Anomaly Detection Application using Stream Mill

Final Project, MS Computer Science

Winter 2012

Ye Tian
Department of Computer Science
University of California, Los Angeles

ABSTRACT
Anomaly Detection, which is defined as detecting events that deviate from a normal behavior, has recently been attracting a lot of attention. The administrator of a server may use anomaly detection to monitor real-time system statistics and to raise alerts if an abnormal usage pattern is detected. Anomaly detection can also be used for detecting suspicious behavior in network traffic. Most research on anomaly detection, however, is application-specific and works based on predefined rules. Furthermore, current anomaly detection algorithms are not scalable and cannot adapt to a changing behavior in the input. Our project takes advantage of Stream Mill, a Data Stream Management System designed at UCLA, to construct a light-weight statistical model of the stream that can be used to make predictions of a parameter of interest based on which anomalous behavior can be detected. The model we use is derived from a classical auto regressive model, which is a one-dimension time-series model. Since a large portion of data-streams corresponding to the applications (e.g., real-time system statistics, web-traffic) of interest are one-dimensional time-series, our approach is attractive. We conduct experiments that demonstrate the scalability and efficiency of our approach.
1. Introduction

Nowadays, data streams are playing a more and more important role in information management. Data stream is a real-time database, which is growing continuously. It is useful in many areas, including wireless sensor networks[1] and traffic analysis[2]. With increasing network bandwidth, data stream applications are facing the challenge of retrieving large amounts of data every second. Despite the fact that hardware is becoming more powerful, people are still interested in finding ways to make the application more efficient. Applications that are specially designed for processing data streams are called Data Stream Management System (DSMS).

[3]Compared to Database Management System, DSMS processes data continuously and produces query results almost in real-time. In terms of language, most DSMS uses one similar to SQL. For example, SQL-TS can find complex patterns by rules, and K*SQL uses regular expression to match patterns. Some DSMS can also deal with data in XML format, using corresponding languages such as XQuery, XSLT or XPath. There're many commercial DSMS, including Streambase[4], Sybase Aleri Event Stream Processor(Coral8)[5], Apama[6] and Truviso[7], as well as more projects on DSMS such as AT&T’s Gigascope[8], Berkeley’s Telegraph[9] and UCLA’s Stream Mill[10].

[11]Stream Mill is a DSMS which uses ESL as its query language. [12]ESL stands for Expressive Stream Language, which is also designed by the same lab. Based on SQL, it extends the power and generality of DSMS. With the help of this language, Stream Mill overcomes a major problem of traditional SQL, that blocking operators cannot be used in continuous streams. For example, aggregates such as MIN, MAX, SORT are blocking operators because they cannot output the correct results until they review the entire dataset, thus no outputs would be provided given the dataset is continuous. Stream Mill solves this problem in two ways — sliding window and User Defined Aggregates (UDA). Sliding window enables queries like “output the maximum price over last 200 tuples every 1 tuple”, where WINDOW SIZE is 200 tuples and the query produces output every time a new tuple is received. UDA provides another solution, which can be defined in both SQL and C/C++. This allows users to create more powerful and sophisticated aggregates that served as a data mining algorithm. There’re currently a lot of applications written using UDA, such as Parallel Finite State Automata (FSA) for processing SAX stream[13].

Recently, there’s growing interest on data stream mining, partially because of increasingly powerful tools. Anomaly detection is one of the areas that draws much attention. This refers to detecting patterns that do not conform to normal behavior[14]. Anomaly Detection is applicable to many domains, such as intrusion detection, fraud detection, sensor networks. Traditionally, anomaly detection is manually carried out by looking at the monitor with graphs and statistics, which is very time consuming and does not meet real-time criteria. Later, many researchers found automated statistical and machine learning approaches to anomaly detection. Hill and Minsker[15] use data-driven regression models to perform automated fault detection for sensors. However, it is incapable of perform anomaly detection on multiple data streams simultaneously. Consequently, they came out with a new method based on Dynamic Bayesian Networks. Besides sensor network, some researchers are also interested in intrusion detection. Jung[18] uses sequential hypothesis testing, which depends on a likely ratio to test a hypothesis and to detect
Portscan. In the same lab, Schechter [19] uses reverse sequential hypothesis testing, which uses a similar process except with observations in reverse chronological order, to detect worm infections. There are still some researchers who are dedicated to more generalized anomaly detection algorithm. Teng[20] detects anomalies by finding sequences that deviates significantly from some sequential rules.

**Our Contribution**

Anomaly detection is useful in many areas. For example, the administrator of a server may use anomaly detection to monitor real-time system statistics and to raise alerts if an abnormal usage pattern is detected. Anomaly detection can also be used for detecting suspicious behavior in network traffic. More commonly, users can track the price of products and wait for the moment when the price is lower than predicted.

However, most researches on anomaly detection are dedicated to specific areas such as sensor network and intrusion detection, while others are based on predefined rules. There is no anomaly detection platform serving general purposes, which means all kinds of data stream and without using predefined assumptions. Meanwhile, the algorithms from these researches are not efficient and self-adaptable. The models are based on data from the very beginning to the current day, thus out-of-date data is still kept. Also, the models are updated every time a new tuple arrives, which may not always be necessary.

In this project, we implement a generalized anomaly detection algorithm based on a light-weight classical statistical model and then develop a web application to monitor the data streams themselves, together with the trend. We keep a fixed length window on the data stream so that the model is always based on previous data in the window. Every time a new tuple arrives, we compare it with the predicted value by the model and decide whether or not to update it according to the difference between two values.

The model we use is based on a classical auto regressive model, which is a one-dimension time-series model. Since most data streams are one-dimension time-series, such as real-time CPU rate and real-time website hits, our model is general to some extent.

Our web application has the following features:

- Log in with Stream Mill account.
- The data is updated every time a unit as demanded.
- There is a real-time graph, which can be hid, for each datastream.
- Both real data and predicted data are shown in each real-time graph.
- User can configure some of the parameters of datastream, such as update frequency.
- When anomalies are detected, the user will receive an email.

**Project Overview**

Here’s the big picture of our project. (Figure 1)
When a user wants to use our product to trace data trends, he opens browser, logs into our web application and activates certain data streams. At this time, our web application server knows what data the user wants, so that it asks our stream mill server to retrieve corresponding data. Each tuple contains four attributes: timestamp, real value, predicted value and a flag indicating whether there’s anomaly or not.

Meanwhile, the stream mill server keeps receiving data streams by listening on specific ports. Each data stream is then processed by our prediction UDA. The UDA maintains a prediction model which is based on past data. Every time a new tuple comes, a prediction is made using the model. Anomaly is detected when the prediction makes a big difference from the real value.

After the output tuple is built, it is stored in a particular place where the older tuple is replaced by new one. Whenever the web application server requests data from stream mill server, the stream mill server will search for the particular place to get the latest tuple. As the result, the user always gets the newest data and prediction.

2. Anomaly Detection

In this project, we use AR(2) model to make prediction of data trend.

[21]Autoregressive (AR) model is a type of random process which is often used to model and predict various types of natural phenomena. The autoregressive model is one of a group of linear prediction formulas that attempt to predict an output of a system based on the previous outputs. In particular, AR(2) predicts an output based on previous 2 outputs.
The notation of AR(p) means autoregressive model of order p. It can be defined as:

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t,$$  \hspace{1cm} (1)

where $\varphi_i$ is parameter of the model, $c$ is a constant and $\varepsilon_t$ is white noise.

Consider the example of sensor network where $X_t$ stands for the data stream produced by a sensor, $\varepsilon_t$ can be considered as measure error.

Let’s consider the p=2 case and omit the white noise now, which makes equation into:

$$X_t = c + \varphi_1 X_{t-1} + \varphi_2 X_{t-2}.$$  \hspace{1cm} (2)

### Ordinary Least Square Method[22]

Ordinary least square (OLS) is the method we use to estimate parameters in (2). It is a classical method in linear regression. It estimates parameters by minimizing sum of squared vertical distances between observed value and value predicted by linear approximation.

Given a dataset $\{X_1, X_2, ..., X_T\}$, where $T > 4$. The goal can be defined as:

$$\text{Find } c, \varphi_1 \text{ and } \varphi_2 \text{ so that minimize}$$

$$\sum_{t=3}^{T} (X_t - c - \varphi_1 X_{t-1} - \varphi_2 X_{t-2})^2.$$  \hspace{1cm} (3)

Denote $\beta = (c, \varphi_1, \varphi_2)'$, $x_t = (1, X_{t-1}, X_{t-2})'$, $y_t = X_t$. (3) can be re-written as:

$$S(\beta) = \sum_{t=3}^{T} (y_t - x_t' \beta)^2 = (Y - X\beta)'(Y - X\beta),$$  \hspace{1cm} (4)

where $Y = (y_3, y_4, ..., y_T)'$, $X = (x_3, x_4, ..., x_T)'$. The goal can be re-written as

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^3} S(\beta)$$  \hspace{1cm} (5)

Since (4) is quadratic in $\beta$ with positive-definite Hessian, the minimum can be obtained by differentiating it with respect to $\beta$.

$$0 = \frac{dS}{d\beta}(\hat{\beta}) = \frac{d}{d\beta}(Y'Y - \beta'X'Y - Y'X\beta + \beta'X'X\beta) \bigg|_{\beta = \hat{\beta}} = -2X'Y + 2X'X\hat{\beta}$$
Assuming that $X$ has full column rank, which means $X'X$ is invertible. Therefore, the estimate of linear model parameter can be given by:

$$\hat{\beta} = (X'X)^{-1}X'Y$$

(6)

**Regression and Forecast Error**

After estimating the parameter, we get the model ($p=2$) as:

$$X_t = \hat{c} + \sum_{i=1}^{2} \hat{\phi}_i X_{t-i} + \varepsilon_t,$$

(7)

where $\hat{\phi}_i$ is the estimate of $\phi_i$ and $\hat{c}$ is the estimate of $c$. Using one of the Yule-Walker equations with constant $c[23]$:  

$$\gamma_0 = \hat{c}E[X_t] + \sum_{i=1}^{2} \hat{\phi}_i \gamma_{t-i} + \sigma^2,$$

(8)

where $\gamma_m = E[X_tX_m]$, which is autocorrelation function of $X$. $\sigma^2$ is the standard deviation of the white noise $\varepsilon_t$. By solving (8), we can calculate $\sigma^2$, which is also the regression error.

With both regression model and error at hand, we can now move on to forecast. Using the model built from $\{X_1, X_2, \ldots, X_T\}$, the next-time value $X_{T+1}$ can be forecasted as:

$$\hat{X}_{T+1} = \hat{c} + \sum_{i=1}^{2} \hat{\phi}_i X_{T+1-i} + \varepsilon_{T+1}.$$

(9)

This equation means that $\hat{X}_{T+1}$ is under a normal distribution with mean of $\hat{c} + \sum_{i=1}^{2} \hat{\phi}_i X_{T+1-i}$ and standard deviation of $\varepsilon_{T+1}$. More often, people use a statistic criterion called confidence interval to indicate the reliability of an estimate. In this project, we use 95% confidence interval, which guarantees a 95% chance that the real value would lay in this interval.
Figure 2 shows us pretty much everything about 95% confidence interval, which is the shaded area. The interval is \([-1.96, 1.96]\) for normal distribution with mean of 0 and standard deviation of 1. As for \(X_{T+1}\), the confidence interval is:

\[
[\hat{c} + \sum_{i=1}^{2} \phi_i X_{T+1-i} - 1.96\sigma_e, \hat{c} + \sum_{i=1}^{2} \phi_i X_{T+1-i} + 1.96\sigma_e]
\]  

(10)

UDA

Gretl Library

3. Data Monitor

We build a web application mainly using PHP and JAVASCRIPT to monitor data stream trends and anomalies.

4. Data Retrieving

5. Experiments
6. Conclusion and Future Work

7. Acknowledgement

Reference

[1]. De Aquino, Figueiredo and etc, Data Stream Based Algorithms For Wireless Sensor Network Applications
[2]. Thomas Plagemann, Vera Goebel and etc, Using Data Stream Management Systems for Traffic Analysis – A Case Study,
[7]. Truviso website: http://www.truviso.com/
[9]. Telegraph website: http://telegraph.cs.berkeley.edu/
[15].David J. Hill and Barbara S. Minsker, Automated fault detection for In-situ Environmental Sensors, HIC2006, Nice, France
Privacy, Oakland, CA, May, 2004


