Sketch based algorithms in Machine Learning

Vipul Venkataraman

Indian Institute of Technology Bombay
Table of Contents

1 Introduction

2 Overview
   - Problems

3 Problem 1
   - Classification Problem
   - Count min sketch
   - The Algorithm
   - Analysis

4 Problem 2
   - Sparse Data Problem
   - The Algorithm
   - Two-way association in Boolean Data
We aim to discuss the various applications of Sketch based algorithms in Machine Learning.

Sketches are a class of data stream summaries, typically formed by linear projections of source data with appropriate (pseudo) random vectors.

An infinite number of data items arrive continuously, whereas the memory capacity is bounded by a small size.

Every item is seen once.

The goal is to use small memory to answer interesting queries with strong precision guarantees.
A Sketch based approach to Classification of High Cardinality Data Streams

- We tackle the problem of massive-domain stream classification where each attribute can take on one of a large number of possible values.

A Sketch based sampling algorithm on Sparse Data

- We study a sketch-based sampling algorithm, which effectively exploits the data sparsity.
- Combines the advantages of both conventional random sampling and more modern randomized algorithms such as local sensitive hashing.
In the classification problem, we use a labeled training data set in order to supervise the classification of unlabeled data instances.

In this problem, we will use the well known count-min sketch to the problem of classification of data streams.

Some examples:

- Internet applications: The number of possible source and destination addresses can be very large.
- Supermarket transactions: The individual items are often drawn from millions of possibilities.
Why Sketch based?

- 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.
- The one-pass constraint for data stream computation makes the problem even more challenging.
We have a $d \times w$ array $CM[i,j]; 1 \leq i \leq d$ and $1 \leq j \leq w$

We use $d$ independent hash functions $h_1, \ldots, h_d$ from $[H]$ to $[w]$, where $H$ is an integer.

We build the count-min sketch array in the following way: for every $x$, an input element we update the sketch by adding $1$ to the $CM[x,h_i(x)]$ for each $i \in [d]$.

Upon a query for a given element $x$, we return the value of the bin with the least value.
The data stream $D$ contains $d$-dimensional records which are denoted by $X_1, X_2, ..., X_N, ...$.

Associated with each record is a class which is drawn from the index \{1,...,k\}.

The attributes of record $X_i$ are denoted by $(x_{i1}, ..., x_{id})$.

The attribute value $x_{in}$ is drawn from the unordered domain set $J_n = \{v_{n1}^{\text{m}}, ..., v_{mn}^{\text{m}}\}$.

The value of $M^n$ can be very large and may range in the order of billions.
Even though an extremely large number of attribute-value combinations may be possible, only a limited number of these possibilities are usually relevant for classification purposes.

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.

A sketch-based approach is very effective in such a scenario.
Algorithm \textit{SketchUpdate}(Labeled Data Stream: \textit{D}, NumClasses: \textit{k}, MaxDim: \textit{r})

begin
    Initialize \textit{k} sketch tables of size \textit{w} \times \textit{h} each with zero counts in each entry;

repeat
    Receive next data point \textit{X} from \textit{D};
    Add 1 to each of the sketch counts in the (class specific) table for all \textit{L} value-combinations in \textit{X} with dimensionality less than \textit{r};

until (all points in \textit{D} have been processed);

end
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.

This new super-string is then hashed into the sketch table as if it is the attribute value for the special super-item $V$.

$L$ 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.
Definitions

- 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 $i$.
- Each super-item $V$ containing a concatenation of the attribute value strings along with the dimension indices for which these strings belong to.
- 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 $g_i(V)$.
- The discriminatory power $\theta_i(V)$ of the super-item $V$ is defined as

$$\theta_i(V) = \frac{f_i(V) - g_i(V)}{f_i(V)}$$
Significance of discriminatory power

- The larger the value of $\theta_i(V)$, the greater the correlation between the attribute-combination V and the class $i$.
- In addition, we have interest in those combinations of attribute values which occur in at least a fraction $s$ of the records belonging to any class $i$.
- We define an attribute value-combination V as $(\theta,s)$-discriminatory with respect to class i, if $f_i(V) \geq s$ and $\theta_i(V) \geq \theta$. 

Vipul Venkataraman | Sketch based algorithms in Machine Learning
**Algorithm** SketchClassify(Test Data Point: Y, NumClasses: k, MaxDim: r)

**begin**
  Determine all $L$ value-combinations specific to test instance;
  Use sketch-based approach to determine which value-combinations are $(\theta, s)$-discriminatory;
  Add 1 vote to each class for which a pattern is $(\theta, s)$-discriminatory;
  Report class with the largest number of votes;
**end**
With probability at least \((1 - \delta)\), the values of \(f_i(V)\) and \(g_i(V)\) are over-estimated to within \(L.\epsilon\) of their true values when we use sketch tables with size \(w = \lceil \ln(\frac{1}{\delta}) \rceil\) and \(h = \lceil \frac{e}{\epsilon} \rceil\) \((from\ the\ original\ count-min\ sketch\ paper)\).

Let \(\beta_i(V)\) be the estimated value of \(\theta_i(V)\) for an attribute-combination \(V\) with \(f_i(V) \geq s\) and \(f_i(V) \geq g_i(V)\). Let \(\epsilon\) be chosen such that \(\epsilon' = \epsilon \cdot \frac{L}{s} \ll 1\).

Then, with probability at least \((1 - 2.\delta)\) it is the case that \(\beta_i(V) \in (\theta_i(V) - \epsilon', \theta_i(V) + \epsilon')\).
The size of the sketch table is given by $O(e.\ln(\frac{1}{\delta}).\frac{L}{\epsilon})$ and there are $k$ of them.

The algorithm for updating the sketch table requires $O(ln(\frac{1}{\delta}).L)$ operations for each record.
We present a sketch-based sampling algorithm, which exploits the data sparsity.

We consider a data matrix $A$ of $n$ rows and $D$ columns, $n$ is the number of data points and $D$ is the number of features.

For sparse data, conventional random sampling may not work well because most of the samples are zeros.

Sampling fixed $D_s$ columns from the data matrix is also inflexible because different rows may have very different sparsity factors, defined as the percentages of non-zero elements.
Why Sampling?

- Sampling can speed up computations
- Sampling can save memory space. The original data are usually so large that they have to be stored on disks. Disk operations are often the bottleneck in databases and search engines. A sample may be small enough to reside in the main memory
- Sampling can generate stable fingerprint. Various hashing or sketching algorithms, can produce a sketch of the data, which is relatively insensitive to changes in the original data
- There are two basic strategies of sampling. The conventional approach is to draw random samples from the data. This approach is simple but often suffers from inaccuracy
- A different strategy is sketching, which may be regarded as special-purpose lossy data compressions. Sketching involves scanning the data at least once
Our sketch-based algorithm only samples the non-zero elements with flexible sample sizes for different data points.

We first consider only the non-zero entries in each row, which is different from the random sampling method.

We then sample $K_i$ non-zero elements from each row and take the minimum amongst all the maximum values of the rows:

$$D_s = \min(max(ID(K_1)), max(ID(K_2)), \ldots, max(ID(K_n)))$$

We can get exactly the same samples as if we directly sampled the first $D_s$ columns from the data matrix.

This way, we can convert sketches into random samples by conditioning on $D_s$. 
The algorithm consists of the following steps

- Construct sketches for all data points
- Construct equivalent random samples from sketches online. Depending on the goal, we can construct different random samples from the same sketches
- Estimate the original space. This step can be very simple, by scaling up (by a factor of $D / D_s$) any summary statistics computed from the samples
- The estimation task will be slightly more involving but still follows simple statistical principles
Note that in order for our algorithm to work, we have to make sure that the columns are random.

For this, we apply a random permutation $\pi$ on the column IDs:

$$\pi : \Omega \rightarrow \Omega, \Omega = \{1, 2, 3, \ldots, D\}.$$

The effective sample size, $D_s$ is computed online.

$$D_s = \min(\max(ID(K_1)), \max(ID(K_2)), \ldots, \max(ID(K_n)))$$

where, $K_i$ is the sketch corresponding to the row $u_i$. 
Two-way association in Boolean Data

- Suppose, we are interested in the intersections among $m$ rows of boolean data, which in terms of postings, are denoted by $P_1, P_2, ..., P_m$.
- There are $N = 2^m$ different combinations of intersections, denoted by $x_1, x_2, ..., x_N$

\[
x_1 = |P_1 \cap P_2 \cap ... \cap P_{m-1} \cap P_m|,
\]

\[
x_2 = |P_1 \cap P_2 \cap ... \cap P_{m-1} \cap \neg P_m|,
\]

.....

\[
x_N = |\neg P_1 \cap \neg P_2 \cap ... \cap \neg P_{m-1} \cap \neg P_m|
\]
- For each $P_i$, we take the smallest $k_i$ elements to form a sketch, $K_i$. Let,

\[
D_s = \min(\max(K_1), \max(K_2), .., \max(K_m))
\]
- After excluding the elements in all $K_i$s that are larger than $D_s$, we intersect the trimmed sketches to generate the sample counts
Let the samples obtained be $S = [s_1, s_2, ..., s_N]^T$

Let the original space be $X = [x_1, x_2, ..., x_N]^T$

The most straightforward estimator would be

$$x'_i = \frac{D}{D_s}s_i,$$

where $1 \leq i \leq N$
1. On Classification of High - Cardinality Data Streams, Charu C. Aggarwal, Philip S. Yu
2. A Sketch - based Sampling Algorithm on Sparse Data, Ping Li, Kenneth W. Church, Trevor J. Hastle