Rule-Based Anomaly Pattern Detection for Detecting Disease Outbreaks

Weng-Keen Wong Andrew Moore
Gregory Cooper[†] Michael Wagner[†]

February 2002
CMU-CS-02-106

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[†] Center for Biomedical Informatics, University of Pittsburgh, Pittsburgh, PA, 15123

This research was supported by the DARPA Biosurveillance Program under DARPA Grant No. F30602-01-2-0550, ARPA Order No. M167/00

Keywords: Machine Learning, Data Mining, Anomaly Detection, Disease Outbreak Detection

Abstract

Searching for anomalies in multidimensional data with a temporal component is a difficult task especially when the exact features of the anomalies are unknown. A standard but simplistic algorithm would be to obtain counts of certain events over a time interval such as a day and mark that interval to contain anomalies if this count exceeds a threshold. This naive approach misses anomalies that aggregate in feature space but do not occur frequently enough to skew the count of monitored events over the time interval. A desired solution should find these anomalous *patterns* rather than individual anomalies. In order to approach this problem, we propose using a rule-based anomaly detection algorithm that characterizes each anomalous pattern with a rule. The significance of each rule is carefully evaluated using Fisher's Exact Test and a randomization test. The performance of our algorithm is compared against the standard algorithm by measuring the number of false positives and the timeliness of detection. Simulated data is used in the evaluation phase. This data was produced by a simulator that simulates the effects of a disease outbreak on a city. The results indicate that our algorithm has significantly better detection times for common significance thresholds while having a slightly higher false positive rate.

1 Introduction

Multidimensional data with a temporal component is available from numerous disciplines such as medicine, engineering, and astrophysics. This data is commonly used for monitoring purposes by a detection system. These systems inspect the data for unusual perturbations and raise an appropriate alert upon discovery of any deviations from the norm. For example, in the case of an intrusion detection system, an anomaly would indicate a possible breach of security [Lane and Brodley, 1999, Eskin, 2000, Maxion and Tan, 2001]. For a disease outbreak detector, irregularities in data related to health care could signal a potential epidemic or bioterrorist attack [Wagner et al., 2001, Goldenberg, 2001].

The majority of detectors only focus on one attribute of the data, usually a count of certain events over some time interval, as a warning signal. Typically these detectors first determine the mean and variance of this warning signal over a training set which is assumed to display the normal behaviour of the system. Alerts are then raised if the test set signal exceeds some threshold, where the standard threshold is defined as three standard deviations above or below the mean. An anomaly is considered to have occurred within the time interval that the alert was raised.

While this approach is successful if the appropriate feature is chosen, many potential anomalies are overlooked if the full space of attributes is not evaluated. Consider the following scenario in a disease outbreak detection system which operates on a database of Emergency Room (ER) cases. A traditional monitoring system would use the daily counts of ER cases exhibiting a specific symptom as an indication of the presence of an epidemic. Now suppose a disease is present in a neighbourhood and it mainly affects elderly males living in that area. This insidious disease does not cause the number of monitored ER cases to exceed the alert threshold yet if one were to examine the number of male senior citizens from the infected area, one would notice an abnormally high number for that day.

This example illustrates a need for anomaly *pattern* detection rather than traditional anomaly detection. Quite often, an isolated irregularity will seem insignificant to the detection system. However, if many similar anomalies are present in the data, then there is sufficient reason to raise an alert. Most detectors are also interested in characterizing anomalies so that an appropriate response can be formulated. Anomaly pattern detection allows the system to classify the anomalies based on the traits of the anomalous group.

Our approach to this problem is to use a rule-based anomaly pattern detector. One of the key advantages to a rule-based system is that the results are a set of rules that can be easily understood by a non-statistician. Each anomalous pattern is summarized by a rule, which in our current implementation consists of one or two components. Each component takes the form $X_i = V_i^j$, where X_i is the ith feature and V_i^j is the jth value of that feature. Multiple components are joined together by a logical AND. A two component rule, as an example, would be something like Gender = Male and Age_Decile = 4.

However, we need to be wary of the pitfalls of rule-based anomaly pattern detection. As the number of attributes in the data increases, so does the space of possible irregularities. In addition, as more components are added to a rule, overfitting becomes a serious concern. As a result, a careful evaluation of significance is needed. Furthermore, temporal data, especially health care data used for disease outbreak detection, is frequently subject to "seasonal" variations. As an illustration, consider a database of visits to physicians which is used to detect if a flu epidemic exists in the current population. Typically, the number of flu cases is higher during the winter than during the summer. Additionally, frequencies of physician visits vary between weekends and weekdays. The definition of what is normal will vary depending on the timing of the events, thereby affecting the discovery of irregular patterns.

2 Rule-based Anomaly Pattern Detection

The basic question asked by all detection systems is whether anything strange has occurred in recent events. This question requires defining two concepts – what it means to be recent and what it means to be an anomaly. Our algorithm will consider all records falling on the current day under evaluation to be recent events. Note that this definition of recent is not restrictive – it can be redefined to include all events within some other time period. In order to define an anomaly, we need to establish the concept of something being normal. Our algorithm is intended to be applied to a database of ER cases and we need to account for environmental factors such as weekend versus weekday differences in the number of cases. Consequently, normal behaviour is assumed to be captured by the events occurring on the days that are exactly five, six, seven, and eight weeks prior to the day under consideration. For another domain, the environmental factors will be different. The definition of what is normal can be easily modified without major changes to our algorithm. We will refer to the events that fit a certain rule for the current day as C_{today}. Similarly, the number of cases matching some criteria from five to eight weeks ago will be called C_{other}.

From this point on, we will refer to our algorithm as WSARE, which is an abbreviation for "What's strange about recent events". WSARE operates on discrete data sets with the aim of finding rules that characterize significant patterns of anomalies. Due to computational issues, the number of components for these rules is two or less. The description of the rule-finding algorithm will begin with an overview followed by a more detailed example.

2.1 Overview of WSARE

The best rule for a day is found by considering all possible one and two component rules over events occurring on that day and returning the one with the best "score". The score is determined by comparing the events on the current day against events in the past. Following the score calculation, the best rule for that day has its p-value estimated by a randomization test. The p-value for a rule is the likelihood of finding a rule with as good a score under the hypothesis that the case features and date are independent. Finally, if we were running the algorithm on a day-by-day basis we would end at this step. However, if we were looking at a history of days, we would need the additional step of using the False Discovery Rate (FDR) method [Benjamini and Hochberg, 1995, Miller et al., 2001] to determine which of the p-values are significant. The days with significant p-values are returned as the anomalies.

2.2 One component rules

In order to illustrate this algorithm, suppose we have a large database of 1,000,000 ER records over a two-year span. This database contains roughly 1000 records a day, thereby yielding approximately 5000 records if we consider the cases for today plus those from five to eight weeks ago. We will refer to this record subset as DB_i, which corresponds to the recent event data set for day i. The algorithm proceeds as follows. For each day i, retrieve the records belonging to DB_i. We first consider all possible one-component rules. For every possible feature-value combination, obtain the counts C_{today} and C_{other} from the data set DB_i. As an example, suppose the feature under consideration is the Age_Decile for the ER case. There are 9 possible Age_Decile values, ranging from 0 to 8. We start with the rule Age_Decile $= 3$ and count the number of cases for the current day i that have Age_Decile $= 3$ and those that have Age_Decile $\neq 3$. The cases from five to eight weeks ago are subsequently examined to obtain the counts for the cases matching the rule and those not matching the rule. The four values form a two-by-two contingency table such as the one shown in Table 1.

2.3 Scoring each one component rule

The next step is to evaluate the "score" of the rule using a hypothesis test in which the null hypothesis is the independence of the row and column attributes of the two-by-two contingency table. In effect, the hypothesis test measures how different the distribution for C_{today} is compared to that of C_{other}. This test will generate a p-value that determines the significance of the anomalies found by the rule. We will refer to this p-value as the *score* in order to differentiate this p-value from the p-value that is obtained later on from the randomization test. Since the counts for the entries in the contingency table involve relatively small numbers, we use Fisher's Exact Test [Good, 2000] to find the score for each rule. If the expected values for all of these counts is above five, we will use the Chi Squared test instead of Fisher's Exact Test. Running Fisher's Exact Test on Table 1 yields a score of 0.00005058, which indicates that the count C_{today} for cases matching the rule Age_Decile $= 3$ are significantly different from the count C_{other}.

Table 1: A sample 2x2 Contingency Table

	C_{today}	C_{other}
$Age\_Decile = 3$	48	45
$Age\_Decile \neq 3$	86	220

2.4 Two component rules

At this point, the best one component rule for a particular day has been found. We will refer to the best one component rule for day i as BR_i^1. The algorithm then attempts to find the best two component rule for the day by adding on one extra component to BR_i^1. This extra component is determined by supplementing BR_i^1 with all possible feature-value pairs, except for the one already present in BR_i^1, and selecting the resulting two component rule with the best score. Scoring is performed in the exact same manner as before, except this time, the counts C_{today} and C_{other} are calculated by counting the records that match the two component rule. The best two component rule for day i is subsequently found and we will refer to it as BR_i^2

BR_i^2, however, may not be an improvement over BR_i^1. We need to perform further hypothesis tests to determine if the presence of either component has a significant effect. This can be accomplished by determining the scores of having each component through Fisher's Exact Test. If we label BR_i^2's components as C_0 and C_1, then the two two-by-two contingency tables for Fisher's Exact Tests are as follows:

Table 2: 2x2 Contingency Table 1 for a two component rule

Records from Today matching C_0 and C_1	Records from Other matching C_0 and C_1
Records from Today matching C_1 and differing on C_0	Records from Other matching C_1 and differing on C_0

Table 3: 2x2 Contingency Table 2 for a two component rule

Records from Today matching C_0 and C_1	Records from Other matching C_0 and C_1
Records from Today matching C_0 and differing on C_1	Records from Other matching C_0 and differing on C_1

Once we have the scores for both tables, we need to determine if they are significant or not. We used the standard α value of 0.05 and considered a score to be significant if it was less than or equal to α. If the scores for the two tables were both significant, then the presence of both components had an effect. As a result, the best rule overall for day i is BR_i^2. On the other hand, if any one of the scores was not significant, then the best rule overall for day i is BR_i^1.

2.5 Finding the p-value for a rule

The algorithm above for determining scores is extremely prone to overfitting. Even if data were generated randomly, most single rules would have insignificant p-values but the best rule would be significant if we had searched over 1000 possible rules. In order to illustrate this point, suppose we perform the standard practice of rejecting the null hypothesis when the p-value is $< \alpha$, where $\alpha = 0.05$. In the case of a single hypothesis test, the probability of making a false discovery under the null hypothesis would be α, which equals 0.05. On the other hand, if we perform 1000 hypothesis tests, one for each possible rule under consideration, then the probability of making a false discovery could be as bad as $1 - (1 - 0.05)^{1000} \approx 1$, which is much greater than 0.05 [Miller et al., 2001]. Thus, if our algorithm returns a significant p-value, we cannot accept it at face value without adding an adjustment for the multiple hypothesis tests we performed. This problem can be addressed using a Bonferroni correction [Bonferroni, 1936] but this approach would be unnecessarily conservative. Instead, we turn to a randomization test in which the date and each ER case features are assumed to be independent. In this test, the case features in the data set DB_i remain the same for each record but the date field is shuffled between records from the current day and records from five to eight weeks ago. The randomization test goes through several iterations until it can accurately determine a p-value for the rule. This p-value indicates how likely it would be to find a rule with as good a score under the hypothesis that the case features are independent of the date. In order to calculate this p-value, the score of the best rule is calculated on the data subset DB_i, which has the dates randomized on each iteration. The algorithm keeps track of how many times the new score was better than the original score. The p-value that is reported is simply the number of better scores over the number of iterations.

2.6 Using FDR to determine which p-values are significant

This algorithm can be used on a day-to-day basis similar to an online algorithm or it can operate over a history of several days to report all significantly anomalous patterns. When using our algorithm on a day-to-day basis, the p-value obtained for the current day through the randomization tests can be interpreted at face value. However, when analyzing historical data, we need to compare the p-values for each day in the history. Comparison of multiple p-values results in a second overfitting opportunity analogous to that caused by performing multiple hypothesis tests to determine the best rule for a particular day. As an illustration, suppose we took 500 days of randomly generated data. Then, approximately 5 days would have a p-value less than 0.01 and these days would naively be interpreted as being significant. Two approaches can be used to correct this problem. Again, the Bonferroni method [Bonferroni, 1936] aims to reduce the probability of making at least one false positive to be no greater than α. However, this tight control over the number of false positives causes many real discoveries to be missed [Miller et al., 2001]. The other alternative is the False Discover Rate (FDR) method [Benjamini and Hochberg, 1995, Miller et al., 2001], which guarantees that the fraction of the number of false positives over the number of tests in which the null hypothesis was rejected will be no greater than α. The FDR method is more desirable as it has a higher power than the Bonferroni method but still has reasonable control over the number of false positives. We incorporate the FDR method into our rule-learning algorithm by first providing an α value and then using FDR to find the cutoff threshold for determining which p-values are significant.

3 The Simulator

Validation of our algorithm becomes a difficult task due to the type of data required. Data consisting of ER cases during a disease outbreak is extremely limited and there are no available databases of ER cases during a bioagent release. To make matters more difficult, evaluation of our anomaly pattern detector requires a large amount of data that has records that are labeled as either anomalies or normal events. In most cases, this task requires a human to perform the labelling by hand, resulting in insufficient amounts of data.

As a result of these limitations, we resort to evaluating our algorithm using data from a simulator.

The simulator is intended to simulate the effects of an epidemic on a population. The world in this simulator consists of a grid in which there are three types of objects – places, people, and diseases. These three objects interact with each other in a daily routine for a fixed number of days. Each of these objects will be described in detail below.

3.1 Places

The three types of places in the simulator include homes, businesses, and restaurants. Their roles are evident from what they represent in real life. People reside in homes, work in businesses and eat in restaurants.

3.2 People

Each person in the simulation has a specified gender and age. Genders for the population are distributed uniformly between male and female while ages follow a normal distribution with mean 40 and standard deviation of 15. People have a home location, a work location, a list of restaurants that they eat at and a list of homes of friends that they like to visit. The locations of work, restaurants, and friends' homes are chosen to be in close proximity to a person's home. On each day, a schedule is generated for a person. In this schedule, people sleep at home until it is time to go to work. They go to work, stop for a lunch break at a restaurant, and then return to work. After work, they spend some time at home before going to a restaurant for dinner. Following dinner, they visit a random selection of friends at their houses. Finally they return home to sleep.

3.3 Diseases

Diseases are the most complex objects in the simulator as they are designed to allow the creation of a large variety of disease models. People, places and grid cells can all serve as infection agents since they can all carry a disease. With infected places, we can create diseases that spread by a contaminated food supply while with infected grid cells, we can model airborne infections. Associated with each disease is a spontaneous generation probability which corresponds to how likely the disease is to appear in the population at each timestep. Typically, this probability is extremely small. Each disease also progresses through several stages at different rates. On each stage, the infected person can exhibit a variety of symptoms. The current simulation chooses randomly from a list of symptoms at each stage of the disease. At the final stage, an infected agent can either recover or die. The deceased are removed from the simulation.

The entire infection process revolves around the infection probability, which controls how easily an infected person can pass the disease on to another on each timestep. A radius parameter determines how close a person needs to be to catch the disease. The simulator only allows a person to have one disease at a time. Should more than one disease infect a person, the priority of an epidemic arbitrates which disease is assigned to the person. Diseases can be designed to spread from one particular type of agent to another for example place to person, person to person, or grid cell to person. Additionally, each disease has a specific demographic group that it infects. Whenever it has an opportunity to spread to a person outside of this demographic group, the infection probability is reduced to a small percentage of its original value.

We do not have hospitals in the simulation. Instead, when people exhibit a certain symptom, we create an emergency room case by adding an entry to a log file. This entry contains information such as the person id, the day, the time, the current location of the person, the home location of the person, and any demographic information about the individual. Most importantly, we add to each entry the actual disease carried by that person.

4 Results

4.1 Simulation Settings

Our results were obtained by running the simulator on a 50 by 50 grid world with 1000 people, 350 homes, 200 businesses, and 100 restaurants. The simulation ran for 180 simulated days with the epidemic being introduced into the environment on the 90th day. There are 9 background diseases that spontaneously appeared at random points in the simulation. At certain stages, these background diseases caused infected people to display the monitored symptom. These background diseases had low infection probabilities as they were intended to provide a baseline for the number of ER cases. The epidemic, on the other hand, had a higher priority than the background diseases and it had a relatively high infection probability, making it spread easily through its target demographic group.

The epidemic that we added to the system will be referred to as Epidemic0. This disease had a target demographic group of males in their 50s. Additionally, the disease is permitted to contaminate places. Epidemic0 had 4 stages with each stage lasting for 2 days. The disease was contagious during all 4 stages. At the final stage, we allowed the person to recover instead of dying in order to keep the total number of people in the simulation constant. Epidemic0 also exhibited the monitored symptom with probability 0.33 on the 3rd stage, probability 1.0 on the final stage, and probability 0 on all other stages. This disease was designed to produce a subtle increase in the number of daily ER counts rather than causing extreme perturbations that could easily be picked up by the naive algorithm.

4.2 Evaluation of performance

We treated our algorithm as if it ran on a day-by-day basis. Thus, for each day in the simulation, WSARE was asked to determine if the events on the current day were anomalous. We evaluated the performance of WSARE against a standard anomaly detection algorithm that treated a day as anomalous when the daily count of ER cases for the monitor symptom exceeded a threshold. The standard detector was allowed to train on the ER case data from day 30 to day 89 in the simulation to obtain the mean μ and variance σ^2. The threshold was calculated by the formula below, in which Φ^{-1} is the inverse to the cumulative distribution function of a standard normal.

$$\text{threshold} = \mu + \sigma * \Phi^{-1}(1 - \frac{\text{p-value}}{2})$$

In order to illustrate the standard algorithm, suppose we trained on the data from day 30 to 89. The mean and variance of the daily counts of the monitored symptom on this training set were deteremined to be 20 and 8 respectively. Given a p-value of 0.05, we calculate the threshold as $20 + 1.96 * 8 = 35.68$. After training, the standard algorithm is run over all the days of data from day 0 to day 179. Any day in which the daily count of the particular symptom exceeds 35.68 is considered to contain anomalous events.

Both the standard algorithm and WSARE were tested using five levels of p-values (0.1, 0.05, 0.01, 0.005, and 0.001). In order to evaluate the performance of the algorithms, we measured the number of false positives and the number of days until the epidemic was detected. Note that there were two files used in this evaluation step. The first file is the database of ER cases produced by the simulator, which we will refer to as DB_{ER}. The second file is the list of anomalous days reported by the algorithm, which we will refer to as DB_{Anom}. We will call the subset of anomalies having a p-value below the ith p-value level as DB^i_{Anom}.

1. **Counting the number of false positives**
   The number of false positives for the ith p-value level was determined by checking each day in DB^i_{Anom} against DB_{ER}. If a case of the epidemic was not reported in DB_{ER} for that day, then the false positive count was incremented. However, since WSARE relies on data from five to eight weeks prior to the current day, detection does not begin until Day 56. In order to be fair, any false positives found before Day 56 in the standard algorithm were not included.

2. **Calculating time until detection**
   The detection time for the ith p-value level was calculated by searching for the first day in DB^i_{Anom} in which an epidemic case appeared in DB_{ER}. If no such days are found, the detection time was set to be 90 days ie. the maximum length between the introduction of the epidemic until the end of the simulation.

Figure 1 plots the detection time in days versus the number of false positives for five different p-value thresholds used in both the standard algorithm and WSARE. These values were generated by taking the average over 30 runs of the simulation. Included in this plot are errorbars in both dimensions.

4.3 Results from Simulated Data

These results indicate that for p-value thresholds above 0.005, the detection time for WSARE is significantly smaller than that of the standard algorithm. On the other hand, as the p-value threshold decreases, the detection time for WSARE is somewhat worse than that of the standard algorithm. However, choosing an extremely low threshold would be unprofitable since all anomalies except those at an unusually high significance level would be ignored. For example, using a threshold of 0.005 corresponds to a 99.5% significance level.

The results also demonstrate that WSARE signals more false positives for higher p-value thresholds. While this behaviour is not desirable, it is tolerable since the number of false positives produced by WSARE differs by a small amount from the count generated by the standard algorithm. In this particular graph, there are at most 2 more false positives identified by WSARE that were not identified by the standard algorithm.

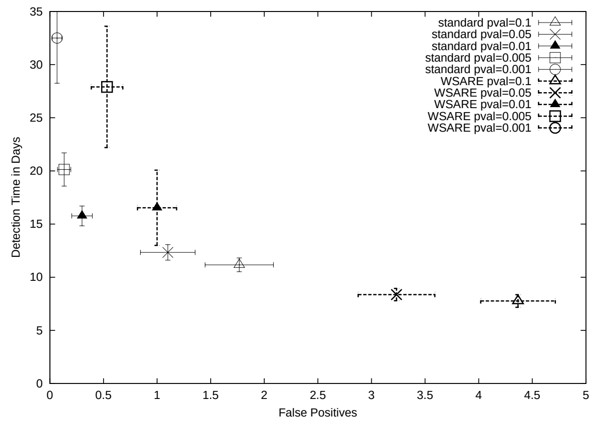

Figure 1: Detection Time vs False Positives

We would also like to show some of the rules learned by WSARE. The rules below were obtained from one of the result generating simulations.

```
### Rule 1: Sat Day97 (daynum 97, dayindex 97)
SCORE = -0.00000011 PVALUE = 0.00249875
 33.33% ( 16/ 48) of today's cases have Age Decile = 5 and Gender = Male
  3.85% (  7/182) of other cases have Age Decile = 5 and Gender = Male

### Rule 2: Tue Day100 (daynum 100, dayindex 100)
SCORE = -0.00001093  PVALUE = 0.02698651
 30.19% ( 16/ 53) of today's cases have Age Decile = 5 and Col2 less than 25
  6.19% ( 12/194) of   other cases have Age Decile = 5 and Col2 less than 25
```

In rule 1, WSARE demonstrates that it is capable of finding the target demographic group that Epidemic0 infects. This rule proves to be significant above the 99% level. On the other hand, Rule 2 discovers something that was not deliberately hardcoded into Epidemic0. Rule 2 states that on Day 100, there is an unusually large number of cases involving people in their fifties that were all in the left half of the grid. Since we had designed the people in the simulation to interact with places that are in close geographic proximity to their homes, we suspected that the locality of interaction of infected individuals would form some spatial clusters of ER cases. Upon further inspection of the log files, we discovered that 12 of the 16 cases from the current day that satisfied this rule were in fact caused by Epidemic0. This example illustrates the capability of WSARE to detect significant anomalous patterns that are completely unexpected.

4.4 Results from Real ER data

We also ran WSARE on an actual ER data collected from hospitals in a major US city. This database contained approximately 70000 records collected over a period of 505 days. Since we are looking at historical data, we need to use FDR to determine which of the p-values are significant. The results are shown below with α for FDR equal to 0.1.

```
### Rule 1: Tue 05-16-2000 (daynum 36661, dayindex 18)
SCORE = -0.00000000  PVALUE = 0.00000000
 32.84% ( 44/134) of today's cases have Time Of Day4 after 6:00 pm
 90.00% ( 27/ 30) of   other cases have Time Of Day4 after 6:00 pm

### Rule 2: Fri 06-30-2000 (daynum 36706, dayindex 63)
SCORE = -0.00000000  PVALUE = 0.00000000
 19.40% ( 26/134) of today's cases have Place2 = NE and Lat4 = d
  5.71% ( 16/280) of   other cases have Place2 = NE and Lat4 = d

### Rule 3: Wed 09-06-2000 (daynum 36774, dayindex 131)
SCORE = -0.00000000  PVALUE = 0.00000000
 17.16% ( 23/134) of today's cases have Prodrome = Respiratory
and age2 less than 40
  4.53% ( 12/265) of other cases have Prodrome = Respiratory
and age2 less than 40

### Rule 4: Fri 12-01-2000 (daynum 36860, dayindex 217)
SCORE = -0.00000000  PVALUE = 0.00000000
 22.88% ( 27/118) of today's cases have Time Of Day4
after 6:00 pm and Lat2 = s
  8.10% ( 20/247) of   other cases have Time Of Day4
after 6:00 pm and Lat2 = s

### Rule 5: Sat 12-23-2000 (daynum 36882, dayindex 239)
SCORE = -0.00000000  PVALUE = 0.00000000
 18.25% ( 25/137) of today's cases have ICD9 = shortness of breath
and Time Of Day2 before 3:00 pm
  5.12% ( 15/293) of   other cases have ICD9 = shortness of breath
and Time Of Day2 before 3:00 pm

### Rule 6: Fri 09-14-2001 (daynum 37147, dayindex 504)
SCORE = -0.00000000  PVALUE = 0.00000000
 66.67% ( 30/ 45) of today's cases have Time Of Day4 before 10:00 am
 18.42% ( 42/228) of   other cases have Time Of Day4 before 10:00 am
```

Rule 1 notices that there are fewer cases after 6:00 pm quite possibly due a lack of reporting by some hospitals. Rule 6 correctly identifies a smaller volume of data being collected before 10:00 am on Day 504. Since Day 504 was the last day of this database, this irregularity was the result of the database being given to us in the morning.

We are currently beginning the process of using input from public health officials of the city concerned to help us validate and measure WSARE's performance.

5 Conclusion

WSARE has been demonstrated to be successful at identifying anomalous patterns in the data. From our simulation results, WSARE has significantly lower detection times than a standard detection algorithm provided the p-value threshold is not at at extremely low level. This condition should not be a problem since most anomalies are reported at a significance level of 95% or 99%, corresponding respectively to p-value thresholds of 0.05 and 0.01. WSARE also has a slightly higher false positive rate than the standard algorithm. However, this difference was shown to be about 2 more false positives in the worst case for our particular simulation. Future research involves making this computationally intensive algorithm more efficient.

We believe the three main innovations in this paper are:

1. Turning the problem of "detect the emergence of new patterns in recent data" into the question "is it possible to learn a propositional rule that can significantly distinguish whether records are most likely to have come from the recent past or longer past?"

2. Incorporating several levels of significance tests into rule learning in order to avoid several levels of overfitting caused by intensive multiple testing

3. Examining the interesting domain of early outbreak detection by means of machine learning tools

References

[Benjamini and Hochberg, 1995] Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B.*, 57:289–300.

[Bonferroni, 1936] Bonferroni, C. E. (1936). Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62.

[Eskin, 2000] Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the 2000 International Conference on Machine Learning (ICML-2000)*, Palo Alto, CA.

[Goldenberg, 2001] Goldenberg, A. (2001). Framework for using grocery data for early detection of bio-terrorism attacks. Master's thesis, Carnegie Mellon University.

[Good, 2000] Good, P. (2000). *Permutation Tests - A Practical Guide to Resampling Methods for Testing Hypotheses.* Springer-Verlag, New York, 2nd edition.

[Lane and Brodley, 1999] Lane, T. and Brodley, C. E. (1999). Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2:295–331.

[Maxion and Tan, 2001] Maxion, R. A. and Tan, K. M. C. (2001). Anomaly detection in embedded systems. Technical Report CMU-CS-01-157, Carnegie Mellon University.

[Miller et al., 2001] Miller, C. J., Genovese, C., Nichol, R. C., Wasserman, L., Connolly, A., Reichart, D., Hopkins, A., Schneider, J., and Moore, A. (2001). Controlling the false discovery rate in astrophysical data analysis. Technical report, Carnegie Mellon University.

[Wagner et al., 2001] Wagner, M. M., Tsui, F. C., Espino, J. U., Dato, V. M., Sittig, D. F., Caruana, R. A., McGinnis, L. F., Deerfield, D. W., Druzdzel, M. J., and Fridsma, D. B. (2001). The emerging science of very early detection of disease outbreaks. *Journal of Public Health Management Practice*, 7(6):51–59.