Anomaly Detection in a Mobile Communication Network

Alec Pawling, Nitesh V. Chawla, and Greg Madey
University of Notre Dame
{apawling,nchawla,gmadey}@cse.nd.edu

Abstract

Cell phone networks produce a massive volume of service usage data which, when combined with location data, can be used to pinpoint emergency situations that cause changes in network usage. Such a change may be the results of an increased number of people trying to call friends or family to tell them what is happening or a decrease in network usage caused by people being unable to use the network. Such events are anomalies and managing emergencies effectively requires identifying anomalies quickly. This problem is difficult due to the rate at which very large volumes of data are produced. In this paper, we discuss the use of data stream clustering algorithms for anomaly detection.

Contact:
Alec Pawling
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556

Tel: 1-574-631-7596
Fax: 1-574-631-9260
Email: apawling@cse.nd.edu

Key words: Anomaly detection, data clustering, data mining

Anomaly Detection in a Mobile Communication Network

Alec Pawling, Nitesh V. Chawla, and Greg Madey

1 Introduction

The city of Baltimore currently uses location information produced by cellular phones to monitor traffic conditions, and the state of Missouri is considering a similar statewide program [7]. The goal of such systems is to reduce traffic congestion and fuel usage by diverting drivers around slow moving areas. There is a significant advantage in using cellular phones: there is no installation or maintenance required for the physical sensors. A cellular phone network may also be used to quickly detect a potential crisis via anomalies in the network as a whole, including call and movement patterns. [9]. This paper will examine the problem of detecting these anomalies using call volumes.

Anomalies need to be detected very rapidly, so that crises can be identified and handled effectively as they arise. Additionally, the volume of data being produced by the network is massive: the network is constantly producing new data as calls are made, text messages are sent, and cell phones move. This data must be analyzed in real time. To further complicate the matter, the way in which people use the available services is likely to change over time, so the system must be dynamic. We aim to remain within the data stream model [1]. Intuitively, a data stream is a sequence of data items that arrive at such a rapid rate that it is only feasible to operate on a small portion of the data. As each data item is seen, it must be either incorporated into a summary that requires a small amount of space or it must be discarded, in which case it cannot be retrieved. The data stream model imposes two algorithmic limitations: each item in the dataset may only be read once, in a predefined order, and memory usage must be sub-linear—typically polylogarithmic

Distinguishing between changes that are merely due to the dynamic nature of the system and anomalies is a difficult problem. Statistical process control is one method of approaching this problem. Statistical process control [2] aims to distinguish between "assignable" and "random" variation. Assignable variations are assumed to have low probability and indicate some anomaly in the underlying process. Random variations, in contrast, are assumed to be quite common and to have little effect on the measurable qualities of the process. These two types of variation may be distinguished based on the difference in some measure on the process output from the mean, μ, of that measure. The threshold is typically some multiple, l, of the standard deviation, σ. So, if the measured output deviates from the mean by less than $|l\sigma|$ the variance is considered random, otherwise it is assignable. Figure 1 shows the range of random variation (with a threshold of 3σ) of telephone usage in a cellular network over the hours of the day. Note that not only is the call volume different depending on the time of the day but so does the range of random variance.

Our solution must be robust to these variations as well as underlying changes in the way in which people use the network over time. These are forms of concept drift—an underlying change in the process producing the data stream [10]. Concept drift may or may not be periodic. Our data contains both (see figure 2). Much of the work in handling concept drift for clustering has focused on using a sliding window, in which the example in the last t time steps are considered in the current set of clusters. This has been shown to produce "meaningless" clusters [6].

2 Related Work

One method for anomaly detection is data clustering. The goal of clustering is to group similar data items together. The concept of similarity is often defined by a distance metric (often Euclidean distance), and anomalies are, intuitively, the data items that are far from all other data items.

Jain, Murty, and Flynn [5] thoroughly review data clustering. There are three major types of clustering algorithms: partitional, agglomerative, and incremental. Partitional algorithms, such as k-means or expectation

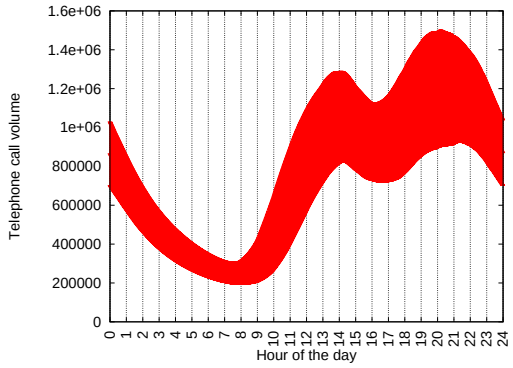


Figure 1: The average and standard deviation of the network load of the telephone service for each minute of the day.

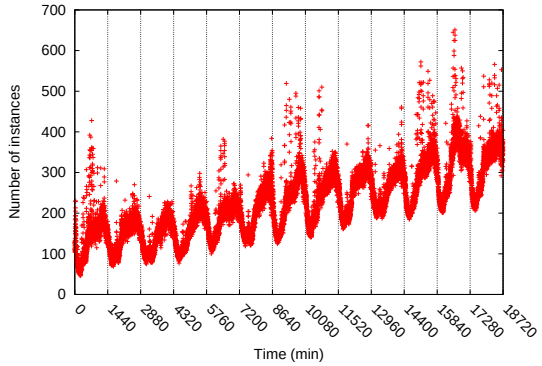


Figure 2: GPRS usage for each minute of a 12 day period.

maximization produce a single partition of the dataset and often have a random initial state. Agglomerative algorithms are a bottom-up approach. Initially, each example is in a unique cluster. A set of hierarchical partitions of the data is formed by iteratively merging neighboring clusters. Agglomerative algorithms tend to be expensive. Incremental clustering considers each example once, immediately deciding to place it in a existing cluster or creating a new cluster. These algorithms tend to be fast, but are also often order dependent.

Portnoy *et al.* [8] use the leader algorithm [4], a simple incremental clustering algorithm for intrusion detection (an application of anomaly detection). Each cluster is defined by a single data item—the first item assigned to the cluster—and data items within a user specified distance, d, of the defining item. The algorithm proceeds as follows: for each data item, find the closest cluster. If the distance between the item and the defining item of the cluster is greater than d, the item defines a new cluster. The algorithm itself fits into the data stream model when the number of clusters found is sub-linear with respect to the number of data items; however, in order to handle arbitrary distributions, Portnoy *et al.* normalize the data set using z-score, which is computed by

$$v_i' = \frac{v_i - \bar{v_i}}{\sigma_i}. \tag{1}$$

Unfortunately, this requires multiple passes over the data. Another problem with this approach is that the proper value of the distance threshold is not intuitive, is fixed over all clusters, and cannot change as the data stream evolves.

3 A Hybrid Clustering Algorithm

Hybrid clustering algorithms consist of two levels: often the first level reduces the size of the data set to improve performance for the final clustering [3]. We use a hybrid clustering algorithm in a different way. We

use an offline algorithm in the first level to establish clusters and leader clustering combined with statistical process control as the second level.

We investigate an algorithm that combines the leader and the standard k-means algorithms. The leader algorithm forms the foundation of this algorithm, using Euclidean distance, and the distance threshold is based on statistical process control. The k-means algorithm is used to create each cluster such that there are enough examples to produce a reasonable standard deviation. In addition to the cluster information, we also keep a set of outliers.

The algorithm requires three parameters, the minimum number of examples required to form a cluster, s, the number of clusters to produce each time when applying k-means, k, and the threshold, t (the distance threshold becomes $t|\vec{\sigma}_i|$). The algorithm first adds examples to the outlier set until there are ks examples, at which time, k-means is applied to the outlier set. Any clusters with more than s members are kept and the s members establish the initial feature means and standard deviations for the clusters. The examples in the remaining clusters are returned to the outlier set. Any subsequent examples are either placed in the nearest cluster, if their distance from the cluster mean is within t times the standard deviation of all features, or added to the outlier set. k-means clustering is applied to the outlier set whenever there are ks examples in the set.

4 Experimental Setup

4.1 Dataset

We use a dataset generated from a database of real world mobile communication information. The data set covers 12 days of network usage of 16 services. For each service used during the 12 days, there is a record in the database consisting of the initiation time (to the second), the duration (in minutes) and the service name. Since the duration is in minutes, we generate an example for each minute in the 12 day period. We step through the records in the database in increasing order of time. For each record, we increment a counter of the number active instances of the service and place an object containing the service name and termination time in a priority queue. When the minute of the current record is greater than the minute of the previous, all expired objects are removed from the priority queue, and the appropriate service instance counts are decremented. The record consisting of the month, day, date, hour, minute, and list of service instances counts is then added to the dataset.

Since the data set is for such a short period of time, we pruned the month, day, and date from the dataset. Additionally, most of the services are rarely used. These features were removed, leaving 6 service features, in addition to the hour and minute.

4.2 Clustering

We clustered the data using three algorithms: expectation maximization, leader (with z-score normalization), and incremental hybrid. We used Weka's [11] implementation of expectation maximization with 10 fold cross validation to determine the number of clusters. We ran the leader algorithm using distance thresholds $d = 1, 2, 3$ and considered clusters with fewer than $10\%, 5\%$, and 1% of the dataset to be anomalies. For the incremental hybrid algorithm, we used $l = 1, 3$ and $k = 5, 10, 20, 30$.

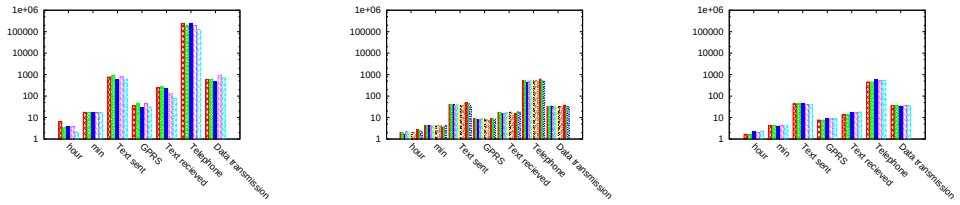
We used sum squared error to evaluate quality of the clusters produced by expectation maximization and incremental hybrid. Additionally, we compare the size of the clusters produced by these two algorithms. Finally, we looked for consistency in the anomalies detected by the system.

5 Results

Table 1 shows the number of clusters produced by each method and figure 3 shows the sizes of clusters produced by expectation maximization, the hybrid algorithm with $k = 20$ and $l = 1, 3$ in terms of the standard deviation of each feature for each vector. Interestingly, the hybrid algorithm produces smaller clusters than expectation maximization, regardless of which algorithm produces more clusters.

Table 1: Number of clusters produced for each method.

Algorithm	Parameters		Number of clusters
EM			5
Leader		$d = 1.0$	542
		$d = 2.0$	73
		$d = 3.0$	23
Hybrid	$l = 1$	$k = 5$	19
		$k = 10$	11
		$k = 20$	11
		$k = 30$	7
	$l = 3$	$k = 5$	2
		$k = 10$	4
		$k = 20$	5
		$k = 30$	3



(a) EM (b) Hybrid ($l = 1, k = 20$) (c) Hybrid ($l = 3, k = 20$)

Figure 3: Standard deviation of clusters

Figure 4 shows the sum squared error for expectation maximization and all trails of the hybrid algorithm. For our trials, the worst cases for hybrid perform comparably to expectation maximization, and several trials perform an order of magnitude better.

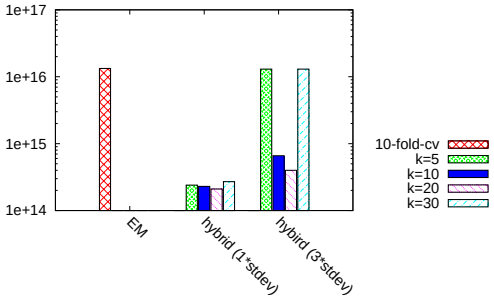


Figure 4: Sum squared error for expectation maximization and hybrid clustering

Table 2 shows the number of anomalies detected by the hybrid algorithm. The leader algorithm performs as expected, with the number of anomalies detected decreasing when either, or both, of the parameters decreases. The same trend seems to hold for the hybrid algorithm for the threshold parameter, but it is not clear how k affects the sensitivity.

6 Conclusion

The leader algorithm is a promising method; however, since the appropriate threshold for our application is not intuitive, it may not be the best method. The hybrid method using k-means does not seem promising;

Table 2: Number of anomalies detected by the hybrid algorithm

Parameters		Number of clusters
$l = 1$	$k = 5$	101
	$k = 10$	122
	$k = 20$	552
	$k = 30$	432
$l = 3$	$k = 5$	25
	$k = 10$	41
	$k = 20$	263
	$k = 30$	137

however, there may be other clustering algorithms that would be more appropriate.

6.1 Future Work

Hierarchical algorithms can be used to partition a dataset at a certain level of dissimilarity. Hierarchical algorithms such as single-link and complete link are completely deterministic and may be more suitable for our application than k-means, which relies on a random initial state. We plan to pursue this, along with examining how the the clusters change over time. This requires more data and is important because we need to ensure that the number of clusters in not growing too quickly.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16, June 2002.

[2] C. Bicking and F. M. Gryna, Jr. *Quality Control Handbook*, chapter Process Control by Statistical Methods, pages 23–1—23–35. McGraw Hill, 1979.

[3] E. Y. Cheu, C. Keongg, and Z. Zhou. On the two-level hybrid clustering algorithm. In *International Conference on Artificial Intelligence in Science and Technology*, pages 138–142, 2004.

[4] J. A. Hartigan. *Clustering Algorithms*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 1975.

[5] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[6] E. Keogh and J. Lin. Clustering of time series subsequences in meaningless: Implications for past and future research. In *Knowledge and Information Systems*. Springer-Verlag, 2004.

[7] D. A. Lieb. Tracking cell phones for real-time traffic data. *Wired*, 2005.

[8] L. Portnoy, E. Eskin, and S. Stolfo. Intrusion detection with unlabeled data using clustering. In *ACM Workshop on Data Mining Applied to Security*, 2001.

[9] T. Schoenharl, G. Madey, G. Szabó, and A.-L. Barabási. WIPER: A multi-agent system for emergency response. In *Proceedings of the 3rd International ISCRAM Conference*, 2006.

[10] A. Tsymbal. The problem of concept drift: Definitions and related work. Technical Report TCD-CS-2004-15, Trinity College Dublin, 2004.

[11] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufman, second edition, 2005.