Real-time Knowledge Discovery and Dissemination for Intelligence Analysis

Bhavani Thuraisingham,
Latifur Khan,
Murat Kantarcioglu,
Sonia Chib
The University of Texas at Dallas
lkhan@utdallas.edu
bhavani.thuraisingham@utdallas.edu
muratk@utdallas.edu
sonia.chib@student.utdallas.edu

Jiawei Han
University of Illinois at Urbana Champagne
hanj@cs.uiuc.edu

Sang Son
University of Virginia
son@virginia.edu

Abstract

This paper describes the issues and challenges for real-time knowledge discovery and then discusses approaches and challenges for real-time data mining and stream mining. Our goal is to extract accurate information to support the emergency responder, the war fighter, as well as the intelligence analyst in a timely manner.

1. Introduction

Data mining/knowledge discovery is the process of posing various queries and extracting useful and often previously unknown and unexpected information, patterns, and trends from large quantities of data, generally stored in databases. These data could be accumulated over a long period of time or they could be large data sets accumulated simultaneously from heterogeneous sources such as different sensor types. The goals of data mining include improving marketing capabilities, detecting abnormal patterns, and predicting the future based on past experiences and current trends. There is clearly a need for this technology for many applications in government and industry.

Much of the focus on data mining has been for analytical applications. However there is a clear need to mine data for applications that have to meet timing constraints. For example, a government agency may need to determine whether a terrorist activity will happen within a certain time or a financial institution may need to give out financial quotes and estimates within a certain time. Consider for example a medical application where the surgeons and radiologists have to work together during an operation. Here, the radiologist has to analyze the images in real-time and give inputs to the surgeon. In the case of military applications, images and video may arrive from the war zone. These images have to be analyzed in real-time so that advice is given to the war fighter. In the case of counter-terrorism applications, the system has to analyze the data about the passenger from the time the passenger gets ticketed until the plane is boarded, and give proper advice to the security agent. For all of these applications there is an urgent need to mine the data and extract knowledge in real-time. Therefore, we need tools and techniques for real-time data mining. The challenge is to determine which data to analyze and which data to discard for future analysis in non real-time.
This paper describes approaches for real-time knowledge discovery. In particular the following techniques are discussed.

Real-time knowledge discovery techniques including real-time clustering, machine learning, association rule mining, outlier and trend detection.

Real-time stream mining techniques including clustering evolving streams and multidimensional stream cube analysis.

The organization of this paper is as follows. A motivating scenario for real-time data mining is discussed in section 2. Some technical issues in real-time data mining are given in Section 3. Adapting data mining techniques to meet real-time constraints including multiple real-time data mining algorithms is described in Section 4. Stream data mining, which is an important aspect of mining sensor data in real-time, is discussed in Section 5. The paper is concluded in section 6.

2. Motivational Scenario

There is a strong need for coordinated activities between multiple organizations (such as Army, Navy, Air force; Coalition members such as US, UK and Australia, or Health organizations such as CDC: Center for Disease Control, HRSA: Human Resources and Services Administration, and Hospitals) in the crisis management domain for working with one another to determine the enemy locations at a specific time or to monitor and manage the spread of infectious diseases. Here, organizations such as local, state and federal governments or coalition members have to work together and integrate the diverse sources of information.

For time-critical applications such as crisis management, there are two major components. One is data sharing and the other is decision support. The data sharing component consists of data integration where data from structured and unstructured sources have to be integrated. The integrated data has to be analyzed using data mining and data warehousing techniques. For real-time applications, active data management techniques will ensure that triggers are enforced on the data so that incidents are propagated in a timely manner. Finally the integrated data, the results obtained from activating triggers as well as the results from data mining are collected and analyzed by a decision support system to give advice to the incident commanders and decision makers. One can carry out decision support analysis using partial data in which case the analysis as well as integration will be carried out incrementally. This is the bottom up approach where it is assumed that not all of the data is available at one time. The top-down approach integrates all of the data first and then carries out the mining and subsequently decision support analysis. For both approaches, the organizations may enforce their security policies including confidentiality policies, privacy policies, dissemination policies and trust polices. Real-time data sharing and security have conflicting requirements. The challenge is to enforce various policies and at the same time share the data so that real-time decision making can be carried out.

In defense applications, organizations such as Air Force, Navy, Army, Central Intelligence Agency and the National Security Agency have to share intelligence and military data. Each organization may enforce its own policies. However the data has to be sent to the war fighter in real-time to make life and death decisions. For example, as a result of the analysis the outcome may be that the enemy forces will be in Location X between 6 and 6:30pm. This information has to be sent to the war fighter so that he has enough time to attack the enemy if need be. In addition to techniques for enforcing policies we also need to make tradeoffs between security and timely processing. The soldiers and the analysts have to work together to get timely and accurate information. One issue with real-time analysis is that we may not be able to get the complete and accurate picture as well as meet timing constraints. In such a situation is partial information better than no information? This depends on the situation under consideration. If the plan is to attack the enemy, the enemy location information will be extremely valuable even if it is partial information. However if the plan is to attack the enemy, then we will need a complete and accurate picture of the situation.

We are investigating secure data sharing across agencies for the Air Force Office of Scientific Research [THUR06]. However our research is focusing on the non real-time data integration and data analysis aspects. While this is very important, there is also a strong need to provide a dependable real-time analysis environment so that the emergency responder as well as the war fighter gets the right information at the right time.

3. Technical Issues

To study effective real-time data mining and data stream mining, the first issue is to examine what kinds of patterns and knowledge that should be mined from huge sets of data. In general, the kinds
of knowledge to be mined can be categorized into the following classes: (1) multidimensional summary, which characterizes and compares summary information of different groups of data in multidimensional space; (2) clustering, which automatically partitions data into multiple groups based on the similarities among data; (3) classification, which builds up predictive models based on the training and testing data sets; (4) frequent pattern mining for association and correlation analysis, which mines frequent patterns for discovering association rules and correlation relationships among data; (5) mining changes, trends, sequential patterns and evolutions with time in the data sets and data streams; and (6) outlier detection, which finds singular or small groups of data that deviate substantially from the properties of the majority of the data. Both real-time and stream data mining involve the accomplishment of one or more of the above data mining tasks.

In recent research, we have performed in-depth investigation on real-time and data stream mining for multidimensional analysis, classification and cluster analysis. In this paper we will discuss some aspects of real-time and data stream mining in the following two major directions: (1) exploration of real-time and data stream mining in several strategically important applications, and (2) development of real-time and data stream mining methods for trend and outlier detection.

4. Real-time Knowledge Discovery Techniques

We believe that the most critical issue in real-time data mining is that at any moment in time the algorithm should be able to stop and produce intermediate output with an indication of how close to optimum the intermediate output is. This will enable a real-time data miner to determine how much weight to put on the conclusions drawn from the intermediate output. The mining should be done efficiently, with high scalability, and incrementally when the new data sets being added into the system or the existing data sets being modified dynamically. Stream data mining adds an additional but critical constraint that a data stream in most cases can be scanned only once because of the huge size and the sequential nature of a data stream, it is unrealistic to repeatedly scan or randomly access of a data stream. Therefore, in our discussion, we will focus on how to develop efficient, scalable, and incremental data mining methods for each knowledge discovery task listed above and discuss how to extend the methods to the data stream environments with the development of single-scan mining method.

In this section, we will discuss each of the data mining tasks and how to extend the major methods for real-time data mining, with an emphasis on efficiency, scalability and incremental mining methods.

4.1. Real-time Multidimensional Summary

For the first data mining task, multidimensional summary, which characterizes and compares summary information of different groups of data in multidimensional space, data cube has been a popular approach for multidimensional aggregation and efficient methods have been developed for computing sparse, dense, iceberg, and closed cubes (ZDN97, BeRa99, HPDW01, XHLW03, XHSL06). Online analytical processing can also be performed with multidimensional databases of rather high dimensions by partial materialization (LHG04). These methods can be generalized to semi-structured databases as well. Moreover, incremental computation can be performed on data cubes with the above mentioned methods. Thus the above computational methods set up a foundation for efficient and incremental computation of multidimensional databases.

4.2. Real-time Clustering

In clustering, the goal is to partition a dataset so that similar points belong to the same cluster and dissimilar points belong to different clusters. One of the most widely used clustering criteria is the squared error distortion, where the goal is to minimize the average squared distance from a point to its nearest center.

While many clustering algorithms are available, the real-time nature of the problem calls for algorithms that make measurable progress towards the optimum solution. The reason is that if an individual is to make a real-time decision based on a data mining algorithm that has not run to completion, then it is important to know how far the algorithm is from the final solution. This will enable the decision-maker to determine how much weight to put on the conclusion drawn from the incomplete run of the data mining algorithm. Ideally, the final solution is the optimum solution, but in the case of clustering, no efficient algorithm is known for finding the optimum solution. On the other hand, constant factor approximation algorithms are known. For example,
the algorithm given in (AGKM04) produces a solution that is no worse than nine times the optimum squared error distortion. Another desirable feature of this algorithm is that it begins with a solution that is at most 2n times the optimum, where n is the number of points and, with each successive iteration, the quality of the solution improves by a \((1- \frac{1}{k})\) factor, where \(k\) is an input parameter and \(k\) is the number of clusters. It can be shown that after roughly \(O((k/\log n) \log n)\) steps that the cost of the solution is at most nine times the optimum. But, more relevant to this context, if a real-time decision is made after the algorithm has completed i iterations, then the cost of the solution is \((1- \frac{1}{k})^i(2n)\).

A more detailed study of clustering algorithms that make measurable progress towards the optimum is needed. Further, the problem of clustering a streaming graph in real-time is quite unexplored. We also need to investigate extensions of such algorithms to the data stream setting in the case where the goal is to find the best clusters (GMMO00) as well as the best evolving clusters (AHKWY03).

4.3. Real-time Machine Learning

In the classification problem, one is given a collection of labeled data, e.g., positive and negative examples, and the goal is to learn a function that distinguishes the two classes. Standard examples of machine learning algorithms include learning halfspaces, decision trees and other function classes (Mit97).

In the real-time version of this problem, the machine learning algorithm must make measurable progress towards the optimum solution. A consistent learning algorithm, i.e., one that maintains a hypothesis that is always consistent with the data seen so far, is one example of an algorithm that is always making progress towards the target concept. For instance, in the context of learning halfspaces, the algorithm of (MT94) always cuts the number of consistent concepts in half.

We need to further investigate such algorithms that make steady progress towards the final target concept so that we can quantify how informed the real-time decision is. In this setting, the number of consistent concepts may serve as a measure of how much weight to put on the decision.

4.4. Association Rules/Frequent Itemsets in Real Time

Association rules uncover patterns of the form \(ABC \rightarrow D\), with the interpretation that whenever events A and B and C occur, D also occurs. A common first step employed in the discovery of such rules is the identification of maximal frequent itemsets, subsets of values that occur frequently in a dataset. The Apriori algorithm (AS94) gives one way of identifying the maximal frequent itemsets.

One of the limitations of this algorithm in the real-time context is that the algorithm does not make measurable progress. In other words, if all the frequent itemsets are of length \(m\), the algorithm does not produce any itemsets until it has iterated through all \(2m-1\) subsets of the frequent itemset that are of length \(1, 2, 3, \ldots, m-1\). Such an approach may not be suitable in the real-time context since no useful output may be produced when needed.

Alternate algorithms do exist that may be more appropriate. For example, (FK96) give an algorithm that outputs frequent itemsets with boundable delay in between itemsets. In particular, if there are \(s\) maximally frequent itemsets and minimally infrequent itemsets, then the algorithm spends time \(O(s(\log s))\) time in between itemsets. No better bound on the delay is known to this date.

We need to study other boundable delay algorithms for the problem of finding frequent itemsets, closed patterns, and constraint-based mining (NLHP98). For an application that makes decisions in real-time, boundable delay algorithms have the advantage of making as much information available to the individual as possible.

4.5. Real-time Trend Detection

For the fifth data mining task, mining changes, trends, sequential patterns and evolutions with time in the data sets, there have been many studies as well. For example, for mining sequential patterns, efficient and scalable algorithms, such as GSP (AS96) PrefixSpan (Pei+04), SPADE (Zaki01), and CloSpan (YaHa03), have been developed and shown high performance in large data sets. Methods for incremental mining of such patterns, such as IncSpan (CYH04), have been developed as well. We will further develop such methods for mining changes, trends and evolutions in real-time for large datasets.

An alternate problem is to cluster real-time sequences of activity, e.g., movement through a sensor network, into a collection of Markov models, where the vertices correspond to sensors and the weight on edge \(u,v\) indicates the probability that there is movement from vertex \(u\) to \(v\). These models are useful both for understanding normal patterns of activity as well as triggering alarms in the event of an unusual real-time sequence.
The problem of inferring such Markov models of maximum likelihood has been studied in the past (KMBRT04). However these algorithms have many limitations – the most notable is that we do not currently know how to quantify the progress that these algorithms make as they infer the collection of Markov models. As indicated, evaluating this progress is important for a real-time miner.

4.6. Real-Time Outlier Detection

Finally, for the last data mining task, outlier detection, which finds singular or small groups of data that deviate substantially from the properties of the majority of the data. It is critically important to find outliers and alarming incidents in many applications, such as homeland security, accident, disasters (fire, explosion, etc.). There have been studies on mining outliers and local outliers in large data sets (KnNg98, SLZO1, BKNS00, JTHW06). In general, the outlier detection method needs to be further developed for different applications. We will devote more effort to investigate outlier detection at real-time, especially for real-time outlier detection for spatiotemporal data sets.

5. Stream Data Mining

Recent emerging applications call for investigation of stream data, where data takes the form of continuous, ordered, potentially infinite data streams, as opposed to finite, statically stored data sets. Huge volumes of dynamically changing data are in the form of data streams, including time-series data, scientific and engineering data, and data produced in other dynamic environments, such as power supply, network traffic, stock exchange, telecommunication, Web clicking, weather or environment monitoring, and so on.

Analysis of stream data poses great challenges to data mining research, due to the unique features of data streams, such as huge (and possibly infinite) volume, dynamically changing behavior, flowing in-and-out in a fixed order, allowing only one scan of the entire data stream, and demanding fast (often real-time) responses. Moreover, a lot of stream data resides at the primitive abstraction level, and it is necessary to perform multi-level, multi-dimensional analysis on such data to find interesting patterns at appropriate levels of abstraction and with appropriate dimension combinations.

Stream data modeling, stream data management, and continuous stream query processing are under popular investigation (Babc+02, BaWi01, GKMS01, GMMO00, GKS01b, DGGR02). Moreover, mining data streams, including classification of stream data (DoHu00, HSD01), clustering data streams (GMMO00, Ocal+02), and finding frequent patterns in data streams (MaMo02) are also under intensive study. Nevertheless, most existing studies have not paid enough attention to one critical fact: most data streams reside at rather low level of abstraction and are multi-dimensional in nature, whereas most analysts are interested in finding characteristics, rules, unusual patterns, and dynamic changes (such as trends and outliers) at relatively high levels of abstraction and in multi-dimensional space.

Let’s examine an example. Suppose that a Web server receives a huge volume of online Web service requests, in the form of data streams. Usually, such stream data resides at rather low level, consisting of time (down to seconds), Web page address (down to concrete URL), user IP address (down to detailed machine IP address), etc. However, an analyst may be interested in (1) finding trends happening in the data streams at certain high levels of abstraction, such as “the Web clicking traffic in North America on Mideast in the last 15 minutes is 40% higher than the last 24 hours on average”, (2) detecting network intrusions and traffic bursting, by multi-dimensional clustering or frequent pattern analysis, and (3) dynamic modeling the Web traversal behavior for certain suspicious groups of users to protect network security and swiftly react to cyber-attacks. On the other hand, from the point of view of a Web analysis provider, given a large volume of fast changing Internet traffic streams, and with limited main memory and computational power, it is only realistic to analyze the changes of Web usage at certain high levels, discover unusual events and patterns, and drill down to some more detailed levels for in-depth analysis, when needed. Moreover, it is critical to have such analysis performed automatically or semi-automatically, and in almost real time, in order to make timely responses.

All these applications demand for the development of mechanisms for multi-dimensional analysis and mining of changes, trends, and unusual patterns of data streams, with low cost and fast response time. In our recent studies, we have developed a data stream cube architecture for multidimensional stream data analysis (Chen+02, Han+05). Moreover, we have studied how to cluster dynamic data streams (AHWY03, AHWY04, AHWY05), classify data streams (WFYH03, AHWY04b, AHWY06), and finding frequent patterns in data streams (LLHH06). We briefly introduce our efficient methods for effective online
mining of data streams and outline further research issues.

5.1. Multidimensional data stream analysis using stream cubes

For multi-dimensional analysis of data streams, we proposed a stream cube architecture (Chen+02, Han+05) for efficient and effective multi-dimensional regression analysis in time series data streams. The major challenges for construction of a stream data cube is the huge volume of data, and the potentially even huger size of stream data cube, and the incremental update of such a stream cube. The proposed architecture has the following features: (1) tilted time window for registering recent data at fine granularity, and more distant ones at coarser granularity; (2) two critical layers: a minimal interest layer and an observation layer, which are used to save minimal number of cuboids whereas achieving the power of high-level observation and low-level drilling, and (3) an efficient data structure for partial computation of data cubes, which uses an H-tree structure (HPDW01) to partially compute data cubes, saving storage space pre-computation efforts, and on-line computation time. The stream data cubes so constructed are much smaller than those constructed from the raw stream data but will still be effective for many multi-dimensional stream data analysis tasks. This architecture has been shown its efficiency and scalability in our construction of the MAIDS system (Mining Alarming Incidents in Data Streams) (Cai+04) and its subsequent experiments with various kinds of stream data. The architecture will be further developed in this project, especially for the analysis of correlations, trends and outliers in the context of data streams, to facilitate multi-dimensional mining of various kinds of patterns in data streams.

5.2. Clustering evolving data streams

For efficient clustering of stream data, typical clustering methods proposed in machine learning, statistics and data mining communities (ABKS99, KHK99, Agga+99, AGGR98, BFR98, GRS98, NgHa94, JaDu88, KaRo90, ZRL96) do not work since they require multiple scans of a database. For single-scan clustering of data streams, new algorithms such as (GMMO00, Ocal+02) have been proposed. Unfortunately, these stream clustering methods cannot be used to discover stream evolution behavior since they suffer from two deficiencies.

First, they assume that the clusters are to be computed over the entire data stream. Although such an assumption may be useful in some applications, a data stream should often be viewed as an infinite process consisting of data which continuously evolves with time. As a result, the underlying clusters may also change considerably with time. The nature of the clusters may vary with both the moment at which they are computed as well as the time horizon over which they are measured. For example, a data analyst may wish to compare clusters occurring now with that in the last hour, day, week, or month. These clusters could be considerably different. Without considering multiple tilted time windows, it is impossible to perform such comparison to find the dynamics and evolution of data streams.

Second, they assume that incremental clustering can be performed by maintaining only a small number of existing clusters (such as k in the k - median or k -means method). That is, the level of granularity to be preserved across windows is at the level of macro-clusters. However, with the fast evolving streaming data, it is obvious that some objects originally belongings to cluster C1 may need to be moved to another cluster C2 or even form a new, independent cluster C3. Since the macro-cluster level does not register any detailed information about such groups of objects, it is impossible to make correct judgment on whether such annihilated groups of objects should be moved to another cluster or form a new one. That is, since the amount of information maintained by a k -means or k -median type approach is too rough in granularity, and once two cluster centers are joined, there is no way to informatively split the clusters when required by the changes in the stream at a later stage. Therefore, the quality of stream clustering could suffer from the lack of sufficient information.

Based on these observations, we proposed a new stream clustering method, called CluStream (AHWY03) based on two major ideas: micro-clustering and tilted time frame, which are useful for efficient bookkeeping in data streams and overcome the two problems: lack of historical information and lack of appropriate level of granularity.

First, we need to explore the concept of micro-clustering (ZRL96) to ensure that clustering can be adapted to the dynamic changes of data streams by using additional information at a relatively fine level of granularity. More concretely, we maintain statistical information about the data locality in terms of microclusters, defined as a direct extension of the cluster feature vector (ZRL96). The microclusters have an additivity property which makes them a natural choice for data streams. When new data objects streaming in, they could be either absorbed by an existing microcluster, or form a new one,
depending on their relatively distance from the nearest microcluster center. The creation of new microclusters may lead to the merge of some nearby existing microclusters or removal of some small and outdated microclusters (treated as old outliers). These may lead to incremental maintenance and dynamic evolution of microclusters in data streams. The summary information in the microclusters can be used for creation of macroclusters (i.e., the high-level \( k \) clusters for an input parameter \( k \)) dynamically based on an efficient clustering algorithm, such as \( k \)-means, and application requirements. Such a microcluster-to-macrocluster framework will provide an efficient and quality clustering mechanism for dynamically changing data streams.

Second, we need to adopt tilted time frame, where microclusters are stored at snapshots in time. In the tilted time frame, the snapshots are maintained at different granularity depending upon their level of recency. This framework facilitates the fading of aged microclusters and discovery of the dynamic changes of the clusters with time. Since the time windows are tilted, the amount of information to be registered is limited and thus it strikes a balance between efficiency and the capability of detection of stream dynamics.

Notice that the proposed method requires us to register some additional information with respect to the tilted time frame and the finer level of granularity. People may wonder whether the overhead may outweigh the potential benefit. Based on our initial experiments, even with very moderate sized microclusters (such as the number of microclusters is only about \( 10^*k \), where \( k \) is the number of macroclusters), the quality of clustering can be enhanced substantially, with only very minimal overhead in computation time (AHWY03). The clustering quality enhancement by this approach is striking for dynamically changing data, such as network intrusion data. Therefore, with the current availability of fast computers and considerable sized main memory (usually more than one million times bigger in bytes than \( k \), where \( k \) is the desired number of macroclusters), it is realistic to trade some computation resource for the quality and flexibility of stream clustering.

With the above discussed framework, i.e., microclustering and tilted time window, several other clustering paradigms, such as density-based clustering (ABKS99), Gaussian distribution-based clustering (HiKe98), and distance- and connection-integrated hierarchical clustering (KHK99), can be re-examined in the context of stream clustering. Efficient and effective stream clustering algorithms can be developed following those paradigms as well.

We need to investigate the efficient and effective methods for stream clustering based on the ideas outline above and construct effective and high-quality methods for clustering dynamic, data-intensive streams to discover the dynamics of data streams.

5.3. Dynamic modeling and classification of data streams

As an essential data mining task, classification has been one of most popularly studied subjects in machine learning, statistics, and data mining (Quin86, WiFr00, Mitc97, HTF01, HaKa06). There have been many well-developed classifiers, such as decision trees, naïve Bayesian classifiers, Bayesian networks, nearest neighbor, neural networks, and support vector machines. In many studies, researchers have found that each classifier has advantages for certain types of data sets (LLS00). For stream data, there is a necessity to dynamically construct classifiers based on the historical and current information, which poses the following challenges.

1. The classifier construction process should be fast and dynamic because the data may arrive at a high rate, with dramatic changes. For example, thousands of packets can be collected from sensor nets every second, and millions of customers may make purchases every day.

2. The classifier should also evolve over time since the labels of similar records may change from time to time. As a result, how to keep trace of this type of evolution and how to discover the cause that leads to the evolution is an important and difficult problem.

To build fast, adaptive, and evolving classifiers in the stream data environment, one needs to re-examine the broad spectrum of available classifiers to see what kinds of classifiers may be good candidates for stream classification. Some of the known classifiers, such as neural networks and support vector machines, may not be good for single-scan, very fast model reconstruction while handling the huge volumes of stream data. Recently, (DoHu00, HSD01) developed a few efficient decision tree induction algorithms for stream data, such as VFDT and CVFDT. However, these stream classification methods are not adequate at handling data streams and discovering stream evolution regularity, based on the following analysis.

First, decision trees have been popularly adopted as the first choice in the classification of stream data,
such as (DoHu00, HSD01), for its simplicity and easy explanation. However, it is difficult to dynamically and drastically change decision trees due to the costly reconstruction once they have been built. In many real applications, there are frequent, dynamic changes in stream data, such as in stock data analysis, traffic or weather modeling, and so on. In addition, a large amount of raw data is needed to build a decision tree. According to the model proposed in (HSD01), it is necessary to keep such raw data in memory or on disk since they may be used later for updating the statistics when old records leave the window and for reconstructing parts of the tree. If the concept drifts often, the related data needs to be scanned multiple times so that the decision tree can be kept updated. This is usually unaffordable for streaming data. Also, after detecting the drift in the model, it may take a long time to accumulate sufficient data to build an accurate decision tree (HSD01). Any drift taking place during that period either cannot be caught or will make the tree unstable.

Second, based on the similar observation as in stream clustering, without storing historical information, it is impossible to detect the evolution regularity of classification models for dynamic data streams. In practice, one may need to compare the models constructed at different time frames and see what has changed. Moreover, it is often necessary to put different weights on the data arriving at different time frames so that aged data can be weighted less and be gradually faded out. This poses the necessity for a tilted time frame model.

Based on the above analysis, we proposed two major components for the construction and dynamic evolution of classification models: (1) tilted time frame, and (2) a classification method that is more adaptable to the changes of data.

First, a tilted time frame similar to that described above is introduced in our stream classification model. This ensures that essential historical information can be saved for fading aged data and for comparing models at different times to discover the evolution of classification models.

Second, instead of refining the decision tree model, we integrate a classification method that is more adaptive to the changes of data. We developed and tested three methods for classification of data streams: (1) ensemble classification (WFYH03), (2) naïve Bayesian classifier with boosting (YYHW05), and (3) k-nearest neighbor approach (AHWY06).

The first such test is to construct an ensemble classifier (WFYH03) that explores the power of ensemble multiple classifiers, each being a simple classification model adaptive to dynamic changes of data streams. Our experiments show the high accuracy and efficiency of such a classification methods, despite of the dynamic changes of data streams.

The second test is to integrate a naïve Bayesian classifier (YYHW05), since it is easy to construct and adapt. The naïve Bayesian classifier, in essence, maintains a set of probability distributions \( P(a_i | v) \) where \( a_i \) and \( v \) are the attribute value and the class label, respectively. To classify a record with several attribute values, it is assumed that the conditional probability distributions of these values are independent of each other. Thus, one can simply multiply the conditional probabilities together and label the record with the class label of the greatest probability. Despite its simplicity, the accuracy of the naïve Bayesian classifier is comparable to other classifiers, such as decision trees (Mite97, DoPa97). To further enhance the accuracy of stream classifier, a boosting technique similar to that described in (FrSc97) can be introduced. The boosting method retains a portion of recent data set as an independent testing set, and then puts additional weight on the misclassified cases, that is, increasing the weight when new similar cases of data streaming in. Our experimental results show that this method leads to a high quality stream classifier.

The third test is to construct a k-nearest neighbor classifier (AHWY06) that performs microclustering in a similar way as clustering evolving data streams and then perform classification dynamically based on k such precomputed and incrementally updated microclusters. This method integrated the stream-based microclustering with online classification based on k-nearest neighbors and leads to high accuracy and high efficiency/scalability in stream classification.

6. Conclusion

In this paper we have discussed issues, approaches, challenges and directions for Real-time and Stream Data Mining. In particular, we discuss real-time clustering and outlier detection approaches as well as stream data mining and multidimensional analysis. For each topic we discuss some approaches we have developed and the research that needs to be done.

Real-time knowledge discovery and dissemination is a new direction. Not only we need to extract useful patterns we also need to extract such patterns in a timely manner so that the emergency responder or the war fighter or the intelligence analyst has this information to make
appropriate decisions. On the one hand we can determine all possibilities ahead of time and execute during run time. At the other extreme we need to extract the nuggets as well as develop models during run time. We can also develop solutions that carry out partial computations ahead of time and the rest during run time.

Several techniques such as real-time query approximation have been proposed by the real-time research community. The knowledge discovery community has developed numerous data mining and machine learning algorithms. This paper attempts to integrate the solutions proposed by both communities.

7. References


[AHWY05] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. On efficient algorithms for high dimensional projected clustering of data streams. Data Mining and Knowledge Discovery, 10:251-273, 2005.


[HPY00] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD’00), pages 1-12, Dallas, TX, May 2000.


[LHHL06] H. Liu, Y. Lu, J. Han, and J. He. Error-adaptive and time-aware maintenance of frequency counts over data streams. In Proc. 2006 Int. Conf. on Web-Age Information Management (WAIM'06), pages 484-495, Hong Kong, China, June, 2006.


[YHW05] J. Yang, X. Yan, J. Han, and W. Wang. Discovering evolutionary classifier over high speed non-static stream. In S. Bandyopadhyay, U. Maulik, L. B.


