Computing Information Gain in Data Streams

Alec Pawling, Nitesh V. Chawla, and Amitabh Chaudhary
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN
{apawling,nchawla,achaudha}@cse.nd.edu

Abstract

Computing information gain in general data streams, in which we do not make any assumptions on the underlying distributions or domains, is a hard problem, severely constrained by the limitations on memory space. We present a simple randomized solution to this problem that is time and space efficient as well as tolerates a relative error that has a theoretical upper bound. It is based on a novel method of discretization of continuous domains using quantiles. Our empirical evaluation of the technique, using standard and simulated datasets, convincingly demonstrates its practicality and robustness. Our results include accuracy versus memory usage plots and comparisons with a popular discretization technique.

1 Introduction

Increasing number of real world applications now involve *data streams*; *e.g.*, applications in telecommunications, e-commerce, stock market tickers, fraud and intrusion detection, sensor networks, astronomy, biology, geography, and other sciences. These data streams, whether commercial or scientific, spatial or temporal, almost always contain valuable knowledge, but are simply too fast and too voluminous for it to be discovered by known techniques. The objective of modern practitioners is to find time and memory efficient ways of modeling and learning from the streaming data, albeit at the cost of some possible loss in accuracy.

There are special cases in which it is relatively easy to learn in data streams. One commonly used scenario is when the stream is assumed to be a random sample drawn from a stationary or a slowly shifting distribution, often called the "i.i.d. assumption". In such a situation, a reasonable sized sample of the data stream can be assumed to describe the overall distribution (and hence, the entire stream) quite accurately, and the problem reduces to learning from such a sample. Another common easy case is when the underlying domain(s) of the data values is discrete, either nominal or consisting of a few numerical values. In discrete domains, the data can be represented by simply counting the number of instances of each value. Often this simple representation is sufficient and the memory usage much less than the actual size of the stream.

In this paper we consider the problem of feature selection in data streams based on computing information gain. Feature selection is a essential component of classification based knowledge discovery, and using information gain for it is one of the most popular methods (*e.g.*, decision tree learning with C4.5 [12]). We would like to solve the problem in a general setting without making any of the previous simplifying assumptions; in particular, we make no assumption whatsoever about the underlying distribution of the data stream. The distribution, if any, can change rapidly and arbitrarily, even classes may suddenly appear and then disappear, as the stream zips by. Further, we do not restrict the type of data; it can be spatial, temporal, neither, or both. Given such a stream of examples, each consisting of a feature vector with values drawn from continuous or discrete domains, we want to be able to compute, at every point in the stream, the maximum possible information gain as if our set of examples was exactly the examples in the stream seen thus far. The constraints are that we see each example in the stream just once and we are allowed to store very few of the examples we see — at most an order polylogarithmic. In addition we can take at most an order polylogarithmic time in processing each example. These are the standard efficiency constraints on stream computations (see, *e.g.*, [1]).

Related Work There has been research in classification in data streams: either on learning a single classifier [7, 4] or a set of classifiers [13]. A classification technique such as decision trees includes the computing of information gain as a component. Hulten et al. [4, 9] build decision trees in data streams from nominal domains — and thereby also compute the information gain — under the i.i.d. assumption on a stationary or a slowly shifting distribution. They make a clever use of the Hoeffding bounds to decide when a sufficiently large sample has been observed and use that to create their tree. If they observe that the distribution is shifting, they change to a different tree that represents the new data. Gehrke et al. developed BOAT [7], an incremental decision tree algorithm that scales to large datasets, often requiring just 2 scans of the data (In streams we usually get just one). A noteworthy strength is that it can handle different splitting criteria. Street and Kim [13] show how to build ensembles of classifiers on blocks of data. They add subsequent classifiers only when the concept is not already encompassed by the previous set of classifiers. Wang et al. [14] also implement an ensemble based approach for streaming data; they weigh each classifier based on their accuracy on the data evolving over time.

Our contributions We give a simple time and space efficient randomized algorithm for computing information gain in streams consisting of data from discrete or continuous domains. We give strict theoretical bounds on the amount of possible error in our computation. The error can be reduced by relaxing the space constraint, allowing the user to choose the appropriate balance between error and efficiency. Our algorithm doesn't need to know anything about the domains in advance, not even the maximum or minimum values possible. It does need to have a reasonable estimate or an upper bound on the size of the entire stream (the number n) for our bounds to be valid. Our technique is based on an original method of discretization using quantiles in tandem with a previously known algorithm for maintaining approximate quantiles in data streams.

We demonstrate the utility of our technique through experiments using standard datasets under different streaming conditions. E.g., we simulate sudden changes in the underlying distribution as could be expected in temporal or sensor data. We also show results on a large dataset artificially generated to stress-test the algorithm. The error in the information gain we compute is well within our theoretical bounds. In addition, the feature rankings we compute are very close to those computed using a *precise computation*, one that has no time or space constraints (essentially, can store all examples in the stream). We plot space and accuracy trade-off curves for the datasets, as well as compare our results with another popular discretization approach, equal interval width binnning [5].

In the following section we give a description of our technique that builds up starting from a naive solution. Following the algorithmic description we present our experiments, the setup and the results. Finally, we draw conclusions.

2 A Randomized Memory-Efficient Computation of Information Entropy

We want to compute the information gain for features in a stream of examples using a small amount of memory (relative to the number of examples seen). Specifically, at any point in the stream, for each feature X_i we would like to compute the value v_i such that a partition based on "$X_i \leq v_i$?"of the examples in the stream seen thus far results in the maximum gain in information. We would like to do this for both discrete as well as continuous features. In addition, we want to ensure that the maximum space we use, in terms of memory, is a function at most an order polylogarithmic in the number of examples. Note, we naturally assume that the number of examples in the stream is much much larger than either the number of features or the number of class labels.

In this section we present a randomized solution to the above problem. We start by looking at a naive approach that works reasonably well for nominal features but not for continuous ones. This leads to the well-investigated idea of discretization of continuous domains; unfortunately, the known techniques of discretization are not designed for data streams and any adaptation is not guaranteed to work well. We present an original technique of discretization that guarantees that only a small amount of error is introduced in the computation of information gain. More importantly, the technique can be extended to work on a stream using a small amount of memory, while introducing only a small additional increase in error — allowing us to achieve our objective.

Naive Approach Using Counters Let there be d features denoted by $\mathbf{X} = \{X_1, X_2, \ldots, X_d\}$ and c class labels denoted by the set $\mathbf{Y}$. Let $S = (\mathbf{x}_{(1)}, y_{(1)}), (\mathbf{x}_{(2)}, y_{(2)}), \ldots, (\mathbf{x}_{(t)}, y_{(t)}), \ldots$ be a stream of examples, in which $\mathbf{x}_{(t)}$ is a feature vector and $y_{(t)}$ is a class label. (We will drop the sequence subscript (t) when not required.) Let S_n be the set of n examples seen till time $t = n$.

Suppose we wanted to answer the following question for all n in a memory-efficient manner: What is the information gain by partitioning the examples in S_n at $X_i \leq v_i$? Let S_n^L be the subset of examples such that $X_i \leq v_i$ and S_n^R the rest of examples in S_n. Also, let $S_{n,y}$ denote the examples in S_n with class label y. Then the information entropy in S_n is

$$I(S_n) = \sum_{y \in \mathbf{Y}} \left(-\frac{|S_{n,y}|}{|S_n|} \lg \frac{|S_{n,y}|}{|S_n|} \right),$$

and the information gain is

$$\mathsf{gain}(S_n, X_i, v_i) = I(S_n) - \left[\frac{|S_n^L|}{|S_n|} I(S_n^L) + \frac{|S_n^R|}{|S_n|} I(S_n^R) \right].$$

We can compute the above for all n by simply maintaining two counters for each class label y: one to track $|S^L_{n,y}|$ and the other to track $|S^R_{n,y}|$. This takes very little memory: $2c$, where c is the number of class labels. If we wanted to answer the question for each feature, we can do the same for each of the d features, taking $2c \cdot d$ space, still much smaller than n.

The problem with the naive approach is that we actually want to answer a more involved question: For all n and for all X_i, what is the *maximum possible* information gain by partitioning the examples in S_n along *any* point on the domain of X_i? Now, even this question can be efficiently answered if each X_i is a nominal (discrete) feature. Let the number of possible values for X_i be m_i. Simply repeat the above approach for each possible value in a feature. The amount of space taken is $2c \sum_i m_i$, which is small enough if we assume that the m_i values are much smaller than n. This assumption, of course, cannot be made for continuous features.

Using Discretization of Continuous Features Modifying continuous feature spaces to behave like nominal feature spaces is common in machine learning algorithms, and the simplest method is to use a form of discretization. In this, a continuous domain is partitioned into a finite number of bins, each with a representative value; as a result, the feature naturally behaves like a nominal feature, although a small amount of error creeps in. There are many known mechanisms for discretization: unsupervised ones like equal interval width and equal frequency intervals [2], as well as supervised ones like entropy based discretization [6]. See Dougherty et al. [5] for a review and comparisons of a variety of discretization techniques. Among these known techniques, some perform better than others, but all lack an important characteristic: they are not designed to give an upper bound on the amount of relative error introduced into the information gain computation due to the discretization. Furthermore, it also not know how to extend these methods to compute information gain in streams. We solve both deficiencies by introducing the *quantile based discretization* method for computing information gain.

Quantile based Discretization The essence of quantile based discretization is simple: to compute the number of examples such that $X_i \leq v_i$, in a manner that bounds the fraction of error introduced by discretization, use "finer" bins in those parts where the number of satisfying examples is small, and "coarser" bins where the number of satisfying examples is large. Given a fixed number of bins to use, this introduces the minimum relative error. But it requires that the bin boundaries cannot be decided in advance; they are based on actual values taken by the examples in the feature domain. We implement this idea using quantiles. We first discuss how to use quantiles to calculate information gain. We then explain why we cannot use precise quantiles and present an approximate quantile structure that suits our purpose.

The ϕth quantile of a set of n elements, for $\phi \in [0, 1]$, is the $\lceil \phi n \rceil$th smallest item in the set. Hence $\phi = 1$ denotes the maximum element, and $0 < \phi \leq 1/n$ denotes the smallest element. We maintain our bin boundaries at the following quantiles: $\alpha^{-0}, \alpha^{-1}, \alpha^{-2}, \ldots$ for some $\alpha = (1+\varepsilon)$, where $\varepsilon > 0$. (It helps to think of α being a number like 2, although in practice ε is quite small.) Notice that the bins become finer as the rank denoted by the quantile decreases.

To estimate the number of examples with label y such that $X_i \leq v_i$ we use the quantiles for the set of examples with label y ordered by values of feature X_i. We first find the largest quantile with a value less than v_i. Let this be $\phi = \alpha^{-k}$. We then approximate the required number of examples by $n\alpha^{-k+0.5}$. This introduces a relative error of at most $1 \pm \varepsilon/2$ in computing the number of examples.

To compute the entropy in a partition at $X_i \leq v_i$ we repeat the above method for each $y \in Y$, using the estimates of the number of examples to estimate the entropy in the left side partition. It can be shown that this estimate has a relative error at most $1 \pm 7\varepsilon$. For the right partition, we can compute the number of examples with label y by simply subtracting the number in the left from the total number of examples with label y. This number can, however, have a relative error more than $1 \pm 7\varepsilon$ if α^{-k} happens to be larger than $1/2$. To avoid this, we use another set of quantiles, $1 - \alpha^{-0}, 1 - \alpha^{-1}, 1 - \alpha^{-2}, \ldots$, to compute the number of examples in that situation.

Now if we want to find the value v_i that results in the minimum entropy, we perform the above computation for each value in the set of quantiles. Let v_i^* be the required best-split value and let the entropy by partitioning at $X_i \leq v_i^*$ be I_i^*. We can guarantee that one of the quantiles, α^k, will have a rank that is at most an α factor away from that of v_i^* and the entropy estimated at quantile α^k is at most a factor

$$1 \pm 7\varepsilon \pm O(\varepsilon^2)$$

away from I_i^*. This error bound on the entropy doesn't translate into an error bound on the information gain if the gain turns out to be very small compared to the entropy. If, however, we know that the true value $\mathsf{gain}(S, X_i)$ is reasonably large, say, larger than $(1/g) \cdot I_i^*$, where $1/g > 0$ then we can guarantee

$$\frac{|\mathsf{gain}_{\text{disc}}(S, X_i) - \mathsf{gain}(S, X_i)|}{\mathsf{gain}(S, X_i)} \leq 7\varepsilon g + o(\varepsilon).$$

Note that feature selection in a stream is interesting only when the information gain is reasonably large. We can easily repeat this process for each feature and compute the maximum possible information gain over all features with bounded relative error. In doing so, we take $O(cd \cdot (1/\varepsilon) \log n)$ space.

Unfortunately, in order to compute quantiles for values in a data stream precisely, either the values must arrive in sorted order or $\Omega(n)$ values must be stored [11]. The former is an unreasonable assumption in a stream and the latter sets us back to square one. Fortunately, approximate quantiles for values in a stream can be computed rather efficiently, and that makes quantile based discretization particularly useful.

Approximate Quantiles in a Data Stream For $\phi \leq 1/2$, a δ-approximate ϕ quantile of a set of n elements is an element that lies in position between $\lceil n\phi(1-\delta)\rceil$ and $\lceil n\phi(1+\delta)\rceil$. Gupta and Zane [8] have shown how to maintain approximate quantiles for values in a data stream using $O(\log^2 n/\delta^3)$ space. They use a randomized data structure that, on being queried, gives a correct δ-approximate ϕ quantile with high probability, *i.e.*, at least $1-1/n$. We give a brief description of their technique below.

Gupta and Zane use randomized samplers to maintain each quantile. Suppose we want to maintain the $\frac{\beta^i}{n}$th quantile. We sample each value in the data stream with a probability $\frac{T}{\beta^i}$ and keep the smallest T values, where T is a reasonable small number. By doing this, we expect the rank of the largest item in the sampler to be β^i, i.e. the $\frac{\beta^i}{n}$th quantile. If we choose T carefully, as a function of δ and n, we can actually ensure that with probability at least $1-1/n$ the largest item in the sampler is a δ-approximation of the quantile. The samplers do, however, need to know n (or an upper bound on n) in advance to ensure this.

To compute information gain we simply have such a sampler for each quantile for each feature-class pair. Using approximate quatiles, instead of precise ones, increases the relative error by just a small constant factor. Thus the information gain can be computed in a stream taking $O(cd \cdot log^2 n/\varepsilon^3)$ space.

3 Experimental Set-up

We ran various experiments to check the robustness of our approach. As we mentioned, we are interested in evaluating not only the precision of the information gain computation but also the memory savings in using the samplers that reflect the statistics and distribution of the data. We ran two different variations of experiments: in one we randomly selected the initial sample and then uniformly selected the same size samples from the data; in the other we purposely skewed the distribution of the data, such that the class distribution in each stream is different from the previous stream and even very different from the actual class distribution in the data. We did the latter by sorting all the examples by a feature value and then selecting samples to constitute the stream. We applied the former approach on all the datasets, while we applied the latter only on the artificial dataset for the proof of concept. We are in the process of including additional datasets in the study. We benchmark our streaming approach against regular equal interval width binning and (offline) precise computation of information gain using all the data.

We also used different values of ε in our experiments. A lower value of ε results in a more accurate computation but requires more memory. As ε increases, accuracy and memory usage both decrease. Both the size and the number of samplers used decreases as ε increases, and since the samplers are holding fewer values, the quality of the approximate decreases. We thus used $\varepsilon \in 0.25, 0.5, 0.75$.

3.1 Datasets

We used four datasets, as shown in Table 1, in our paper. Each one has varying characteristics, and three of the datasets are real-world while one is artificial. The artificial dataset has 1 million examples and 2 classes with a balanced distribution. The independent features in the artificial datasets are drawn by N(0,1). Random noise is added to all the features by N(0,0.1). The features are then rescaled and shifted randomly. The relevant features are centered and rescaled to a standard deviation of 1. The class labels are then assigned according to a linear classification from a random weight vector, drawn from N(0,1), using only the useful features, centered about their mean. The Can dataset was generated from the Can ExodusII data using the AVATAR [3] version of the Mustafa Visualization tool. The portion of the can being crushed was marked as "very interesting" and the rest of the can was marked as "unknown." A dataset of size 443,872 samples with 8,360 samples marked as "very interesting" was generated. The Covtype and letter datasets are from UCI repository [10]. The covtype dataset has 7 classes with a highly imbalanced distribution; we selected the 10 continuous features from the actual 54 features in the covtype data (the leftover 44 features are all binary). The letter dataset has 26 classes that are fairly balanced.

Dataset	Number of Examples	Number of Classes	Number of Features
Artificial	1,000,000	2	6
Covtype	581,012	7	10
Can	443,872	2	9
Letter	20,000	26	16

Table 1. Datasets.

We used the following variations for each dataset:

- Artificial Dataset: We randomly selected examples from the training set to constitute each stream. To evaluate biased and changing data distributions, we implemented two scenarios. As the first scenario, we sorted the examples by a feature and then sampled from the resulting data distribution. As the second scenario, we changed the class distribution in the different chunks of streaming data. The original data distribution is 50:50. However, we set up a streams of size 1000 such that class sizes alternate between 10:990 and 990:10 after every 200000 examples. The comparison benchmark for both is the precise information gain calculation that assumes every data point seen so far is stored.

- Covtype Dataset: We randomly selected examples from the training set to constitute each stream. This maintained the original distribution.

- Can Dataset: We randomly selected examples from the training set to constitute each stream. This maintained the original distribution.

- Letter Dataset: We randomly selected examples from the training set to constitute each stream. This maintained the original distribution.

3.2 Results

Tables 2 to 5 show the information gain by using our proposed quantile method, equal interval width binning, and the precise offline calculation. The features in the tables are sorted in decreasing order of information gain for the precise computation. For both our method and equal interval width binning, we calculated information gain on the streaming data. Each entry in the Table is the final information gain value once all the available data is streamed in. As is evident from the Tables, if the streams are uniformly randomly sampled from the available data, then both the equal interval width binning and our method achieve similar information gain values for the features as compared to the precise calculation. The key point is that the overall ranking of features is maintained. However, once we artificially changed the distribution of the data, by sorting on a feature value, the equal interval width binning computation of information gain significantly deteriorated in its performance. The quantile-based method for computing information gain still performs fairly well, and provides the same ranking to the features as by the precise method. Note that while the actual information gain is not exactly the same as precise calculation, it is the ranking of features that is more relevant for feature selection. So, at the end both the techniques will select the same top-ranked features, and the information gain values are highly comparable.

Figures 1 to 4 show the information trend for the best feature for all the datasets along with the specified assumptions, where the best feature was chosen from the complete offline evaluation. We wanted to evaluate the impact of streaming data on the best feature value. The benchmark to our approach is the precise computation, which assumes that every data element is stored as the stream arrives, thus requiring more memory. We expected the best feature to be the most sensitive to changes in the distribution of streaming data, as all the feature values and corresponding classes are not well represented in the first few segments. As evident from the Figures, the quantile-based approximation approach for all three ε values considered closely follows the trend of the precise computation. We noted in the Tables that the final feature rankings are same as precise, and the values are very close. And the Figures establish the trend that matches the precise computation. Figure 5 shows the information gain computation for the changing data distribution on the artificial dataset. Again, the quantile based method closely follows the precise calculation.

Figures 6 to 9 shows the memory requirement of the quantile based approach as compared to precise. Examining the memory requirements of calculating information gain precisely and with approximate quantiles for different values of ε provides insight into the scalability of our method. While using the approximate quantiles with $\varepsilon = 0.25$ requires significantly more memory than calculating information gain precisely for all four data sets, using approximate quantile with $\varepsilon = 0.5, 0.75$ require significantly less memory for the larger data sets. We expect the approximate quantile based approach to be more beneficial when the datasets are very large in the order of millions of records, as demonstrated by the artificial dataset. It is remarkable that the memory required by the quantile method is just 1/10th of the total memory requirement by the precise method for the much larger artificial dataset. We observe an order of magniture savings for both covtype and can datasets, which are moderately large for a streaming scenario. For the smaller letter dataset, with only 20K examples, the samplers require more memory to keep all the relevant statistics than what will be required for keeping the entire dataset. This is due to the fact that there are so many classes relative to the number of examples. There are not many more examples for each class than there are locations in the zeroth samplers, so the memory used to store values in the other samplers is wasted. Moreover, if the datasets are indeed that small, one can easily store the entire dataset and even recompute the information gain or re-build a classifier as the new stream arrives.

precise	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 0.75$	binning
0.4378	0.4474	0.4365	0.4378	0.4376
0.0499	0.0577	0.0568	0.0499	0.0499
0.0052	0.0096	0.0120	0.0052	0.0052
0.0013	0.0026	0.0032	0.0013	0.0013
0.0004	0.0022	0.0022	0.0004	0.0004
0.0000	0.0009	0.0008	0.0000	0.0000

Table 2. Information gain of all features in the artificial data set.

precise	$\varepsilon = 0.75$	binning
0.4378	0.4558	0.0000
0.0499	0.0724	0.0498
0.0052	0.0243	0.0052
0.0013	0.0243	0.0013
0.0004	0.0174	0.0004

Table 3. Information gain of all features in the artificial data set when examples appear in increasing order of the most significant feature.

precise	$\varepsilon = 0.5$	$\varepsilon = 0.75$	binning
0.2989	0.3011	0.3143	0.2986
0.0820	0.0866	0.0850	0.0819
0.0613	0.0622	0.0651	0.0612
0.0380	0.0368	0.0456	0.0379
0.0217	0.0248	0.0279	0.0217
0.0156	0.0189	0.0220	0.0156
0.0150	0.0166	0.0186	0.0150
0.0128	0.0161	0.0156	0.0128
0.0089	0.0127	0.0181	0.0089
0.0079	0.0140	0.0159	0.0079

Table 4. Information gain of all features in the covtype data set.

precise	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 0.75$	binning
0.3966	0.3966	0.3990	0.4014	0.3971
0.3824	0.3824	0.3805	0.3864	0.3828
0.3721	0.3721	0.3702	0.3661	0.3729
0.3706	0.3706	0.3709	0.3693	0.3711
0.3387	0.3397	0.3374	0.3320	0.3397
0.2930	0.2920	0.2916	0.2861	0.2928
0.2811	0.2811	0.2862	0.2836	0.2812
0.2530	0.2530	0.2494	0.2451	0.2536
0.2169	0.2169	0.2111	0.1927	0.2171
0.2006	0.2006	0.1976	0.1995	0.2005
0.1970	0.1970	0.2032	0.2029	0.1979
0.0693	0.0693	0.0719	0.0723	0.0690
0.0501	0.0501	0.0519	0.0620	0.0500
0.0480	0.0480	0.0534	0.0554	0.0478
0.0341	0.0341	0.0340	0.0346	0.0344
0.0038	0.0038	0.0038	0.0042	0.0038

Table 5. Information gain of all features in the letter data set.

precise	$\varepsilon = 0.25$	$\varepsilon = 0.5$	$\varepsilon = 0.75$	binning
0.0132	0.0137	0.0158	0.0124	0.0131
0.0129	0.0127	0.0150	0.0136	0.0129
0.0109	0.0111	0.0112	0.0115	0.0109
0.0035	0.0040	0.0043	0.0060	0.0035
0.0029	0.0032	0.0040	0.0034	0.0029
0.0020	0.0021	0.0029	0.0030	0.0020
0.0013	0.0016	0.0017	0.0019	0.0013
0.0012	0.0014	0.0017	0.0024	0.0012
0.0006	0.0007	0.0009	0.0011	0.0006

Table 6. Information gain of all features in the can data set.

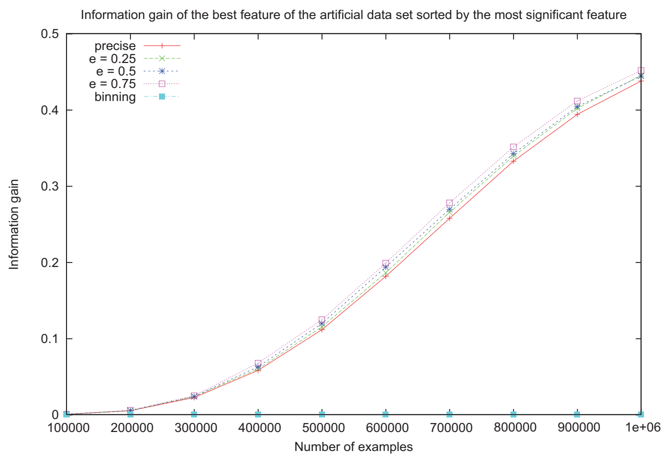

Figure 1. Information gain trend for the best feature in the Artificial dataset.

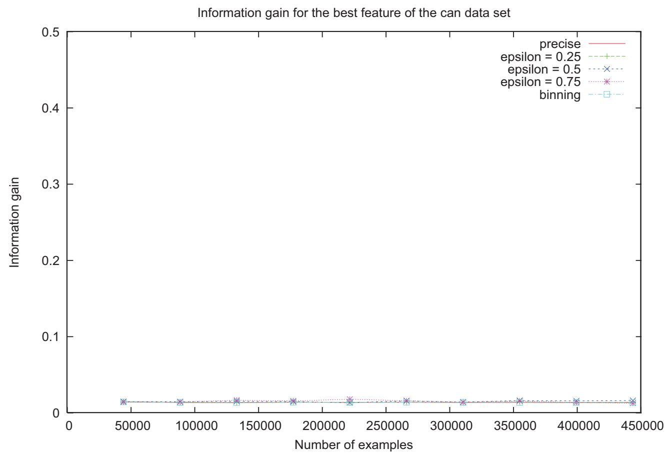

Figure 3. Information gain trend for the best feature in the Can dataset.

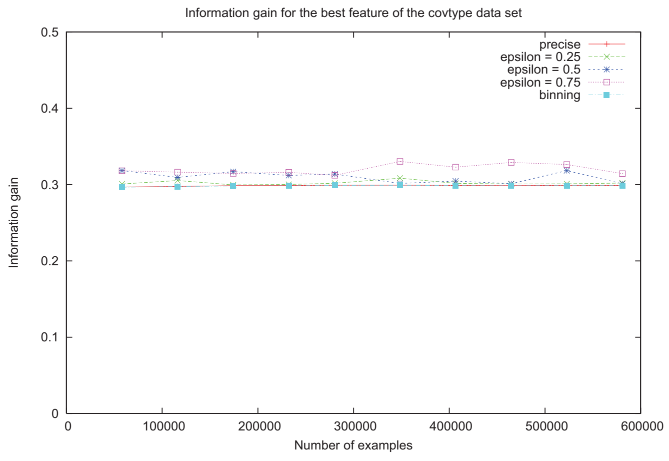

Figure 2. Information gain trend for the best feature in the Covtype dataset.

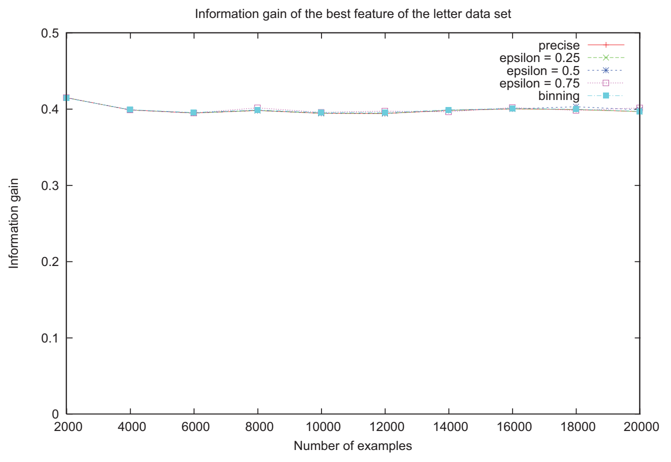

Figure 4. Information gain trend for the best feature in the Letter dataset.

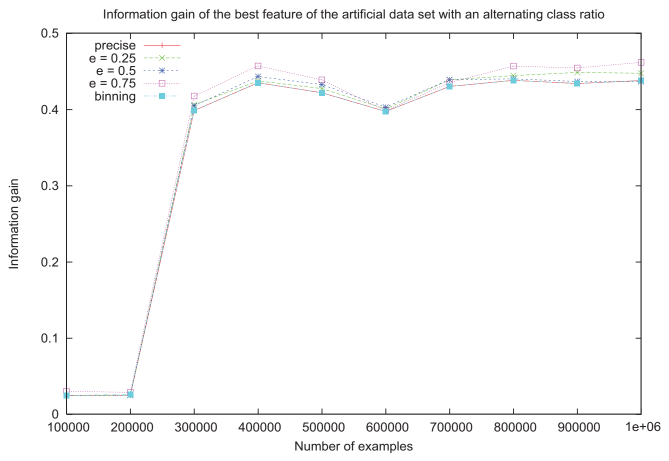

Figure 5. Information gain values for the best feature on the artificial dataset, where the class distribution changes over the streaming data.

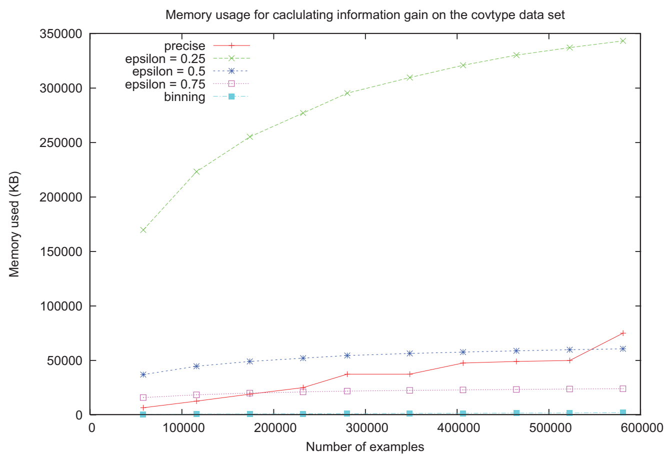

Figure 7. Memory utilized for information gain computation on the Covtype dataset.

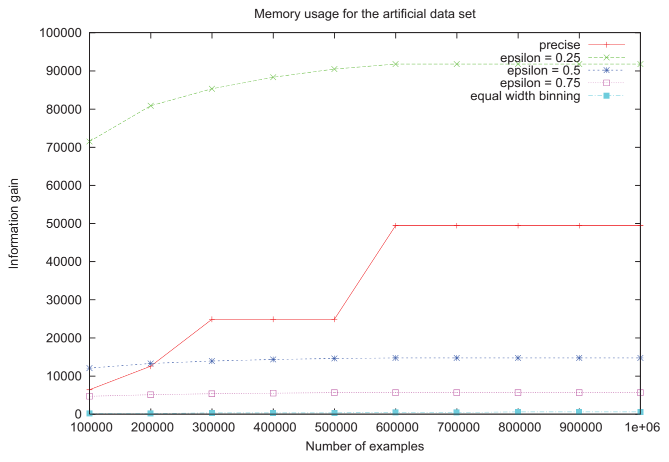

Figure 6. Memory utilized for information gain computation on the Artificial dataset.

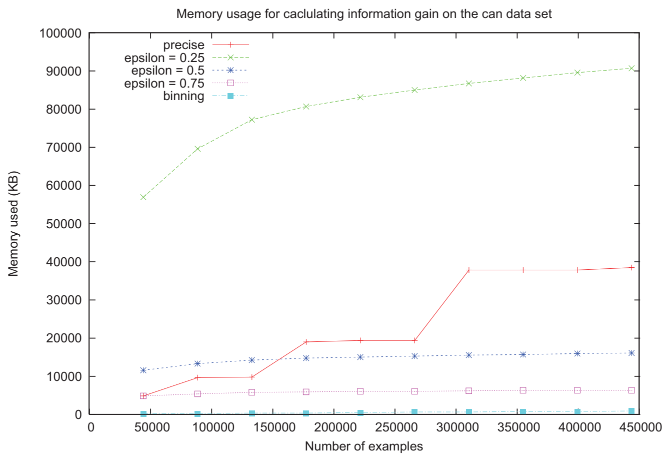

Figure 8. Memory utilized for information gain computation on the Can dataset.

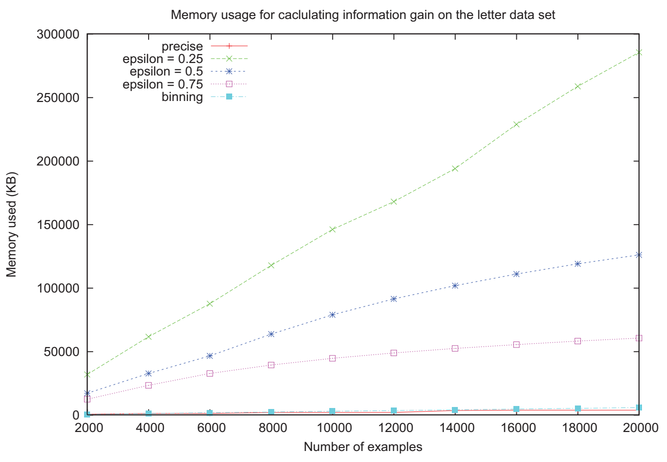

Figure 9. Memory utilized for information gain computation on the Letter dataset.

4 Conclusions

We have designed a randomized solution to computing information gain in general data streams, without making any assumptions on the underlying distributions or domains. The approach uses an original discretization technique based on quantiles. The essential idea is to use finer bins where the relative error due to discretization can be large. We tie this technique to a known method for computing approximate quantiles in data streams and obtain a time and memory efficient solution that has strict error bounds.

We have demonstrated the accuracy, memory efficiency, and robustness of the solution using a variety of datasets. We show that its memory usage is much lower than the corresponding usage by a regular precise computation, and its accuracy much better than an approach using equal interval width binning (which, in fact, completely breaks down if the distribution in the data stream abruptly changes). Based on our theoretical and empirical analysis it is clear that our algorithm's memory efficiency, relative to a regular computation, will be even more dramatic for larger datasets (streams that consist of tens of millions to billions of examples). The algorithm's robustness has been amply demonstrated in the tests at simulate sudden changes in distributions. We thus conclude that it is a practical solution to computing information gain in general data streams.

As a part of our ongoing research, we are testing larger datasets with a variety of dataset distributions, including ones in which the classes can appear and disappear arbitrarily in the stream. Later, we would like to extend the algorithm to build a decision tree in data streams. We hope to demonstrate the practicality of the solution for any statistical or machine learning method that requires discretization of continuous features, be they spatial, temporal, or otherwise.

Acknowledgements

We are grateful to Philip Kegelmeyer and Sandia National Labs for providing the Can visualization dataset.

References

[1] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of 21st ACM Sym. on Principles of Database Systems (PODS 2002)*, 2002.

[2] J. Catlett. On changing continuous attributes into ordered discrete attributes. In *Proceedings of the European Working Session on Learning, Springer-Verlag*, pages 164–178, 1991.

[3] N. V. Chawla and L.O. Hall. Modifying MUSTAFA to capture salient data. Technical Report ISL-99-01, University of South Florida, Computer Science and Eng. Dept., 1999.

[4] P. Domingos and G. Hulten. Mining high-speed data streams. In *Knowledge Discovery and Data Mining*, pages 71–80, 2000.

[5] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202, 1995.

[6] I. Fayyad and R. Kohavi. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1993.

[7] Johannes Gehrke, Venkatesh Ganti, Raghu Ramakrishnan, and Wei-Yin Loh. BOAT — optimistic decision tree construction. In *ACM SIGMOD Conference International Conference on Management of Data*, pages 169–180, 1999.

[8] Anupam Gupta and Francis X. Zane. Counting inversions in lists. In *SODA '03: Proceedings on the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 253–254. Society for Industrial and Applied Mathematics, 2003.

[9] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *KDD*, pages 97–106, 2001.

[10] C.J. Merz and P.M. Murphy. Uci repository of machine learning databases. Univ. of CA., Dept. of CIS, Irvine, CA., Machine readable data repository, http://www.ics.uci.edu/ mlearn/MLRepository.html.

[11] I. Munro and M. Paterson. Selection and sorting with limited storage. In *Proc. IEEE FOCS*, 1980.

[12] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1992.

[13] W. N. Street and Y. Kim. A streaming ensemble algorithm (SEA) for large -scale classification. In F. Provost and R. Srikant, editors, *KDD'01*, pages 377–382, 2001. San Francisco, CA.

[14] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *ACM SIGKDD Conference*, 2003.