CS 395T: Sublinear Algorithms Fall 2014

Lecture 8 — Sept 23, 2014

Prof. Eric Price *Scribe: Xinyang Yi*

1 Overview

In today's class, we will talk about 'coresets'. Generally, coresets is a compressed representation of the original data. The concept of coresets can be utilized to solve the following problems in a streaming setting, if we only have insertion of elements and no deletions:

- Spectral approximation of a graph.
- k-median clustering.
- k-mean clustering.
- Principal component analysis (PCA).

For these problems, our task is to find a coreset for the original data and solve the problem on the coreset. In addition, as we will see, using the technique "merge and reduce" of coresets, we can find streaming algorithms that solve these problems with insertions. In this class, we will focus on the k-median problem.

2 K-median problem

In particular, we consider the **k-median** problem in a grid plane. Formally, let $[\triangle]$ denote the set of natural numbers up to $\triangle$. Let P^n denote a set of n points on $[\triangle]^2$,i.e.,

$$P^n = \{p_1, p_2, ..., p_n \in [\triangle]^2\}.$$

The k-median problem is to find k centers such that the total distance from any point to the nearest center is minimized. Formally, we want to find k points $c_1, c_2, ..., c_k \in [\triangle]^2$ such that the following quantity is minimized:

$$\sum_{j=1}^{n} \min_{j \in [k]} \|p_i - c_j\|.$$

Here $\|\cdot\|$ denote some distance metric. In the special case where $\|\cdot\|$ is l_1 norm and points are distributed in 1-D space, the center that minimizes the total distance is actually the coordinate-wise median of the cluster assigned to that center. This is the reason why the problem is called k-median. Departing from this problem, if we wanted to minimize the squared Euclidean distance, then centers would be the *mean* of their clusters. This would be the **k-means** problem.

For any set of k points $C = \{c_1, c_2, ..., c_k \in [\triangle]^2\}$, we define

$$d(P, C) = \sum_{j=1}^{n} \min_{j \in [k]} \|p_i - c_j\|.$$

For any optimal solution, the corresponding optimal summation of distance is denoted as

$$OPT(P, k) = \min_{C \,:\, C=k} d(P, C).$$

Generally, the k-median problem is NP-hard. The ϵ-approximate k-median problem is to find a C such that

$$d(P, C) \leq (1 + \epsilon)OPT(P, k).$$

A nearly linear time approximation algorithm is proposed by Kolliopoulos and Rao in [KR99].

Theorem 1. *([KR99]) The exists an algorithm that solves ϵ-approximate in time $O((1/\epsilon)^{O(1/\epsilon)} n \log n \log k)$.*

3 Coreset for k-median

We now introduce how to use coreset to solve k-median problem in a stream. The basic idea to find a set of weighted points to replace the original potentially big set of points. Then we can store the compressed point set (called coreset) and run any existing approximate algorithm on it. So the coreset should capture the underlying structure such that the solution for coreset is also good for original points. In particular, let (S, W) denote a set of points $\{s_1, s_2, ..., s_t\}$ with weights $\{w_1, w_2, ..., w_t\}$.

Definition 2. *A (k, ϵ)-coreset for P is a weighted set of points (S, W) such that*

$$d(P, (S, W)) \leq \epsilon \cdot OPT(P, k).$$

The natural question is how to construct a coreset for a given set of points. We hope the constructed coreset has small size t. From existing literature, we already know these exists one algorithm that works with guarantee $t = O(\frac{k^2}{\epsilon^2})$ and another one that works with $t = O(\frac{k \log \triangle}{\epsilon^2})$. We will introduce the second algorithm later. Before that, let's investigate the following property of coreset.

Theorem 3. *Let (S, W) be an (k, ϵ)-coreset for P. Suppose set $\widetilde{C}$ is an optimal solution for k-median problem on (S, W), i.e.,*

$$d((S, W), \widetilde{C}) = OPT((S, W), k).$$

Then we have

$$d(P, \widetilde{C}) \leq (1 + 2\epsilon) \cdot OPT(P, k).$$

Remarks. (1) For weighted set (S, W), $d(\cdot, \cdot)$ is similarly defined as

$$d((S, W), C) = \sum_{i=1}^{t} \min_{j \in [k]} w_i \|s_i - c_j\|$$

and

$$d(P,(S,W)) = \sum_{i=1}^{n} \min_{j\in[t]} \|p_i - s_j\|.$$

(2) Consequently, Theorem 3 tells us an optimal solution for coreset is also a good approximate solution for the original problem.

Now we prove Theorem 3.

Proof. First, by triangle inequality, we observe that for any set of points C,

$$d((S,W),C) \le d(P,C) + d(P,(S,W)) \tag{1}$$

and

$$d(P,C) \le d((S,W),C) + d(P,(S,W)). \tag{2}$$

Suppose $\overline{C}$ is an optimal solution for k-median on P, plug $\overline{C}$ into (1) results in

$$d((S,W),\overline{C}) \le (1+\epsilon)\cdot OPT(P,k).$$

By utilizing (2) and plug $\widetilde{C}$ into it, we have

$$\begin{aligned} d(P,\widetilde{C}) &\le d((S,W),\widetilde{C}) + d(P,(S,W)) \\ &\le d((S,W),\overline{C}) + d(P,(S,W)) \\ &\le (1+\epsilon)\cdot OPT(P,k) + \epsilon\cdot OPT(P,k) \\ &\le (1+2\epsilon)\cdot OPT(P,k). \end{aligned}$$

□

4 Coreset Construction

Now we show how to construct coreset with $t \le O(\frac{k\log\triangle}{\epsilon^2})$. Let's first consider the slightly easier case when we know an optimal solution for P, say $\overline{C} = \{c_1, c_2, ..., c_k\}$. Then we are able to construct coreset based on $\overline{C}$. (We will in fact show that any constant factor approximation to $\overline{C}$ is sufficient.) The basic idea is to construct a finite number of points S_j for each point c_j such that for any points $p \in [\triangle]^2$,

$$\min_{p'\in S_j} \|p - p'\| \le \epsilon\|p - c_j\|.$$

In order to meet our target $O(\frac{k\log\triangle}{\epsilon^2})$, we expect $|S_j| = O(\frac{\log\triangle}{\epsilon^2})$. One way to achieve this is to construct a heterogeneous scale grid around c_j such that the length of grid side is proportional to the distance of its center from c_j. Without loss of generality, let's assume c_j is the center of $[\triangle]^2$, i.e., $(\triangle/2, \triangle/2)$. Let $T(S)$ denote the square with side length S centered at $(\triangle/2, \triangle/2)$. Let $E(T(S))$ denote the set points uniformly distributed on the edge of $T(S)$ and separate the edge into $O(\frac{1}{\epsilon})$ segments. Then S_j can be explicitly constructed as the following set of points:

$$S_j = \{(x,y) : (x,y) \in E(T((1+\epsilon)^k)), k \in [\log_{1+\epsilon}(\triangle)]\}.$$

Then it's easy to show that for any point p, there exists an absolute constant such that

$$\min_{p' \in S_j} \|p - p'\| \leq C\epsilon \|p - c_j\|.$$

In addition we note that $|S_j| = O(\frac{1}{\epsilon} \frac{\log \triangle}{\log(1+\epsilon)}) = O(\frac{\log \triangle}{\epsilon^2})$. We summarize our analysis in the following results.

Lemma 4. *Using $\overline{C}$, an optimal solution for k-median problem on P, there exists a method to construct a set of points S such that*

$$d(P, S) \leq \epsilon \cdot d(P, \overline{C})$$

and

$$|S| \leq O(\frac{k \log p}{\epsilon^2}).$$

We will discuss the concept of covering and packing number in the future. At that time we may have a more intuitive understanding of the construction.

Now the unsolved problem is that we do not know the optimal solution $\overline{C}$. But suppose we know an approximate solution C such that for some $c > 1$

$$d(p, C) \leq c d(p, \overline{C}),$$

then we can construct a coreset S based on C using the techniques we discussed before. Then we have

$$\begin{aligned} d(P, S) &\leq \epsilon d(P, C) \\ &\leq c\epsilon d(P, \overline{C}) \\ &= c\epsilon \cdot OPT(P, k). \end{aligned}$$

Therefore a constant approximate solution C is enough for constructing a (k, ϵ)-coreset. Recall that in Theorem 3, we there exists an algorithm that solves approximate k-median problem efficiently. We now have the following result.

Theorem 5. *For P, there exists an algorithm that returns a (k, ϵ)-coreset with size $O(\frac{k \log \triangle}{\epsilon^2})$ in time $O(n \log n \log k)$.*

5 Merge and Reduce

We have shown how to construct coreset for k-median problem. Next, we introduce a merge and reduce technique for coreset. This method will help us compress incoming data points sequentially thus results in a streaming algorithm for k-median with insertions.

For notation simplicity, we drop the subscript and superscript of (weighted) point set, say P (S, W), just use $P, (S, W)$. We use $A + B$ denote the union of two set of points.

Suppose $(S^{(1)}, W^{(1)})$ is (k, ϵ)-coreset for $P^{(1)}$ and $(S^{(2)}, W^{(2)})$ is (k, ϵ)-coreset for $P^{(2)}$. Consider the problem how to get a coreset for $P^{(1)} + P^{(2)}$ using their own coresets. A simple way would be running algorithm to construct (k, ϵ')-coreset, denoted as $(\overline{S}, \overline{W})$, for $(S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)})$. We expect that $(\overline{S}, \overline{W})$ is a good coreset for $P^{(1)} + P^{(2)}$.

Theorem 6. *Suppose $(\overline{S}, \overline{W})$ is (k, ϵ')-coreset for $(S^{(1)}, W^{(1)})$ and $(S^{(2)}, W^{(2)})$ which are (k, ϵ)-coresets for $P^{(1)}, P^{(2)}$ respectively. Then it's $(k,\ \epsilon + (1+2\epsilon)\epsilon')$-coreset for $P^{(1)} + P^{(2)}$.*

Proof. First, we note that

$$OPT(P^{(1)}, k) + OPT(P^{(2)}, k) \leq OPT(P^{(1)} + P^{(2)}, k). \tag{3}$$

Then

$$\begin{aligned}
&d(P^{(1)} + P^{(2)}, (\overline{S}, \overline{W})) \\
\overset{(a)}{\leq} &d(P^{(1)} + P^{(2)}, (S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)})) + d((S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)}), (\overline{S}, \overline{W})) \\
= &d(P^{(1)}, (S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)})) + d(P^{(2)}, (S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)})) \\
&+ d((S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)}), (\overline{S}, \overline{W})) \\
\leq &d(P^{(1)}, (S^{(1)}, W^{(1)})) + d(P^{(2)}, (S^{(2)}, W^{(2)})) + d((S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)}), (\overline{S}, \overline{W})) \\
\overset{(b)}{\leq} &\epsilon \cdot OPT(P^{(1)}, k) + \epsilon \cdot OPT(P^{(2)}, k) + \epsilon' \cdot OPT((S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)}), k) \\
\overset{(c)}{\leq} &\epsilon \cdot OPT(P^{(1)} + P^{(2)}, k) + \epsilon' \cdot OPT((S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)}), k).
\end{aligned}$$

Here (a) follows from triangle inequality. (b) follows from the definition of coreset. (c) follows from (3). We need to provide an upper bound for $OPT((S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)}), k)$. Note that $(S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)})$ is a (ϵ, k)-coreset for $P^{(1)} + P^{(2)}$. By using Theorem 3, we have

$$OPT((S^{(1)} + S^{(2)}, W^{(1)} + W^{(2)}), k) \leq (1 + 2\epsilon) OPT(P^{(1)} + P^{(2)}, k).$$

Finally we conclude that

$$d(P^{(1)} + P^{(2)}, (\overline{S}, \overline{W})) \leq (\epsilon + (1 + 2\epsilon)\epsilon') OPT(P^{(1)} + P^{(2)}, k)$$

□

Streaming Algorithm. Suppose we have total n points and our points come in batches with size m. We construct a series of coresets in multiple levels and each level maintain a coreset with size m. Level 0 contains the raw points. Each level at most keep one batch except for the level 0. If level i receives a coreset from level $i-1$, if it already has one coreset, merge them to be a new coreset with size m and send it to level $i+1$. If it has empty coreset, just keep the received coreset. Each level apply the same (ϵ', k)-coreset construction algorithm. Hence we have

$$m = \frac{k \log \triangle}{\epsilon'^2}.$$

This procedure will result in $O(\log n)$ levels. Following from Theorem 6, the coreset constructed in the top level a $(\epsilon' \log n, k)$-coreset for the total n points. Equivalently, if we want a (ϵ, k)-coreset at the end of our streaming algorithm, we can set $\epsilon' = \epsilon / \log n$, then the space complexity of our streaming algorithm turns out to be $O(\frac{k \log \triangle \log^3 n}{\epsilon^2})$.

References

[KR99] Stavros G. Kolliopoulos, Satish Rao. A Nearly Linear-Time Approximation Scheme for the Euclidean kappa-median Problem. *ESA 1999*:378-389.