Chapter 3

A SURVEY OF CLASSIFICATION METHODS IN DATA STREAMS

Mohamed Medhat Gaber, Arkady Zaslavsky and Shonali Krishnaswamy
Caulfield School of Information Technology
Monash University,
900 Dandenong Rd, Caulfield East,
Melbourne VIC3145, Australia
{Mohamed.Medhat.Gaber, Arkady.Zaslavsky, Shonali.Krishnaswamy} @infotech.monash.edu.au

Abstract With the advance in both hardware and software technologies, automated data generation and storage has become faster than ever. Such data is referred to as data streams. Streaming data is ubiquitous today and it is often a challenging task to store, analyze and visualize such rapid large volumes of data. Most conventional data mining techniques have to be adapted to run in a streaming environment, because of the underlying resource constraints in terms of memory and running time. Furthermore, the data stream may often show concept drift, because of which adaptation of conventional algorithms becomes more challenging. One such important conventional data mining problem is that of classification. In the classification problem, we attempt to model the class variable on the basis of one or more feature variables. While this problem has been extensively studied from a conventional mining perspective, it is a much more challenging problem in the data stream domain. In this chapter, we will re-visit the problem of classification from the data stream perspective. The techniques for this problem need to be thoroughly re-designed to address the issue of resource constraints and concept drift. This chapter reviews the state-of-the-art techniques in the literature along with their corresponding advantages and disadvantages.

1. Introduction

Classification problems [19, 20] have been studied thoroughly as a major category of the data analysis tasks in machine learning, statistical inference [18] and data mining. Classification methods represent the set of supervised learning techniques where a set of dependent variables needs to be predicted based on another set of input attributes. There are two main distinctive approaches under

the supervised learning category: classification and regression. Classification is mainly concerned with categorical attributes as dependent variables; however regression is concerned with numerical attributes as its output. The classification process is divided into two phases: model building and model testing. In model building, a learning algorithm runs over a data set to induce a model that could be used in estimating an output. The quality of this estimation is assessed in the model testing phase. The model building process is referred to as training as well. Classification techniques [19, 20] have attracted the attention of researchers due to the significance of their applications. A variety of methods such as decision trees, rule based methods, and neural networks are used for the classification problem. Many of these techniques have been designed to build classification models from static data sets where several passes over the stored data is possible. This is not possible in the case of data streams, in which it is necessary to process the entire data set in one pass. Furthermore, the classification problem needs to be re-designed in the context of *concept drift*, a unique problem in the case of data streams.

The applications of data stream classification can vary from critical astronomical and geophysical applications [6] to real-time decision support in business and industrial applications [24, 25]. There are several potential scenarios for such applications. For example, classification and analysis of biosensor measurements around a city for security reasons is an important emerging application. The analysis of simulation results and on-board sensor readings in scientific applications has its potential in changing the mission plan or the experimental settings in real time. Web log and clickstream analysis is an important application in the electronic commerce domain. The classification of data streams generated from the marketplace such as stock market streaming information is another appealing application. Decision trees created from stock market data in distributed streaming environment have been used in MobiMine [24, 25].

The process of adapting classification models to many of the above applications is often non-trivial. The most important challenge with regard to classification is that of concept drifting of evolving data streams. The process of *concept drift* results from the natural tendency of the underlying data to evolve over time. The classifier is most likely to be outdated after a time window due to the continuous change of the streaming information on a temporal basis. We discuss this issue along with a number of other challenges for the classification problem. Solution approaches used in addressing these issues are summarized in order to emphasize their advantages, and drawbacks. This summarization also provides some insight for other stream mining techniques due to the shared research issues across different applications. A thorough discussion of classification techniques in data streams is given as a guide to researchers as well

as practitioners. The techniques are presented in an easy way with illustrative figures depicting each algorithm in a diagrammatic way.

The chapter is organized as follows. Research issues with regard to the stream classification problems are discussed in section 2. Section 3 represents the approaches proposed as solutions to address the previous research issues. Section 4 provides a survey of the classification techniques in stream mining literature. Techniques surveyed include the Ensemble-based Classification [30], Very Fast Decision Trees (VFDT) [9] with its extensions [22], [23], On-Demand Classification [3], On-Line Information Network (OLIN) [26], Lightweight Classification (LWClass) [14], Scalable Classification Algorithm by Learning decisiOn Patterns (SCALLOP) [12] and Adaptive Nearest Neighbor Classification for Data-streams (ANNCAD) [27]. This selection of techniques is based on the soundness of the techniques and how well the techniques addresses important research challenges. Finally, the chapter is concluded with a summary in section 5.

2. Research Issues

In this section, we will address the primary research issues encountered in the context of stream mining. While many of these issues are shared across different stream mining applications, we discuss these issues with a special emphasis on the problem of classification [4, 9, 10, 13, 14, 16, 17, 21, 28].

- **High Speed Nature of Data Streams:** The inherent characteristic of data streams is its high speed. The algorithm should be able to adapt to the high speed nature of streaming information. The rate of building a classification model should be higher than the data rate. Furthermore, it is not possible to scan the data more than once. This is referred to as the *one-pass* constraint.

- **Unbounded Memory Requirements:** Classification techniques require data to be resident in memory for building the model. The huge amounts of data streams generated rapidly dictate the need for unbounded memory. This challenge has been addressed using load shedding, sampling, aggregation, and creating data synopsis. The memory issue is an important motivation behind many of the developed techniques in the area.

- **Concept Drifting:** Concept drifts change the classifier results over time. This is because of the change in the underlying data patterns. It is also referred to as data stream *evolution* [1]. This results in the model becoming stale and less relevant over time. The capture of such changes would help in updating the classifier model effectively. The use of an outdated model could lead to a very low classification accuracy.

- **Tradeoff between Accuracy and Efficiency:** The main tradeoff in data stream mining algorithms is between the accuracy of the output with regard to the application and the time and space complexity. In many cases, approximation algorithms can guarantee error bounds, while maintaining a high level of efficiency.

- **Challenges in Distributed Applications:** A significant number of data stream applications run in mobile environments with limited bandwidth such as sensor networks and handheld devices. Thus knowledge structure representation is an important issue. After extracting models and patterns locally from data stream generators or receivers, it is important to transfer the data mining output to the user. The user could be a mobile user or a stationary one getting the results from mobile nodes. This is often a challenge because of the bandwidth limits in transferring data. Kargupta et al. [24] have addressed this problem by using Fourier transformations to efficiently represent decision trees for the purpose of transmission over limited bandwidth links.

- **Visualization of data stream mining results:** Visualization of traditional data mining results on a desktop has been a research issue for more than a decade. Visualization of mining results in small screens of a Personal Digital Assistant (PDA) for example is a real challenge and an open research problem. Given a scenario for a businessman on a move and the data are being streamed and analyzed on his PDA. The results of this analysis should be efficiently visualized in a way that allows him to take a quick decision. The pioneering work on representation of decision trees in a mobile device has been suggested by Kargupta et al [24].

- **Modelling change of mining results over time:** In some cases, the user is not interested in mining data stream results, but how these results are changing over a temporal basis. The classification changes could help in understanding the change in data streams over time.

- **Interactive Mining environment to satisfy user results:** Mining data streams is a highly application oriented field. For example, the user should be able to change the classification parameters to serve the special needs of the user under the current context. The fast nature of data streams often makes it more difficult to incorporate user-interaction.

- **The integration of data stream management systems and data stream mining approaches:** The integration among storage, querying, mining and reasoning of the incoming stream would realize robust streaming systems that could be used in different applications [5, 7]. Along this line, current database management systems have achieved this goal over

static stored data sets. However, this goal has not been fully realized for the case of data streams. An important future research issue is to integrate the stream mining algorithms with known stream management systems in order to design complete systems for stream processing.

- **Hardware and other Technological Issues:** The technological issue of mining data streams is an important one. How do we represent the data in such an environment in a compressed way? Which platforms are best suited such special real-time applications? Hardware issues are of special concerns. Small devices generating data streams are not designed for complex computations. Currently emulators are used for such tasks and it is a real burden for data stream mining applications which run in resource-constrained environments. Novel hardware solutions are required to address this issue.

- **Real time accuracy evaluation and formalization:** In many cases, resource constrained methods work with a trade-off between accuracy and efficiency of the designed method. Therefore, we need a feedback of the current achieved accuracy with relation to the available resources. This is needed to adjust the algorithm parameters according to the available resources. This formalization would also help in making decisions about the reliability of the output.

Among the above-mentioned issues, the first three are of special significance. Thus, we will use them as the basis for comparing different stream classification techniques in this chapter. We also note that many of these issues are shared among all mining techniques in streaming environment. The following section concisely summarizes the approaches that are used as solutions addressing the above issues.

3. Solution Approaches

Many of the afore-mentioned issues can be solved using well-established statistical and computational approaches. While, specific methods for stream classification will be discussed later, it is useful to understand the broad characteristics of different methods which are used to adapt conventional classification techniques to the case of data streams. We can categorize these solutions as data-based and task-based ones. In data-based solutions, the idea is to examine only a subset of the whole data set or to transform the data vertically or horizontally to an approximate smaller size data representation. Such an approach allows us to utilize many known data mining techniques to the case of data streams. On the other hand, in task based solutions, some standard algorithmic modification techniques can be used to achieve time and space efficient solutions [13]. Table 3.1 shows the data-based techniques, while Table 3.2 shows

Technique	Definition	Pros	Cons
Sampling	Choosing a data subset for analysis	Error Bounds Guaranteed	Poor for anomaly detection
Load Shedding	Ignoring a chunk of data	Efficient for queries	Very poor for anomaly detection
Sketching	Random projection on feature set	Extremely Efficient	May ignore Relevant features
Synopsis Structure	Quick Transformation	Analysis Task Independent	Not sufficient for very fast stream
Aggregation	Compiling summary statistics	Analysis Task Independent	May ignore Relevant features

Table 3.1. Data Based Techniques

Technique	Definition	Pros	Cons
Approximation Algorithms	Algorithms with Error Bounds	Efficient	Resource adaptivity with data rates not always possible
Sliding Window	Analyzing most recent streams	General	Ignores part of stream
Algorithm Output Granularity	Highly Resource aware technique with memory and fluctuating data rates	General	Cost overhead of resource aware component

Table 3.2. Task Based Techniques

the task-based techniques. Each table provides a definition, advantages and disadvantages of each technique.

While the methods in Tables 3.1 and 3.2 provide an overview of the broad methods which can be used to adapt conventional methods to classification, it is more useful to study specific techniques which are expressly designed for the purpose of classification. In the next section, we will provide a review of these methods.

4. Classification Techniques

This section reviews the state-of-the-art of data stream classification techniques. We have provided an overview of some of the key methods, how well they address the research problems discussed earlier.

4.1 Ensemble Based Classification

Wang et al. [30] have proposed a generic framework for mining concept drifting data streams. The framework is based on the observation that many data stream mining algorithms do not address the issue of concept drift in the evolving data. The idea is based on using an ensemble of classification models such as decision trees using C4.5, RIPPER, naïve Bayesian and others to vote for the classification output to increase the accuracy of the predicted output.

This framework was developed to address three research challenges in data stream classification:

1. **Concept Drift:** The accuracy of the output of many classifiers is very sensitive to concept drifts in the evolving streams. At the same time, one does not want to remove excessive parts of the stream, when there is no concept drift. Therefore, a method needs to be designed to decide which part of the stream to be used for the classification process.

2. **Efficiency:** The process of building classifiers is a complex computational task and the update of the model due to concept drifts is a complicated process. This is especially relevant in the case of high speed data streams.

3. **Robustness:** Ensemble based classification has traditionally been used in order to improve robustness. The key idea is to avoid the problem of overfitting of individual classifiers. However, it is often a challenging task to use the ensemble effectively because of the high speed nature of the data streams.

An important motivation behind the framework is to deal with the expiration of old data streams. The idea of using the most recent data streams to build and use the developed classifiers may not be valid for most applications. Although the old streams can affect the accuracy of the classification model in a negative way, it is still important to keep track of this data in the current model. The work in [30] shows that it is possible to use weighted ensemble classifiers in order to achieve this goal.

The work in [30] uses weighted classifier ensembles according to the current accuracy of each classifier used in the ensemble. The weight of any classifier is calculated and contributed to predict the final output. The weight of each classifier may vary as the data stream evolves, and a given classifier may become more or less important on a particular sequential chunk of the data. The framework has outperformed single classifiers experimentally. This is partly because of the greater robustness of the ensemble, and partly because of more effective tracking of the change in the underlying structure of the data. More interesting variations of similar concepts may be found in [11]. Figure 3.1 depicts the proposed framework.

4.2 Very Fast Decision Trees (VFDT)

Domingos and Hulten [9, 22] have developed a decision tree approach which is referred to as *Very Fast Decision Trees (VFDT)*. It is a decision tree learning system based on Hoeffding trees. It splits the tree using the current best attribute taking into consideration that the number of examples used satisfies the Hoeffding bound. Such a technique has the property that its output is (asymptotically) nearly identical to that of a conventional learner. VFDT is an extended version of such a method which can address the research issues of data streams. These research issues are:

- **Ties of attributes:** Such ties occur when two or more attributes have close values of the splitting criteria such as information gain or gini index. We note that at such a moment of the decision tree growth phase, one must make a decision between two or more attributes based on only the set of records received so far. While it is undesirable to delay such split decisions indefinitely, we would like to do so at a point when the errors are acceptable.

- **Bounded memory:** The tree can grow till the algorithm runs out of memory. This results in a number of issues related to effective maintenance of the tree.

- **Efficiency and Accuracy:** This is an inherent characteristic of all data stream algorithms.

The extension of Hoeffding trees in VFDT has been done using the following techniques.

- The key question during the construction of the decision tree is the choice of attributes to be used for splits. Approximate ties on attributes are broken using a user-specified threshold of acceptable error measure for the output. By using this approach, a crisp criterion can be determined on when a split (based on the inherently incomplete information from the current data stream) provides acceptable error. In particular, the Hoeffding inequality provides the necessary bound on the correctness of the choice of split variable. It can be shown for any small value of δ, that a particular choice of the split variable is the correct choice (same as conventional learner) with probability at least $1 - \delta$, if a sufficient number of stream records have been processed. This "sufficient number" increases at the relatively modest rate of $\log(1/\delta)$. The bound on the accuracy of each split can then be extrapolated to the behavior of the entire decision tree. We note that the stream decision tree will provide the same result as the conventional decision tree, if for every node along the path for given test instance, the same choice of split is used. This

can be used to show that the behavior of the stream decision tree for a particular test instance differs from the conventional decision tree with probability at most $1 - \delta/p$, where p is the probability that a record is assigned to a leaf at each level.

- Bounded memory has been addressed by de-activating the least promising leaves and ignoring the poor attributes. The calculation of these poor attributes is done through the difference between the splitting criteria of the highest and lowest attributes. If the difference is greater than a pre-specified value, the attribute with the lowest splitting measure will be freed from memory.

- The VFDT system is inherently I/O bound; in other words, the time for processing the example is lower than the time required to read it from disk. This is because of the Hoeffding tree-based approach with a crisp criterion for tree growth and splits. Such an approach can make clear decisions at various points of the tree construction algorithm without having to re-scan the data. Furthermore, the computation of the splitting criteria is done in a batch processing mode rather than online processing. This significantly saves the time of recalculating the criteria for all the attributes with each incoming record of the stream. The accuracy of the output can be further improved using multiple scans in the case of low data rates.

All the above improvements have been tested using special synthetic data sets. The experiments have proved efficiency of these improvements. Figure 3.2 depicts the VFDT learning system. The VFDT has been extended to address the problem of concept drift in evolving data streams. The new framework has been termed as CVFDT [22]. It runs VFDT over fixed sliding windows in order to have the most updated classifier. The change occurs when the splitting criteria changes significantly among the input attributes.

Jin and Agrawal [23] have extended the VFDT algorithm to efficiently process numerical attributes and reduce the sample size calculated using the Hoeffding bound. The former objective has been addressed using their Numerical Interval Pruning (NIP) technique. The pruning is done by first creating a histogram for each interval of numbers. The least promising intervals to be branched are pruned to reduce the memory space. The experimental results show an average of 39% of space reduction by using NIP. The reduction of sample size is done by using properties of information gain functions. The derived method using multivariate delta method has a guarantee of a reduction of sample size over the Hoeffding inequality with the same accuracy. The experiments show a reduction of 37% of the sample size by using the proposed method.

4.3 On Demand Classification

Aggarwal et al. have adopted the idea of micro-clusters introduced in CluStream [2] in On-Demand classification in [3]. The on-demand classification method divides the classification approach into two components. One component continuously stores summarized statistics about the data streams and the second one continuously uses the summary statistics to perform the classification. The summary statistics are represented in the form of class-label specific micro-clusters. This means that each micro-cluster is associated with a specific class label which defines the class label of the points in it. We note that both components of the approach can be used in online fashion, and therefore the approach is referred to as an *on-demand classification method.* This is because the set of test instances could arrive in the form of a data stream and can be classified efficiently on demand. At the same time, the summary statistics (and therefore training model) can be efficiently updated whenever new data arrives. The great flexibility of such an approach can be very useful in a variety of applications.

At any given moment in time, the current set of micro-clusters can be used to perform the classification. The main motivation behind the technique is that the classification model should be defined over a time horizon which depends on the nature of the concept drift and data evolution. When there is smaller concept drift, we need a larger time horizon in order to ensure robustness. In the event of greater concept drift, we require smaller time horizons. One key property of micro-clusters (referred to as the *subtractive property*) ensures that it is possible to compute horizon-specific statistics. As a result it is possible to perform the classification over a wide variety of time horizons. A hold out training stream is used to decide the size of the horizon on which the classification is performed. By using a well-chosen horizon it is possible to achieve a high level of classification accuracy. Figure 3.3 depicts the classification on demand framework.

4.4 Online Information Network (OLIN)

Last [26] has proposed an online classification system which can adapt to concept drift. The system re-builds the classification model with the most recent examples. By using the error-rate as a guide to concept drift, the frequency of model building and the window size is adjusted over time.

The system uses info-fuzzy techniques for building a tree-like classification model. It uses information theory to calculate the window size. The main idea behind the system is to change the sliding window of the model reconstruction according to the classification error rate. If the model is stable, the window size increases. Thus the frequency of model building decreases. The info-fuzzy technique for building a tree-like classification model is referred to as the Info-

A_1	A_2	...	A_n	Class	Weight
$Value(A_1)$	$Value(A_2)$	...	$Value(A_n)$	Class Category	X = # items Contributing
...	...	...	...	...	...

Table 3.3. Typical LWClass Training Results

Fuzzy Network (IFN). The tree is different than conventional decision trees in that each level of the tree represents only one attribute except the root node layer. The nodes represent different values of the attribute. The process of inducing the class label is similar to the one of conventional decision trees. The process of constructing this tree has been termed as Information Network (IN). The IN technique uses a similar procedure of building conventional decision trees by determining if the split of an attribute would decrease the entropy or not. The measure used is mutual conditional information that assesses the dependency between the current input attribute under examination and the output attribute. At each iteration, the algorithm chooses the attribute with the maximum mutual information and adds a layer with each node represents a different value of this attribute. The iterations stop once there is no increase in the mutual information measure for any of the remaining attributes that have not been considered in the tree. OLIN system repeatedly uses the IN algorithm for building a new classification model. The system uses the information theory to calculate the window size (refers to number of examples). It uses a less conservative measure than Hoeffding bound used in VFDT [9, 22] reviewed earlier in this chapter. This measure is derived from the mutual conditional information in the IN algorithm by applying the likelihood ratio test to assess the statistical significance of the mutual information. Subsequently, we change the window size of the model reconstruction according to the classification error rate. The error rate is calculated by measuring the difference between the error rate during the training at one hand and the error rate during the model validation at the other hand. A significance increase in the error rate indicates a high probability of a concept drift. The window size changes according to the value of this increase. Figure 3.4 shows a simple flow chart of the OLIN system.

4.5 LWClass Algorithm

Gaber et al [14] have proposed Lightweight Classification techniques termed as LWClass. LWClass is based on Algorithm Output Granularity. The algorithm output granularity (AOG) introduces the first resource-aware data analysis approach that can cope with fluctuating data rates according to the available memory and the processing speed. The AOG performs the local data analysis on resource constrained devices that generate or receive streams of informa-

tion. AOG has three stages of mining, adaptation and knowledge integration as shown in Figure 3.5 [14].

LWClass starts with determining the number of instances that could be resident in memory according to the available space. Once a classified data record arrives, the algorithm searches for the nearest instance already stored in the main memory. This is done using a pre-specified distance threshold. This threshold represents the similarity measure acceptable by the algorithm to consider two or more data records as an entry into a matrix. This matrix is a summarized version of the original data set. If the algorithm finds a nearest neighbor, it checks the class label. If the class label is the same, it increases the weight for this instance by one, otherwise it decrements the weight by one. If the weight is decremented down to zero, this entry will be released from the memory conserving the limited memory on streaming applications. The algorithm output granularity is controlled by the distance threshold value and is changing over time to cope with the high speed of the incoming data elements. The algorithm procedure could be described as follows:

1. Each record in the data stream contains attribute values for a_1, a_2,..., a_n attributes and the class category.

2. According to the data rate and the available memory, the algorithm output granularity is applied as follows:

   2.1 Measure the distance between the new record and the stored ones.

   2.2 If the distance is less than a threshold, store the average of these two records and increase the weight for this average as an entry by 1. (The threshold value determines the algorithm accuracy and is chosen according to the available memory and data rate that determines the algorithm rate). This is in case that both items have the same class category. If they have different class categories, the weight is decreased by 1 and released from memory if the weight reaches zero.

   2.3 After a time threshold for the training, we come up with a matrix represented in Table 3.3.

3. Using Table 3.3, the unlabeled data records could be classified as follows. According to the available time for the classification process, we choose nearest K-table entries and these entries are variable according to the time needed by the process.

4. Find the majority class category taking into account the calculated weights from the K entries. This will be the output for this classification task.

4.6 ANNCAD Algorithm

Law et al [27] have proposed an incremental classification algorithm termed as Adaptive Nearest Neighbor Classification for Data-streams (ANNCAD). The algorithm uses Haar Wavelets Transformation for multi-resolution data representation. A grid-based representation at each level is used.

The process of classification starts with attempting to classify the data record according to the majority nearest neighbors at finer levels. If the finer levels are unable to differentiate between the classes with a pre-specified threshold, the coarser levels are used in a hierarchical way. To address the concept drift problem of the evolving data streams, an exponential fade factor is used to decrease the weight of old data in the classification process. Ensemble classifiers are used to overcome the errors of initial quantization of data. Figure 3.6 depicts the ANNCAD framework.

Experimental results over real data sets have proved the achieved accuracy over the VFDT and CVFDT discussed earlier in this section. The drawback of this technique represented in inability of dealing with sudden concept drifts as the exponential fade factor takes a while to have its effect felt. In fact, the choice of the exponential fade factor is an inherent flexibility which could lead to over-estimation or under-estimation of the rate of concept drift. Both errors would result in a reduction in accuracy.

4.7 SCALLOP Algorithm

Ferrer-Troyano et al. [12] have proposed a scalable classification algorithm for numerical data streams. This is one of the few rule-based classifiers for data streams. It is inherently difficult to construct rule based classifiers for data streams, because of the difficulty in maintaining the underlying rule statistics. The algorithm has been termed as Scalable Classification Algorithm by Learning decisiOn Patterns (SCALLOP).

The algorithm starts by reading a number of user-specified labeled records. A number of rules are created for each class from these records. Subsequently, the key issue is to effectively maintain the rule set after arrival of each new record. On the arrival of a new record, there are three cases:

a) **Positive covering:** This is the case of a new record that strengthens a current discovered rule.

b) **Possible expansion:** This is the case of a new record that is associated with at least one rule, but is not covered by any currently discovered rule.

c) **Negative covering:** This is the case of a new record that weakens a currently discovered rule.

For each of the above cases, a different procedure is used as follows:

a) **Positive covering:** The positive support and confidence of the existing rule is re-calculated.

b) **Possible expansion:** In this case, the rule is extended if it satisfies two conditions:

   - It is bounded within a user-specified growth bounds to avoid a possible wrong expansion of the rule.
   - There is no intersection between the expanded rule and any already discovered rule associated with the same class label.

c) **Negative covering:** In this case, the negative support and confidence is re-calculated. If the confidence is less than a minimum user-specified threshold, a new rule is added.

After reading a pre-defined number of records, the process of rule refining is performed. Rules in the same class and within a user-defined acceptable distance measure are merged. At the same time, care is taken to ensure that these rules do not intersect with rules associated with other class labels. The resulting hypercube of the merged rules should also be within certain growth bounds. The algorithm also has a refinement stage. This stage releases the uninteresting rules from the current model. In particular, the rules that have less than the minimum positive support are released. Furthermore, the rules that are not covered by at least one of the records of the last user-defined number of received records are released. Figure 3.7 shows an illustration of the basic process.

Finally a voting-based classification technique is used to classify the unlabeled records. If there is a rule covers the current record, the label associated with that rule is used as the classifier output. Otherwise, a voting over the current rules within the growth bounds is used to infer the class label.

5. Summary

Stream classification techniques have several important applications in business, industry and science. This chapter reviews the research problems in data stream classification. Several approaches in the literature have been summarized with their advantages and drawbacks. While the selection of the techniques is based on the performance and quality of addressing the research challenges, there are a number of other methods [11, 8, 15, 22, 31] which we have not discussed in greater detail in this chapter. Many of these techniques are developed along similar lines as one or more techniques presented in this chapter.

The major research challenges in data stream classification are represented in concept drifting, resource adaptivity, high data rates, and the unbounded memory requirements. While many methods have been proposed to address some of

Method	Concept Drift	High Speed	Memory Req.
Ensemble-based Classification	X		
VFDT		X	X
On-Demand Classification	X	X	X
Online Information Network	X		
LWClass		X	X
ANNCAD	X		
SCALLOP	X		

Table 3.4. Summary of Reviewed Techniques

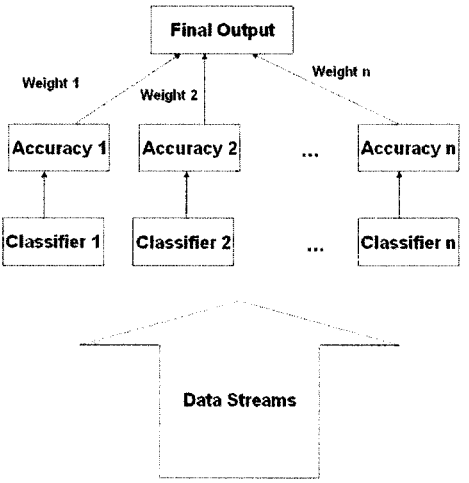

Figure 3.1. The ensemble based classification method

these issues, they are often unable to address these issues simultaneously. Table 3.4 summarizes the previously reviewed techniques in terms of addressing the above challenges. The area of data stream classification is still in its infancy. A number of open challenges still remain in stream classification algorithms; particular in respect to concept drift and resource adaptive classification.

References

[1] Aggarwal C. (2003) A Framework for Diagnosing Changes in Evolving Data Streams. *Proceedings of the ACM SIGMOD Conference.*

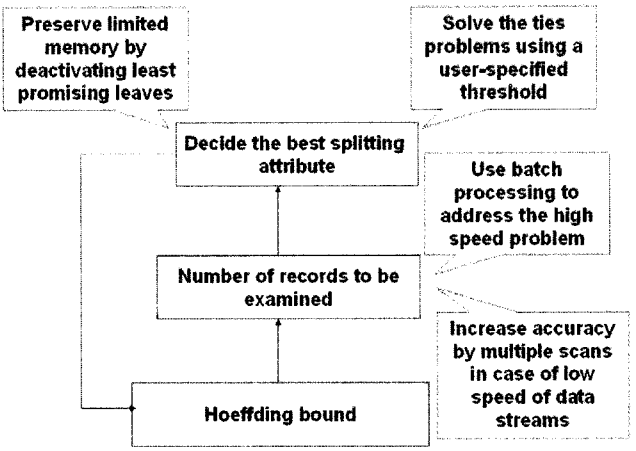

Figure 3.2. VFDT Learning Systems

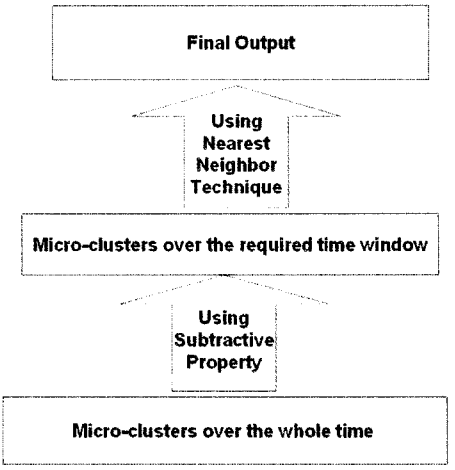

Figure 3.3. On Demand Classification

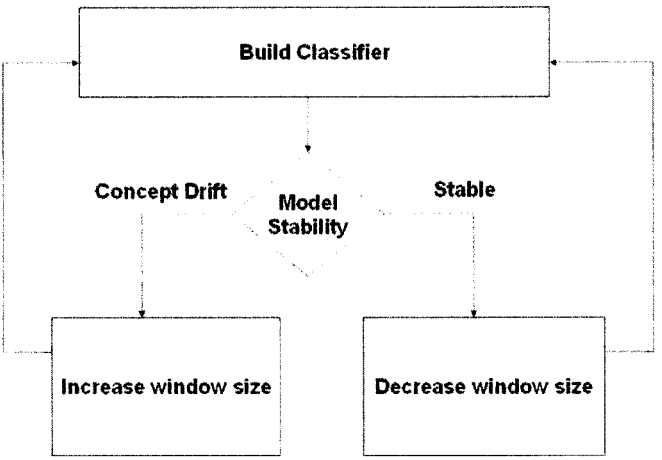

Figure 3.4. Online Information Network System

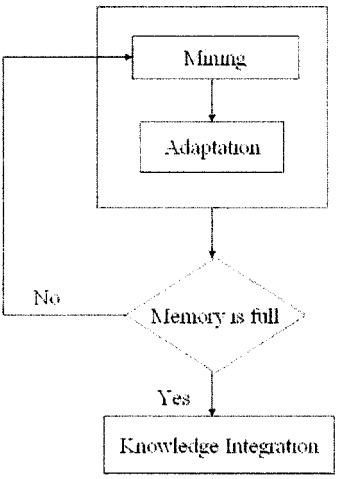

Figure 3.5. Algorithm Output Granularity

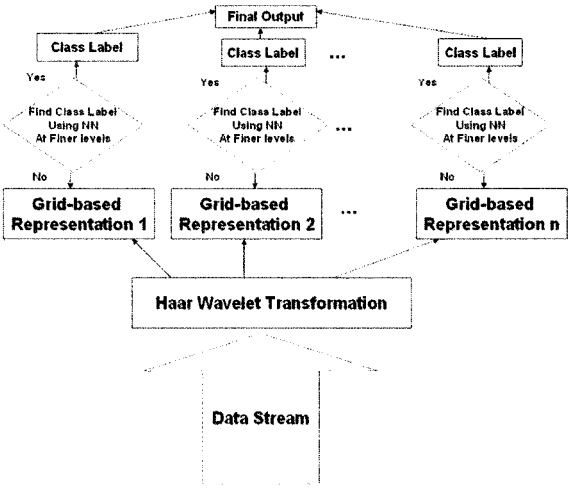

Figure 3.6. ANNCAD Framework

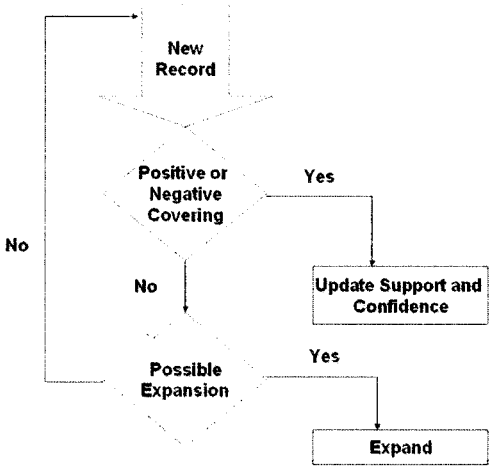

Figure 3.7. SCALLOP Process

[2] Aggarwal C., Han J., Wang J., Yu P. S., (2003) A Framework for Clustering Evolving Data Streams, *Proc. 2003 Int. Conf. on Very Large Data Bases (VLDB'03), Berlin, Germany, Sept. 2003.*

[3] Aggarwal C., Han J., Wang J., Yu P. S., (2004) On Demand Classification of Data Streams, *Proc. 2004 Int. Conf. on Knowledge Discovery and Data Mining (KDD'04), Seattle, WA.*

[4] Babcock B., Babu S., Datar M., Motwani R., and Widom J. (2002) Models and issues in data stream systems. In *Proceedings of PODS.*

[5] Babcock B., Datar M., and Motwani R. (2003) Load Shedding Techniques for Data Stream Systems (short paper) In *Proc. of the 2003 Workshop on Management and Processing of Data Streams (MPDS 2003).*

[6] Burl M., Fowlkes C., Roden J., Stechert A., and Mukhtar S. (1999), Diamond Eye: A distributed architecture for image data mining, in *SPIE DMKD, Orlando.*

[7] Cai Y. D., Clutter D., Pape G., Han J., Welge M., Auvil L. (2004) MAIDS: Mining Alarming Incidents from Data Streams. *Proceedings of the 23rd ACM SIGMOD (International Conference on Management of Data).*

[8] Ding Q., Ding Q, and Perrizo W., (2002) Decision Tree Classification of Spatial Data Streams Using Peano Count Trees, *Proceedings of the ACM 124 Symposium on Applied Computing*, Madrid, Spain, pp. 413–417.

[9] Domingos P. and Hulten G. (2000) Mining High-Speed Data Streams. In *Proceedings of the Association for Computing Machinery Sixth International Conference on Knowledge Discovery and Data Mining.*

[10] Dong G., Han J., Lakshmanan L. V. S., Pei J., Wang H. and Yu P. S. (2003) Online mining of changes from data streams: Research problems and preliminary results, In *Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams.*

[11] Fan W. (2004) Systematic data selection to mine concept-drifting data streams. *ACM KDD Conference*, pp. 128-137.

[12] Ferrer-Troyano F. J., Aguilar-Ruiz J. S. and Riquelme J. C. (2004) Discovering Decision Rules from Numerical Data Streams, *ACM Symposium on Applied Computing*, pp. 649-653.

[13] Gaber, M, M., Zaslavsky, A., and Krishnaswamy, S. (2005) Mining Data Streams: A Review. *ACM SIGMOD Record*, Vol. 34, No. 1, June 2005, ISSN: 0163-5808.

[14] Gaber, M, M., Krishnaswamy, S., and Zaslavsky, A., (2005). On-board Mining of Data Streams in Sensor Networks, Accepted as a chapter in the forthcoming book *Advanced Methods of Knowledge Discovery from Complex Data*, (Eds.) Sanghamitra Badhyopadhyay, Ujjwal Maulik, Lawrence Holder and Diane Cook, Springer Verlag, to appear.

[15] Gama J., Rocha R. and Medas P. (2003), Accurate Decision Trees for Mining High-Speed Data Streams, *Proceedings of the Ninth International Conference on Knowledge Discovery and Data Mining.*

[16] Garofalakis M., Gehrke J., Rastogi R. (2002) Querying and mining data streams: you only get one look a tutorial. SIGMOD Conference, 635.

[17] Golab L. and Ozsu T. M. (2003) Issues in Data Stream Management. In *SIGMOD Record*, Volume 32, Number 2, pp. 5–14.

[18] Hand D. J. (1999) Statistics and Data Mining: Intersecting Disciplines *ACM SIGKDD Explorations*, 1, 1, pp. 16-19.

[19] Hand D.J., Mannila H., and Smyth P. (2001) *Principles of data mining*, MIT Press.

[20] Hastie T., Tibshirani R., Friedman J. (2001) *The elements of statistical learning: data mining, inference, and prediction*, New York: Springer.

[21] Henzinger M., Raghavan P. and Rajagopalan S. (1998), Computing on data streams , Technical Note 1998-011, Digital Systems Research Center, Palo Alto, CA.

[22] Hulten G., Spencer L., and Domingos P. (2001) Mining Time-Changing Data Streams. *ACM SIGKDD Conference.*

[23] Jin R. and Agrawal G. (2003), Efficient Decision Tree Construction on Streaming Data, in *Proceedings of ACM SIGKDD Conference.*

[24] Kargupta, H., Park, B., Pittie, S., Liu, L., Kushraj, D. and Sarkar, K. (2002). MobiMine: Monitoring the Stock Market from a PDA. *ACM SIGKDD Explorations*, Volume 3, Issue 2. Pages 37–46. ACM Press.

[25] Kargupta H., Bhargava R., Liu K., Powers M., Blair S., Bushra S., Dull J., Sarkar K., Klein M., Vasa M., and Handy D. (2004) VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring. *Proceedings of SIAM International Conference on Data Mining.*

[26] Last M. (2002) Online Classification of Nonstationary Data Streams,*Intelligent Data Analysis*, Vol. 6, No. 2, pp. 129-147.

[27] Law Y., Zaniolo C. (2005) An Adaptive Nearest Neighbor Classification Algorithm for Data Streams, *Proceedings of the 9th European Conference on the Principals and Practice of Knowledge Discovery in Databases*, Springer Verlag, Porto, Portugal.

[28] Muthukrishnan S. (2003) Data streams: algorithms and applications. *Proceedings of the fourteenth annual ACM-SIAM symposium on discrete algorithms.*

[29] Park B. and Kargupta H. (2002) Distributed Data Mining: Algorithms, Systems, and Applications. To be published in the Data Mining Handbook. Editor: Nong Ye.

[30] Wang H., Fan W., Yu P. and Han J. (2003) Mining Concept-Drifting Data Streams using Ensemble Classifiers, in the *9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington DC, USA.*

[31] Wang K., Zhou S., Fu A., Yu J. (2003) Mining changes of classification by correspondence tracing. *SIAM International Conference on Data Mining.*