A benchmark for histogram building algorithms of numeric streams

Stefan Sebastian, 242

Contents

1 Introduction

Main goals Motivation

1.1 Histograms

Short description of histograms use + why streaming

//todo reformulate, benahim2.1 A histogram is a set of B pairs (called bins) of real numbers (p1,m1),...,(pB,mB), where B is a preset constant integer. The histogram is a compressed and approximate representation of a set S of real numbers

Paper structure

2 Algorithms

3 Numeric Histogram

3.1 Overview

The Numeric Histogram algorithm first appeared in the paper by Ben-Haim and Tom-Tov on building a streaming parallel decision tree algorithm[1]. The main focus was to describe a method for tree based classifiers for large data sets in a distributed environment, however they needed a data structure that can summarize large amounts of data accurately. Thus, they proposed a histogram data structure that can adapt to the requirements of a streaming environment.

The histogram maintains a fixed number of bins of the shape (p, m) where p represents a central value of the interval and m the number of points in it. Initially the histogram has some bins allocated but the values are not known. They are filled as data comes is added into the structure and once the allocated number of bins is reached the central bin values are updated depending on incoming values. In this regard, the algorithm is robust to changes in data patterns over a longer period of time, which is one of the main concerns of streaming analysis algorithms.

3.2 Procedures

The proposed data structure contains four procedures: update, merge, sum and uniform. But for the purpose of this benchmark only two were needed: update and sum. The update procedure adds a new point to the histogram data structure and is described in algorithm 1.

The sum procedure, depicted in algorithm 2, obtains the estimated number of points between $(-\infty, n]$ where n is the input value. In order to estimate the number of values in a range $[a, b]$ we can calculate $sum(b) - sum(a)$. The algorithm assumes that for each bin (p, m), there are $m/2$ points to the left of p and $m/2$ to the right. This means that the number of points in the interval $[p_i, i_{i+1}]$ is equal to $(m_i + m_{i+1})/2$, which is the area of the trapezoid $(p_i, 0), (p_i, m_i), (p_{i+1}, m_{i+1}), (p_{i+1}, 0)$, divided by $p_{i+1} - p_i$. Similarly, we can estimate the number of points in the interval $[p_i, p]$ by adding calculating its projection on the line from (p_i, m_i) to (p_{i+1}, m_{i+1}), then find the area of the new trapezoid and dividing again by $p_{i+1} - p_i$. The algorithm will not work properly if the point p is smaller than

Data: histogram $h = (p_1, m_1), .., (p_b, m_b)$, a point p
Result: a histogram that represents the set $S \bigcup \{p\}$
binary search for the closest p_i larger than p;
if $p_i = p$ **then**
  $m_i = m_i + 1$;
else
  add a new bin of shape (p, 1) to the histogram at the i-th position;
  find the closest two bins by their p values;
  merge those bins, moving p proportional to their m values;
end

Algorithm 1: add procedure

p_0 or larger than p_b. For this reason the data structure should be initialized with a lower and upper bound into which all incoming data should fit.

Data: a histogram h, a point p such that $p_1 < p < p_b$
Result: estimated number of points in the interval $[-\infty, p]$
binary search to find i such that $p_i <= p <= p_{i+1}$;
set $s = (m_i + m_p) \cdot (b - p_i)/2 \cdot (p_{i+1} - p_i)$;
where $m_p = m_i + (m_{i+1} - m_i) \cdot (p - p_i)/(p_{i+1} - p_i)$;
for $j < i$ **do**
  $s = s + m_j$;
end
$s = s + m_i/2$;

Algorithm 2: sum procedure

3.3 In practice

This algorithm was adapted and implemented into the open source project Apache Hive[2], which is a datawarehouse solution built on top of Hadoop to provide data query and analysis methods. The project is used actively in industry, handling reporting tasks for large volumes of data, for e.g. data produced by 435M monthly users of Chitika[3]. The version of the algorithm presented in this report is based on the Hive implementation, called NumericHistogram[4]. The optimal number of bins is left as a choice to the user, however the suggested range is between 20 and 80.

3.4 Optimal Streaming Histograms

TODO describe the algorithm from https://amplitude.com/blog/2014/08/06/optimal-streaming-histograms

4 Benchmark

TODO describe benchmark methodology tests results

References

[1] Y. Ben-Haim and E. Tom-Tov, "A streaming parallel decision tree algorithm," *J. Mach. Learn. Res.*, vol. 11, pp. 849–872, Mar. 2010.

[2] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, "Hive: A warehousing solution over a map-reduce framework," *Proc. VLDB Endow.*, vol. 2, pp. 1626–1629, Aug. 2009.

[3] "Poweredby." `https://cwiki.apache.org/confluence/display/Hive/PoweredBy`. Accessed: 2019-05-01.

[4] "Numerichistogram." `https://github.com/apache/hive/blob/f37c5de6c32b9395d1b34fa3c02ed06d1bfbf6eb/ql/src/java/org/apache/hadoop/hive/ql/udf/generic/NumericHistogram.java`. Accessed: 2019-05-01.