

Following roll-up dataset generation and forecast variable selection, a series of forecast datasets for each forecast variable at various look-ahead ranges was generated. For minimum temperature regression, datasets were generated for 15-minute, 1-hour, 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges. Maximum gust regression datasets were generated for 1-hour, 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges. For classifying freezing events and classifying gusty events, datasets were generated for 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges.

Predictor Variable Selection

In this experiment, each forecast is treated as its own separate neural network problem, much like an approach suggested in research [9]. Rather than using one constant set of predictors for each look-ahead range, a forward stepwise-regression procedure was employed at each look-ahead range in order to select predictor variables. To perform this regression procedure, the R statistical programming language was used [26]. The predictors chosen by step-wise regression for each forecast in the 15-minute, 1-hour, 6-hour, and 24-hour datasets are shown in tables 2, 3, 4, and 5 respectively.

Table 2.

Predictors used for predictions from the base dataset (15 minute interval)

Possible predictor variables are: month (20), day (21), indoor humidity (22), indoor temperature (23), outdoor humidity (24), outdoor temperature (25), absolute pressure (26), wind (27), gust (28), wind direction (29), relative pressure (30), dew point (31), wind chill (32), wind level (33), gust level (34)						
Look-ahead	15 minutes	1 hour	3 hours	6 hours	12 hours	24 hours
Minimum temperature prediction & Freeze event prediction	21,23,25,29	21,22,23,24,25,27,28,29,30,31,32,34	21,22,23,24,25,27,28,30,31,32,33	22,23,24,25,27,28,29,30,31,32,33	20,21,22,23,24,26,27,28,29,30,31,32,33,34	20,22,23,24,25,26,28,29,30,31,32,33,34
Maximum gust prediction & Gusty event prediction	NA	20,21,22,23,25,26,27,28,29,30,31,32,33,34	20,21,23,24,25,26,27,28,30,31,33,34	20,21,23,24,25,26,27,28,29,30,33,34	20,21,22,23,25,26,27,28,29,30,31,34	20,21,22,23,24,26,27,28,29,30,31,32

Table 3.

Predictors used for predictions from the 1 hour roll-up dataset

Possible predictor variables are: month (1), day(2), average indoor humidity (3), average indoor temperature (4), average outdoor humidity (5), average outdoor temperature (6), minimum outdoor temperature (7), maximum outdoor temperature (8), average absolute pressure (9), average wind (10), minimum wind (11), maximum wind (12), average wind direction (13), average gust (14), minimum gust (15), maximum gust (16), average relative pressure (17), average dew point (18), average wind chill (19), average wind level (20), average gust level (21)						
Look-ahead	15 minutes	1 hour	3 hours	6 hours	12 hours	24 hours
Minimum temperature prediction & Freeze event prediction	5,6,7,8,9,18, 21	3,5,7,9,14,18	3,5,6,7,8,9, 10,11,14,19, 20	1,2,3,4,5,6,7, 8,10,11,13, 17,18,19,20	1,2,3,4,5,6, 7,10,11,15, 17,18,19,20 ,21	1,2,3,5,6,7 ,9,10,11, 12,15,18, 19,20,21
Maximum gust prediction & Gusty event prediction	NA	5,6,7,11,14, 16,17,18,19, 20,21	4,5,6,8,9,13, 14,16,17,18, 20,21	1,2,4,5,6,8,9, 13,14,16,17, 18,20,21	1,2,3,4,5,9, 13,15,16,17 ,19	1,2,3,4,5,6 ,9,13,15, 16,17,18

Table 4.

Predictors used for predictions from the 6 hour roll-up dataset

Possible predictor variables are: month (1), day(2), average indoor humidity (3), average indoor temperature (4), average outdoor humidity (5), average outdoor temperature (6), minimum outdoor temperature (7), maximum outdoor temperature (8), average absolute pressure (9), average wind (10), minimum wind (11), maximum wind (12), average wind direction (13), average gust (14), minimum gust (15), maximum gust (16), average relative pressure (17), average dew point (18), average wind chill (19), average wind level (20), average gust level (21)						
Look-ahead	15 minutes	1 hour	3 hours	6 hours	12 hours	24 hours
Minimum temperature prediction & Freeze event prediction	4,5,6,7,8,10, 11,12,15,19	3,4,5,6,7,8, 15,19,20	5,7,8,15,16, 21	5,7,8,15,21	2,3,7,11,18, 21	2,3,7,8,9, 10,12,14, 18,20
Maximum gust prediction & Gusty event prediction	NA	4,10,11,16, 20,21	4,5,8,10,13, 16,17,18,20, 21	4,5,9,11,12, 13,15,16,17	3,4,9,12,13, 15,16,17	2,5,6,9,12, 13,15,16, 17

Table 5.

Predictors used for predictions from the 24 hour roll-up dataset

Possible predictor variables are: month (1), day(2), average indoor humidity (3), average indoor temperature (4), average outdoor humidity (5), average outdoor temperature (6), minimum outdoor temperature (7), maximum outdoor temperature (8), average absolute pressure (9), average wind (10), minimum wind (11), maximum wind (12), average wind direction (13), average gust (14), minimum gust (15), maximum gust (16), average relative pressure (17), average dew point (18), average wind chill (19), average wind level (20), average gust level (21)						
Look-ahead	15 minutes	1 hour	3 hours	6 hours	12 hours	24 hours
Minimum temperature prediction & Freeze event prediction	7,10,14	2,3,7,9,11,15,16	2,3,7,8,9,11, 15,16	2,3,7,9,11,15,16,19	2,4,7,9,11, 16,19	7,9,11,15, 16,19
Maximum gust prediction & Gusty event prediction	NA	2,12,15,16	11,15,16	11,12,16	2,7,16	4,16,17

Forward stepwise-regression is a statistical procedure that is used to identify suitable predictor variables. Beginning with a predictor variable that has the highest correlation to the forecast variable, it steps forward and tests combinations of other predictor variables and chooses the set of predictors that explains the greatest variance in the forecast variable [2]. In the absence of a priori meteorological knowledge about what predictors are best suited for a given forecast variable, stepwise regression is a sensible approach to selecting predictor variables. However, there are some limitations that should be noted. The stepwise-regression procedure is linear in nature and is intended to choose predictors suitable for a linear regression model. It is possible that this procedure can fail to capture the complex, non-linear nature of weather phenomena [9].

Data Preprocessing

After predictor variables were identified, datasets were generated for each look-ahead range in each roll-up level using MySQL database queries. For the reasons described in chapter 4, each dataset was min-max normalized.

CHAPTER 6

APPLYING NEURAL NETWORKS TO THE DATASETS

Software Frameworks

This experiment incorporated two Java software frameworks to implement neural networks. RPROP and PSO neural networks were implemented using the Encog neural network framework [10]. RBF neural networks were implemented using the Weka (Waikato Environment for Knowledge Analysis) framework [27]. All Java source code was compiled and executed using Java Standard Runtime Environment 1.7 update 15.

Test Machine Description

Dataset generation, data preprocessing, neural network training, and testing were all performed on an HP Pavilion dv6 Notebook with eight gigabytes of RAM and four 2.30GHz Intel Core i5 processors. The 64-bit edition of Linux Mint 14 operating system was used as the main testing platform in order to capitalize on rich shell scripting functionality, which aided tremendously in automating much of the neural network testing.

Basic Process

The basic steps taken to apply a neural network to a given forecasting problem are as follows:

1. Export the dataset as a comma separated value (CSV) file from MySQL.

2. Divide the dataset into 10 separate folds, each containing a training and test-data set (10-Fold Cross Validation).
3. For each fold, train the neural network using the training dataset.
4. Using the trained neural network and the test dataset in each fold, generate regressions if predicting minimum temperature or maximum gust numerical values. If predicting freezing events or gusty events, generate classifications.
5. If performing a regression, average the RMSE from each validation fold. If performing a classification, average the HSS from each validation fold.

Selecting a Neural Network Architecture

As mentioned earlier in chapter 2, there are myriad ways to determine the number of hidden neurons. In this experiment, a specific forecast from a specific dataset is viewed as a distinct neural network problem. For instance, across all datasets, a distinct set of neural networks was trained to predict minimum temperature at the 1-hour look-ahead range, while a different distinct set of neural networks was trained to predict minimum temperature at the 3-hour look-ahead range. This process was repeated for each look-ahead forecast in every dataset. The combination of partitioning each forecast problem into a set of neural network problems and performing 10-fold cross validation for each neural network in each set made it unfeasible to adopt a trial and error method to determine the optimum number of hidden layers and hidden layer neurons. The trial and error method seems more appropriate for fine-tuning a neural network used for a specific forecasting problem, such as determining the optimum neural network structure for predicting maximum gust values at the 12-hour look-ahead range from the 1-hour dataset.

In order to generate a variety of neural network topologies, this experiment used an approach based off the number of inputs and outputs in the given forecasting problem. An existing strategy suggests that the number of hidden layer neurons should be in the range $[2n^{1/2}, 2n+m]$, where n is the number of input neurons and m is the number of output neurons [14]. This suggestion formed the basis for topology generation in this experiment. Table 6 describes the neural network topology scheme that was applied to each forecasting problem and configuration details about each type of neural network used in this experiment.

Table 6.

Neural network configuration details.

	Each topology is in the form i-h-o, where i is the number of input neurons, h is the number of hidden neurons, and o is the number of output neurons. $\max = 2i + o \quad \text{mid} = \frac{i + \max}{2}$				
	Topology 1	Topology 2	Topology 3	Topology 4	Topology 5
	i-i-o	i-mid/2-o	i-mid-o	i-max-o	i-i/2-o
Improvement Strategies (RPROP)	0.01,0.06,5_1 0.01,0.2,5_3	0.01,0.06,5_1 0.01,0.2,5_3	0.01,0.06,5_1 0.01,0.2,5_3	0.01,0.06,5_1 0.01,0.2,5_3	0.01,0.06,5_1 0.01,0.2,5_3
Number of particles (PSO)	20	20	20	20	20
RBF Function	Gaussian	Gaussian	Gaussian	Gaussian	Gaussian
For minimum temperature prediction using RPROP networks, we used an improvement strategy of 0.01, 0.06, 5. For the remaining three variables, we used an improvement strategy of 0.01, 0.2, 5. The improvement strategy is an Encog local minima avoidance feature [10]. The first number represents the target error rate. The second number represents the reset threshold. The final number represents the number of cycles. If the neural network does not produce an error rate below the threshold in the specified number of cycles, then the weights and biases are reset to random values.					

Unfortunately, it was necessary to stray from the network topology scheme in some cases. For instance, when using RPROP and PSO neural networks to perform classification of gusty events on large datasets, such as the 1-hour and 15-minute datasets, it was necessary to sidestep the general scheme described in table 6 and increase the number of hidden layer neurons dramatically in order to reach any convergence at all. This possible over parameterization may have skewed the test results and the assessment of RPROP and PSO network performance in classifying gusty events with large datasets.

Finally, it is necessary to describe the reasons for not expanding past one layer of hidden neurons for RPROP and PSO neural networks (RBF neural networks only have one hidden layer). During initial testing of RPROP and PSO neural networks using two hidden layers, extremely high convergence times were observed across all forecasting problems with no marked improvement in RMSE or HSS scores. In order to collect as much information as possible from a variety of look-ahead ranges and time-summarized datasets, it was necessary to abandon testing with two hidden layers.

CHAPTER 7

EXPERIMENT RESULTS AND DISCUSSION

Minimum Temperature Prediction

Shown below in figures 3, 4, and 5 are the results of applying RPROP, PSO, and RBF neural networks on the experimental datasets to determine minimum temperature predictions over specified look-ahead ranges. Each figure shows the performance of a given neural network type across various datasets as measured by RMSE values calculated at the 15-minute, 1-hour, 3-hour, 6-hour, 12-hour, and 24-hour look-ahead ranges.

A RMSE prediction threshold line is drawn at 0.1, although this value was arbitrarily chosen. What is considered "good" as an RMSE value can vary from application to application depending on what is acceptable to end users and there is no universally agreed upon meteorological standard. For this experiment, an average difference of 10% between the squared differences of predicted and observed values is acceptable. Any points that fall above this threshold line represent poor forecasts and any points that fall below this threshold represent feasible forecasts.

In addition to plotting the actual RMSE values calculated at each look-ahead range, lines of best fit are shown for these RMSE values to indicate the performance characteristics of each dataset (15-minute, 1-hour, 6-hour, and 24-hour) as look-ahead ranges increase. Intuitively, RMSE values are expected to increase as forecast ranges increase, so sensible trend lines should exhibit positive slopes. Also, the point where a given line of best fit intersects with the prediction threshold line reveals the extent to which a neural network is generating feasible forecasts for a given dataset. The rightmost intersection represents the dataset whose neural networks produce the most feasible forecasts (the most dots under the prediction threshold line).

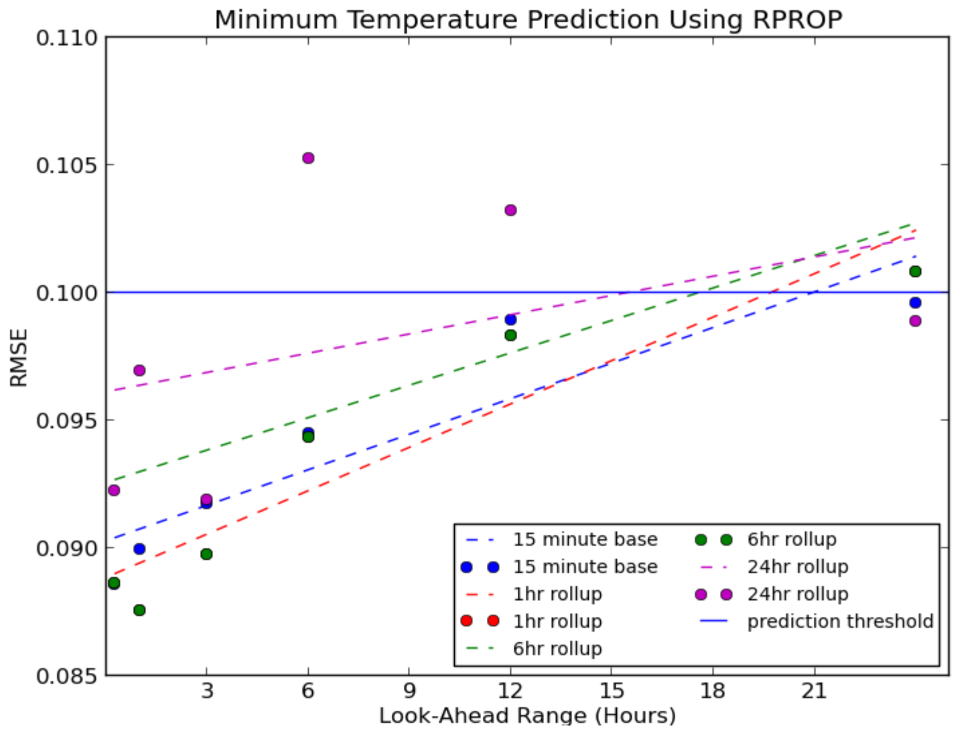


Figure 3. Minimum temperature prediction using the RPROP neural network.

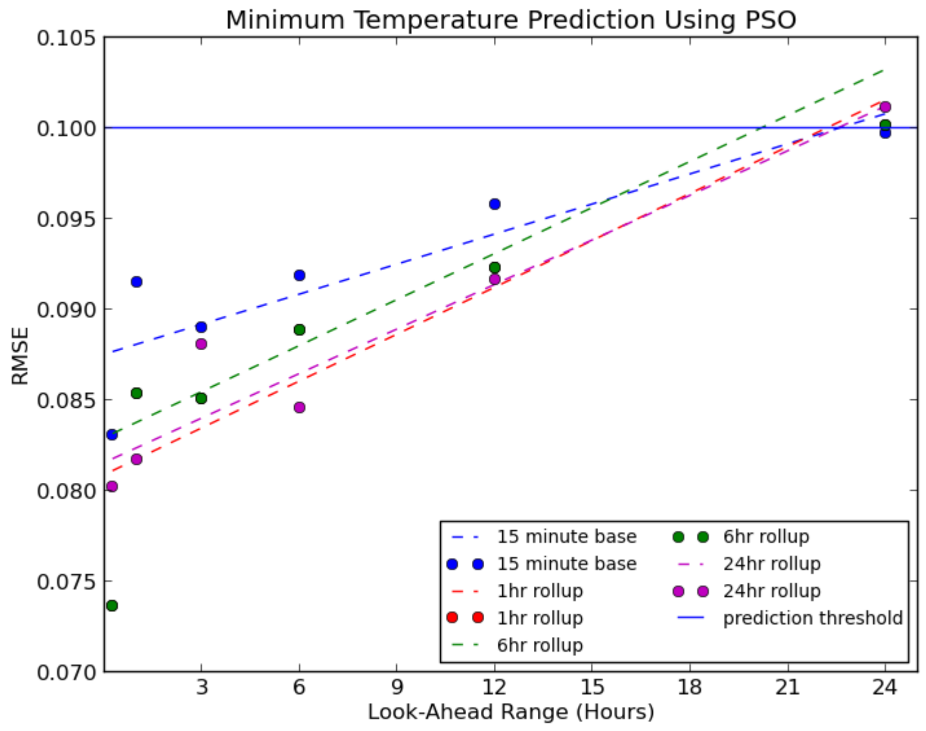


Figure 4. Minimum temperature prediction using the PSO neural network.

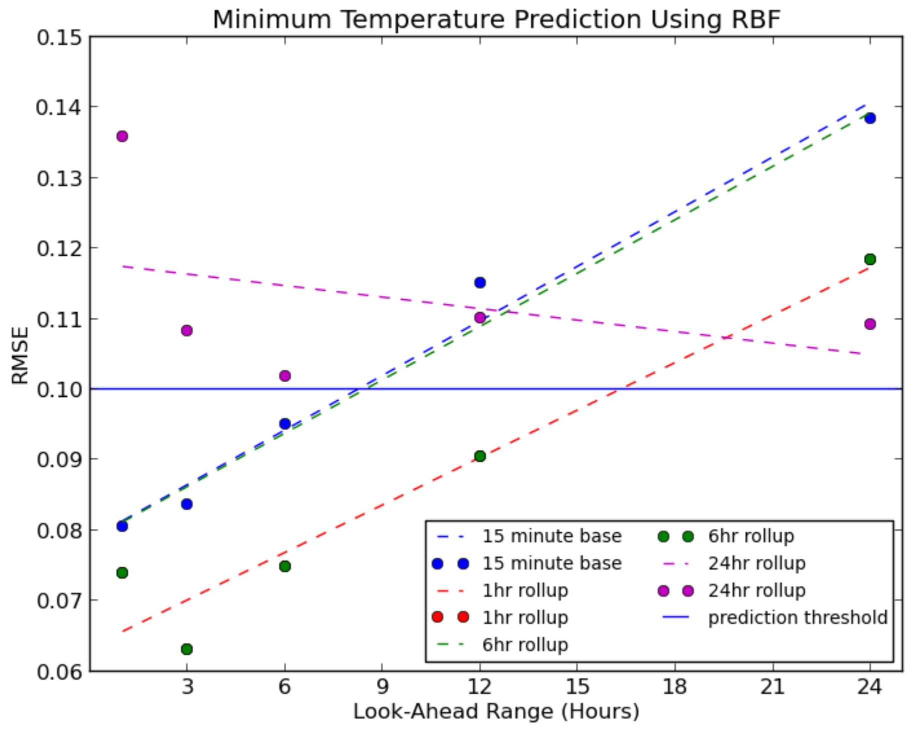


Figure 5. Minimum temperature prediction using the RBF neural network.

Figures 3 and 4 illustrate that RPROP and PSO networks generate very strong forecasts at nearly all look-ahead ranges using most datasets. Also, in figures 3 and 4, the intersection of the 15-minute interval line of best fit with the prediction threshold line occurs at a point further right than any other intersection. This indicates that the RPROP and PSO neural networks perform better on the base 15-minute interval datasets than on the rolled-up datasets as look-ahead range increases. This forecast performance is likely due to the larger number of training tuples available in the 15-minute interval datasets than in the rolled-up datasets. The presence of both positively and negatively sloped lines of best fit in figure 5 reveals that the RBF neural network performed erratically on these datasets. Figure 5 also shows a large amount of scatter of RMSE values, which indicates poor overall forecasting performance for the RBF neural networks. In conclusion, RPROP and PSO neural networks can produce feasible minimum temperature regressions at all look-ahead ranges using any dataset.

In addition to measuring forecast accuracy, convergence times were measured for neural networks trained on each dataset. Figure 6 shows the convergence time performance of all neural networks using the largest dataset (the 15-minute dataset). Most neural networks converged in similar time up the 6-hour look-ahead. Although RBF neural networks did not perform as well as RPROP and PSO networks in regressing minimum temperature values, they exhibited the optimum line of best fit for convergence times, particularly after the 12-hour look-ahead range. Next, the experimental results of neural network performance on forecasting maximum gust values are discussed.

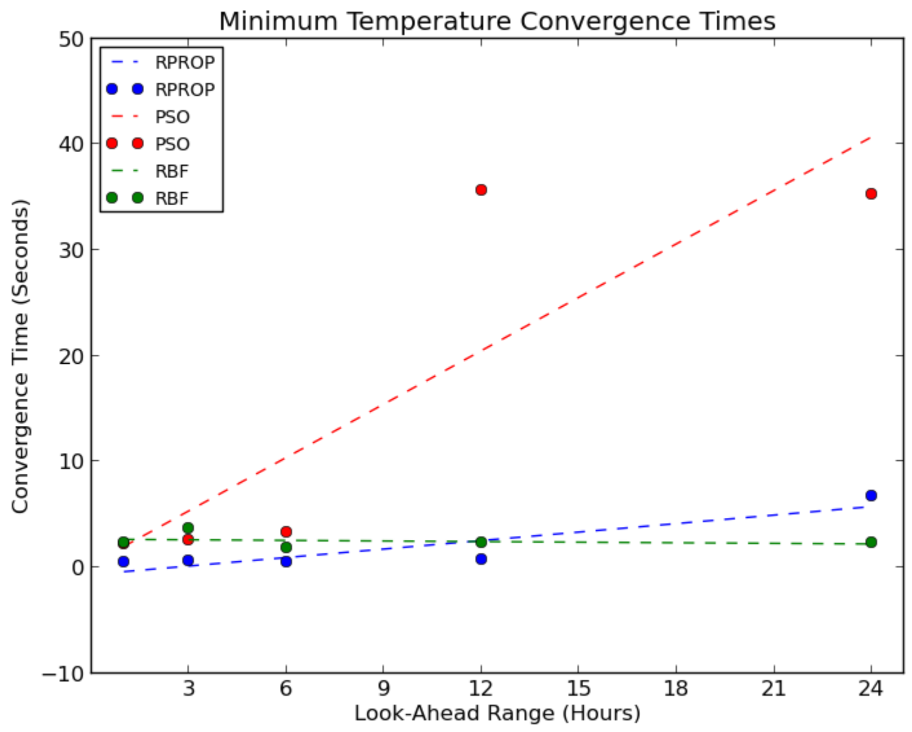


Figure 6. Convergence times for minimum temperature prediction.

Maximum Gust Prediction

Shown below in figures 7, 8, and 9 are the results of applying RPROP, PSO, and RBF neural networks on the experimental datasets to determine maximum gust predictions over the specified look-ahead ranges. The relatively larger number of RMSE values above the prediction threshold line indicates that maximum gust prediction functions are tougher for neural networks to approximate than minimum temperature functions.

While RPROP and RBF networks were able to reach convergence at all look-ahead ranges, PSO proved unable to converge past the 12-hour look-ahead range. It is possible that PSO is not well-suited to approximating the function that models maximum gusts past this look-ahead range, which highlights the importance of testing a variety of neural networks in a given forecasting problem. Figure 9 reveals that RBF networks did not perform as well as RPROP networks, particularly at and above the 3-hour look-ahead range. It is interesting to note that the 24-hour roll-up provided very low RMSE values up to the 12-hour look-ahead range for the RPROP network. Perhaps the maximum gust dataset is well-suited for time summarization at the 24-hour level. Finally, figure 7 corroborates earlier observations in that it shows the line of best fit for the 15-minute dataset intersecting at the rightmost point on the threshold prediction line. This intersection location indicates that using the larger dataset yields more accurate RMSE values as look-ahead range increases. In conclusion, these figures illustrate that the RPROP neural network is the optimum choice and yields feasible maximum gust predictions at the 1-6 hour look-ahead range.

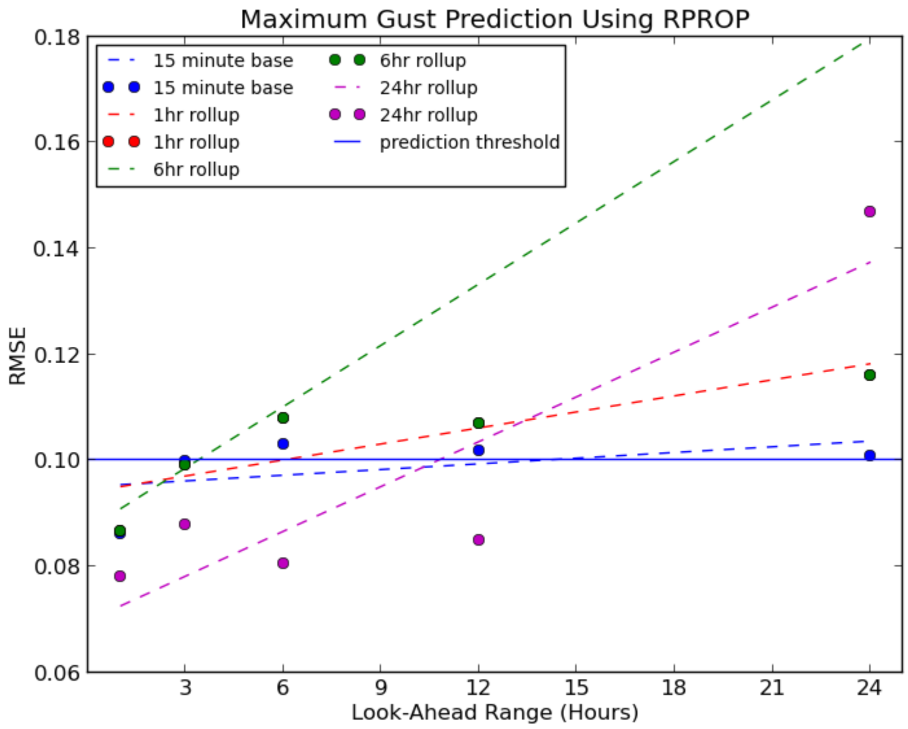


Figure 7. Maximum gust prediction using the RPROP neural network.

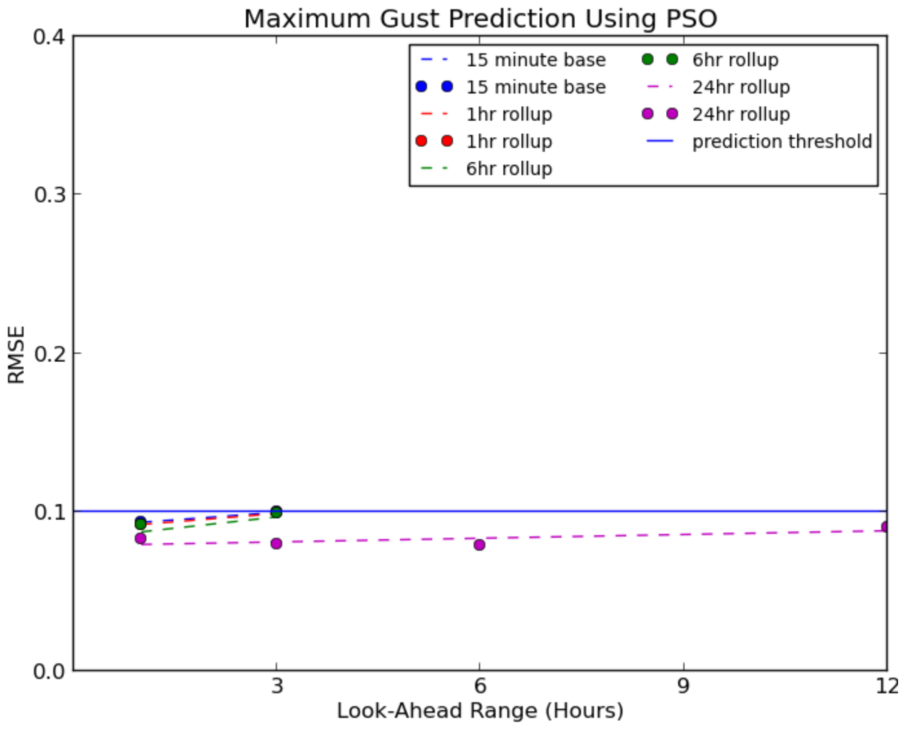


Figure 8. Maximum gust prediction using the PSO neural network.

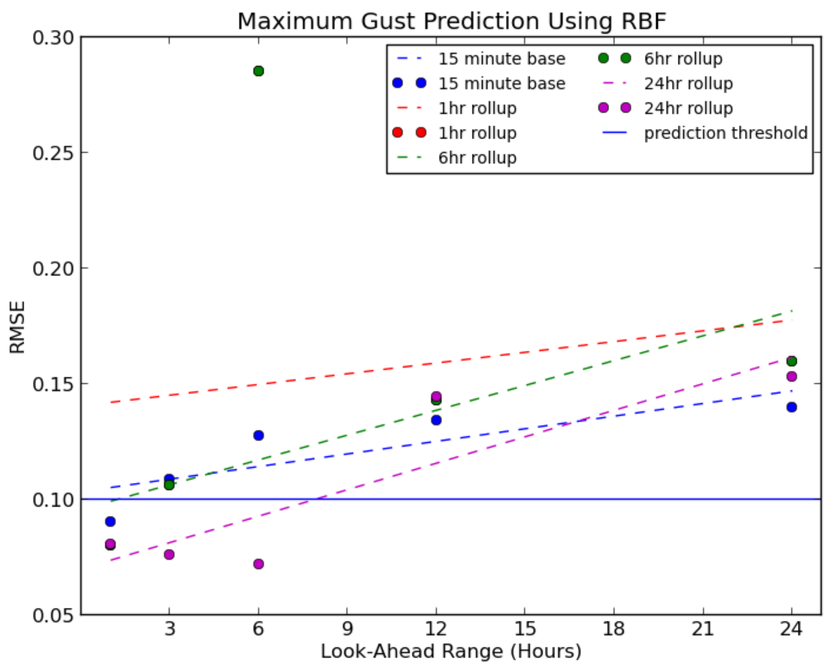


Figure 9. Maximum gust prediction using the RBF neural network.

Figure 10, shown below, depicts the convergence times and respective lines of best fit per neural network type on the 15-minute dataset. The extreme increase in convergence time for the RPROP algorithm reinforces the earlier observation that the maximum gust function becomes extremely difficult to approximate at the 6-hour look-ahead range. Again, the RBF network exhibited the fastest convergence times across all look-ahead ranges by a significant margin, although it performed worse in terms of prediction accuracy as shown in figure 9. Next the ability of neural networks to classify freezing events using the defined datasets is analyzed.

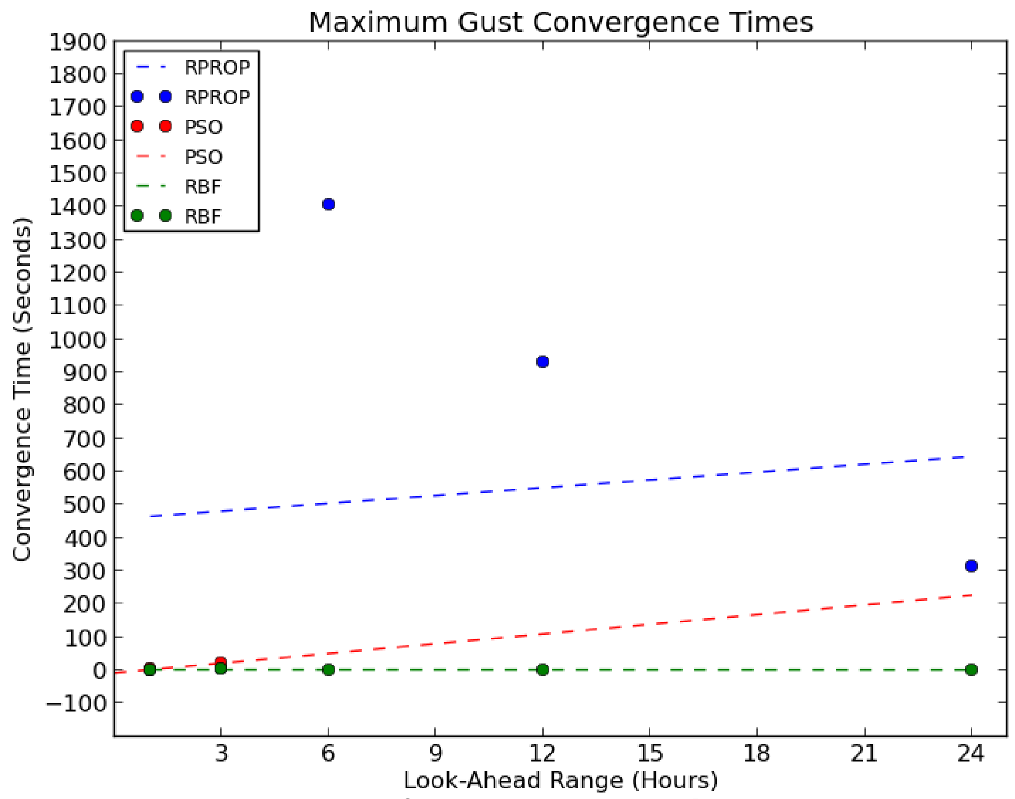


Figure 10. Convergence times for maximum gust prediction.

Freezing Event Classification

As mentioned in chapter 3, a skill score gives a more reliable understanding of the predictive capabilities of a classification system than a numeric error measure. An arbitrary HSS prediction threshold of 0.70 is drawn in each figure to illustrate the point below which forecasts are deemed poor. Intuitively, an HSS score of 0.70 means that the forecasts from the experiment's neural-network based system represent a 70% improvement over forecasts generated at random by a system that knows nothing about the experimental data.

Shown below in figures 11, 12, and 13 are the results of applying RPROP, PSO, and RBF neural networks on the experimental classification datasets to determine freezing events. A positive freezing event indicates that a minimum temperature equal or less than 32 degrees Fahrenheit was observed in the specified look-ahead range. The presence of both positively and negatively sloped lines of best fit in the RPROP and PSO models immediately calls into question their effectiveness at classifying freezing temperatures. Intuitively, HSS scores are expected to decrease as the look-ahead range increases, so it does not make sense for a line of best fit of HSS scores to have a positive slope. Additionally, both the RPROP and PSO networks exhibit a high amount of scatter for plotted HSS values above and below the prediction threshold line, which indicates that they may have difficulty in approximating the function that classifies events as freezing or not freezing.

While RPROP and PSO neural networks did not perform remarkably well in classifying freezing events, the RBF neural network exhibited sensible characteristics. For most roll-up levels, the HSS values decrease as the look-ahead range decreases, which makes sense. Additionally, the RBF neural network’s line of best fit for the 15-minute dataset has the smallest negative slope, which indicates that it will have the rightmost intersection with the prediction threshold line. In conclusion, RBF neural networks trained using the 15-minute dataset render feasible forecasts across all look-ahead ranges.

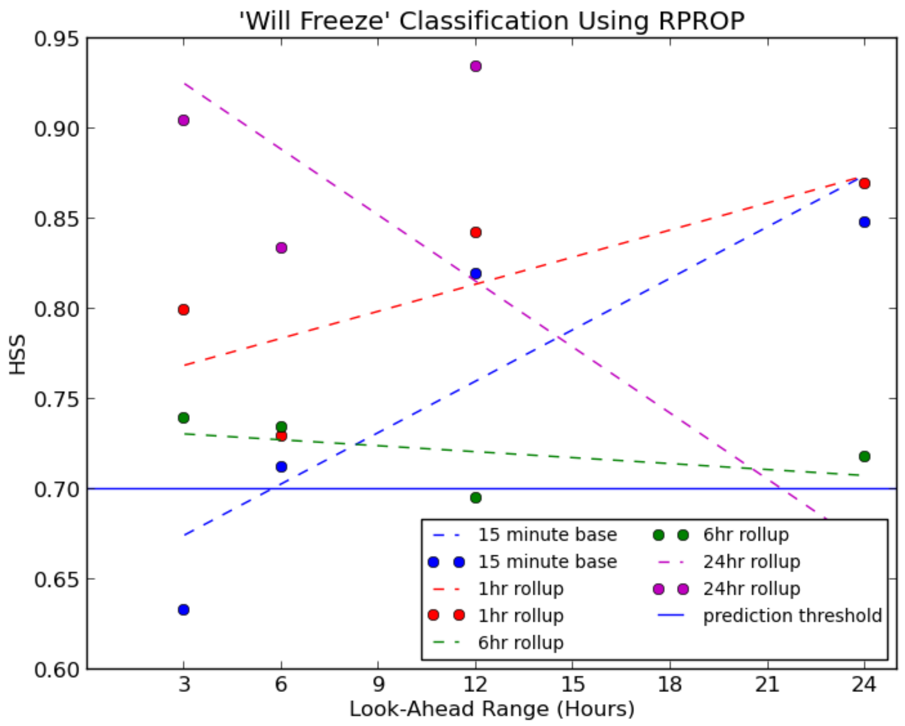


Figure 11. Predicting freezing events using the RPROP neural network.

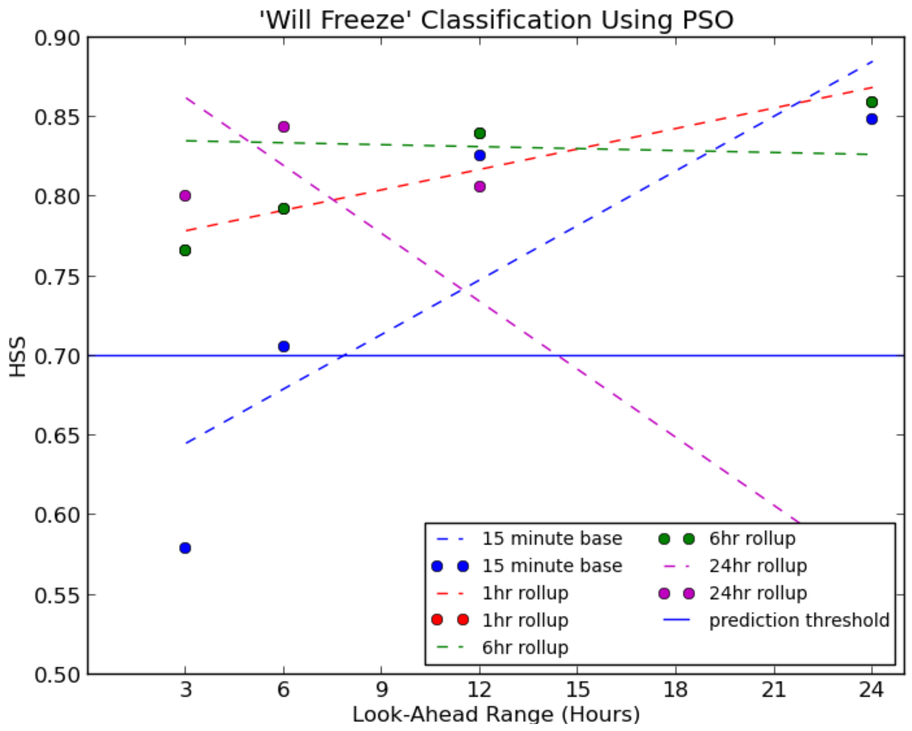


Figure 12. Predicting freezing events using the PSO neural network.

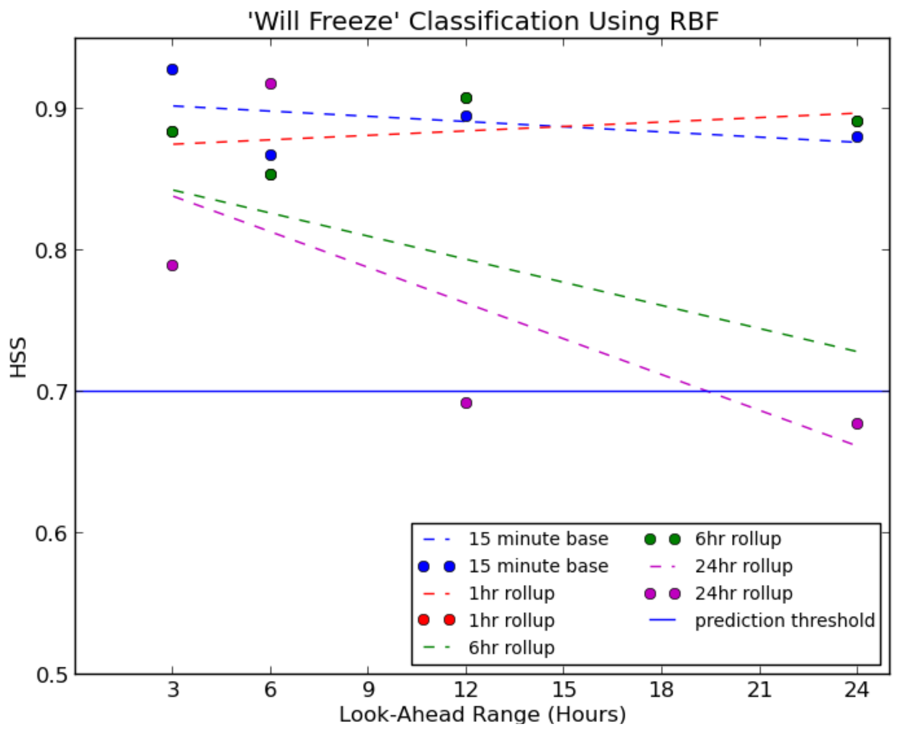


Figure 13. Predicting freezing events with the RBF neural network.

Figure 14 illustrates the convergence times observed when using neural networks to classify positive and negative freezing events from the 15-minute dataset, which contains the largest amount of tuples. This figure shows that after the 6-hour look-ahead range, PSO convergence times increased almost exponentially, which indicates that PSO neural networks may not be suitable for classifying freezing events. In contrast, RPROP and RBF neural networks both exhibited relatively small-sloped lines of best fit across all look-ahead ranges. Although RPROP exhibited lower convergence times than RBF, RBF significantly outperformed RPROP in classifying freezing events, so for the experimental dataset, RBF is the optimum neural network for classifying freezing events.

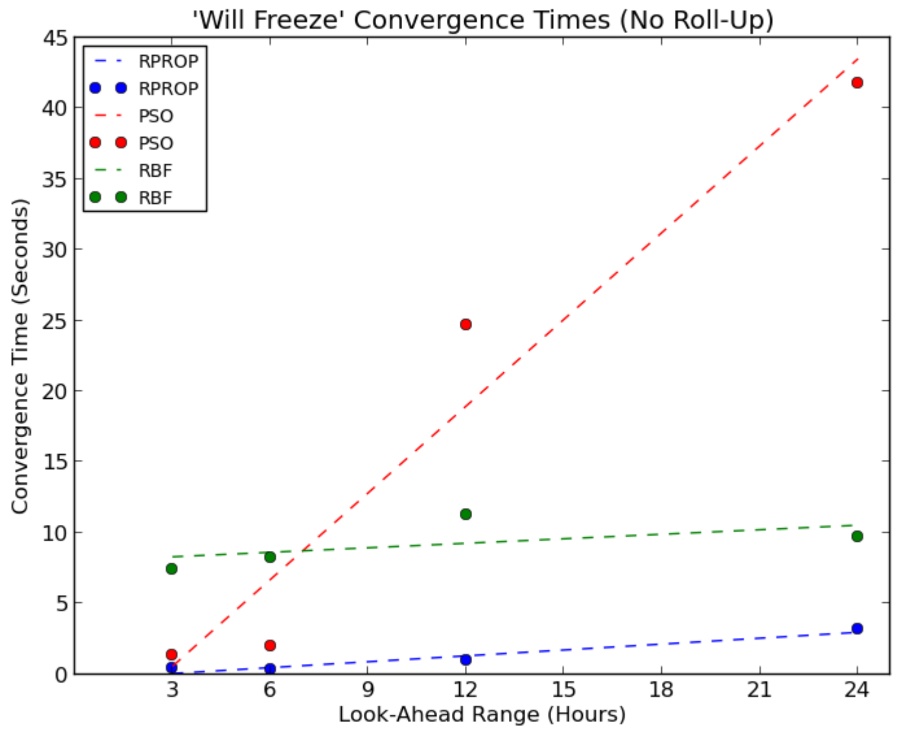


Figure 14. Convergence times for neural networks predicting freezing events.

Gusty Event Classification

In addition to classifying freezing events, this experiment tested how effectively neural networks can classify gusty events. Again, a gusty event is considered to be positive for a maximum gust of 10 miles-per-hour occurring in a given forecast range and negative otherwise. This proved to be the toughest test. Unfortunately, PSO networks were unable to converge on any dataset, so they are omitted here.

As figure 15 reveals, the RPROP network failed to reach convergence on the base 15-minute dataset. Additionally, the presence of positively and negatively sloped lines of best fit indicates that RPROP networks may not be suitable for approximating the gusty classification function. Also, the 1-hour dataset, which offers the most training data after the 15-minute data set, exhibits a positively sloped line of best fit which does not make sense. How is the HSS actually getting better as time goes on? One possibility is that the RPROP networks have been over-parameterized in an attempt to reach a convergence. In turn, this could have caused the RPROP networks to overfit the training data, which may have artificially caused HSS values to increase with look-ahead ranges. However, previous analysis of classifying freezing events depicted that RPROP neural networks did not perform as well as RBF neural networks. Unfortunately, this experiment did not produce any conclusive evidence of RPROP and PSO performance in classifying gusty events.

Whereas the RPROP neural network performed rather indeterminably in the classification of gusty events, the RBF neural network performed much more stably. Figure 16 reveals that except for the 1-hour dataset's line of best fit, the lines of best fit exhibited negative slopes to indicate decreasing HSS values over increased look-ahead ranges, which is intuitively sound. However, the RBF neural network did generate a majority of HSS values under the prediction threshold limit. Using the 15-minute dataset as a guide, the RBF neural network prediction system only generates HSS scores between 0.5 and 0.6. While the 24-hour dataset did produce HSS values above the prediction threshold line at the 6-hour and 12-hour look-ahead ranges, it is likely that this level of roll-up did not provide enough training information to the neural network. In conclusion, these observations illustrate that gusty events cannot be feasibly predicted using RBF networks. However, it is worth noting that RBF neural networks exhibited remarkably better convergence times across all look-ahead ranges than RPROP neural networks, as indicated in figure 17.

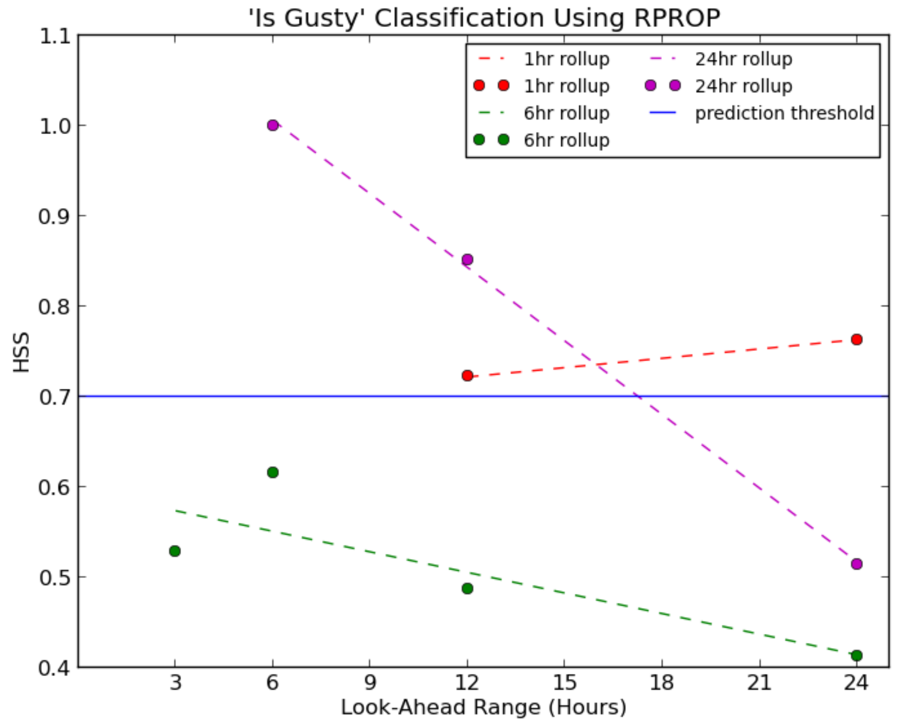


Figure 15. Predicting gusty events with the RPROP neural network.

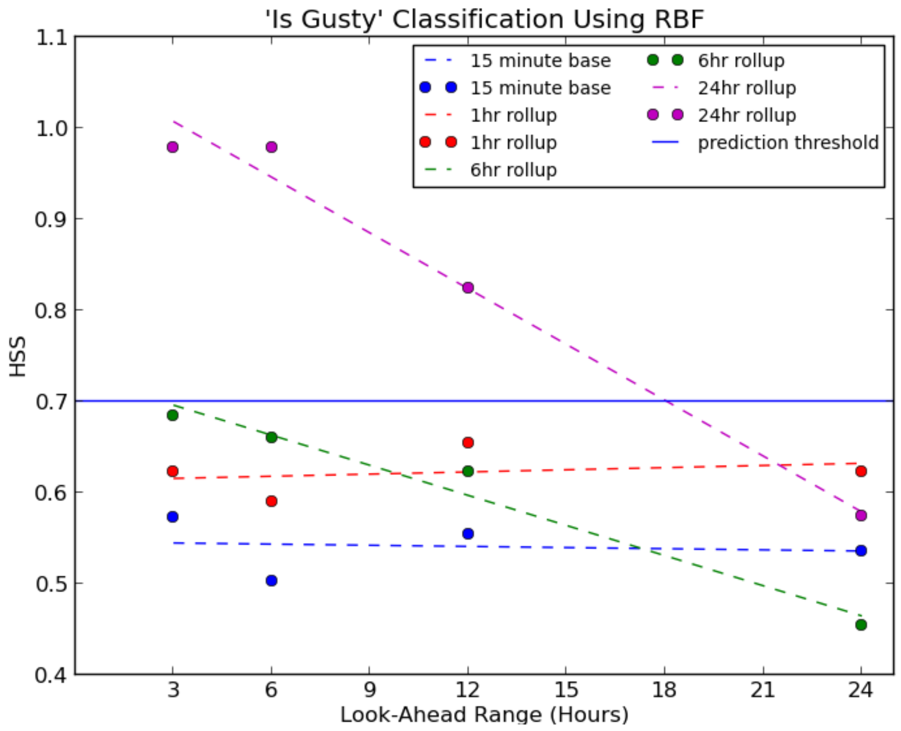


Figure 16. Predicting gusty events with the RBF neural network.

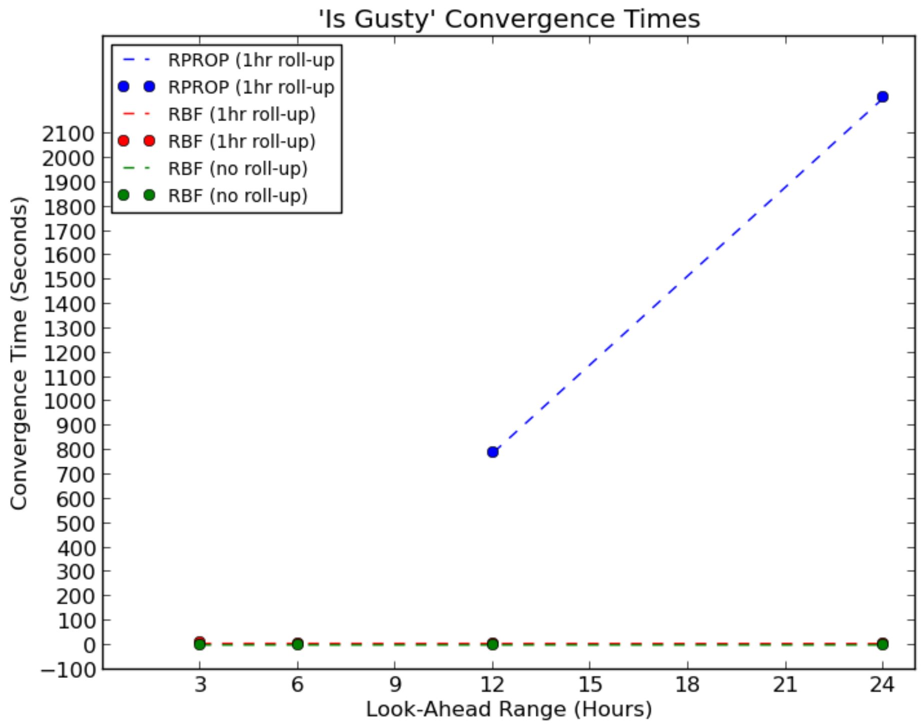


Figure 17. Convergence times for gusty event prediction.

CHAPTER 8

CONCLUSION

Feasibility Assessment

This paper examines the effectiveness of using neural networks to forecast weather variables, both in existing research and with an experimental dataset. First, it presents a survey of existing literature to examine how neural networks have been used to generate both regression and classification forecasts. From this survey, a body of important lessons learned have been identified and discussed, such as the importance of normalizing data, methods of dealing with imperfect training data, and approaches to selecting predictor variables.

Following the survey of neural network research, three neural network architectures are applied to experimental datasets in order to forecast minimum temperature values, maximum gust values, freezing event classifications, and gusty event classifications. This experiment revealed that RPROP and PSO neural networks can predict minimum temperature values very effectively up to 24 hours out regardless of the base dataset used. It also illustrated that RPROP and RBF neural networks can generate reasonable forecasts of maximum gust values in the 3 to 6 hour range. Next, it showed that RBF neural networks can be applied to our dataset to classify freezing events with great accuracy up to 24 hours out, whereas RPROP and PSO neural networks did not perform well in this task. Additionally, the experiment explains that while RBF neural networks exhibited the most reasonable behavior when classifying gusty events, they were not able to generate forecasts with HSS scores above 0.6 for even the closest of look-ahead ranges. In most cases, particularly when acting as a classifier, RBF neural networks converged significantly faster than their RPROP and PSO counterparts. In conclusion, the results of the experiment show sufficient reason to investigate the predictability of other weather variables present in the base dataset using neural networks.

In nearly all forecasts generated, using the 15-minute dataset generally produced more accurate results than using rolled-up datasets. For rolled-up datasets, the derived attributes in the training tuples were not able to compensate for the loss of available training tuples. This observation mirrors the general idea of neural networking that the more examples of training data a neural network processes, the more successful it will be in regressing or classifying variables.

Acknowledgement of Limitations

While the results of the experiment are well-founded, it is appropriate to acknowledge several shortcomings. First, the predictive models built from the neural networks were not used to forecast actual weather, but instead validated using 10-fold cross validation. While cross-validation is a powerful and statistically proven verification method, it is not as good an indicator of predictive performance as real-world validation is. Next, the convergence times required for RPROP and PSO networks were severely underestimated, particularly as roll-up levels decreased and datasets became larger. In particular, extremely high convergence times were observed when using RPROP and PSO networks to classify positive and negative gusty events. To compensate, the number of hidden layer neurons was increased dramatically, which may have over-parameterized these particular neural networks. Ultimately, this means that the assessment of RPROP and PSO performance in classifying gusty events may be skewed.

Future Work

There are many possibilities to expand this experiment into future work. Examining the effect of incorporating a distributed computing architecture on convergence times and forecast accuracy is one such avenue. Additionally, it would be worthwhile to train neural networks from a more recent dataset and subsequently use them as real-world forecasting system in order to determine their forecasting capabilities. Next, it would be useful to examine how reducing the number of predictor variables would affect neural network performance and convergence times by applying basic set theory to the predictor variables listed in tables 2, 3, 4, and 5. In other words, for a given forecast problem, predictor variables that are common to a given temporal roll-up across all look-ahead ranges could be used, or predictor variables that are common to a given look-ahead range across all temporal roll-ups could be used. Finally, it would be worthwhile to assess the predictability of additional variables in the dataset, from both a regression and classification standpoint.

REFERENCES

[1] T. Hall, H.E. Brooks, and C.A. Doswell III, “Precipitation forecasting using a neural network,” *Weather and Forecasting*, vol. 14, no. 3, pp. 338-345, Jun. 1999.

[2] R.J. Kugliowski and A.P. Barros, “Localized precipitation forecasts from a numerical weather prediction model using artificial neural networks,” *Weather and Forecasting*, vol. 13, no. 4, pp. 1194-1204, Dec. 1998.

[3] J. Harris, “Neural Networks,” class powerpoint slides for CSCI 7130, Computer Science Department, Georgia Southern University, Dec. 2011.

[4] J. Heaton, *Introduction to Encog 2.5: Revision 3-November 2010*, Heaton Research, Inc., 2010.

[5] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, Third Edition*, Prentice Hall, 2010.

[6] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*, Morgan Kaufmann Publishers, 2005.

[7] J. Kennedy and R.C. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, vol. 4, pp. 1942-1948.

[8] C. Marzban and G.J. Stumpf, “A neural network for tornado prediction based on Doppler radar-derived attributes,” *Journal of Applied Meteorology*, vol. 35, no. 5, pp. 617-626, May 1996.

[9] K. Koizumi, “An objective method to modify numerical model forecasts with newly given weather data using an artificial neural network,” *Weather and Forecasting*, vol. 14, no. 1, pp. 109-118, Feb. 1999.

[10] “Encog Machine Learning Framework,” Internet: http://www.heatonresearch.com/encog, [Mar. 24, 2013].

[11] H.M. Abdul-Kader, “Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting,” *International Journal of Computer Science and Network Security*, vol. 9, no. 3, pp. 92-99, 2009.

[12] X. Hu, “PSO Tutorial,” Internet: http://www.swarmintelligence.org/tutorials.php, [Mar. 24, 2013].

[13] B. Cheng and D.M. Titterington, “Neural networks: a review from a statistical perspective,” *Statistical Science*, vol. 9, no. 1, pp. 2-54, Feb. 1994.

[14] D. Fletcher and E. Goss, “Forecasting with neural networks: an application using bankruptcy data”, *Information & Management*, vol. 24, no. 3, pp. 159-167, Mar. 1993.

[15] H.D. Navone and H.D. Ceccatto, “Predicting Indian monsoon rainfall: a neural network approach,” *Climate Dynamics*, vol. 10, pp. 305-312, 1994.

[16] D. Fabbian, R. De Dear, and S. Lellyett, “Application or neural network forecasts to predict fog at Canberra International Airport,” *Weather and Forecasting*, vol. 22, no. 2, pp. 372-381, Apr. 2007.

[17] N.Q. Hung, M.S. Babel, S. Weesakul, and N.K. Tripathi, “An artificial neural network model for forecasting in Bangkok, Thailand,” *Hydrology and Earth System Sciences*, vol. 13, no. 8, pp. 1413-1425, Aug. 2009.

[18] T. Santhanam and A.C. Subhajini, “An efficient weather forecasting system using radial basis function neural network,” *Journal of Computer Science,* vol. 7, no. 7, pp. 962-966, Jun. 2011.

[19] J-P Boulanger, F. Martinez, and E.C. Segura, “Neural network based daily precipitation generator (nngen-p),” *Climate Dynamics*, vol. 28, pp. 307-324, Sep. 2006.

[20] H. Yuan, X. Gao, S.L. Mullen, S. Sorooshian, J. Du, and H-M. H. Juang, “Calibration of probabilistic quantitative precipitation forecasts with an artificial neural network,” *Weather and Forecasting,* vol. 22, no. 6, pp. 1287-1303, Dec. 2007.

[21] I. Maqsood, M.R. Khan, and A. Abraham, “An ensemble of neural networks for weather forecasting,” *Neural Computing and Applications*, vol. 13, no. 2, pp. 112-122, Jun. 2004.

[22] “Heidke Skill Score (HSS),” Internet: http://www.eumetcal.org/resources/ukmeteocal/verification/www/english/msg/ver_categ_forec/uos3/uos3_ko1.htm, [Mar. 24, 2013].

[23] Yuval and W.W. Hsieh, “An adaptive nonlinear mos scheme for precipitation forecasts using neural networks,” *Weather Forecasting,* vol. 18, no. 2, pp. 303-310, Apr. 2003.

[24] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques, Third Edition*, Morgan Kaufmann Publishers, 2012.

[25] M.A. Mazurowski, P.A. Habas, J.M. Zurada, J.Y. Lo, J.A. Baker, and G.D. Tourassi, "Training neural network classifiers for medical decision making: the effects of imbalanced datasets on classification performance," in *Neural Networks: Advances in Neural Networks Research, International Joint Conference on Neural Networks 2007* , vol. 21, no 2-3, pp. 427-436, 2008.

[26] "The R Project for Statistical Computing," Internet: http://www.r-project.org/, [Mar. 24, 2013].

[27] "Weka 3: Data Mining Software in Java," Internet: http://www.cs.waikato.ac.nz/ml/weka/index.html, [Mar. 24, 2013].