Clustering Perturbation Resilient k-Median Instances

Maria-Florina Balcan and Yingyu Liang

Problem Setup

- **k-Median Clustering**: Given a set of n points in metric space, find centers \( x = \{x_1, \ldots, x_k\} \) to minimize \( \sum_{i \in P} \min_{c \in C} d(p_i, c) \).

- **New Direction**: exploit additional stability properties of the data \( \alpha \)-perturbation of \( d \): a function \( d' \) s.t. \( \forall p, q \in S, d'(p, q) \in [1, \alpha]d(p, q) \).

Definition. [Bilu-Linial, [CC10; Anawshi-Blum-Sheffer, [IP112] An instance is \( \alpha \)-perturbation resilient if the optimal clustering under \( \alpha \)-perturbation of the distance is unique and equal to the original optimal clustering.

Structure Property of \((\alpha, \epsilon)\)-PR k-Median

- **Theorem.** Assume \( \min_i |C_i| = \Omega(\epsilon n) \).

- **Definition.** [Balcan-Liang, ICALP12] An instance is \((\alpha, \epsilon)\)-perturbation resilient, if the optimal clustering under any \( \alpha \)-perturbation of distance can be obtained by moving at most \( \epsilon \) fraction of the points in the original optimal clustering.

Our Results

Efficient algorithm for \((\alpha, \epsilon)\)-PR k-median instances

- Produces \((1 + O(\epsilon/\rho))-\text{approx} \) for \( \alpha > 4 \), where \( \rho = \min_i |C_i|/n \).
- Improve over the bound \( \alpha > 2 + \sqrt{7} \ln \frac{n}{\epsilon} \) [Balcan-Liang, ICALP12]

Sublinear time algorithm for constructing implicit clustering

- Produces \(2(1 + O(\epsilon/\rho))-\text{approx} \) for \( \alpha > 4 \).
- Running time logarithmic in \( \# \) points

Algorithms

- **Algorithm 1** \((1 + O(\epsilon/\rho))-\text{approx} \) algorithm
  1. Generate a list of blobs as described above
  2. Use existing robust linkage algo to link them into a tree
  3. Use dynamic programming to get the lowest cost pruning

- **Algorithm 2** (Sublinear time algorithm)
  1. Sample points from the original data
  2. Run Algorithm 1 on the sample

Generating Blobs: continue

3. More difficult case: with bad points, \( n_C \) unknown
   Algo: Start with a threshold \( t \), build \( F_t, H_t \), pull out blobs

4. General case: with bad points, clusters have different sizes
   Algo: Pull out blobs as before
   Keypoint: when \( t = |C_i| \), all good points in \( C_i \) are pulled out

\[ \begin{align*}
F_t & \quad G_1 \quad G_2 \quad G_3 \quad G_4 \\
\Rightarrow & \quad B \\
H_t & \quad G_1 \quad G_2 \quad G_3 \quad G_4 \\
\Rightarrow & \quad B
\end{align*} \]

Analysis

- Each blob only contains good points from one cluster
- Blobs from the same cluster are first linked by robust linkage
- There is a pruning that assigns all good points correctly

Sublinear Time Algorithm

Let \( \mathcal{C} \) be the output of Algorithm 1 on a sample \( S' \) of size \( \Theta(\frac{1}{\epsilon}) \). To show cost\((S, \mathcal{C}) \approx \text{cost}(S, \mathcal{C})\):

- Average costs of \( S' \) and \( S \) are close on any set of centers, so it suffices to show cost\((S', \mathcal{C}) \approx \text{cost}(S', \mathcal{C})\).
- Algorithm 1 builds a tree with a pruning \( \mathcal{P} \) assigning all good points correctly. Let \( \mathcal{C}' \) be the best centers for it.
- Use cost\((\mathcal{P}, \mathcal{C}')\) as a bridge:
  - cost\((S', \mathcal{C}') \approx \text{cost}(\mathcal{C}, \mathcal{C}') \approx \text{cost}(\mathcal{P}, \mathcal{C}')\)
  - cost\((\mathcal{P}, \mathcal{C}') \leq 2 \times \text{cost}(\mathcal{P}, \mathcal{C}) \approx 2 \times \text{cost}(S', \mathcal{C})\)