**Time Series Data: Stock Data Prediction**

Undergraduate # 12

Team member 1 (Michael Capriles, 810855729, *mcaprile@kent.edu*)

Team member 2 (Adam Duryea, 810599988, *aduryea@kent.edu*)

Team member 3 (Shane Favours, student ID, *sfavours@kent.edu*)

Team member 4 (Samuel Pittman, 810666476, *spittma4@kent.edu*)

**1. Introduction**

* Motivation examples of this project
* Real applications

The motivations for our project came from the need to monitor and make predictions about ever increasing amounts of data. Time series data comes in a number of different flavors and formats from temperatures from weather sensors, oxygen and other element concentration levels in mines, traffic wait times in seconds, levels of a particular chemical in blood, and of course financial data from stocks or other sales markets. All these sources require a method of accurately and rapidly detecting patterns and making predictions based on previously observed results. Knowing these values would help us save lives via detection of diseases or prevent the buildup of harmful levels of gases within mines. IT also helps us save money by reducing traffic congestion time and prevent wasteful spending within markets or make smarter choices in investment options.

With the Stock Market always going up and down wouldn’t it be wonderful if there was a way to predict whether a company’s stock would go up and down in price. With time-series data, we can use past data to predict what the future of a company’s stock will do tomorrow. By studying the past data of a company, you can have an educated prediction to know whether you should buy or sell tomorrow. Time-series data can also be used to look at past weather patterns and predict what the future weather could possibly be.

This can be used in real applications for tracking and predicting what a company stock will do tomorrow. Using a company’s stock past data, we can look for patterns to predict whether or not a company stock will either rise or fall tomorrow. It could also be used with other data for predicting other things in life. In addition to this, any application that uses a time series data could be used to predict future data e.g. Network Monitoring, Monitoring Weather Temperature.

**2. Project Description**

* Brief descriptions of your project
* Challenges and technical contributions (new problems or new solutions?) in your project
* The workload distribution for each member in your team

Our project is to examine a company stock data and look for patterns that could be used to predict how the stock would perform tomorrow. Our project includes research to compare the different Efficiency Similarity Search prediction approaches outlined in the paper “Efficient Similarity Search over Future Stream Time Series” by Xiang Lang, and Lei Chen [1]. The three prediction approach methods outlined in the paper (and to be compared in our project) are the polynomial prediction method, Discrete Fourier Transform prediction, and finally, probabilistic prediction methods.

Once successfully implemented and analyzed, the most effective method was chosen and further implement functionality to support not only our goal project of stock prediction, but develop an effective and efficient platform to be used with a wide array of time series based prediction applications, such as weather forecast predictions. By creating a class based implementation of our decided method of focus, it allows us and further research teams to easily apply the algorithm to a number of different applications for analysis and improvement.

Each team member worked on one of the methods provided in the research paper, *Efficient Similarity Search over Future Stream Time Series* [1] we read by Xiang Lian. Team members are helped to contribute and peer review and collaborate on each method of prediction. Once the a prediction method is chosen the team will divide the implementation up further for things such as further efficiency tuning, GUI, documentation to implement the solution into other industries/projects, testing, and so forth. Collaboration and peer review and thinking was quintessential the success of developing our project. Every player brings their own talent to the group that made it successful.

**3. Background**

* Related papers (or surveys for graduate teams)
* Software tools (DBMS, GUI, IDE, existing library, …)
* Required hardware
* Related programming skills (functions, Internet programming, object-oriented programming, distributed environment, etc.)

The paper that we read for this project was *Efficient Similarity Search over Future Stream Time Series* by Xiang Lian [1]. This paper showed us different techniques on how to predict future data using past data. The type of hardware required would be a computer running an OS within the past ten years. Our recommendations are that you use a Windows computer that runs Windows 7 or later.

Programming skills are Object Oriented Programming, using Databases, et al.

Programming Language we are using is C++.

Software tools we are using Microsoft Visual Studios, CLion IDE, GitHub for a team project repository, various sources of data from the internet, etc.

Primary source of data was weather data pulled from [www.data.gov](http://www.data.gov), a government funded source of data with a huge collection of accurate, consistent, and real world data. Our primary dataset was data that was pulled from a National Renewable Energy Laboratory’s on site weather data collection equipment at their Renewable Energy Laboratory located in Denver, CO. The data was then further process in excel and fed into our program of choice. [2]

**4. Problem Definition**

* Formal (mathematical) definitions of problems
* Challenges of tackling the problems
* A brief summary of general solutions in your project

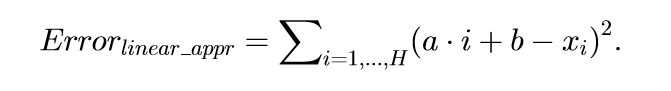
The formal definition is brought to us by Xiang Lian and Lei Chen in the paper *Efficient Similarity Search over Future Stream Time Series* [1].

The biggest difficulty we had was trying to understand how to implement the solutions given to us in the paper. It was hard converting the techniques into code that could be used to process the data and predict the next value. We found that translating mathematical and theoretical problems, equations, and solutions into tangible, function code to be the most challenging issue. Coupled with understanding how big data works, how to process and normalize the data, and just working with it in general, the project has been challenging.

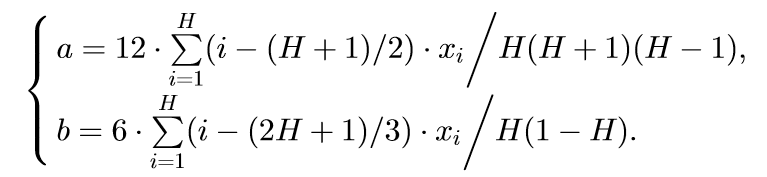
The primary basis for our project has been the three different prediction methods outlined in the research paper of focus. Below we will outline general solutions to each prediction approach:

Time Series: Is an array of data points that are indexed in a time arrangement. Time series is a sequence of data that is taken at successive equally spaced points in time. Due to this it can be used to look for patterns through the historical data that can be used to predict future occurrences based on previous patterns.

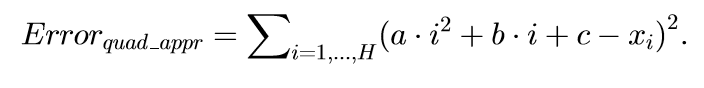
Polynomial Prediction: This prediction technique is done by using commonly used curves that we can use to approximate the older data and you can use this to predict future values according to these curves. Polynomial curve has two different techniques, the linear curve and the quadratic curve. There is a domain [min, max] that is used in the time series Ti. We use H values x1, x2,……, xH and we use this to predict Δt consecutive values like xH+1, xH+2,....xH+Δt into the future. Next you should consider the linear prediction which will be (H + Δt) values which we can approximate by an individual line in the form of x = a \* t + b where t will be the timestamp, x will be the evaluated value and the parameters of a and b characterize these (H + Δt) data. Because of this the predicted Δt values for the linear predictor are (H + 1) \* a + b, (H + 2) \* a + b,...., and (H + Δt) \* a + b, corresponding to xH+1, xH+2,....., and xH+Δt. To measure the approximation error we use,



We next need need to find good coefficients a and b for the error linear approximation. We use,

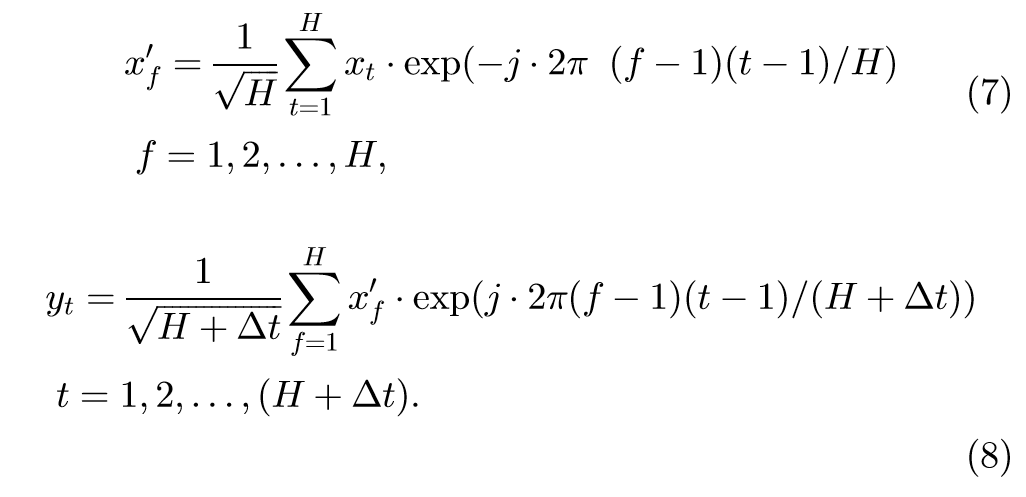


For the quadratic prediction, we can approximate these values by a quadratic curve in the form x = a \* t2 + b \* t + c, where a, b, and c are parameters that characterize the data. Approximation error of a quadratic curve is calculated by



As the program processes information, it begins to learn and create a polynomial curve that serves as a historical data representation so that we can accurately predict data in the future. By using commonly used curves we can better approximate our historical data as well as future data. Once a curve is selected, we try to minimize the approximate error by measuring the squared Euclidean distance between the actual series and our prediction to get a more accurate prediction. A major benefit of this method is that this solution is easily adaptable for accuracy by changing the number of entries that are used to build our approximation, important for good performance for things such as online streaming.

DFT Prediction: Is the next prediction technique we approximate data in an alternative domain, which is frequency domain. Believe that H historical data in values x1,x2,..., xH to predict Δt consecutive values xH+1, xH+2,...., xH+Δt for the future. First DFT obtains H coefficients x’1,x’2,...,x’H of the recent H historical data which is x1,x2,....,xH then reconstruct (H + Δt) to maybe y1,y2,...,yH+Δt. Δt reconstructed data of yH+1,yH+2,....,yH+Δt are the predicted values of xH+1,xH+2,..., and xH+Δt. DFT coefficients can be summarized in the proceeding formula.



DFT prediction is these frequencies of the H recorded data and (H + Δt) values (including both H recorded data and Δt future data) are expected to be very identical. What we do is measure the error in H DFT coefficients from the recorded data and (H + Δt) values, in other words the euclidean distance between H DFT coefficients from recorded data and from future values divided by the sum of H square coefficients which will come from (H + Δt) values. If any remaining data is left in the same frequency domain on the recorded data then future data is likely the DFT method will provide a correct answer. DFT method is approximates historical by using the frequency domain. Semantically, this approach is similar to the polynomial prediction model and follows a similar train of thought, however, it is based off of the frequency domain. If the historical and future data remains within the frequency domain, then the prediction will be accurate. This method has the ability to adapt at low cost due to easily modifiable DFT coefficients.

Probabilistic Prediction: The last prediction approach is based on the belief through observing that those subsequences that appear frequently in the recorded data have a higher percentage of recurring again in the future. We use statistics on the entire recorded data instead of the most recent values to predict the future values like polynomial or DFT. This probabilistic approach will extract the symbolic representation of the historical subsequences and this includes their aggregate information which can be used to predict the future symbols with probabilities because of aggregates that can summarize the entire data which can output future subsequences for identical search. Consider this, the prediction problem on a single time series we can call T where we extend to the case of n(>1) in the time series. The current timestamp, the most recent H recorded data have to predict future Δt values. To store the statistics of this we create a structure called the *aggregate trie*. So to update the trie we do in four steps.

1. divide it into *h* disjoint segments with identical length *l*, that is, H = h \* l,

2. obtain the mean value avgi for each segment i, where 1 ≤ i ≤ h,

3. convert each avgi into a symbol si, that is, transforming the subsequence to its symbolic representation s1s2 ...sh, and

4. insert the string s1s2 ...sh into the aggregate trie with height *h*.

So an example of this transformation, we’ll use the value of domain of a time series T’ is [-1.5, 1.5]. Next split it into three smaller ranges of equal sizes like [0.5, 1.5], [-0.5, 0.5), and [-1.5, -0.5), where we shall say each smaller range correspond to three symbols a, b, c. This time there is a case of priori knowledge of the data stream is known. you can divide the value domain into small ranges of different lengths. Next divide the time series T’ of length H into seven chunks of equal size. Take the average value avgi within each chunk i, and convert each avgi into a unique symbol. The mean value avg1 from the first chunk will go into the range [0.5, 1.5] so map it with the symbol of “a”. So we can have the discrete version of T’ in a string like “abcbbaa”, a bonus of this is time series will have space efficiency. Each node entry in the aggregate trie has a triple <freq, hit, miss> where *freq* is the frequency that a string appears in the time series, *hit* is the number of times our prediction was correct, and *miss* is the number of times it fails. If the symbol is wrongly predicted, we shall send back this confidence by adding it to aggregate miss as a penalty. The higher the confidence is, the greater the penalty will be in the case of failure.

**5. The Proposed Techniques**

* Framework (problem settings)
* Details of major techniques (e.g., pruning methods in lemmas/theorems; illustrated with toy examples)
* Encoding or indexing of data
* Query processing algorithms (pseudo code) and query optimizations
* This section can be split into multiple sections if you have many contents to present

The Framework is based predominantly on two missions, the first one is predicting the future values for each time series, and the second one being answering similarity queries on subsequences in the future. These techniques are designed to solve the problem of more accurately predicting future time series data. They could be used for stock market, weather data, or social media data predictions. They analyze past data streams and use them to predict future data.

The purpose of our project was to implement a prediction method for Time Data that works with live streaming series data. There were four different prediction methods that were attempted, they were the Polynomial Prediction, Linear Prediction, Probabilistic Prediction, and Discrete Fourier Transformation Prediction method given that it is a form of scalable machine learning which functions on the frequency domain. In the end we chose to pursue the Probabilistic Prediction method as it shows the greatest promise in accuracy and speed of predictions. In order to track and organize our data we used an R-tree indexing system with a height of 7 in order to store our data and reference it. The height of 7 was chosen due to hardware constraints. We implemented the probabilistic predictor as a class, for easy incorporation into client software,

The time series data to be predicted must be converted into a series of discrete symbols. Each symbol generally represents a band of values (unless the data itself is already narrow and discrete). The number of different symbols greatly affects performance, both in memory used, and prediction accuracy, and must stay constant, so choosing how to map the data onto symbols is an important choice when designing client software for the probabilistic predictor.

The algorithm uses a radix tree to process and predict the symbols. Each non-leaf node, has a number of children equal to the number of unique symbols. The number of levels of the tree are easily variable, and only really limited by system resources. The taller the tree the more complex patterns it is able to learn. The more symbols the tree uses, the more granularity the prediction has.

Each node in the tree keeps track of three values: frequency, hit, and miss. Sequences of symbols with a length equal to tree height minus one (to account for the root) can be used to traverse the tree from top to bottom. When the next symbol is received, the tree height minus one latest symbols (including the new one) are fed into the tree, increasing the frequency by one. Additionally, if there are existing predictions, is traversed down the path of the prediction increasing either hit or miss by the probability of the prediction depending on whether the prediction was correct or not. If a prediction had not been made for the new symbol yet, before inputting the symbol our predictor creates a prediction for it; this helps it learn faster even when the client software is not asking for predictions very often.

To predict a future symbol the tree is traversed using the latest tree height minus two symbols. Then, each child node of the deepest node reached is polled for their probability. The symbol corresponding to child node with the highest probability is selected as the prediction, and the prediction and probability are stored before giving them to the client software.

To predict farther than one symbol in advance, symbols are predicted from the last known symbol to the requested symbol. Previous predictions are used in the traversal sequence for each further out symbol. The probability of each new prediction is multiplied by the probability of the previous prediction upon which it is based, resulting in rapidly decreasing probabilities as predictions go further into the future.

**6. Visual Applications**

* GUI design
* Design modules (with descriptions, figures, and/or flowcharts)

For ease of interpretation we created and implemented a simple user interface that displayed information such as prediction accuracy, number of nodes, amount of data created, and predicted value.

**7. Experimental Evaluation**

* Experimental settings
  + Descriptions of real/synthetic data sets
  + Competitors (baseline method, or existing techniques to compare with)
  + Parameter settings
  + Evaluation measures
* The performance report (pruning power, recall/precision/f-measure, CPU time, I/O cost, communication cost, index construction time/space, etc.)
* Screen captures

For our experiment we used a weather data from Seattle that was generated over a 24 hour period. However our Algorithm is compatible with all forms of Time Series Data. Existing methods of streaming time series analysis focused on solely on prediction speed while our design focuses on a mix of both speed and accuracy. Other methods of predicting streaming time series tend to have a basis in machine learning algorithms and working with data that has already been generated. To evaluate the effectiveness of our model we compared our predicted values with the actual outcome value. The values we used in our project were the temperature in Denver, CO in the degrees Celsius.

Some of the competitors to our time series prediction technique are: Dynamic Time Warping, Longest Common Subsequence, Auto Regressive Integrated Moving Average, Edit Distance on Real Sequence, and Edit Distance with Real Penalty. For our time series we are going to use Euclidean Distance.

We tried using stock data but unfortunately it didn’t work well with our prediction process. So instead we used hourly temperatures from weather data [2]; this data was pulled from [www.data.gov](http://www.data.gov), a government funded source of data with a huge collection of accurate, consistent, and real world data. Our primary dataset was data that was pulled from a National Renewable Energy Laboratory’s on site weather data collection equipment at their Renewable Energy Laboratory located in Denver, CO. The data was then further process in excel and fed into our program of choice.. As a control to compare our prediction against, we used a naive prediction that always assumed the next symbol is the same as the last.

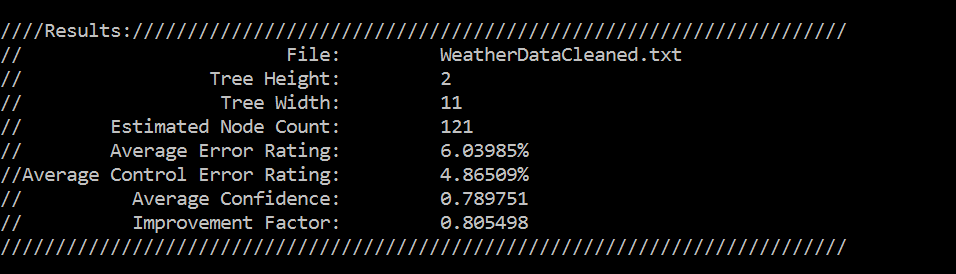
To create our mapping from temperature to symbols we considered the minimum useful granularity of temperature, which we decided was five degrees celsius. By examining a sample of our temperature data, we determined that we would need eleven distinct symbols to represent our data to that degree of granularity. The bands on the upper and lower ends actually stretch to infinity in their direction, but were placed such that in most cases temperature values only use five degrees of their domain.

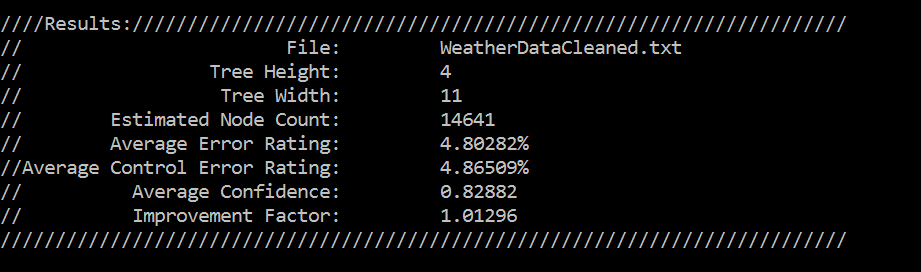
By doing some calculations of system resources we determined that the tallest tree we could have was seven levels tall (not counting the root). As a result of this we decided to preprocess the hourly temperature data into four hour averages, resulting in six values per day of data; this way the symbol sequence of an entire day would be shorter than our tree, thus allowing it to capture an entire day night cycle.

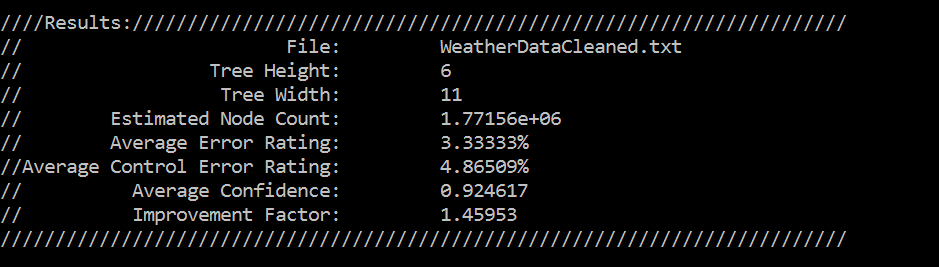
Data was then fed into the predictor to train it. Then, we fed a year of data into it one value at a time. After each value was fed in, we asked it to predict the next symbol, used recorded an error rating using the formula (err=|ActualSymbol - PredictedSymbol|/11), and did the same with the naive control prediction. We output all this information to the console. After the year of input is processed we reported an average error for both the probabilistic predictor and the naive control predictor, and found that the probabilistic predictor was almost twice as accurate.

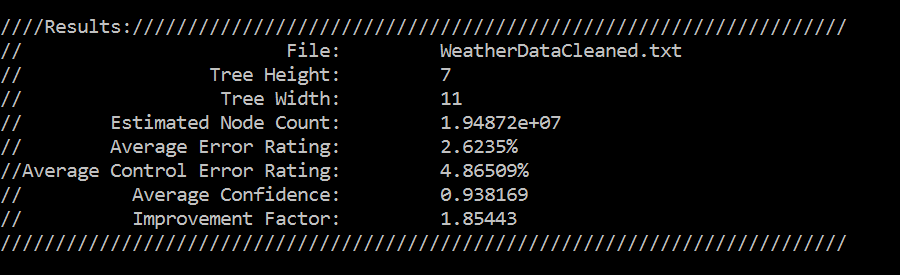
We then found that, with some optimizations of the implementation, we could get an eight level tree to run on a 64-bit machine we had available. With eight levels it is almost three times as accurate as the naive control prediction.

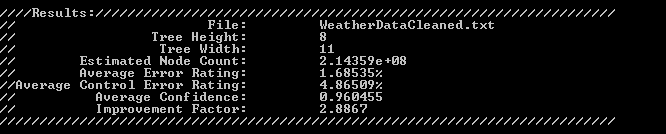
We then repeated the test with shorter trees and found that the probabilistic predictor’s accuracy increased rapidly as the height of the tree increased. With greater resources we could run the predictor with taller trees resulting in vastly increased accuracy. Current hardware limited our tree height and subsequently our accuracy, to eight nodes tall.











**8. Future Work**

* Possible project extensions

There are a number of ways to further improve upon our algorithm that we have developed. Possible future work could be extending the tree height to increase accuracy, along with upgraded hardware to handle the increased complexity. Ideally, a cloud implementation (HADOOP, AWS, Azure, etc.) based program would be the most effective way to utilize the algorithm due to its increased ability for high levels of memory. It would allow us to increase the height of the tree, and subsequently make more accurate predictions. We could implement the software in Java from C++ which may yield some difficulties in transcribing the tree but is possible.

**9. References**

[1] Xiang Lian, Lei Chen. Efficient Similarity Search over Future Stream Time Series. In 2008.

[2] NREL RSF Weather Data 2011, [www.data.gov](http://www.data.gov), rsfweatherdata2011.csv, https://catalog.data.gov/dataset/nrel-rsf-weather-data-2011-675fe