Coresets for Vector Summarization with Applications to Network Graphs

Dan Feldman, Sedat Ozer, Daniela Rus

Computer Science and Artificial Intelligence Laboratory, MIT

Presented by Zexi Huang

October 27, 2017

Outline

Outline

Background

- Data availability is not and is a problem.
  - GPS traces, phone call histories and social media postings can be used to identify social structures and predict activity patterns.
  - When the number of agents in a network becomes extremely large, algorithm running on it can become intractable due to lack of memory.
- Specifically, to represent a social network with a graph model, $O(n^2)$ space is required for the adjacency (proximity) matrix, where n is the number of nodes.
- To derive such a proximity matrix from GPS data, proximity vectors, based on distance between each pair of nodes, should be average over time.

Contribution

- They design an algorithm that maintains a compact representation for streaming proximity data.
  - For n nodes, only $O(Nn\log T)$ is needed instead of $O(n^2)$, where N is an error parameter, and T is number of elements in stream.
- They prove that the error introduced in this sparse representation is bounded by $\frac{1}{N}$ times the variance of elements in stream.

Outline

Overview

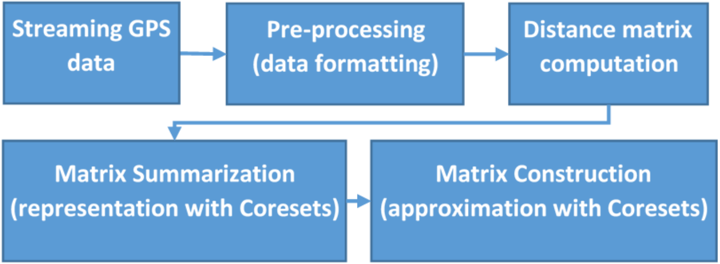


Figure 1: The overview of the framework.

GPS data and proximity

- Input: a stream of T GPS points $(time, userID, longitude, latitude)$.
- Maintain in memory:
  - pos: an array of length n storing current locations of users.
  - proximity coresets: for each node, a $O(N \log T)$ number of sparse proximity vectors.
- Whenever an new record of node u arrives:
  1. $pos[u]$ is updated according to the current position.
  2. For each node v, $dist(u, v) = ||pos[u] - pos[v]||$ is computed, $p = (0, ..., 0, prox_u = e^{-dist(u,v)}, 0, ..., 0)$ is generated and added to the proximity coreset of node v.

Problem

Problem

Consider a stream of T sparse vectors $p_1, p_2, ..., p_T$. Maintain a subset of $N \ll T$ input vectors, and a corresponding vector of positive reals (weights), $w_1, w_2, ..., w_N$, where the sum $\hat{p} := \sum_{i=1}^{N} w_i p_i$ approximates the sum $\bar{p} := \sum_{i=1}^{T} p_i$ up to a provably small error that depends on the variance $var(p) := \sum_{i=1}^{T} ||p_i - \bar{p}||^2$ and an error parameter $\epsilon := f(N)$,

$$||\bar{p} - \hat{p}||^2 \leq \epsilon var(p) \quad (1)$$

Outline

Off-line Coreset Algorithm

Algorithm 1 Coreset(P, u, ϵ)

$\bar{p} \leftarrow \sum_{j=1}^{T} u_j p_j, x \leftarrow \sum_{j=1}^{T} u_j ||p_j - \bar{p}||$

for $i \leftarrow 1$ to n **do**

  $q_i \leftarrow \frac{(p_i - \bar{p}, x)}{||(p_i - \bar{p}, x)||}$, $s_i \leftarrow \frac{u_i ||(p_i - \bar{p}, x)||}{\sum_{j=1}^{T} u_j ||(p_j - \bar{p}, x)||}$

end for

$A \leftarrow$ collection of shifted q_i.

Use Frank-Wolfe method to find a coreset S of $\lceil \alpha/\epsilon \rceil$ vectors and the respective weight vector w', where α is a constant.

for $i \leftarrow 1$ to $\lceil \alpha/\epsilon \rceil$ **do**

  $w_i'' \leftarrow \frac{\sum_{j=1}^{T} u_j ||(p_j - \bar{p}, x)|| w_j'}{||(p_i - \bar{p}, x)||}$

  $w_i = \frac{w_i''}{\sum_{j=1}^{\lceil \alpha/\epsilon \rceil} w_j''}$

end for

return (S, w)

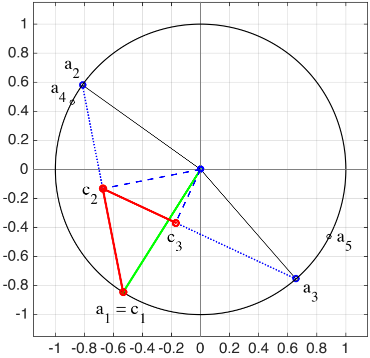
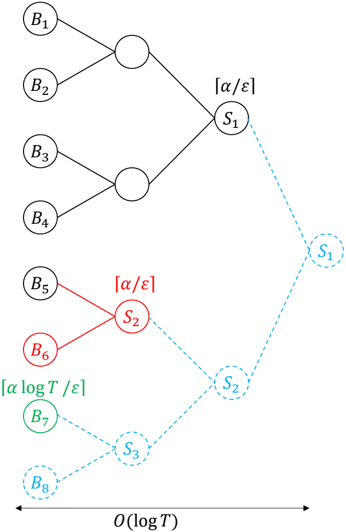


Figure 2: Illustration of the first three steps for Frank-Wolfe method.

Streaming Algorithm

Algorithm 2 Streaming-Coreset$(stream, \epsilon)$

```
while stream is not empty do
    B_i ← next ⌈α log T/ϵ⌉ vectors in stream.
    Insert B_i into the binary tree.
    while The tree can grow upwards do
        Form a new parent S_i
        S_i ← Coreset(C_i1 ∪ C_i2, w_i1 ∪ w_i2, ϵ)
    end while
end while
S ← union of all root nodes.
w ← union of all weight vectors of root nodes.
return (S, w)
```



Spatial complexity:

$$O(\lceil \tfrac{\alpha}{\epsilon} \log T \rceil) + \lceil \tfrac{\alpha}{\epsilon} \rceil O(\log T) = O(\tfrac{1}{\epsilon} \log T)$$

Parallel Computation

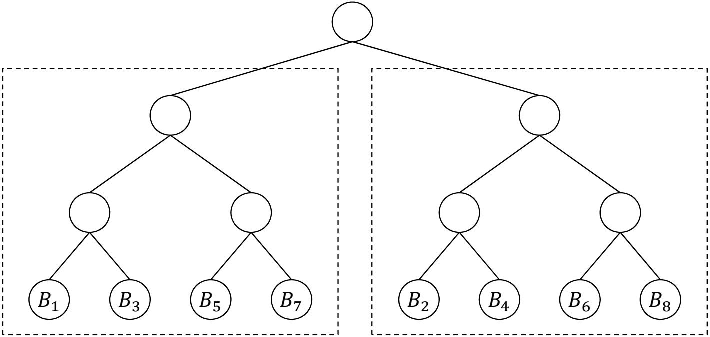


Figure 3: Coreset computation of streaming data that is distributed into $M = 2$ machines. The odd/even vectors in the stream are compressed by the machine on the left/right, respectively. A server (possibly one of these machines) can collect the root nodes of each machine to obtain the final coreset.

Outline

Theorem

Let $u \in D^T$ *be a distribution over a set* $P = \{p_1, ..., p_T\}$ *of* T *vectors in* R^n*, and let* $N \geq 1$*. Denote* (S, w) *as the output of a call to Coreset*$(P, u, 1/N)$*. Then* $w \in D^T$ *consists* $O(N)$ *non-zero entries, such that the sum* $\bar{p} = \sum_{i=1}^{T} u_i p_i$ *deviates from the sum* $\hat{p} = \sum_{i=1}^{T} w_i p_i$ *by at most a* $(1/N)$*-fraction of the variance* $var_u = \sum_{i=1}^{T} u_i ||p_i - \hat{p}||$*, i.e.,*

$$||\bar{p} - \hat{p}||^2 \leq \frac{1}{N} var_u \tag{2}$$

Augmentation and α

Augmentation

$x \leftarrow \sum_{j=1}^{T} u_j ||p_j - \bar{p}||$, $q_i \leftarrow \frac{(p_i - \bar{p}, x)}{||(p_i - \bar{p}, x)||}$, $s_i \leftarrow \frac{u_i ||(p_i - \bar{p}, x)||}{\sum_{j=1}^{T} u_j ||(p_j - \bar{p}, x)||}$
$\Rightarrow || \sum_i (s_i - w'_i) q_i ||^2 = || \sum_i s_i q_i - \sum_j w'_j q_j ||^2 \leq \frac{1}{N}$, where w has at most N non-zero entries.

α

- It suffices to prove that $||\bar{p} - \hat{p}||^2 \leq \frac{\alpha}{N} var_u$ for $O(N)$ vectors.
  - Replacing N with N/α leads to $O(N/\alpha) = O(N)$ complexity.
- By loosing the upper bound with $\frac{\alpha}{N} var_u$, it is easy to bound other quantities.
- $\alpha = 3$ is sufficiently large for the theorem to hold.

Outline

Experiment 1

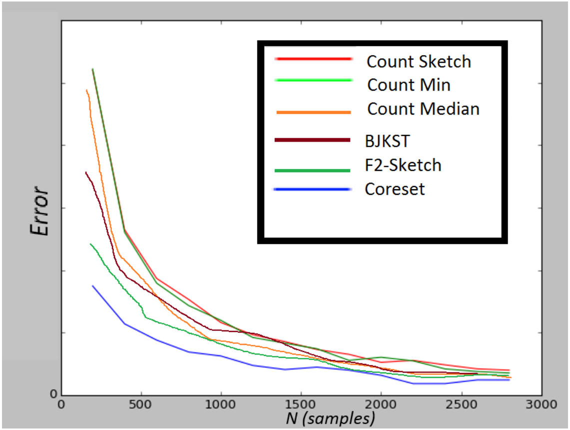


Figure 4: Coreset algorithm compared with other sketch algorithms on a synthetic standard gaussion dataset.

Experiment 2

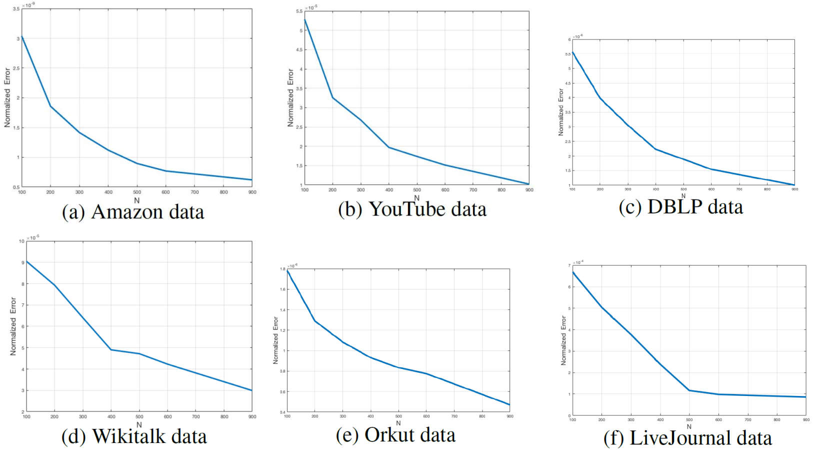


Figure 5: Coreset algorithm on several networks from Stanford Large Network Dataset (SNAP).

Experiment 3

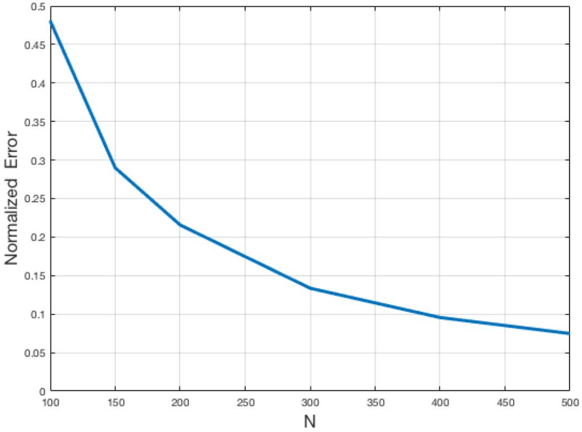


Figure 6: Coreset algorithm on NYC GPS dataset, which contains 13,249 taxi cabs and 14,776,616 GPS entries.

Outline

Remarks

- Summary: A coreset algorithm is proposed to summarize streaming data sets, which takes a stream of vectors as input and maintain their sum using small memory.
- Slight problems/regrets:
  - Mixed use of n: First number of nodes, then number of entries.
  - No error guarantee for streaming algorithm.