Adaptive Histogram Algorithms for Approximating Frequency Queries in Dynamic Data Streams

Joseph S. Gomes
Department of Computer Science, Bowie State University, Bowie, MD, USA

Abstract – *Histograms can be used as summaries of frequency data. However, staying within the error tolerance becomes problematic when dealing with dynamic data streams. For dynamic data streams, the histograms can be reconstructed every time data is either discarded or collected - which is very inefficient. If a histogram is to be employed as a quick estimate of stream data, updating the histogram non-destructively can be done using the following approach: decrement one from each bucket where data is to leave the histogram, and increment one to each bucket where data is to enter the histogram. In this paper, we empirically prove this method to be a generally strong way to control loss of accuracy. The costs of executing this error-minimizing layer are trivial to processing, memory, and should consequentially maximize uptime. This method was tested on two histogram algorithms including Equivalent Width and Variance Optimal in four specified histogram data-density scenarios including sparse, balanced, dense, and very dense, while using two different random value distribution sources including the Uniform distribution and Gaussian distribution.*

Keywords: Histogram, Frequency queries, Data Stream, Approximation, Algorithm

1 Introduction

Histograms, utilized as a summary of frequency data, have been proven to be accurate-enough measures to approximate count (or frequency) queries. Staying within the error tolerance becomes problematic when dealing with dynamic data, such as streams, due to the potential for shifts in source data. This shifting can happen even if a stream is modeled by using a single random distribution, especially when observing a relatively smaller number of values as related to a large or infinite data set. Elements can also expire and become irrelevant, as well as new elements can come into existence. Attempting to run a histogram on dynamic data without a method of controlling error will become disastrous, especially on specialized histograms such as the Variance-Optimal and Maximum Difference due to the way they interpret data inherent to their original construction. In lieu of reconstructing a histogram every time data is either discarded or collected - which is prohibitive in processing power, memory space, and real life uptime - a method for reducing cumulative error is necessary if not imperative. Therefore, if a histogram is to be employed as a quick estimate of stream data, updating the histogram non-destructively can be done using the following approach: decrement one from each bucket where data is to leave the histogram, and increment one to each bucket where data is to enter the histogram. The costs of executing this error-minimizing layer are trivial to processing, memory, and should maximize uptime. In this paper, we have tested this method on two histogram algorithms including Equi-Width and Variance Optimal (also known as V-Opt or V-Optimal).

2 Background

Here we will discuss some of the basics of queries and histograms. Various types of frequency based queries are discussed that can benefit from histograms.

2.1 Queries

There are three types of queries that can benefit from histograms.

2.1.1 Selection queries with equality constraints

Selection queries with equality constraints return the tuples in a table that satisfy a certain equality criteria. You can also associate a count function to these selection queries. For example, in order to determine the number of employees with age 65, the following SQL-like query could be executed:

select count(*) from employee where age=65

This query can be further extended to include other values in the following manners:

select count(*) from Employee where age=62 or age=65

select count(*) from Employee where age in (62,65,67)

If a certain amount of error is acceptable, a histogram can be used to estimate the frequency of a certain value in a data source. The individual frequencies can then be added to get the combined counts of extended queries above.

2.1.2 Range queries

The range query can have either a lower or an upper limit on its input, or both limits explicitly declared. For example, suppose an executive in a company wants to know how many of their employees are making salaries between the range of 50,000 and 100,000 inclusively. The SQL-query would be structured as:

select count (employee_id) from Salary where salary >= 50000 and salary <= 100000

Here also histograms can be used to compute the count for the whole range by adding frequencies for each individual value in the range or by manipulating frequencies of whole buckets.

2.1.3 Join Queries

Join queries are more targeted towards pattern matching. Just as in selection queries, a count function can be attached to join queries as well. Consider the scenario where personal info for employees is stored in table Employee and their salary information is stored in table Salary.

Select emp.first_name, emp.last_name, emp.age, s.salary from Employee emp join Salary s on (emp.empId =s.empId) where s.salary>=100000 and emp.age <=35

If accuracy can be sacrificed for efficiency, histograms can be used to provide estimates on each data source, which in turn can be used to generate join query result-counts. Histogram information on the individual tables can also be used to create an optimized query plan that would run faster.

2.1.4 Histogram Algorithms

Histograms are compressed versions of an entire data set that are used as statistical tools of approximation. There are many ways to create a histogram that mainly differ in the method of setting boundaries commonly known as buckets (Ioannidis Y. 2003). Each individual bucket can represent a range of values where the range can be as small as a distinct value, and as large as the whole data range. How to break up a data set into subsets to store in a single bucket is up to the histogram algorithm. The easiest way to visualize a histogram is to picture a bar graph with the bucket ranges on the horizontal axis and the frequency counts on the vertical axis. By design, histograms are excellent tools for single count queries and range queries. In the following subsections, we will briefly describe two of the most popular algorithms that will be used in this research. Both the examples will be using the same sample array: {0, 2, 2, 3, 3, 4, 5, 8, 8}

2.1.5 Equivalent Width

One of the more basic histogram algorithms, the equal-width or equiWidth method separates data into horizontally equal-sized buckets. This means that each bucket will represent, as closely as possible, an equal number of values. The benefits of this algorithm lie in the simplicity – and consequentially the inexpensive computing cost to construct it. The time complexity is O(n) due to the single pass required to process through the pre-sorted source array to build the histogram.

2.1.6 Variance-Optimal (V-Opt)

The variance-optimal (also commonly referred to as V-Opt and V-Optimal) histogram is one of the more complicated histogram algorithms. It is also broadly regarded as one of the more accurate histogram algorithms. The premise behind this algorithm is to minimize the sum of all intra-bucket variances. There exists a dynamic programming algorithm with O(n^2B) time complexity that follows the following recurrence relation

$$SSE^*(i,k) = \min_{1 \le j < i} \{SSE^*(j, k-1) + SSE([j+1, i])\} \quad (1)$$

In the above equation, n is the number of elements in the data array, B is the number of buckets to construct the histogram and $SSE^*(i, k)$ is the minimum sum of squared error *(SSE)* for the prefix vector $F[1, i]$, i.e. the first i values of the frequency array corresponding to the data array using at most k buckets. Notice, for this algorithm we first need to convert the data array to a frequency array F, where $F[i]$ is the frequency of the value mapped to slot i. As you can see that this histogram construction algorithm can become cost-prohibitive as n grows bigger.

3 Approximating count queries using Histograms

Suppose we would like to know the frequency of value v. In order to do this, we would first find out which bucket i value v belongs to and then find the frequency f_i and number of values n_i for that bucket. Then the approximate frequency for v can estimated to be f_i / n_i. For our sample array, according to the equiWidth histogram the frequency of 3 is 4/3 = 1.33 and according to v-optimal histogram the answer is 6/6 = 1. As you can see, for v = 3, v-opt has a higher error (2 – 1 = 1) than equiWidth (2 – 1.33 = 0.67). However, for v = 2, error in v-opt estimation is 2 – 2 = 0, whereas equiWidth has an error of 2 – 1 = 1. On average, v-opt produces lower error if all values in the data range are equally likely to be queried as it is designed to minimize average intra-bucket variance (SSE) which ultimately minimizes average error.

3.1 Histograms for Dynamic Data Streams

The goal of histograms is to be as accurate in estimating data distribution as possible while improve speed of

answering queries and also the speed of histogram construction. The speed of query execution results from the faster access of a histogram array versus searching and counting an entire data array. Since error is produced by the averaging of the frequency among all the values located in the same bucket, it can be reduced by increasing the number of buckets in the histogram. At one end of the spectrum is the one-bucket histogram that will have the highest error and fastest construction. On the other end is the *n*-bucket histogram where *n* is the number of distinct values, which will have the slowest construction time and no error. As one can see, the two goals reducing error and increasing construction speed are at odds with each other. One may think one should just use *n* buckets since accuracy is more important than offline histogram construction time. However, using an *n*-bucket histogram not only increases construction time but it also requires more space to store the histogram. As a result choosing the right number of buckets and the right algorithm is crucial for system performance.

With that established, the problem becomes even more difficult when running frequency queries on continuous data streams comes into picture. Since a histogram could become increasingly inaccurate with each new data point entering the stream, we investigate the effectiveness of a simple but fast method in mitigating the potentially distortional effects a stream could have on histograms.

3.2 Dynamic Data Streams

In order to emulate a dynamic data source, a sliding window based method is used. This sliding window approach used in experiments is very similar in nature to a stock or financial ticker on a news channel. The television can only show as much data that can fit on the screen, just like a fixed-size sliding window stores data. As the ticker scrolls, stock data exits the screen and new and more relevant information comes into the picture. The point of this parallel is to illustrate the core workings of the sliding window mechanism, as applied to processing data stream simulations. In this paper we consider only fixed-size sliding windows as opposed to variable-sized sliding windows.

3.3 Reducing Inaccuracy

In the scope of this paper, inaccuracy, *I*, is measured using a normalized error technique given by equation 2.

$$I = \frac{\left|V_A - V_H\right|}{V_A} \qquad (2)$$

Here V_H is the answer returned by the histogram and V_A is the the actual frequency in the sliding window.

By establishing a metric for representing inaccuracy, a measured comparative difference can be calculated from various testing simulations. Static histograms have error built in to begin with. Therefore, if a distribution changes skew even slightly, the histogram would become increasingly less accurate, depending on how significant a change. Even in the same distribution, depending on how many values the histogram stores and covers in the overall range, shifts could be expected – still leading to an increase in inaccuracy. As a result, to be able to answer queries with higher accuracy, histograms need to be updated in accordance with change in streaming data. This is not a big problem for equiWidth since the bucket boundaries stay the same and only frequencies change. However, this is very problematic for v-optimal histograms since the new data could render the existing histogram boundaries incorrect as far as minimizing expected intra-bucket variance is concerned.

In an attempt to control for such a force, this paper explores a bucket tweaking method. To combat the complications that dynamic streaming data presents to histogram algorithms, some sort of mitigation method must be installed to keep the histogram up to date so to speak. The histogram algorithms are capable of handling static data, but cannot compensate themselves for data growing stale and driving inaccuracy up, as occurs with dynamic data. Since this shifting effect appears to affect the specialized partition scheme algorithms the most, it would be prudent to attack the problem at its core: the buckets. Instead of moving the buckets around in a desperate attempt to account for an entire distribution shift, consider smaller, incremental changes designed to be an agile response to dynamic activity in the simulated data stream. By using the sliding window approach previously discussed and already proven to be an effective way to manage data streams, it is possible to determine what data is becoming stale, and what data is fresh. In combination of the agile concept proposed and the sliding window approach, the following adaptive method is used to keep the histogram as up to date as reasonably possible. For every data point that leaves the sliding window, decrement the corresponding bucket in the histogram by one. Conversely, for every data point entering the sliding window, increment the corresponding bucket by one. In these experiments, a single unit of data expires as a single unit of data becomes relevant. That is an inherent property of a fixed-size sliding window, however, it would be possible to change the size of the stream sample if the situation deems necessary.

4 Experimental setup

Two different pseudo-random distributions are used in this paper to avoid single distribution bias – Uniform and Gaussian. Similar to the problem of distribution bias is the issue of histogram density. To be precise, density in the

scope of this research refers to the ratio between the number of values in the histogram as it relates to the range the histogram attempts to model. For example, a histogram summarizing 100 distinct values with 150 total occurrences has density 1.5. To address the density bias, four different histogram densities given below were tested.

Name	Density
Sparse	1
Balanced	1.5
Dense	2
Very Dense	2.5

4.1 Performance metric

For each combination of input characteristics, we use 0 – 1000 as the value range. Once the data arrays are filled, queries are posed after each new data addition. Altogether 10000 queries are executed and their error measured. At the end of each trial, the normalized errors obtained from equation 2 are summed, and divided by the number of queries processed to determine an average normalized error per query.

5 Results

This section discusses all the simulation results and presents several charts. The x-axis of each chart shows the number of buckets as a percentage of the size of the zero-based range, and the y-axis shows the measure of inaccuracy determined by the normalized difference per query. Each chart title and legend clearly dictates what simulation data is presented. These charts are separated based on pseudorandom distribution and histogram algorithm. Each chart shows the performance of the histogram algorithm as it applies to all four density-ratios and three types of source data and histogram pairings – static data with static histograms (*static* scenario), dynamic data with static histograms (*dynamic* scenario) and dynamic data with adaptive histograms (*adaptive* scenario). The density-ratios are color-coordinated. The source data and histogram pairings are coordinated on the graph by line-type, such that: static scenario simulations are represented by a solid line, dynamic scenario simulations are shown with a dashed line, and the *adaptive* scenario simulations are shown as a dotted line. The static scenario represents the baseline as the results cannot get any better than this. The purpose of testing the dynamic scenario is two-fold. One, it determines the feasibility of using a static histogram in a data stream management system. If the skew of a distribution in data stream suddenly morphs, how does that affect the histogram? Two, as alluded to with the prior question, if and how does the accuracy of the histogram change as the data stream progresses with new information processed by the sliding window? Finally, the adaptive scenario is used to test the effectiveness of our adaptive histogram method in estimating answers to frequency based queries.

5.1 EquiWidth on Uniform Data

This distribution is usually better suited for analysis by histograms because the skew is very low and consistently even, regardless of the point in question within the distribution chart. The results shown in Figure 3 are as expected, from the static showing less inaccuracy with more buckets, to the dynamic data losing accuracy with more buckets, through to the adaptive method following the same pattern as the static data. The static data here is very consistent in accuracy from the sparse density-ratio up through the very dense density-ratio. As previously mentioned, the dynamic data does increase in inaccuracy, which again proves the point that histograms alone are ill suited to model streaming data. Combined with the fact that they become less accurate with more buckets, this means that there must be a different solution. In this case, the solution again is the adaptive method. With the EquiWidth histogram, queries were nearly just as accurate as the static data in the lower number of buckets scenarios, and proceeded to follow the static data very closely in accuracy. This strongly suggests that the adaptive method works well in these categories.

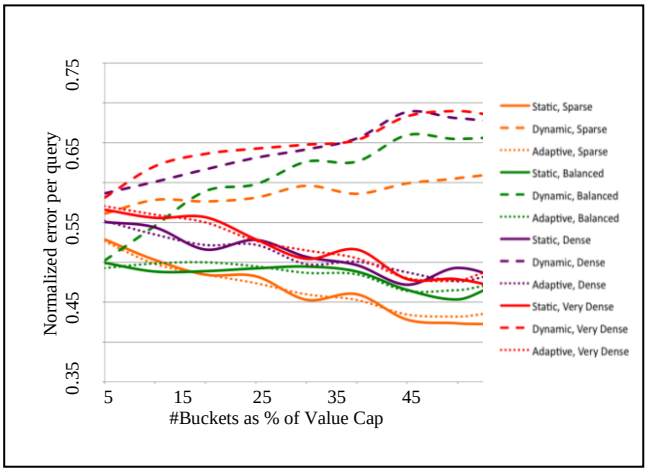

Figure 1: Equiwidth error on Uniform data

5.2 V-Opt on Uniform Data

Proceeding onward from the equivalent width analysis, we now study the results of the variance optimal algorithm processing uniform pseudorandom data. The chart is shown in Figure 4. Again, this is excellent result. The graph shows the decreasing slope for static data, showing that the more buckets the variance optimal

histogram has at its disposal, the more accurate the results it can return.

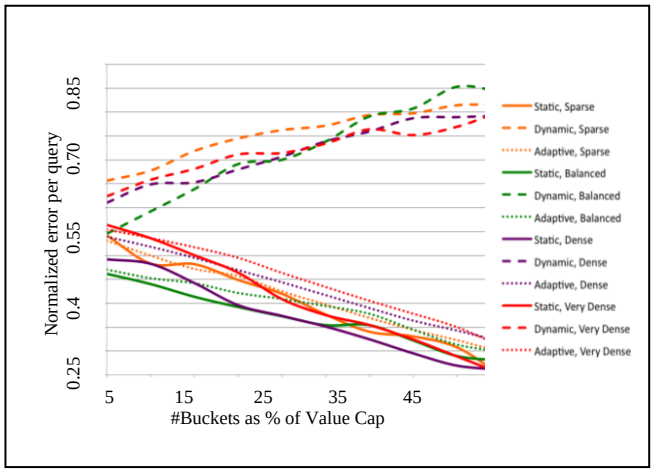

Figure 2: V-Optimal error on Uniform data

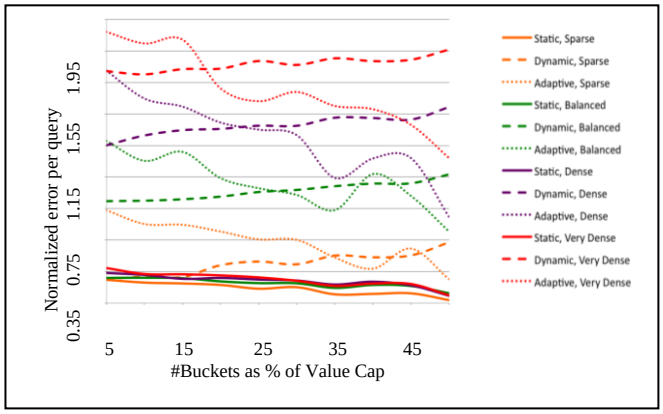

Figure 5: EquiWidth error on Gaussian data

Variance optimal histogram test data shows consistency in accuracy when the density-ratio is at least 1:1. The dynamic data lines tell a story. As more buckets are introduced in a dynamic setting, the histogram gets less accurate rapidly and consistently. This shows that the variance optimal algorithm is sensitive and susceptible to changes in its underlying data. Due to the nature of the algorithm basing its partitioning scheme on minimizing variance, it is of no surprise that it loses accuracy as rapidly as it does in these stressful scenarios. Conversely, the adaptive method again shows excellent mitigation of error creep introduced by dynamic data. The slope nearly matches the static data slope. Also, the level of accuracy provided by the adaptive method nearly matches the level of accuracy that the static data typically returns. At peak effectiveness, at the 50% number of buckets mark, the difference in accuracy is around 0.05 units. By contrast, the dynamic data at the same point was ten times less accurate.

5.3 EquiWidth on Gaussian Data

Gaussian pseudorandom data provides a different challenge, as it is a more erratic distribution with regard to frequency. This poses a complication to histograms, because the skew is both greater than a relatively flat line and changing depending on the point in question within the distribution chart.

The chart displaying the results of this simulation is at the end of this subsection, shown in Figure 5. Proceeding to the analysis, the vast majority of the results are as expected. The static lines are consistent and at a downward slope – confirming that the more buckets, the more accurate. They are also closely clustered, suggesting that density-ratio does not have a great effect on inaccuracy. Regardless of how much data is in the histogram, the appearance is that the query will be approximately a half point away in accuracy, or fifty percent of the actual frequency. The dynamic data lines are ascending in slope, which is also expected. Bear in mind that over the same sized range, the larger the number of buckets, the smaller range size they individually cover normally. This smaller range size on one hand does make them better representations of the distribution curve, but also makes them more sensitive to changes in underlying data. A larger bucket may not be more accurate, but it will not lose accuracy as quickly either. Additionally, the dynamic data scenario is affected by the density ratio. The more data packed into the histogram, the less accurate. A balanced histogram is already off by one full point in estimation in the five percent bucket situation, and nearly reaches two full points away from the true value in the very dense scenario with fifty percent. Bearing in mind that each point represents a multiplicative factor applied to the actual frequency value, inaccuracy rising to an increased error of 200% is problematic. The adaptive lines show a lot of promise here. As they have more buckets to work with, they return greater accuracy. This is exactly the opposite of what occurs with dynamic data, and precisely what the goal of this research sought out to determine. Data density does appear to have an effect as well, such that the denser the histogram, the sooner the adaptive benefits take place, with respect to an ascending number of buckets. All of these effects combine to solve many of the problems dynamic data present, and that is with the more difficult pseudorandom distribution to model in histograms. At peak operating efficiency, the data charts suggest that it is possible to re-gain over a half point of accuracy in the dense and very dense scenarios. This does not adversely

affect the attempt to devise a generally acceptable method of mitigating dynamic error creep, due to the fact that generalizations are not universally perfect. Although this does make the adaptive method appear to be more of a situational fix at first glance, given enough situations where the adaptive method presents benefits, this can be utilized as a generalized solution to counteract the negative effects of dynamic data on histograms.

5.4 V-Opt on Gaussian Data

Moving forward, the next algorithm for analysis is the variance optimal algorithm using Gaussian data. The chart displaying the results of this simulation is shown in Figure 6. As previously exposed with the equivalent-width algorithm, the expected results of the static data increasing in accuracy as more buckets are provided is expected. Also expected, is the overall decrease of accuracy when running on dynamic data, and the increase in inaccuracy with more buckets on dynamic data. The dynamic data can result in as much as two points of divergence from the true frequency count, as shown in the fifty percent case with a density-ratio of very dense.

The adaptive technology, on the other hand, is really starting to shine. As seen with the equivalent width algorithm, the adaptive method works better with denser histograms. In two categories, balanced and dense data density ratios, the adaptive method was completely more accurate than the dynamic scenario. The very dense scenario was more accurate starting with the ten percent bucket point, forward to all counts of buckets higher. The overall downward sloping dotted line in all scenarios is very encouraging for the adaptive method to mitigate the inaccuracy complications that dynamic data can present.

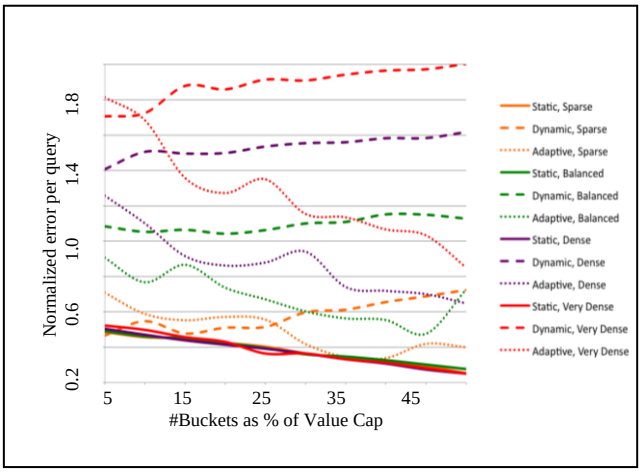

Figure 6: V-Optimal error on Gaussian data

6 Future Applications

The future applications for research exist as an extension of what was done in this paper. Iteratively speaking, more algorithms could be tested with more random distributions – which could yield either more evidence confirming these results, or information pointing to the contrary. Similarly, another way to extend the research presented in this paper could be to develop other methods of keeping histograms fresh. Perhaps it is possible to dynamically alter the bucket boundaries to reflect a best-guess real-time estimation for the summary of a dynamic data stream. Following the same logic, it could also be possible to trigger a full histogram reconstruction should the level of inaccuracy exceed a specified error threshold. These would certainly benefit the field.

Further ahead, it is possible that one could start testing these concepts against higher-dimensional histograms, determining if the results found here extend to more complicated algorithms. Common sense is not factual until empirically proven in the eyes of the scientific community, and as such, results could be proven contrary to common sense. This research was also limited to the scope of single selection count queries based on frequency. It would not take much to extend this research to range queries and join queries.

7 Related Work

Amongst many researchers in the field studying histograms, one in particular took the time to compose a comprehensive study of the history of histograms: Yannis Ioannidis of the University of Athens. In his paper entitled, *"The History of Histograms (abridged),"* Again jumping in time, now to the mid-1990s, Yannis Ioannidis writes another paper, now with Viswanath Poosala at the University of Wisconsin, *"Balancing Histogram Optimality and Practicality for Query Result Size Estimation"* (Ioannidis & Poosala, 1995). Jagadish et. al. developed a dynamic programming method of calculating the bucket boundaries in a variance optimal histogram in the paper "*Optimal Histograms With Quality Guarantees."* By using an inherent property of the variance optimal histograms, the sum-squared error (SSE) is determined to be possible to calculate in an iterative fashion. This ability, in turn, saves repeated and redundantly wasteful calculations when intermediate results are stored in an internal matrix. Later Yannis Ioannidis and Viswanath Poosala delve into answering queries with an approximation while improving database response time (Ioannidis Y. E., 1999), in *"Histogram-Based Approximations of Set-Valued Query Answers"*. At that time, database responses were slow to return information – even when precision was not necessary. By using histograms, they were able to generate approximations for early reports in on-line database queries when the precise answer was not absolutely critical, but response time was. With regard to utilizing sliding window technology, Arvind Arasu and Gurmeet Singh Manku pioneered the efforts of maintaining

approximations and quantiles over both fixed and variable-sized sliding windows processing streaming data (Arasu & Manku, 2004). Gurmeet Singh Manku and Rajeev Motwani devised two methods of approximating frequently occurring singleton frequency counts over data streams using sliding windows with small memory footprints and output error limited to user-specification (Manku & Motwani, 2002). In other related work, sliding window technology has been used in conjunction with other algorithms designed to count and maintain aggregate data in dynamic data streams with provable low-memory (Datar, Gionis, Indyk, & Motwani, 2002).

8 Conclusions

While histograms inherently are not designed, nor are capable of by default, to handle dynamic data sources, they can be adjusted with certain methods of error-minimization to compensate for what otherwise could be considered cumulative error. This paper has empirically shown that when using a sliding window approach for using the histogram over a dynamic data stream, the histograms grow stale and less accurate. The data analyzing partition scheme based histogram algorithms suffer the most, which is problematic because they are also the most accurate under static data. In order to make them work on dynamic data, their buckets must be modified in such a manner to keep them fresh.

Overall the adaptive method is a strong success. In the Equivalent Width histogram tests using Gaussian pseudorandom data, the adaptive histogram process was an okay performer. It did increase accuracy over the dynamic data on static histograms scenarios with more buckets or more dense data. In overlapping cases, performance was even better. Specifically, there was a 20% reduction of error at the 25% number of buckets ratio when the histogram was very densely packed. The Variance Optimal histogram on Gaussian data benefitted strongly from the adaptive method. In the balanced, dense, and very dense ratios, inaccuracy decreased between 40% and 70% from the dynamic data estimates across the spectrum of number of buckets, and above 15% number of buckets ratio for very dense data. For EquiWidth histograms operating on uniformly distributed data, the adaptive method proved to be an excellent performer. The accuracy rivaled static data results, within 2% throughout all density ratios and number of bucket ratios. Additionally, VOpt histograms with uniformly distributed data maintained a level of accuracy within 10% of all of the static data tests, throughout all number of bucket ratios and data density ratios. In summary, the adaptive method showed okay to excellent results, depending on pseudorandom distribution. Although not as strong on Gaussian data, the adaptation drastically improved accuracy when histograms were processing uniformly distributed data. There was an overall trend of a greater decrease in inaccuracy with more buckets.

This paper has shown that an agile method using minimal incremental changes can aid in maintaining the accuracy of the histograms. In conclusion, this method is a strong candidate for error mitigation in histograms applied to dynamic data sources for a few reasons. It is inexpensive to maintain, as two high level seek operations and two basic operations at most are required per update: seeking the corresponding bucket in a histogram for a given value (twice if inserting and removing), and incrementing or decrementing that value as necessary. Those operations are not processor or memory intensive tasks. Finally, the decrease in inaccuracy over a broad range of tests suggests the potential for true generalization.

References

[1] Ioannidis, Y. (2003). The history of histograms (abridged). VLDB '2003: Proceedings of the 29th international conference on Very large data bases , 19-30.

[2] Ioannidis, Y. & Poosala V. (1995). Balancing Histogram Optimality and Practicality for Query Result Size Estimation. SIGMOD '95: Proceedings of the 1995 ACM SIGMOD International conference on Management of data, 233 – 244.

[3] Jagadish, H. V., Koudas, N., Muthukrishnan, S., Poosala, V., Sevcik, K. C., & Suel, T. (1998). Optimal Histograms with Quality Guarantees. Proceedings of the 24rd International Conference on Very Large Data Bases , 275-286.

[4] Ioannidis, Y. E. (1999, September). Histogram-Based Approximation of Set- Valued Query-Answers. VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases , 174-185.

[5] Arasu, A., & Manku, G. S. (2004). Approximate Counts and Quantiles over Sliding Windows. PODS '04: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems , 286-296.

[6] Manku, G. S., & Motwani, R. (2002). Approximate frequency counts over data streams. VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases , 346-357.

[7] Datar, M., Gionis, A., Indyk, P., & Motwani, R. (2002). Maintaining stream statistics over sliding windows: (extended abstract). SODA '02: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms , 635 – 644.