Design and Analysis of Algorithms

Approximation Algorithms – Part 4

Pan Peng

School of Computer Science and Technology
University of Science and Technology of China

Outline

- Approximation algorithms for clustering problems
  - k-center
  - k-median
  - k-means
  - Uncapacitated Facility Location

Basic definitions for clustering problems

Basic definitions

Metric space:
given a set V and a distance function $d : V \times V \rightarrow \mathbb{R}^+$, a space (V, d) is a metric space if d satisfies the following metric properties:

- $d(u, v) = 0$ iff $u = v$ (reflexivity)
- $d(u, v) = d(v, u)$ for all $u, v \in V$ (symmetry)
- $d(u, v) + d(v, w) \geq d(v, w)$ for all $u, v, w \in V$ (triangle inequality)

Basic definitions

Metric space:
given a set V and a distance function $d : V \times V \to \mathbb{R}^+$, a space (V, d) is a metric space if d satisfies the following metric properties:

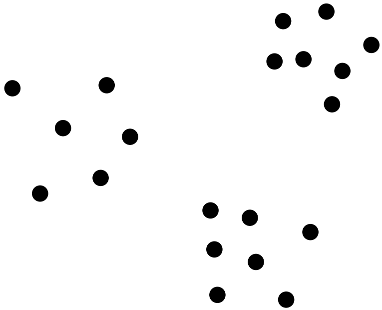
- $d(u, v) = 0$ iff $u = v$ (reflexivity)
- $d(u, v) = d(v, u)$ for all $u, v \in V$ (symmetry)
- $d(u, v) + d(v, w) \geq d(v, w)$ for all $u, v, w \in V$ (triangle inequality)

Notations:

- given two subsets $A, B \subseteq V$, let $d(A, B) = \min_{p \in A, q \in B} d(p, q)$
- given $p \in V$ and $A \subseteq V$, let $d(p, A) = \min_{q \in A} d(p, q)$

Center based clustering

Given n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (V, d) and an integer k.

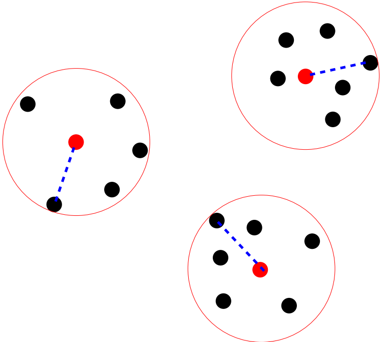


Center based clustering

Given n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (V, d) and an integer k.

Goal: cluster/partition P into k clusters $C_1, C_2, \ldots, C_k$ which are induced by choosing k centers $c_1, c_2, \ldots, c_k$ from V.

- Each point p_i is assigned to its nearest center from $c_1, \ldots, c_k$ and this induces a clustering.



Center based clustering

Typically, we want to choose $c_1, \ldots, c_k$ to minimize the clustering objective

$$\sum_{i=1}^{n} d(p_i, \{c_1, \ldots, c_k\})^q$$

for some q

Center based clustering

Typically, we want to choose $c_1, \ldots, c_k$ to minimize the clustering objective

$$\sum_{i=1}^{n} d(p_i, \{c_1, \ldots, c_k\})^q$$

for some q

- k-**center**: $q = \infty$; equivalently, $\min_{c_1,\ldots,c_k \in V} \max_{i=1,\ldots,n} d(p_i, \{c_1, \ldots, c_k\})$
- k-**median**: $q = 1$; equivalently, $\min_{c_1,\ldots,c_k \in V} \sum_{i=1}^{n} d(p_i, \{c_1, \ldots, c_k\})$
- k-**means**: $q = 2$; equivalently, $\min_{c_1,\ldots,c_k \in V} \sum_{i=1}^{n} d(p_i, \{c_1, \ldots, c_k\})^2$

Center based clustering

Typically, we want to choose $c_1, \ldots, c_k$ to minimize the clustering objective

$$\sum_{i=1}^{n} d(p_i, \{c_1, \ldots, c_k\})^q$$

for some q

- k-**center**: $q = \infty$; equivalently, $\min_{c_1,\ldots,c_k \in V} \max_{i=1,\ldots,n} d(p_i, \{c_1, \ldots, c_k\})$
- k-**median**: $q = 1$; equivalently, $\min_{c_1,\ldots,c_k \in V} \sum_{i=1}^{n} d(p_i, \{c_1, \ldots, c_k\})$
- k-**means**: $q = 2$; equivalently, $\min_{c_1,\ldots,c_k \in V} \sum_{i=1}^{n} d(p_i, \{c_1, \ldots, c_k\})^2$

- all problems are NP-hard
- for k-center, k-median, we focus on "discrete version", i.e., the centers are from P

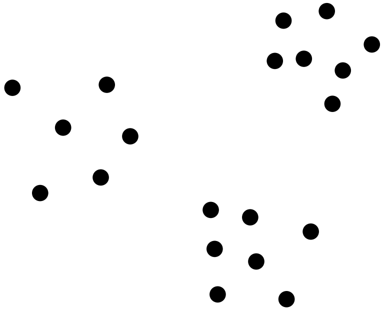
k-center

k-center problem

Input: a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (P, d) and an integer k.

Objective: find a set C with $|C| = k$ such that the cost of C, i.e., $\mathrm{cost}(C) := \max_{i=1,\ldots,n} d(p_i, C)$, is minimized.

(**) every vertex (or point) in a cluster is in distance at most $\mathrm{cost}(C)$ from its respective center.

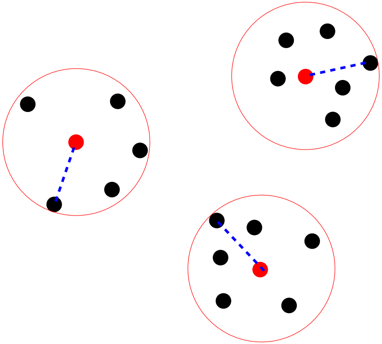


k-center problem

Input: a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (P, d) and an integer k.

Objective: find a set C with $|C| = k$ such that the cost of C, i.e., $\mathrm{cost}(C) := \max_{i=1,\ldots,n} d(p_i, C)$, is minimized.

(**) every vertex (or point) in a cluster is in distance at most $\mathrm{cost}(C)$ from its respective center.



An approximation algorithm

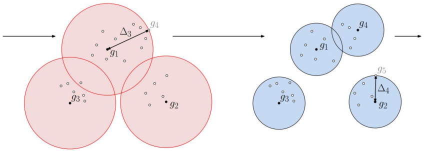
Idea: start with an empty set of centers, and in each iteration pick a new center that is the farthest point from the set of centers chosen so far

An approximation algorithm

Idea: start with an empty set of centers, and in each iteration pick a new center that is the farthest point from the set of centers chosen so far

A greedy algorithm GreedykCenter

1. $S \leftarrow \emptyset$
2. for each $u \in P$, set $d[u] = \infty$. (initialize distances)
3. for $i = 1$ to k
   3.1 let $g_i \in P$ such that $d[g_i]$ is maximum
   3.2 add g_i to S
   3.3 for every point j in P, update $d[j] = \min\{d[j], d(j, g_i)\}$
   3.4 $\Delta = \max_{j \in P} d[j]$
4. return (S, Δ)



Analysis

Running time

1. The i-th iteration of choosing the i-th center takes $O(n)$ time
2. There are k such iterations.
3. The overall algorithm takes $O(nk)$ time.

Theorem 1: The algorithm $\textsc{GreedykCenter}$ is 2-approximation algorithm for the k-center problem

Analysis

Running time

1. The i-th iteration of choosing the i-th center takes $O(n)$ time
2. There are k such iterations.
3. The overall algorithm takes $O(nk)$ time.

Theorem 1: The algorithm $\textsc{GreedykCenter}$ is 2-approximation algorithm for the k-center problem

The proof makes use of the following result.

Lemma 1 Suppose $\exists\ k+1$ points $q_1, \ldots, q_{k+1} \in P$ such that $d(q_i, q_j) > 2R$ for all $i \neq j$. Then $\mathrm{OPT} > R$, where OPT is the cost of optimal solution.

Proof of Lemma 1

- Assume that $\mathrm{OPT} \leq R$.

Proof of Lemma 1

- Assume that $\mathrm{OPT} \leq R$.
- Then $\exists\ C = \{c_1, \ldots, c_k\}$ that induces k clusters $C_1, \ldots, C_k$ such that

Proof of Lemma 1

- Assume that $\mathrm{OPT} \leq R$.
- Then $\exists\ C = \{c_1, \ldots, c_k\}$ that induces k clusters $C_1, \ldots, C_k$ such that
  - for each C_h and $p \in C_h$, $d(p, c_h) \leq R$

Proof of Lemma 1

- Assume that $\mathrm{OPT} \leq R$.
- Then $\exists\ C = \{c_1, \ldots, c_k\}$ that induces k clusters $C_1, \ldots, C_k$ such that
  - for each C_h and $p \in C_h$, $d(p, c_h) \leq R$
- By the pigeon hole principle, some q_i, q_j $(i \neq j)$ are in the same cluster C_h

Proof of Lemma 1

- Assume that $\mathrm{OPT} \leq R$.
- Then $\exists\ C = \{c_1, \ldots, c_k\}$ that induces k clusters $C_1, \ldots, C_k$ such that
  - for each C_h and $p \in C_h$, $d(p, c_h) \leq R$
- By the pigeon hole principle, some q_i, q_j $(i \neq j)$ are in the same cluster C_h
- $\Longrightarrow d(q_i, q_j) \leq d(q_i, c_h) + d(q_j, c_h) \leq 2R$, which contradicts the assumption of the lemma.

Proof of Theorem 1

- Let $\Delta, S = \{g_1, \ldots, g_k\}$ be the output of $\textsc{GreedykCenter}$

Proof of Theorem 1

- Let $\Delta, S = \{g_1, \ldots, g_k\}$ be the output of $\textsc{GreedykCenter}$
- Note that $\Delta = \max_{p \in P} d(p, S)$

Proof of Theorem 1

- Let $\Delta, S = \{g_1, \ldots, g_k\}$ be the output of $\textsc{GreedykCenter}$
- Note that $\Delta = \max_{p \in P} d(p, S)$
- Suppose that $\Delta > 2 \cdot \mathrm{OPT}$, where OPT is the cost of optimal solution.

Proof of Theorem 1

- Let $\Delta, S = \{g_1, \ldots, g_k\}$ be the output of $\textsc{GreedykCenter}$
- Note that $\Delta = \max_{p \in P} d(p, S)$
- Suppose that $\Delta > 2 \cdot \mathrm{OPT}$, where OPT is the cost of optimal solution.
- Then $\exists p \in P$ such that $d(p, S) > 2\mathrm{OPT}$; thus, for any i, $d(g_i, p) \geq 2\mathrm{OPT}$; this also implies $p \notin S$

Proof of Theorem 1

- Let $\Delta, S = \{g_1, \ldots, g_k\}$ be the output of $\textsc{GreedykCenter}$
- Note that $\Delta = \max_{p \in P} d(p, S)$
- Suppose that $\Delta > 2 \cdot \mathrm{OPT}$, where OPT is the cost of optimal solution.
- Then $\exists p \in P$ such that $d(p, S) > 2\mathrm{OPT}$; thus, for any i, $d(g_i, p) \geq 2\mathrm{OPT}$; this also implies $p \notin S$
- Since $\textsc{GreedykCenter}$ chose the farthest point in each iteration, and did not choose p in each of the k iterations,

$$d(g_i, \{g_1, \ldots, g_{i-1}\}) > 2\mathrm{OPT} \text{ for each } i = 2, \ldots, k$$

Proof of Theorem 1

- Let $\Delta, S = \{g_1, \ldots, g_k\}$ be the output of $\textsc{GreedykCenter}$
- Note that $\Delta = \max_{p \in P} d(p, S)$
- Suppose that $\Delta > 2 \cdot \mathrm{OPT}$, where OPT is the cost of optimal solution.
- Then $\exists p \in P$ such that $d(p, S) > 2\mathrm{OPT}$; thus, for any i, $d(g_i, p) \geq 2\mathrm{OPT}$; this also implies $p \notin S$
- Since $\textsc{GreedykCenter}$ chose the farthest point in each iteration, and did not choose p in each of the k iterations,

$$d(g_i, \{g_1, \ldots, g_{i-1}\}) > 2\mathrm{OPT} \text{ for each } i = 2, \ldots, k$$

- $\implies$ the distance between each pair of points in $\{g_1, \ldots, g_k, p\}$ is $> 2\mathrm{OPT}$.

Proof of Theorem 1

- Let $\Delta, S = \{g_1, \ldots, g_k\}$ be the output of $\textsc{GreedykCenter}$
- Note that $\Delta = \max_{p \in P} d(p, S)$
- Suppose that $\Delta > 2 \cdot \mathrm{OPT}$, where OPT is the cost of optimal solution.
- Then $\exists p \in P$ such that $d(p, S) > 2\mathrm{OPT}$; thus, for any i, $d(g_i, p) \geq 2\mathrm{OPT}$; this also implies $p \notin S$
- Since $\textsc{GreedykCenter}$ chose the farthest point in each iteration, and did not choose p in each of the k iterations,

$$d(g_i, \{g_1, \ldots, g_{i-1}\}) > 2\mathrm{OPT} \text{ for each } i = 2, \ldots, k$$

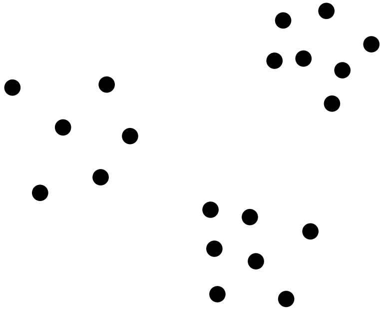
- $\Longrightarrow$ the distance between each pair of points in $\{g_1, \ldots, g_k, p\}$ is $> 2\mathrm{OPT}$.
- By Lemma 1, the cost of optimal solution is $> \mathrm{OPT}$, a contradiction.

k-median

k-median problem

Input: a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (P, d) and an integer k.

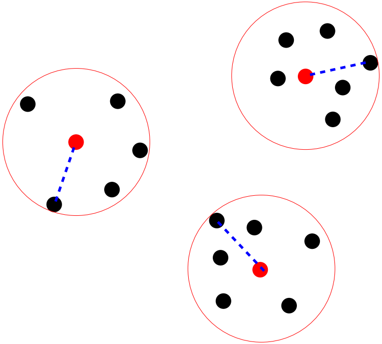
Objective: find a set C with $|C| = k$ such that the cost of C, i.e., $\mathrm{cost}(C) := \sum_{i=1}^{n} d(p_i, C)$, is minimized.



k-median problem

Input: a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (P, d) and an integer k.

Objective: find a set C with $|C| = k$ such that the cost of C, i.e., $\mathrm{cost}(C) := \sum_{i=1}^{n} d(p_i, C)$, is minimized.



Some facts

- one can obtain a constant factor approximation algorithm via LP rounding,
- in the following, we present a constant factor approximation algorithm based on local search

The algorithm

LS-KMedian

1. start with any $S \subseteq P$ with $|S| = k$
2. while there is any pair $a \in P \setminus S, r \in S$ such that $\mathrm{cost}(S - r + a) < \mathrm{cost}(S)$ do
   update $S \leftarrow S - r + a$
3. output S

The algorithm

LS-KMEDIAN

1. start with any $S \subseteq P$ with $|S| = k$
2. while there is any pair $a \in P \setminus S, r \in S$ such that $\mathrm{cost}(S - r + a) < \mathrm{cost}(S)$ do
   update $S \leftarrow S - r + a$
3. output S

note:

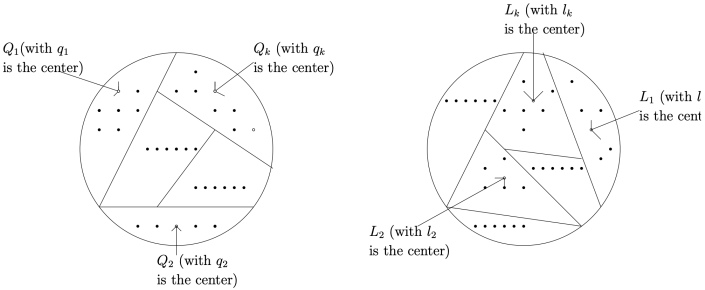
- We use $S - r + a$ as an abbreviation for $(S \setminus \{r\}) \cup \{a\}$.
- i.e., we are swapping r and a

Analysis of LS-KMEDIAN: Notations

- let $L = \{\ell_1, \ell_2, \ldots, \ell_k\}$ be the algorithm's output

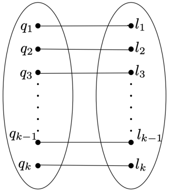
Analysis of LS-KMEDIAN: Notations

- let $L = \{\ell_1, \ell_2, \ldots, \ell_k\}$ be the algorithm's output
- let $Q = \{q_1, q_2, \ldots, q_k\}$ be the optimum solution



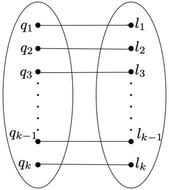
Analysis of LS-KMEDIAN: Notations

- let $\pi : Q \to L$ be the map such that $\pi(q_i) =$ closest point in L to q_i.



Analysis of LS-KMEDIAN: Notations

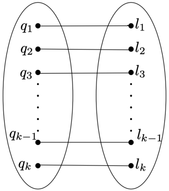
- let $\pi : Q \to L$ be the map such that $\pi(q_i) =$ closest point in L to q_i.



- $d_j^{(L)}$: distance between any point $j \in P$ to its closest center in L

Analysis of LS-KMEDIAN: Notations

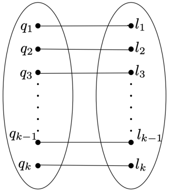
- let $\pi : Q \to L$ be the map such that $\pi(q_i) =$ closest point in L to q_i.



- $d_j^{(L)}$: distance between any point $j \in P$ to its closest center in L
- $d_j^{(Q)}$: distance between point $j \in P$ to its closest center in Q

Analysis of LS-KMEDIAN: Notations

- let $\pi : Q \to L$ be the map such that $\pi(q_i) =$ closest point in L to q_i.

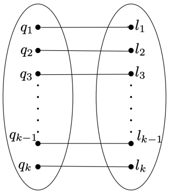


- $d_j^{(L)}$: distance between any point $j \in P$ to its closest center in L
- $d_j^{(Q)}$: distance between point $j \in P$ to its closest center in Q

- L_i : the set of all points assigned to center ℓ_i in solution L

Analysis of LS-KMEDIAN: Notations

- let $\pi : Q \to L$ be the map such that $\pi(q_i) =$ closest point in L to q_i.

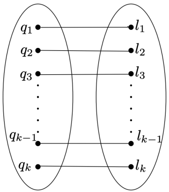


- $d_j^{(L)}$: distance between any point $j \in P$ to its closest center in L
- $d_j^{(Q)}$: distance between point $j \in P$ to its closest center in Q

- L_i : the set of all points assigned to center ℓ_i in solution L
- Q_i : the set of all points assigned to center q_i in solution Q

Analysis of LS-KMEDIAN: Notations

- let $\pi : Q \to L$ be the map such that $\pi(q_i)$ = closest point in L to q_i.



- $d_j^{(L)}$: distance between any point $j \in P$ to its closest center in L
- $d_j^{(Q)}$: distance between point $j \in P$ to its closest center in Q

- L_i : the set of all points assigned to center ℓ_i in solution L
- Q_i : the set of all points assigned to center q_i in solution Q
- note

$$\mathrm{OPT} = \mathrm{cost}(Q) = \sum_{j \in P} d_j^{(Q)}, \mathrm{cost}(L) = \sum_{j \in P} d_j^{(L)}$$

Analysis of LS-KMEDIAN: when π is a bijection

We first prove a special case when π is a bijection.

Theorem 2:

If π is a bijection, the algorithm LS-KMEDIAN outputs a set L of centers such that $\mathrm{cost}(L) \leq 3 \cdot \mathrm{OPT}$.

Proof ideas:

- w.l.o.g., let $\pi(q_i) = \ell_i$
- **Lemma 2:**

  for any $i \in \{1, \ldots, k\}$, it holds that

  $$0 \leq \mathrm{cost}(L + q_i - \ell_i) - \mathrm{cost}(L) \leq \sum_{j \in Q_i} (d_j^{(Q)} - d_j^{(L)}) + 2 \sum_{j \in L_i} d_j^{(Q)}$$

Proof of Theorem 2 based on Lemma 2

Proof of Theorem 2:

- By Lemma 2,

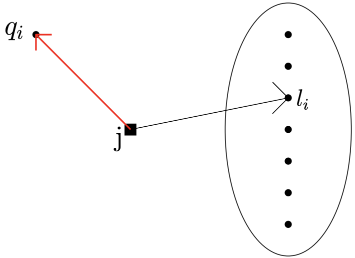
$$\begin{aligned}
0 &\leq \sum_{i=1,\dots,k} (\mathrm{cost}(L + q_i - \ell_i) - \mathrm{cost}(L)) \\
&\leq \sum_{i=1,\dots,k} \Big(\sum_{j \in Q_i} (d_j^{(Q)} - d_j^{(L)}) + 2 \sum_{j \in L_i} d_j^{(Q)} \Big) \\
&= \sum_{i=1,\dots,k} \sum_{j \in Q_i} d_j^{(Q)} - \sum_{i=1,\dots,k} \sum_{j \in Q_i} d_j^{(L)} + 2 \sum_{i=1,\dots,k} \sum_{j \in L_i} d_j^{(Q)} \\
&= \mathrm{OPT} - \mathrm{cost}(L) + 2\mathrm{OPT}
\end{aligned}$$

- Thus, $\mathrm{cost}(L) \leq 3 \cdot \mathrm{OPT}$

Proof of Lemma 2

Proof

- Let $L' = L + q_i - \ell_i$. Then $\mathrm{cost}(L') = \sum_{j \in P} d(j, L')$
- We need to consider, for any $j \in P$, how the distance between j and its center changes when the center set is updated from L to L'.
- **Case 1:** $j \in Q_i$. We connect j to q_i in L', i.e., j switches its center from ℓ_i to q_i. Thus, $d(j, L') \leq d_j^{(Q)}$

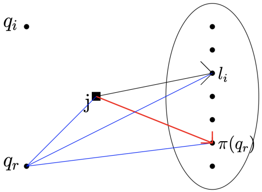


Proof of Lemma 2 - cont.

Proof

- **Case 2:** $j \in L_i \setminus Q_i$. Then j is assigned to $\pi(q_r)$ where q_r represents the center that j is assigned to in Q. Note $\pi(q_r) \neq \ell_i$ and so $\pi(q_r) \in L'$. Thus

$$\begin{aligned} d(j,\pi(q_r)) &\leq d(j,q_r) + d(q_r,\pi(q_r)) = d_j^{(Q)} + d(q_r,\pi(q_r)) \\ &\leq d_j^{(Q)} + d(q_r,\ell_i) \qquad \text{(by definition of } \pi) \\ &\leq d_j^{(Q)} + d(q_r,j) + d(j,\ell_i) \\ &= d_j^{(Q)} + d_j^{(Q)} + d_j^{(L)} \end{aligned}$$



Thus, $d(j,L') \leq d(j,\pi(q_r)) \leq 2d_j^{(Q)} + d_j^{(L)}$

Proof of Lemma 2 - cont.

Proof

- **Case 3:** $j \in V \setminus (L_i \cup Q_i)$, it does not change its center, so $d(j, L') \leq d_j^{(L)}$
- Since $-\mathrm{cost}(L) = -\sum_{j \in P} d_j^{(L)}$, combining the above cases,

$$\mathrm{cost}(L + q_i - \ell_i) - \mathrm{cost}(L) \leq \sum_{j \in Q_i} d_j^{(Q)} - \sum_{j \in Q_i} d_j^{(L)} + 2 \sum_{j \in L_i} d_j^{(Q)}$$

Analysis of LS-KMEDIAN: general π

We have the following theorem for the case that π is a general map.

Theorem 3:

If π is a general map (not necessarily a bijection), the algorithm LS-KMEDIAN outputs a set L of centers such that $\mathrm{cost}(L) \leq 5 \cdot \mathrm{OPT}$.

Proof ideas:

- need to define a set of candidate swaps
- prove that for any pair of candidate swaps, some property similar to Lemma 2 holds.
- details are omitted here.

Analysis of LS-KMEDIAN: running time

- The running time of the naive implementation may depend on $\text{poly}(D)$, where D is the maximum distance, assuming all integer distances
- Not polynomial time in the input size (i.e., $n, \log D$)!
- We ensure that each swap satisfies that

$$\text{cost}(S') - \text{cost}(S) < -\varepsilon \text{cost}(S),$$

  where S' is the set after swap
- Then if we start with solution S_0, and with solution S_k after k swaps, then

$$1 \leq \text{cost}(S_k) \leq (1-\varepsilon)^k \text{cost}(S_0) \leq (1-\varepsilon)^k \cdot nD$$

- Then the number of iterations is $O(\frac{1}{\varepsilon}\log(nD))$
- using this one can set $\varepsilon = 1/n^3$, and show that

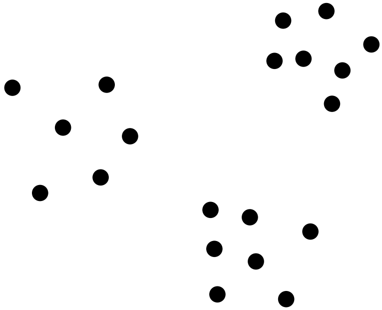
  Theorem 4: The algorithm LS-KMEDIAN is a $5 + o(1)$ approximation algorithm.

k-means

k-means problem

Input: a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (P, d) and an integer k.

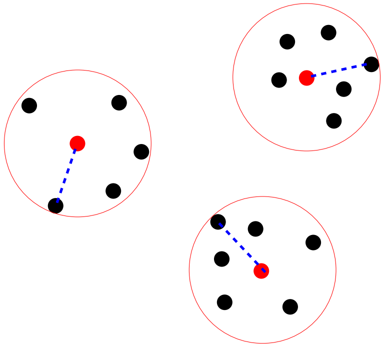
Objective: find a set C with $|C| = k$ such that the cost of C, i.e., $\mathrm{cost}(C) := \sum_{i=1}^{n} d(p_i, C)^2$, is minimized.



k-means problem

Input: a set of n points $P = \{p_1, p_2, \ldots, p_n\}$ in a metric space (P, d) and an integer k.

Objective: find a set C with $|C| = k$ such that the cost of C, i.e., $\mathrm{cost}(C) := \sum_{i=1}^{n} d(p_i, C)^2$, is minimized.



Some facts

- the discrete setting (i.e., all the centers are from P): one can obtain constant factor approximation via local search (similar to the approach for k-median)

Some facts

- the discrete setting (i.e., all the centers are from P): one can obtain constant factor approximation via local search (similar to the approach for k-median)
- Note: the squared distance does NOT satisfy triangle inequality, while it satisfies a relaxed triangle inequality, so that we can apply local search technique

Some facts

- the discrete setting (i.e., all the centers are from P): one can obtain constant factor approximation via local search (similar to the approach for k-median)
- Note: the squared distance does NOT satisfy triangle inequality, while it satisfies a relaxed triangle inequality, so that we can apply local search technique

- in the following, we consider the continuous version of the Euclidean k-means:

Some facts

- the discrete setting (i.e., all the centers are from P): one can obtain constant factor approximation via local search (similar to the approach for k-median)
- Note: the squared distance does NOT satisfy triangle inequality, while it satisfies a relaxed triangle inequality, so that we can apply local search technique

- in the following, we consider the continuous version of the Euclidean k-means:
  - $P = \{p_1, \ldots, p_n\}$, a set of points in $\mathbb{R}^d$

Some facts

- the discrete setting (i.e., all the centers are from P): one can obtain constant factor approximation via local search (similar to the approach for k-median)
- Note: the squared distance does NOT satisfy triangle inequality, while it satisfies a relaxed triangle inequality, so that we can apply local search technique

- in the following, we consider the continuous version of the Euclidean k-means:
  - $P = \{p_1, \ldots, p_n\}$, a set of points in $\mathbb{R}^d$
  - $d(x, y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2}$

Some facts

- the discrete setting (i.e., all the centers are from P): one can obtain constant factor approximation via local search (similar to the approach for k-median)
- Note: the squared distance does NOT satisfy triangle inequality, while it satisfies a relaxed triangle inequality, so that we can apply local search technique

- in the following, we consider the continuous version of the Euclidean k-means:
  - $P = \{p_1, \ldots, p_n\}$, a set of points in $\mathbb{R}^d$
  - $d(x, y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2}$
  - the centers are now allowed to be in $\mathbb{R}^d$, not necessarily in P

Some facts

- the discrete setting (i.e., all the centers are from P): one can obtain constant factor approximation via local search (similar to the approach for k-median)
- Note: the squared distance does NOT satisfy triangle inequality, while it satisfies a relaxed triangle inequality, so that we can apply local search technique
- in the following, we consider the continuous version of the Euclidean k-means:
  - $P = \{p_1, \ldots, p_n\}$, a set of points in $\mathbb{R}^d$
  - $d(x, y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2}$
  - the centers are now allowed to be in $\mathbb{R}^d$, not necessarily in P
- still NP-hard, even for $d = 2$

Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily

Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat

Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat
    2.1 for $i = 1, \ldots, k$

Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat
   2.1 for $i = 1, \ldots, k$
      2.1.1 $C_i \leftarrow$ set of the points in P closest to c_i (called the assignment step)

Lloyd's algorithm

Algorithm $\textsc{LloydKMeans}$:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat
   2.1 for $i = 1, \ldots, k$
      2.1.1 $C_i \leftarrow$ set of the points in P closest to c_i (called the assignment step)
   2.2 for $i = 1, \ldots, k$

Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat
   - 2.1 for $i = 1, \ldots, k$
     - 2.1.1 $C_i \leftarrow$ set of the points in P closest to c_i (called the assignment step)
   - 2.2 for $i = 1, \ldots, k$
     - 2.2.1 $c_i \leftarrow \mathrm{ct}(C_i) = \frac{1}{|C_i|} \sum_{p \in C_i} p$ (called the update step)

Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat
   2.1 for $i = 1, \ldots, k$
      2.1.1 $C_i \leftarrow$ set of the points in P closest to c_i (called the assignment step)
   2.2 for $i = 1, \ldots, k$
      2.2.1 $c_i \leftarrow \mathrm{ct}(C_i) = \frac{1}{|C_i|} \sum_{p \in C_i} p$ (called the update step)
3. until convergence (e.g., when quality of solution no longer improves)

Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat
   2.1 for $i = 1, \ldots, k$
      2.1.1 $C_i \leftarrow$ set of the points in P closest to c_i (called the assignment step)
   2.2 for $i = 1, \ldots, k$
      2.2.1 $c_i \leftarrow \mathrm{ct}(C_i) = \frac{1}{|C_i|} \sum_{p \in C_i} p$ (called the update step)
3. until convergence (e.g., when quality of solution no longer improves)
4. return $c_1, \ldots, c_k$ and $C_1, \ldots, C_k$

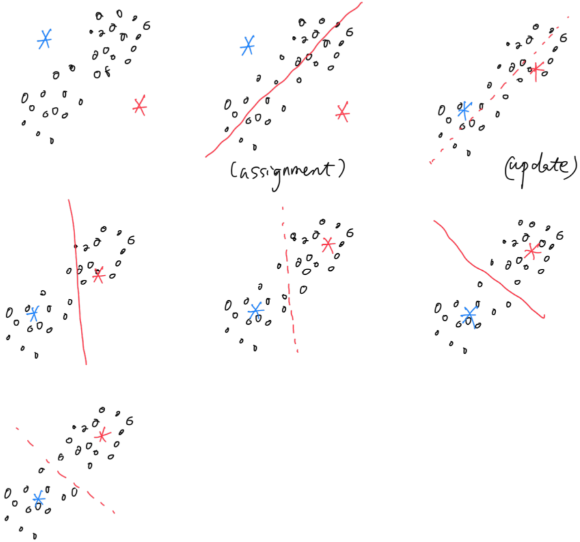
Lloyd's algorithm

Algorithm LloydKMeans:

1. pick k centers $c_1, \ldots, c_k$ arbitrarily
2. repeat
   2.1 for $i = 1, \ldots, k$
      2.1.1 $C_i \leftarrow$ set of the points in P closest to c_i (called the assignment step)
   2.2 for $i = 1, \ldots, k$
      2.2.1 $c_i \leftarrow \mathrm{ct}(C_i) = \frac{1}{|C_i|} \sum_{p \in C_i} p$ (called the update step)
3. until convergence (e.g., when quality of solution no longer improves)
4. return $c_1, \ldots, c_k$ and $C_1, \ldots, C_k$

Remark: Lloyd's algorithm is also known as k-means algorithm.

An illustration



Analysis of LloydKMeans: some properties

Lemma 3: Given a set $P = \{p_1, \ldots, p_n\}$, let $\mathrm{ct}(P) = \frac{1}{n}\sum_{i=1}^{n} p_i$ be the centroid of P. Then for any $x \in \mathbb{R}^d$,

$$\sum_i \|p_i - x\|^2 = \sum_i \|p_i - \mathrm{ct}(P)\|^2 + n \cdot \|\mathrm{ct}(P) - x\|^2$$

Analysis of LLOYDKMEANS: some properties

Lemma 3: Given a set $P = \{p_1, \ldots, p_n\}$, let $\mathrm{ct}(P) = \frac{1}{n}\sum_{i=1}^{n} p_i$ be the centroid of P. Then for any $x \in \mathbb{R}^d$,

$$\sum_i \|p_i - x\|^2 = \sum_i \|p_i - \mathrm{ct}(P)\|^2 + n \cdot \|\mathrm{ct}(P) - x\|^2$$

Proof:

- $$\begin{aligned}\sum_i \|p_i - x\|^2 &= \sum_i \|p_i - \mathrm{ct}(P) + \mathrm{ct}(P) - x\|^2 \\ &= \sum_i \|p_i - \mathrm{ct}(P)\|^2 + 2(\mathrm{ct}(P) - x) \cdot \sum_i (p_i - \mathrm{ct}(P)) \\ &+ n \cdot \|\mathrm{ct}(P) - x\|^2\end{aligned}$$

Analysis of LloydKMeans: some properties

Lemma 3: Given a set $P = \{p_1, \ldots, p_n\}$, let $\mathrm{ct}(P) = \frac{1}{n}\sum_{i=1}^{n} p_i$ be the centroid of P. Then for any $x \in \mathbb{R}^d$,

$$\sum_i \|p_i - x\|^2 = \sum_i \|p_i - \mathrm{ct}(P)\|^2 + n \cdot \|\mathrm{ct}(P) - x\|^2$$

Proof:

- $$\begin{aligned} \sum_i \|p_i - x\|^2 &= \sum_i \|p_i - \mathrm{ct}(P) + \mathrm{ct}(P) - x\|^2 \\ &= \sum_i \|p_i - \mathrm{ct}(P)\|^2 + 2(\mathrm{ct}(P) - x) \cdot \sum_i (p_i - \mathrm{ct}(P)) \\ &\quad + n \cdot \|\mathrm{ct}(P) - x\|^2 \end{aligned}$$
- by definition of $\mathrm{ct}(P)$, $\sum_i (\mathrm{ct}(P) - p_i) = 0$

Analysis of $\textsc{LloydKMeans}$: some properties

Lemma 3: Given a set $P = \{p_1, \dots, p_n\}$, let $\mathrm{ct}(P) = \frac{1}{n}\sum_{i=1}^{n} p_i$ be the centroid of P. Then for any $x \in \mathbb{R}^d$,

$$\sum_i \|p_i - x\|^2 = \sum_i \|p_i - \mathrm{ct}(P)\|^2 + n \cdot \|\mathrm{ct}(P) - x\|^2$$

Proof:

- $$\begin{aligned}\sum_i \|p_i - x\|^2 &= \sum_i \|p_i - \mathrm{ct}(P) + \mathrm{ct}(P) - x\|^2 \\ &= \sum_i \|p_i - \mathrm{ct}(P)\|^2 + 2(\mathrm{ct}(P) - x) \cdot \sum_i (p_i - \mathrm{ct}(P)) \\ &+ n \cdot \|\mathrm{ct}(P) - x\|^2\end{aligned}$$
- by definition of $\mathrm{ct}(P)$, $\sum_i (\mathrm{ct}(P) - p_i) = 0$
- thus, $\sum_i \|p_i - x\|^2 = \sum_i \|p_i - \mathrm{ct}(P)\|^2 + n \cdot \|\mathrm{ct}(P) - x\|^2$

Analysis of LloydKMeans: some properties

Lemma 3: Given a set $P = \{p_1, \ldots, p_n\}$, let $\mathrm{ct}(P) = \frac{1}{n}\sum_{i=1}^{n} p_i$ be the centroid of P. Then for any $x \in \mathbb{R}^d$,

$$\sum_i \|p_i - x\|^2 = \sum_i \|p_i - \mathrm{ct}(P)\|^2 + n \cdot \|\mathrm{ct}(P) - x\|^2$$

Analysis of LloydKMeans: some properties

Lemma 3: Given a set $P = \{p_1, \ldots, p_n\}$, let $\mathrm{ct}(P) = \frac{1}{n}\sum_{i=1}^{n} p_i$ be the centroid of P. Then for any $x \in \mathbb{R}^d$,

$$\sum_i \|p_i - x\|^2 = \sum_i \|p_i - \mathrm{ct}(P)\|^2 + n \cdot \|\mathrm{ct}(P) - x\|^2$$

Corollary 1: Let $P = \{p_1, \ldots, p_n\}$ be a set of points. The sum of squared distances of p_i's to a point x is minimized when x is the centroid, i.e., $x = \mathrm{ct}(P) = \frac{1}{n}\sum_i p_i$.

Analysis of LLOYDKMEANS: some properties

Lemma 4: The algorithm LLOYDKMEANS always halt after a finite number of steps.

Proof:

- Let $T =$ the sum of squares of distances of each point to its cluster center

Analysis of LloydKMeans: some properties

$\boxed{\text{Lemma 4: The algorithm LloydKMeans always halt after a finite number of steps.}}$

Proof:

- Let $T =$ the sum of squares of distances of each point to its cluster center
- we prove that T always improves until convergence

Analysis of LloydKMeans: some properties

Lemma 4: The algorithm LloydKMeans always halt after a finite number of steps.

Proof:

- Let $T =$ the sum of squares of distances of each point to its cluster center
- we prove that T always improves until convergence

1) after update step: by Lemma 3, for each cluster, the sum of squares of distances of each point to its center improves, i.e., T becomes smaller

Analysis of LloydKMeans: some properties

Lemma 4: The algorithm LloydKMeans always halt after a finite number of steps.

Proof:

- Let $T =$ the sum of squares of distances of each point to its cluster center
- we prove that T always improves until convergence

1) after update step: by Lemma 3, for each cluster, the sum of squares of distances of each point to its center improves, i.e., T becomes smaller
2) after assignment step: T also improves, as for each point p the distance between p and its center is improving (each point is assigned to a center that is closest to it)

Analysis of LloydKMeans: some properties

Lemma 4: The algorithm LloydKMeans always halt after a finite number of steps.

Proof:

- Let $T =$ the sum of squares of distances of each point to its cluster center
- we prove that T always improves until convergence

1) after update step: by Lemma 3, for each cluster, the sum of squares of distances of each point to its center improves, i.e., T becomes smaller
2) after assignment step: T also improves, as for each point p the distance between p and its center is improving (each point is assigned to a center that is closest to it)

- The running time $O(n \cdot k \cdot d \cdot R)$, where R is the number of assignment and update steps until convergence

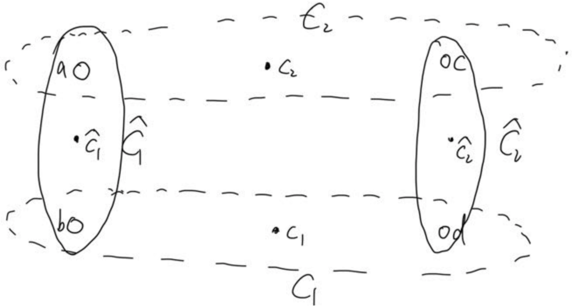
Issues of LloydKMeans: exponential number of iterations

Regarding R, the number of iterations, of LloydKMeans:

- good in practice
- worst-case in theory $R = 2^{\Omega(\sqrt{n})}$, superpolynomial

Issues of LloydKMeans: stuck in a local optimum

Note: Algorithm k-means can get stuck in arbitrarily poor local minima



- Lloyd's algorithm C_1, C_2
- optimal clusters: $\hat{C}_1, \hat{C}_2$

We can solve these issues by appropriately sampling the initial centers.

D^2-sampling

Algorithm D^2-SAMPLING

1. let $S = \{c_1\}$, where c_1 is a point sampled uniformly at random from P
2. for $i = 2, \ldots, k$ do
   2.1 choose c_i randomly where $\Pr[c_i = p] \propto d(p, S)^2$ (i.e., with probability proportional to $d(p, S)^2$)
   2.2 update $S \leftarrow S \cup \{c_i\}$
3. output S

D^2-sampling

Algorithm D^2-SAMPLING

1. let $S = \{c_1\}$, where c_1 is a point sampled uniformly at random from P
2. for $i = 2, \ldots, k$ do
   2.1 choose c_i randomly where $\Pr[c_i = p] \propto d(p, S)^2$ (i.e., with probability proportional to $d(p, S)^2$)
   2.2 update $S \leftarrow S \cup \{c_i\}$
3. output S

Recall that $\text{cost}(C) = \sum_{i=1}^{n} d(p_i, C)^2$, and OPT is the cost of the optimal solution.

Theorem 5: Let S be the output of D^2-SAMPLING. Then

$$\text{E}[\text{cost}(S)] \leq 8(\ln k + 2)\text{OPT}.$$

Proof ideas of Theorem 5

Given two sets of points A, C, let $D(A, C) = \sum_{a \in A} \min_{c \in C} d(a, c)^2$.

Proof ideas of Theorem 5

Given two sets of points A, C, let $D(A, C) = \sum_{a \in A} \min_{c \in C} d(a, c)^2$.

Lemma 5: Let $Q \subseteq \mathbb{R}^d$ be a point set and $c \in Q$ be a point chosen uniformly at random from Q, then

$$\mathrm{E}[D(Q, c)] = 2 \cdot D(Q, \mathrm{ct}(Q)),$$

where $\mathrm{ct}(Q)$ is the centroid of Q, i.e., $\mathrm{ct}(Q) = \frac{1}{|Q|} \sum_{p \in Q} p$.

Proof ideas of Theorem 5

Given two sets of points A, C, let $D(A,C) = \sum_{a \in A} \min_{c \in C} d(a,c)^2$.

Lemma 5: Let $Q \subseteq \mathbb{R}^d$ be a point set and $c \in Q$ be a point chosen uniformly at random from Q, then

$$\mathrm{E}[D(Q,c)] = 2 \cdot D(Q, \mathrm{ct}(Q)),$$

where $\mathrm{ct}(Q)$ is the centroid of Q, i.e., $\mathrm{ct}(Q) = \frac{1}{|Q|} \sum_{p \in Q} p$.

Proof:

$$\mathrm{E}[D(Q,c)] = \sum_{p \in Q} \frac{1}{|Q|} D(Q,p) = \frac{1}{|Q|} \sum_{p \in Q} \sum_{p' \in Q} d(p,p')^2$$

$$= \frac{1}{|Q|} \sum_{p \in Q} \left[\sum_{p' \in Q} d(p', \mathrm{ct}(Q))^2 + |Q| \cdot d(p, \mathrm{ct}(Q))^2 \right]$$

$$= \sum_{p \in Q} d(p, \mathrm{ct}(Q))^2 + \sum_{p' \in Q} d(p', \mathrm{ct}(Q))^2 = 2 \cdot D(Q, \mathrm{ct}(Q))$$

Proof ideas of Theorem 5

Given two sets of points A, C, let $D(A, C) = \sum_{a \in A} \min_{c \in C} d(a, c)^2$.

Lemma 5: Let $Q \subseteq \mathbb{R}^d$ be a point set and $c \in Q$ be a point chosen uniformly at random from Q, then

$$\mathrm{E}[D(Q, c)] = 2 \cdot D(Q, \mathrm{ct}(Q)),$$

where $\mathrm{ct}(Q)$ is the centroid of Q, i.e., $\mathrm{ct}(Q) = \frac{1}{|Q|} \sum_{p \in Q} p$.

Think of Q as a cluster in the optimal clustering. The above lemma says the first step of D^2-sampling is good.

Proof ideas of Theorem 5

Given two sets of points A, C, let $D(A, C) = \sum_{a \in A} \min_{c \in C} d(a, c)^2$.

Lemma 5: Let $Q \subseteq \mathbb{R}^d$ be a point set and $c \in Q$ be a point chosen uniformly at random from Q, then

$$\mathrm{E}[D(Q, c)] = 2 \cdot D(Q, \mathrm{ct}(Q)),$$

where $\mathrm{ct}(Q)$ is the centroid of Q, i.e., $\mathrm{ct}(Q) = \frac{1}{|Q|} \sum_{p \in Q} p$.

Think of Q as a cluster in the optimal clustering. The above lemma says the first step of D^2-sampling is good.

Corollary 2: Let $C \subseteq \mathbb{R}^d$ be a cluster from the optimal solution $\mathcal{C} = \{C_1, \ldots, C_k\}$. If c is chosen uniformly at random from P, then

$$\mathrm{E}[D(C, c) \mid c \in C] = 2 \cdot D(C, \mathrm{ct}(C))$$

Proof ideas of Theorem 5

Given two sets of points A, C, let $D(A, C) = \sum_{a \in A} \min_{c \in C} d(a, c)^2$.

Lemma 5: Let $Q \subseteq \mathbb{R}^d$ be a point set and $c \in Q$ be a point chosen uniformly at random from Q, then

$$\mathrm{E}[D(Q, c)] = 2 \cdot D(Q, \mathrm{ct}(Q)),$$

where $\mathrm{ct}(Q)$ is the centroid of Q, i.e., $\mathrm{ct}(Q) = \frac{1}{|Q|} \sum_{p \in Q} p$.

Think of Q as a cluster in the optimal clustering. The above lemma says the first step of D^2-sampling is good.

Corollary 2: Let $C \subseteq \mathbb{R}^d$ be a cluster from the optimal solution $\mathcal{C} = \{C_1, \ldots, C_k\}$. If c is chosen uniformly at random from P, then

$$\mathrm{E}[D(C, c) \mid c \in C] = 2 \cdot D(C, \mathrm{ct}(C))$$

Remark: for the k-means clustering and the optimal solution $\mathcal{C}$ defined as above, it holds that $\mathrm{OPT} = \sum_{i=1}^{k} D(C_i, \mathrm{ct}(C_i))$

Proof ideas of Theorem 5 - cont.

Lemma 6: Let $Q \subseteq \mathbb{R}^d$ be a point set and let $S \subseteq \mathbb{R}^d$ be an arbitrary finite set and sample a point $c \in Q$ with probability proportional to $d(c, S)^2$. Then

$$\mathrm{E}[D(Q, S \cup \{c\})] \leq 8 \cdot D(Q, \mathrm{ct}(Q))$$

Proof ideas of Theorem 5 - cont.

Lemma 6: Let $Q \subseteq \mathbb{R}^d$ be a point set and let $S \subseteq \mathbb{R}^d$ be an arbitrary finite set and sample a point $c \in Q$ with probability proportional to $d(c, S)^2$. Then

$$\mathrm{E}[D(Q, S \cup \{c\})] \leq 8 \cdot D(Q, \mathrm{ct}(Q))$$

Corollary 3: Let $C \subseteq \mathbb{R}^d$ be a cluster from the optimal solution $\mathcal{C} = \{C_1, \ldots, C_k\}$. Let S be an arbitrary finite set. If c is sampled at random with probability proportional to $d(c, S)^2$, then

$$\mathrm{E}[D(C, S \cup \{c\}) \mid c \in C] \leq 8 \cdot D(C, \mathrm{ct}(C))$$

Proof ideas of Theorem 5 - cont.

Lemma 6: Let $Q \subseteq \mathbb{R}^d$ be a point set and let $S \subseteq \mathbb{R}^d$ be an arbitrary finite set and sample a point $c \in Q$ with probability proportional to $d(c, S)^2$. Then

$$\mathrm{E}[D(Q, S \cup \{c\})] \leq 8 \cdot D(Q, \mathrm{ct}(Q))$$

Corollary 3: Let $C \subseteq \mathbb{R}^d$ be a cluster from the optimal solution $\mathcal{C} = \{C_1, \ldots, C_k\}$. Let S be an arbitrary finite set. If c is sampled at random with probability proportional to $d(c, S)^2$, then

$$\mathrm{E}[D(C, S \cup \{c\}) \mid c \in C] \leq 8 \cdot D(C, \mathrm{ct}(C))$$

The above says if the sampled center c belongs to a new cluster, then this new cluster is "constant-approximated"

Proof ideas of Theorem 5 - cont.

To finish the proof of Theorem 5, we need to analyze the probability that c is sampled from an "already" covered cluster, and combine the above corollaries to show that D^2-SAMPLING gives an $O(\ln k)$-approximation guarantee.

Proof ideas of Theorem 5 - cont.

To finish the proof of Theorem 5, we need to analyze the probability that c is sampled from an "already" covered cluster, and combine the above corollaries to show that D^2-SAMPLING gives an $O(\ln k)$-approximation guarantee.

(see the paper "k-means++: The advantage of careful seeding" by Arthur and Vassilvitskii, SODA 2007)

k-means++

Algorithm k-$\textsc{Means++}$

1. Run D^2-$\textsc{Sampling}$ on the input P, k to obtain a set S of centers
2. Run $\textsc{LloydKMeans}$ on P, k with initial center set S

k-means++

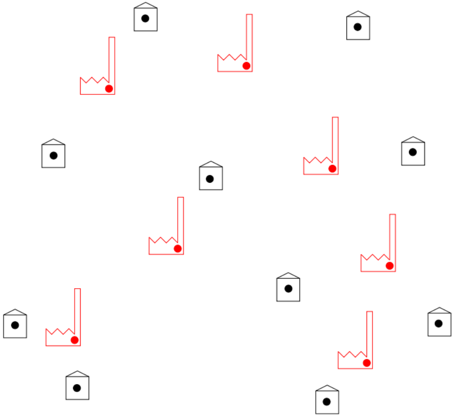
Algorithm k-$\textsc{Means++}$

1. Run D^2-$\textsc{Sampling}$ on the input P, k to obtain a set S of centers
2. Run $\textsc{LloydKMeans}$ on P, k with initial center set S

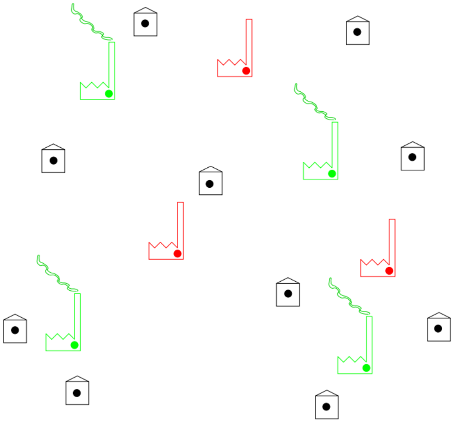
Note: Since the sum of squares of distances of each point to its cluster center always improves, the output of the k-$\textsc{Means++}$ also satisfies the inequality in Theorem 5.

Uncapacitated Facility Location

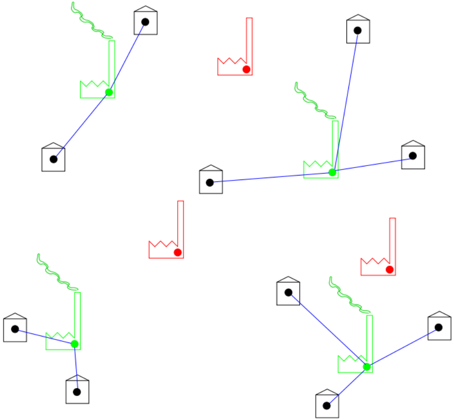
Uncapacitated Facility Location



Uncapacitated Facility Location



Uncapacitated Facility Location



Uncapacitated Facility Location (UFL)

Input:

- a finite set $\mathcal{D}$ of clients
- a finite set $\mathcal{F}$ of potential facilities
- a fixed cost $f_i \in \mathbb{R}_+$ for opening each facility $i \in \mathcal{F}$
- a service cost $c_{ij} \in \mathbb{R}_+$ for each $i \in \mathcal{F}$ and $j \in \mathcal{D}$
- we consider the metric UFL: the facilities and clients are from a metric space, and the costs c_{ij} satisfy 1) $c_{ij} \geq 0$; 2) $c_{ij} + c_{i'j} + c_{i'j'} \geq c_{ij'}$ for all $i, i' \in \mathcal{F}$ and $j, j' \in \mathcal{D}$.

Goal: to find a subset S of facilities (called *open*), and an assignment $\sigma : \mathcal{D} \to S$ of clients to open facilities such that

- the sum of facility costs and service costs

$$\sum_{i \in S} f_i + \sum_{j \in \mathcal{D}} c_{\sigma(j)j}$$

  is minimum

Integer Linear Programming Formulation

$$
\begin{aligned}
\min \quad & \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij} & \\
\text{s.t.} \quad & x_{ij} \le y_i & \forall\, i \in \mathcal{F}, j \in \mathcal{D} \\
& \sum_{i \in \mathcal{F}} x_{ij} = 1 & \forall\, j \in \mathcal{D} \\
& x_{ij} \in \{0, 1\} & \forall\, i \in \mathcal{F}, j \in \mathcal{D} \\
& y_i \in \{0, 1\} & \forall\, i \in \mathcal{F}
\end{aligned}
$$

Linear Programming Relaxation

$$
\begin{aligned}
\min \quad & \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{D}} c_{ij} x_{ij} & \\
\text{s.t.} \quad & x_{ij} \le y_i & \forall\, i \in \mathcal{F}, j \in \mathcal{D} \\
& \sum_{i \in \mathcal{F}} x_{ij} = 1 & \forall\, j \in \mathcal{D} \\
& x_{ij} \ge 0 & \forall\, i \in \mathcal{F}, j \in \mathcal{D} \\
& y_i \ge 0 & \forall\, i \in \mathcal{F}
\end{aligned}
$$

The Dual LP

$$
\begin{aligned}
\max \quad & \sum_{j \in \mathcal{D}} v_j \\
\text{s.t.} \quad & v_j - w_{ij} \le c_{ij} && \forall\, i \in \mathcal{F}, j \in \mathcal{D} \\
& \sum_{j \in \mathcal{D}} w_{ij} \le f_i && \forall\, i \in \mathcal{F} \\
& w_{ij} \ge 0 && \forall\, i \in \mathcal{F}, j \in \mathcal{D}
\end{aligned}
$$

Some properties of the LP and its dual

Suppose we have a primal LP (P) of the form $\min c^\top x$, s.t., $Ax \leq b, x \geq 0$. Its dual is a packing LP $\max b^\top y$, s.t., $yA^\top \geq c, y \geq 0$.

Some properties of the LP and its dual

Suppose we have a primal LP (P) of the form $\min c^\top x$, s.t., $Ax \le b, x \ge 0$. Its dual is a packing LP $\max b^\top y$, s.t., $yA^\top \ge c, y \ge 0$.

Def A feasible solution x to (P) and a feasible solution y to (D) satisfy

- the primal complementary slackness condition with respect to each other if the following is true:
  - for each i, $x_i = 0$ or the corresponding dual constraint is tight, i.e., $\sum_j A_{i,j} y_j = c_i$
- the dual complementary slackness condition with respect to each other if the following is true:
  - for each j, $y_j = 0$ or the corresponding primal constraint is tight, i.e., $\sum_i A_{j,i} x_i = b_j$

Some properties of the LP and its dual

Suppose we have a primal LP (P) of the form $\min cx$, s.t., $Ax \leq b, x \geq 0$. Its dual is a packing LP $\max b^\top y$, s.t., $yA^\top \geq c, y \geq 0$.

Theorem: Suppose (P) and (D) are a primal and dual pair of LPs that both have finite optima. Then

- the optimum values of (P) and (D) are the same;
- a feasible solution x to (P) and a feasible solution y to (D) satisfy the primal-dual complementary slackness properties with respect to each other if and only if they are both optimum solutions to the respective LPs

LP-based algorithm for the metric UFL problem

Algorithm $\text{LP-FACILITYLOCATION}$

1. Compute an optimum solutions (x^*, y^*) and (v^*, w^*) to the primal and dual LP. (By complementary slackness, $x^*_{ij} > 0$ implies $v^*_j - w^*_{ij} = c_{ij}$ and thus $c_{ij} \leq v^*_j$)
2. Let G be the bipartite graph with vertex set $\mathcal{F} \cup \mathcal{D}$ containing an edge $\{i, j\}$ iff $x^*_{ij} > 0$
3. Assign clients to clusters iteratively as follows:
   3.1 In iteration k, let j_k be a client $j \in \mathcal{D}$ not assigned yet and with smallest v^*_j value.
   3.2 Create a new cluster containing j_k and those vertices of G that have distance 2 from j_k and not assigned yet
   3.3 Continue until all clients are assigned to clusters
4. For each cluster k, we choose a neighbor i_k of j_k with minimum f_{i_k}, open i_k, and assign all clients in this cluster to i_k

Analysis of LP-FacilityLocation

- The service cost for client j in cluster k is at most
$$c_{i_k j} \leq c_{ij} + c_{ij_k} + c_{i_k j_k} \leq v_j^* + 2v_{j_k}^* \leq 3v_j^*,$$
where i is the common neighbor of j and j_k
- The facility cost f_{i_k} can be bounded by
$$\begin{aligned} f_{i_k} &= f_{i_k} \sum_{i:\text{neighbor of } j_k} x_{ij_k}^* && \text{(by the equation constraint of the LP)} \\ &\leq \sum_{i:\text{neighbor of } j_k} f_i x_{ij_k}^* && \text{(i_k is cheapest neighboring facility of j_k)} \\ &\leq \sum_{i:\text{neighbor of } j_k} f_i y_i^* && \text{(by the LP constraint)} \end{aligned}$$
- since j_k, $j_{k'}$ cannot have a common neighbor for $k \neq k'$, the total facility cost is
$$\sum_k f_{i_k} \leq \sum_k \sum_{i:\text{neighbor of } j_k} f_i y_i^* \leq \sum_{i \in \mathcal{F}} f_i y_i^*$$

Analysis of LP-FacilityLocation

- The total cost is at most
$$3\sum_{j\in\mathcal{D}} v_j^* + \sum_{i\in\mathcal{F}} y_i^* f_i$$
- Note $\sum_{i\in\mathcal{F}} y_i^* f_i \le \mathrm{OPT}_{\mathrm{LP}}$, where $\mathrm{OPT}_{\mathrm{LP}} = \sum_{i\in\mathcal{F}} f_i y_i^* + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{D}} c_{ij}x_{ij}^*$ is the value of the optimum solution of LP.
- Note that by weak duality, $\sum_{j\in\mathcal{D}} v_j^* \le \mathrm{OPT}_{\mathrm{LP}}$.
- $\mathrm{OPT}_{\mathrm{LP}} \le \mathrm{OPT}$, where OPT is the value of optimum solution for the UFL problem.
- The running time is polynomial: solving LP and its dual and other steps, all can be done in polynomial time

Theorem: The algorithm LP-FacilityLocation is a 4-approximation algorithm for the metric UFL problem.