COL870: Clustering Algorithms

Ragesh Jaiswal, CSE, IIT Delhi
Streaming Clustering
Streaming Clustering

- Here are some of the results known for the streaming $k$-means/median.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Space requirement</th>
<th>Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>[GNMO00]</td>
<td>$O(n^c)$</td>
<td>$O(2^{1/c})$</td>
</tr>
<tr>
<td>[MCP03]</td>
<td>$O(k \cdot \text{polylog}(n))$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>[C09]</td>
<td>$O(\frac{dk}{\epsilon}(\log n)^8)$</td>
<td>$(1 + \epsilon)$</td>
</tr>
</tbody>
</table>

- We already studied the first one. We shall skip the second result (project topic). The third result is through a concept known as coreset which we will not discuss (project topic).
- We will look at the notion of coresets and see how to use them to construct streaming algorithms.
Before we study the concept of coresets, let us see whether it is possible to get a $(1 + \varepsilon)$-approximation for the $k$-means problem.

There are algorithms that run in time $O(nd \cdot 2^{\tilde{O}(k/\varepsilon)})$ and give a $(1 + \varepsilon)$-approximation guarantee for the $k$-means problem.
A corset is a subset of input such that we can get a good approximation to the original input by solving the optimisation problem directly on the corset.

- \((k, \varepsilon)\)-coreset for \(k\)-means/median: For a weighted point set \(P \subset \mathbb{R}^d\), a weighted set \(S \subset \mathbb{R}^d\) is a \((k, \varepsilon)\)-coreset of \(P\) for \(k\)-means/median clustering, if for any set \(C\) of \(k\) points in \(\mathbb{R}^d\), we have:

\[
(1 - \varepsilon) \cdot \Phi_C(P) \leq \Phi_C(S) \leq (1 + \varepsilon) \cdot \Phi_C(P).
\]
(\(k, \varepsilon\))-coreset for \(k\)-means/median: For a weighted point set \(P \subset \mathbb{R}^d\), a weighted set \(S \subset \mathbb{R}^d\) is a \((k, \varepsilon)\)-coreset of \(P\) for \(k\)-means/median clustering, if for any set \(C\) of \(k\) points in \(\mathbb{R}^d\), we have:

\[(1 - \varepsilon) \cdot \Phi_C(P) \leq \Phi_C(S) \leq (1 + \varepsilon) \cdot \Phi_C(P).\]

There is an algorithm that outputs a \((k, \varepsilon)\) coreset of size \(O(k^2 \varepsilon^{-2} (\log n)^2)\) in a general metric space and a \((k, \varepsilon)\)-coreset of size \(O(dk^2 \varepsilon^{-2} \log n \log (k/\varepsilon))\) in \(\mathbb{R}^d\).
Claim 1: If $C_1$ and $C_2$ are the $(k, \varepsilon)$ for disjoint sets $P_1$ and $P_2$ respectively, then $C_1 \cup C_2$ is a $(k, \varepsilon)$ coreset for $P_1 \cup P_2$. 
Claim 1: If \( C_1 \) and \( C_2 \) are the \((k, \varepsilon)\) coresets for disjoint sets \( P_1 \) and \( P_2 \) respectively, then \( C_1 \cup C_2 \) is a \((k, \varepsilon)\) coreset for \( P_1 \cup P_2 \).

Claim 2: If \( C_1 \) is a \((k, \varepsilon)\)-coreset for \( C_2 \) and \( C_2 \) is a \((k, \delta)\)-coreset for \( C_3 \), then \( C_1 \) is a \((k, \varepsilon + \delta + \varepsilon \cdot \delta)\)-coreset for \( C_3 \).
Claim 1: If $C_1$ and $C_2$ are the $(k, \varepsilon)$ for disjoint sets $P_1$ and $P_2$ respectively, then $C_1 \cup C_2$ is a $(k, \varepsilon)$ coreset for $P_1 \cup P_2$.

Claim 2: If $C_1$ is a $(k, \varepsilon)$-coreset for $C_2$ and $C_2$ is a $(k, \delta)$-coreset for $C_3$, then $C_1$ is a $(k, \varepsilon + \delta + \varepsilon \cdot \delta)$-coreset for $C_3$.

How do we obtain a streaming algorithm for the $k$-means problem that uses only space that has logarithmic dependence on the stream size using the above claims?
- **Claim 1**: If $C_1$ and $C_2$ are the $(k, \varepsilon)$ for disjoint sets $P_1$ and $P_2$ respectively, then $C_1 \cup C_2$ is a $(k, \varepsilon)$ coreset for $P_1 \cup P_2$.

- **Claim 2**: If $C_1$ is a $(k, \varepsilon)$-coreset for $C_2$ and $C_2$ is a $(k, \delta)$-coreset for $C_3$, then $C_1$ is a $(k, \varepsilon + \delta + \varepsilon \cdot \delta)$-coreset for $C_3$.

- **Claim 3**: There is an algorithm that outputs a $(k, \varepsilon)$ coreset of size $O(k^2 \varepsilon^{-2} (\log n)^2)$ in a general metric space and a $(k, \varepsilon)$-coreset of size $O(dk^2 \varepsilon^{-2} \log n \log (k/\varepsilon))$.

- **Claim 4**: There are algorithms that run in time $O(nd \cdot 2^{\tilde{O}(k/\varepsilon)})$ and give a $(1 + \varepsilon)$-approximation guarantee for the $k$-means problem.

Consider hypothetical buckets $P_0, P_1, \ldots, P_{\lceil \log n \rceil}$ such that $|P_i| = 2^i \cdot M$, where $M = (k/\varepsilon)^2$.

As the data comes, we will try putting in the bucket $P_0$. In case this is full, we try to move the contents of $P_0$ to $P_1$ is possible and so on.

We try to maintain $(k, \delta_j)$-coreset for $P_j$ at all times, where $1 + \delta_j = \prod_{l=0}^{j} (1 + \rho_l)$ and $\rho_j = \frac{\varepsilon}{c(j+1)^2}$. 
- Claim 1: If $C_1$ and $C_2$ are the $(k, \varepsilon)$ for disjoint sets $P_1$ and $P_2$ respectively, then $C_1 \cup C_2$ is a $(k, \varepsilon)$ coreset for $P_1 \cup P_2$.
- Claim 2: If $C_1$ is a $(k, \varepsilon)$-coreset for $C_2$ and $C_2$ is a $(k, \delta)$-coreset for $C_3$, then $C_1$ is a $(k, \varepsilon + \delta + \varepsilon \cdot \delta)$-coreset for $C_3$.
- Claim 3: There is an algorithm that outputs a $(k, \varepsilon)$ coreset of size $O(k^2 \varepsilon^{-2} (\log n)^2)$ in a general metric space and a $(k, \varepsilon)$-coreset of size $O(dk^2 \varepsilon^{-2} \log n \log (k/\varepsilon))$.
- Claim 4: There are algorithms that run in time $O(nd \cdot 2^{\tilde{O}(k/\varepsilon)})$ and give a $(1 + \varepsilon)$-approximation guarantee for the $k$-means problem.

- Consider hypothetical buckets $P_0, P_1, ..., P_{\lceil \log n \rceil}$ such that $|P_i| = 2^i \cdot M$, where $M = (k/\varepsilon)^2$.
- As the data comes, we will try putting in the bucket $P_0$. In case this is full, we try to move the contents of $P_0$ to $P_1$ is possible and so on.
- We try to maintain $(k, \delta_j)$-coreset for $P_j$ at all times, where $1 + \delta_j = \prod_{l=0}^{j} (1 + \rho_l)$ and $\rho_j = \frac{\varepsilon}{c(j+1)^2}$.
- **Claim 1**: If $C_1$ and $C_2$ are the $(k, \varepsilon)$ for disjoint sets $P_1$ and $P_2$ respectively, then $C_1 \cup C_2$ is a $(k, \varepsilon)$ coreset for $P_1 \cup P_2$.

- **Claim 2**: If $C_1$ is a $(k, \varepsilon)$-coreset for $C_2$ and $C_2$ is a $(k, \delta)$-coreset for $C_3$, then $C_1$ is a $(k, \varepsilon + \delta + \varepsilon \cdot \delta)$-coreset for $C_3$.

- **Claim 3**: There is an algorithm that outputs a $(k, \varepsilon)$ coreset of size $O(k^2 \varepsilon^{-2} (\log n)^2)$ in a general metric space and a $(k, \varepsilon)$-coreset of size $O(dk^2 \varepsilon^{-2} \log n \log (k/\varepsilon))$.

- **Claim 4**: There are algorithms that run in time $O(nd \cdot 2^{O(k/\varepsilon)})$ and give a $(1 + \varepsilon)$-approximation guarantee for the $k$-means problem.

- **Claim 5**: There is a streaming algorithm that outputs a $(1 + \varepsilon)$-approximate solution using space $O((dk/\varepsilon)^2 \cdot (\log n)^8)$ and amortised update time $O(dk \cdot \text{polylog}(ndk/\varepsilon))$.

- Consider hypothetical buckets $P_0, P_1, \ldots, P_{\lceil \log n \rceil}$ such that $|P_i| = 2^i \cdot M$, where $M = (k/\varepsilon)^2$.

- As the data comes, we will try putting in the bucket $P_0$. In case this is full, we try to move the contents of $P_0$ to $P_1$ is possible and so on.

- We try to maintain $(k, \delta_j)$-coreset for $P_j$ at all times, where $1 + \delta_j = \prod_{l=0}^{j} (1 + \rho_l)$ and $\rho_l = \frac{\varepsilon}{c(j+1)^2}$.

- **Claim 5**: There is a streaming algorithm that outputs a $(1 + \varepsilon)$-approximate solution using space $O((dk/\varepsilon)^2 \cdot (\log n)^8)$ and amortised update time $O(dk \cdot \text{polylog}(ndk/\varepsilon))$.

Ragesh Jaiswal, CSE, IIT Delhi

COL870: Clustering Algorithms
End