MACHINE LEARNING METHODS FOR THE DETECTION OF RWIS SENSOR MALFUNCTIONS

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

ADITYA POLUMETLA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

JULY 2006

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of Master's thesis by

Aditya Polumetla

and have found it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

_____Dr. Richard Maclin_____
Name of Faculty Advisor(s)

_____
Signature of Faculty Advisor(s)

_____
Date

GRADUATE SCHOOL

Abstract

Mn/DOT uses meteorological information obtained from Road Weather Information System (RWIS) sensors for the maintenance of roads and to ensure safe driving conditions. It is important that these sensors report accurate data in order to make accurate forecasts, as these forecasts are used extensively by Mn/DOT. Real time detection of sensor malfunctions can reduce the expense incurred in performing routine maintenance checks and re-calibrations of the sensors, while guaranteeing accurate data.

In this work we predict RWIS sensor values using weather information from nearby RWIS sensors and other sensors from the AWOS network. Significant and/or systemic deviations from the predicted values are used to identify malfunctions. Based on historical data collected from the sensor and its nearby locations, we construct statistical models that can be used to predict current values. We use machine learning (ML) methods to build these models. We employ three types of ML models: classification algorithms, regression algorithms and Hidden Markov Models (HMMs). We use classification algorithms such as J48 decision trees, Naive Bayes and Bayesian Networks, regression algorithms such as Linear Regression, Least Median Square (LMS), M5P regression trees, MultiLayer Perceptron, RBF Networks and Conjunctive Rule, and HMMs to predict the variables temperature, precipitation type and visibility.

We selected a representative sample of the RWIS sites in Minnesota. We employed different representations of the data to try improve the model efficiency. To use temperature in regression algorithms and HMMs, we developed a method to discretize temperature. The Viterbi algorithm used in HMM was modified to obtain the symbol observed along the most probable path.

From the results, we observed that LMS and M5P are highly accurate in predicting temperature and visibility. Predicting precipitation works well with J48 decision trees and

Bayesian Belief Networks. HMMs can predict temperature class values accurately but fail in case of precipitation type. Our experiments suggest hese methods can be efficiently used to detect malfunctions of the sensors that report these variables.

Acknowledgements

I would like to take this opportunity to acknowledge all those who helped me during this thesis work. I would like to thank my advisor Dr. Richard Maclin for introducing me to the world of machine learning, his valuable suggestions and guidance during the course of this thesis work, and his patience in reviewing my thesis.

I would like to thank my committee members Dr. Donald Crouch and Dr. Robert McFarland for evaluating my thesis and for their suggestions.

I would like to thank the Northland Advanced Transportation Systems Research Laboratories for providing the funding for this research and its members Dr. Richard Maclin, Dr. Donald Crouch and Dr. Carolyn Crouch for their ideas and feedback. I would especially like to thank my team members at various times during the project including Saiyam Kohli, Ajit Datar, Jeff Sharkey and Jason Novek for their help in the research work.

I would also like to thank all the faculty and staff at the Computer Science Department at the University of Minnesota Duluth for their assistance during my master's course-work.

My fellow graduate students who offered great help during my study and stay in Duluth, I would like to thank them all.

Finally, I would like to thank my parents and grandparents for their endless support and encouraging me to do my best.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many state transportation departments, such as the Minnesota Department of Transportation (Mn/DOT), have Road Weather Information System (RWIS) units located along major roadways. Each RWIS unit employs a set of sensors that collect weather and pavement condition information. The information from these sensors is used to determine the current conditions on and near the roads. This information is used to conduct roadway related maintenance and for ensuring safe driving conditions. The current weather information and forecasts based on this data are used for the allocation of resources such as the timing of operations such as snow removal, the selection of the right amount of materials such as salt for ice removal from roads, and the mobilization of maintenance personnel and equipment efficiently. These decisions help in running the organization in an efficient and cost effective manner.

During the winter months, the condition of the roads may get hazardous if the proper amount of salt is not applied at the right time. Information about weather and road conditions is obtained from the data reported by the RWIS sensors. This makes it important to keep these sensors in working order and ensure the readings are accurate. The maintenance of RWIS units is expensive and is done manually. Routine re-calibrations and maintenance checks are needed to ensure the proper working of these sensors. It would be beneficial if there existed an automated system that could detect malfunctions in real time and alert the maintenance personnel. We propose to use Machine Learning (ML) methods to form models to use to create such a system.

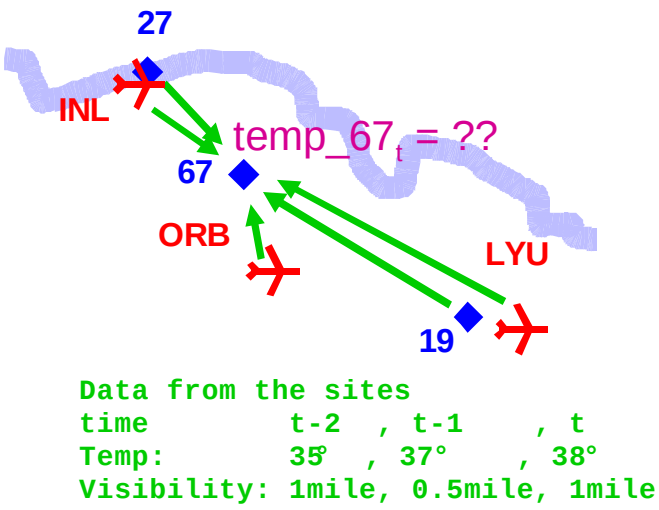

Figure 1.1: Predicting temperature value at RWIS site 67 for the time *t* using weather data from nearby sensors.

To detect a malfunctioning sensor using ML methods, we propose to observe the sensor's output over a period of time to determine any significant and/or systemic variations from the actual conditions present that might indicate a sensor malfunction. Figure 1.1 shows an overview of our proposed approach. This method works by predicting a sensor value at a site, for example site 67 from the Figure 1.1, which is then compared with the actual sensor reading to detect malfunctions. To predict the temperature value for site 67 at a given time *t*, we can use the current and a couple of previous hours of weather data, such as temperature and visibility, obtained from a set of sites that are located close to site 67. Figure 1.1 shows the nearby sites whose data correlate with the weather conditions at site 67. We attempt to build models for a site using ML methods which are used to predict a value at the site. To build such a model, ML methods require historical weather data obtained from the site and its nearby sites to learn the weather patterns. The model for site 67 uses the temperature and visibility information from the nearby sites to predict the temperature value that will be seen at site 67 at future times.

1.1 Building Models of Sensor Data using Machine Learning

Machine Learning (ML) methods build models based on previous observations which can then be used to predict new data. The model built is a result of a learning process that extracts useful information about the data generation process of the system using the previous observations. ML methods take a set of data corresponding to the process (in this case the weather at a sensor) and construct a model of that process in a variety of ways to predict that process. The resulting model can be applied to future data to attempt to predict sensor values. The resulting predictions can then be compared to sensor values reported and in cases where there are significant deviations, these sensors can be flagged as possibly malfunctioning.

We propose to use a variety of ML methods such as classification methods (e.g., J48 Decision Trees, Naive Bayes and Bayesian Networks), regression methods (e.g., Linear Regression, Least Median Squares) and Hidden Markov Models to try to predict this data. In all cases we are attempting to identify cases where sensors appear to have failed or are malfunctioning.

Classification algorithms label a given observation into one of a set of possible distinct categories. For example, we could take information about the recent temperature and humidity at a site and nearby sites as well as precipitation at the nearby sites and attempt to predict whether it is raining at that site. Raining or not raining would be the labels for this problem. Models built might be in the form of decision trees, lists of rules, neural networks, etc.

Regression algorithms develop a model for a system by finding equations that predict a continuous-valued result from the measured input values of a given observation. For example, we might use information about the temperature and humidity at other sites to

try to predict the temperature value at a site.

HMMs predict values that are produced by a system in the form of a sequence generated over time. For example, the sequence of values of raining or not raining at a site over the course of a 24 hour period can be viewed as a time sequence. HMMs use the information from the value observed one step previously to predict the present value. HMMs capture time sequence information in the form of a graph where states represent information about the world (possibly including information that cannot be observed) and transitions between states which can emit the symbols (in this case, raining or not raining) which we observe. The process of determining the probabilities of emitting symbols and the probabilities of transitions between states is the basis of HMM learning.

1.2 Thesis Statement

In this thesis we propose to build models that can predict weather conditions at a given RWIS location using the current information from that location and surrounding locations (see Figure 1.1). We use ML algorithms including classification, regression and HMM methods to build the models to predict weather conditions at a selected sample of RWIS sites in the state of Minnesota. We use these predicted values and compare them with the values reported by the RWIS sensors to identify possible malfunctions. Of all the weather conditions reported by RWIS units we focus on predicting temperature, precipitation type and visibility. We hypothesize that the models we build can accurately detect deviations in the expected sensor readings that will allow us to identify sensor malfunctions.

1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 presents the background for this thesis with a detailed description of the weather data collection systems present and the sensors they use. It describes the basic concepts of machine learning and presents various ML

algorithms and HMMs used. Chapter 3 discusses the use of ML methods in detecting RWIS sensor malfunctions. It presents the process of selecting the sites used, feature representations used by ML algorithms and HMMs, and the general methodology used by the ML methods in predicting sensor values. Chapter 4 presents the experiments done to predict temperature, precipitation type and visibility using ML algorithms, and the results obtained from these experiments. Chapter 5 discusses research related to this work. Chapter 6 discusses extensions that can be made to the the proposed system. Chapter 7 presents conclusions and summarizes the work done in this thesis.

Chapter 2

Background

This chapter presents the background for this thesis. In the first section we introduce the two weather data collection systems, the Road Weather Information System (RWIS) and the Automated Weather Observing System (AWOS). We describe these sensors and the data format they use for recording values. In the second section we discuss the basic concepts of machine learning and its use in data mining problems. We then present various machine learning algorithms used in this thesis.

2.1 Description of Sensors

Two major automated systems are present in the United States that collect, process and distribute meteorological data, namely the Road Weather Information System (RWIS) [Aurora; Boselly et al., 1993] and the Automated Weather Observing System (AWOS) [FAA, 2006]. RWIS units are used by state and local roadway maintenance organizations for diverse roadway related operations. RWIS units have sensors that collect weather and pavement condition information. These units are generally installed on highways. The weather information gathered is used for understanding the current conditions on roadways at a specific location.

AWOS units are operated and controlled by Federal Aviation Administration (FAA). AWOS systems are installed at airports across the country. The meteorological sensors located on AWOS units measure and distribute weather information at the airports that is used by pilots and airport administrators. This information is used to keep the runways in proper condition, determine flight plans, and for landing and takeoff of airplanes. At

present, a newer version of AWOS called the Automated Surface Observing System (ASOS) is also being used. A detailed description of these sensor systems, RWIS and AWOS, is given in the following sections.

2.1.1 The Road Weather Information System

The Road Weather Information System (RWIS) [Aurora; Boselly et al., 1993] is used for collection, transmission, model generation, and distribution of weather and road condition information. An RWIS is a collection of various systems that work together. The systems that form an RWIS are meteorological sensors, data communication equipment, weather profiles, site specific models, forecast and prediction algorithms, data processing systems and display systems to interface with humans.

The component of RWIS that collects weather data is called the Environmental Sensor Station (ESS) [Manfredi et al., 2005]. The ESS stations are placed at strategic locations in the road network, usually on state highways, in a grid-wise manner. A typical ESS consists of a tower, two road sensors embedded in and below the road for measurement of pavement conditions, an array of weather sensors located on the tower for meteorological observations, a Remote Processing Unit (RPU) for data collection and storage, and remote communication hardware that connects the ESS to a central server which is present at a maintenance center. The road sensors measure road surface temperature, surface condition (dry, wet and snow), water-film level and freezing temperature based on residual salt content on the road. The weather sensors measure air temperature, dew point, precipitation (type, intensity and rate), amount of precipitation accumulation, wind speed and direction, air pressure, visibility, relative humidity and solar radiation.

The RPUs are able to collect the raw data sent by the various sensors on the ESS and store it. They are not designed to process the data that is collected and thus the data is

transmitted to a central server present at the maintenance center using the remote communication hardware. The communication between the RPU and the central server is done using radio signals.

The central server located at the highway maintenance centers has processing capabilities and performs data-related tasks such as communication with the RPUs, and collection, archiving and distribution of data. A set of site-specific weather models and data processing algorithms are loaded onto the central server. The central server uses these models and its data processing capabilities to come up with forecasts. The central server has a number of displays for human interaction with the RWIS system.

State and local organizations dealing with highway maintenance, such as the Minnesota Department of Transportation (Mn/DOT[1]), use RWIS for maintenance of roads and to ensure safe driving conditions in all seasons. The data collected and forecasts made by RWIS are used for the allocation of resources, timing of operations such as repairs, selection of the right amount of materials such as salt in case of ice removal, and mobilizing personnel and equipment. All of these decisions help in running the organization in a cost effective manner. The current weather and road conditions are relayed to motorists to help them in planning their trips and estimating travel times. In addition to the uses mentioned above, the data is also shared with various government, commercial and university related sources.

Mn/DOT maintains 93 RWIS sites spread all across the state of Minnesota. Figure 2.1 shows the location of the 79 of the 93 RWIS sites present in Minnesota. The remaining 14 sites added recently are not included in our original map. Each RWIS site has a specific number, assigned by Mn/DOT, associated with it. The latitude and longitude of each RWIS site shown in the Figure 2.1 are given in Table A.1 in the appendix.

[1.] http://www.dot.state.mn.us/

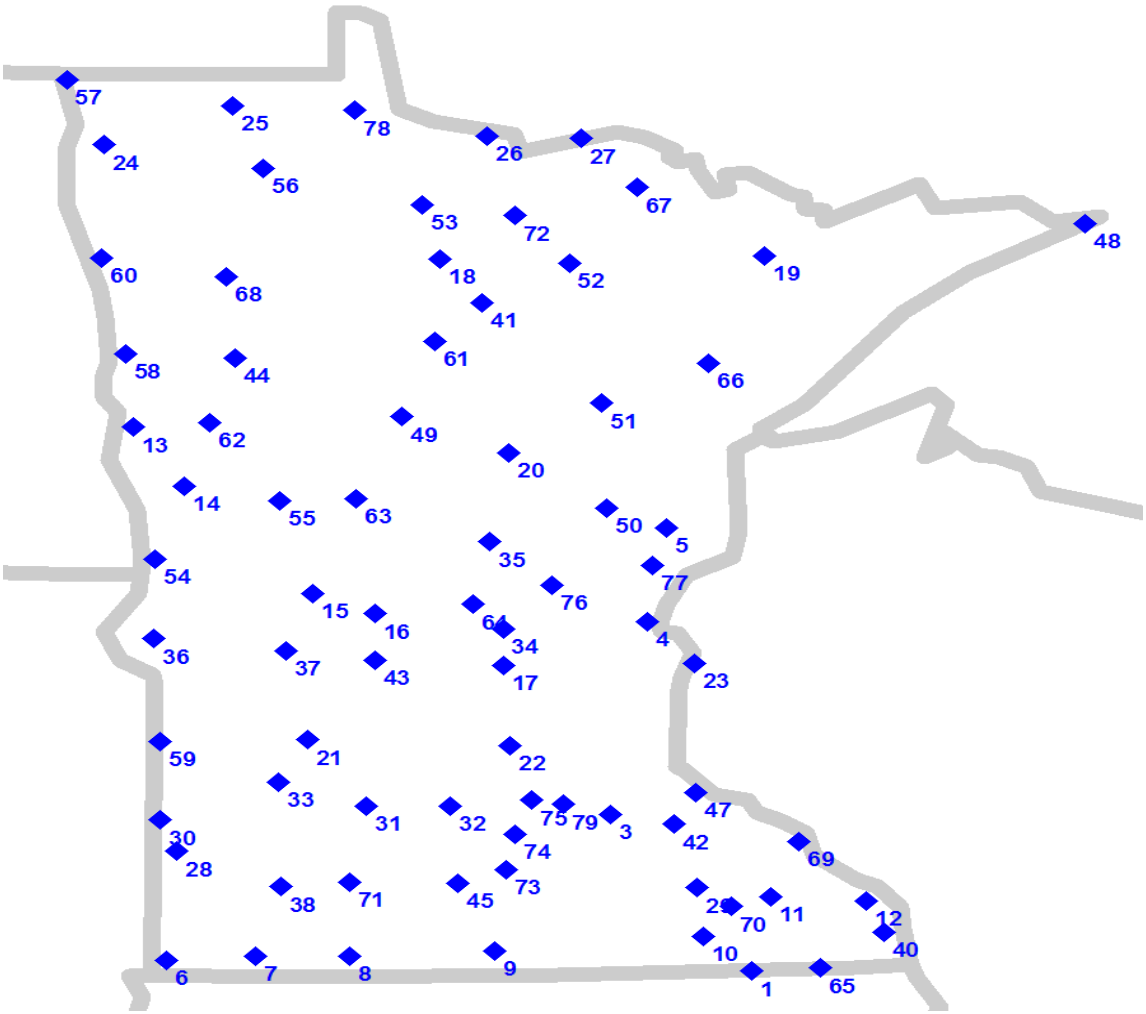

Figure 2.1: RWIS sites in Minnesota

2.1.1.1 Data from RWIS sensors

RWIS sensors report observed values every 10 minutes, resulting in 6 records per hour. Greenwich Mean Time (GMT) is used for recording the values. The meteorological conditions reported by RWIS sensors are air temperature, surface temperature, dew point, visibility, precipitation, the amount of precipitation accumulation, wind speed and wind direction. Following is a short description of these variables along with the format they follow.

(i) Air Temperature: Air temperature is recorded in Celsius in increments of one hundredth of a degree, with values ranging from -9999 to 9999 and a value of 32767 indicating an error. For example, a temperature of 10.5 degree Celsius is reported as 1050.

(ii) Surface Temperature: Surface temperature is the temperature measured near the road surface. It is recorded in the same format as air temperature.

(iii) Dew Point: Dew point is defined as the temperature at which dew forms. It is recorded in the same format as air temperature.

(iv) Visibility: Visibility is the maximum distance to which it is possible to see without any aid. Visibility reported is the horizontal visibility recorded in one tenth of a meter with values ranging from 00000 to 99999. A value of -1 indicates an error. For example, a visibility of 800.2 meters is reported as 8002.

(v) Precipitation: Precipitation is the amount of water in any form that falls to earth. Precipitation is reported using three different variables, *precipitation type*, *precipitation intensity* and *precipitation rate*. A coded approach is used for reporting precipitation type and intensity. The codes used and the information they convey are given in Table 2.1.

The precipitation type gives the form of water that reaches the earth's surface. Precipitation type with a code of 0 indicates no precipitation, a code of 1 indicates the presence of some amount of precipitation but the sensor fails to detect the form of the water, a code of 2 represents rain, and the codes 3, 41 and 42 respectively represent snowfall with an increase in intensity.

The precipitation intensity is used to indicate how strong the precipitation is when present. When no precipitation is present then intensity is given by code 0. As we move from codes 1 to 4 they indicate an increase in intensity of precipitation. Codes 5 and 6 are

Table 2.1: Codes used for reporting precipitation type and precipitation intensity by RWIS Sensors.

Code	*Precipitation Intensity*	*Code*	*Precipitation Type*
0	No precipitation	0	None
1	Precipitation detected, not identified	1	Light
2	Rain	2	Slight
3	Snow	3	Moderate
41	Moderate Snow	4	Heavy
42	Heavy Snow	5	Other
		6	Unknown

used to indicate an intensity that cannot be classified into any of previous codes and in cases when the sensor is unable to measure the intensity respectively. A value of 255 for precipitation intensity indicates either an error or absence of this type of sensor.

Precipitation rate is measured in millimeters per hour with values ranging from 000 to 999 except for a value of -1 that indicates either an error or absence of this type of sensor.

(vi) Precipitation Accumulation: Precipitation accumulation is used to report the amount of water falling in millimeters. Values reported range from 00000 to 99999 and a value of -1 indicating an error. Precipitation accumulation is reported for the last 1, 3, 6, 12 and 24 hours.

(vii) Wind Speed: Wind speed is recorded in tenths of meters per second with values ranging from 0000 to 9999 and a value of -1 indicating an error. For example, a wind speed of 2.05 meters/second is reported as 205.

(viii) Wind Direction: Wind direction is reported as an angle with values ranging from

0 to 360 degrees. A value of -1 indicates an error.

(ix) Air Pressure: Air pressure is defined as the force exerted by the atmosphere at a given point. The pressure reported is the pressure when reduced to sea level. It is measured in tenths of a millibar and the values reported range from 00000 to 99999. A value of -1 indicates an error. For example, air pressure of 1234.0 millibars is reported as 12340.

2.1.2 The Automated Weather Observing System

The Automated Weather Observing System (AWOS) is a collection of systems including meteorological sensors, data collection system, centralized server and displays for human interaction which are used to observe and report weather conditions at airports in order to help pilots in maneuvering the aircraft and airport authorities for proper working of airports and runways.

AWOS sensors [AllWeatherInc] are placed at strategic locations such as runways at the airport. These sensors record hourly weather conditions and are capable of reporting air temperature, visibility, dew point, wind speed and direction, precipitation type and amount, humidity, air pressure and cloud cover. The sensors are connected to a powerful computer that analyzes the data gathered and broadcasts weather reports. The information collected from the AWOS sensors are used by pilots, air traffic control and maintenance personnel for safe operation of runways and for determining flight routes. The AWOS system provides update to pilots approaching an airport using a non-directional beacon. The meteorological information gathered is also used by the National Weather Service for forecasting and other weather services. There are over 600 AWOS sites located across the US according to the FAA. The sites are named in accordance with the code assigned to the airport they are located in, for example the city of Duluth in Minnesota that has an airport with DLH has its AWOS sensor named KDLH.

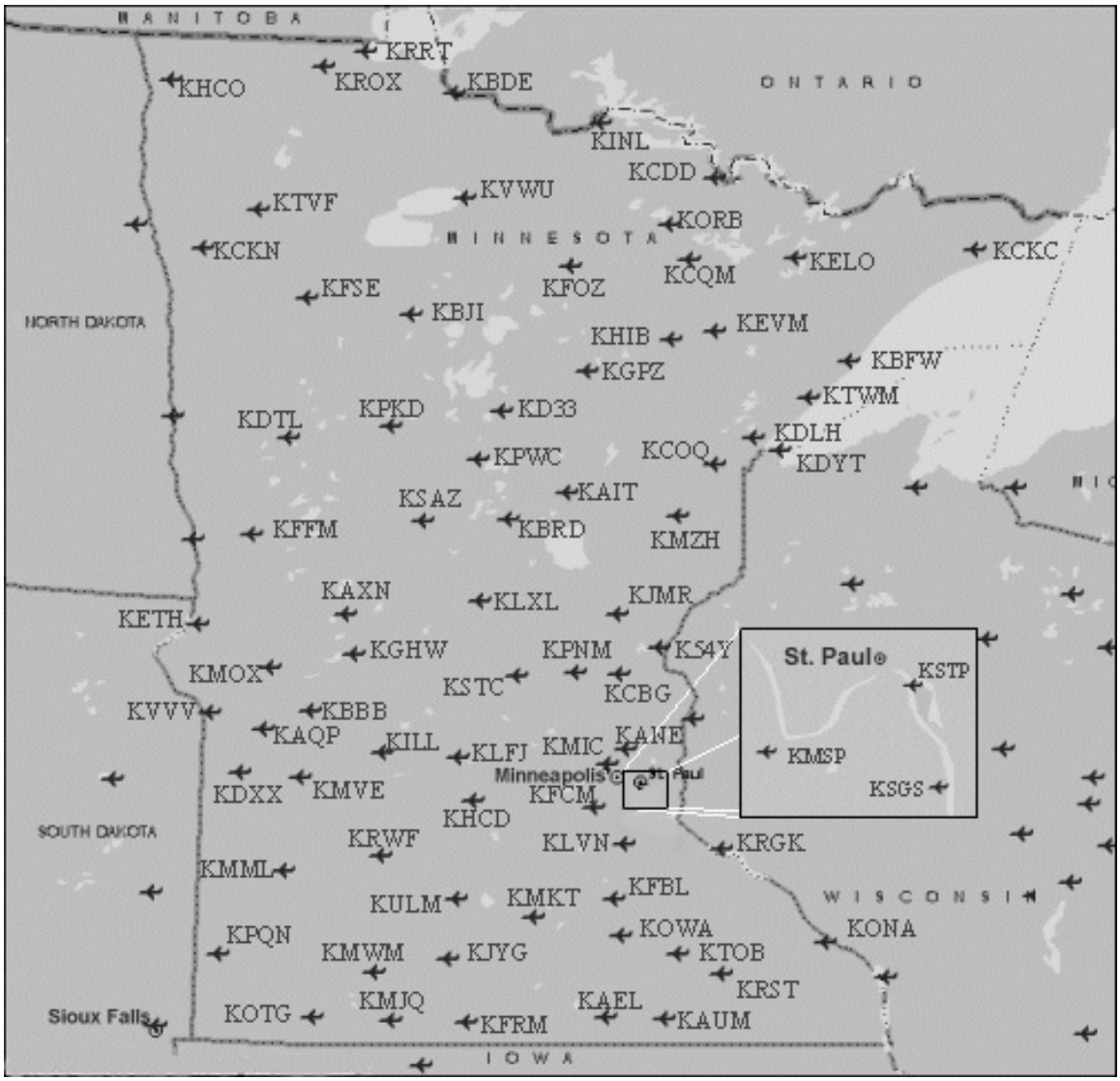

Figure 2.2: AWOS Sites in Minnesota

Figure 2.2 shows the location of AWOS sites across the state of Minnesota and Table A.2 gives the latitude and longitude of these sites.

2.1.2.1 Data from AWOS sensors

AWOS sensors report values on an hourly basis. The local time is used in reporting. For example, the sites in Minnesota use Central Time (CT). AWOS sensors report air

temperature, dew point, visibility, weather conditions in a coded manner, air pressure and wind information. Following is a description of these variables.

(i) Air Temperature: The air temperature value is reported as integer values in Fahrenheit ranging from -140 F to 140 F.

(ii) Dew Point: The dew point temperature is reported in the same format as air temperature.

(iii) Visibility: The visibility is reported as a set of values ranging from 3/16 mile to 10 miles. These values can be converted into floating point numbers for simplicity.

(iv) Weather Code: AWOS uses a coded approach to represent the current weather condition. 80 different possible codes are listed to indicate various conditions. Detailed information about these codes is given in the document titled "Data Documentation of Data Set 3283: ASOS Surface Airways Hourly Observations" published by the National Climatic Data Center [NCDC, 2005].

(v) Air Pressure: The air pressure is reported in tenth of a millibar increments. For example, an air pressure of 123.4 millibars is reported as 1234.

(vi) Wind Speed and Wind Direction: AWOS encodes wind speed and direction into a single variable. Wind speed is measured in knots ranging from 0 knots to 999 knots. Wind direction is measured in degrees ranging from 0 to 360 in increments of 10. The single variable for wind has five digits with the first two representing direction and the last three representing speed. For example, the value for a wind speed of 90 knots and direction of 80 degrees will be 80090.

In this work we propose to predict malfunctions of a RWIS sensor using other nearby RWIS and AWOS sensors. A sensor is said to be malfunctioning when the values

reported by it deviate from the actual conditions present. We will be applying various machine learning algorithms and Hidden Markov Models in order to find a model that can detect significant variations in the readings obtained from a sensor. The following section presents a brief description of the fields of machine learning and data mining along with a description of the machine learning algorithms used in this thesis.

2.2 Machine Learning

Learning can be defined in general as a process of gaining knowledge through experience. We humans start the process of learning new things from the day we are born. This learning process continues throughout our life where we try to gather more knowledge and try to improve what we have already learned through experience and from information gathered from our surroundings.

Artificial Intelligence (AI) is a field of computer science whose objective is to build a system that exhibits intelligent behavior in the tasks it performs. A system can be said to be intelligent when it has learned to perform a task related to the process it has been assigned to without any human interference and with high accuracy. Machine Learning (ML) is a sub-field of AI whose concern is the development, understanding and evaluation of algorithms and techniques to allow a computer to learn. ML intertwines with other disciplines such as statistics, human psychology and brain modeling. Human psychology and neural models obtained from brain modeling help in understanding the workings of the human brain, and especially its learning process, which can be used in the formulation of ML algorithms. Since many ML algorithms use analysis of data for building models, statistics plays a major role in this field.

A process or task that a computer is assigned to deal with can be termed the knowledge or task domain (or just the domain). The information that is generated by or obtained from the domain constitutes its knowledge base. The knowledge base can be represented

in various ways using Boolean, numerical, and discrete values, relational literals and their combinations. The knowledge base is generally represented in the form of input-output pairs, where the information represented by the input is given by the domain and the result generated by the domain is the output. The information from the knowledge base can be used to depict the data generation process (i.e., output classification for a given input) of the domain. Knowledge of the data generation process does not define the internals of the working of the domain, but can be used to classify new inputs accordingly.

As the knowledge base grows in size or gets complex, inferring new relations about the data generation process (the domain) becomes difficult for humans. ML algorithms try to learn from the domain and the knowledge base to build computational models that represent the domain in an accurate and efficient way. The model built captures the data generation process of the domain, and by use of this model the algorithm is able to match previously unobserved examples from the domain.

The models built can take on different forms based on the ML algorithm used. Some of the model forms are decision lists, inference networks, concept hierarchies, state transition networks and search-control rules. The concepts and working of various ML algorithms are different but their common goal is to learn from the domain they represent.

ML algorithms need a dataset, which constitutes the knowledge base, to build a model of the domain. The dataset is a collection of instances from the domain. Each instance consists of a set of attributes which describe the properties of that example from the domain. An attribute takes in a range of values based on its attribute type, which can be discrete or continuous. Discrete (or nominal) attributes take on distinct values (e.g., *car = Honda, weather = sunny*) whereas continuous (or numeric) attributes take on numeric values (e.g., *distance = 10.4 meters, temperature = 20ºF*).

Each instance consists of a set of input attributes and an output attribute. The input attributes are the information given to the learning algorithm and the output attribute contains the feedback of the activity on that information. The value of the output attribute is assumed to depend on the values of the input attributes. The attribute along with the value assigned to it define a feature, which makes an instance a feature vector. The model built by an algorithm can be seen as a function that maps the input attributes in the instance to a value of the output attribute.

Huge amounts of data may look random when observed with the naked eye, but on a closer examination, we may find patterns and relations in it. We also get an insight into the mechanism that generates the data. Witten & Frank [2005] define data mining as a process of discovering patterns in data. It is also referred to as the process of extracting relationships from the given data. In general data mining differs from machine learning in that the issue of the efficiency of learning a model is considered along with the effectiveness of the learning. In data mining problems, we can look at the data generation process as the domain and the data generated by the domain as the knowledge base. Thus, ML algorithms can be used to learn a model that describes the data generation process based on the dataset given to it. The data given to the algorithm for building the model is called the training data, as the computer is being trained to learn from this data, and the model built is the result of the learning process. This model can now be used to predict or classify previously unseen examples. New examples used to evaluate the model are called a test set. The accuracy of a model can be estimated from the difference between the predicted and actual value of the target attribute in the test set.

Predicting weather conditions can also be considered as an example of data mining. Using the weather data collected from a location for a certain period of time, we obtain a model to predict variables such as temperature at a given time based on the input to the model. As weather conditions tend to follow patterns and are not totally random, we can use current meteorological readings along with those taken a few hours earlier at a location and also readings taken from nearby locations to predict a condition such as the

temperature at that location. Thus, the data instances that will be used to build the model may contain present and previous hour's readings from a set of nearby locations as input attributes. The variable that is to be predicted at one of these locations for the present hour is the target attribute. The type and number of conditions that are included in an instance depend on the variable we are trying to predict and on the properties of the ML algorithm used.

WEKA [Witten & Frank, 2005], for Waikato Environment for Knowledge Analysis, is a collection of various ML algorithms, implemented in Java, that can be used for data mining problems. Apart from applying ML algorithms on datasets and analyzing the results generated, WEKA also provides options for pre-processing and visualization of the dataset. It can be extended by the user to implement new algorithms.

Suppose that we want to predict the present temperature at a site *C* (see Figure 2.3). To do this we use eight input attributes: the previous two hours temperature together with the present hour temperature at *C* and two nearby locations *A* and *B*. The output attribute is the present hour temperature at *C*. Let *temp_<site><hour>* denote temperature taken at hour <hour> at location <site>, then the data instance will take the form,

$$temp\_A_{t-2}, temp\_A_{t-1}, temp\_A_t, temp\_B_{t-2}, temp\_B_{t-1}, temp\_B_t, temp\_C_{t-2}, temp\_C_{t-1}, temp\_C_t$$

with the last attribute, $temp\_C_t$, being the output attribute. We will refer to this example as 'our weather example' in the following sections in this chapter.

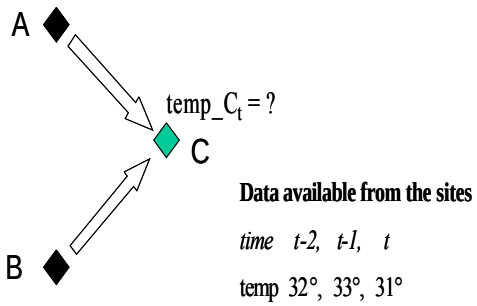

Figure 2.3: Using data from nearby sites to predict temperature for the location C.

ML algorithms can be broadly classified into two groups, classification and regression algorithms. We describe these two types of classifications and some of the ML algorithms from each of these groups.

2.2.1 Classification Algorithms

Algorithms that classify a given instance into a set of discrete categories are called classification algorithms. These algorithms work on a training set to come up with a model or a set of rules that classify a given input into one of a set of discrete output values. Most classification algorithms can take inputs in any form, discrete or continuous although some of the classification algorithms require all of the inputs also to be discrete. The output is always in the form of a discrete value. Decision trees and Bayes nets are examples of classification algorithms.

To be able to apply classification algorithms on our weather example we need to convert the output attribute into classes. This is generally done by discretization, which is the process of dividing a continuous variable into classes. Discretization can be done in many ways, a simple approach would be to divide the temperature into ranges of 5 degrees and giving each range a name or by using entropy-based algorithms [Fayyad & Irani, 1993; Dougherty et al., 1995]. Inputs attributes can be left as continuous if the algorithm deals with them or they can be converted into discrete values depending on the algorithm.

We describe in detail the classification algorithms that have been used in this thesis in the sub-sections below.

2.2.1.1 The J48 Decision Tree Algorithm

J48 is a decision tree learner based on C4.5 [Quinlan, 1993]. C4.5 is an update of the ID3

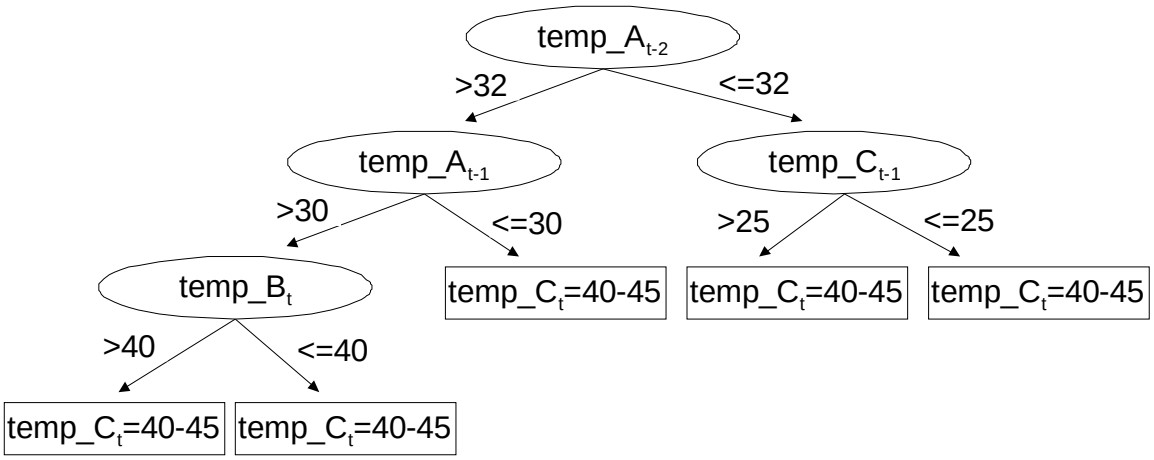

Figure 2.4: A Decision Tree to predict the current temperature at site *C* based on temperature readings taken from a set of nearby sites.

algorithm [Quinlan, 1986]. We describe here the ID3 algorithm.

A decision tree classifies a given instance by passing it through the tree starting at the top and moving down until a leaf node is reached. The value at that leaf node gives the predicted output for the instance. At each node an attribute is tested and the branches from the node correspond to the values that attribute can take. When the instance reaches a node, the branch taken depends on the value it has for the attribute being tested at the node. A decision tree that can be used to predict the present hour temperature for site *C* in our weather example is given Figure 2.4. So if we were to classify an instance in our weather example with this tree we would start at the root node that tests the attribute *temp_*A_{t-2} and based on the value taken by this attribute in the given instance we will take the left or right branch. When we reach a node after taking a branch, the attribute associated with it is tested and the corresponding branch taken until we reach a leaf node, which gives the value taken for the output attribute *temp_*C_t.

The ID3 algorithm builds a decision tree based on the set of training instances given to it. It takes a greedy top-down approach for the construction of the tree, starting with the creation of the root node. At each node the attribute that best classifies all the training instances that have reached that node is selected as the test attribute. At a node only those

attributes are considered which were not used for classification at other nodes above it in the tree. To select the best attribute at a node, the *information gain* for each attribute is calculated and the attribute with the highest information gain is selected. Information gain for an attribute is defined as the reduction in *entropy* caused by splitting the instances based on values taken by the attribute. The information gain for an attribute A at a node is calculated using

$$InformationGain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \left(\frac{|S_v|}{|S|} Entropy(S) \right),$$

where S is the set of instances at that node and $|S|$ is its cardinality, S_v is the subset of S for which attribute A has value v, and entropy of the set S is calculated as

$$Entropy(S) = \sum_{i=1}^{numclasses} - p_i \log_2 p_i ,$$

where p_i is the proportion of instances in S that have the i^{th} class value as output attribute.

A new branch is added below the node for each value taken by the test attribute. The training instances that have the test attribute value associated with the branch taken are passed down the branch, and this subset of training instances is used for the creation of further nodes. If this subset of training instances has the same output class value then a leaf is generated at the branch end, and the output attribute is assigned that class value. In the case where no instances are passed down a branch then a leaf node is added at the branch end that assigns the most common class value in the training instances to the output attribute. This process of generating nodes is continued until all the instances are correctly classified or all the attributes have been used or when its not possible to divide the examples.

Extensions were added to the basic ID3 algorithm to (1) deal with continuous valued attributes, (2) deal with instances that have missing attribute values and to (3) prevent overfitting the data (explained below).

When a discrete valued attribute is selected at a node the number of branches formed is

equal to the number of possible values taken by the attribute. In the case of a continuous valued attribute two branches are formed based on a threshold value that best splits the instances into two. For example, in Figure 2.4 the attribute at the root node, $temp\_A_{t-2}$, has a threshold value of 32. The threshold is the selected as the value of the attribute that maximizes the information gain of the given training instances. Fayyad & Irani [1993] extended this approach to split a continuous-valued attribute into more than two intervals.

There may arise cases where an instance has no value for an attribute (i.e., missing values) or has an unknown attribute value. The missing value can be replaced by the most common value for that attribute among the training instances that reach the node where this attribute is tested. In C4.5, the probability for each possible value taken by the attribute with missing value is calculated, based on the number of times it is seen in the training instances at a node. The probability values are then used for calculation of information gain at the node.

In the ID3 algorithm, sometimes due to too small of a training set being used, the tree built correctly classifies the training instances but fails when applied on the entire distribution of data because it focuses on the spurious correlation in the data when the remaining amount of data is small; this is know as *overfitting*. To avoid overfitting, C4.5 uses a technique called rule-post pruning. In rule post-pruning, after the tree is built, it is converted into a set of rules. For example, the rule generated for leftmost path of the tree in Figure 2.4 is

IF ($temp\_A_{t-2} > 32$ AND $temp\_A_t > 30$ AND $temp\_B_t > 40$)
THEN $temp\_C_t = 40\text{-}45$.

From each rule generated for the tree, those antecedents are pruned (i.e., removed) which do not reduce the accuracy of the model. Accuracy is measured based on the instances present in the validation set, which is a subset of the training set not used for building the model.

2.2.1.2 Naive Bayes

Naive Bayes [Good, 1965; Langley et al., 1992] is a simple probabilistic classifier based on Bayes' rule. The naive Bayes algorithm builds a probabilistic model by learning the conditional probabilities of each input attribute given a possible value taken by the output attribute. This model is then used to predict an output value when we are given a set of inputs. This is done by applying Bayes' rule on the conditional probability of seeing a possible output value when the attribute values in the given instance are seen together. Before describing the algorithm we first define the Bayes' rule.

Bayes' rule states that

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)},$$

where *P(A|B)* is defined as the probability of observing *A* given that *B* occurs. *P(A|B)* is called posterior probability, and *P(B|A)*, *P(A)* and *P(B)* are called prior probabilities. Bayes' theorem gives a relationship between the posterior probability and the prior probability. It allows one to find the probability of observing *A* given *B* when the individual probabilities of *A* and *B* are known, and the probability of observing *B* given *A* is also known.

The naive Bayes algorithm uses a set of training examples to classify a new instance given to it using the Bayesian approach. For an instance, the Bayes rule is applied to find the probability of observing each output class given the input attributes and the class that has the highest probability is assigned to the instance. The probability values used are obtained from the counts of attribute values seen in the training set.

In our weather example, for a given instance with two input attributes $temp\_A_t$ and $temp\_B_t$, with values *a* and *b* respectively, the value v_{MAP} assigned by the naive Bayes algorithm to the the output attribute $temp\_C_t$ is the one that has the highest probability

across all possible values taken by output attribute; this is known as the maximum-a-posteriori (MAP) rule. The probability of the output attribute taking *a* value v_j when the given input attribute values are seen together is given by

$$P(v_j \mid a,b).$$

This probability value as such is difficult to calculate. By applying Bayes theorem on this equation we get

$$P(v_j \mid a,b) = \frac{P(a,b \mid v_j)P(v_j)}{P(a,b)} = P(a,b \mid v_j)P(v_j),$$

where $P(v_j)$ is the probability of observing v_j as the output value, $P(a,b|v_j)$ is the probability of observing input attribute values *a, b* together when output value is v_j. But if the number of input attributes *(a, b, c, d, ...)* is large then we likely will not have enough data to estimate the probability $P(a, b, c, d, ... \mid v_j)$.

The naive Bayes algorithm solves this problem by using the assumption of conditional independence for the all the input attributes given the value for the output. This means it assumes that the values taken by an attribute are not dependent on the values of other attributes in the instance for any given output. By applying the conditional independence assumption, the probability of observing an output value for the inputs can be obtained by multiplying the probabilities of individual inputs given the output value. The probability value $P(a, b \mid v_j)$ can then be simplified as

$$P(a,b \mid v_j) = P(a \mid v_j)P(b \mid v_j),$$

where $P(a \mid v_j)$ is the probability of observing the value *a* for the attribute $temp\_A_t$ when output value is v_j. Thus the probability of an output value v_j to be assigned for the given input attributes is

$$P(v_j \mid a,b) = P(v_j)P(a \mid v_j)P(b \mid v_j).$$

Learning in the Naive Bayes algorithm involves finding the probabilities of $P(v_j)$ and $P(a_i|v_j)$ for all possible values taken by the input and output attributes based on the training set provided. $P(v_j)$ is obtained from the ratio of the number of time the value v_j is

seen for the output attribute to the total number of instances in the training set. For an attribute at position i with value a_i, the probability $P(a_i|v_j)$ is obtained from the number of times a_i is seen in the training set when the output value is v_j.

The naive Bayes algorithm requires all attributes in the instance to be discrete. Continuous valued attributes have to be discretized before they can be used. Missing values for an attribute are not allowed, as they can lead to difficulties while calculating the probability values for that attribute. A common approach to deal with missing values is to replace them by a default value for that attribute.

2.2.1.3 Bayesian Belief Networks (Bayes Nets)

The naive Bayes algorithm uses the assumption that the values of all the input attributes are conditionally independent given the value of the output attribute. But there may be cases when assuming conditional independence of all the given inputs, may not lead to appropriate predictions. Bayesian Belief Networks or Bayes Nets introduce the idea of applying conditional independence on a certain number of inputs rather than on all of them. This notion avoids the global assumption of conditional independence while maintaining some amount of conditional independence among the inputs.

A Bayesian Belief Network [Friedman et al., 1997; Pearl, 1988] is a directed acyclic graphical network model that gives the joint probability distribution for a set of attributes. Each attribute in the instance is represented in the network in the form of a node. In the network a directed connection from node X to node Y is made when X is a parent of Y which means that there is a dependence relation of Y on X, or in other words X has an influence on Y. Thus in this network an attribute at a node is conditionally independent of its non-dependents in the network given the state of its parent nodes. These influences are represented by conditional probabilities, which gives the probability of a value at a node that is conditional on the value of its parents. These probability values for a node are

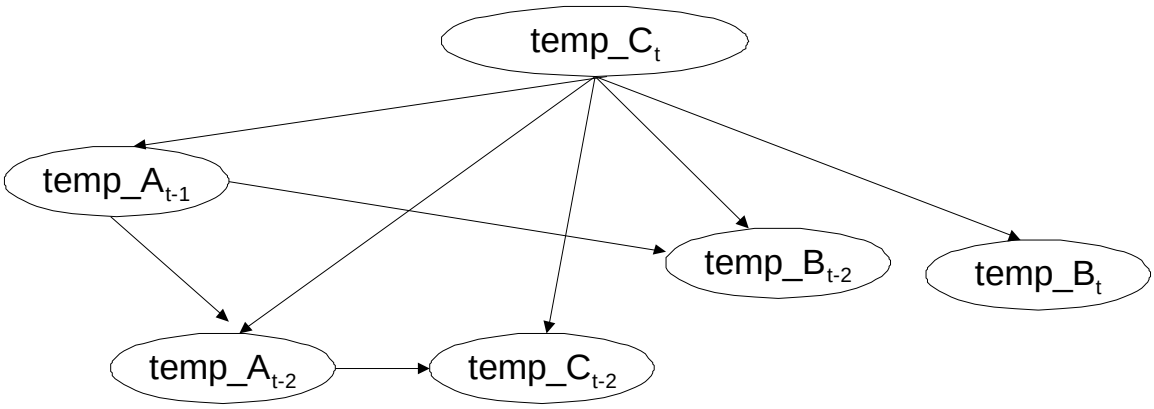

Figure 2.5: A Bayesian network to predict temperature *temp_C$_t$* at a site. The arrows represent a direct relation between nodes. Each node is associated with a CPT.

arranged in a tabular form called a Conditional Probability Table (CPT). In the case of nodes with no parents, the CPT gives the distribution of the attribute at that node.

When a node is connected to a set of nodes, which are one step above in the hierarchy, these parent nodes have an influence on its behavior. This node is not affected by other nodes present in the given pool of nodes. It means the node is conditionally independent of all non-parent nodes when given its parents. The nodes which are more than one step above in hierarchy, that is parents of parents of a node, are not considered as directly influencing the node, as these nodes affect the nodes which are parents to the node in question and thus indirectly influence it. Thus the parents are considered for calculating the joint probability, as only the direct parents of a node influence the conditional probabilities at this node. Using conditional independence between nodes, the joint probability for a set of attribute values $y_1,y_2,..,y_n$ represented by the nodes $Y_1,Y_2,...,\ Y_n$ is given by

$$P(y_1,.....,y_n) = \prod_{i=1}^{n} P(y_i \mid Parents(Y_i)),$$

where $Parents(Y_i)$ are the immediate parents of node Y_i. The probability values can be obtained directly from the CPTs associated with the node.

A Bayesian network requires that both input and output attributes be discrete. A simple Bayesian network for predicting temperature at a site in our weather example, using only a few of the input instances, is shown in Figure 2.5. Each node in the tree is associated with a CPT. For example, the CPT for the node $temp\_A_{t-2}$ will contain the probability of each value taken by it when all possible values for $temp\_A_{t-1}$ and $temp\_C_t$ (i.e., its parents) are seen together. For a given instance, the Bayesian network can be used to determine the probability distribution of the target class by multiplying all the individual probabilities of values taken up by the individual nodes. The class value that has the highest probability is selected. The probability of a class value taken by the output attribute $temp\_C_t$ for the given input attributes, using parental information of nodes from the Bayesian network in Figure 2.5 is

$$P(temp\_C_t|temp\_A_{t-1}, temp\_A_{t-2}, temp\_B_t, temp\_B_{t-2}, temp\_C_{t-2}) = P(temp\_C_t)*P(temp\_A_{t-1}|temp\_C_t)*P(temp\_A_{t-2}|temp\_A_{t-1},temp\_C_t)*P(temp\_B_t|temp\_C_t)*P(temp\_B_{t-2}|temp\_A_{t-1},temp\_C_t)*P(temp\_C_{t-2}|temp\_A_{t-2},temp\_C_t).$$

Learning in Bayes' Nets from a given training set involves finding the best performing network structure and calculating CPTs. To build the network structure, we start by assigning each attribute a node. Learning the network connections involves moving though the set of possible connections and finding the accuracy of the network for the given training set. The accuracy of the network can be determined by using a scoring criterion such as the Akaike's Information Criterion [Akaike, 1974], the Minimum Description Criterion [Rissanen, 1978] or the Cross-Validation Criterion. Allen & Greiner [2000] present a brief description of these scoring criterions along with their empirical comparisons. For a network, the CPTs are calculated at each node based on the information obtained from the training set.

The K2 algorithm [Cooper & Herskovits, 1992] can be used to learn the Bayesian network structure. K2 puts the given nodes in an order and then processes one node at a time. It adds an edge to this node from previously added nodes only when the network

accuracy is increased after this addition. When no further connections can be added to the current node that increase the accuracy, the algorithm then moves to another node. This process continues until all nodes have been processed.

When all variables present in the network are seen in the training data, the probability values in the CPTs can be filled by counting the required terms. In the case of training data with missing variables the gradient ascent training [Russel et al., 1995] method can be used to learn values for the CPTs.

2.2.2 Regression Algorithms

Algorithms that develop a model based on equations or mathematical operations on the values taken by the input attributes to produce a continuous value to represent the output are called of regression algorithms. The input to these algorithms can take both continuous and discrete values depending on the algorithm, whereas the output is a continuous value. We describe in detail the regression algorithms that have been used in this thesis below.

2.2.2.1 Linear Regression

The Linear Regression algorithm of WEKA [Witten & Frank, 2005] performs standard least squares regression to identify linear relations in the training data. This algorithm gives the best results when there is some linear dependency among the data. It requires the input attributes and target class to be numeric and it does not allow missing attributes values. The algorithm calculates a regression equation to predict the output (x) for a set of input attributes $a_1,a_2,.....,a_k$. The equation to calculate the output is expressed in the form of a linear combination of input attributes with each attribute associated with its respective weight $w_0,w_1,....,w_k$, where w_1 is the weight of a_1 and a_0 is always taken as the constant 1. An equation takes the form

$$x = w_0 + w_1 a_1 + \ldots\ldots\ldots + w_k a_k .$$

For our weather example the equation learned would take the form

$$temp\_C_t = w_0 + w_{At-2}\, temp\_A_{t-2} + w_{At-1}\, temp\_A_{t-1} + w_{At}\, temp\_A_t + w_{Bt-2}\, temp\_B_{t-2} + w_{Bt-1}\, temp\_B_{t-1} + w_{Bt}\, temp\_B_t + w_{Ct-2}\, temp\_C_{t-2} + w_{Ct-1}\, temp\_C_{t-1} ,$$

where $temp\_C_t$ is value assigned to the output attribute, and each term on the right hand side is the product of the values of the input attributes and the weight associated with each input.

The accuracy of predicting the output by this algorithm can be measured as the absolute difference between the actual output observed and the predicted output as obtained from the regression equation, which is also the error. The weights must be chosen in such as way that they minimize the error. To get better accuracy higher weights must be assigned to those attributes that influence the result the most.

A set of training instances is used to update the weights. At the start, the weights can be assigned random values or all set to a constant (such as 0). For the first instance in the training data the predicted output is obtained as

$$w_0 + w_1 a_1^{(1)} + \ldots\ldots\ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)} ,$$

where the superscript for attributes gives the instance position in the training data. After the predicted outputs for all instances are obtained, the weights are reassigned so as to minimize the sum of squared differences between the actual and predicted outcome. Thus the aim of the weight update process is to minimize

$$\sum_{i=1}^{n} \left(x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)} \right),$$

which is the sum of the squared differences between the observed output for the i^{th} training instance ($x^{(i)}$) and the predicted outcome for that training instance obtained from the linear regression equation.

2.2.2.2 LeastMedSquare

The WEKA LeastMedSquare or Least Median Squares of Regression algorithm [Rousseeuw, 1984] is a linear regression method that minimizes the median of the squares of the differences from the regression line. The algorithm requires input and output attributes to be continuous, and it does not allow missing attribute values. Standard linear regression is applied to the input attributes to get the predict the output. The predicted output x is obtained as

$$w_0 + w_1 a_1^{(1)} + \ldots\ldots\ldots + w_k a_k^{(1)} = \sum_{j=0}^{k} w_j a_j^{(1)} ,$$

where the a_i are input attributes and w_i are the weights associated with them.

In the LeastMedSquare algorithm, using the training data, the weights are updated in such a way that they minimize the median of the squares of the difference between the actual output and the predicted outcome using the regression equation. Weights can be initially set to random values or assigned a scalar value. The aim of the weight update process is to determine new weights to minimize

$$\underset{i}{median}\left(x^{(i)} - \sum_{j=0}^{k} w_j a_j^{(i)} \right),$$

where i ranges from 1 to the number of instances in the training data that is being used, $x^{(i)}$ is the actual output for the training instance i, and the predicted outcome for that training instance is obtained from the regression equation.

2.2.2.3 M5P

The M5P or M5Prime algorithm [Wang & Witten, 1997] is a regression-based decision tree algorithm, based on the M5 algorithm by Quinlan [1992]. M5P is developed using M5 with some additions made to it. We will first describe the M5 algorithm and then the

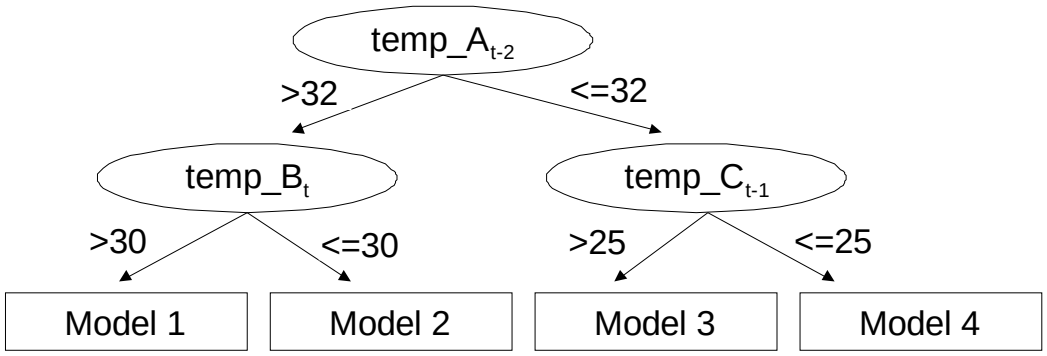

Figure 2.6: A M5 model tree for predicting temperature at a site. The decision taken at a node is based on the test of the attributes mentioned at that node. Each model at a leaf takes the form $w_0+w_1a_1+ .... +w_ka_k$ where k is the number of input attributes.

features added to it in M5P.

M5 builds a tree to predict numeric values for a given instance. The algorithm requires the output attribute to be numeric while the input attributes can be either discrete or continuous. For a given instance the tree is traversed from top to bottom until a leaf node is reached. At each node in the tree a decision is made to follow a particular branch based on a test condition on the attribute associated with that node. Each leaf has a linear regression model associated with it of the form

$$w_o + w_1 a_1 + ... + w_k a_k ,$$

based on some of the input attributes $a_1,a_2,.....,a_k$ in the instance and whose respective weights $w_0,w_1,....,w_k$ are calculated using standard regression (2.2.2.1). As the leaf nodes contain a linear regression model to obtain the predicted output, the tree is called a model tree. When the M5 algorithm is applied on our weather example, the model tree generated will take a form as shown in Figure 2.6.

To build a model tree, using the M5 algorithm, we start with a set of training instances. The tree is built using a divide-and-conquer method. At a node, starting with the root node, the instance set that reaches it is either associated with a leaf or a test condition is chosen that splits the instances into subsets based on the test outcome. A test is based on

an attributes value, which is used to decide which branch to follow. There are many potential tests that can be used at a node. In M5 the test that maximizes the error reduction is used. For a test the expected error reduction is found using

$$\Delta error = stdev(S) - \sum_i \left(\frac{|S_i|}{|S|} stdev(S_i) \right),$$

where S is the set of instance passed to the node, *stdev(S)* is its standard deviation, S_i is the subset of S resulting from splitting at the node with the i^{th} outcome for the test. This process of creating new nodes is repeated until a there are too few instances to proceed further or the variation in the output values in the instances that reach the node is small.

Once the tree has been built, a linear model is constructed at each node. The linear model is a regression equation. The attributes used in the equation are those that are tested or are used in linear models in the sub-trees below this node. The attributes tested above this node are not used in the equation as their effect on predicting the output has already been captured in the tests done at the above nodes. The linear model built is further simplified by eliminating attributes in it. The attributes whose removal from the linear model leads to a reduction in the error are eliminated. The error is defined as the absolute difference between the output value predicted by the model and the actual output value seen for a given instance.

The tree built can take a complex form. The tree is pruned so as to make it simpler without losing the basic functionality. Starting from the bottom of the tree, the error is calculated for the linear model at each node. If the error for the linear model at a node is less than the model sub-tree below then the sub-tree for this node is pruned. In the case of missing values in training instances, M5P changes the expected error reduction equation to

$$\Delta error = \frac{m}{|S|} * \beta(i) * \left[stdev(S) - \sum_i \left(\frac{|S_i|}{|S|} stdev(S_i) \right) \right],$$

where m is the number of instances without missing values for that attribute, S is the set of instances at the node, $\beta(i)$ is the factor multiplied in case of discrete attributes, j takes

values L and R with S_L and S_R being the sets obtained from splitting at that attribute.

2.2.2.4 MultiLayer Perceptron

A MultiLayer Perceptron (MLP) [Bishop, 1995] is a neural network that is trained using backpropagation. MLPs consist of multiple layers of computational units that are connected in a feed-forward way forming a directed connection from lower units to a unit in a subsequent layer. The basic structure of MLP consists of an input layer, one or more hidden layers and one output layer. Units in the hidden layer are termed *hidden* as their output is used only in the network and is not seen outside the network. An MLP with two hidden layers that can be used to predict temperature in our weather example is shown in Figure 2.7. The output from a unit is used as input to units in the subsequent layer. The connection between units in subsequent layers has an associated weight.

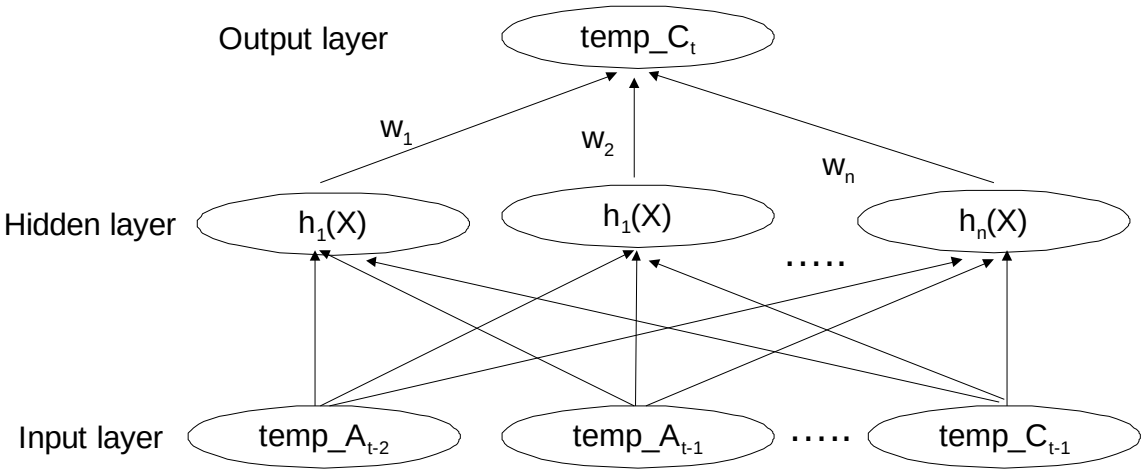

Figure 2.7: A multilayer perceptron with two hidden layers to predict temperature at a site. Each connection is associated with a weight. Hidden and output units are sigmoid units.

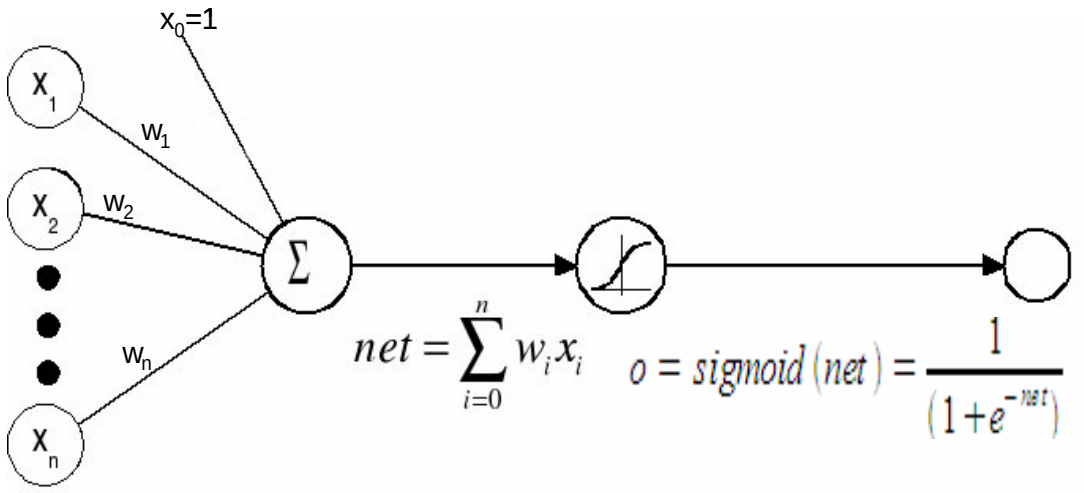

Figure 2.8: A sigmoid unit that takes inputs x_i, w_i the weights associated with the inputs and *sigmoid* the resulting output from the unit.

The hidden and output units are based on sigmoid units. A sigmoid unit calculates a linear combination of its input and then applies the sigmoid function on the result. The sigmoid function, for net input x is

$$sigmoid(x) = \frac{1}{(1+e^{-x})}.$$

The output of a sigmoid unit, *sigmoid(x)*, is a continuous function of its input (x) and is in the range of 0 to 1. A sigmoid unit is shown in Figure 2.8. In addition to the inputs supplied to it, the sigmoid unit also takes in a constant input of 1.

An MLP learns its weights using the backpropagation algorithm [Rumelhart et al., 1986]. The backpropagation algorithm takes a set of training instances for the learning process. For the given feed-forward network, the weights are initialized to small random numbers. Each training instance is passed through the network and the output from each unit is computed. The target output is compared with the output computed by the network to calculate the error and this error value is fed back through the network. To adjust the weights, backpropagation uses gradient descent to minimize the squared error between the target output and the computed output. At each unit in the network, starting from the

output unit and moving down to the hidden units, its error value is used to adjust weights of its connections so as to reduce the error. The weights are updated using

$$w_{ji} = w_{ji} + \eta \delta_j x_{ji},$$

where w_{ji} is the weight from unit i to unit j, x_{ij} is the input from unit i to unit j, η is the learning rate and δ_j is error obtained at unit j. This process of adjusting the weights using training instances is iterated for a fixed number of times or is continued until the error is small or cannot be reduced.

To improve the performance of the backpropagation algorithm, the weight-update made at the n^{th} iteration of the backpropagation is made partially dependent to the amount of weight changed in the n-1^{st} iteration. The amount by which the n-1^{st} iteration contributes is determined by a constant term called momentum (α). The new rule used for weight-update at the n^{th} iteration is

$$\Delta w_{ji}(n) = \eta \delta_j x_{ji} + \alpha \, \Delta w_{ji}(n-1).$$

This momentum term is added to achieve faster convergence to a minimum in some cases.

2.2.2.5 RBF Network

An RBF or Radial Basis Function Network [Buhmann & Albovitz, 2003; Orr, 1996] is another type of a feed-forward neural network. It has three layers: the input, hidden and output layer. It differs from an MLP in the way the hidden layer units perform calculations. An RBF Network can build both regression and classification models. We will describe the regression model.

In an RBF Network, inputs from the input layer are mapped to each of the hidden units. The hidden units use radial functions for activation such as Gaussian, multiquadric, inverse-multiquadric and Cauchy [Orr, 1996]. The RBF Network of WEKA [Witten & Wang, 2005] uses the bell-shaped Gaussian function. The activation *h(x)* of the Gaussian

function for a given input x decreases monotonically as the distance between the center c of the Gaussian and x increases. A Gaussian function is useful in finding the activation at a hidden unit, as the activation of inputs depends on their closeness to center of the hidden unit and thus can be used as a effective method to distinguish between inputs. The Gaussian function is of the form

$$h(x) = \exp\left(\frac{-(x-c)^2}{r^2}\right).$$

The output layer takes in linear combination of outputs from hidden units and is similar to a regression model. An RBF Network for our weather models will be of the form shown in Figure 2.9.

AN RBF Network takes the inputs and the hidden units as points in space. The activation of a hidden unit depends on the distance between the point in space representing the input values and the point for that hidden unit. The distance is converted

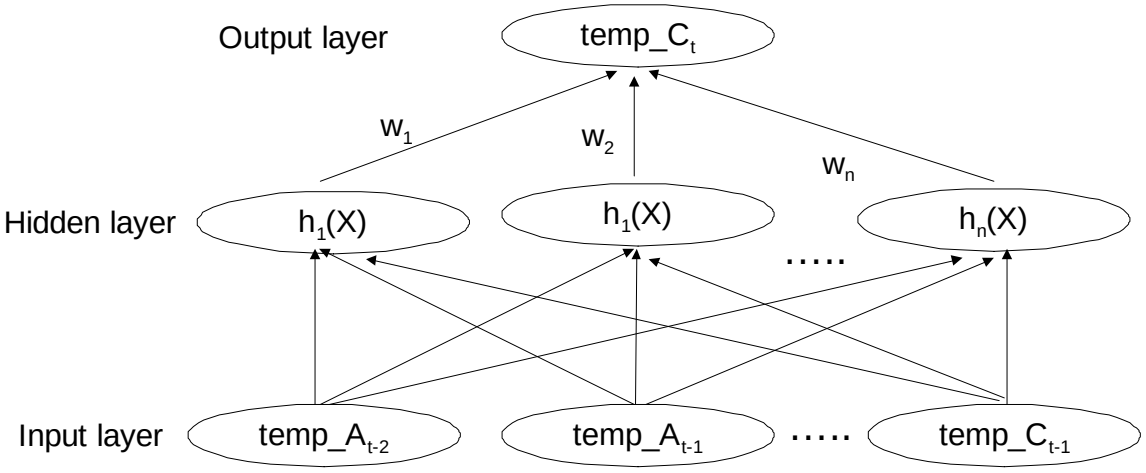

Figure 2.9: An RBF Network with *n* hidden units to predict temperature at a site. X is the input vector given to the network and output. *temp_C*$_t$ is the sum of the products of activations from the hidden units and the weights associated with them. Each hidden unit has its own learned center.

into a similarity measure by the Gaussian function. The point in space for the hidden unit is obtained from the center of the Gaussian for that hidden unit. The width of the Gaussian is a learned parameter as well.

An RBF Network is trained to learn the centers and widths of the Gaussian function for hidden units, and then to adjust weights in the regression model that is used at the output unit. To learn the centers of the Gaussian functions the k-means clustering algorithm can be used that clusters the training instances to obtain k Gaussian functions for each attribute in the instance. After the parameters for the Gaussian function at the hidden units have been found, the weights from these units to the output unit are adjusted using Linear Regression. The process can be repeated to learn in an EM manner.

2.2.2.6 The Conjunctive Rule Algorithm

The Conjunctive Rule algorithm in WEKA learns a single rule that can predict an output value. It can predict both discrete and numeric classes. An example of a conjunctive rule that can be developed from our weather example is

IF $temp\_A_{t-2} > 30$ *AND* $temp\_C_{t-1} < 90$ *AND* $temp\_B_{t-1} > 40$ *THEN* $temp\_C_t = 30$.

A conjunctive rule consists of a set of antecedents (e.g., $temp\_A_{t-2} > 30$, $temp\_C_{t-1} < 90$) *AND*ed together to give the consequent (e.g., $temp\_C_t$). Antecedents consist of relations between relevant attributes and the consequent indicates an output value.

The learning process in the Conjunctive Rule algorithm attempts to come up with a rule for all relevant attributes based on the training data. The algorithm learns by calculating the variance reduction for all possible antecedents and then selects the one that reduces the variance the most. In cases when the learned rule becomes too complex then it is pruned using reduced error pruning similar to that in the J48 system.

In cases when a test instance is seen that is not covered by the rule, then the attribute

whose value is not included in the rule is assigned the default value for that attribute.

2.3 Predicting Time Sequence Data - Hidden Markov Models

A discrete process taking place in the real world generates what can be viewed as a symbol at each step. For example, a coin toss can lead to a series of heads and tails. Another example is the temperature at a certain location that is a result of a number of weather conditions and the location. These conditions and the location add up to form a real-world time series. As time goes by and the process keeps running we get a sequence of symbols generated by the process. Based on the outcome of the process these symbols can be discrete (e.g., precipitation type) or continuous (e.g., temperature). These sequences of observed symbols can be used to generate a statistical model that describes the workings of the process and the generation of symbols over time. This model can be used to identity or classify other sequences of symbols. One such model that can be used is the Hidden Markov Model (HMM).

When information gathered from a system is generated in a sequential manner, state transition networks can be used to represent knowledge about this system. This network consists of a set of states and transitions between states. Transitions between states are triggered by events in the domain. HMM state transition networks have an initial state and an accepting or end state. The network recognizes a sequence of events if these events start at the initial state and end in the accepting state when all the events have occurred.

A Markov model is a probabilistic model over a finite set of states, where the probability of being in a state at a time *t* depends only on the previous state visited at time *t-1*. An HMM (see Figure 2.10 for an example) is a model where the system being modeled is assumed to be a Markovian process with unknown parameters.

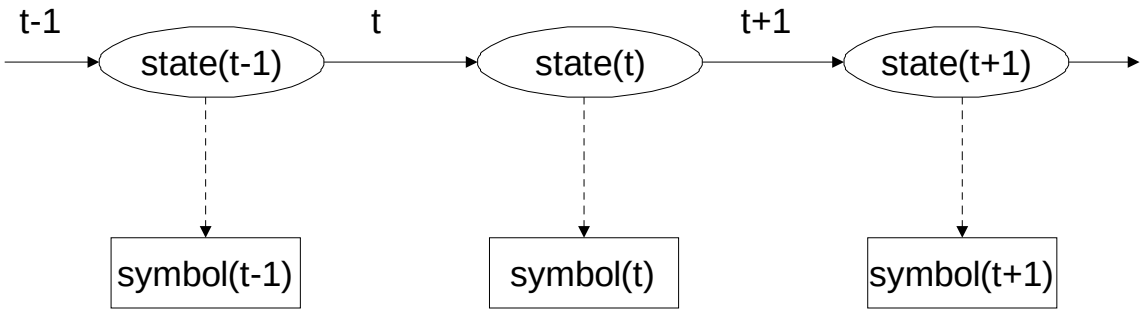

Figure 2.10: A Hidden Markov Model. A new state is visited when a transition occurs after a certain amount of time. Each state emits a symbol when reached.

In an HMM the states are referred to as hidden because the system we wish to model may have underlying causes that cannot be observed. For example, factors that determine the position and velocity of an object when the only information available is the position of the object are not observable. The hidden or non-observable process taking place in the system can be determined from the process that generates a sequence of observable symbols.

HMMs are used in many fields such as natural language processing, bioinformatics, genomics, optical character recognition and speech recognition. In speech recognition [Rabiner, 1989] the speech input is broken down into smaller segments which can be classified into a set of predefined symbols. The long sequences of symbols thus obtained are used to build a model using HMM learning methods. The generated model is used for processing sequences in order to identify or recognize the source of speech input.

An HMM can be used in our example of predicting weather conditions such as temperature. In order to predict temperature at a location, the temperature has to be converted into classes and each class represented by a unique symbol. For example, if we take hourly readings then we get a sequence of 24 symbols for a day. The sequences obtained for a set of days can be used to build an HMM model that predicts the temperature class value for each hour at that location.

The components that constitute an HMM are a finite set of states, a set of transitions between these states with a probability value associated with them and a set of output symbols emitted by the states. After each time interval a new state is entered depending on the transition probability from the previous state. This follows a simple Markov model in which the probability of entering a state depends only on the previous state. The state sequence obtained over time is called the path. The transition probability a_{km} of moving from a state k to a state m is given by the probability of being in state m at time t when at time $t\text{-}1$ we were in state k

$$a_{km} = P(path_t = m \mid path_{t-1} = k).$$

After a transition is made to a new state a symbol is emitted from the new state. Thus as the state path is followed we get a sequence of symbols ($x_1, x_2, x_3, \ldots$). The symbol which is observed at a state is based on a probability distribution that depends on the state. The probability that a symbol b is emitted on transition to state k, called the emission probability (e), is determined by

$$e_b(k) = P(x_t = b \mid path_t = k).$$

An initial state distribution gives the probability of being in a particular state at the start of the path. The choice of all these parameters, that is, the transition probabilities, emission symbols and initial state probabilities, are important in building an efficient HMM.

In our weather example, let the temperature be broken down into three classes, namely *p, q,* and *r*. Also assume that the HMM built has four states, namely *1, 2, 3* and *4*. In this case our state transition probability matrix will be of size 4 by 4 and the emission probability matrix will be of size 3 by 4. The model built by an HMM learner will have these four states and at each state we can observe any of the three symbols *p, q* and *r* based on their respective emission probabilities at that respective state. The transition from one state to another is associated with the transition probability between the respective states.

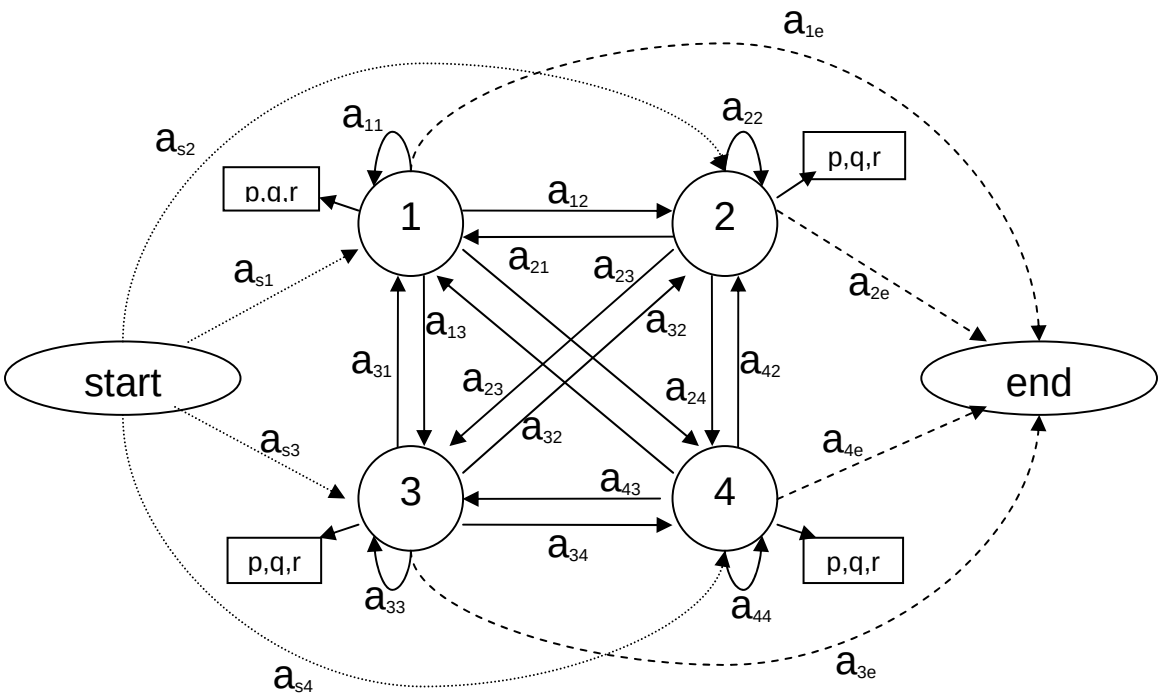

Figure 2.11: A HMM with states 1, 2, 3 and 4 that emit a symbol when reached. Transitions start at the start state and end at the end state. The arrows represent all possible state transitions.

The transition probability of moving from state 1 to state 2 is represented by a_{12}. As the sequence has a length of 24, we will be seeing 24 transitions including a transition from the start state and always ending at the end state after the 24th symbol is seen. Figure 2.11 shows a HMM with the four states 1, 2, 3 and 4 together with the start state, end state and all possible transitions between these states. The HMM in the figure allows all possible state transitions. The transition probability associated with each transition is given along with the transition. In the figure, symbols *p*, *q* and *r* are emitted at each state, *1, 2, 3* and *4*, are given in rectangular blocks. Emission of each symbol at a state is determined by its emission probability at that state. Transitions always start at the start state and end at the end state.

According to Rabiner & Juang [1986], for any HMM there are three basic questions that

have to be answered for the model to be used effectively,

1. What is the probability that an observed sequence is produced by the given model?
A model can generate many sequences based on the transitions taking place and symbols emitted each time a state is visited. The probability of the observed sequence being produced by the model gives an estimate of how good the model is. The Forward algorithm or the Backward algorithm [Baum & Petrie, 1966] can be used to find the probability of observing a sequence in a given model.

2. How do we find an optimal state sequence (path) for a given observation sequence?
As there exists many paths that yield a given sequence of symbols, to select the optimal state path we need to find the path with the highest probability of occurrence. The Viterbi algorithm [Viterbi, 1967; Forney, 1973] is a procedure that finds the single best path in the model for a given sequence of symbols.

3. How do we adjust the model parameters, that is, the transition probabilities, emission probabilities and the initial state probability to generate a model that best represents the training set?
Model parameters need to be adjusted so as to maximize the probability of the observed sequence. The Baum-Welch algorithm [Rabiner & Juang, 1986] is a method that uses an iterative approach to solve this problem. It starts with preassigned probabilities and tries to adjust them based on the observed sequences in the training set.

The Forward-Backward, the Baum-Welch and the Viterbi algorithms are described in detail in the following subsections.

2.3.1 The Forward Algorithm and The Backward Algorithm

A sequence of observations can be generated by taking different paths in the model. To

Table 2.2: The Forward Algorithm

Forward Algorithm

Initialization (i = 0) : $f_0(0) = 1, f_k(0) = 0$ *for* $k > 0$

Recursion (i = 1 .... L) : $f_m(i) = e_m(x_i) \sum_k f_k(i-1) a_{km}$

Termination : $P(x) = \sum_k f_k(L) a_{k0}$

$f_m(i)$ - prob. of seeing the symbol at position i in state m

$e_m(x_i)$ - probability of emitting the symbol x_i by state m

a_{km} - probability of transition from state k to state m

$P(x)$ - probability of observing the entire sequence x

L - length of the sequence

find the probability of this sequence we need to add up the probabilities that are obtained for each possible path the sequence can take. The Forward algorithm and the Backward algorithm uses dynamic programming to calculate this probability instead of having to enumerate the probabilities observed across all the possible paths.

As HMMs follow the Markovian process, the probability of being in a particular state depends only on the state that was observed before it. The Forward algorithm uses this approach to find the probability of the symbol sequence. After having observed $i - 1$ symbols in a sequence x of length L, let us say the symbol at position i in the sequence is seen in state m. Then the probability of seeing this symbol at position i in state m can be found by multiplying the emission probability of this symbol at state m with the sum of products of the probability of being at any state (with k states in the model) when the last symbol ($i - 1^{st}$) was seen with the transition probability of moving from that state to the present state m. It is called the forward probability and is given by

$$f_m(i) = e_m(x_i) \sum_k f_k(i-1) a_{km} \ .$$

In case a transition is not possible then the transition probability is taken as 0. Before

starting the sequence we initialize the probability of being in the start state with no symbols observed to be 1, that is, $f_0(0) = 1$. We calculate $f_m(i)$ for each position i in the sequence as we move through the sequence. The probability of the observing the entire sequence, $P(x)$, is the probability value obtained after we have seen all the symbols in the sequence after having reached the end state. The pseudocode for the Forward algorithm [Durbin et al., 1989] is given in Table 2.2.

The Backward algorithm can also be used to calculate the probability of the observed sequence. As the name says, it works analogously to the Forward algorithm. Instead of calculating the probability values from the start state, as is done by the Forward algorithm, the Backward algorithms starts at the end state and moves towards the start state. In each backward step, it calculates the probability of being at that state when considering that the rest of the sequence from that state to the end state has already been observed. We define the backward probability $b_k(i)$ as

$$b_k(i) = p(x_{i+1}.....x_L \mid path_i = k),$$

which is the probability of observing the rest of the sequence when we are in state *k* after observing *i* symbols.

Table 2.3: The Backward Algorithm

Backward Algorithm

Initialization (i = L) : $b_k(L) = a_{k0}$ *for all k*

Recursion (i = L-1 ..... 1) : $b_k(i) = \sum_m a_{km} e_m(x_{i+1}) b_m(i+1)$

Termination : $P(x) = \sum_m a_{0m} e_m(x_1) b_m(1)$

$b_k(i)$ - probability of observing rest of the sequence when in state k and having already seen *i* symbols

$e_m(x_i)$ - is the probability of emitting the symbol x_i by state *m*

a_{km} - is the probability of transition from state *k* to state *m*

$P(x)$ - is the probability of observing the entire sequence

At the start of the Backward algorithm we initialize the probability of being at a state *k* after observing an entire sequence, $b_k(L)$, to the transition probability from this state *k* to the end state.

The backward probabilities at each position in the symbol sequence are calculated moving from the end of the sequence to the start. The final probability of the sequence obtained *P(x)* by either Forward or Backward algorithm is the same. The pseudocode of the Backward algorithm [Durbin et al., 1989] is shown in Table 2.3.

2.3.2 The Baum-Welch Algorithm

The Baum-Welch algorithm, also called as the Forward-Backward algorithm, is used to adjust the HMM parameters when the path taken by each training sequence is not known. The HMM parameters can be initialized to predetermined values or to a constant before applying the Baum-Welch algorithm. As the path taken is not known, the Baum-Welch algorithm uses the counts of number of times each parameter is used when the observed set of symbols in the training sequence is given to the present HMM. The algorithm constitutes of two steps, the Expectation step (E Step) and the Maximization step (M Step). In the Expectation step we first find the forward and backward probability values at each position in the sequence. When forward and backward probabilities are combined together to obtain the probability of the entire sequence with symbol *k* being observed at state *i* and is given by

$$P(path_i = k \mid x) = \frac{f_k(i) b_k(i)}{f_N(L)}.$$

Using all of the training set of sequences, we can calculate the expected number of times a symbol *c* is emitted at state *k* using

$$n_{k,c} = \sum_{x^j} \left[\frac{1}{f_N^j(L)} \sum_{\{i \mid x_i^j = c\}} f_k^j(i) b_k^j(i) \right]$$

Table 2.4: The Baum-Welch Algorithm

Baum-Welch Algorithm

Initialize the parameters of HMM and pseudocounts $n'_{k,c}$ and $n'_{k->l}$

Iterate until convergence or for a fixed number of times

- E Step: for each training sequence $j = 1 \ldots n$
  - calculate the forward probability $f_k(i)$ for the sequence j
  - calculate the backward probability $b_k(i)$ for the sequence j
  - add the contribution of sequence j to $n_{k,c}$ and $n_{k->l}$
- M Step: update HMM parameters using the expected counts $n_{k,c}$ and $n_{k->l}$ and the pseudocounts $n'_{k,c}$ and $n'_{k->l}$.

The number of times a transition from state k to m occurs is given by

$$n_{k\to m} = \sum_{x^j} \frac{\sum_i f_k{}^j(i) a_{km} e_m(x^j{}_{i+1}) b_k{}^j(i+1)}{f_N{}^j(L)},$$

where the superscript j refers to an instance in the training set.

The Maximization step uses the counts of the number of times a symbol is seen at a state and the number of times a transition occurs between two states which were obtained from the Expectation step to update the transition and emission probabilities in order to maximize the performance. The updated emission probability is

$$e_k(c) = \frac{n_{k,c} + n'_{k,c}}{\sum_{c'} (n_{k,c} + n'_{k,c})}.$$

The transition probability is updated using

$$a_{km} = \frac{n_{k\to m} + n'_{k\to m}}{\sum_m (n_{k\to m} + n'_{k\to m})}.$$

Pseudocounts $n'_{k,c}$ and $n'_{k\to m}$ for emission and transition probabilities respectively are taken into account because it prevents the numerator or denominator from taking a value

of zero which happens when a particular state is not used in the given set of observed sequences or a transition does not take place, respectively. Pseudocounts are usually set to the actual counts plus small decimal values. Table 2.4 shows the pseudocode of the Baum-Welch algorithm.

2.3.3 The Viterbi Algorithm

The Viterbi algorithm is used to find the most probable path taken across the states in the HMM. It uses dynamic programming and a recursive approach to find the path. The algorithm checks all possible paths leading to a state and gives the most probable one. The calculations are done using induction, in an approach similar to the forward algorithm, but instead of using a summation, the Viterbi algorithm uses maximization.

Table 2.5: The Viterbi Algorithm

Viterbi Algorithm

Initialization (i = 0) : $v_0(0) = 1, v_k(0) = 0 \; for \; k > 0$

Recursion (i = 1 ..... L) : $v_m(i) = e_m(x_i) \max_k (v_k(i-1)a_{kl})$

$ptr_i(m) = \arg\max_k (v_k(i-1)a_{kl})$

Termination : $P(x,path^*) = max_k(v_k(L)a_{k0})$

$path^*_L = argmax_k(v_k(L)a_{k0})$

Traceback (i = L.....1) : $path^*_{i-1} = ptr_i(path^*_i)$

$v_m(i)$ - probability of the most probable path obtained after observing the first i characters of the sequence and ending at state m

$ptr_m(i)$ - pointer that stores the state that leads to state m after observing i symbols

$path^*_i$ - state visited at position i in the sequence

The probability of the most probable path obtained after observing the first *i* characters of the sequence and ending at state *m*, represented by $v_m(i)$ is

$$v_m(i+1) = e_m(i+1)\max_k(v_k(i)a_{km}) .$$

The algorithm states from the start state and thus $v_0(0)$ is initialized to 1. The algorithm keeps track of the best state used during a transition using pointers. The pointer $ptr_i(m)$ stores the state that leads to state *m* after observing *i* symbols in the given sequence, it is found using

$$ptr_i(m) = \arg\max_k(v_k(i)a_{km}) .$$

The most probable path is found by moving through the pointers backwards starting from the end state to the start state. Sometimes we may obtain more than one path as the most probable; in such cases one path is randomly selected. The pseudocode for the Viterbi Algorithm [Durbin et al., 1989] is shown in Table 2.5.

Chapter 3

ML Methods for Weather Data Modeling

To detect abnormal behavior of an RWIS sensor we build a model that provides us with a predicted value for a weather condition, which we can compare to the actual value reported by the sensor and calculate the difference between the two to measure likely sensor malfunctions. We can build such models using machine learning (ML) methods that can predict the weather conditions at the RWIS site. To build a weather model, ML methods require historical weather data obtained from the site and its nearby sites to learn the weather patterns. By including nearby sites, we provide additional information for the methods that can be used to indicate current climatic conditions at the site used for predictions.

To predict the temperature at a given time for site 67, in Figure 3.1, we can use the current and a couple of previous hour's weather data, such as temperature and visibility,

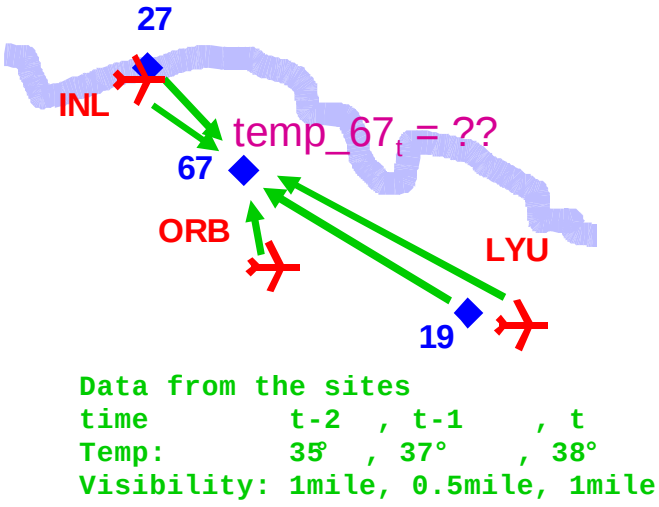

Figure 3.1: Predicting temperature value at a site using weather data from nearby sensors.

obtained from a set of sites that are located close to the site 67. Nearby sites to site 67 are indicated by arrows that point from them to site 67 (the arrows show that the information from these sites will be used in making predictions about the site they point to). ML methods are used to build the weather model for site 67 using historical data for the different variables used for predictions. The model built for site 67 will use the temperature and visibility information from the nearby sites and predict the temperature value that will be seen at site 67 at a time

In this chapter we discuss the use of Machine Learning (ML) methods to detect RWIS sensor malfunctions. In the first section, we describe the process of selecting RWIS and AWOS sites that can be used for modeling followed by the description of variables in the weather data collected from the selected sites. In the next section we describe the feature representation used by ML and HMM methods. In the final section we describe the general approach followed by these methods to predict weather variables.

3.1 Choosing RWIS - AWOS Sites

To predict weather variables at a site we gather relevant weather information from the site and this information is used to build a predictive model by using various machine learning methods. We then use the predictive model to classify new or previously unseen data. The prediction of values reported by the sensors at a site can be made more accurate if along with the information for the present site we consider the information obtained from the sensors located at sites surrounding the present site. Todey et al., [2002] report a significant improvement in analysis of weather data when using a combined dataset obtained from the sites in the RWIS and the AWOS network. To predict values reported by an RWIS sensor at a site we use meteorological data from surrounding RWIS sites and also data gathered from AWOS sites.

Out of the 76 RWIS sites present in Minnesota, we selected 13 sites to be used to detect

RWIS sensor malfunctions at these locations. The selection of the sites was based on the climatic conditions and landscape at the locations these sites were situated. Sites in regions that have micro-climates, such as Duluth and many places in southern Minnesota, were not selected because of the climatic conditions at these sites do not reflect changes happening in their surroundings and have their own unique ecosystems. Similarly sensors located in urban areas, like Minneapolis, were not selected because of drastic climatic changes that occur in such areas due to human involvement. We further grouped the selected 13 RWIS sites into three sets in order to prevent macro-climate comparisons. As Minnesota has a diverse landscape, the climatological conditions in the north do not always reflect on the conditions in the south regions. The aim of grouping the RWIS sites into sets was to prevent comparisons between two sites present in totally different climatological regions. Each set can be compared to a simple climatological regime and the climatic changes at a site in the set are reflected on other sites, not necessarily at that instant but after a certain duration of time. Grouping helps in predicting the weather condition at a site when that condition is known in other sites in the set.

Along with the weather information from the RWIS sites, we use meteorological data gathered from the AWOS sites to help with the prediction of values at an RWIS site. The location of the RWIS sites with respect to the site's topography is variable, as these sensors sit near a roadway and are sometimes located on bridges. AWOS sites are located at airports on a flat surrounding topography, which leads to better comparability between weather data obtained from these sites. Thus including surrounding AWOS information can be beneficial in predicting values at an RWIS site.

We associated each RWIS site, in the 13 selected for prediction, with all the AWOS sites that were at a distance of at least 30 miles from it. 30 miles is chosen as a measure for association so as to pair at least one AWOS site with each RWIS site. For distances of more then 30 miles, it was seen that some RWIS sites were paired with the same AWOS sites. The distance is calculated using the latitude and longitude coordinates of the respective RWIS and AWOS sites (Tables A1 and A2). All RWIS sites are paired with

one AWOS site, with the exception of site 20, which is associated with two AWOS sites namely KAIT and KBRD, with KAIT being the closer one. Due to the comparatively smaller distance between an RWIS site and its associated AWOS site, we find there is often a correlation between the values observed at these sites, which we can use as the basis of our models. Figure 3.2 shows the the locations of the RWIS and AWOS sites that we grouped together and Table 3.1 lists these groups.

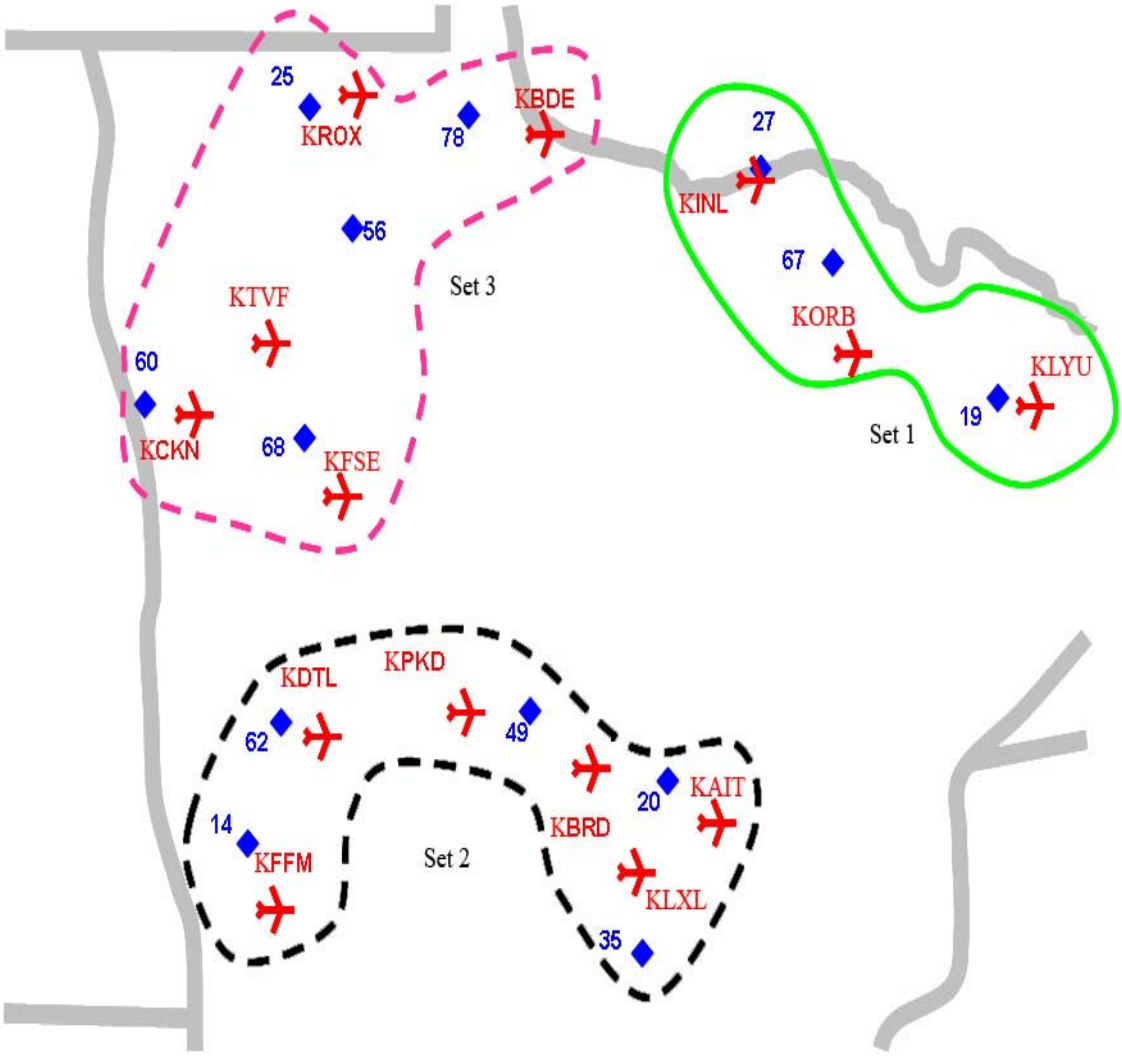

Figure 3.2: Grouping of RWIS and AWOS sites into three sets. This map also shows the locations of the selected RWIS and AWOS sites across Minnesota.

Table 3.1: Grouping of the selected 13 RWIS sites into three sets, along with their respective AWOS sites.

Set	*RWIS Sites*	*AWOS Sites*
Set 1	19, 27, 67	KLYU, KINL, KORB
Set 2	14, 20, 35, 49, 62	KFFM, KAIT, KBRD, KLXL, KPKD, KDTL
Set 3	25, 56, 60, 68, 78	KROX, KTVF, KCKN, KFSE, KBDE

3.2 Features Used

Of all the available features reported by RWIS sensors, we decided to focus on predicting air temperature, precipitation type and visibility. These three features were selected because they represent critical aspects of weather data for Mn/DOT.

All of these variables (temperature, precipitation type and visibility) are also reported by the AWOS sites. However, the data format used for reporting these variables by RWIS and AWOS sensors differs (refer to Sections 2.1.1.1 and 2.1.2.1 for details). To make the data from these two sources usable in ML algorithms and HMMs, and for comparisons, the data needs to be transformed into a common format and in some cases pre-processing of the variables may be required based on the requirements of the algorithms used.

3.2.1 Transformation of the Features

To use data from RWIS and AWOS together, for predictions and comparisons, we converted the data reported by them to follow a common format. It is also the case that RWIS reported data every 10 minutes whereas AWOS provides hourly reports. Thus, the RWIS data needs to be averaged if used along with AWOS data. The changes that were made to the features reported by RWIS sites to arrive at a common format are

- RWIS uses Greenwich Mean Time (GMT) and AWOS uses Central Time (CT) when reporting data. The reporting time in RWIS is changed from GMT to CT.
- Variables like air temperature, surface temperature and dew point that are reported in Celsius by RWIS are converted to Fahrenheit, which is the format used by AWOS. To convert six readings per hour to a single hourly reading in RWIS, a simple average is taken.
- Distance, which is measured in kilometers by RWIS for visibility and wind speed is converted into miles, the format used by AWOS. A single hourly reading is obtained through a simple average in RWIS.
- In order to obtain hourly averages for precipitation type and intensity reported by RWIS, we use the most frequently reported code for that hour. While for precipitation rate a simple average is used. RWIS sites report precipitation type and intensity as separate variables, whereas AWOS combines them into a single weather code [NCDC, 2005]. Mapping precipitation type and intensity reported by RWIS to the AWOS weather codes is not feasible and needs some compromises to be made. We thus keep these variables in their original format.

Of the three features selected for use in predictions, precipitation type is the one whose direct comparison between RWIS and AWOS values cannot be done because each uses a different format for reporting it. For broader comparison, we combine all the codes that report different forms of precipitation, in both RWIS and AWOS, into a single code which indicates the presence of some form of precipitation.

Apart from using the features obtained from the RWIS and AWOS sites, we also made use of historical information to represent our training data. This increases the amount of weather information we have for a given location or region. We collected hourly temperature values for the AWOS sites mentioned in the three sets (refer to Table 3.1), for a duration of seven years ranging from 1997 to 2004, from the Weather Underground

website[2]. For many locations, the temperature was reported more than once an hour, in such cases the average of the temperature across the hour was taken. As we already have readings for temperature from two different sources, RWIS and AWOS, we use the information gathered from the website to adjust our dataset by deriving values such as the projected hourly temperature. To calculate the projected hourly temperatures for an AWOS site we use past temperature information obtained from the website for this location. The projected hourly temperature for an hour of a day is defined as the sum of the average temperature reported for that day in the year and the monthly average difference in temperature of that hour in the day for the respective month.

The steps followed to calculate the projected hourly temperature for an AWOS site are

1. Obtain the hourly temperatures from the data collected from wunderground.com for the respective AWOS site.
2. The average temperature for a day was calculated as the mean of the hourly readings across a day.
3. The hourly differences for each hour was calculated as the difference between the average daily temperature and actual temperature seen at that hour.
4. The average difference in temperature for a particular month (monthly average difference) per hour was calculated as the average of all hourly differences in a month for that hour obtained from all the years in the data collected.
5. The projected hourly temperature for a day is obtained from the sum of the average temperature for that day and the monthly average difference of that hour in the day.

For example, let the temperature value seen at the AWOS site KORB for the first hour for January 1st for year 1997 is 32ºF. The temperature values for all 24 hrs seen on January 1st are averaged and let this value be 30ºF. The hourly difference for this day for the first hour will be 2ºF. Let the averages of all the first hour values for the month of January seen in the data for KORB from years 1997 to 2004 be 5ºF. Then the projected

2. http://www.wunderground.com/

temperature value for the first hour in January 1st, 1997 will be 35ºF, which is the sum of the average temperature seen on January 1st 1997 and the monthly average difference for the first hour in the month of January.

The projected hourly temperature was used as a feature in the datasets used for predicting weather variables at an RWIS site and is also used in the process of discretization of temperature, which will be discussed in the following section.

3.2.2 Discretization of the Features

Continuous features need to discretized when used in HMMs and classification algorithms. HMMs require all of the features to be discrete. Classification algorithms need the output attributes to be discrete and also the inputs attributes in case the algorithm cannot deal with continuous inputs. Regression algorithms can take inputs with discrete attributes. Discretization of features involves finding a set of values that split the continuous sequence into intervals and each interval is given a single discrete value. Discretization can be done using unsupervised or supervised methods. In unsupervised discretization, the attribute is divided into a fixed number of equal intervals, without any prior knowledge of the target (the output attributes) class values of instances in the given dataset. In supervised discretization, the splitting point is determined at a location which increases the information gain with respect to the given training dataset [Quinlan, 1986]. Dougherty et al., [1995] give a brief description of the process and a comparison of unsupervised and supervised discretization. WEKA provides a wide range of options to discretize any continuous variable, using supervised and unsupervised mechanisms.

In this thesis we propose a new method for discretization of temperature values obtained from RWIS sensors using temperature information obtained from other sources. Using the projected hourly temperature (refer to Section 3.2.1) for an AWOS site along with the current reported temperature for the closest RWIS site, we determine the class value for

the current RWIS temperature value.

To determine the class value, the actual reported temperature at a RWIS site is subtracted from the projected hourly temperature for the AWOS site closest to it for that specific hour. This difference is then divided by the standard deviation of the projected hourly temperature for that AWOS site. The result indicates how much the actual value deviates above or below the projected value, that is, the number of standard deviation from the projected value (or the mean).

$$num\_stdev = (actual\_temp - proj\_temp) / std\_dev$$

The classes are divided according to the number of standard deviations from the mean, For example

Class Value		**Class Value**	
1	num_stdev < -2	6	0.25 < num_stddev ≤ 0.5
2	-2 ≤ num_stdev ≤ -1	7	0.5 < num_stddev ≤ 1
3	-1 < num_stdev ≤ -0.5	8	1 < num_stddev ≤ 2
4	-0.5 < num_stdev ≤ -0.25	9	num_stdev > 2
5	-0.25 < num_stdev ≤ 0.25		

Thus the number of standard deviations from the mean obtained for a given temperature value is mapped to one of the ranges and the temperature is assigned the respective class value. The ranges and the number of splits can be determined by the user or based on requirements of the algorithm.

For example, to convert the actual temperature 32ºF at an RWIS site we calculate the projected temperature for that hour at the associated AWOS site KORB, which might be 30ºF, and the standard deviation of projected temperatures at KORB for the year, which might be 5.06. Then our new representation for 32ºF is calculated as 0.396, which means

temperature 32ºF is 0.396 standard deviations above from the mean at site KORB. 0.396 maps to class 6 in the split example given above. We thus arrive at the class value of 6 for the temperature 32ºF. Such a representation has an advantage as the effect of season and time of day is at least partially removed from the data.

3.3 Feature Vectors

To build a predictive model, using ML algorithms, that can predict the values reported by an RWIS sensor, we require a training set that contains data related to the weather conditions. A test set is used to evaluate the performance of the model built. ML algorithms require the feature vector for the dataset to consist of a set of input attributes and an output attribute, with the output attribute being the predicted variable. ML algorithms use the information from the feature vector in the training set to build the model. The input attributes for a feature vector in the test set are applied on the model built from the training set to predict the output attributes value and this predicted value is compared with the actual value to estimate the model performance. The feature vector for ML algorithms takes the form

$$input_1, ..., input_n, output$$

where $input_i$ is an input attribute and *output* is the output attribute.

For example, to predict temperature at time *t* for the RWIS site 67 denoted as $temp67_t$, from the Figure 3.1, we will be using the temperature values at time *t* and *t -1* for all the associated RWIS sites which are 27 and 67. The feature vector for this example will be

$$temp19_{t-1}, temp27_{t-1}, temp67_{t-1}, temp19_t, temp27_t, temp67_t$$

where $temp67_t$ is the output attribute or dependent variable (and is therefore only available during the process of building the model as this is what we are predicting) and the rest form the input attributes or independent variables.

Predicting a value reported by a RWIS sensor corresponds to predicting that weather

condition reported by the sensor. As variations in a weather condition, like air temperature, are not completely independent but depend on other conditions such as wind, precipitation, and air pressure, all these variables can be used to predict that condition at a location. The previous hour's readings (or further back) from a sensor can be used to predict present conditions, as changes in weather conditions follow a pattern and are not totally random. The RWIS-AWOS sets include sites from locations that do not belong to any micro-climatic regions, and thus the climatic conditions at a location have some correlation to the condition at another location in the set. Using this information we include data obtained from the nearby sites, both AWOS and RWIS, to predicted variables at an RWIS site. This can be done by including all the sites in the RWIS-AWOS set that a particular RWIS site belongs to.

In brief, to predict a value for an RWIS sensor we use other weather variables such as the previous hour's readings (or further back) for these variables and these respective readings from nearby RWIS and AWOS sites (see Figure 3.1).

3.4 Feature Symbols for HMMs

The instance in a dataset used by HMM for predicting a class value of a variable consists of a string of symbols that form the feature symbol. The symbols that constitute the feature symbol are unique and are generated from the training set.

For predicting a weather variable value at an RWIS site, taking hourly readings for a day of the variable's value into consideration we get a string of length 24. This string forms the feature symbol which is used in the dataset. When information from a single site is used for predictions, the class values taken by the variables form the symbol set.
When the variable information from two of more sites are used together, a string of class values is obtained by appending the variable's value from each site together. For example, to predict temperature class of site 19, we include temperature data from sites 27 and 67,

which belong to the first set of RWIS sites. The combination of class values from these three sites seen at a particular hour will form a class string. For example if at sites 19, 27 and 67 the temperature class values seen are 3, 5 and 4 respectively then the class string formed by appending the class values in the order their sites were listed in this example will be 354.

The sequence of hourly class strings for a day form an instance in the dataset. All unique class strings that are seen in the training set are arranged in ascending order, if possible, and each class string is assigned a symbol. For hourly readings taken during a day we arrive at a string which is 24 symbols long, which forms the feature symbols used by HMM.

3.5 Methods Used for Weather Data Modeling

To be able detect RWIS sensor malfunctions, we need to check for significant variations and/or deviations between the values reported by this sensor and the actual weather conditions present at the location of this site. To determine the actual weather condition at a location, we try to predict a value for that weather variable. Based on the difference between the predicted value and the value reported by the RWIS sensor we can detect sensor malfunctions. To predict a variable's value a function is derived that can explain the system of weather variations. We use ML methods to build such a function, in other words a predictive model, using the weather data collected from the past. This model is used to predict present values for a weather variable. Both classification and regression algorithms and HMMs are used for modeling the weather data.

Malfunctions in a sensor are easily detected if the predicted values are highly accurate. This makes the accuracy of the prediction made by ML methods a key factor in detecting sensor malfunctions. The performance of an algorithm on the data provided is evaluated using the cross-validation technique.

We will first describe cross-validation and then present the general approach taken by the ML methods for prediction of weather variables at the RWIS sites.

3.5.1 Cross-Validation

Cross-validation is a technique that is used to evaluate the performance of a model built by an algorithm. In cross-validation, a subset of the data provided is kept aside and the remaining data, the training set, is used by the algorithm to build a model. The part of dataset not used in training, the test set, is then used to evaluate the performance of the model by measuring the accuracy with which the model classifies the test set instances.

In *n*-fold cross-validation, the dataset is divided into *n* subsets of equal size. One subset is used as a test set and the remaining *n-1* subsets are used for training. The cross-validation is performed *n* times with each of the subsets being used as a test set exactly once. The performance of the model on each of the subset used as test set is averaged to calculate the overall performance of the algorithm. The advantage of using *n*-fold cross-validation is that each instance of the dataset gets to be in the test set once and it can be used to evaluate the performance of the model within a single dataset. Kohavi [1995] suggests using 10 or 20-fold cross-validations for better estimates.

Multiple *n*-fold cross-validations can be performed on a single dataset by randomizing the data in the dataset before splitting the it into *n* subsets. This method puts different data into the *n* subsets created each time. Data can be randomized using a random number generator and a different random order can be obtained each time by changing the seed value given to the generator.

3.5.2 General Classification Approach

The classification algorithms are used to classify the given instance into a set of discrete categories. Discrete variables like precipitation type and discretized temperature values are predicted using this approach. The classification algorithms described in Section 2.2.1, namely J48 decision trees, Naive Bayes and Bayesian Networks were used to predict these variables. As all these classification algorithms allow continuous input attributes, the feature vector used included the current and a couple of previous hour's temperature readings for various RWIS - AWOS sites along with the variable we are trying to predict.

Each dataset was built according to a feature vector format we derived based on the weather data available for the RWIS and the AWOS sites. The dataset is then split into a training set and a test set using the cross-validation method. The classification algorithms use the training set to build a model and the test set is used to evaluate the performance of the model. Multiple *n*-fold cross-validation are performed to obtain a better estimate of the model performance.

Classification algorithm predict the class value taken by the output attribute, in our case precipitation type and temperature class value, for a given instance in the test set. The prediction results are represented in form of a confusion matrix, with rows corresponding to actual values and columns corresponding to predicted values for the output attributes. Each block in the confusion matrix gives the number of times the actual class is predicted as the class given by the column. The numbers in the diagonal blocks give the number of time the predicted class value was equal to the actual class value. Thus, the sum of entries along the diagonals divided by the total number of instances present in the test set, gives percentage of the number of correctly classified instances. In the case of multiple *n*-fold cross-validations, the confusion matrices obtained for each test set seen are averaged to obtain a confusion matrix with the mean values.

For example, for the two confusion matrices Matrix 1 and Matrix 2, their Average Matrix can be obtained by averaging the values in their respective blocks. Foe example, the value at row 1 and column 1 in the Average Matrix is obtained by averaging the row 1 and column 1 values in the matrices A and B, that is, average of 10 and 15 is 12.5.

Matrix 1

Actual \ Predicted	Class A	Class B	Total
Class A	10	20	30
Class B	30	40	70
Total	40	60	100

Matrix 2

Actual \ Predicted	Class A	Class B	Total
Class A	15	15	30
Class B	30	40	70
Total	45	55	100

Average Matrix

Actual \ Predicted	Class A	Class B	Total
Class A	12.5	17.5	35
Class B	30	40	65
Total	42.5	57.5	100

The error in the results is obtained from predicting the class value for the discretized temperature is determined by using the absolute distance between actual and predicted class values. A distance between two adjacent classes is taken as 1. In case of the actual and predicted class values being the same the distance is 0, which represents a correctly classified instance. The greater the distance the poorer the prediction.

We can use the percentage of instances that were classified correctly when no precipitation was present and when precipitation was present to determine the accuracy of the model built. This method is particularly important for determining the accuracy for precipitation because for most of the time no precipitation is reported, and thus in cases when precipitation is reported the algorithm may try to classify it as no precipitation by assuming the data is noise.

For the classification model built for an RWIS site using historical data, we can provide the current weather information for the attributes present in the feature vector and the

model will predict a class value for that output at an RWIS site. For the discretized temperature the absolute distance between the predicted and actual values is used as a measure to detect sensor malfunctions, while for precipitation type we need to compare the performance ratios for detecting precipitation and no precipitation present for a misclassification to be able to identify a sensor malfunction.

3.5.2 General Regression Approach

Regression algorithms are used to determine the value taken by the output attribute in the given instance, based on an equation or mathematical operations. Continuous variables like temperature and visibility can be predicted using this approach. The regression algorithms described in section 2.2.2, namely Linear Regression, Least Median Square, M5P, MultiLayer Perceptron, RBF Nets and Conjunctive Rule are used to predict these variables. The feature vector for these regression attributes includes the current and a couple of previous hour's temperature readings for various RWIS - AWOS sites along with other information, with one sites temperature being the output attribute.

Each dataset is built according to the feature vector produced from the weather data available for the RWIS and AWOS sites. The dataset is split into a training set and a test set. The regression algorithms use the training set to build a model and the test set is used to evaluate the performance of the model. Multiple repetitions of *n*-fold cross-validation is used to obtain a better estimate of the model performance.

For a given set of input attributes the model will predict a value for the output. The performance of regression algorithms can be determined by the difference between the actual value and predicted value, which gives the amount of error in the prediction made. For example, the mean absolute error when actual temperature value is 32ºF and the predicted values is 35ºF is 3ºF. The mean of the absolute errors across all instances in the test set gives the performance of the algorithms on the test set. In the case of multiple

n-fold cross-validations, the error value is averaged across all the test sets seen.

To the regression model built for an RWIS site using historical data, we can provide the current weather information for the attributes present in the feature vector and the model will predict a value for the concerned output at the RWIS site. The closeness of the predicted value to the actual value depends on the efficiency of the model. For a model whose prediction accuracy is high a slight difference may indicate sensor malfunction. Even if the model is not efficient in predicting accurately, its consistency can be used to determine variations in values reported.

3.5.4 General HMM Approach

HMMs can be used to predict the class value of the weather variables seen at an RWIS site in a set. As HMMs require the variables to be discrete, precipitation type and discretized temperature can be predicted using HMMs. Each instance in the training set and the test set used in HMM consists of a string of symbols. For a given hour, the symbol string generated consists of a variable's class value seen at the RWIS site we are trying to predict. The class values seen at other RWIS sites that belong to a set can also be included. When a variable's information from two or more sites is used, the variable value from each site is appended to form a class string and all the unique class strings seen in training data are assigned a symbol. Thus for each day we get a symbol string of length 24 in the dataset.

To build the model from a given number of states and symbols emitted at a state, the Baum-Welch algorithm is applied on the training set to determine the initial state, transmission and emission probabilities. Each symbol present in the symbol set that was generated using the training set is emitted by a state. Instances from the test set are then passed to the Viterbi algorithm which gives the most probable state path.

To predict a variables class value, we need the symbol that would be observed across the most probable state path found by the Viterbi algorithm. We modified the Viterbi algorithm (described in Section 2.3.3), which we call the *modified Viterbi algorithm*, to obtain the most probable symbol that will be observed at each state visited across the most probable state path. Table 3.2 shows the modified Viterbi.

The symbol with the highest probability at a state is the one that has the highest emission probability at that state. As we are trying to predict the class value of a variable at a site, which forms one part in the class string that was converted into a symbol and the predicted value can be any of the class values taken by the variable, the predicted variable's class values in the class string is replaced with all possible values taken by the variable to get a set of possible symbols for the given symbol. Of the set of possible symbols we consider only those symbols that are seen in the training set. The symbol with the highest emission probability at the concerned state is predicted at that state. For example, if a variable took class values 1, 2 and 3 and the class string is of the from '1AB' with A and B being class values of other variables. Then all possible class strings for this variable would be 1AB, 2AB and 3AB.

We calculate the error in prediction as the absolute distance between the actual class value reported and the predicted class value. For calculating the error we find the distance with respect to the class value of the site being predicted. The distances between class values of other sites added to the class string are not taken in to account. For example, the class string for predicting temperature value at site 19 using sites 27 and 67 temperature class information has class values arranged as value of 19 followed by the values at 27 and 67. If the predicted class string for a given time is 345 and the actual class string seen for the time is 435, we get the error as a distance of 1, which is the difference between class values of the first position in the class strings.

Table 3.2 The Modified Viterbi Algorithm

Modified Viterbi Algorithm

Initialization (i = 0) : $v_0(0) = 1, v_k(0) = 0$ *for* $k > 0$

Recursion (i = 1 ..... L) :

$$symbolset = allpossiblesymbols(x_i)$$
$$v_m(i) = \max_k (v_k(i-1)a_{km}) * \max_s (e_m(symbolset_s))$$
$$ptr_i(m) = \arg\max_k (v_k(i-1)a_{kl})$$
$$bestsymbol_i(m) = \arg\max_s (e_m(symbolset_s))$$

Termination : $P(x,path^*) = max_k(v_k(L)a_{k0})$

$path^*_L = argmax_k(v_k(L)a_{k0})$

Traceback (i = L.....1) : $path^*_{i-1} = ptr_i(path^*_i)$

Symbol Observed (i = 1 ... L): $symbol(i) = bestsymbol_i\,(\,path^*_i\,)$

$v_m(i)$ - probability of the most probable path obtained after observing the first *i* characters of the sequence and ending at state *m*

$ptr_m(i)$ - pointer that stores the state that leads to state *m* after observing *i* symbols

$path^*_i$ - state visited at position i in the sequence

$bestsymbol_i(m)$ - the most probable symbol seen at state *m* at position in the sequence

$symbolset_s$ – the set of all possible symbols that can be emitted from a state when a certain symbol is actually seen

$allpossibesymbols(x_i)$ – function that generates all possible symbols for the given symbol x_i.

$symbol(i)$ – symbol observed ith position in the string.

The modified Viterbi algorithm works similarly to the Viterbi algorithm. The Viterbi algorithm calculates the probability of the most probable path obtained after observing the first *i* characters of the given sequence and ending at state *m* using

$$v_m(i) = e_m(x_i)\max_k (v_k(i-1)a_{km}),$$

where $e_m(x_i)$ is the emission probability of the symbol at state *m* and a_{km} is the transition probability of moving from state k to state *m*. In the modified Viterbi algorithm, in place of $e_m(x_i)$ we use the value of emission probability value that is maximum of all of the possible symbols that can be seen at state *m* when the symbol x_i which is present in the

actual symbol sequence. We calcualte $v_m(i)$ in the modified Viterbi algorithm as

$$v_m(i) = \max_k (v_k(i-1)a_{km}) * \max_s (e_m(symbolset_s)).$$

The actual symbol seen in the symbol string for that respective time is taken and all its possible symbols are found. Of all of the possible symbols or class strings, only those that are seen in the training instances are considered in the set of possible symbols. The symbol from the set of possible symbols that has the highest emission probability is selected as the symbol observed for the given state. To remember the symbol that was observed at a state the modified Viterbi algorithm uses the pointer $bestsymbol_i(m)$ which contains the position of the selected symbol in the set of possible symbols seen at state m for a position i in the sequence. It is given by

$$bestsymbol_i(m) = \arg\max_s (e_m(symbolset_s)).$$

The most probable path generated by the algorithm is visited from the start and by remembering the symbols, using $bestsymbol_i(m)$, that were observed at the state for a particular time we can find the symbols that were with the highest emission probability at each state in the most probable state path. This new symbol sequence gives the observed sequence for that given input sequence, in other words, it is the predicted sequence for the given actual sequence.

To predict the class values for a variable at a RWIS site for a given instance, we pass the instance through the modified Viterbi algorithm which then gives the symbols that have the highest emission probabilities along the most probable path. The symbol found is then converted back into class string and the class value for the concerned RWIS sites is regarded as the predicted value.

For example, to predict temperature class values at site 19 we use the class string has temperature class values from sites 27 and 67 appended to the value observed at site 19. Let a given class string sequence for a day be

343, 323,243,544,.....,324 (a total of 24, one for each hour in a day)

This sequence is passed to the modified Viterbi algorithm which gives the observed sequence seen along with most probable state path. Let this output sequence for the given sequence be

433,323,343,334,....,324

As we are interested in predicting the class values at site 19, taking only the first class value from a class string we get, for actual values

3,3,2,5,....,5

the observed (or predicted) values for the day are

4,3,3,3,...,3

The distance between these class values will then give the accuracy of the predictions made by the model.

In cases when the class string obtained from the test set is missing in the set of class strings seen in the training data, we need to replace it, as it does not have a corresponding symbol associated with it and thus is not recognized by the HMM. For variables like discretized temperature, which have a relation between two class values, the missing class string is replaced with a class string seen in the training data that is closest in distance to the missing class string. The distance measure used is the Manhattan distance. The difference between two classes is taken as their distance, for example class 3 and 5 have a distance of 2 between them. The distance between two class strings is the sum of the distances between each individual class value, for example, class strings 355 and 533 have a distance of 6 between them. When finding the closest we start with a distance of 1 and change the class value for site to be predicted first and then the other sites according to their distance from the site used for prediction. If no class string is found at a distance of 1 we try by incrementing the distance. In case of precipitation type, where each class value is unique, the class value is changed to indicate no precipitation present, as this is the case that is seen for most of the instances. Replacing is also done with some changed at a time and changing the vales at the current and then the nearby sites.

The performance of the HMM on the given test set, on precipitation type and discretized temperature, can be evaluated using the same methods that were used in the general classification approach.

As the HMM needs a full length symbol string at a time to make predictions, we can give the past 23 hour observed data and the present hour's data from the concerned RWIS site to predict the value for the present hour. The HMM will predict the class value of the variable used in the data. To compare actual and predicted values for detecting sensor malfunctions we can use the method that was described in the general classification approach.

Chapter 4

Experiments and Results

This chapter presents the experiments we performed to predict various weather variables using ML methods. We first discuss our experiments to predict the variables temperature, precipitation type, and visibility at various RWIS sites using different machine learning (ML) algorithms. In the next section we discuss our experiments to predict the variables temperature and precipitation type at various RWIS sites using HMMs. For each experiment we present the methodology, the results obtained, and discuss the accuracy of that method in predicting values reported by RWIS sensors and the use of this methodology in detecting RWIS sensor malfunctions.

4.1 Using ML Algorithms to Predict Weather Variables

To detect an RWIS sensor malfunction, we will be predicting the weather condition reported by the sensor and comparing it with the value reported by the sensor. As discussed in Chapter 3, we will be using, as input attributes in the dataset, the related information gathered from nearby RWIS sites that belong to the same group and the respective AWOS sites associated with them. We evaluated the performance of each algorithm on the data using the 10-fold cross-validation technique.

WEKA, described in Section 2.2, contains implementations of many ML algorithms. It allows the user to select an ML algorithm and apply it to a given dataset. Using the selected algorithm, it builds a model based on the training set provided and uses the model to classify instances in the training set and gives the performance results. A brief description of WEKA along with the command-line statements used to run different algorithms is given in Appendix B. We performed the experiments involving ML

algorithms mentioned previously using WEKA. We averaged the results of ten 10-fold cross-validation runs for our ML experiments.

4.1.1 Predicting Temperature

In this section we describe the experiments performed, using both classification and regression algorithms, to predict the temperature value at an RWIS site. The experiments were performed using default options provided by WEKA for the respective algorithms. All our datasets were built from the year 2003 data because at the time the experiments were performed 2003 was the latest year for which we had an entire year's data for the selected 13 RWIS sites.

4.1.1.1 Experiment 1: Predicting temperature using regression methods

In this experiment we predict the current hour temperature value at an RWIS site using regression algorithms. The temperature data from the RWIS site and the RWIS - AWOS in its group are used to form the feature vector. The results obtained from predicting temperature values were analyzed.

Methodology

We used the regression algorithms Linear Regression (LR), LeastMedSquare (LMS), M5P, Multilayer Perceptron (MLP), RBF Network (RBF), and Conjunctive Rule (CR) to predict the current temperature at an RWIS site. Our feature vector for the dataset consists of the current and the previous three hour temperature values obtained from the RWIS and the AWOS sites in a group of related sensors (the grouping of sites into sets is described in Section 3.1), and the current hour temperature at the RWIS sites. We calculate the temperature as the difference between the temperature values reported at an RWIS site and the projected hourly temperature (described in Section 3.2.1) at its corresponding AWOS site. For example, if the temperature at the RWIS site 67 is 32°F at

time *t* and the projected hourly temperature at the time *t* for its corresponding AWOS site, KLYU, is 30°F, then the temperature of site 19 for hour t in the feature vector is 2°F. During the calculation of hourly projected values we used historical averages (from years 1997 to 2004) for a month and the average temperature for a day to extract temperature values as deviations from the average. Using historical data provides additional information apart from what we already have (i.e., the RWIS and the AWOS data).

We used the weather data collected from the year 2003 to build our dataset using the described feature vector. For each RWIS-AWOS group we get a different dataset. To predict temperature at an RWIS site we use the dataset describing the sites in that group.

The current temperature value at the RWIS site to be predicted is the output attribute and all other attributes in the feature vector form the input attributes. We used the regression algorithms mentioned above and the dataset generated to try to predict the current hour temperature value at the selected 13 RWIS sites.

Results

Our testing involved ten 10-fold cross-validation runs so each instance in the dataset is predicted 10 times (i.e., one in each cross-validation), the average of these 10 values gives the final predicted value. We used absolute error between the actual value reported by the sensor and the value predicted by the algorithm, to evaluate the performance of the algorithm.

Figure 4.1 shows the mean of absolute error and the standard deviation obtained across all RWIS sites for each algorithm. The ordering of algorithms in the figure is according to the order in which algorithms were describing in Chapter 2. The mean absolute error obtained for each individual RWIS site for an algorithm, in a detailed form, are shown in Appendix C.

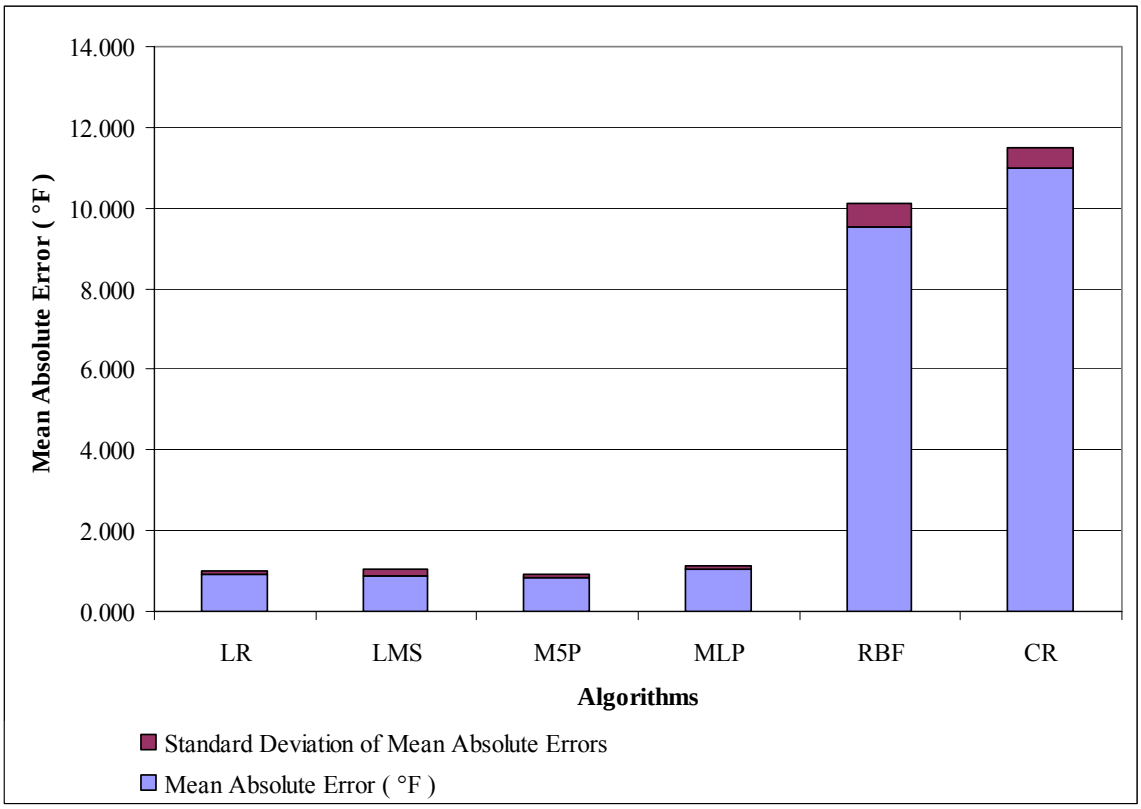

Figure 4.1: The mean of the absolute error and the standard deviation obtained from predicting temperature across all 13 RWIS sites using regression algorithms. The four methods LR, LMS, M5P and MLP clearly outperform RBF and CR.

Discussion

By comparing the mean absolute errors for CR and RBF with other methods, we see that they fail to predict temperature values for all sites. It may be that these algorithms could have performed better if we had significantly tuned the parameters for these algorithms. Mean error for LMS, LR and M5P is seen to slightly less than 1°F and a slightly more than 1°F for MLP. It can be observed from these values that the predictions of temperature value by these algorithms, LMS, LR, M5P and MLP is accurate to ±1°F.

The standard deviation value for the mean absolute errors across different sites for an algorithm measures the variation in the error values across the sites. A small standard deviation suggests the model is fairly independent of the site location and can predict

with the same accuracy on any site. This is favorable as one model can be used across all sites rather than having a separate model for each individual site. The CR and RBF algorithms show higher standard deviations when computed with the rest of the algorithms used, but this is likely a function of their high error. The best standard deviation results are obtained from M5P with 0.058 as the standard deviation value across sites.

Combining the results and giving priority to the algorithms that has a low absolute error and has a similar behavior across sites, we find that M5P performed the best followed by LR. It can be clearly seen that CR and RBF did not do well concerning prediction of temperature and were not used in the the other experiments that followed.

To detect RWIS temperature sensor malfunctions models built from M5P or LMS can be used. When the difference between the error reported for an hour and the mean absolute error obtained from testing the model is greater than 1.96 standard deviations, with standard deviation of error calculated from the test results, we can say with 95% accuracy that the sensor has failed.

4.1.1.2 Experiment 2: Predicting temperature using regression methods, with precipitation type included as inputs.

In this experiment we predict the current hour temperature value at an RWIS site using regression algorithms. The temperature data from the RWIS site and the RWIS - AWOS sites in its group along with the precipitation type seen at the RWIS sites are used to form the feature vector. The results obtained from predicting temperature values were analyzed.

Methodology

Due to the fact that presence of precipitation affects the temperature at a location, we

added it to the feature vector, so that the model built uses this information in predicting temperature. We used the regression algorithms LMS, LR, and M5P to predict the current temperature class value at an RWIS site. We use these three algorithms as they were the top three of the algorithms tried in Experiment 1 (see Section 4.1.1.1) and are arranged in ascending order with respect to the mean absolute errors across all sites (see Figure 4.1). Our feature vector for the dataset consists of the current and the previous three hour temperature values from the RWIS and the AWOS sites from a group of related sensors and the precipitation type observed at the current hour at the RWIS sites.

We used the weather data collected from the year 2003 to build our dataset using the described feature vector. The current temperature at the RWIS site to be predicted is the output attribute and all other attributes in the feature vector form the input attributes. We use the regression algorithms mentioned above and the dataset generated to try predict current hour temperature class value at the selected 13 RWIS sites.

Results

We use the mean absolute error obtained from the ten 10-fold cross-validation runs to evaluate the performance of the algorithm. Figure 4.2 shows a comparison of the mean absolute error and the standard deviation averaged for all RWIS sites, using the regression algorithms obtained from this experiment and from Experiment 1 (see Section 4.1.1.1) in which precipitation type information was not included. The mean absolute error obtained for each individual RWIS site for an algorithm, in a detailed form, from this experiment are shown in Appendix C.

Discussion

Of the three algorithms (LR, LMS and M5P) used, we see that M5P shows better results in predicting temperature with lesser mean absolute error and consistency in predictions across the sites (with respect to the standard deviation of errors across various sites). Significant variation of absolute errors reported by these three algorithms was not seen, with all them predicting temperature with an accuracy close to 0.95°F.

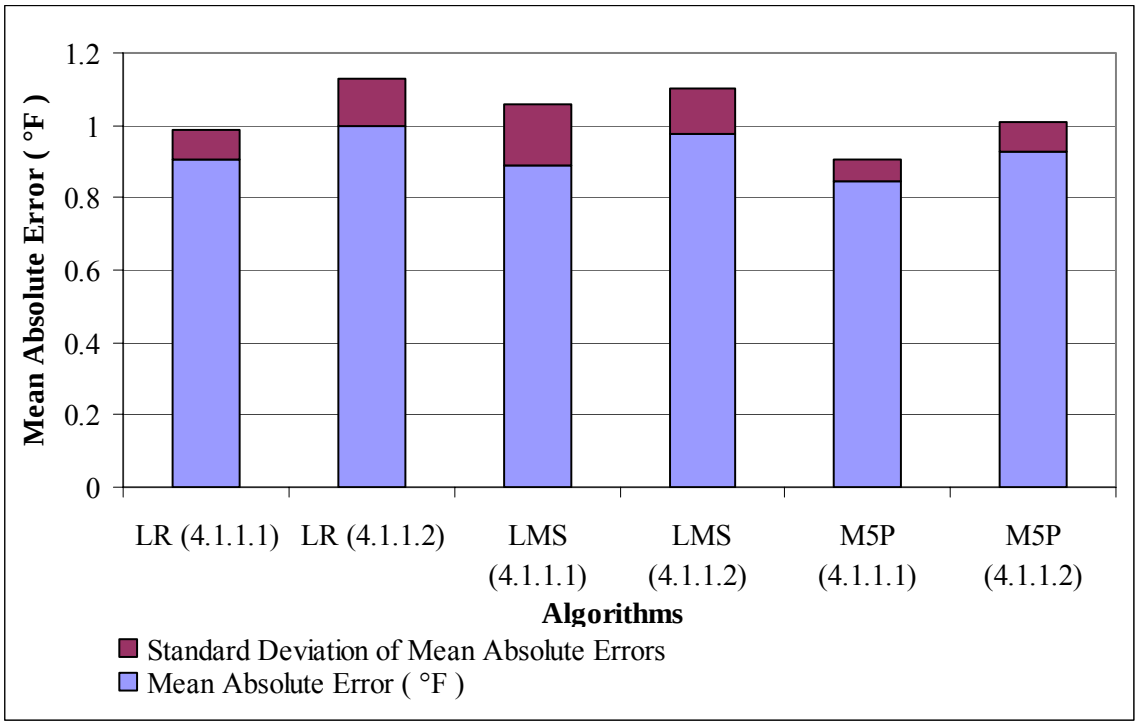

Figure 4.2: The comparison of mean of absolute error and standard deviation obtained from predicting temperature from Experiment 1 without precipitation (Section 4.1.1.1) and Experiment 2 (Sections 4.1.1.2) with precipitation.

We also note that including precipitation type in the dataset as an additional source of information does not decrease the error of the model built. It can be observed from Figure 4.2 that the mean absolute error for the algorithms LR, LMS and M5P is higher for this experiment when compared with Experiment 1. It was observed the mean of absolute error increased by approximately 0.09°F when precipitation type was included in the dataset.

4.1.1.3 Experiment 3: Predicting temperature class using classification methods

In this experiment we predicted the current hour temperature class value at an RWIS site using classification algorithms. We discretized the temperature values for its use in

classification algorithms. The temperature data from the RWIS site and the RWIS-AWOS sites in its group along with the temperature's class value at the RWIS sites are used to form the feature vector. The results obtained from predicting temperature class values were analyzed.

Methodology

We used the classification algorithms J48 decision trees and Naive Bayes (NB) to predict the current temperature class value at an RWIS site. Our feature vector for the dataset consists of the current and the previous three hour temperature values from the RWIS and the AWOS sites from a group of related sensors and the class value for the current hour temperature at the RWIS sites. We discretized the temperature value using the method described in Section 3.2.2 and the class distribution was set according to the example mentioned at the end of that section which divides temperature into nine different classes.

We used the weather data collected from the year 2003 to build our dataset using the described feature vector. The class value for the current temperature at the RWIS site to be predicted is the output attribute and all other attributes in the feature vector form the input attributes. We used the classification algorithms mentioned above and the dataset generated to try predict the class for the current hour temperature at the selected 13 RWIS sites.

Results

We used the absolute distance between the class value of the temperature reported by the RWIS sensor and the temperature class predicted to evaluate the performance of the classification algorithms used. A distance of 0 indicates that the predicted class value is same as the class value reported. The percentage of instances that were reported with a distance ranging from 1 to 6, for both J48 and NB used for predictions, are shown in the Figure 4.3.

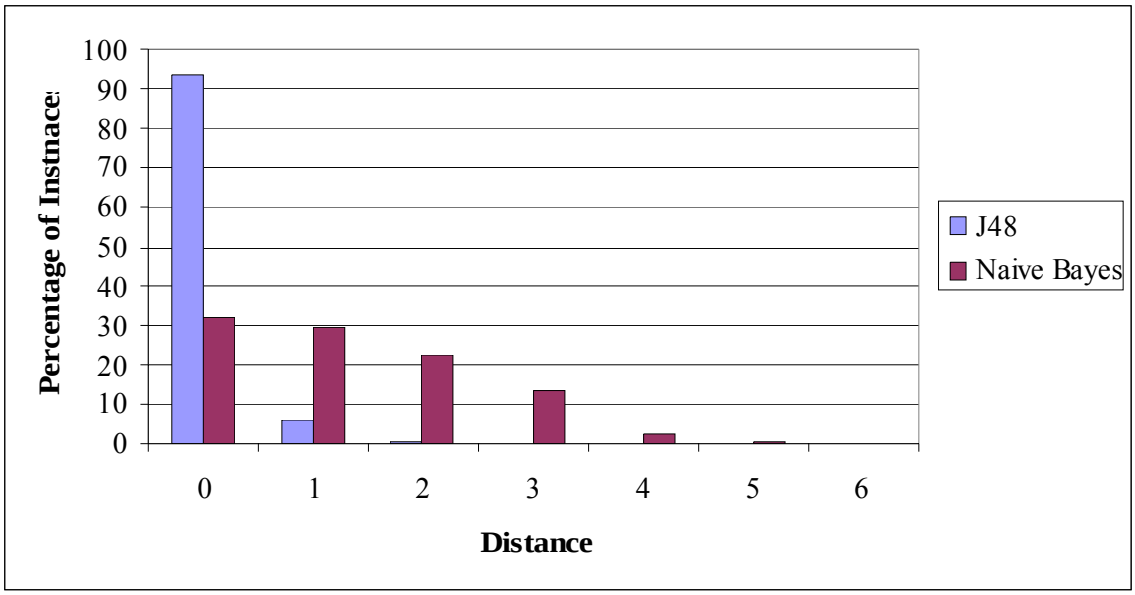

Figure 4.3: The distance between actual and predicted temperature class obtained from J48 and Naive Bayes algorithms.

Discussion

We see that from the results in Figure 4.3 that the J48 outperforms the the Naive Bayes algorithm by classifying 93.6% of the instances in the dataset whereas only 32.337% instances were correctly classified by the NB algorithm. No instances were reported having a distance of more than 3 between actual and predicted class value when prediction was done using J48. As 99.435% of the instances in the dataset were predicted with a distance of either 0 or 1, using the J48 algorithm, we can predict RWIS sensor malfunctions when the distance between the actual and predicted temperature class value is greater than 1. For all the malfunction cases in our dataset the difference in the class value is 2 or greater, thus we can say that sensor malfunctions can be detected with high accuracy with the J48 algorithm.

4.1.2 Predicting Precipitation Type

This section describes the experiments we performed using the the classification algorithms to predict precipitation type at an RWIS site. The experiments were performed

using the default options provided by WEKA for the respective algorithms. We employ only classification algorithms because precipitation type is reported by RWIS sensors in the form of class values and classification algorithms are used to classify the given instance into class values taken by the output attribute. The dataset was built from the year 2003 data because at the time the experiments were performed 2003 was the latest year for which we had an entire year's data for the selected 13 RWIS sites.

4.1.2.1 Experiment 4: Predicting precipitation type using classification methods

In this experiment we predict the current hour precipitation type reported at an RWIS site using classification algorithms. The temperature data from the RWIS site and the RWIS-AWOS sites in its group along with the precipitation type reported at the RWIS sites are used to form the feature vector. The results obtained from predicting precipitation type were analyzed.

Methodology

We used the classification algorithms J48 decision trees, Naive Bayes (NB) and Bayesian Networks (Bayes Nets) to predict the precipitation type for the current hour at an RWIS site. The K2 algorithm is used to learn the Bayesian network structure. Our feature vector for the dataset consists of the current and previous three hour temperature values from the RWIS and the AWOS sites in a group of related sensors and the precipitation type observed at the current hour for the RWIS sites. We included the temperature information to help in the prediction process as there is a correlation between temperature and precipitation observed at a location. Precipitation information from AWOS sites was not used because the effect of precipitation is localized and does not effect the occurrence of precipitation at nearby and other locations.

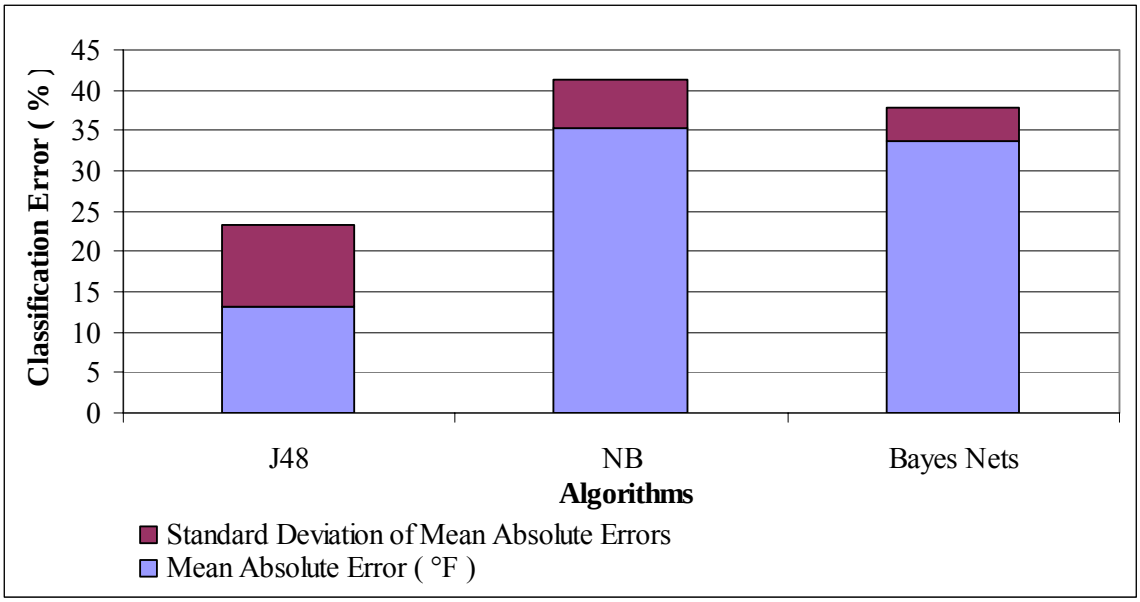

Figure 4.4: The classification error and the standard deviation obtained from predicting precipitation type across all 13 RWIS sites using classification algorithms.

We used the weather data collected from the year 2003 to build our dataset using the described feature vector. The precipitation type at the current hour for the RWIS site to be predicted is the output attribute and all other attributes in the feature vector formed the input attributes. We used the classification algorithms mentioned above and the dataset generated to try predict current precipitation type for the selected 13 RWIS sites.

Results

When classification algorithms are run using WEKA, the output is presented as a confusion matrix and statistical results such as the classification error, root mean squared errors and the percentage of correctly classified instances are given by WEKA. Figure 4.4 shows the classification error (as reported by WEKA) and standard deviation obtained from predicting precipitation type across all RWIS sites for an algorithm. These values were obtained after averaging the reported values for each cross-validation run. The classification error and standard deviation values obtained for each individual sites for the

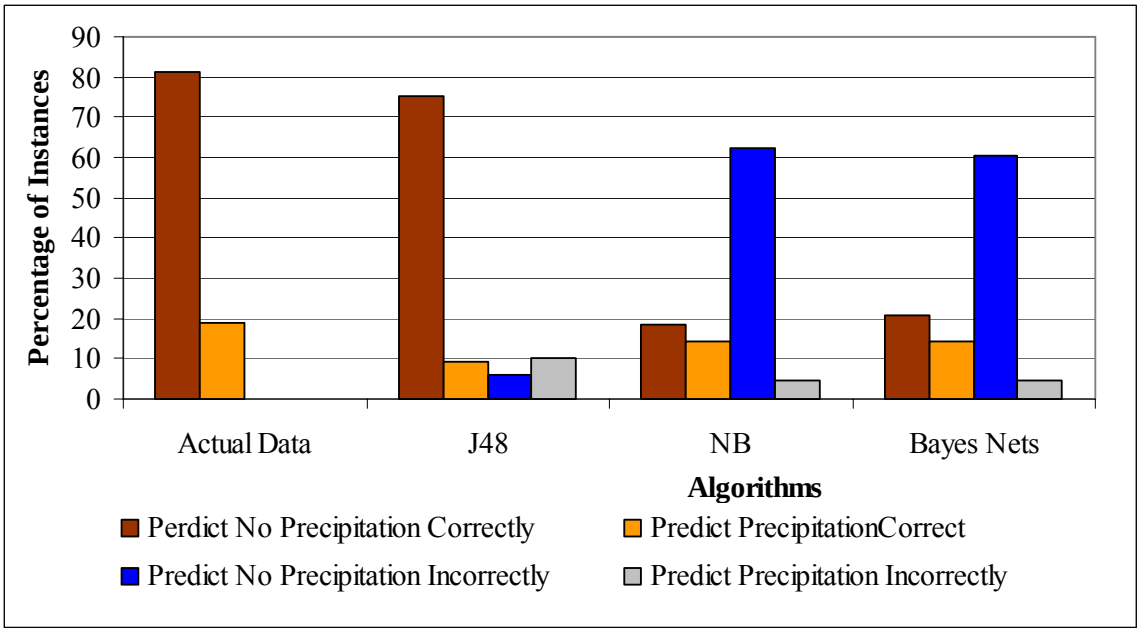

Figure 4.5: The percentage of instances with precipitation present and with no precipitation present predicted using classification algorithms.

algorithms used in a detailed form are shown in Appendix C.

For predicting precipitation type, we use the percentage of instances that were correctly classified and incorrectly classified when precipitation was present and when no precipitation was present to determine the accuracy of the algorithm used. Figure 4.5 shows the percentage of instances that were reported as precipitation present and no precipitation present in the actual data. For each algorithm the figure shows the percentage of instances these values were predicted correctly and incorrectly when precipitation was present and when precipitation was not present. A detailed analysis of the these percentage values of correctly and incorrectly classified instances for the algorithms used across each individual site is given in Appendix C.

Discussion

From the classification errors reported by WEKA for the three regression algorithms used, we see that J48 performs better in predicting precipitation type but a high standard

deviation across sites shows that it is not consistent in predictions across different sites, with site 35 having 0.077 as classification error and site 67 with 0.356 as its classification error. Other than site 67 none of the sites have classification error above 0.26 which is lower than the mean absolute errors reported by NB and Bayes Net.

Some RWIS sites report precipitation as present or not present while the remaining sites report the type of precipitation observed. In order to compare sites, we combined all different type of precipitation together and reported precipitation as present. This allows comparison o the percentage of instances when precipitation was correctly classified between sites.

Predicting precipitation is a challenging task. It observed from analyzing the RWIS data that the instances with precipitation present were very few, this is because precipitation does not continue for a long time and may be present for an hour or two leading to few observations reported over a period of time. As seen in Figure 4.5, Naive Bayes and Bayesian Networks predict only about 19% of the instances as no precipitation present when in the actual data 81% of the instances were reported as no precipitation. They perform poorly for classifying no precipitation. We found that J48 classifies instances with no precipitation present correctly, with identifying 75% of the instances as no precipitation present when in actual data 81% of the instances had reported no precipitation. But it fails to report the presence of precipitation with the accuracy it predicts precipitation.

The detection of RWIS sensor malfunctions using prediction of precipitation type is a tough task because of the failure of the algorithms to correctly classify precipitation. We found that combination of J48 and Bayes Net can be used to detect malfunctions, with absence of precipitation being calculated through J48 and Bayesian Networks being used to report presence of precipitation. We choose Bayesian Networks over Naive Bayes because of its smaller mean absolute error and standard deviation. Due to varying accuracies prediction among different sites (see Appendix C), each individual site

requires its own model and specific percentages with which they correctly classify precipitation to identify the sensor malfunctions. For example, for site 62, J48 mis-classifies 3.84% of instances when predicting no precipitation and Bayes Nets mis-classifies 0.77% of instances when reporting presence of precipitation. For this site we can say that when J48 predicts incorrectly when no precipitation is present and with the difference in classification error for the site (i.e., 0.062 for site 62) and the absolute error reported by WEKA is greater than 1.96 standard deviations (standard deviation calculated from the errors obtained on the test set) then we can say with 95% accuracy that the sensor has failed. The same approach is followed when Bayes Nets wrongly reports absence of precipitation. Combination of J48 and Bayes Nets produce high accuracy in detection sensor malfunctions when each of them is individually responsible for classifying absence of precipitation and presence of precipitation respectively.

4.1.3 Predicting Visibility

This section describes the experiments we performed using regression algorithms to predict visibility at an RWIS site. The experiments were performed using the default options provided by WEKA for the respective algorithms.

4.1.3.1 Experiment 5: Predicting visibility using regression methods

In this experiment we predict the current hour visibility reported at an RWIS site using regression algorithms. The temperature data from the RWIS site and the RWIS-AWOS sites in its group along with the precipitation type and visibility reported at the RWIS sites are used to form the feature vector. The results obtained from predicting visibility were analyzed.

Methodology

We used the regression algorithms LR, LMS, M5P, and MLP to predict the visibility for

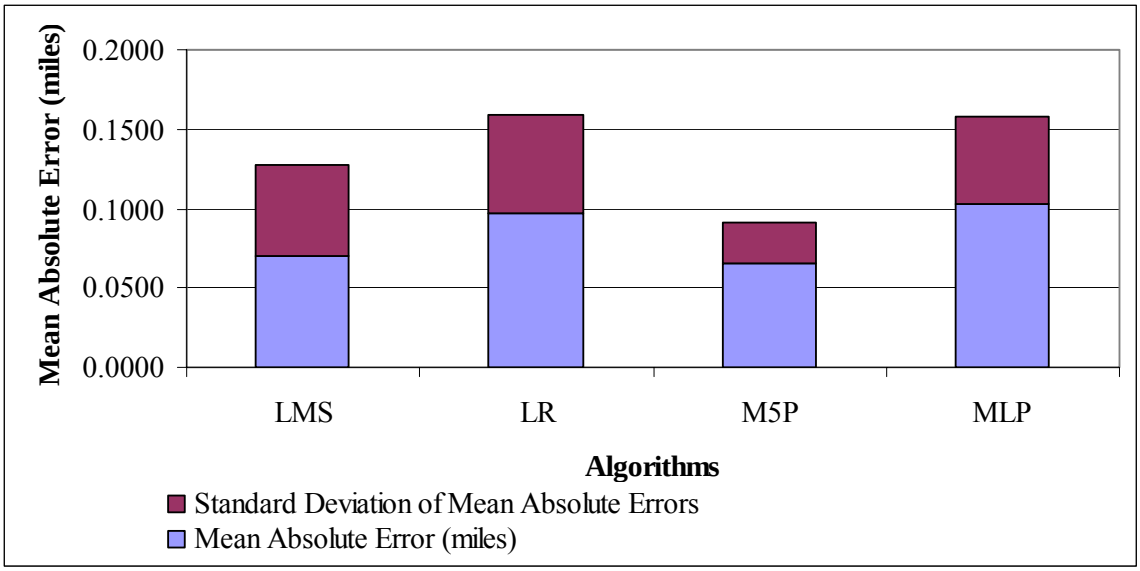

Figure 4.6: The mean of the absolute error obtained from predicting visibility across the RWIS sites that report visibility using various algorithms.

the current hour at an RWIS site. As visibility is affected by the presence of precipitation, we included precipitation as an attribute in the feature vector. Our feature vector for the dataset consists of the current and previous three hour temperature values obtained from the RWIS and the AWOS sites in a group of related sensors, the precipitation type observed at the current hour for the RWIS sites and the visibility at the current hour the RWIS sites. The RWIS sites that report visibility are included in the feature vector.

We used weather data collected from the years 2002 and 2003 to build our dataset using the described feature vector. The visibility at the current hour at the RWIS site to be predicted is the output attribute and all other attributes in the feature vector form the input attributes. We used the regression algorithms mentioned above and the dataset generated to try predict current visibility for the RWIS sites that report it.

Results

The mean absolute error obtained from site 67 in RWIS Set 1 was excluded when averaging values for the sites used in prediction, this was done because site 67 reports

visibility up to 10 miles whereas all other sites report only up to 1.09 miles. Figure 4.6 shows the mean of absolute error and standard deviation obtained from predicting visibility across all RWIS sites, excluding site 67, for an algorithm. The overall mean absolute errors were obtained after averaging the values reported for each cross-validation run. The mean absolute errors obtained for each individual sites with respect to the algorithm used are given in Appendix C.

Discussion

We found that the LMS and the M5P algorithms predict visibility with almost the same absolute error, with LMS being a little bit higher. But the standard deviation of errors across various sites was low for M5P when compared to LMS. Thus, we can say M5P can be used to report visibility across different sites accurately and its predictions are site independent and a model created using one site can be used to predict values at another site. We can use M5P to detect sensor malfunctions. We can say that when the the difference between error from prediction and the mean absolute error for M5P is greater than 1.96 standard deviations, with standard deviation of calculated from the error values obtained during cross-validations, the sensor is malfunctioning.

4.2 Using HMMs to Predict Weather Variables

Based on the instance given, an HMM tries to predict the most probable path taken across the states. For the weather data the "path" is the predicted values of the weather sensor using the surrounding sensors as additional information. Both HMMs and classification algorithms can be used to classify a discrete attributes such as precipitation type. When the data is present in the form of time series, such as hourly precipitation type observations for a time will form a hourly series of values, HMMs can be used to identity the path of predicted values from which we can further obtain the values observed through the path.

We used the modified Viterbi algorithm (see Table 3.2) to predict the symbol observed when a state in the most probable path is reached for the given instance. HMMs require a training dataset to build the model, that is to estimate the initial state, transition and emission probabilities. The test dataset is used to evaluate the model. Multiple *n*-fold cross-validation can be used to estimate the model performance.

As HMMs require the presence of all symbols in the symbol sequence, any day with no value (i.e., missing) reported by a sensor at an hour for any of the RWIS sites in the set, was omitted from the dataset used for predictions. HMMs require class value information, thus we discretized the temperature for using it with HMMs by following the method described in Section 3.2.2. We found that with larger class distribution for sites to be used for prediction helps in better comparison between the predicted and actual class values whereas the nearby sites are used to provide additional information about the surrounds weather conditions and they need not be taken with greater precision. Thus for the RWIS site that were used for predictions, the temperature was broken down into nine classes based on the value obtained for number of standard deviations the actual temperature value differs from the predicted value, the classes were divided as

Class Value		**Class Value**	
1	num_stdev < -2	6	0.25 < num_stdev ≤ 0.5
2	-2 ≤ num_stdev ≤ -1	7	0.5 < num_stdev ≤ 1
3	-1 < num_stdev ≤ -0.5	8	1 < num_stdev ≤ 2
4	-0.5 < num_stdev ≤ -0.25	9	num_stdev > 2
5	-0.25 < num_stdev ≤ 0.25		

For the RWIS sites that were appended along with the site to be predicted in the class string, the temperature was broken down into five classes with the classes divided as

Class Value		Class Value	
1	num_stdev < -1	4	0.5 < num_stdev ≤ 1
2	-1 ≤ num_stdev ≤ -0.5	5	num_stdev > 1
3	-0.5 < num_stdev ≤ -0.5		

For example, to predict temperature values at site 19 we use temperature values from site 27 and 67 as additional information. The temperature of site 19 was be broken into 9 classes and temperature at sites 27 and 67 will be broken into 5 classes.

In the following experiments related to HMMs, we trained the HMM (Baum-Welch algorithm) using the methods available in the 'HMM Toolbox for MATLAB' developed by Murphy [1998] and the modified Viterbi algorithm was used for testing. We performed ten iterations of the Baum-Welch algorithm during the training of the HMM. We set the number of states in the model to 24 and each state was allowed to emit all possible symbols obtained from the training set. As we are using hourly readings for a day, each instance in the dataset is a string of symbols with a length of 24.

4.2.1 Predicting Temperature

This section describes the experiments we performed using HMMs to predict the class value of the temperature at an RWIS site.

4.2.1.1 Experiment 6: Comparison of two methods for training HMM

Two different methods were devised to train the HMMs. In this experiment we predict the temperature class value reported at an RWIS site using HMMs. The temperature class information from the RWIS site and the other RWIS sites in its group are used to form

the feature symbols. The results obtained from predicting temperature class value were used to compare the two methods.

Methodology

We trained the HMM to predict the temperature class value in two different ways

Method A: In this approach we merge all the data from a group and then train the HMM using this data. As noted earlier, the site being used for prediction has the temperature split into 9 classes and other sites in the group have temperature split into 5 classes. Thus, for each site we have its own merged dataset.

We create the dataset in which each instance is a string of symbols representing hourly readings per day, where a symbol for an hour was obtained from appending that hour's temperature class value for all of the RWIS sites in a group together. For example to predict temperature class values at site 19 we will be using the temperature classes from sites 27 and 67, if the temperature class values at a time *t* seen at sites 19, 27 and 67 are 6, 3 and 2 then the class string generated for the time *t* will be 632. We estimated the initial state, transition, and emission probabilities by applying the Baum-Welch algorithm on this dataset with combined class information.

Method B: In this approach the data from the sites used is not merged. Data from each site is used for training the model and the emission probabilities of all sites in the group are multiplied to obtain a single emission matrix that is used in the final model.

We created a separate dataset for each RWIS site, in which each instance is a string of symbols representing hourly temperature class value per day. We used the dataset for each RWIS site to estimate the initial state, transition and emission probabilities using the Baum-Welch algorithm. The emission probability matrices obtained for each of the RWIS sites in a group were then joined together by multiplying the respective values in each box in the matrix. The resultant emission matrix was made stochastic, that is, all

rows and columns are made to sum to 1. We then applied the m-estimate upon the probability values in the matrix, using m = 20 and p = 1 / #classes, to make the probabilities with very small values bigger, where in our case the number of classes denotes the total number of class values taken by the variable we are trying to predict. By applying the m-estimate we get new values in each block of the matrix given by

$$matrix(row.col) = (matrix(row,col) + mp) / (1 + m)$$

For example, in the matrix shown below we apply m-estimate on it using m=20 and p=½ (considering the number of classes be equal to 2) to get the resultant matrix.

initial matrix

0.00111	0.99889
0.49751	0.50249

matrix after applying m-estimate

0.47624	0.52376
0.49988	0.50012

We used the initial state and transition probabilities for the RWIS whose temperature value is to be predicted in the model.

At the time of performing these experiments we had data from RWIS sites from 2002 to April 2004, we used the entire data available for building the datasets. We used the temperature values at the RWIS sites for the years 2002 and 2003 in the training set while test set contained temperature information from January 2004 to April 2004.

We discretized the temperature values, by breaking the temperature into nine classes. Training was done using the two methods mentioned above and temperature class value was predicted for instances in test data using the modified Viterbi algorithm.

Results

We used the absolute distance between the actual and predicted class values for temperature as a measure to estimate the performance of the algorithm. Figure 4.7 shows the percentage of instances in the training set with each distance between the actual and predicted class averaged across the results from different RWIS site, when the HMM is

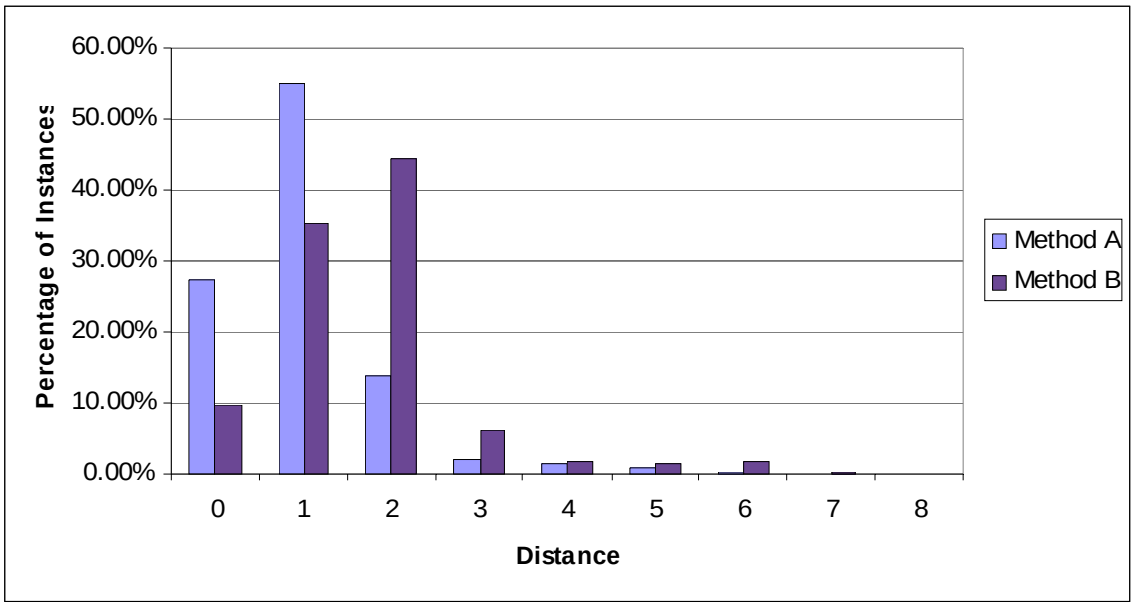

Figure 4.7: The percentage of instances with each distance to actual value when the HMM is trained using the two different methods.

trained using *Method A* and *Method B*. A distance of zero indicates a perfectly classified instance.

Discussion

Smaller distances between the actual and predicted values indicate more accuracy in predictions. From Figure 4.7 we see that Method A classified the more instances at with distance 1 or less whereas Methods 2 had comparatively lesser number of instances classified with distance below 2. This indicates that Method A outperforms Method B. As Method A gave better results we will be following this approach for training the HMM in the experiments performed in the methodologies described next.

4.2.1.2 Experiment 7: Predicting emperature class using HMMs

In this experiment we predict the temperature class value reported at an RWIS site using HMMs. The temperature class information from the RWIS site and the other RWIS sites

in its group are used to form the feature symbols.

Methodology

Using temperature data collected for each RWIS site, ranging from January 2002 to April 2004, we created a dataset for each RWIS site to be used for predictions. In this dataset each instance is a string of symbols representing hourly readings for a day, where a symbol for an hour was obtained from appending that hour's temperature class value for all RWIS sites in a group together. In each class string the RWIS site predicted was taken first and the other sites in the group were added in order of their nautical distance from the concerned RWIS site. This order is useful while finding the closest symbol when a class string seen in the test set is missing from the training set used. We estimated the performance of HMM in predicting the current hour's temperature class value at an RWIS sites using the dataset of the group this site belongs to and applied a single 10-fold cross-validation to estimate the model evaluation.

Results

We evaluated the performance of the HMM using the absolute distances between the actual and predicted values. The respective distance values for all RWIS sites in a group were averaged so as to reflect on the performance of the HMM in predicting temperature class values for that respective set. Figure 4.8 shows the percentage values for each distance for the three sets, with the percentage calculated as the number of instances classified with a certain distance over the total instances present in the dataset, with each instance being an hourly reading. The percentage for each distance across all the RWIS sites used for prediction is shown in Appendix C.

Discussion

We do not combine the results of different sets to reach an overall percentage value as the format of the symbol sequence in dataset for each site was different. It can be seen from Figure 4.8 that the most of the temperature class values are predicted with a distance of 1,

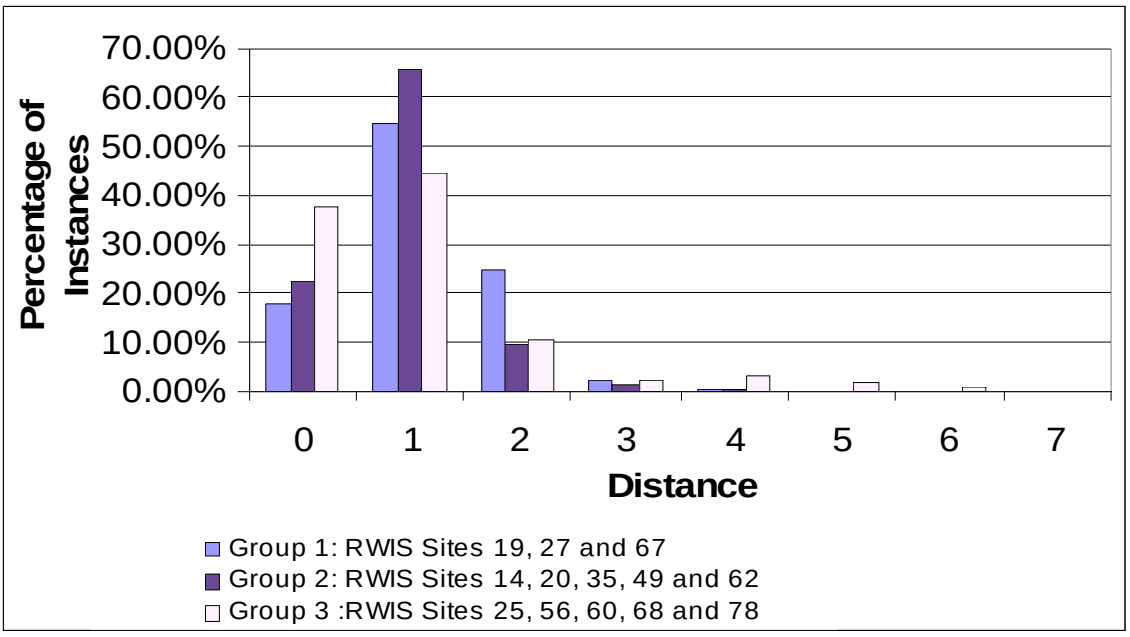

Figure 4.8: Percentage of instances having a certain distance from the actual class value when predicting temperature class using HMMs.

for all sets. This is also reflected in the results from individual sites shown in Appendix C, with the exception where a distance of 2 is seen most of the times.

The percentage of instances predicted with a distance of 3 or less in the groups 1, 2 and 3 are 99.75%, 99.43% and 94.34% respectively. Approximately 3% of the data in Group 3 is predicted with a distance of 4, which is due to the results obtained for RWIS site 67 in which approximately 15% of the data is predicted with distance 4. Taking aside the results for site 67, we can say that, when the distance between the actual temperature class and the predicted class is more than three then there is a malfunction in the RWIS temperature sensor. The erratic behavior of site 67 may be due to presence of errors in it.

4.2.1.3 Experiment 8: Site independent prediction of temperature class using HMMs.

In this experiment we predict the temperature class value reported for a site in an RWIS

group. A single dataset was generated for a group that has the first class value in the class string as the class value to be predicted. The temperature class information from the RWIS sites in a group are used to form the feature symbols. The results obtained from predicting temperature class were analyzed.

Methodology

In order to increase the size of data used for predictions and to perform site independent predictions we create a single dataset for each RWIS group. Each of the RWIS sites in a group was taken as the predicted site and the symbol strings obtained from it were appended to the dataset. The class string was constructed using the predicted site's class value which is added first followed by the class values of the other sites in the group with the predicted site are added. The order of the near sites was chosen with respect to the distance from the predicted site, thus the site closest to the predicted site was added first and so on. For example, to create a single dataset for the RWIS Set 1 we first add data instances taking RWIS site 19 as the site to be predicted with sites 27 and 67 as nearby sites, followed by taking site 27 to be the site predicted and then using site 67 as the site predicted. Using this method, we get a single large dataset whose size is equal to the sizes of datasets generated for each site in a RWIS group separately added together. By predicting the first class seen in the dataset we perform a site independent evaluation of the HMM and focuses on prediction accuracy in a group.

The bigger dataset was generated for each group using the temperature data from the years 2002 and 2003. We used this dataset was used to predict the first class value (which is the predicted site) seen in the class string using HMMs and the model was evaluated using ten 10-fold cross validation runs.

Results

The distance between actual and predicted class value was used to evaluate the performance of the model. Figure 4.9 shows the percentage of instances (with an instance being an hourly reading) with a certain distance between the actual and predicted value,

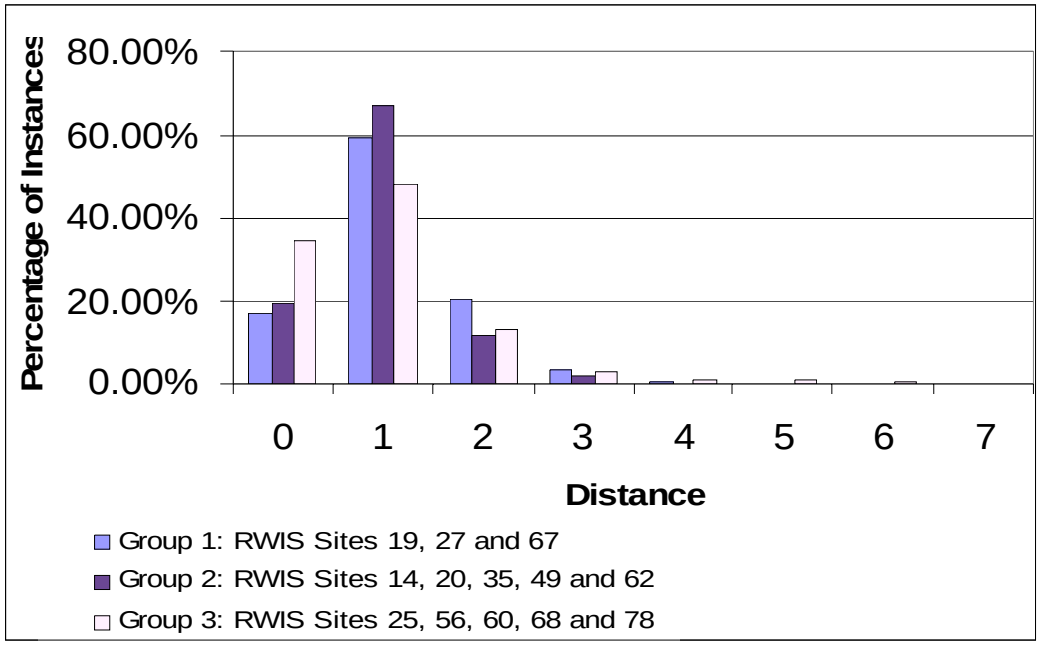

Figure 4.9: Percentage of instances having a certain distance from the actual class value when predicting temperature class by applying ten 10-fold cross-validation on HMMs and using the extended dataset focusing on predicting class value for an RWIS group.

for each of the three RWIS sets used. The percentage values were obtained after averaging the values reported for each cross-validation run.

Discussion

Comparing the results of Experiment 7 with this experiment, we observed (see Figures 4.8 and 4.9) that the percentage of instances with a certain distance is almost the same, that is the the results obtained for a group using a combined dataset and from averaging results from within a group where each site was trained using its dataset are about the same. From this we conclude that using a single model built from the combined dataset values for any site in a group can be can be predicted with accuracy comparable to that of the overall group.

The percentage of instances predicted with a distance of 3 or less in the groups 1, 2 and 3 were 99.76%, 99.71% and 97.88% respectively, which covers almost the entire data.

Thus, we can say that when the distance between the actual temperature class and the predicted class is more than three then there is a malfunction in the RWIS temperature sensor.

4.2.2 Predicting Precipitation Type

This section describes the experiments performed using HMMs to predict the class value of the precipitation at an RWIS site.

4.2.2.1 Experiment 9: Predicting precipitation type using HMMs.

In this experiment we predict the precipitation type reported at an RWIS site. The precipitation type information from the RWIS sites in a group are used to form the feature symbols. The results obtained from predicting temperature class were analyzed.

Methodology

We performed ten 10-fold cross validations using HMMs, to predict the precipitation type at an RWIS site. Years 2002 and 2003 were selected for generation of the data as these where the most recent years for which we had entire yearly data for the RWIS sensors when this experiment was conducted. The dataset used for prediction contained precipitation type information for the years 2002 and 2003 with each instances consisting of hourly class strings for a particular day. The class string in the dataset instance was obtained from appending the precipitation type of RWIS site used for prediction and from the nearby sites, which were arranged according to the distance from the site used for predictions.

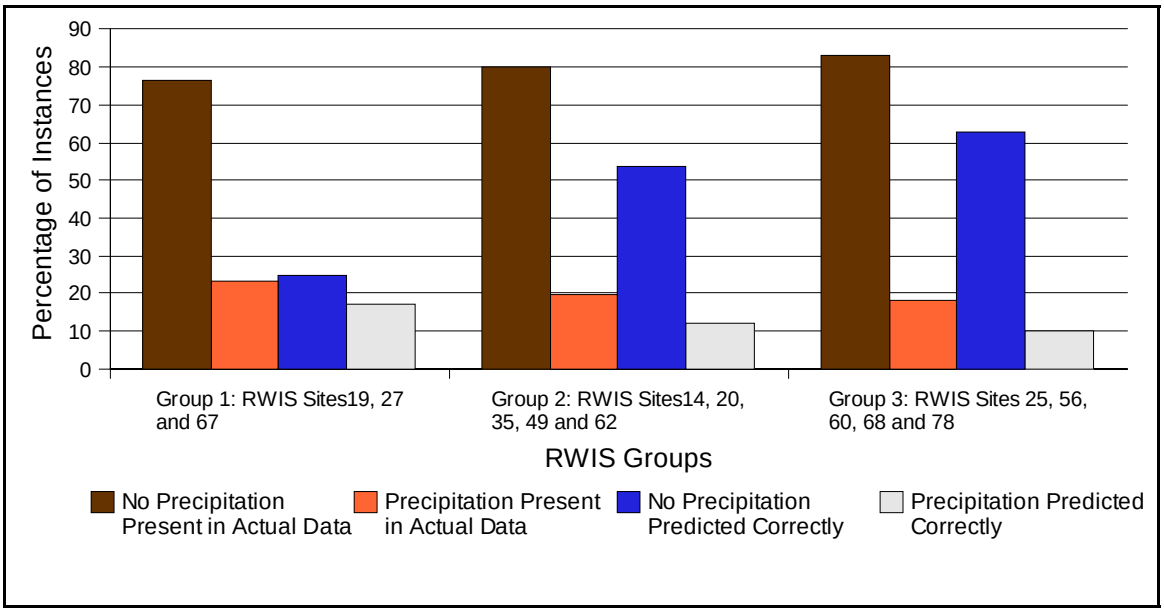

Figure 4.10: The percentage of instances with precipitation present and with no precipitation present predicted correctly using classification algorithms.

Results

We cannot use the distance between actual and predicted value to be used as a measure of performance evaluation, as each precipitation type reported is unique and has a specific meaning. We observed that some RWIS sites report precipitation type as present or not present while the remaining sites indicate the type of precipitation when present. In order to compare the performance across all sites, any form of precipitation occurring was taken as precipitation present, averaged over all sites present in a set. Figure 4.10 shows the percentage of instances in the dataset that were correctly predicted as precipitation present or as not present, along with the percentage of instances where no precipitation and precipitation present were reported in the actual data. These values for each of the 13 RWIS sites on which prediction was done is shown in Appendix C.

Discussion

From the percentage values in Figure 4.10 and Appendix C we can say that HMM fails to predict precipitation when actually present for most of the sites with not even half of the instances that report precipitation being correctly classified. For predicting absence of

precipitation most of the sites had poor values but some showed better results. It can be seen from the values in Appendix C that the percentage of instances correctly classified varies with respect to a site. We see that there are a large number of instances where the absence of precipitation was reported as precipitation and as these are seen for large number of instances, this can be seen as a factor which reduces the accuracy of the model. We find that there are also a lot of misclassifications occurring during prediction of precipitation type values. From the these results we can conclude that HMM trained using the methodology described is not a good option for predicting precipitation and thus cannot be used for detection of RWIS precipitation sensor malfunctions. This may be due to the training methodology we had used in our HMM.

Overall Conclusions

We compare the values predicted by the various ML methods to the actual values observed at an RWIS sensor to detect sensor malfunctions. Accuracy of the algorithm in predicting values plays a major role in determining the accuracy with which malfunctions can be identified. From the experiments performed to predict temperature at an RWIS sensor we found using current and previous three hours temperature readings and the temperature at an RWIS site in the dataset, the classification algorithms LMS and M5P gave results accurate to ±1°F and had low standard deviation across sites. LMS and M5P were identified to be able to detect sensor malfunctions accurately. The algorithms RBF Networks and CR failed to predict temperature values. The use of precipitation as an additional source of information had no significant changes on the accuracy of the algorithms. A threshold distance of 2 between actual and predicted class value was identified to be able determine sensor malfunctions when using J48 to predict temperature class values.

The J48, Naive Bayes and Bayes showed varied results when predicting presence or absence of precipitation. A combination of J48 and Bayes Nets was identified to be able to detect malfunctions of a senor when predicting precipitation type. Visibility was best classified using M5P. An issue in using visibility is that RWIS sensor measures sensors

to a maximum of 1.09 miles and all distances above as taken as 1.09 miles, this prevents the algorithms from capturing changes that happen beyond this point.

HMMs were found to be effective in classifying temperature class but failed to predict precipitation type information. A threshold distance of 3 was identified to accurately identify sensor malfunctions. When predicting temperature class using HMM, a single model of a group obtained from using the combined dataset gave similar accuracy when compared to average of accuracies from predicting temperature at single sites over a group. From this we conclude that use a single model, built from using datasets of all sites together, can be effectively used to identity malfunctions at any site in the group.

Chapter 5

Related Work

Machine learning methods have been used in various fields such as bio-informatics, natural language processing and speech recognition. In this chapter we discuss work related to ours using RWIS technology. We then discuss the work done using ML methods in areas related to weather data modeling and forecasting, and time series prediction.

5.1 Using RWIS sensors

The Road Weather Management Program of the Federal Highway Administration (FHA) along with National Weather Service (MWS) sponsor research projects which deal with using weather information obtained from RWIS sensors for roadway maintenance and related operations. These projects are described in a technical report by FHA [2004].

Knight et al., [2004] describe the advantages and difficulties in integrating the RWIS, AWOS and Automated Surface Observation System (ASOS) sensors together to form a mesonet, which can be used for weather forecasting. They discuss the benefits of including data from different networks operating the same region in weather forecasting. Our work differs from theirs in that we focus on predicting individual sensors, rather than trying to produce a model for an entire network of sensors.

Gallus et al., [2004] use Artificial Neural Networks to develop a time series prediction model for predicting frost conditions at roadways. From 180 different weather variables, they found the 8 – 10 most important variables that are beneficial in predictions by neural

networks. The presence of frost was predicted. Frost predictions using RWIS data with ANNs did not provide good results, which was attributed to the methods used by RWIS in collection frost information. They also compare the data from RWIS and AWOS networks and report bias in values reported by the sensors, which is attributed to sitting positions of these sensors. In our work we use the Multilayer Perceptron algorithm to predict continuous variables. We include previous hours information in the dataset and which improves the accuracy of our results. We found some correlation between RWIS and AWOS sites that are grouped together, the bias found by Gallus et al. may be attributed to the location of the sensors. They used sensors in Iowa and we the sensors from Minnesota, which are different climatological regimes.

5.2 Weather Data Modeling and Forecasting using Machine Learning Algorithms

Hall et al. [1999] use ANNs to predict the probability of precipitation and the amount of precipitation. They include weather-related variables measured both at ground and at the upper atmosphere in the dataset along with observed rainfall information. They build two different network models to determine the probability of precipitation and the amount of precipitation. They report a change in significant variables for predicting precipitation for cold and warm seasons, which led to the development of different models for each season. They build network models which allows the users to change input variables in cases when inclusion of some variables causes the performance to drop thus allowing year round interoperability of the model and use of the model at different locations. They find that precipitation occurs when the precipitation probability predicted by the model is above 38.5%. They report a high accuracy in the predictions made, for both probability and amount of precipitation, by the use of neural networks. In our work, we use a single model whose predictions are independent of the time from which the input is derived. As indicators of model accuracy for predicting precipitation type we use the percentage of instances that are classified correctly and incorrectly for both cases when precipitation is

present and precipitation is not present.

Schizas et al., [1991] use artificial neural networks for predicting minimum temperature of a day by taking temperature, wind, visibility and previous day's minimum temperature as inputs. Their best results are obtained by a network with two hidden layers and 40 output units with each output unit representing a range in temperature. Gain and momentum were chosen to be 0.01 and 0.9 respectively. Their network predicts with an accuracy of 68% at a temperature confidence range of ±3°C. In out work we use previous hours temperature values with the current temperature to improve the prediction process. We use the MultiLayer Perceptron algorithm to predict hourly temperature values. The network we build has the number of hidden units determined by the number of attributes and the classes.

Park et al., [1991] use artificial neural networks to forecast electric load to detect grid failures. They use a combination of time series and artificial neural networks They build the network using past and present information of load and weather variables. Each network is built to trace load patterns to detect errors and to recognize different load conditions. They build different networks by varying the number of hidden layers (using 1, 5 and 10 hidden layers) and using load and different forms of temperature, such as average, peak and low and past temperature values. They report the best results when using the past 2 days information on a network with 10 hidden units. Our work differs from theirs in that we have the number of hidden units used by the model fixed. We use the average of the sum of number of attributes used and the number of classes as the number of hidden units. We used the temperature values in the same form reported by the RWIS sensors.

Cano et al., [2004] use Bayesian Networks to predict rainfall and wind conditions from a data mining point of view. The weather data was collected from a set of 100 stations that were arranged in a grid-wise manner. They construct the Bayesian network using historical data obtained from the sites and was is to predict present conditions. Each site

used for modeling is taken as a node in the Bayesian network. They use the K2 learning method in training of the Bayesian network. To improve the search criterion the parents in the network are allowed to include nodes which have a climatic similarity with the parent node. They report improvement in the efficiency of the search process and better results when this condition is applied. Our work differs from theirs in that we use the K2 search algorithm without giving it any prior knowledge for building the network. This allows the network to identify patterns that are not seen or measurabe.

McMillan et al., [2005] build a Bayesian hierarchical regime switching model to describe the behaviors of ozone over space and time. The model built uses the relationship between ozone to estimate spacial fields of ozone and weather condition. The model uses weather conditions like temperature, wind speed and wind direction to forecast ozone levels. Bayesian hierarchical modeling is used to build the models. The weather conditions are treated as fixed and know. The changes in the ozone field are treated as first-order Markov models in time. The models are used to detect ozone levels from the given meteorological conditions and are used to capture key ozone behaviors. The model captures various dependencies of ozone on the meteorological factors. We use Bayesian models to predict temperature, precipitation type and visibility at an RWIS sensor. We use Hidden Markov Models to predict temperature class values by taking the data in the form of a time series with each sequence consisting of hourly temperature class values.

Lau & Weng [1995] use wavelet transform to climate time series analysis. Wavelet transformations are used for study of non-stationary processes occurring over a period of time. They applied a wavelet transform to study variations in global ice volumes and temperature data. They provide a tutorial on the use of wavelet transform in weather related time series domain In our work we use Hidden Markov Models when data was formatted as a sequence in the time series.

5.3 Time Series Prediction using HMMs

Bellone et al., [2000] use a non-homogeneous HMM to predict precipitation amounts during winter months. They use weather data consisting of precipitation amounts, daily geopotential height, temperature, atmospheric pressure and relative humidity from different locations in Washington state area to build and evaluate the model. In the HMM we developed to predict precipitation type, we use temperature information along with precipitation type as inputs. Their model uses 6 states where each state corresponded to different amount of precipitation and assumed precipitation occurrence to be conditionally spatially independent. Our work differs from theirs in that to predict a variable such as precipitation type for a site we use precipitation type information from nearby sites. We did not include any other variable when predicting a particular variable. Our model has 24 states and the symbols observed at each state are used for predicting the output values. We modify the Viterbi algorithm to use the symbols emitted at each state in the most probable path for a given sequence to determine the predicted value.

Zhang [2001] uses HMMs to predict and analyze financial time series, which are a sequence of prices of financial entities, like stocks, observed over a time period at a stock market. Information from other stock markets is used help in the prediction process. An HMM is used to predict the S&P 500 Index. The general EM algorithm is modified to an exponentially weighted EM algorithm to give more emphasis to current data. The HMM developed performs better than the top mutual funds and neural networks. In our work we use information from surrounding sites to help in the prediction process of values at a site.

Chapter 6

Future Work

In this thesis, we build predictive models using both machine learning (ML) algorithms and Hidden Markov Models (HMM). These models are used to predict weather conditions and compare them with actual values to detect RWIS sensor malfunctions. In this chapter we discuss some possible improvements to the models and the general approach to enhance our work. Some obvious areas for potential improvement would be to use different ML methods to build a predictive model using larger datasets, and including sites with micro-climates and the development of malfunction models.

Many different algorithmic approaches are present in the field of machine learning. The use of other algorithms, such as Kalman filters, for weather data modeling could be explored. Kalman filters [Harvey, 1989] are used to estimate the state of a dynamic system from the data provided about the system. They are in some sense regression version of HMMs. The hidden state variables in Kalman filters are continuous, making the state sequence a sequence of numbers or a vector of real numbers. Kalman filters use linear operators with added noise to generate hidden states and outputs. they deal with linear systems and can also be extended to non-linear problems. Weather variables such as temperature can be used directly on Kalman filters (temperature was discretized for use in HMMs) as the state variables are continuous. Thus, a sequences of daily temperatures obtained for a duration of time can be used to build predictive models by Kalman filters.

In this work, we use data collected for one or two years for training and testing of the models. The performance of the model may be improved by using a larger data set spanning a larger duration of time. The number of features in the dataset for ML

algorithms was limited to the present and previous three hour's temperature values along other variables like temperature offset, precipitation type and visibility. More features can be added to the feature vector by selecting the features that most affect a particular variable. In HMMs, the class string contained the value of a variable from the site used for predictions and its nearby sites. Other variable information can also be added to the class string. To prevent the length of the class string getting to be too long in such cases, which will lead to large number of symbols and very noisy probability estimates, information from two or more variables can be combined to form a new class. In this work the models built include information from data collected all round the year. The yearly data can be split into different climatic periods, such as winter, spring, summer and fall and build models for the respective period.

In this work we focus on 13 RWIS sites for predictions. The work can be extended to all the RWIS sites present in Minnesota. Models for sensors at sites with significant micro-climates, that we not included in the selected sites, can be built. Weather at sites with micro-climates tend to have a different pattern from its surrounding sites, in order to build models for such sensors more focus should be given on the historical and current information collected from the site and less on the surrounding sites. Additional information about the weather conditions, such as wind, air pressure and cloud cover, could help in determining the weather patterns followed at such sites. Information from weather advisories could be used to identify the weather condition at the location, and can also be used to match it with the current weather conditions reported by the sensors to identify sensor accuracy.

The RWIS data available to us did not include much information on days where re-calibrations or maintenance work was done. The maintenance records were made manually and did not provide much computer-understandable data. Information about malfunctions and their effect on the readings recorded by the sensor was not available. To deal with this issue, malfunction models could be generated by adding additional physical sensors to an existing sensor. The additional sensor can be then tampered with or altered

to simulate conditions of mis-calibrations and malfunctions. The data from the two sensors, one with induced malfunctions and one recording actual conditions, can be compared and the resulting differences used to build malfunction models. These models could give us a range for potential errors for that sensor to identify malfunction or mis-calibrations in it.

Miscalibrations in the sensors that lead to a small scale deviations from the actual values and is seen for a long duration of time. Such slow drifts in the recorded readings are difficult to identify. New models could be built or the existing enhanced to identify such drifts in the readings, by examining for instance the long term historical differences in sensor values between sites and calculating the likelihood of the current history for a sensor.

Chapter 7

Conclusions

In this research we attempt to detect RWIS sensor malfunctions using real time sensor data. Malfunctions are identified as significant deviations in values reported by the sensor from the actual conditions present at the site. We use machine learning (ML) methods to build models for an RWIS site which are used to predict a value at that site. The predicted value is then compared to the actual value from the site to detect sensor malfunctions. To build models for RWIS sites, ML methods use historical weather data obtained from a representative sample of RWIS and AWOS sites.

We build models for RWIS sites to predict temperature, precipitation type and visibility which were identified as critical aspects of weather data for Mn/DOT. We use a variety of ML methods such as classification methods (e.g., J48 decision tree, Naive Bayes and Bayesian Networks), regression methods (e.g., Linear Regression, Least Median Square, M5P, MultiLayer Perceptron, RBF Networks and Conjunctive Rule) and Hidden Markov Models (HMMs) to predict this data. The effectiveness and accuracy of these methods in predicting this data was analyzed and their ability to detect sensor malfunctions identified.

From the results obtained for the ML methods applied on different representations of the data to predict temperate at an RWIS site we see that Conjunctive Rule and RBF Network fail completely with high errors for predicting temperature. It may be that these algorithms could have performed better if we had significantly tuned the parameters for these algorithms. The prediction of temperature by Linear Regression, Least Median Squares, M5P and Multilayer Perceptron is accurate to ±1°F. Models built by M5P and Least Median Square are used to detect sensor malfunctions because of their low standard

deviation across different sites.

Including precipitation type as an additional source of information for predicting temperature has no significant effect and the results are comparable to the experiments done without using precipitation type. The prediction of temperature class value is the best when J48 decision trees are used, which correctly classifies almost all instances. A threshold distance of 2 is identified to detect malfunctions when J48 is applied to predict temperature class values.

For detecting sensor malfunctions when predicting precipitation type, a combination of results from J48 decision trees and Bayesian Networks are used as J48 identified instances with no precipitation accurately and Bayesian Networks has accuracy for predicting the presence of precipitation. The model built using M5P shows a reasonable accuracy for predicting visibility.

Hidden Markov Models (HMMs) perform well for classifying discretized temperature values. The accuracy for models built at different sites varies with a considerable amount. A threshold distance of 3 between the actual and the predicted temperature class value is used to detect malfunctions. But in most cases on temperature sensors, the HMM models performed very well. Combining the data from different sites that belong to the same group and predicting the temperature class values without site information gives similar results as when using a dataset for a respective site to predict its temperature class value. We find the model built by HMM for precipitation type have a high error when classifying the presence or absence of precipitation. It was concluded not to use the HMM model for detecting sensor malfunctions.

We believe that the models built using selective ML methods for predicting temperature, predicting type and visibility can be used effectively for detecting RWIS sensor malfunctions and can be extended to other RWIS sites that were not included in the experiments performed.

Bibliography

[Akaike, 1974] Akaike, H., *A new look at the statistical model identification*. IEEE Transaction on Automatic Control , vol. AC-19, pp. 716-723, 1974.

[Allen & Greiner, 2000] Allen, T. and Greiner, R., *A model selection criteria for learning belief nets: An empirical comparison*, Proceedings of the International Conference on Machine Learning, pp. 1047-1054, 2000.

[AllWeatherInc] All Weather Inc., *Automated Weather Observing System (AWOS): International Technical Description,* http://www.allweatherinc.com/pdf/int_awos.pdf.

[Aurora] Aurora Program, *About RWIS,*
http://www.aurora-program.org/what_is_rwis.cfm.

[Baum & Petrie, 1966] Baum, L. and Petrie, T., *Statistical inference for probabilistic functions of finite state markov chains*, Annals of Mathematical Statistics, vol. 37, 1966.

[Bellone et al., 2000] Bellone, E., Huges, J. and Guttorp, P., *A hidden Markov model for downscaling synoptic atmospheric patterns to precipitation amounts*, Climate Research, vol. 15, pp. 1 – 12, 2000.

[Bishop, 1995] Bishop, C., *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[Boselly & Ernst, 1993] Boselly, S., and Ernst, D., *Road Weather Information Systems, Volume 2: Implementation Guide*, Report SHRP-H-351, Strategic Highway Research Program, National Research Council, Washington, DC, 1993.

[Boselly et al., 1993] Boselly, S., Thornes, J., Ulberg, C., and Ernst, D., *Road Weather Information Systems, Volume 1: Research Report*, Report SHRP-H-350, Strategic Highway Research Program, National Research Council, Washington, DC, 1993.

[Buhmann & Albovitz, 2003] Buhmann, M., Albowitz, M., *Radial Basis Functions: Theory and Implementations*, Cambridge University Press, 2003.

[Cano et al., 2004] Cano, R., Sordo, C. and Gutierrez, J., *Applications of Bayes nets in meteorology*, Advances in Bayesian networks, pp. 309 – 327, Springer 2004.

[Cooper & Herskovits, 1992] Cooper, G. and Herskovits, E., *A Bayesian Method for the Induction of Probabilistic Networks from Data*, Machine Learning, vol. 9, pp. 309-347, 1992

[Dietterich, 2002] Dietterich, T., *Machine learning for sequential data: A review*, Lecture Notes in Computer Science, vol. 2396, pp. 15-30, 2002.

[Dougherty et al., 1995] Dougherty, J., Kohavi, R. and Sahami, M., *Supervised and unsupervised discretization of continuous features*, Proceedings of the Twelfth International Conference on Machine Learning, pp 94-202, 1995.

[Durbin et al., 1989] Durbin, R., Eddy, S., Krogh, A. and Mitchison, G., *Biological Sequence Analysis: Probabilistic models of proteins and nucleic acids*, Cambridge University Press, 1998

[FAA, 2003] Federal Aviation Administration (FAA), *Automated Surface Observing System (ASOS) / Autoamted Weather Observing System (AWOS)*, updated Feb 2003, http://www.faa.gov/asos/

[FAA, 2006] Federal Aviation Administration (FAA), *Mechanism Data Report: Automated Weather Observing System*, updated April 2006, http://www.nas-architecture.faa.gov/nas5/mechanism/mech_data.cfm?mid=37.

[Fayyad & Irani, 1993] Fayyad, U. and Irani, K., *Multi-interval discretization of continuous-valued attributes for classification learning*, Proceedings of 13th International Joint Conference on Artificial Intelligence, pp 1022-1027, Morgan Kaufmann, 1993.

[FHA, 2004] *Collaborative Research on Road Weather Observations and Predictions by Universities, State Departments of Transportations, and National Weather Service Forecast Offices*, US. Department of Transportation Federal Highway Administration, Publication No. FHWA-HRT-04-109, October, 2004.

[Forney, 1973] Forney, J., *The Viterbi algorithm*, Proceedings of the IEEE, vol. 61, no. 3, pp. 268–278, March 1973

[Forsyth & Rada, 1986] Forsyth, R. and Rada, R., *Machine Learning applications in expert systems and information retrieval*, Ellis Horwood Ltd., 1986.

[Friedman et al., 1997] Friedman, N., Geiger, D. and Goldszmidt, M., *Bayesian network classifiers*, Machine Learning, vol. 29, pp. 131-163, 1997.

[Gallus et al., 2004] Gallus, W. Jr, Jungbluth, K. and Burkheimer, D., *Improved Frost Forecasting through Coupled Artificial Neural Network Time-Series Prediction Techniques and a Frost Deposition Model*, US. Department of Transportation Federal Highway Administration, Publication No. FHWA-HRT-04-109, pp. 19 – 26, October, 2004.

[Good, 1965] Good, I., *The Estimation of Probabilities: An Essay on Modern Bayesian Methods*, M.I.T. Press, 1992.

[Hall et al., 1999] Hall, T., Brooks, H. and Doswell, C., *Precipitation forecasting using a Neural Network*, Weather and Forecasting, vol. 14, num. 3, pp. 338-345, 1999.

[Harvey, 1989] Harvey, A., *Forecasting, Structural Time Series Models and the Kalman Filter*, Cambridge University Press, Cambridge, 1989.

[Heckerman et al., 1995] Heckerman, D., Geiger, D. and Chickering, D., *Learning Bayesian networks: The combination of knowledge and statistical data*, Machine Learning, vol. 2, pp. 197-243, 1995.

[Holland, 1962] Holland, J., *Outline for a logical theory of adaptive systems*, Journal for Association of Computing Machinery, vol. 3, pp. 297-314, 1962.

[Knight et al., 2004] Knight, P., Ayers, B., Ondrejik, D. and Uzowke, A., *Developing an Interactive Mesonet for PennDOT*, US. Department of Transportation Federal Highway Administration, Publication No. FHWA-HRT-04-109, pp. 10 – 17, October, 2004.

[Kohavi, 1995] Kohavi, R., *A study of cross-validation and bootstrap for accuracy estimation and model selection*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995.

[Langley et al., 1992] Langley, P., Iba, W. and Thompson, K., *An Analysis of Bayesian Classifiers*, Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 223-228, AAAI Press, 1992.

[Langley, 1996] Langley, P., *Elements of Machine Learning*, Morgan Kaufmann, San Francisco, 1996.

[Lau &Weng, 1995] Lau, K. and Weng, H., *Climate Signal Detection Using Wavelet Transform: How to Make a Time Series Sing*, Bulletin of the American Meteorological Society: Vol. 76, No. 12, pp. 2391- 2402, 1995.

[Littlestone, 1988] Littlestone, N., *Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm*, Machine Learning, vol. 2, pp. 285-318, 1988.

[Manfredi et al., 2005] Manfredi, J., Walters, T., Wilke, G., Osborne, L., Hart, R., Incrocci, T. and Schmitt, T., *Road Weather Information System Environmental Sensor Station Siting Guidelines*, US. Department of Transportation Federal Highway Administration, Publication No. FHWA-HOP-05-026, April 2005.

[McMillan et al., 2005] McMillan, N., Bortnick, S., Irwin, M. and Berliner, M., *A hierarchical Bayesian model to estimate and forecast ozone through space and time*, Atmosphereic Environment, vol. 39, pp. 1373 -1382, 2005.

[Mitchell, 1997] Mitchell, T., *Machine Learning*, McGraw Hill, 1997

[Murphy, 1998] Murphy, K., *Hidden Markov Model (HMM) Toolbox for MATLAB*, 1998, http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html.

[NCDC, 2005] National Climatic Data Center, *Data Documentation for Data Set 3282 (DSI-3282): ASOS Surface Airways Hourly Observations*, National Climatic Data Center, Asheville, NC, May 2005.

[Nilsson, 1996] Nilsson, N., *Introduction to Machine Learning*. Unpublished draft, Department of Computer Science, Stanford University, 1996.

[Orr, 1996] Orr, M., *Introduction to radial basis function networks*. Technical report, Institute for Adaptive and Neural Computation of the Division of Informatics at Edinburgh University, Scotland, UK, 1996, http://www.anc.ed.ac.uk/~mjo/papers/intro.ps.gz.

[Park et al., 1991] Park, D., El-Sharkawi, M., Marks, R. II, Atlas, L. and Damborg, M., *Electric load forecasting using an artificial neural network*, IEEE Transactions on Power Systems, vol. 6, issue 2, pp. 442 – 449, May 1991

[Pearl, 1988] Pearl, J., *Probabilistic reasoning in intelligent systems*, Morgan Kaufman, 1988

[Quinlan, 1986] Quinlan, R., *Induction of Decision Trees,* Machine Learning, vol. 1, pp. 81-106, 1986.

[Quinlan, 1992] Quinlan, R., *Learning with Continuous Classes*, Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, pp. 343-348. World Scientific, Singapore, 1992.

[Quinlan, 1993] Quinlan, R., C4.5: *Programs for Machine Learning*, Morgan Kaufmann, San Francisco, 1993.

[Rabiner & Juang, 1986] Rabiner, L. and Juang, B., *An Introduction to Hidden Markov Models*, IEEE ASSP Magazine, pp. 4-15 , January 1986.

[Rabiner, 1989] Rabiner, L., *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, Proceedings of IEEE, Vol. 77, Number 2, February 1989.

[Rissanen, 1978] Rissanen, J., *Modeling by shortest data description*. Automatica, vol. 14, pp. 465-471, 1978.

[Rousseeuw, 1984] Rousseeuw, P., *Least Median Squares of Regression*, Journal of American Statistical Association, vol. 49, pp. 871-880, December 1984.

[Rumelhart et al., 1986] Rumelhart, D., Hinton, G. and Williams, R., *Learning internal representations by error propagation*, Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol.I, pp. 318–362, MIT Press, 1986.

[Russell et al., 1995] Russell, S., Binder, J., Koller, D. and Kanazawa, K., *Local Learning in Probabilistic Networks with Hidden Variables*, Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1995.

[Schizas, 1991] Schizas, C., Michaelides, S., Pattichis, C. and Livesay, R., *Artificial neural networks in forecasting minimum temperature*, Second International Conference on Artificial Neural Networks, pp. 112 -114, November 1991.

[Todey et. al., 2002] Todey, D., Herzmann, D., Gallus, Jr. W., and Temeyer, B., *An intercomparison of RWIS data with AWOS and ASOS observations in the state of Iowa*, The Third Symposium on Environmental Applications:Facilitating the Use of Environmental Information, American Meteorological Society, January, 2002.

[Viterbi, 1967] Viterbi, A., *Error bounds for convolutional codes and asymptotically optimum decoding algorithm*, IEEE Transactions on Information Theory, vol. IT-13, no. 2, pp. 260–269, April 1967.

[Wang & Witten, 1997] Wang, Y. and Witten, I., *Inducing Model Trees for Continuous Classes*, In Poster Papers of the Ninth European Conference on Machine Learning, pp. 128-137, Prague, Czech Republic, April, 1997.

[Witten & Frank, 2005] Witten, I. and Frank, E., *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[Zhang, 2004] Zhang, Y., *Prediction of Financial Time Series with Hidden Markov Models*, Masters Thesis, Simon Fraser University, 2004.

Appendix A

RWIS and AWOS Site Locations

Table A.1: Latitude and longitude coordinates for RWIS sites in Minnesota.

SITE ID	LON	LAT	SITE ID	LON	LAT	SITE ID	LON	LAT
1	-92.354	43.508	28	-96.320	44.270	56	-95.731	48.434
3	-93.292	44.478	29	-92.708	44.022	57	-97.202	48.971
4	-92.993	45.643	30	-96.435	44.459	58	-96.732	47.296
5	-92.839	46.213	31	-94.995	44.545	59	-96.442	44.936
6	-96.378	43.608	32	-94.414	44.543	60	-96.917	47.881
7	-95.770	43.637	33	-95.614	44.692	61	-94.480	47.371
8	-95.119	43.639	34	-94.021	45.616	62	-96.111	46.877
9	-94.122	43.663	35	-94.104	46.144	63	-95.054	46.413
10	-92.681	43.727	36	-96.494	45.558	64	-94.233	45.767
11	-92.205	43.956	37	-95.562	45.488	65	-91.885	43.515
12	-91.544	43.912	38	-95.597	44.057	66	-92.488	47.211
13	-96.668	46.847	40	-91.437	43.719	67	-92.970	48.294
14	-96.291	46.491	41	-94.127	47.604	68	-96.001	47.770
15	-95.366	45.838	42	-92.858	44.409	69	-91.990	44.285
16	-94.931	45.713	43	-94.932	45.432	70	-92.482	43.902
17	-94.026	45.395	44	-95.934	47.274	71	-95.120	44.084
18	-94.432	47.874	45	-94.368	44.076	72	-93.878	48.135
19	-92.048	47.854	47	-92.698	44.601	73	-94.034	44.154
20	-93.954	46.688	48	-89.685	47.976	74	-93.963	44.366
21	-95.410	44.949	49	-94.723	46.912	75	-93.843	44.572
22	-93.986	44.906	50	-93.262	46.340	76	-93.664	45.876
23	-92.676	45.385	51	-93.274	46.978	77	-92.948	45.985
24	-96.915	48.572	52	-93.485	47.840	78	-95.051	48.789
25	-95.961	48.820	53	-94.554	48.207	79	-93.625	44.544
26	-94.067	48.628	54	-96.494	46.043			
27	-93.373	48.602	55	-95.602	46.404			

SITEID: Site number for a location with RWIS senor

LON: Longitude

LAT: Latitude

Table A.2: Latitude and Longitude coordinates for AWOS sites in Minnesota.

SITE	LON	LAT	SITE	LON	LAT	SITE	LON	LAT
K8D3	-97.006	45.672	KFFM	-96.157	46.284	KMVE	-95.71	44.969
KACQ	-93.555	44.077	KFGN	-94.901	49.321	KMWM	-95.109	43.913
KADC	-95.211	46.45	KFKA	-92.189	43.683	KMZH	-92.805	46.419
KAEL	-93.367	43.682	KFOZ	-93.65	47.783	KOEO	-92.694	45.314
KAIT	-93.677	46.548	KFRM	-94.416	43.644	KONA	-91.708	44.077
KANE	-93.211	45.145	KFSD	-96.747	43.592	KORB	-92.856	48.016
KAQP	-96.004	45.228	KFSE	-95.773	47.593	KOTG	-95.579	43.655
KATY	-97.158	44.922	KGDB	-95.561	44.756	KOVL	-95.033	44.779
KAUM	-92.933	43.665	KGFK	-97.183	47.95	KOWA	-93.261	44.123
KAXN	-95.395	45.866	KGHW	-95.32	45.644	KPKD	-95.073	46.901
KBBB	-95.651	45.332	KGNA	-90.341	47.751	KPNM	-93.608	45.56
KBDE	-94.612	48.728	KGPZ	-93.51	47.211	KPQN	-96.3	43.983
KBFW	-91.416	47.249	KGYL	-94.081	44.76	KPWC	-94.382	46.725
KBJI	-94.934	47.509	KGYL	-94.092	44.761	KRGK	-92.485	44.589
KBKX	-96.817	44.308	KHCD	-94.382	44.859	KROS	-92.953	45.698
KBRD	-94.137	46.398	KHCO	-96.943	48.753	KROX	-95.697	48.856
KCBG	-93.265	45.56	KHIB	-92.839	47.387	KRPD	-91.778	45.419
KCDD	-92.481	48.271	KHYR	-91.453	46.031	KRRT	-95.348	48.941
KCFE	-93.85	45.164	KHZX	-93.317	46.619	KRST	-92.499	43.907
KCHU	-91.504	43.596	KILL	-95.089	45.116	KRWF	-95.082	44.547
KCKC	-90.383	47.838	KINL	-93.403	48.566	KSAZ	-94.807	46.381
KCKN	-96.622	47.842	KJKJ	-96.663	46.839	KSBU	-94.093	43.595
KCOQ	-92.506	46.7	KJMR	-93.272	45.886	KSGS	-93.033	44.857
KCQM	-92.689	47.822	KJYG	-94.558	43.986	KSTC	-94.06	45.547
KDLH	-92.194	46.842	KLJF	-94.507	45.097	KSTP	-93.06	44.934
KDTL	-95.886	46.825	KLSE	-91.261	43.886	KSUW	-92.103	46.694
KDXX	-96.178	44.986	KLVN	-93.228	44.628	KTKC	-95.611	44.258
KDYT	-92.043	46.722	KLXL	-94.347	45.949	KTOB	-92.832	44.018
KEAU	-91.486	44.875	KMGG	-93.986	45.236	KTVF	-96.183	48.066
KELO	-91.831	47.825	KMIC	-93.354	45.062	KTWM	-91.745	47.049
KETH	-96.544	45.78	KMJQ	-94.987	43.65	KULM	-94.502	44.32
KEVM	-92.498	47.425	KMKT	-93.919	44.222	KVVV	-96.424	45.306
KFAR	-96.825	46.922	KMML	-95.822	44.45	KVWU	-94.517	48.154
KFBL	-93.311	44.325	KMOX	-95.968	45.567	KXVG	-94.204	46.99
FCM	-93.457	44.827	KMSP	-93.217	44.881			

SITE: Airport code where the AWOS unit is located

LON: Longitude

LAT: Latitude

Appendix B

Using WEKA

WEKA is written in Java and is organized into packages arranged in a hierarchical manner. Details of the packages and the hierarchy are given by Witten & Frank [2005]. WEKA can be run using its graphical user interface or through entering textual commands in the command prompt. The general structure of the WEKA textual command, to perform multiple 10-fold cross-validations on a dataset using an algorithm (classifier) is

```
java -mx1024M -cp classpath callClassifier classifier_path
classifier_options -t trainset.arff -x 10 -s seed_value -c
attribute_index
```

where -cp specifies the path (i.e., the class path) where WEKA is located, callClassifier3 is a java class that is used to output the complete class probability without which WEKA outputs an evaluative result of the algorithm, *classifier_path* is the location of the algorithm in the WEKA package hierarchy, *classifier_options* specifies the options taken by an algorithm, -t specifies the training file, -x specifies the number of folds for cross-validation, -s is used to indicated the seed value when a multiple n-fold cross-validations need to be preformed, -c specifies the output attributes position in the dataset provided. The -T option is used when a test file is used for evaluating the model, when not used a cross validation is preformed on the training set provided.

WEKA requires the data in the train/test file to be in ARFF format. The general format of an ARFF file is given in Table B1. The string @relation is used to mention the name of the dataset, @attribute is used to define the attributes name and type and @data is used to

3. http://alex.seewald.at/WEKA/callClassifier.java

Table B.1: Format of an ARFF file.

```
@relation Predict_Temp_Year2002

@attribute temperature_site1 real
@attribute temperature_site2 real
@attribute precipitation       (yes, no)

% used for comments

@data
23,22,yes
12,23,no
23,32,no
.............
```

indicate the start of the data, which is in a comma-separated form.

Following are the *classifier_path* for the machine learning algorithms that were used in this thesis along with their default options (*classifier_options*)

Linear Regression

```
weka.classifiers.functions.LinearRegression -S 0 -R 1.0E-8
```

where -*S* specifies the attribute selection methods with 0 representing the M5 method, and -*R* specifies the value of the ridge parameter.

Least Median Square

```
weka.classifiers.functions.LeastMedSq –S 4 –G 0
```

where -S specifies the size of random samples used to generate the least squared regression function, and -G specifies the seed value used to select subsets of the training data.

M5Prime

```
weka.classifiers.trees.M5P –M 4.0
```

where -*M* specifies the minimum number of instances.

Multilayer Perceptron

```
weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M
0.2 -N 500 -V 0 -S 0 -E 20 -H a
```

where -*L* specifies the learning rate, -M specifies the momentum, -*N* specifies the number of training epochs, -*V* specifies validation set size, -*S* specifies the seed value taken by the random number generator (random values are used for initialization of weights), -*E* specifies the validation threshold, and -*H* specifies the number of hidden layers with its value 'a' representing *(num_attributes+num_classes)/2* layers.

RBF Network

```
weka.classifiers.functions.RBFNetwork -B 2 -S 1 -R 1.0E-8 -
M -1 -W 0.1
```

where -*B* specifies the number of clusters generated by K-means, -*S* specifies the value of the seed passed on to the K-means, -*R* specifies the value of the ridge parameter, -*M* specifies the number of iterations to be performed by logistic regression, and -*W* specifies the minimum standard deviation for the clusters.

Conjunctive Rule

```
weka.classifiers.rules.ConjunctiveRule -N 3 -M 2.0 -P -1
-S 1
```

where -*N* specifies the amount of data used for pruning, -*M* specifies the minimum total weight of the instances in a rule, -*P* specifies the minimum number of antecedents allowed in a rule when pre-pruning is used, and -*S* specifies the seed value used.

J48

```
weka.classifiers.trees.J48 -C 0.25 -M 2
```

where -*C* specifies the confidence factor, and -*M* specifies the minimum number of instances taken by a leaf

Naive Bayes

```
weka.classifiers.bayes.NaiveBayes
```

Bayes Net

```
weka.classifiers.bayes.BayesNet -D
-Q weka.classifiers.bayes.net.search.local.K2 -- -P 1
-E weka.classifiers.bayes.net.estimate.SimpleEstimator -- -
A 0.5
```

-*D* is used to prevent memory problems with ADTree is used, -*Q* specifies the search algorithm, and -*E* specifies the estimator used for finding the CPTs. K2 search algorithm is given by weka.classifiers.bayes.net.search.local.K2 with its option -*P* specifying the maximum number of parents taken by a node in the Bayesian network. The estimator used for filling up the CPTs is weka.classifiers.bayes.net.estimate.SimpleEstimator, with its option -*A* specifying the alpha value of the estimator.

Appendix C

Detailed Results

Table C.1: Results obtained from using regression algorithms to predict temperature at an RWIS site (Experiment 1). Feature vector consists of temperature information from RWIS-AWIS sites in a set along with temperature offset for the RWIS sites. The table has mean absolute error values averaged over ten 10-fold cross-validations.

		ML Algorithms					
RWIS Site		**LMS**	**LR**	**M5P**	**RBF**	**CR**	**MLP**
Set 1	19	0.908	0.960	0.936	9.521	11.150	1.059
	27	1.217	0.896	0.873	9.465	10.199	1.058
	67	1.069	0.918	0.885	10.062	11.596	1.108
Set 2	14	0.659	0.795	0.751	8.478	10.605	0.789
	20	0.743	0.820	0.776	9.417	10.821	1.001
	35	0.553	0.977	0.864	9.523	10.817	1.051
	49	0.916	0.913	0.898	9.579	11.017	1.074
	62	0.800	0.779	0.769	9.383	11.040	0.892
Set 3	25	0.984	1.062	0.889	10.386	11.957	1.097
	56	0.925	0.913	0.807	10.510	11.512	1.133
	60	0.889	0.867	0.833	9.675	11.078	1.002
	68	0.958	1.015	0.901	9.017	10.439	1.235
	78	0.929	0.875	0.809	8.945	10.449	1.012
Mean of Abs. Errors (ºF)		0.888	0.907	0.845	9.535	10.975	1.039
StdDev of Abs. Errors		0.171	0.083	0.058	0.559	0.503	0.110

StdDev refers to Standard Deviation

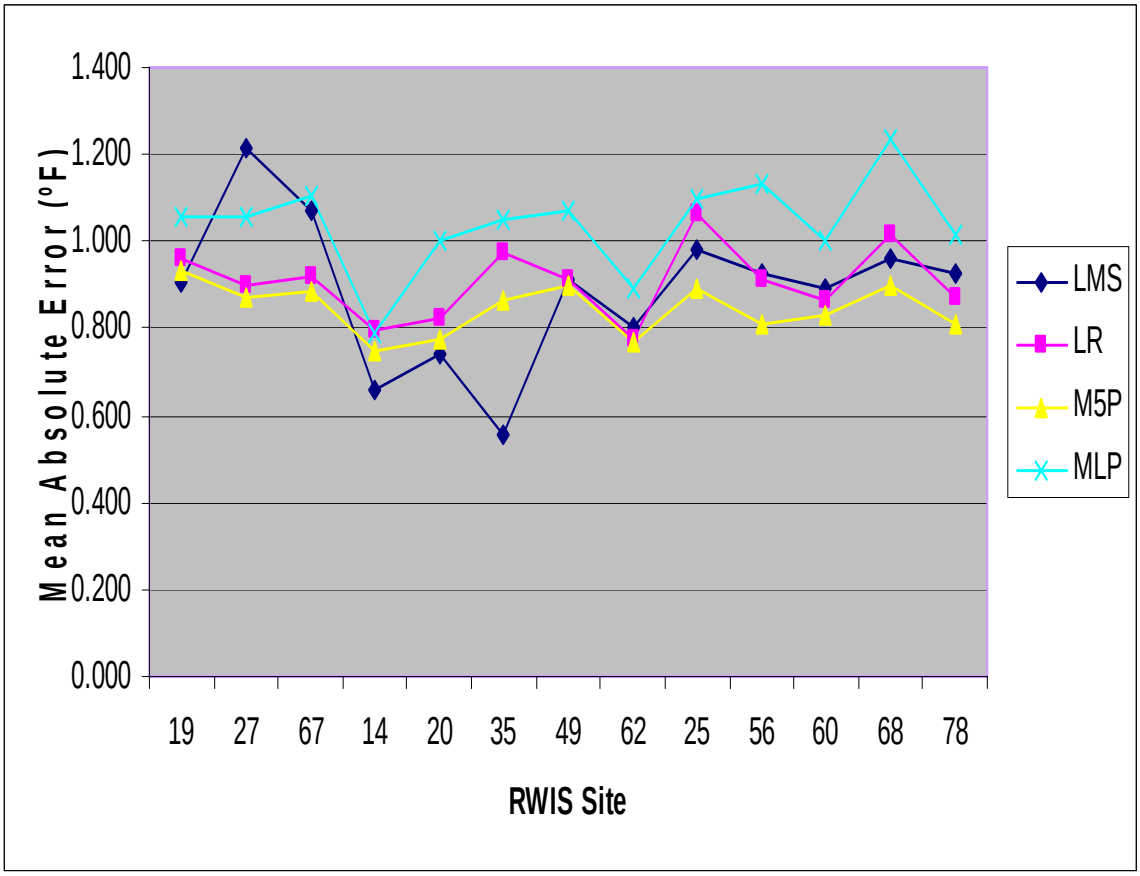

Figure C.1: Mean absolute errors for different RWIS sites obtained from predicting temperature using regression algorithms.

Table C.2: Results obtained from using regression algorithms to predict temperature at an RWIS site (Experiment 2). Feature vector consists of temperature information from RWIS-AWIS sites in a set along with precipitation type for the RWIS sites. The table has mean absolute error values averaged over ten 10-fold cross-validations.

		ML Algorithms		
	RWIS Site	**LMS**	**LR**	**M5P**
Set 1	19	1.023	1.115	1.001
	27	0.935	0.959	0.931
	67	1.001	1.006	0.984
Set 2	14	0.726	0.788	0.771
	20	0.862	0.872	0.827
	35	1.052	1.062	0.938
	49	1.051	1.014	1.022
	62	0.848	0.827	0.815
Set 3	25	1.217	1.222	1.004
	56	1.084	1.077	0.992
	60	1.007	0.981	0.924
	68	1.046	1.154	0.973
	78	0.876	0.891	0.856
Mean of Abs Errors (ºF)		0.979	0.997	0.926
StdDev of Abs Errors		0.127	0.130	0.083

StdDev refers to standard deviation

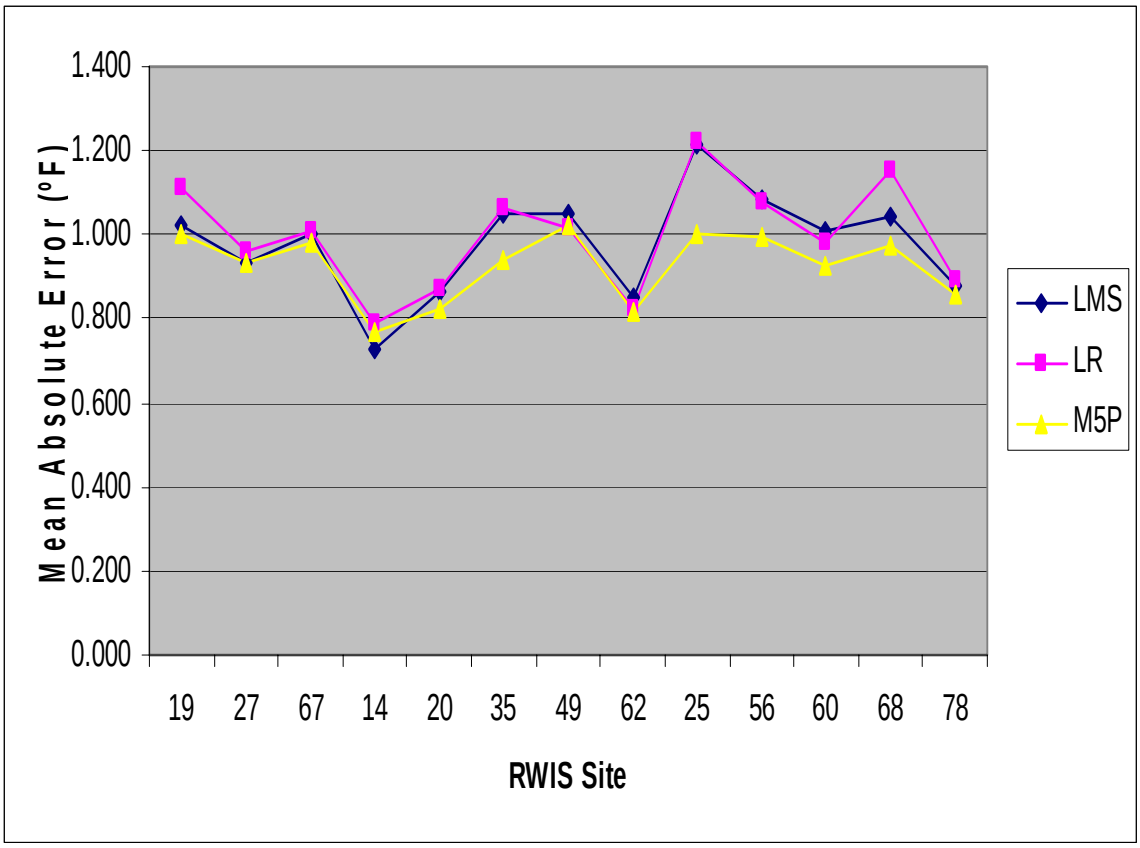

Figure C.2: Mean absolute errors for different RWIS sites obtained from predicting temperature using regression algorithms, with precipitation type information added to the feature vector.

Table C.3: Results obtained from using classification algorithms to predict precipitation type at an RWIS site (Experiment 4). Feature vector consists of temperature information from RWIS-AWIS sites in a set along with precipitation type for the RWIS sites. The table has classification error values, as reported by WEKA, averaged over ten 10-fold cross-validations.

		ML Algorithms		
	RWIS Site	**J48**	**NB**	**Bayes Net**
Set 1	19	0.064	0.325	0.346
	27	0.256	0.420	0.363
	67	0.356	0.472	0.414
Set 2	14	0.213	0.384	0.330
	20	0.265	0.450	0.379
	35	0.077	0.312	0.337
	49	0.062	0.328	0.328
	62	0.061	0.312	0.341
Set 3	25	0.068	0.342	0.342
	56	0.072	0.345	0.333
	60	0.095	0.317	0.323
	68	0.061	0.253	0.315
	78	0.065	0.318	0.231
Mean of Classification Errors		0.132	0.352	0.337
StdDev of Classification Errors		0.102	0.062	0.041

StdDev refers to Standard Deviation

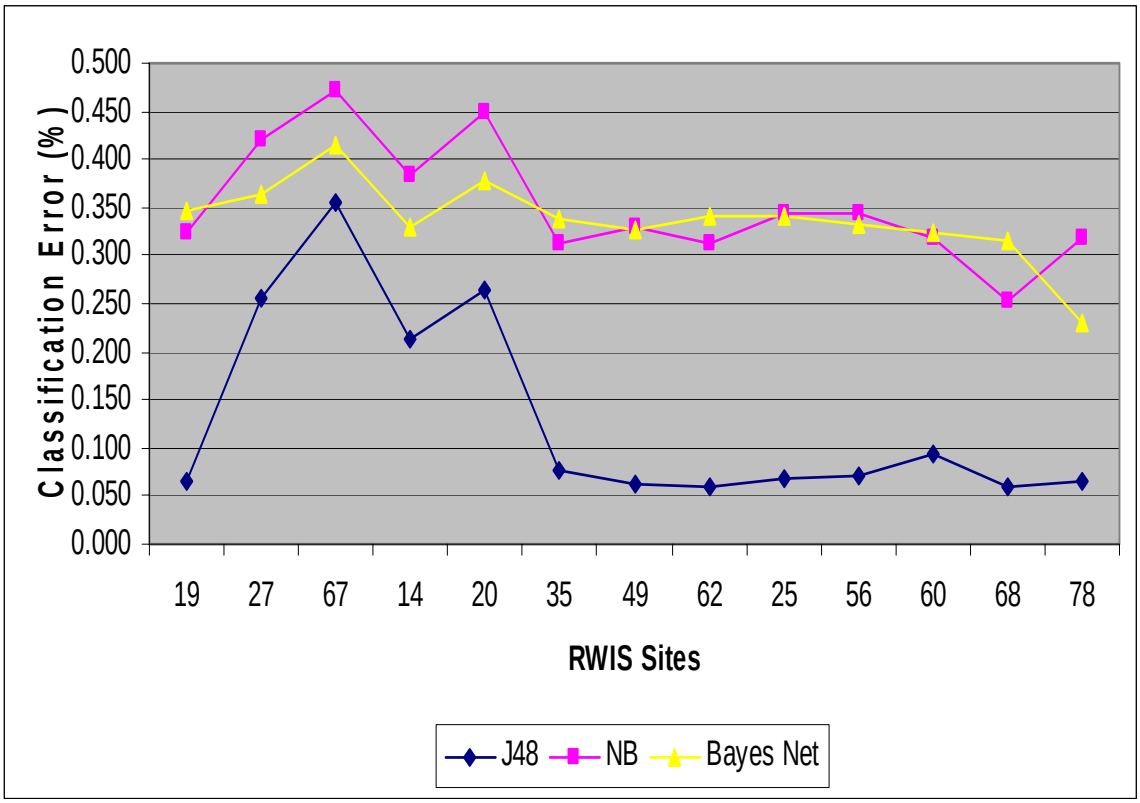

Figure C.3: Classification errors for different RWIS sites obtained from predicting precipitation using classification algorithms.

Table C.4: Percentage of instances predicted correctly using classification algorithms (Experiment 4)

	Actual Data		J48			
RWIS Site	**NP**	**P**	**NP->NPP**	**P->PP**	**P->NPP**	**NP->PP**
19	86.43	13.57	81.99	5.59	4.44	7.98
27	76.29	23.71	68.78	13.31	7.51	10.40
67	63.65	36.35	50.74	18.55	12.91	17.79
25	84.92	15.08	80.49	6.61	4.43	8.47
56	86.09	13.91	81.31	4.73	4.78	9.18
60	79.59	20.41	71.95	8.82	7.63	11.59
68	87.74	12.26	85.31	5.29	2.43	6.97
78	83.33	16.67	78.30	9.46	5.03	7.21
14	78.48	21.52	73.66	10.33	4.82	11.19
20	70.49	29.51	60.79	15.77	9.70	13.74
35	84.15	15.85	78.15	6.22	6.00	9.63
49	86.41	13.59	82.47	5.90	3.94	7.69
62	85.87	14.13	82.03	6.99	3.84	7.13
Average	**81.02**	**18.98**	**75.06**	**9.05**	**5.96**	**9.92**
Std Dev	**7.23**	**7.23**	**9.97**	**4.36**	**2.84**	**3.12**

[table continued on the next page]

Std Dev: Standard Deviation

NP: No precipitation reported

P: Precipitation reported

NP->NPP: No precipitation predicted when No precipitation was reported

P->PP: Precipitation predicted when precipitation was reported

P->NPP: No precipitation predicted when precipitation was reported

NP->P: Precipitation predicted when precipitation was reported

Table C.4: Percentage of instances predicted correctly using classification algorithms (Experiment 4) [table continued from previous page]

	Naive Bayes				Bayes Nets			
RWIS Site	**NP->NPP**	**P->PP**	**P->NPP**	**NP->PP**	**NP->NPP**	**P->PP**	**P->NPP**	**NP->PP**
19	7.88	13.11	78.55	0.46	1.81	13.48	84.62	0.09
27	43.72	14.14	32.57	9.57	52.02	11.92	24.27	11.79
67	27.66	12.55	35.99	23.80	40.62	18.89	23.03	17.46
25	2.53	14.92	82.38	0.16	2.29	15.02	82.63	0.07
56	2.99	13.78	83.10	0.13	4.63	13.75	81.46	0.16
60	3.33	19.95	76.26	0.46	1.17	20.32	78.41	0.09
68	28.22	10.78	59.52	1.48	13.82	11.43	73.90	0.86
78	13.18	15.68	70.15	0.99	35.23	13.41	48.10	3.26
14	49.44	12.06	29.04	9.46	57.24	9.69	21.24	11.83
20	35.03	19.88	35.45	9.63	45.98	16.11	24.50	13.40
35	9.00	14.79	75.15	1.06	2.44	15.65	81.71	0.20
49	7.44	13.20	78.97	0.39	6.81	13.41	79.60	0.18
62	10.29	13.36	75.58	0.76	2.28	13.97	83.60	0.16
Average	**18.55**	**14.47**	**62.47**	**4.51**	**20.56**	**14.39**	**60.46**	**4.60**
Std Dev	**16.31**	**2.73**	**21.19**	**6.98**	**22.01**	**2.90**	**27.49**	**6.47**

Table C.5: Results obtained from using regression algorithms to predict visibility at an RWIS site (Experiment 5). Feature vector consists of temperature information from RWIS-AWIS sites in a set along with precipitation type and visibility for the RWIS sites. The table has mean absolute error values averaged over ten 10-fold cross-validations.

		ML Algorithms			
	RWIS Site	**LMS**	**LR**	**M5P**	**MLP**
Set 1	67	1.6519	1.7436	1.6584	1.8020
Set 2	35	0.0785	0.1204	0.1024	0.1838
	49	0.0491	0.0756	0.0613	0.0643
	62	0.0511	0.0785	0.0618	0.0716
Set 3	56	0.0456	0.0722	0.0573	0.0975
	68	0.0160	0.0296	0.0244	0.0478
	78	0.1791	0.2081	0.0845	0.1545

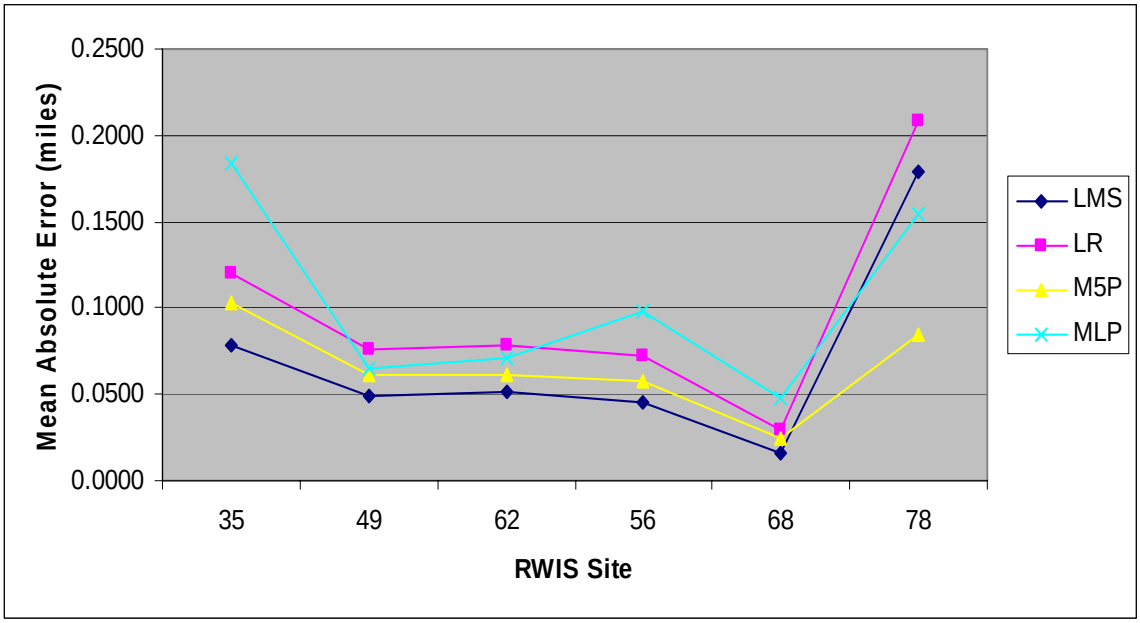

Figure C.4: Mean absolute errors for different RWIS sites obtained from predicting visibility using regression algorithms.

Table C.6: Percentage of instances with a certain distance between the actual and predicted temperature class value, obtained by using HMM to predict temperature class (Experiment 7).

	Set 1			Set 2				
	19	**27**	**67**	**14**	**20**	**35**	**49**	**62**
Distance 0	18.48%	16.18%	19.30%	21.94%	27.37%	22.86%	18.55%	22.37%
1	68.31%	28.67%	67.63%	73.11%	59.03%	65.06%	60.16%	70.88%
2	12.08%	49.85%	12.19%	4.83%	12.24%	11.09%	16.17%	4.38%
3	1.09%	4.59%	0.88%	0.11%	1.32%	0.95%	3.57%	1.15%
4	0.04%	0.71%	0.00%	0.00%	0.04%	0.04%	1.12%	0.53%
5	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.35%	0.67%
6	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.08%	0.00%
7	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
8	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%

	Set 3				
	25	**56**	**60**	**68**	**78**
Distance 0	42.65%	44.42%	39.60%	20.11%	39.69%
1	52.82%	49.00%	53.09%	27.20%	39.26%
2	4.33%	6.14%	6.96%	18.24%	16.86%
3	0.20%	0.42%	0.35%	8.36%	1.99%
4	0.01%	0.02%	0.00%	14.59%	0.51%
5	0.00%	0.00%	0.00%	7.17%	1.20%
6	0.00%	0.00%	0.00%	3.83%	0.49%
7	0.00%	0.00%	0.00%	0.46%	0.00%
8	0.00%	0.00%	0.00%	0.02%	0.00%

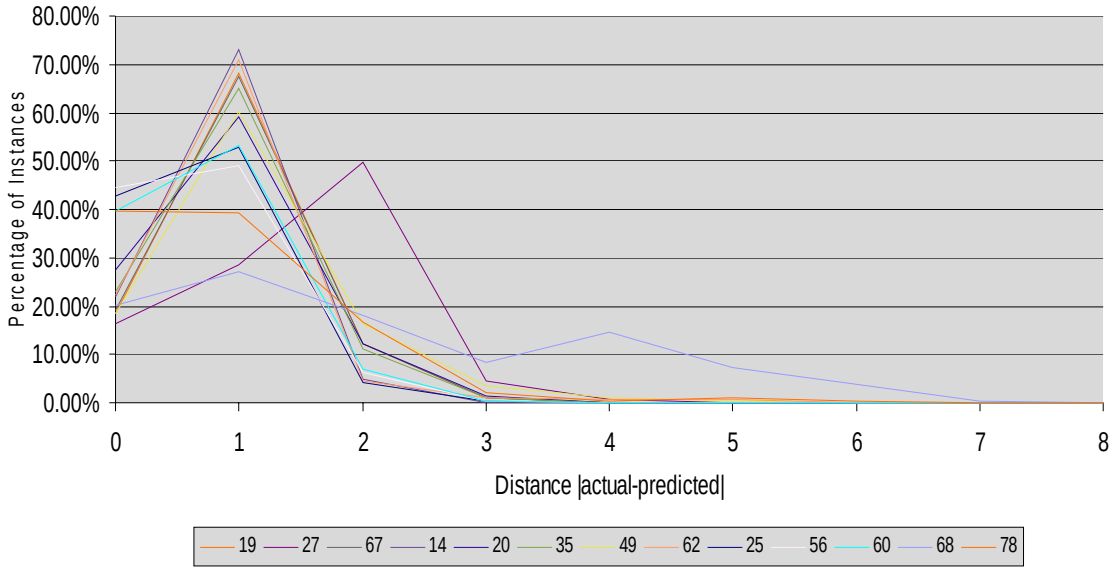

Figure C.5: Percentage of instances with a certain distance between the actual and predicted temperature class value, obtained by using HMM to predict temperature class

Table C.7: Percentage of instances with a certain distance between the actual and predicted temperature class value, obtained by using extended dataset focusing on predicting value for an RWIS set rather than for an RWIS site, and applying ten 10-fold cross validations using HMM to predict temperature class (Experiment 8).

	Set 1	Set 2	Set 3
Distance 0	17.03%	19.29%	34.54%
1	59.30%	67.05%	47.84%
2	20.16%	11.40%	12.89%
3	3.27%	1.97%	2.61%
4	0.24%	0.17%	0.95%
5	0.00%	0.13%	0.83%
6	0.00%	0.00%	0.29%
7	0.00%	0.00%	0.05%

Set 1: RWIS Sites 19, 27 and 67 **Set 2:** RWIS Sites 14, 20, 35, 49, 62

Set 3: RWIS Sites 25, 56, 60, 68, 78

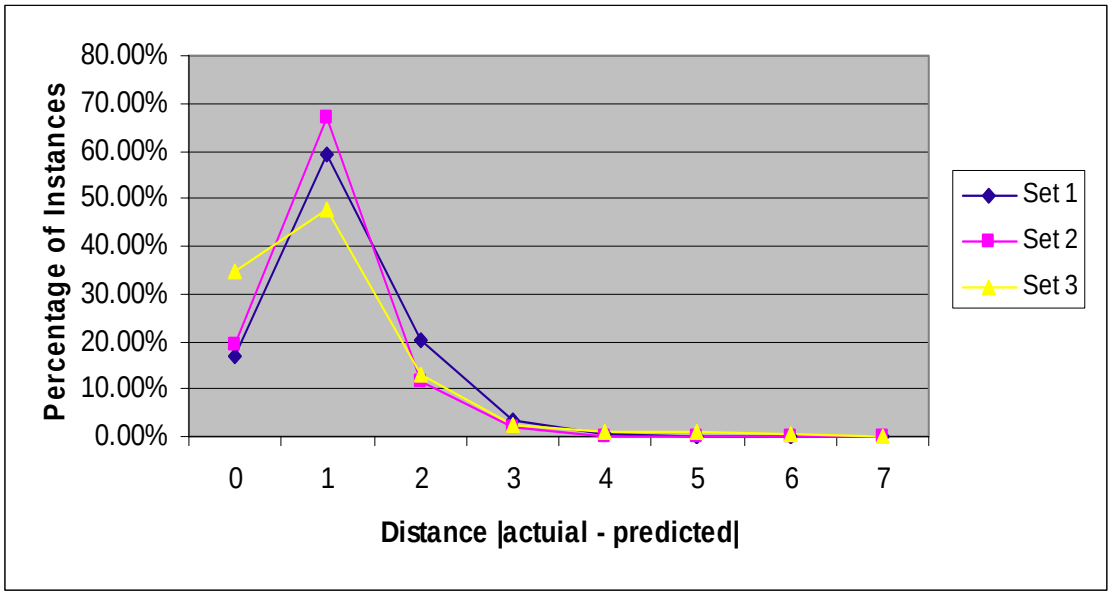

Figure C.6: Percentage of instances with a certain distance between the actual and predicted temperature class value, obtained by using HMM to predict temperature class with training done using the extended dataset.

Table C.8: Results obtained from predicting precipitation type using HMM (Experiment 9).

	RWIS Site	% of instance with no precipitation reported in data	% of instance with precipitation reported in data	% of instances where no precipitation is predicted correctly	% of instances where precipitation is predicted correctly
Set 1	**19**	85.71	14.28	55.22	4.79
	27	77.23	22.77	13.33	18.72
	67	67.3	32.69	5.36	28.35
Set 2	**14**	74.61	24.38	27.22	18.08
	20	72.74	27.25	16.96	21.43
	35	83.58	16.41	71.19	6.85
	49	84.66	15.33	71.24	7.34
Set 3	**62**	85.82	14.17	80.8	6.38
	25	83.1	16.9	73.4	7.7
	56	85.25	14.74	74.25	4.27
	60	78	20	53.64	9.39
	68	86.41	13.59	73.69	5.15
	78	83.89	16.11	55.91	7.73