Clustering

Clustering

What?

- Given some input data, partition the data in multiple groups

Why?

- Approximate large/infinite/continuous set of objects with **finite** set of representatives
  - Eg. Vector quantization, codebook learning, dictionary learning
  - applications: HOG features for computer vision
- Find meaningful **groups** in data
  - In exploratory data analysis, gives a good understanding and summary of your input data
  - applications: life sciences

So how do we formally do clustering?

Clustering: the problem setup

Given a set of objects *X*, how do we compare objects?

need a way to compare objects

- We need a comparison function (via distances or similarities)

Given: a set *X* and a function $\rho : X \times X \rightarrow \mathbf{R}$

d needs to have some sensible structure

- (X,ρ) is a **metric space** iff for all $x_i, x_j, x_k \in X$

Perhaps we can make *d* a metric!

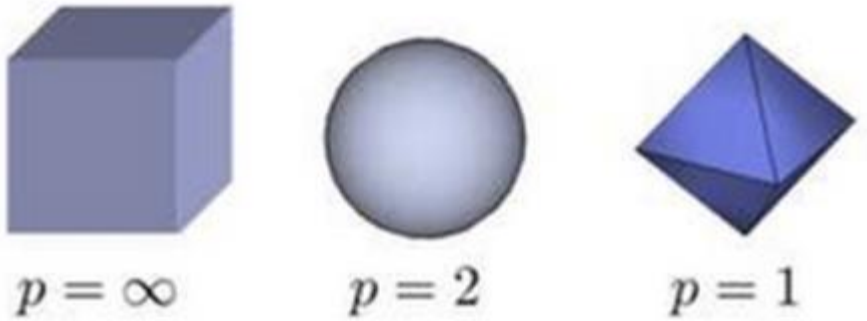
  - $\rho(x_i, x_j) \geq 0$ (equality iff $x_i = x_j$)
  - $\rho(x_i, x_j) = \rho(x_j, x_i)$
  - $\rho(x_i, x_j) \leq \rho(x_i, x_k) + \rho(x_k, x_j)$

A useful notation: given a set $T \subseteq X$

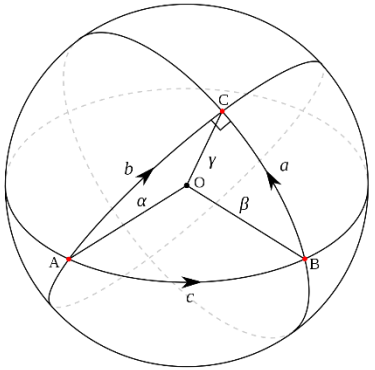
$$\rho(s, T) := \inf_{t \in T} \rho(s, t)$$

Examples of metric spaces

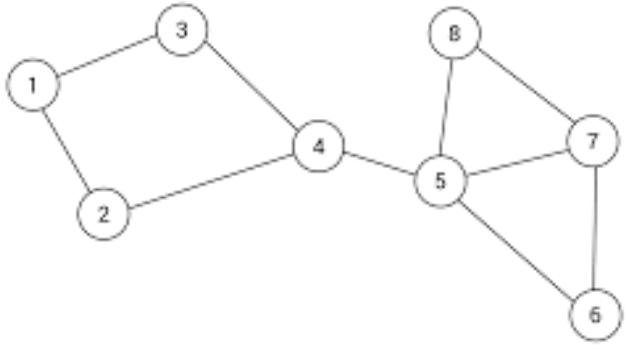
- L_2, L_1, L_∞ in $\mathbf{R}^d$



- (shortest) geodesics on manifolds;



- shortest paths on (unweighted) graphs



Covering of a metric space

- Covering, ε-covering, covering number

Given a set *X*

- $C \subseteq \wp(X)$, ie the powerset of *X*, is called a **cover** of $S \subseteq X$ iff

$$\bigcup_{c \in C} c \supseteq S$$

- if X is endowed with a metric ρ, then $C \subseteq X$ is an **ε-cover** of $S \subseteq X$ iff

$$\forall s \in S, \exists c \in C : \rho(s, c) \leq \epsilon$$

ie

$$\sup_{s \in S} \rho(s, C) \leq \epsilon$$

- **ε-**covering number $N(\varepsilon, S)$ of a set $S \subseteq X$, is the cardinality of the smallest ε-cover of *S*.

Examples of ε-covers of a metric space

- is S an ε-cover of S?

  Yes! For all $\varepsilon \geq 0$

- Let S be the vertices of a d-cube, ie, $\{-1,+1\}^d$ with L_∞ distance
  - Give a 1-cover?

    $C = \{ 0^d \}$ $N(1, S) = 1$

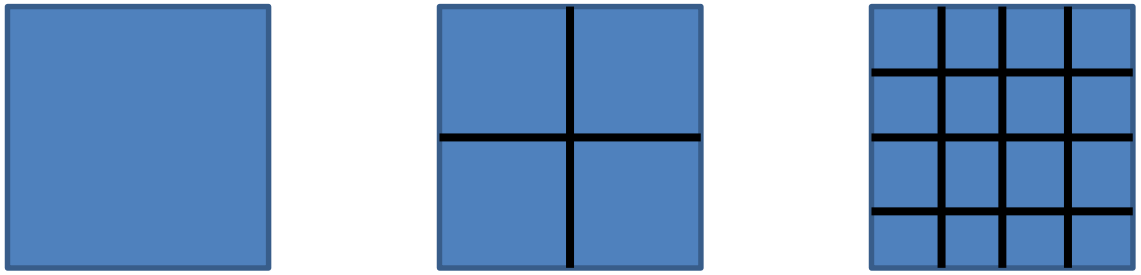
  - How about a ½-cover?

    $N(½, S) = 2^d$

  - 0.9 cover?
  - 0.999 cover? $N(0.999, S) = 2^d$ How do you prove this?

Examples of ε-covers of a metric space

- Consider $S = [-1,1]^2$ with L_∞ distance
  - what is a good 1-cover? ½-cover? ¼-cover?



What is the growth rate of $N(\varepsilon,S)$ as a function of ε ?

- What about $S = [-1,1]^d$?

What is the growth rate of $N(\varepsilon,S)$ as a function of the dimension of S?

The *k*-center problem

Consider the following optimization problem on a metric space (X,ρ)

Input: *n* points $x_1, \ldots, x_n \in X$; a positive integer *k*

Output: $T \subseteq X$, such that $|T| = k$

Goal: minimize the "*cost*" of *T*, define as

$$\mathrm{cost}(T) := \max_{x_1,\ldots,x_n} \rho(x_i, T)$$

How do we get the optimal solution?

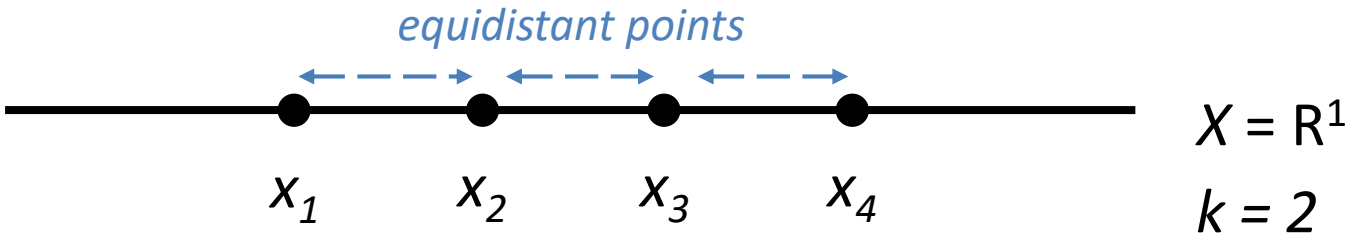
A solution to the *k*-center problem

- Run *k*-means?

  No… we are not in a Euclidean space (not even a vector space!)

- Why not try testing selecting *k* points from the given *n* points?

  Takes time… $\Theta(n^k)$ time, does not give the optimal solution!!



- Exhaustive search

  Try all partitionings of the given *n* datapoints in *k* buckets

  Takes very long time… $\Theta(k^n)$ time,

  unless the space is structured, unclear how to get the centers

- Can we do polynomial in both *k* and *n*?

  A greedy approach… farthest-first traversal algorithm

Farthest-First Traversal for *k*-centers

Let $S := \{ x_1, \ldots, x_n \}$

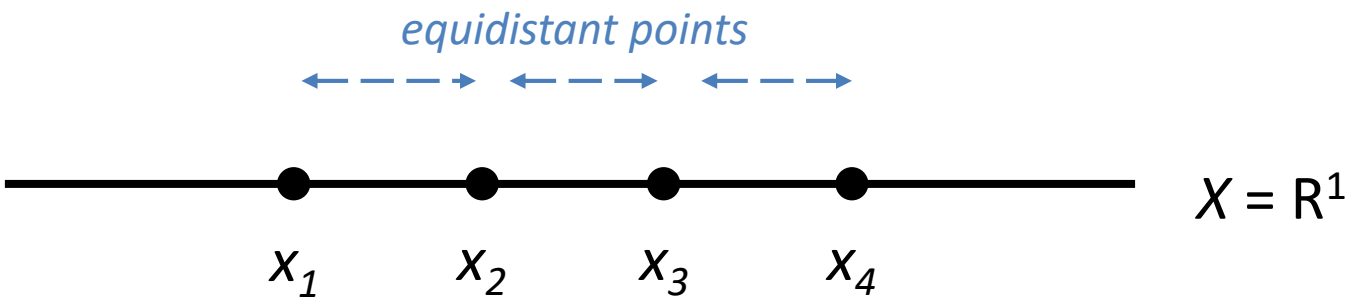
- arbitrarily pick $z \in S$ and let $T = \{ z \}$
- so long as $|T| < k$
  - $z := \text{argmax}_{x \in S} \rho(x, T)$
  - $T \leftarrow T \cup \{ z \}$
- return T

runtime?

solution quality?

Properties of Farthest-First Traversal

- The solution returned by farthest-first traversal is **not** optimal



- Optimal solution?

- Farthest first solution?

How does cost(**OPT**) vs cost(**FF**) Compare?

Properties of Farthest-First Traversal

For the previous example we know,

$$\text{cost(FF)} = 2\ \text{cost(OPT)}$$ [regardless of the initialization!]

But how about for a data in a general metric space?

Theorem: Farthest-First Traversal is 2-optimal for the *k*-center problem!

ie, $\text{cost(FF)} \leq 2\ \text{cost(OPT)}$ for *all* datasets and all *k*!!

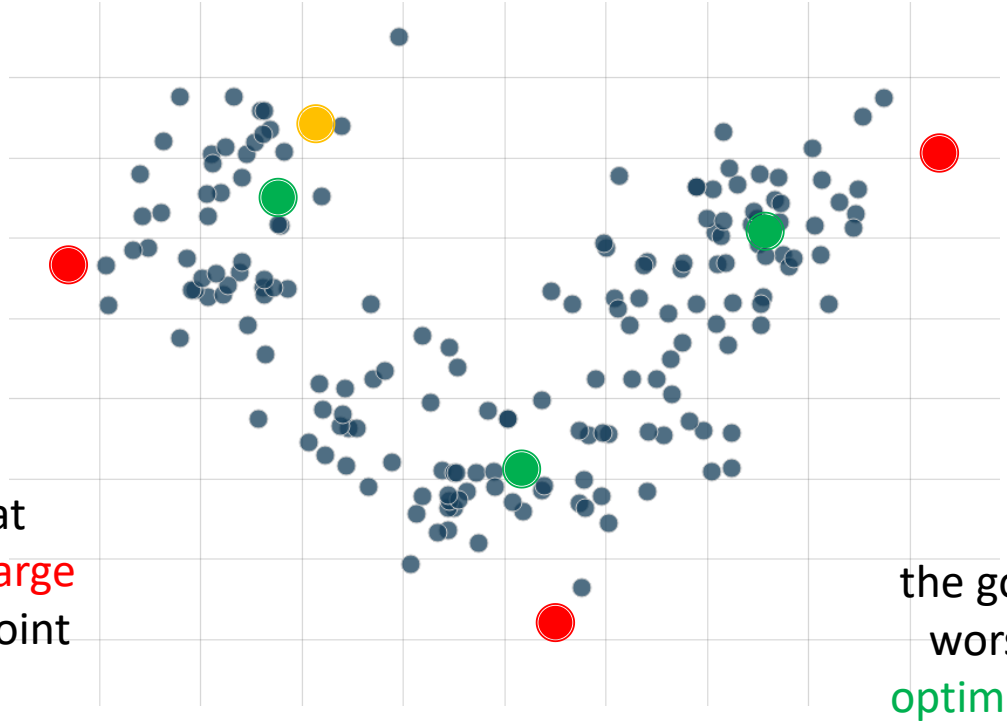
Properties of Farthest-First Traversal

Theorem: Let T* be an optimal solution to a given *k*-center problem, and let T be the solution returned by the farthest first procedure. Then,

$$\text{cost}(T^*) \leq \text{cost}(T) \leq 2\,\text{cost}(T^*)$$

Proof Visual Sketch:

Let's pick another point



say k = 3

optimal assignment

farthest first assignment

If we can ensure that optimal must incur a large cost in covering this point then we are good

the goal is to compare worst case cover of optimal to farthest first

Properties of Farthest-First Traversal

Theorem: Let T* be an optimal solution to a given *k*-center problem, and let T be the solution returned by the farthest first procedure. Then,

$$\text{cost}(T^*) \leq \text{cost}(T) \leq 2\,\text{cost}(T^*)$$

Proof:

Let r := cost(T) = $\max_{x \in S} \rho(x, T)$, let x_0 be the point which attains the max

Let T' := T U $\{x_0\}$

Observation:

- for all distinct t,t' in T', $\rho(t, t') \geq r$
- $|T^*| = k$ and $|T'| = k+1$
- must exists $t^* \in T^*$, that covers at least two elements t_1, t_2 of T'

Thus,

since $\rho(t_1, t_2) \geq r$, it must be that either $\rho(t_1, t^*)$ or $\rho(t_2, t^*) \geq r/2$

Therefore: $\text{cost}(T^*) \geq r/2$.

Doing better than Farthest-First Traversal

can you do better than Farthest First traversal for the *k*-center problem?

- k-centers problem is NP-hard!

  proof: see hw1 ☺

- in fact, even (2- ε)-poly approximation is not possible for general metric spaces (unless P = NP) [Hochbaum '97]

k-center open problems

Some related **open problems**:

- Hardness in Euclidean spaces (for dimensions $d \geq 2$)?
  - Is *k*-center problem hard in Euclidean spaces?
  - Can we get a better than 2-approximation in Euclidean spaces?
  - How about hardness of approximation?

- Is there an algorithm that works better in practice than the farthest-first traversal algorithm for Euclidean spaces?

Interesting extensions:

- asymmetric k-centers problem, best approx. O(log*(*k*)) [Archer 2001]
- How about average case?
  - Under "perturbation stability", you can do better [Balcan et al. 2016]

The *k*-medians problem

- A variant of *k*-centers where the cost is the aggregate distance
  (instead of worst-case distance)

Input: *n* points $x_1, \ldots, x_n \in X$; a positive integer *k*

Output: $T \subseteq X$, such that $|T| = k$

Goal: minimize the "*cost*" of *T*, define as

$$\mathrm{cost}(T) := \sum_{x_1,\ldots,x_n} \rho(x_i, T)$$

remark: since it considers the aggregate, it is somewhat robust to outliers
(a single outlier does not necessarily dominate the cost)

An LP-Solution to *k*-medians

Observation: the objective function is linear in the choice of the centers

perhaps it would be amenable to a linear programming (LP) solution

Let $S := \{ x_1, \ldots, x_n \}$

Define two sets of binary variables y_j and x_{ij}

- y_j := is j^{th} datapoint one of the centers? $j = 1,\ldots,n$
- x_{ij} := is i^{th} datapoint assigned to cluster centered at j^{th} point $i,j = 1,\ldots,n$

Example: S = {0,2,3}, T = {0,2}

datapoint "0" is assigned to cluster "0"

datapoint "2" and "3" are assigned to cluster "2"

$x_{11} = x_{22} = x_{32} = 1$ (the rest of x_{ij} are zero); $y_1 = y_2 = 1$ and $y_3 = 0$

k-medians as an (I)LP

y_j := is j one of the centers
x_{ij} := is i assigned to cluster j

$$\min_{(x_{ij},y_j)_{i,j\in[n]}} \sum_{i,j} x_{ij}\rho(x_i,x_j)$$

Tally up the cost of all the distances between points and their corresponding centers

Linear

such that

$$\sum_j x_{ij} = 1 \qquad \forall i \in [n]$$

Each point is assigned to exactly on cluster

$$\sum_j y_j = k \qquad \forall i \in [n]$$

There are exactly k clusters

$$x_{ij} \leq y_j \qquad \forall i,j \in [n]$$

i^{th} datapoint is assigned to j^{th} point only if it is a center

Discrete / Binary

$$x_{ij} \in \{0,1\}, y_j \in \{0,1\} \qquad \forall i,j \in [n]$$

The variables are binary

Properties of an ILP

Any NP-complete problem can be written down as an ILP

Why?

Can be relaxed into an LP.

- How?

  Make the integer constraint into a 'box' constraint… $x_{ij} \in \{0,1\}, y_j \in \{0,1\}$

  $x_{ij} \in [0,1], y_j \in [0,1]$

- Advantages
  - Efficiently solvable.
  - Can be solved by off-the-shelf LP solvers
    - Simplex method (exp time in worst case but usually very good)
    - Ellipsoid method (proposed by von Neumann, $O(n^6)$)
    - Interior point method (Karmarkar's algorithm '84, $O(n^{3.5})$)
    - Cutting plane method
    - Criss-cross method
    - Primal-dual method

Properties of an ILP

Any NP-complete problem can be written down as an ILP

Can be relaxed into an LP.

- Advantages – Efficiently solvable
- Disadvantages
  - Gives a fractional solution (so not an exact solution to the ILP)
  - Conventional fixes – do some sort of rounding mechanism

    Deterministic rounding
    - Can be shown to have arbitrarily bad approximation.

    Randomized rounding (*flip a coin with the bias as per the fractional cost and assign the value as per the outcome of the coin flip*)
    - Can be sometimes have good average case or with high probability!
    - Sometimes the solution is not even in the desired solution set!
    - Derandomization procedures exist!

Back to *k*-medians… with LP relaxation

y_j := is j one of the centers
x_{ij} := is i assigned to cluster j

note: $\text{cost}(\text{OPT}_{\text{LP}}) \leq \text{cost}(\text{OPT})$

Linear

$$\min_{(x_{ij},y_j)_{i,j\in[n]}} \sum_{i,j} x_{ij}\rho(x_i,x_j)$$

Tally up the cost of all the distances between points and their corresponding centers

such that

$$\sum_j x_{ij} = 1 \qquad \forall i \in [n]$$

Each point is assigned to exactly on cluster

$$\sum_j y_j = k \qquad \forall i \in [n]$$

There are exactly k clusters

$$x_{ij} \leq y_j \qquad \forall i,j \in [n]$$

i^{th} datapoint is assigned to j^{th} point only if it is a center

Also LINEAR!

$$x_{ij} \in [0,1], y_j \in [0,1] \qquad \forall i,j \in [n]$$

RELAXATION to box constraints

A Deterministic procedure for *k*-medians LP

$S := \{ x_1, \ldots, x_n \}$, data from a metric space (X,ρ); k = # centers

y_j := is j^{th} datapoint one of the centers?

x_{ij} := is i^{th} datapoint assigned to cluster centered at j^{th} point? $i,j \in [n]$

The Algorithm [Lin and Vitter '92]

Run the LP for *k*-medians problem on input S, with k centers

Define $c_i := \sum_j x_{ij}\, \rho(x_i, x_j)$ $\quad i \in [n]$

note: $\sum_i c_i = \text{cost}(\text{OPT}_{LP})$

$T \leftarrow \phi$

while $S \neq \phi$

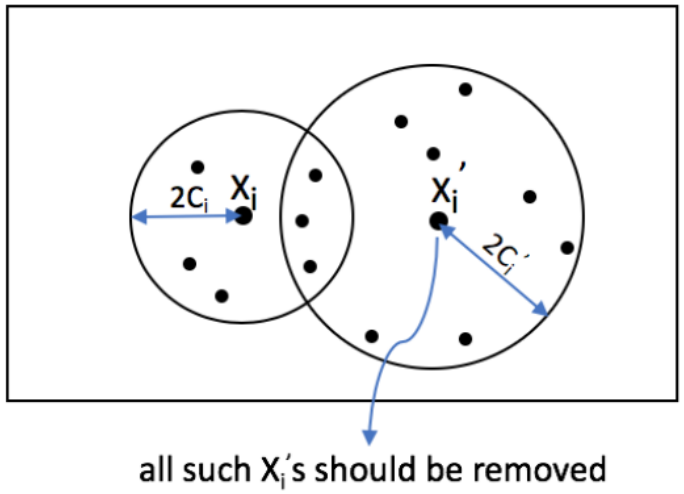
pick $x_i \in S$ with smallest c_i

$T \leftarrow T \cup \{ x_i \}$

$A_i := \{ x_{i'} : B(x_i, 2c_i) \cap B(x_{i'}, 2c_{i'}) \neq \phi \}$

$S \leftarrow S \setminus A_i$

return T



how good the output set T ? cost(T) ? $|T|$?

Properties of the deterministic procedure

Theorem 1: $\text{cost}(T) \le 4\ \text{cost}(\text{OPT}_{LP})$

$\Rightarrow \text{cost}(T) \le 4\ \text{cost}(\text{OPT})$

Got an approx. good solution in (deterministic) poly time!

Theorem 2: $|T| \le 2k$

umm… not exactly k centers… but close enough ☺

Remark: The result can be generalized to

$\text{cost}(T) \le 2(1+1/\varepsilon)\ \text{cost}(\text{OPT}_{LP})$, with $|T| \le (1+\varepsilon)k$

[when $A_i := \{ x_{i'} : B(x_i, (1+1/\varepsilon)c_i) \cap B(x_{i'}, (1+1/\varepsilon)c_{i'}) \}$]

Properties of the deterministic procedure

Run the LP

Define $c_i := \Sigma_j x_{ij}\, \rho(x_i, x_j)$ $\quad i \in [n]$

$T \leftarrow \phi$

while $S \neq \phi$

pick x_i with smallest c_i

$T \leftarrow T \cup \{ x_i \}$

$A_i := \{x_{i'}: B(x_i, 2c_i) \cap B(x_{i'}, 2c_{i'}) \neq \phi\}$

$S \leftarrow S \setminus A_i$

return T

Theorem 1: $\text{cost}(T) \leq 4\, \text{cost}(\text{OPT}_{\text{LP}})$

Proof:

Pick any $x_q \in S$ and let x_i is the first point in T for which $x_q \in A_i$, then

- $c_i \leq c_q$
- $\rho(x_q, x_i) \leq 4\, c_q$

why?

$\exists\, x_p \in S$ s.t. $\rho(x_q, x_i) \leq \rho(x_q, x_p) + \rho(x_p, x_i) \leq 2c_q + 2c_i \leq 4\, c_q$

sum over all points q, we get…. $\text{cost}(T) \leq 4\, \text{cost}(\text{OPT}_{\text{LP}})$!

Properties of the deterministic procedure

Theorem 2: $|T| \leq 2k$

Proof:

Pick any $x_i \in T$, then

$$\underbrace{\sum_{j \in B(Xi,2Ci)} y_j \geq \sum_{j \in B(Xi,2Ci)} x_{ij}}_{\text{LP}} \geq \tfrac{1}{2}$$

via Markov's Ineq!

$$\min_{(x_{ij},y_j)_{i,j\in[n]}} \sum_{i,j} x_{ij}\rho(x_i,x_j)$$

$$\text{s.t.} \quad \sum_j x_{ij} = 1 \qquad \sum_j y_j = k$$

$$x_{ij} \leq y_j \qquad x_{ij} \in [0,1], y_j \in [0,1]$$

Markov: for Z non-neg
$P[Z \geq a] \leq E[Z]/a$

Recall $\forall i$: (i) $\Sigma_j x_{ij} = 1$ (ii) $c_i := \Sigma_j x_{ij} \rho(x_i,x_j) = E_x \rho(x_i,x_j)$

Define: random variable Z_i takes value $\rho(x_i,x_j)$ with probability x_{ij}

So: (i) $EZ_i = c_i$

(ii) $\Sigma_{j \in B(Xi,2Ci)} x_{ij} = P[Z_i \leq 2c_i] = P[Z_i \leq 2\,EZ_i] \geq 1 - P[Z_i \geq 2\,EZ_i] \geq \tfrac{1}{2}$

Properties of the deterministic procedure

$$\min_{(x_{ij},y_j)_{i,j\in[n]}} \sum_{i,j} x_{ij}\rho(x_i,x_j)$$

$$\text{s.t.} \quad \sum_j x_{ij} = 1 \qquad \sum_j y_j = k$$

$$x_{ij} \le y_j \qquad x_{ij} \in [0,1], y_j \in [0,1]$$

Theorem 2: $|T| \le 2k$

Proof:

Pick any $x_i \in T$, then

$\Sigma_{j\in B(Xi,2Ci)}\, y_j \;\ge\; \Sigma_{j\in B(Xi,2Ci)}\, x_{ij} \;\ge\; ½$

So, $k = \Sigma_j y_j \;\ge\; \Sigma_{Xi\in T}\Sigma_{j\in B(Xi,2Ci)}\, y_j \;\ge\; \Sigma_{Xi\in T}\Sigma_{j\in B(Xi,2Ci)}\, x_{ij} \;\ge\; |T|/2$

because the balls are disjoint by choice of x_i in T *via* A_i

■

Related problems to *k*-medians

- asymmetric k-medians is known to be hard to approximate via factor $O(\log^*(k)\text{-}\Omega(1))$

The *k*-means problem

Input: *n* points $x_1, \ldots, x_n \in R^d$; a positive integer *k*

Output: $T \subseteq X$, such that $|T| = k$

Goal: minimize the "*cost*" of *T*, define as

$$\text{cost}(T) := \sum_{i=1}^{n} \min_{\mu_j \in T} \|x_i - \mu_j\|^2$$

A solution to the *k*-means problem

- Exhaustive search

  Try all partitionings of the given n datapoints in k buckets

  Takes very long time… $\Theta(k^n)$ time,

  once we have the partitions, it's easy to get the centers

- An efficient exact algorithm?

  Unfortunately no… unless P = NP, or if $k = 1$ or $d = 1$

- Some approximate solutions

  Lloyd's method (most popular method!), Hartigan's method

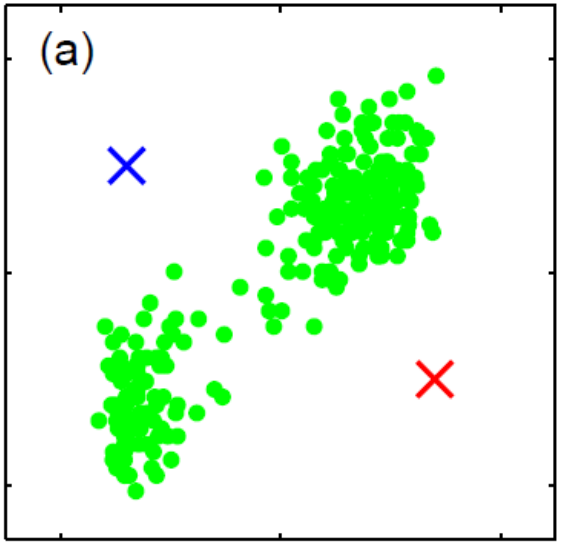
Lloyd's method to approximate *k*-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and intended number of groupings *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (assuming the data partition is fixed)

Demo:



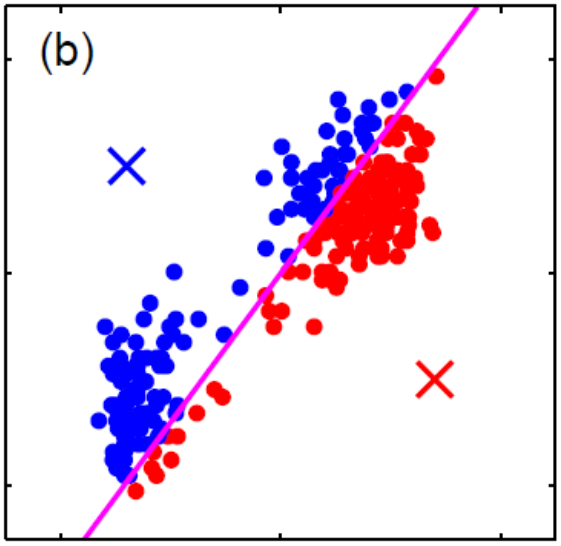
Lloyd's method to approximate *k*-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and intended number of groupings *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (assuming the data partition is fixed)

Demo:



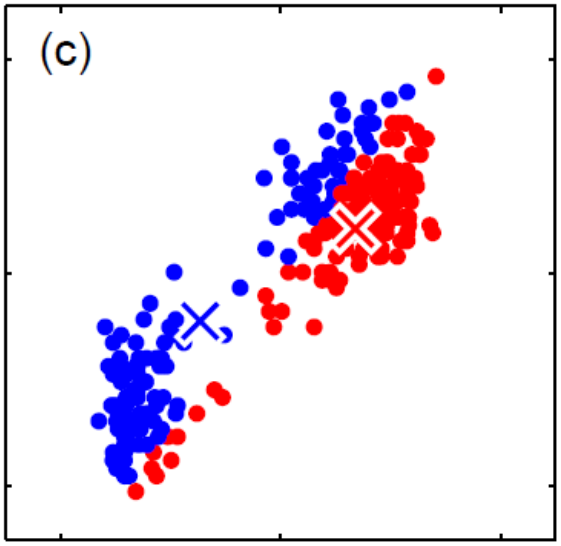
Lloyd's method to approximate *k*-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and intended number of groupings *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (assuming the data partition is fixed)

Demo:



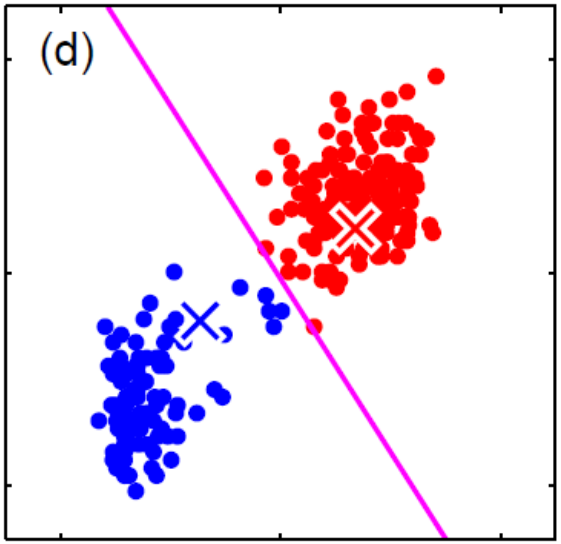
Lloyd's method to approximate *k*-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and intended number of groupings *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (assuming the data partition is fixed)

Demo:



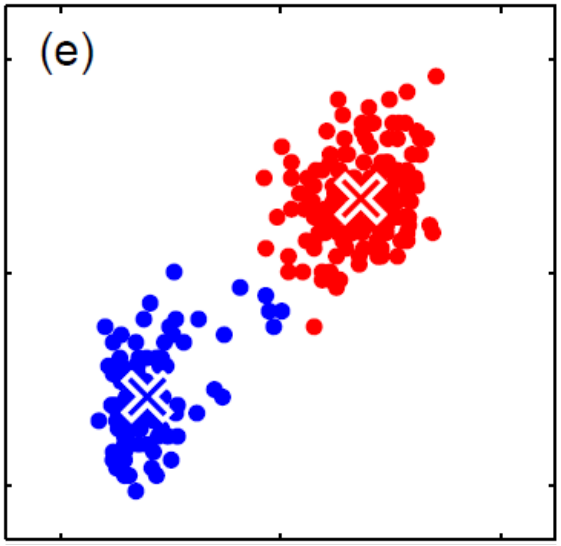
Lloyd's method to approximate *k*-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and intended number of groupings *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (assuming the data partition is fixed)

Demo:



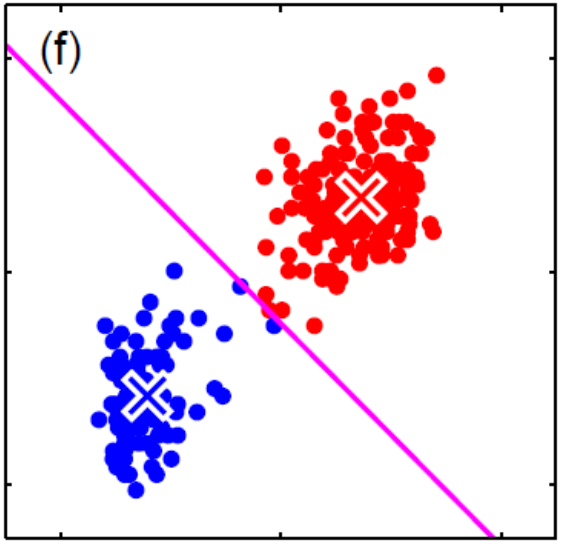
Lloyd's method to approximate *k*-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and intended number of groupings *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (assuming the data partition is fixed)

Demo:



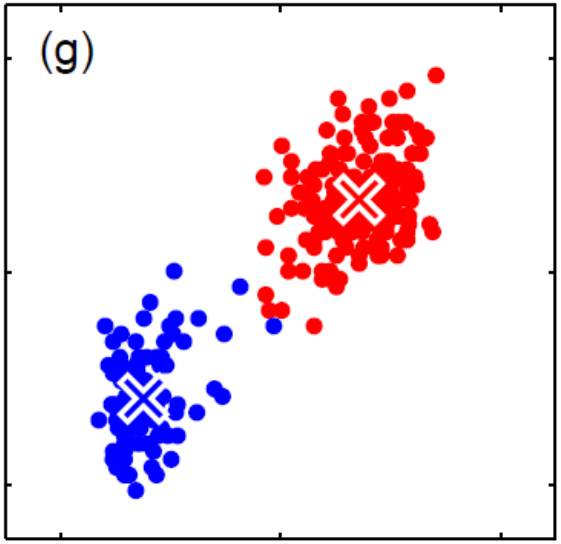
Lloyd's method to approximate *k*-means

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and intended number of groupings *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (say randomly)
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition) (assume centers are fixed)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (assuming the data partition is fixed)

Demo:

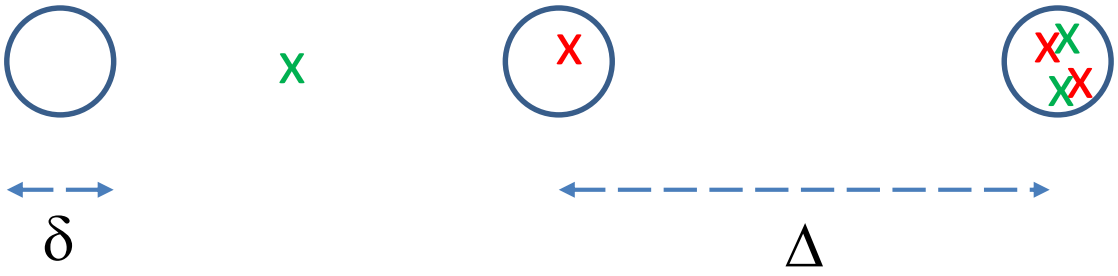


Properties of Lloyd's method

The quality of the output/centers returned by the Lloyd's method can be arbitrarily bad!

That is, the ratio $\text{cost}(T_{LLOYD}) / \text{cost}(OPT)$ is unbounded

This is the case for even seemingly ideal inputs…



$k = 3$

at (random) initialization

at convergence

$\text{cost}(OPT) = O(n\delta^2)$ $\text{cost}(T_{LLOYD}) = \Omega(n\Delta^2)$

What about farthest first initialization?

does not work when data has some outliers

Hardness of *k*-means

Theorem: *k*-means optimization is NP-hard

We'll show a reduction from a known hard problem to 2-means…

$$\text{3SAT} \leq_P \text{NAE-3SAT} \leq_P \text{NAE-3SAT*} \leq_P \text{Generalized 2-means} \leq_P \text{2-means}$$

[Dasgupta '08]

First, we need a reformulation of *k*-means
and formally define the generalized *k*-means problem

k-means reformulation

Formulation 1

Input: *n* points $\{ x_1, \dots, x_n \} = S \subseteq R^d$; a positive integer *k*

Output: $T \subseteq X$, such that $|T| = k$

Goal: minimize the "*cost*" of *T*, define as

$$\text{cost}(T) := \sum_{i=1}^{n} \min_{\mu_j \in T} \|x_i - \mu_j\|^2$$

Optimal

$\mu_j = E_{i \in Pj}\, x_i$

Input: *n* points $\{ x_1, \dots, x_n \} = S \subseteq R^d$; a positive integer *k*

Output: $P_1,\dots,P_k \subseteq [n]$, $\bigsqcup P_i = [n]$, $\mu_1,\dots, \mu_k \in R^d$

Goal: minimize the "*cost*" of $(P_1,\dots,P_k ; \mu_1,\dots, \mu_k)$

$$\text{cost}(P_1, \dots, P_k; \mu_1, \dots, \mu_k) := \sum_{j=1}^{k} \sum_{i \in P_j} \|x_i - \mu_j\|^2$$

Formulation 2

k-means reformulation

Observation: $E_X \| X - EX \|^2 = \frac{1}{2} E_{X,Y} \| X - Y \|^2$

Why? Basic algebra…

Input: *n* points $\{ x_1, \dots, x_n \} = S \subseteq R^d$; a positive integer *k*

Output: $P_1,\dots,P_k \subseteq [n]$, $\bigsqcup P_i = [n]$

Goal: minimize the "*cost*" of $(P_1,\dots,P_k)$

$$\text{cost}(P_1, \dots, P_k) := \sum_{j=1}^{k} \frac{1}{2|P_j|} \sum_{i,i' \in P_j} \|x_i - x_{i'}\|^2$$

Formulation 3

A distance-based generalization of *k*-means

Standard *k*-means

Input: *n* points $\{ x_1, \ldots, x_n \} = S \subseteq R^d$; a positive integer *k*

Output: $P_1,\ldots,P_k \subseteq [n]$, $\bigsqcup P_i = [n]$

Goal: minimize the "*cost*" of $(P_1,\ldots,P_k)$

$$\text{cost}(P_1, \ldots, P_k) := \sum_{j=1}^{k} \frac{1}{2|P_j|} \sum_{i,i' \in P_j} \|x_i - x_{i'}\|^2$$

D_{ij} can be viewed as sq. Euclidean distances between x_i and x_j

Input: *n* x *n* symmetric matrix *D*; a positive integer *k*

Output: $P_1,\ldots,P_k \subseteq [n]$, $\bigsqcup P_i = [n]$

Goal: minimize the "*cost*" of $(P_1,\ldots,P_k)$

$$\text{cost}(P_1, \ldots, P_k) := \sum_{j=1}^{k} \frac{1}{2|P_j|} \sum_{i,i' \in P_j} D_{i,i'}$$

Generalized *k*-means

A quick review of NP hardness and reductions

- NP-hard problems admit polynomial time reductions from all other problems in NP

  notation: Given two (decision) problems A and B

  $A \leq_P B$ A reduces to B (in poly-time)

  usage: Want to show B "hard". Pick a known hard problem A.
  Assume B can be solved. Show that then A can be solved.
  Therefore B is at least as hard as A

- Specifically, how to show a reduction?
  - Given an instance α of A, transform (in poly-steps) into an instance β of B
  - Run decision algorithm for B on instance β
  - Use the solution of β to get a solution for α

Hardness of *k*-means

Theorem: *k*-means optimization is NP-hard

We'll show a reduction from a known hard problem to 2-means…

known (hard) problem A

show problem B is at least as hard

$$3\text{SAT} \leq_P \text{NAE-3SAT} \leq_P \text{NAE-3SAT*} \leq_P \text{Generalized 2-means} \leq_P \text{2-means}$$

[Dasgupta '08]

known (hard) problems

Known hard problems

3-SAT

Input: A 3-CNF Boolean formula over *n* variables

Output: True iff an assignment exists satisfying the formula

3-Conjuncitve Normal Form (CNF)
A Boolean formula expressed as an AND over m clauses, each of which has exactly 3 literals

Example:

Variables: $x_1, x_2, x_3, \ldots, x_n$ each $x_i \in \{0,1\}$

Formula (3-CNF):

$$(x_1 \vee x_5 \vee \neg x_{32}) \wedge (x_{26} \vee \neg x_{18} \vee \neg x_{11}) \wedge (x_5 \vee x_{33} \vee x_{89}) \ldots$$

literal

clause

Known hard problems

3-SAT

Input: A 3-CNF Boolean formula over *n* variables

Output: True iff an assignment exists satisfying the formula

NAE-3SAT (or "Not All Equal" 3-SAT)

3SAT with an additional requirement that in each clause there is at least one literal that is true, and at least one literal that is false.

NAE-3SAT* (a modification on NAE-3SAT)

Each pair (x_i, x_j) of variables appear in at most 2 clauses.

- once as: either $(x_i \vee x_j)$ or $(\neg x_i \vee \neg x_j)$, and
- once as: either $(\neg x_i \vee x_j)$ or $(x_i \vee \neg x_j)$

Generalized 2-means

Input: *n* x *n* symmetric matrix *D*

Output: $P_1, P_2 \subseteq [n]$, $P_1 \sqcup P_2 = [n]$

Goal: minimize the "*cost*" of (P_1, P_2)

$$\text{cost}(P_1, P_2) := \sum_{j=1}^{2} \frac{1}{2|P_j|} \sum_{i,i' \in P_j} D_{i,i'}$$

will first show… NAE-3SAT* $\leq_P$ Generalized 2-means

[Dasgupta '08]

Hardness of Generalized 2-means

Theorem: NAE-3SAT* $\leq_P$ Generalized 2-means

Proof:

Given an instance ϕ of NAE-3SAT* with n variables $x_1,\ldots,x_n$ & m clauses

We'll construct an instance of generalized 2-means as follows.

Let $2n$ x $2n$ distance matrix $D(\phi)$, each row/col corresponds to the variables $x_1,\ldots,x_n, \neg x_1,\ldots,\neg x_n$. Defined as

$$D_{\alpha,\beta} = \begin{cases} 0 & if\ \alpha = \beta \\ 1+\Delta & if\ \alpha = \overline{\beta} \\ 1+\delta & if\ \alpha \sim \beta \\ 1 & otherwise. \end{cases}$$

$\alpha \sim \beta$ means that either variable α and β or $\neg\alpha$ and $\neg\beta$ occurred together in a clause in ϕ

$$\Delta := \frac{5m}{5m+2n} \qquad \delta := \frac{1}{5m+2n}$$

observations

$$0 < \delta < \Delta < 1$$

$$4\delta m < \Delta \leq 1 - 2\delta n$$

Proof Contd.

A quick example:

Let NAE-3SAT* instance be: $(x_1 \vee \neg x_2 \vee x_3)$

$$D_{\alpha,\beta} = \begin{cases} 0 & \text{if } \alpha = \beta \\ 1+\Delta & \text{if } \alpha = \overline{\beta} \\ 1+\delta & \text{if } \alpha \sim \beta \\ 1 & \text{otherwise.} \end{cases}$$

	x_1	x_2	x_3	$\bar{x_1}$	$\bar{x_2}$	$\bar{x_3}$
x_1	0	1	$1+\delta$	$1+\Delta$	$1+\delta$	1
x_2	1	0	1	$1+\delta$	$1+\Delta$	$1+\delta$
x_3	$1+\delta$	1	0	1	$1+\delta$	$1+\Delta$
$\bar{x_1}$	$1+\Delta$	$1+\delta$	1	0	1	$1+\delta$
$\bar{x_2}$	$1+\delta$	$1+\Delta$	$1+\delta$	1	0	1
$\bar{x_3}$	1	$1+\delta$	$1+\Delta$	$1+\delta$	1	0

Agenda: the instance ϕ of NAE-3SAT* is satisfiable iff $D(\phi)$ admits generalized 2-means cost of $n - 1 + (2\delta m/n)$.

Proof Contd.

Lemma: If the instance ϕ of NAE-3SAT* is satisfiable, then $D(\phi)$ admits generalized 2-means cost of $n - 1 + (2\delta m/n) =: c(\phi)$.

Consider any satisfiable assignment of ϕ, and partition all the $(2n)$ literals into those assigned `true` (partition P_1) and those assigned `false` (partition P_2).

$$|P_1| = |P_2| = n$$

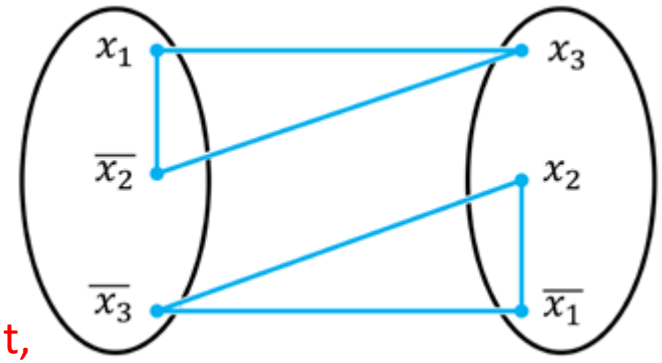
(x_i in P1 iff $\neg x_i$ in P2)

By defn of NAE-3SAT*, each clause contributes one pair to P_1 and one to P_2

$$\text{cost}(P_1, P_2) = \sum_{j=1}^{2} \frac{1}{2|P_j|} \sum_{i,i' \in P_j} D_{i,i'}$$

$$= 2 \cdot \frac{1}{2n} \cdot 2\left[\binom{n}{2} + \delta m \right]$$

all pairs contribute 1 unit,
m pairs contribute $1+\delta$



■

Proof Contd.

$C(\phi) = n - 1 + (2\delta m/n)$

Lemma: For any partition P_1, P_2 which contains a variable and its negation, then $\text{cost}(P_1,P_2) \geq n - 1 + \Delta/(2n) > c(\phi)$

Let $n' = |P_1|$, then $$\text{cost}(P_1, P_2) = \sum_{j=1}^{2} \frac{1}{2|P_j|} \sum_{i,i' \in P_j} D_{i,i'}$$

$$> \frac{1}{n'}\left(\binom{n'}{2} + \Delta\right) + \frac{1}{2n - n'}\binom{2n - n'}{2} = n - 1 + \frac{\Delta}{n'} \geq n - 1 + \frac{\Delta}{2n}$$

all pairs contribute at least 1 unit

Since $\Delta > 4\,\delta m$, $\text{cost}(P_1,P_2) > c(\phi)$

Proof Contd.

$C(\phi) = n - 1 + (2\delta m/n)$

Lemma: If $D(\phi)$ admits to a 2-clustering of cost $\leq c(\phi)$, then ϕ is a satisfiable instance of NAE-3SAT*

Let P_1,P_2 be a 2-clustering with cost $\leq c(\phi)$, then P_1 & P_2 cannot contain a variable and its negation (see previous lemma) $|P_1|=|P_2|=n$

Then the clustering cost is

$$\frac{2}{n}\left[\binom{n}{2} + \delta \sum_{\text{clauses}} \begin{cases} 1 & \text{if clause is split across } P_1 \text{ and } P_2 \\ 3 & \text{otherwise} \end{cases}\right]$$

since this is $\leq c(\phi)$, it must be that each clause is split across.

Therefore it is a satisfiable instance of NAE-3SAT*!

■

Generalized 2-means is hard

Hence, NAE-3SAT* $\leq_P$ Generalized 2-means

Hardness of *k*-means

Theorem: *k*-means optimization is NP-hard

We'll show a reduction from a known hard problem to 2-means…

3SAT $\leq_P$ NAE-3SAT $\leq_P$ NAE-3SAT* $\leq_P$ Generalized 2-means $\leq_P$ 2-means

[Dasgupta '08]

Hardness of 2-means

Theorem: Generalized 2-means $\leq_P$ 2-means

Need to show generalized 2-means in embeddable in R^d so that we can run a 2-means to solve it.

Claim: Any n x n symmetric matrix $D(\phi)$ can be embedded in squared L_2 iff $u^T D u \leq 0$ for all u in R^n s.t. $\Sigma_i u_i = 0$.

proof… see hw2 ☺

Hardness of *k*-means… thoughts

We have shown that 2-means is hard in $d = \Theta(n)$ dimensions

What about when $d = 2$?

- There are elegant reductions available when $k = 2$ and $d = 2$.

[Vattani '09, Aloise et al. '09, Mahajan et al. '09]

Approximating *k*-means with guarantees

Lloyd's method

Given: data $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n \in \mathbf{R}^d$, and number of centers *k*

Alternating optimization algorithm:

- Initialize cluster centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$
- Repeat till no more changes occur
  - Assign data to its closest center (this creates a partition)
  - Find the optimal centers $\vec{c}_1, \vec{c}_2, \ldots \vec{c}_k$ (for the partition)

Heavily depends on initialization

- Random initialization doesn't work
- Farthest first is sensitive to outliers
- Can something else be done?
  - We explore **probabilistic farthest first initialization**!

Probabilistic farthest first for *k*-means

Probabilistic farthest first initialization (kmeans++) [Arthur and Vassilvitskii'07]

- Initialize C by picking a point x_i uniform at random from the dataset S
- Pick a new center c_j as the point x_i from S with probability

$$P_i := \rho^2(x_i, C) / \sum_{x_k \in S} \rho^2(x_k, C)$$

- $C \leftarrow C \cup \{c_j\}$
- Repeat till $|C| = k$

Theorem: Let C be the initialization via kmeans++

$$\mathbf{E}[\text{cost}(C)] \leq O(\log(k))\ \text{cost}(\text{OPT})$$

Approximation guarantee for *k*means++

Theorem: Let *C* be the initialization via kmeans++ [Arthur and Vassilvitskii'07]

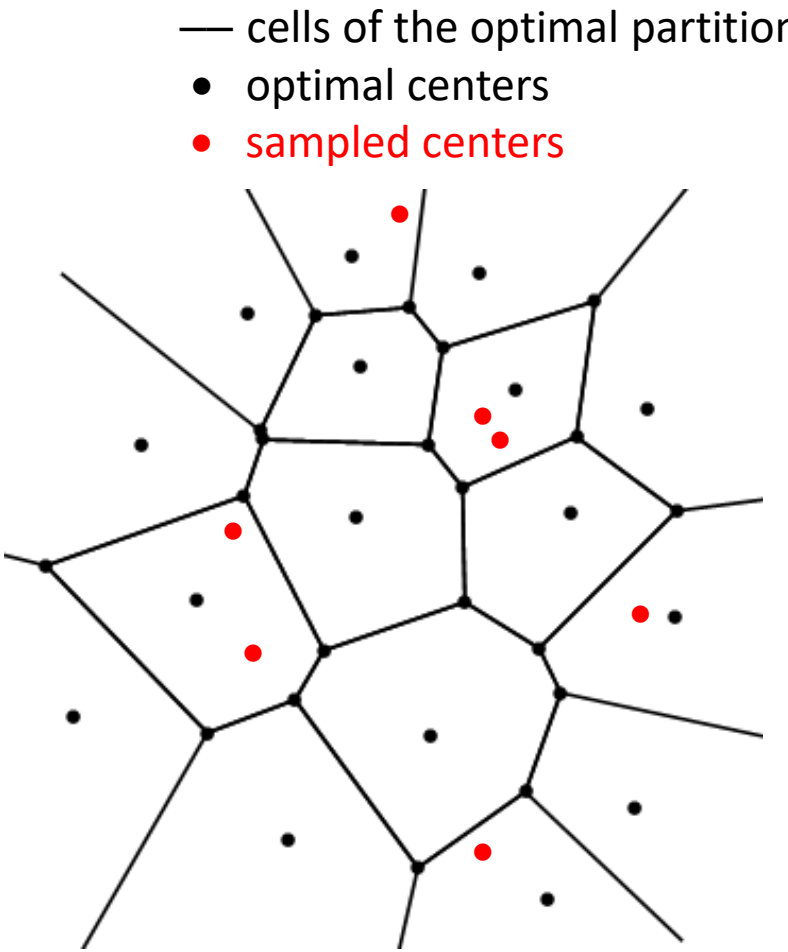
$$\mathbf{E}[\text{cost}(C)] \leq 8(\log(k)+2)\,\text{cost}(\text{OPT})$$

Proof Idea:

Consider the partition induced by the optimal clustering and analyze how the probabilistic sampling covers these cells.

If a sample hits a cell, then its relative cost would be small.

Ideally want to show that all/most cells are hit.



Approximation guarantee for *k*means++

Observation: For a set of points $S = \{x_1, \ldots, x_n\}$ and any z

$$\sum_{x \in S} \| x - z \|^2 = \sum_{x \in S} \| x - \mu_S \|^2 + |S| \, \| x - \mu_S \|^2.$$

Notation:

- $\phi(A)$ = cost of subset of datapoints $A \subseteq S$ wrt centers C
- $\phi = \phi(S)$ = cost(C)
- $\phi_{OPT}(A)$ = cost of subset of datapoints $A \subseteq S$ wrt centers OPT
- $\phi_{OPT} = \phi_{OPT}(S)$ = cost(OPT)

Let's analyze how the probabilistic kmeans++ initialization affects the cost

Approximation guarantee for *k*means++

Claim: Let A be a cell in induced by OPT. Let C be just one cluster chosen u.a.r. from A, then $\mathbf{E}[\phi(A)] = 2\phi_{OPT}(A)$

Proof:

$$\mathbf{E}[\phi(A)] = \frac{1}{|A|}\sum_{a_0 \in A}\sum_{a \in A}\|a - a_0\|^2$$

$$= \frac{1}{|A|}\sum_{a_0 \in A}\left(\sum_{a \in A}\|a - \mu_A\|^2 + |A| \cdot \|a_0 - \mu_A\|^2\right)$$

$$= 2\sum_{a \in A}\|a - \mu_A\|^2$$

Approximation guarantee for *k*means++

Claim: Let A be a cell in induced by OPT. Let C be an arbitrary set of clusters. If we add a random center to C from A (with probabilistic farthest first weighting), then $\mathbf{E}[\phi(A)] \leq 8\,\phi_{OPT}(A)$

Proof:

$$\mathbf{E}[\phi(A)] = \sum_{a_0 \in A} \frac{\rho^2(a_0, C)}{\sum_{a \in A} \rho^2(a, C)} \sum_{a \in A} \min\left\{\rho^2(a, C), \|a - a_0\|^2\right\}$$

Observation:

$$\begin{aligned}\rho^2(a_0, C) = \|a_0 - c_{a_0}\|^2 &\leq \|a_0 - c_a\|^2 \leq (\rho(a, C) + \|a - a_0\|)^2 \\ &\leq 2\rho^2(a, C) + 2\|a - a_0\|^2\end{aligned}$$

$$\rho^2(a_0, C) \leq \frac{2}{|A|} \sum_{a \in A} \rho^2(a, C) + \frac{2}{|A|} \sum_{a \in A} \|a - a_0\|^2$$

Approximation guarantee for *k*means++

Claim: Let A be a cell in induced by OPT. Let C be an arbitrary set of clusters. If we add a random center to C from A (with probabilistic farthest first weighting), then $\mathbf{E}[\phi(A)] \leq 8\,\phi_{OPT}(A)$

Proof:

$$\mathbf{E}[\phi(A)] = \sum_{a_0 \in A} \frac{\rho^2(a_0, C)}{\sum_{a \in A} \rho^2(a, C)} \sum_{a \in A} \min\left\{\rho^2(a, C), \|a - a_0\|^2\right\}$$

$$\leq \frac{2}{|A|} \sum_{a_0 \in A} \frac{\sum_{a \in A} \rho^2(a, C)}{\sum_{a \in A} \rho^2(a, C)} \sum_{a \in A} \min\left\{\rho^2(a, C), \|a - a_0\|^2\right\}$$

$$+ \frac{2}{|A|} \sum_{a_0 \in A} \frac{\sum_{a \in A} \|a - a_0\|^2}{\sum_{a \in A} \rho^2(a, C)} \sum_{a \in A} \min\left\{\rho^2(a, C), \|a - a_0\|^2\right\}$$

$$\leq \frac{4}{|A|} \sum_{a_0 \in A} \sum_{a \in A} \|a - a_0\|^2 \qquad = 8\phi_{\mathrm{OPT}}(A)$$

Approximation guarantee for *k*means++

Shown so far:

- Picking the first center (uar) increases the cost by a factor of ≤ 2
- Picking subsequent centers (pff) increases the cost by a factor of ≤ 8

But… our sampling may not hit each OPT cell!

Approximation guarantee for *k*means++

Claim: Let C be some clustering. Pick u > 0 be uncovered cells from OPT, and X_u be the corresponding points from these cells. Suppose we add $t \leq u$ clusters (with pff sampling). Let C' be the resulting clustering. Then,

$$\mathbf{E}[\phi'] \leq \left(\phi(X_c) + 8\,\phi_{OPT}(X_u)\right) . (1+H_t) + (u-t / u)\,\phi(X_u)$$

$X_c := X - X_u$
$H_t := \Sigma_{i \in t} (1/i)$

Claim $\Rightarrow$ Theorem

$$\mathbf{E}[\text{cost}(C)] \leq 8(\ln(k)+2)\,\text{cost}(\text{OPT})$$

why?

- Consider the clustering after the picking the first center (u.a.r.), let A be the corresponding partition.
- Using t = u = k – 1 and applying the claim

$$\mathbf{E}[\phi'] \leq \left(\phi(A) + 8\,\phi_{OPT} - 8\,\phi_{OPT}(A)\right) . (1+H_t)$$

- $H_{k-1} \leq 1 + \ln k$

■

Approximation guarantee for *k*means++

Claim: Let C be some clustering. Pick u > 0 be uncovered cells from OPT, and X_u be the corresponding points from these cells. Suppose we add $t \leq u$ clusters (with pff sampling). Let C' be the resulting clustering. Then,

$$\mathbf{E}[\phi'] \leq \left(\phi(X_c) + 8\,\phi_{OPT}(X_u)\right) . (1+H_t) + (u-t / u)\,\phi(X_u)$$

$X_c := X - X_u$
$H_t := \Sigma_{i \in t} (1/i)$

Proof: will show by induction: (t-1,u) and (t-1,u-1) $\Rightarrow$ (t,u)

Base cases:

(t=0,u>0) $\quad \mathbf{E}[\phi'] = \phi = \phi(X_c) + \phi(X_u)$

(t=1,u=1)

If *t* was picked from the uncovered cell… happens with prob $\phi(X_u) / \phi$

$$\mathbf{E}[\phi'] \leq \phi(X_c) + 8\,\phi_{OPT}(X_u)$$

If *t* was picked from already covered cells… happens with prob $\phi(X_c) / \phi$

So, $\quad \mathbf{E}[\phi'] \leq (\phi(X_u)/\phi)\left(\phi(X_c) + 8\,\phi_{OPT}(X_u)\right) + (\phi(X_c) / \phi)\,\phi$

$$\leq 2\,\phi(X_c) + 8\,\phi_{OPT}(X_u)$$

base cases done

Approximation guarantee for *k*means++

Inductive case: (t-1,u) and (t-1,u-1) $\Rightarrow$ (t,u)

If the first center (of *t*) was picked from already covered cells, happens w.p. $(\phi(X_c)/\phi)$
The center can only reduce ϕ, now applying the IH on (t-1,u), its contribution to $\mathbf{E}[\phi']$

$$(\phi(X_c) / \phi) . [(\phi(X_c) + 8\, \phi_{OPT}(X_u)).(1+H_{t-1}) + (u-(t-1) / u)\, \phi(X_u)]$$

If the first center "a" (of *t*) was picked from an uncovered cell A, happens w.p. $(\phi(A)/\phi)$
Applying the IH on (t-1,u-1) as cell A is added to covered cells… contribution to $\mathbf{E}[\phi']$

$$(\phi(A)/\phi)[\Sigma_a\, p_a(\phi(X_c)+\phi(a) + 8\phi_{OPT}(X_u)-8\phi_{OPT}(A))(1+H_{t-1}) +(u-t)/(u-1))(\phi(X_u)-\phi(A))]$$

$$\leq \quad (\phi(A)/\phi) . [(\phi(X_c)+ 8\phi_{OPT}(X_u))(1+H_{t-1}) + (u-t)/(u-1))(\phi(X_u)-\phi(A))]$$

Combining the two cases and with a few approximations, yields the claim.

■

k-means Approximation

- kmeans++ seeding is log(*k*) optimal

  can also be shown that this analysis is tight

- How about other approximations?
  - Constant approximations are available…
    - $9 + \varepsilon$ via local swap algorithm [Kanungo et al. '04]
    - $1 + \varepsilon$ (but runtime exponential dependence on *k* or *d*)

      [Matousek '00, Feldman et al. '07, Friggstad '16]