Title: AMS Sketch
Name: Graham Cormode[1]
Affil./Addr. Department of Computer Science, University of Warwick, Coventry, UK
Keywords: streaming algorithms; second-moment estimation; Euclidean norm, sketch
SumOriWork: 1996; Alon, Matias, Szegedy

AMS Sketch

GRAHAM CORMODE[1]
Department of Computer Science, University of Warwick, Coventry, UK

Years aud Authors of Summarized Original Work

1996; Alon, Matias, Szegedy

Keywords

streaming algorithms; second-moment estimation; Euclidean norm, sketch

Problem Definition

Streaming algorithms aim to summarize a large volume of data into a compact summary, by maintaining a data structure that can be incrementally modified as updates are observed. They allow the approximation of particular quantities. The AMS Sketch is focused on approximating the sum of squared entries of a vector defined by a stream of updates. This quantity is naturally related to the Euclidean norm of the vector, and so has many applications in high-dimensional geometry, and in data mining and machine learning settings that use vector representations of data.

The data structure maintains a linear projection of the stream (modeled as a vector) with a number of randomly chosen vectors. These random vectors are defined implicitly by simple hash functions, and so do not have to be stored explicitly. Varying the size of the sketch changes the accuracy guarantees on the resulting estimation. The fact that the summary is a linear projection means that it can be updated flexibly, and sketches can be combined by addition or subtraction, yielding sketches corresponding to the addition and subtraction of the underlying vectors.

Key Results

The AMS sketch was first proposed by Alon, Matias and Szegedy in 1996 [1]. Several refinements or variants have subsequently appeared in the literature, for example in the work of Thorup and Zhang [4]. The version presented here works by using hashing to map each update to one of t counters, rather that taking the average of t repetitions of an “atomic” sketch, as was originally proposed. This hashing-based variation is often referred to as the “fast AMS” summary.

Data Structure Description

The AMS summary maintains an array of counts which are updated with each arriving item. It gives an estimate of the ℓ_2-norm of the vector v that is induced by the sequence of updates. The estimate is formed by computing the norm of each row, and taking the median of all rows. Given parameters ε and δ, the summary uses space $O(1/\varepsilon^2 \log 1/\delta)$, and guarantees with probability at least $1 - \delta$ that its estimate is within relative ε-error of the true ℓ_2-norm, $\|v\|_2$.

Initially, v is taken to be the zero vector. A stream of updates modifies v by specifying an index i to which an update w is applied, setting $v_i \leftarrow v_i + w$. The update weights w can be positive or negative.

The AMS summary is represented as a compact array C of $d \times t$ counters, arranged as d rows of length t. In each row j, a hash function h_j maps the input domain U uniformly to $\{1, 2, \dots t\}$. A second hash function g_j maps elements from U uniformly onto $\{-1, +1\}$. For the analysis to hold, we require that g_j is *fourwise* independent. That is, over the random choice of g_j from the set of all possible hash functions, the probability that any four distinct items from the domain get mapped to $\{-1, +1\}^4$ is uniform: each of the 16 possible outcomes is equally likely. This can be achieved by using polynomial hash functions of the form $g_j(x) = 2((ax^3 + bx^2 + cx + d \mod p) \mod 2) - 1$, with parameters a, b, c, d chosen uniformly from the prime field p.

The sketch is initialized by picking the hash functions to use, and initializing the array of counters to all zeros. For each update operation to index i with weight w (which can be either positive or negative), the item is mapped to an entry in each row based on the hash functions h, and the update applied to the corresponding counter, multiplied by the corresponding value of g. That is, for each $1 \le j \le d$, $h_j(i)$ is computed, and the quantity $wg_j(i)$ is added to entry $C[j, h_j(i)]$ in the sketch array. Processing each update therefore takes time $O(d)$, since each hash function evaluation takes constant time.

The sketch allows an estimate of $\|v\|_2^2$, the squared Euclidean norm of v, to be obtained. This is found by taking the sum of the squares of row of the sketch in turn, and finds the median of these sums. That is, for row j, it computes $\sum_{k=1}^{t} C[j, k]^2$ as an estimate, and takes the median of the d such estimates. The query time is linear in the size of the sketch, $O(td)$, as is the time to initialize a new sketch. Meanwhile, update operations take time $O(d)$.

The analysis of the algorithm follows by considering the produced estimate as a random variable. The random variable can be shown to be correct in expectation: its expectation is the desired quantity, $\|v\|_2^2$. This can be seen by expanding the expression of the estimator. The resulting expression has terms $\sum_i v_i^2$, but also terms of the form $v_i v_j$ for $i \neq j$. However, these "unwanted terms" are multiplied by either +1 or -1 with equal probability, depending on the choice of the hash function g. Therefore, their expectation is zero, leaving only $\|v\|_2$. To show that it is likely to fall close to its expectation, we also analyze the variance of the estimator, and use the Chebyshev inequality to argue that with constant probability, each estimate is close to the desired value. Then taking the median of sufficient repetitions amplifies this constant probability to close to certainty.

This analysis shows that that the estimate is between $(1 - \varepsilon)\|v\|_2^2$ and $(1 + \varepsilon)\|v\|_2^2$. Taking the square root of the estimate gives a result that is between $(1 - \varepsilon)^{1/2}\|v\|_2$ and $(1 + \varepsilon)^{1/2}\|v\|_2$, which means it is between $(1 - \varepsilon/2)\|v\|_2$ and $(1 + \varepsilon/2)\|v\|_2$.

Note that since the updates to the AMS sketch can be positive or negative, it can be used to measure the Euclidean distance between two vectors v and u: we can build an AMS sketch of v and one of $-u$, and merge them together by adding the sketches. Note also that a sketch of $-u$ can be obtained from a sketch of u by negating all the counter values.

Applications

The sketch can also be applied to estimate the inner-product between a pair of vectors. A similar analysis shows that the inner product of corresponding rows of two sketches (formed with the same parameters and using the same hash functions) is an unbiased estimator for the inner product of the vectors. This use of the summary to estimate the inner product of vectors was described in a follow-up work by Alon, Matias, Gibbons and Szegedy [2], and the analysis was similarly generalized to the fast version by Cormode and Garofalakis [3]. The ability to capture norms and inner products in Euclidean space means that these sketches have found many applications in settings where there are high dimensional vectors, such as machine learning and data mining.

URLs to Code and Data Sets

Sample implementations are widely available in a variety of languages.

C code is given by the MassDal code bank: `http://www.cs.rutgers.edu/~muthu/massdal-code-index.html`.
C++ code due to Marios Hadjieleftheriou is available from `http://hadjieleftheriou.com/sketches/index.html`.

Cross-References

Count-Min Sketch

Recommended Reading

1. Alon N, Matias Y, Szegedy M (1996) The space complexity of approximating the frequency moments. In: ACM Symposium on Theory of Computing, pp 20–29
2. Alon N, Gibbons P, Matias Y, Szegedy M (1999) Tracking join and self-join sizes in limited storage. In: ACM Principles of Database Systems, pp 10–20
3. Cormode G, Garofalakis M (2005) Sketching streams through the net: Distributed approximate query tracking. In: International Conference on Very Large Data Bases
4. Thorup M, Zhang Y (2004) Tabulation based 4-universal hashing with applications to second moment estimation. In: ACM-SIAM Symposium on Discrete Algorithms