

On Classification of High-Cardinality Data Streams

Charu C. Aggarwal*

Philip S. Yu†

Abstract

The problem of massive-domain stream classification is one in which each attribute can take on one of a large number of possible values. Such streams often arise in applications such as IP monitoring, super-store transactions and financial data. In such cases, traditional models for stream classification cannot be used because the size of the storage required for intermediate storage of model statistics can increase rapidly with domain size. Furthermore, the one-pass constraint for data stream computation makes the problem even more challenging. For such cases, there are no known methods for data stream classification. In this paper, we propose the use of massive-domain counting methods for effective modeling and classification. We show that such an approach can yield accurate solutions while retaining space- and time-efficiency. We show the effectiveness and efficiency of the sketch-based approach.

1 Introduction

A well known problem in the data mining domain is that of classification [4, 12, 13]. In the classification problem, we use a labeled training data set in order to supervise the classification of unlabeled data instances. The problem of classification has also been widely studied in the data stream domain [1, 7, 8, 9, 11, 10, 14]. In this paper we will examine the problem of *massive-domain stream classification*. A massive-domain stream is defined as one in which each attribute takes on an extremely large number of possible values. Some examples are as follows:

- (1) In internet applications, the number of possible source and destination addresses can be very large. For example, there may be well over 10^8 possible IP-addresses.
- (2) The individual items in supermarket transactions are often drawn from millions of possibilities.
- (3) In general, when the attributes contain very detailed information such as locations, addresses, names, phone-numbers or other such information, the domain size is very large. Recent years have seen a considerable increase in such applications because of advances in data collection techniques.

We note that the computational and space-efficiency challenges are dual problems which arise in the context of a variety of massive-domain scenarios. For example, the techniques presented in [2] discuss the computational and space-efficiency challenges associated with clustering massive-domain data streams.

The problem of massive-domain size naturally occurs in the space of *discrete attributes*, whereas most of the known data stream classification methods are designed on the space of *continuous attributes*. The one-pass restrictions of data stream computation create a further challenge for the *computational approach* which may be used for discriminatory analysis. Thus, the massive-domain size creates challenges in terms of space-requirements, whereas the stream model further restricts the classes of algorithms which may be used in order to create space-efficient methods. For example, consider the following types of classification models:

- Techniques such as decision trees [4, 13] require the computation of the discriminatory power of each possible attribute value in order to determine how the splits should be constructed. In order to compute the relative behavior of different attribute values, the discriminatory power of different attribute values (or combinations of values) need to be maintained. This becomes difficult in the context of massive data streams.
- Techniques such as rule-based classifiers [5, 12] require the determination of combinations of attributes which are relevant to classification. With increasing domain size, it is no longer possible to compute this efficiently either in terms of space or running time.

The stream scenario presents additional challenges for classifiers. This is because the one-pass constraint dictates the choice of data structures and algorithms which can be used for the classification problem. All stream classifiers such as that discussed in [7] implicitly assume that the underlying domain size can be handled with modest main memory or storage limitations.

One observation is that massive-domain data sets are often noisy, and most combinations of dimensions may not have any relationship with the true class label. While the number of possible relevant combinations may be small enough to be stored within reasonable space limitations, the *intermediate computations* required for *determining* such combinations may not be

*IBM T. J. Watson Research Center, charu@us.ibm.com

†University of Illinois at Chicago, psyu@cs.uic.edu

feasible from a space and time perspective. This is because the determination of the most discriminatory patterns requires intermediate computation of statistics of patterns which are not relevant. When combined with the one-pass constraint of data streams, this is a very challenging problem. In this paper, we will use a sketch-based approach to perform classification of data streams with massive domain-sizes. The idea is to create a sketch-based model which can approximately identify combinations of attributes which have high discriminatory power. This approximation is used for classification.

This paper is organized as follows. In the next section we will discuss the process of construction of the sketch on the massive-domain data set, and its use for classification. In section 3, we discuss the experimental results. Section 4 contains the conclusions and summary.

2 A Sketch Based Approach to Classification

We will first introduce some notations and definitions. We assume that the data stream \mathcal{D} contains d -dimensional records which are denoted by $\overline{X}_1 \dots \overline{X}_N \dots$. Associated with each records is a class which is drawn from the index $\{1 \dots k\}$. The attributes of record \overline{X}_i are denoted by $(x_i^1 \dots x_i^d)$. It is assumed that the attribute value x_i^k is drawn from the unordered domain set $J_k = \{v_1^k \dots v_{M^k}^k\}$. We note that the value of M^k denotes the domain size for the k th attribute. The value of M^k can be very large, and may range in the order of millions or billions. When the discriminatory power is defined in terms of subspaces of higher dimensionality, this number multiplies rapidly to very large values. Such intermediate computations will be difficult to perform on even high-end machines.

Even though an extremely large number of attribute-value combinations may be possible over the different dimensions and domain sizes, only a limited number of these possibilities are usually relevant for classification purposes. Unfortunately, the intermediate computations required to effectively compare these combinations may not be easily feasible. The one-pass constraint of the data stream model creates an additional challenge in the computation process. In order to perform the classification, it is not necessary to explicitly determine the combinations of attributes which are related to a given class label. The more relevant question is the determination of *whether some combinations of attributes exist* that are strongly related to some class label. We will see that a sketch-based approach is very effective in such a scenario.

Sketch based approaches [6] were designed for enumeration of different kinds of frequency statistics of data

Algorithm *SketchUpdate*(Labeled Data Stream: \mathcal{D} ,
NumClasses: k , MaxDim: r)

```

begin
  Initialize  $k$  sketch tables of size  $w \cdot h$  each
  with zero counts in each entry;
  repeat
    Receive next data point  $\overline{X}$  from  $\mathcal{D}$ ;
    Add 1 to each of the sketch counts in the
    (class-specific) table for all  $L$  value-combinations
    in  $\overline{X}$  with dimensionality less than  $r$ ;
  until(all points in  $\mathcal{D}$  have been processed);
end

```

Figure 1: Sketch Updates for Classification (Training)

sets. In this paper, we will extend the well known count-min sketch [6] to the problem of classification of data streams. In this sketch, we use $w = \lceil \ln(1/\delta) \rceil$ pairwise independent hash functions, each of which map onto uniformly random integers in the range $h = [0, e/\epsilon]$, where e is the base of the natural logarithm. The data structure itself consists of a two dimensional array with $w \cdot h$ cells with a length of h and width of w . Each hash function corresponds to one of w 1-dimensional arrays with h cells each. In standard applications of the count-min sketch, the hash functions are used in order to update the counts of the different cells in this 2-dimensional data structure. For example, consider a 1-dimensional data stream with elements drawn from a massive set of domain values. When a new element of the data stream is received, we apply each of the w hash functions to map onto a number in $[0 \dots h-1]$. The count of each of the set of w cells is incremented by 1. In order to *estimate* the count of an item, we determine the set of w cells to which each of the w hash-functions map, and determine the minimum value among all these cells. Let c_t be the true value of the count being estimated. We note that the estimated count is at least equal to c_t , since we are dealing with non-negative counts only, and there may be an over-estimation because of collisions among hash cells. As it turns out, a probabilistic upper bound to the estimate may also be determined. It has been shown in [6], that for a data stream with T arrivals, the estimate is at most $c_t + \epsilon \cdot T$ with probability at least $1 - \delta$.

In typical subspace classifiers such as rule-based classifiers, we use low dimensional projections such as 2-dimensional or 3-dimensional combinations of attributes in the antecedents of the rule. In the case of data sets with massive domain sizes, the number of possible combinations of attributes (even for such low-dimensional combinations) can be so large that the corresponding statistics cannot be maintained explicitly during intermediate computations. However, the sketch-based

method provides a unique technique for maintaining counts by creating *super-items* from different combinations of attribute values. Each super-item V containing a concatenation of the attribute value strings along with the dimension indices for which these strings belong to. Let the actual value-string corresponding to value i_r be $S(i_r)$, and let the dimension index corresponding to the item i_r be $dim(i_r)$. In order to represent the dimension-value combinations corresponding to items $i_1 \dots i_p$, we create a new string by concatenating the strings $S(i_1) \dots S(i_p)$ and the dimension indices $dim(i_1) \dots dim(i_p)$.

This new super-string is then hashed into the sketch table as if it is the attribute value for the special super-item V . For each of the k -classes, we maintain a separate sketch of size $w \cdot h$, and we update the sketch cells for a given class only when a data stream item of the corresponding class is received. *It is important to note that the same set of w hash functions are used for updating the sketch corresponding to each of the k classes in the data.* Then, we update the sketch once for each 1-dimensional attribute value for the d different attributes, and once for each of the super-items created by attribute combinations. For example, consider the case when we wish to determine discriminatory combinations or 1- or 2-dimensional attributes. We note that there are a total of $d + d \cdot (d - 1)/2 = d \cdot (d + 1)/2$ such combinations. Then, the sketch for the corresponding class is updated $L = d \cdot (d + 1)/2$ times for each of the attribute-values or combinations of attribute-values. In general, L may be larger if we choose to use even higher dimensional combinations, though for cases of massive domain sizes, even a low-dimensional subspace would have a high enough level of specificity for classification purposes. This is because of the extremely large number of combinations of possibilities, most of which would have very little frequency with respect to the data stream. For all practical purposes, we can assume that the use of 2-dimensional or 3-dimensional combinations provides sufficient discrimination in the massive-domain case. We further note that L is dependent only on the dimensionality and is independent of the domain size along any of the dimensions. For modest values of d , the value of L is typically much lower than the number of possible combinations of attribute values.

The sketch-based classification algorithm has the advantage of being simple to implement. For each of the classes, we maintain a separate sketch table with $w \cdot d$ values. Thus, there are a total of $w \cdot d \cdot k$ cells which need to be maintained. When a new item from the data stream arrives, we update $L \cdot w$ cells of the i th sketch table. Specifically, for each item or super-item we update the count of the corresponding w cells

in the sketch table by one unit. The overall approach for updating the sketch table is illustrated in Figure 1. The input to the algorithm is the data stream \mathcal{D} , the maximum dimensionality of the subspace combinations which are tracked and the number of classes in the data set.

2.1 Determining Discriminative Combinations of Attributes

The key to using the sketch-based approach effectively is to be able to efficiently determine discriminative combinations of attributes. While one does not need to determine such combinations *explicitly* in closed form, it suffices to be able to *test* whether a *given* combination of attributes is discriminative. As we will see, this suffices to perform effective classification of a given test instance. Let us consider the state of the data stream, when N records have arrived so far. The number of data streams records received from the k different classes are denoted by $N_1 \dots N_k$, so that we have $\sum_{i=1}^k N_i = N$.

We note that most combinations of attribute values have very low frequency of presence in the data stream. Here, we are interested in those combinations of attribute-values which have high relative presence in one class compared to the other classes. We note that we are referring to high *relative presence for a given class* in combination with a moderate amount of absolute presence. For example, if a particular combination of values occurs in 0.5% of the records corresponding to the class i , but it occurs in less than 0.1% of the records belonging to the other classes, then the relative presence of the combination in that particular class is high enough to be considered significant. Therefore, we will define the discriminative power of a given combination of values (or super-item) V . Let $f_i(V)$ denote the fractional presence of the super-item V in class i , and $g_i(V)$ be the fractional presence of the super-item V in all classes other than i . In order to identify classification behavior specific to class i , we are interested in a super-item V , if $f_i(V)$ is significantly greater than $g_i(V)$. Therefore, we will define the discriminatory power of a given super-item V .

DEFINITION 1. *The discriminatory power $\theta_i(V)$ of the super-item V is defined as the fractional difference in the relative frequency of the attribute-value combination V in class i versus the relative presence in classes other than i . Formally, the value of $\theta_i(V)$ is defined as follows:*

$$(2.1) \quad \theta_i(V) = \frac{f_i(V) - g_i(V)}{f_i(V)}$$

Since we are interested only in items from which $f_i(V)$ is greater than $g_i(V)$ the value of $\theta_i(V)$ in super-items

of interest will lie between 0 and 1. The larger the value of $\theta_i(V)$, the greater the correlation between the attribute-combination V and the class i . A value of $\theta_i(V) = 0$ indicates no correlation, whereas the value of $\theta_i(V) = 1$ indicates perfect correlation. In addition, we have interested in those combinations of attribute values which occur in at least a fraction s of the records belonging to any class i . Such attribute value combinations are referred to as *discriminatory*. We formally define the concept of (θ, s) -discriminatory as follows:

DEFINITION 2. We define an attribute value-combination V as (θ, s) -discriminatory with respect to class i , if $f_i(V) \geq s$ and $\theta_i(V) \geq \theta$.

In other words, the attribute-value combination V occurs in at least a fraction s of the records belonging to class i , and has a discriminatory power of at least θ with respect to class i . Next, we will discuss an approach for the *decision problem* of testing whether or not a given attribute-value combination is discriminatory. We will leverage this approach for the classification process.

In order to estimate the value of $\theta_i(V)$, we need to estimate the value of $f_i(V)$ and $g_i(V)$. We note that the value of N_i is simply the sum of the counts in the table corresponding to the i th class divided by L . This is because we update exactly L sketch-table entries for each incoming record. In order to estimate $f_i(V)$ we take the minimum of the w hash cells to which V maps for various hash functions in the sketch table corresponding to class i . This is the approach used for estimating the counts as discussed in [6]. The value of $g_i(V)$ can be estimated similarly, except that we compute a sketch table which contains the sum of the (corresponding cell-specific) counts in the $(k-1)$ classes other than i . We note that this composite sketch table represents the other $(k-1)$ classes, since we use the same hash-function on each table. As in the previous case, we estimate the value of $g_i(V)$ by using the minimum cell value from all the w cells to which V maps. The value of $\theta_i(V)$ can then be estimated from these values of $f_i(V)$ and $g_i(V)$. We make the following observation:

LEMMA 2.1. With probability at least $(1-\delta)$, the values of $f_i(V)$ and $g_i(V)$ are respectively over-estimated to within $L \cdot \epsilon$ of their true values when we use sketch tables with size $w = \lceil \ln(1/\delta) \rceil$ and $h = \lceil e/\epsilon \rceil$.

Proof. The above result is a direct corollary of the results presented in [6], since each incoming record adds a count of at least L to the hash table. The values of $f_i(V)$ and $g_i(V)$ are never under-estimated since the counts are non-negative. Note that $f_i(V)$ and $g_i(V)$ are fractional counts obtained by dividing the counts by N_i

and $(N - N_i)$ respectively. Since the true counts are over-estimated to within $N_i \cdot l \cdot \epsilon$ and $(N - N_i) \cdot L \cdot \epsilon$ respectively according to [6], the result follows.

Next, we will try to compute the accuracy of estimation of $\theta_i(V)$. Note that we are only interested in those attribute-combinations V for which $f_i(V) \geq s$ and $f_i(V) \geq g_i(V)$, since such patterns have sufficient statistical counts and are also discriminatory with $\theta_i(V) \geq 0$.

LEMMA 2.2. Let $\beta_i(V)$ be the estimated value of $\theta_i(V)$ for an attribute-combination V with fractional selectivity at least s and $f_i(V) \geq g_i(V)$. Let ϵ be chosen such that $\epsilon' = \epsilon \cdot L/s \ll 1$. With probability at least $1 - \delta$, it is the case that $\beta_i(V) \leq \theta_i(V) + \epsilon'$.

Proof. Let $f'_i(V)$ be the estimated value of $f_i(V)$ and $g'_i(V)$ be the estimated value of $g_i(V)$. Then we have:

$$(2.2) \quad \beta_i(V) = 1 - g'_i(V)/f'_i(V)$$

Note that $\beta_i(V)$ is as large as possible when $g'_i(V)$ is as small as possible (as close to $g_i(V)$ as possible) and $f'_i(V)$ is as large as possible. According to Lemma 2.1, we know that with probability at least $1 - \delta$, $f'_i(V)$ is no larger than $f_i(V) + \epsilon \cdot L$.

$$\begin{aligned} \beta_i(V) &\leq 1 - \frac{g_i(V)}{(f_i(V) + \epsilon \cdot L)} \\ &= 1 - \frac{g_i(V)}{f_i(V) \cdot (1 + \epsilon \cdot L/f_i(V))} \\ &\leq 1 - \frac{g_i(V)}{f_i(V) \cdot (1 + \epsilon \cdot L/s)} \\ &= 1 - \frac{g_i(V)}{f_i(V) \cdot (1 + \epsilon')} \\ &\approx 1 - \frac{g_i(V) \cdot (1 - \epsilon')}{f_i(V)} \quad (\text{since } \epsilon' \ll 1) \\ &= \theta_i(V) + \epsilon' \cdot f_i(V)/g_i(V) \\ &\leq \theta_i(V) + \epsilon' \quad (\text{since } f_i(V) \leq g_i(V)) \end{aligned}$$

This completes the proof.

Next, we will examine the case when the value of $\theta_i(V)$ is under-estimated.

LEMMA 2.3. Let $\beta_i(V)$ be the estimated value of $\theta_i(V)$ for an attribute-combination V with fractional selectivity at least s and $f_i(V) \geq g_i(V)$. Let ϵ be chosen such that $\epsilon' = \epsilon \cdot L/s \ll 1$. With probability at least $1 - \delta$, it is the case that $\beta_i(V) \geq \theta_i(V) - \epsilon'$.

Proof. As in the previous lemma, let $f'_i(V)$ be the estimated value of $f_i(V)$ and $g'_i(V)$ be the estimated value of $g_i(V)$. Then we have:

$$(2.3) \quad \beta_i(V) = 1 - g'_i(V)/f'_i(V)$$

Data Dimensionality	Max. Pattern Dimensionality	L	$\epsilon = \epsilon' \cdot s/L$	# Cells= $\lceil e \cdot \ln(1/\delta)/\epsilon \rceil$	Storage Required	Storage $\epsilon' X 20$
10	2	55	$10^{-4}/55$	6884350	26.3 MB	1.315 MB
10	3	175	$10^{-4}/175$	$21.9 * 10^6$	83.6 MB	4.18 MB
20	2	210	$10^{-4}/210$	$26.3 * 10^6$	100.27 MB	5.01 MB
20	3	1920	$10^{-4}/1920$	$24 * 10^7$	916.85 MB	45.8 MB
50	2	1275	$10^{-4}/1275$	$15.96 * 10^6$	608.85 MB	30.5 MB
100	2	5050	$10^{-4}/5050$	$63.21 * 10^7$	2.41 GB	120.5 MB
200	2	20100	$10^{-4}/20100$	$25.16 * 10^8$	9.6 GB	480 MB

Table 1: Storage Req. of sketch table for different data and pattern dimensionalities ($\epsilon' = 0.01$, $\delta = 0.01$, $s = 0.01$)

Note that $\beta_i(V)$ is as large as possible when $f'_i(V)$ is as small as possible (as close to $f_i(V)$ as possible) and $g'_i(V)$ is as large as possible. According to Lemma 2.1, we know that with probability at least $1 - \delta$, $g'_i(V)$ is no larger than $g_i(V) + \epsilon \cdot L$. Therefore, with probability at least $(1 - \delta)$, we have: Therefore, with probability at least $(1 - \delta)$, we have:

$$\begin{aligned}
\beta_i(V) &\geq 1 - \frac{g_i(V) + \epsilon \cdot L}{f_i(V)} \\
&= 1 - \frac{g_i(V) \cdot (1 + \epsilon \cdot L/f_i(V))}{f_i(V)} \\
&\geq 1 - \frac{g_i(V) \cdot (1 + \epsilon \cdot L/s)}{f_i(V)} \\
&= 1 - \frac{g_i(V) \cdot (1 + \epsilon')}{f_i(V)} \\
&= \left(1 - \frac{g_i(V)}{f_i(V)}\right) - \epsilon' \cdot \frac{g_i(V)}{f_i(V)} \\
&= \theta_i(V) - \epsilon' \cdot \frac{g_i(V)}{f_i(V)} \\
&\geq \theta_i(V) - \epsilon' \quad (\text{since } f_i(V) \leq g_i(V))
\end{aligned}$$

This completes the proof.

We can combine the results of Lemma 2.2 and 2.3 to conclude the following:

LEMMA 2.4. *Let $\beta_i(V)$ be the estimated value of $\theta_i(V)$ for an attribute-combination V with fractional selectivity at least s and $f_i(V) \geq g_i(V)$. Let ϵ be chosen such that $\epsilon' = \epsilon \cdot L/s \ll 1$. With probability at least $1 - 2 \cdot \delta$, it is the case that $\beta_i(V) \in (\theta_i(V) - \epsilon', \theta_i(V) + \epsilon')$.*

This result follows from the fact each of the inequalities in Lemmas 2.2 and 2.3 are true with probability at least $1 - \delta$. Therefore, both inequalities are true with probability at least $(1 - 2 \cdot \delta)$. Another natural corollary of this result is that any pattern which is truly (θ, s) -discriminatory will be discovered with probability at least $(1 - 2 \cdot \delta)$ by using the sketch based approach

Algorithm *SketchClassify*(Test Data Point: \bar{V} , NumClasses: k , MaxDim: r)

begin

Determine all L value-combinations specific to test instance;

Use sketch-based approach to determine which value-combinations are (θ, s) -discriminatory;

Add 1 vote to each class for which a pattern is (θ, s) -discriminatory;

Report class with largest number of votes;

end

Figure 2: Classification (Testing Phase)

in order to determine all patterns which are at least $(\theta - \epsilon', s \cdot (1 - \epsilon'))$ -discriminatory. Furthermore, the discriminatory power of such patterns will not be over- or under-estimated by an inaccuracy greater than ϵ' .

The process of determining whether a super-item V is (θ, s) requires us to determine $f_i(V)$ and $g_i(V)$ only. The value of $f_i(V)$ may be determined in a straightforward way by using the sketch based technique of [6] in order to determine the estimated frequency of the item. We note that $f_i(V)$ can be determined quite accurately since we are only considering those patterns which have a certain minimum support. The value of $g_i(V)$ may be determined by adding up the sketch tables for all the other different classes, and then using the same technique.

At this point, we will provide a practical feel of the space requirements of using such an approach. What are the space requirements of the approach for classifying large data streams under a variety of practical parameter settings? Consider the case where we have a 10-dimensional massive domain data set which has at least 10^7 values over each attribute. Then, the number of possible 2-dimensional and 3-dimensional value combinations are $10^{14} * 10 * (10 - 1)/2$ and $10^{21} * 10 * (10 - 1) * (10 - 2)/6$ respectively. We note that even the former require translates to an order of magnitude of about 4500 tera-bytes. Clearly, the intermediate-

space requirements for aggregation based computations of many standard classifiers are beyond most modern computers. On the other hand, for the sketch-based technique, the requirements continue to be quite modest. For example, let us consider the case where we use the sketch based approach with $\epsilon' = 0.01$ and $\delta = 0.01$. Also, let us assume that we wish to have the ability to perform discriminatory classification on patterns with specificity at least $s = 0.01$. We have illustrated the storage requirements for data sets of different dimensionalities in Table 1. While the table is constructed for the case of $\epsilon' = 0.01$, the storage numbers in the last column of the table are illustrated for the case of $\epsilon' = 0.2$. We will see later that the use of much large values of ϵ' can provide effective and accurate results. It is clear that the storage requirements are quite modest, and can be held in main memory in most cases. For the case of stringent accuracy requirements of $\epsilon' = 0.01$, the high dimensional data sets may require a modest amount of disk storage in order to capture the sketch table. However, a practical choice of $\epsilon = 0.2$ always results in a table which can be stored in main memory. These results are especially useful in light of the fact that even data sets of small dimensionalities cannot be effectively processed with the use of traditional methods.

2.2 Classification with Sketch Table In this section, we will show how to leverage the sketch table in order to perform the classification for a given test instance \bar{Y} . The first step is to extract all the L patterns in the test instance with dimensionality less than r . Then, we determine those patterns which are (θ, s) -discriminatory with respect to at least one class. The process of finding (θ, s) -discriminatory patterns has already been discussed in the last section. We use a voting scheme in order to determine the final class label. Each pattern which is (θ, s) -discriminatory constitutes a vote for that class. The class with the highest number of votes is reported as the relevant class label. The overall procedure is reported in Figure 2.

2.3 Optimizations The approach discussed above can be optimized in several ways. One key bottle neck is the case when the data dimensionality is on the higher side, and the value of L becomes large. For each training instance all L patterns need to be processed. This can reduce the computational efficiency of processing each record. However, for a fast data stream it may not be necessary to process all the patterns in a given record. Instead, we may use only a random sample of $L' < L$ patterns from each training instance. In such cases, both the time and space-complexity is reduced by a factor of L'/L . We note that the use of sampling

results in a *sampled specificity* of s' for a given pattern, which is slightly different from s . Since the sketch-based approach tries to estimate this sampled specificity, an additional inaccuracy of $|(s - s')|$ arises from the use of this approach. As long as the value of $|(s' - s)|/s$ is small, the sampling based approach estimates the true value of s quite effectively. It can be shown that the standard deviation of s'/s for a pattern in class i by a random sample of size $N_i \cdot L'/L$ is given by $O(\sqrt{L}/\sqrt{L' \cdot N_i})$. For large values of N_i , this standard-deviation is almost zero. Furthermore, this inaccuracy reduces with progression of the data stream, since the value of N_i increases continually over time. Even for small values of L' , the value of $O(\sqrt{L}/\sqrt{L' \cdot N_i})$ is likely to reduce rapidly in most cases. For example, in the case of Table 1, a choice of $L' = 100$ for the most challenging case of 200-dimensional data would result in a standard deviation of about 10^{-2} , once N_i exceeds two million. The requirement is much lower for all the other cases in the table. It is reasonable to expect N_i to be large, since our approach is designed for the case of fast data streams. Thus, the large volume of the stream works in favor of the accuracy of sampling the intermediate patterns.

2.4 Computational and Space Complexity Let us examine the case where we wish to track patterns with specificity s . Also assume that we wish to obtain an accuracy within ϵ' with probability at least $(1 - \delta)$, and the number of patterns tracked for each record is L . Then, the size of the sketch table is given by $O(e \cdot \ln(1/\delta) \cdot L/\epsilon')$. In the event that only L' samples are used from each record, the size of the corresponding sketch table is $O(e \cdot \ln(1/\delta) \cdot L'/\epsilon')$. The algorithm for updating the sketch table requires $O(\ln(1/\delta) \cdot L)$ operations for each record. As in the previous case, if we sample only L' patterns from each record, this can be reduced to $O(\ln(1/\delta) \cdot L')$ operations for each record.

3 Experimental Results

In this section, we will discuss the experimental results of the sketch-based technique. All results were tested on an IBM T41p laptop running Windows XP Professional, a CPU speed of 1.69 GHz and 1 GB of main memory.

We note that most known data stream classification methods are designed for the continuous domain, and this does not work well for the case of large discrete domain sizes. Furthermore, the massive size of the domain ensures that straightforward modifications of stream- and other known classifiers cannot be used in this case. The only classifier which can be implemented in a limited way is the nearest neighbor classifier. Even in the case of the nearest neighbor classifier, one cannot

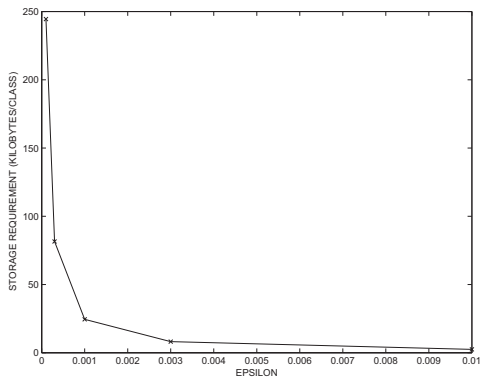


Figure 3: Storage Requirement with increasing ϵ for $\delta = 0.01$ (Analytical Curve which plots $e \cdot \ln(1/\delta)/(512 \cdot \epsilon)$)

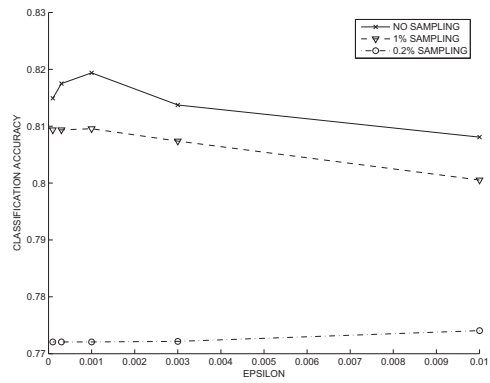


Figure 6: Classification Accuracy with increasing ϵ for data set IP2007.06.03

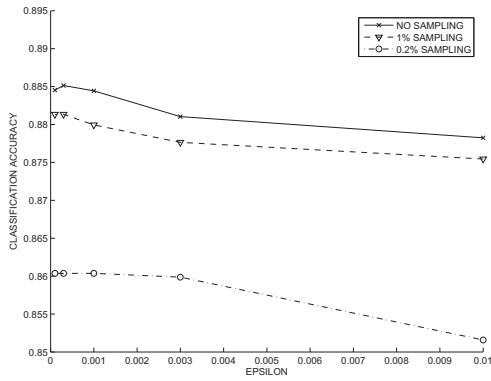


Figure 4: Classification Accuracy with increasing ϵ for data set IP2007.06.01

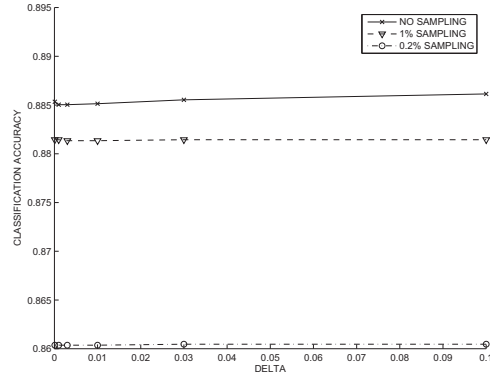


Figure 7: Classification Accuracy with increasing δ for data set IP2007.06.01

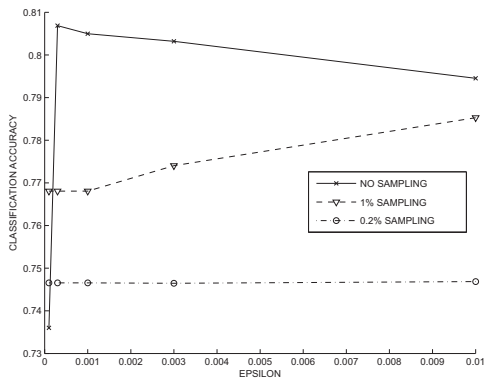


Figure 5: Classification Accuracy with increasing ϵ for data set IP2007.06.02

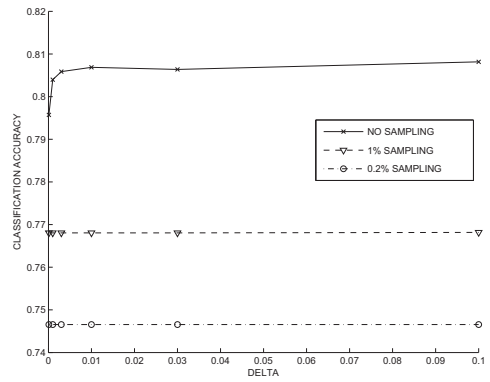


Figure 8: Classification Accuracy with increasing δ for data set IP2007.06.02

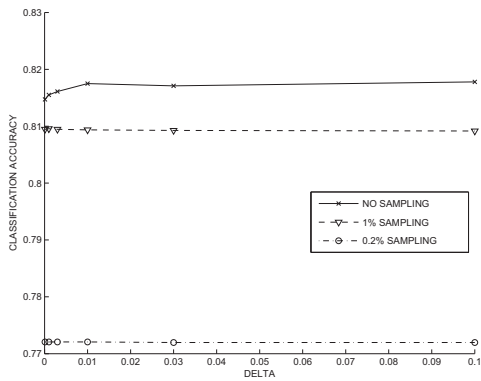


Figure 9: Classification Accuracy with increasing δ for data set IP2007.06.03

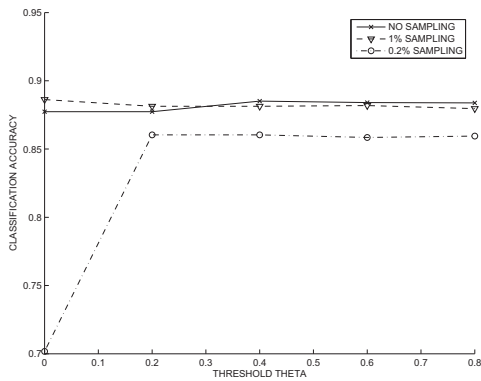


Figure 10: Classification Accuracy with increasing θ for data set IP2007.06.01

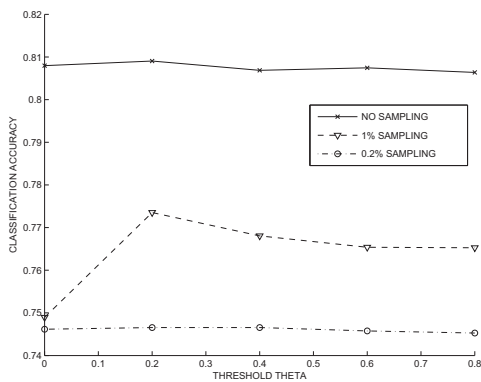


Figure 11: Classification Accuracy with increasing θ for data set IP2007.06.02

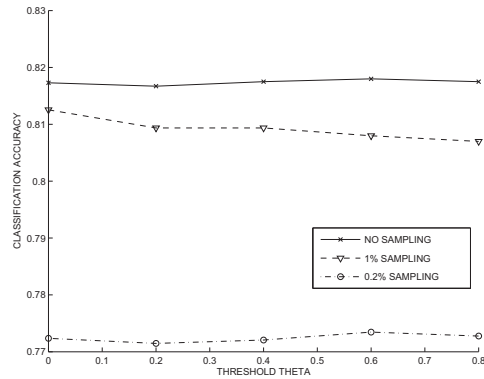


Figure 12: Classification Accuracy with increasing θ for data set IP2007.06.03

use the nearest neighbor over the entire training stream because of efficiency reasons. Therefore, we will show that when a truncated training data set is used (for both classifiers) in order to allow for implementation of the nearest neighbor classifier, our classification technique is significantly superior on the truncated data. Then, we will show that our technique is extremely robust and efficient on the full stream for a very broad range of parameter-settings. In fact, we will show that our technique is so insensitive to parameter-settings, that one can confidently pick parameters from a wide range without significantly affecting the accuracy of the technique. We will study the following: **(1)** Robustness of the technique for a variety of sketch table storage requirements, sampling rates, and parameter values. **(2)** Efficiency of sketch table construction for a variety of storage requirements, sampling rates, and parameter values.

We tested the algorithms on intrusion detection data sets from the log files of a large network server of IBM. These data sets contained alerts from different sensors. Each alert comprised several pieces of information including the source and destination IP addresses, sensor id, time stamp, and severity level. The alert also had a class label attached to it depending upon the nature of the alert. Three of the attributes corresponding to the source and destination IP addresses, and the sensor id are massive domain attributes. Clearly, it is infeasible to utilize known data stream classification methods for such domains from a space-efficiency perspective. It is also non-trivial to modify traditional classifiers for use with such a domain in the stream scenario. For our experimental testing, we used three data sets denoted by *IP2007.06.01*, *IP2007.06.02* and *IP2007.06.03* respectively. Each data stream consisted of a set of alerts received over a period of one day. The labels on the data sets correspond to the dates on which the corresponding

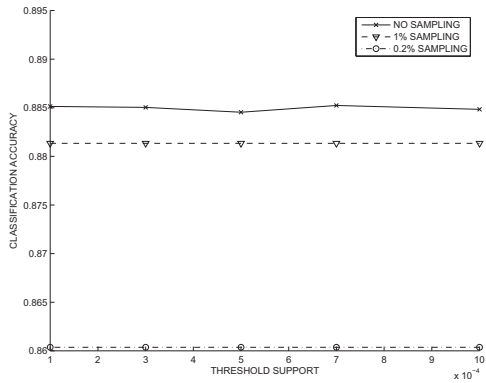


Figure 13: Classification Accuracy with increasing support for data set IP2007.06.01

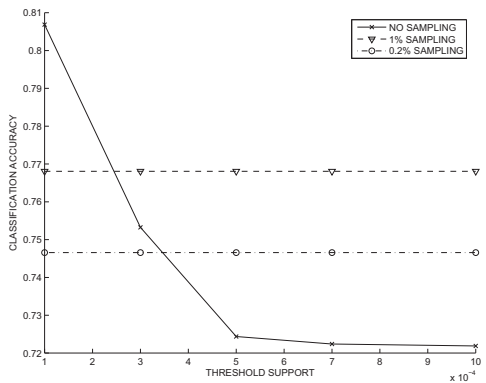


Figure 14: Classification Accuracy with increasing support for data set IP2007.06.02

alerts were collected.

We will first illustrate the effectiveness of the technique over a variety of parameter settings. Unless otherwise mentioned, the default values for different parameters were as follows: $\epsilon = 0.003$, $\delta = 0.01$, $samplefactor = 1$, $\theta = 0.4$, and $minsupport = 0.0001$. We note that this choice of parameters was deliberately picked in the middle of the ranges along which the algorithm was tested, and are not necessarily the optimal values. We will also show that the effectiveness of the technique is extremely high over a wide range of parameter values. We further note that only 81 Kilobytes of sketch table space is required for each class at the default value of the parameters. This is well within the limitations of most systems, even when there are a very large number of classes.

In Figure 3, we have illustrated the storage requirements for each class-specific sketch table (given by $e \cdot \ln(1/\delta) / (\epsilon \cdot 512)$ Kilobytes) with increasing values of ϵ for the range of parameters tested in the paper. The value of ϵ is illustrated on the X-axis, and the stor-

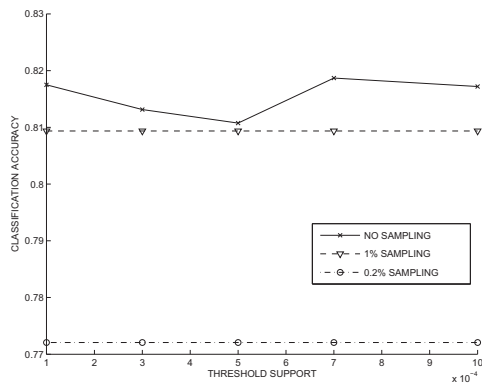


Figure 15: Classification Accuracy with increasing support for data set IP2007.06.03

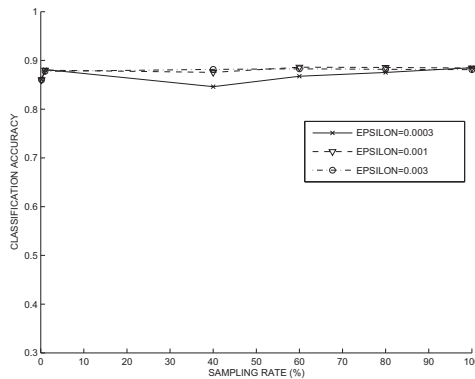


Figure 16: Classification Accuracy with increasing sampling rate for data set IP2007.06.01

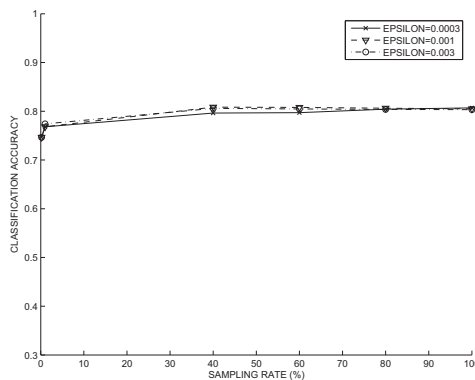


Figure 17: Classification Accuracy with increasing sampling rate for data set IP2007.06.02

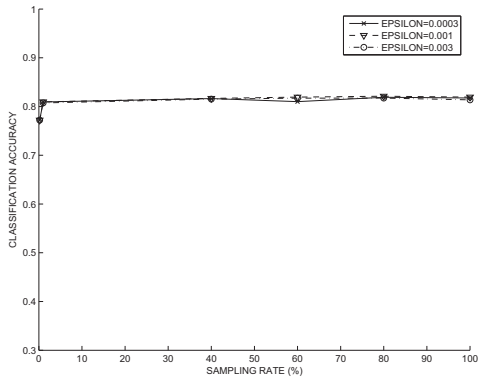


Figure 18: Classification Accuracy with increasing sampling rate for data set IP2007.06.03

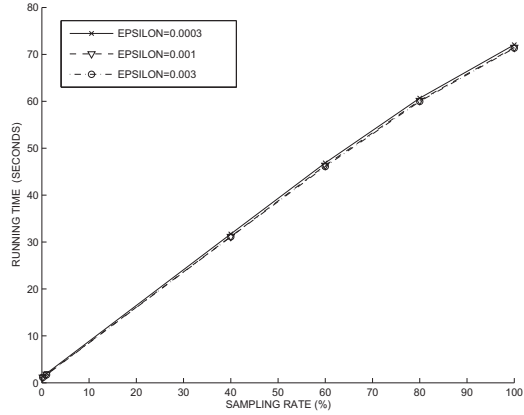


Figure 21: Running Time with increasing sampling rate for data set IP2007.06.03

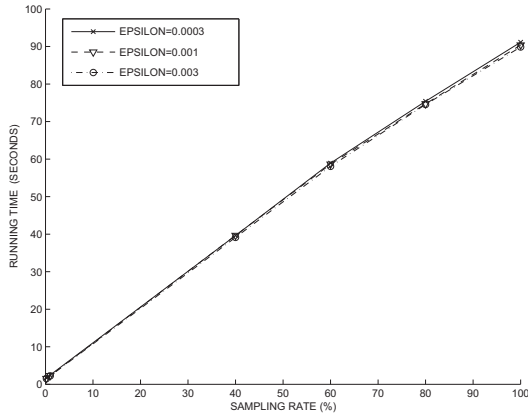


Figure 19: Running Time with increasing sampling rate for data set IP2007.06.01

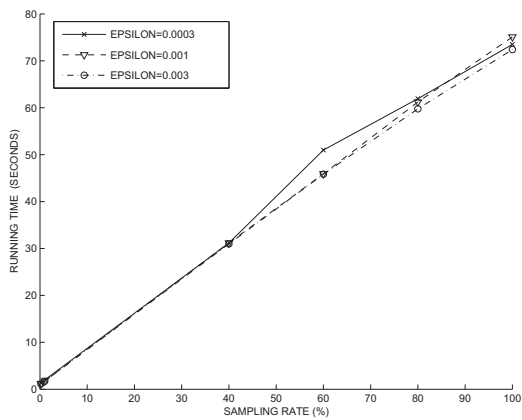


Figure 20: Running Time with increasing sampling rate for data set IP2007.06.02

Data Set	Sketch-Based (Accuracy)	Nearest Neighbor (Accuracy)
IP2007.06.01.c	79.23%	67.52%
IP2007.06.02.c	80.49%	70.86%
IP2007.06.03.c	81.31%	70.03%

Table 2: Comparison with Nearest Neighbor Classifier for Limited Case of Truncated Data Sets

age requirements are illustrated on the Y-axis. This analytical curve makes the assumption that 2 bytes are required for each entry in the sketch table. In each case, the sketch table contains only a few kilobytes, and can therefore be implemented even on the simplest of hardware.

We compared the sketch-based classifier to the nearest neighbor classifier. We note that the nearest neighbor classifier cannot be used directly with the entire training data stream, since it is not practical to determine the nearest neighbor over the entire training data. Therefore, we used a sample of 10^5 points from the original training data stream in order to create truncated data sets. We refer to these truncated data sets as *IP2007.06.01.c*, *IP2007.06.02.c* and *IP2007.06.03.c*, with the last two characters “.c” representing the truncated nature of the data set. We used these truncated data sets for both classifiers, and even for the truncated case, the nearest neighbor classifier was more than four times as slow as the sketch-based classifier. In Table 2, we have illustrated the accuracy of the two classifiers for the three data sets. In each case, the sketch-based classifier had an accuracy which was typically at least 10% higher than the nearest neighbor classifier. This is in spite of the fact that the nearest neighbor classifier was at least four times as slow as the sketch-based classifier. Thus, while our limited data scenario makes the

nearest neighbor classifier implementable, it continues to be less accurate than the sketch-based classifier.

Next, we will illustrate the robustness of our classifier on the entire training stream, a case for which our classifier is the only available technique because of its ability to summarize the classification behavior of the massive domain in a compact way. We will show that the sketch-based classifier continues to maintain its high accuracy over the entire data stream over a very wide range of input parameters. This illustrates the technique is extremely robust to the size of the data, as well as the parameter settings which are used for classification. In Figures 4, 5, and 6, we have illustrated the classification accuracy of the technique with increasing value of the parameter ϵ . On the X -axis, we have illustrated the parameter ϵ , whereas the classification accuracy is illustrated on the Y -axis. The other parameters were set at the default values specified at the beginning of this section. We have illustrated the classification accuracy with different levels of sampling. The solid line corresponds to the original algorithm where sampling is not used. The other two lines correspond to varying levels of sampling. We have also shown curves for sampling levels of 0.2% and 1% respectively. As we will see, such levels of sampling lead to large speedup factors of 500 and 100 respectively. In all cases, the classification accuracy of the technique was well above 75%. In cases where sampling was not used the classification accuracy was over 80% over a large range of the parameter ϵ in all three data sets. This is an extremely high level of accuracy since the data set contained well over a 100 classes, which makes the classification process extremely difficult. We further note that even when the value of $\epsilon = 0.01$ was used (which corresponds to a very low storage requirement of 2.44KB per class), the classification accuracy was essentially unchanged in most cases from more refined versions of the sketch table. While such large values of ϵ no longer provide theoretical guarantees, the results show that the accuracy is maintained from a practical point of view. This is because even when there are collisions in the sketch table, the overall bias in the class distribution of individual cells is sufficient for maintaining the accuracy of the classification process. The other observation is that even though a very low rate of sampling is used, the classification accuracy continues to be robust. For such sampling rates, we will see that the training stream can be processed very efficiently.

In Figures 7, 8, and 9, we have illustrated the classification accuracy of the approach with increasing value of the parameter δ . The parameter δ is illustrated on the X -axis, and the classification accuracy is illustrated on the Y -axis. The other parameters were set at the default

values specified at the beginning of this section. As in the previous cases, the algorithms were tested over the three different sampling rates. In each case, the classification accuracy is extremely stable for different values of the parameter δ on all three data sets. Furthermore, the techniques did not degrade significantly for the use of lower sampling rate.

We also tested the robustness of the technique for different values of the parameter θ . The other parameters were set to the default values specified at the beginning of this section. The results are illustrated in Figures 10, 11 and 12 for the three data sets. In each cases, the accuracy was stable for different values of the parameter θ , though the accuracy sometimes dropped for very low values of the parameter. We further note that the default choice of $\theta = 0.4$ was picked along the middle of the range, and was not necessarily tailored to optimizing the accuracy over the other charts. Nevertheless, the stability in the accuracy values ensured that the results in other charts continued to be quite accurate.

In Figures 13, 14, and 15, we have illustrated the accuracy of the technique for increasing values of the support parameter. All other parameters were set to their default values specified earlier. The minimum support threshold is denoted on the X -axis, and the classification accuracy is denoted on the Y -axis. The only exception is Figure 14 in which the classification accuracy dips at high values of the support. This dip happens only for the case when the entire data stream is used. The reason for this is that at large values of the support, many spurious patterns are created because of collisions in the underlying sketch table. The collisions are fewer when sampling is used, and the classification accuracy continues to be robust for the sampled data. Thus, this is one of the interesting cases in which the sampled data actually provided higher accuracy because of fewer collisions in the sketch table.

We also tested the classification accuracy and the efficiency with increasing sampling rate. In this case, we used three different values of ϵ corresponding to 0.0003, 0.001 and 0.003 respectively. The results are illustrated in Figures 16, 17, and 18 respectively. The sampling rates are illustrated in the X -axis, and the classification accuracy is illustrated on the Y -axis. All other parameters were set to their default values. An interesting observation is that the classification accuracy is maintained even at very low sampling rates. In most data sets, the reduction in classification accuracy from the use of 1% sampling all the way to 100% sampling was less than 1%.

We also tested the running time of the method for different sampling rates. The running time was

defined in terms of the portion of the time required for training on the entire data stream. The sampling rate is illustrated on the X -axis, and the running time is illustrated on the Y -axis. As in the previous case, we used the same three settings for the parameter ϵ . Thus, we can examine the behavior of different sampling rates over different sketch table sizes. The results are illustrated in Figures 19, 20, and 21 respectively. The sampling rate is illustrated on the X -axis, and the overall running time was illustrated on the Y -axis. In each case, it is clear that the running time scaled linearly with the sampling rate. This is important because it means that we can scale up the running time by large factors by using low sampling rates such as 1%. For example, the use of 1% sampling typically requires less than 5 seconds of processing on each of the three data streams. This translated to a processing rate of over 10^5 records per second in the sampled case. As is evident from our earlier charts, such sampling rates did not significantly reduce the classification accuracy of the method. Thus, robust classification results can be obtained very efficiently with the use of the sketch-based technique.

4 Conclusions and Summary

In this paper, we presented a first approach for classification of massive-domain data streams. Such data domains cannot be handled with traditional classifiers because of the fact that the intermediate computations are computationally and storage intensive. This problem is further exacerbated in the case of a data stream, because of the one-pass constraint. The use of a sketch based approach alleviates this problem since it is able to track the *relevant attribute* combinations in an effective way in very low storage requirements. The technique is also extremely efficient, since only a few updates to the sketch table are required for each record. If desired, the technique can be made substantially more efficient with the use of sampling. Our experimental results show that the technique is extremely stable and robust over different choices of the input parameters and sampling rates.

References

- [1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, *A Framework for On-Demand Classification of Evolving Data Streams*, IEEE Transactions on Knowledge and Data Engineering, 18(5), (2006), pp. 577–589.
- [2] C. C. Aggarwal, *A Framework for Clustering Massive-Domain Data Streams*, ICDE Conference, (2009) pp. 102–113.
- [3] C. C. Aggarwal, *Data Streams: Models and Algorithms*, Springer, (2007).
- [4] L. Brieman, J. Friedman, and C. Stone, *Classification and Regression Trees*, Chapman and Hall, (1984).
- [5] W. Cohen, *Fast Effective Rule Induction*, ICML Conference, (1995), pp. 115–123.
- [6] G. Cormode, and S. Muthukrishnan, *An Improved Data-Stream Summary: The Count-min Sketch and its Applications*, Journal of Algorithms, 55(1), (2005), pp. 58–75.
- [7] P. Domingos, and G. Hulten, *Mining High-Speed Data Streams*, ACM KDD Conference, (2000) pp. 71–80.
- [8] W. Fan, *Systematic data selection to mine concept-drifting data streams*, ACM KDD Conference, (2004), pp. 128–137.
- [9] J. Gama, R. Rocha, and P. Medas, *Accurate Decision Trees for Mining High-Speed Data Streams*, ACM KDD Conference, (2003), pp. 523–528.
- [10] G. Hulten, L. Spencer, and P. Domingos, *Mining Time-Changing Data Streams*, ACM KDD Conference, (2001), pp. 97–106.
- [11] R. Jin, and G. Agrawal, *Efficient Decision Tree Construction on Streaming Data*, ACM KDD Conference, (2003), pp. 571–576.
- [12] B. Liu, W. Hsu, and Y. Ma, *Integrating Classification and Association Rule Mining*, ACM KDD Conference, (1998), pp. 80–86.
- [13] J. R. Quinlan, *C4.5: Programs in Machine Learning*, Morgan-Kaufmann Inc, (1993).
- [14] H. Wang, W. Fan, P. S. Yu, and J. Han, *Mining Concept-Drifting Data Streams using Ensemble Classifiers*, ACM KDD Conference, (2003), pp. 226–235.