

Clustering



Partially based on Nina Mishra's (@HP Lab) presentation slides

Edward Chang
Electrical Engineering
UC Santa Barbara

Outline

⌘ Motivation

⌘ Algorithms

☐ Inter-Intra

☐ K-Center

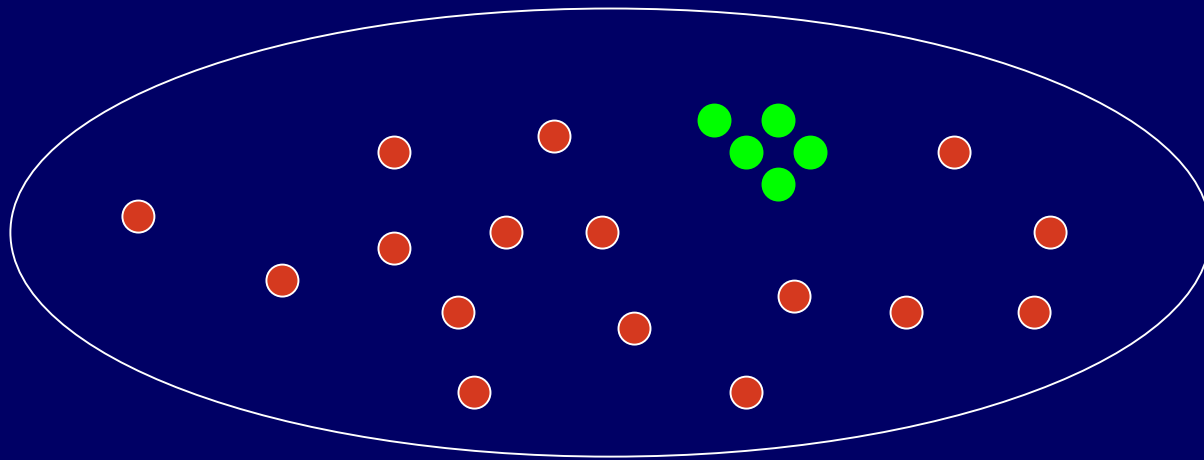
☐ K-median

☐ TSVQ

Foundation of Information Retrieval

Three Monumental Challenges

- ⌘ Formulating a "good" feature space
- ⌘ Formulating a perceptual distance function
- ⌘ Having an efficient clustering algorithm

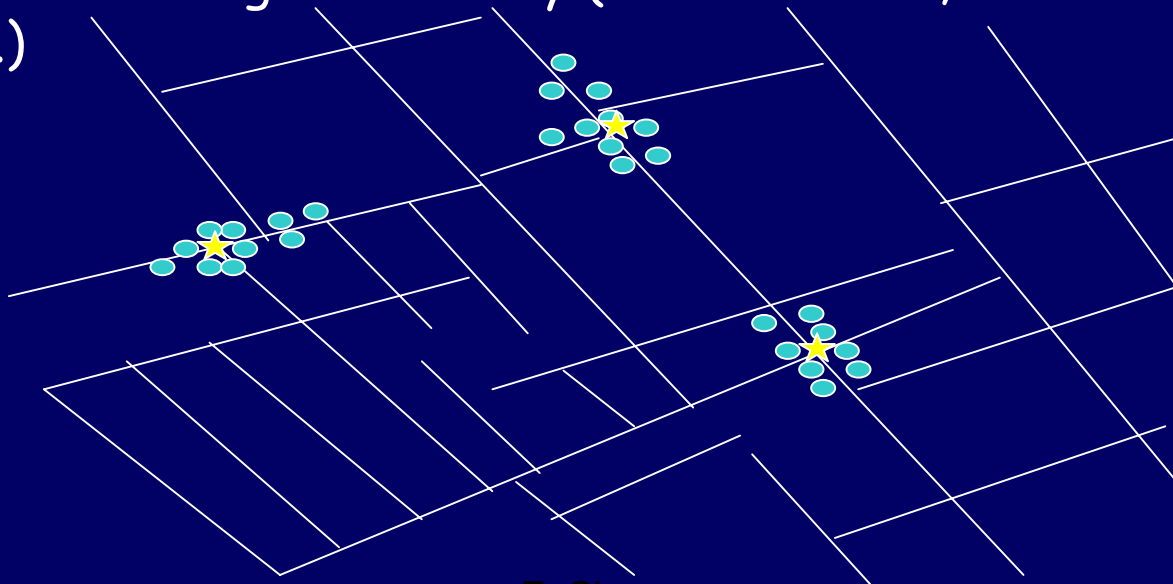


What Is Clustering ?

- ⌘ *Cluster*: a collection of objects that are "similar" to one another
- ⌘ Used either as
 - ⊞ a *stand-alone tool* to get insight into data distribution or
 - ⊞ as a *preprocessing step* for other algorithms, e.g., to discover classes, and associations between classes
- ⌘ *Clustering* is partitioning of data into meaningful groups called *clusters*.
 - ⊞ To help understand the natural grouping or structure in a data set.

The ?First? Application of Clustering

- ⌘ A London physician plotted the location of cholera cases on a map during an outbreak in the 1850s.
- ⌘ The locations indicated that cases were clustered around certain intersections where there were polluted wells -- thus exposing both the problem and the solution.
- ⌘ Not all clustering is this easy (2 dimensional, small number of points.)



Some Modern Applications of Clustering

⌘ *Operations Research*

- ☒ Facility Location Problem: locate fire stations so as to minimize the maximum/average distance a fire truck must travel

⌘ *Signal Processing*

- ☒ Vector Quantization: Transmit large files (e.g., video, speech) by computing quantizers

⌘ *Indexing*

- ☒ Grouping similar objects together to facilitate efficient similarity searches

...Applications of Clustering

⌘ *Marketing:*

- ☒ Segmentation of customers for target marketing
- ☒ Segmentation of customers based on online clickstream data

⌘ *Web*

- ☒ To discover categories of content
- ☒ Search results

⌘ *In practice, clustering is one of the most widely used data mining techniques*

- ☒ Association rule algorithms produce too many rules
- ☒ Other machine learning algorithms require labeled data

What we need for clustering?

⌘ In order to cluster, we need

- ☑ Data items (points, sequences)
- ☑ A distance function and
- ☑ A method for evaluating our clustering results

Points/Metric Space

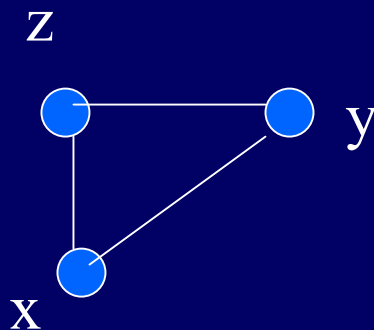
⌘ Points could be in $\mathbb{R}^d, \{0,1\}^d, \dots$

⌘ Metric Space: $\text{dist}(x,y)$ is a distance metric if

⊡ Reflexive: $\text{dist}(x,y)=0$ iff $x=y$

⊡ Symmetric: $\text{dist}(x,y)=\text{dist}(y,x)$

⊡ Triangle Inequality: $\text{dist}(x,y) \leq \text{dist}(x,z) + \text{dist}(z,y)$



Example of Distance Metrics

⌘ The distance between $x = \langle x_1, \dots, x_n \rangle$ and $y = \langle y_1, \dots, y_n \rangle$ is:

⊞ L2 norm: $\sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$

⊞ Manhattan Distance (L1 norm):

$$|x_1 - y_1| + \dots + |x_n - y_n|$$

⌘ Documents: Cosine measure

$$\cos \theta = \frac{\vec{x} \bullet \vec{y}}{\|\vec{x}\| \|\vec{y}\|}$$

⊞ Similarity, not distance

⊞ I.e., more similar \rightarrow close to 1

⊞ Less similar \rightarrow close to 0

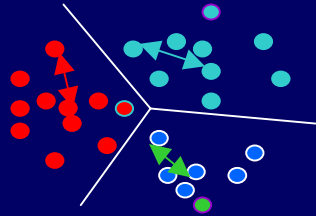
⊞ Not a metric space, but $1 - \cos \theta$ is

What is a Good Clustering?

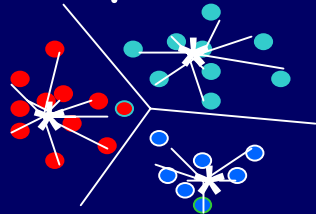
- ⌘ A *good clustering* method will produce clusters where
 - ⊡ the *intra-cluster distance* is *small*
 - ⊡ the *inter-cluster distance* is *large*

Clustering Quality Measures

- Given a set S of n points in R^d
 - *K-Center*: Find k centers such that the maximum radius of a cluster is minimized.

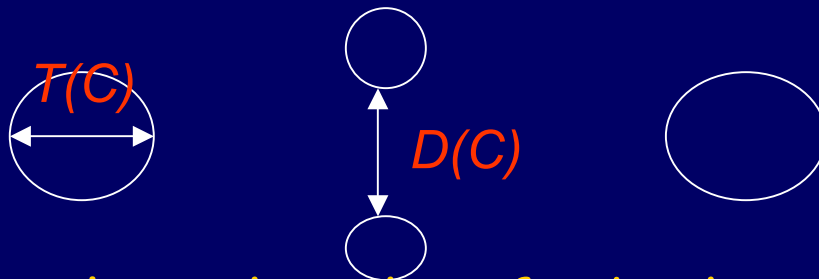


- *k-Median*: Find k centers that Minimize the Average Distance from a point to its nearest center



Inter-Intra

- ⌘ Distance between clusters is the minimum distance $d(x,y)$ for x and y in different clusters (aka "single-linkage") $D(C)$
- ⌘ Tightness of a clustering is the maximum diameter of any cluster $T(C)$.
- ⌘ Objective Function: $G(C)$
 - ⊞ Maximize (Distance between clusters - Tightness)

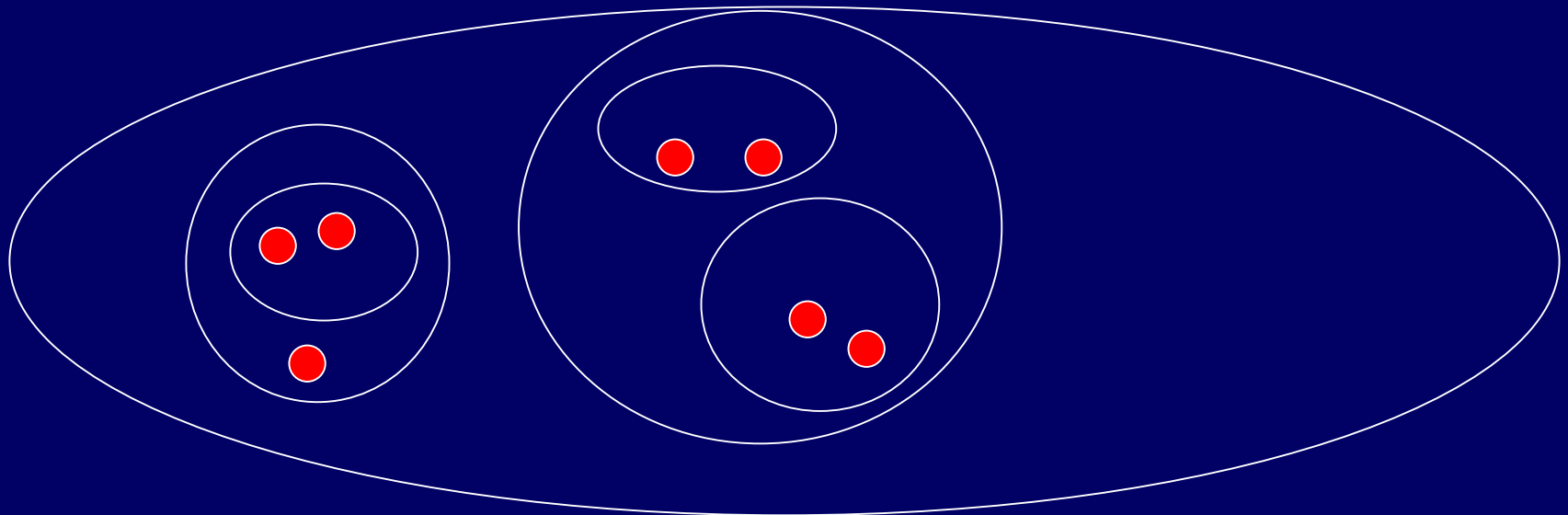


- ⌘ Next: An algorithm that finds the *optimum* solution
 - ⊞ Most clustering problems are NP-hard. So this is a rarity.

Hierarchical Agglomerative Clustering (HAC)

⌘ Standard clustering approach

- ⌘ Initially, all points are in own clusters
- ⌘ Merge two closest clusters
- ⌘ Repeat



HAC

⌘ The Algorithm (More Specific)

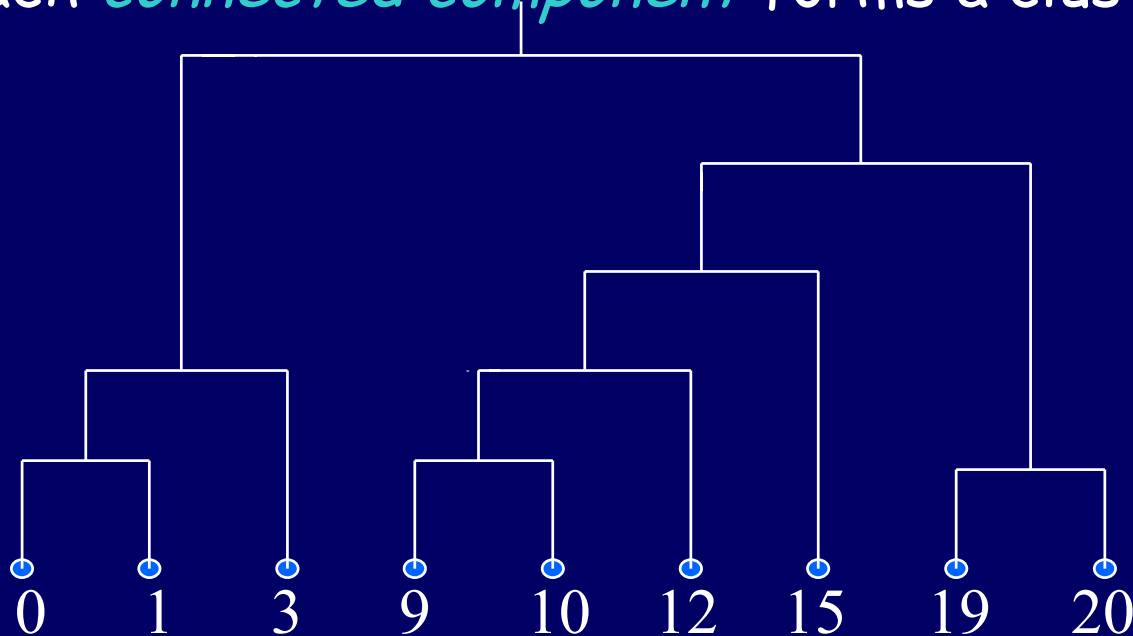
- ☒ Start with each point in a cluster by itself
- ☒ Repeatedly merge the two clusters that are closest until there is just one cluster, where

$$\text{dist} (C_1, C_2) = \min_{p \in C_1, q \in C_2} \text{dist} (p, q)$$

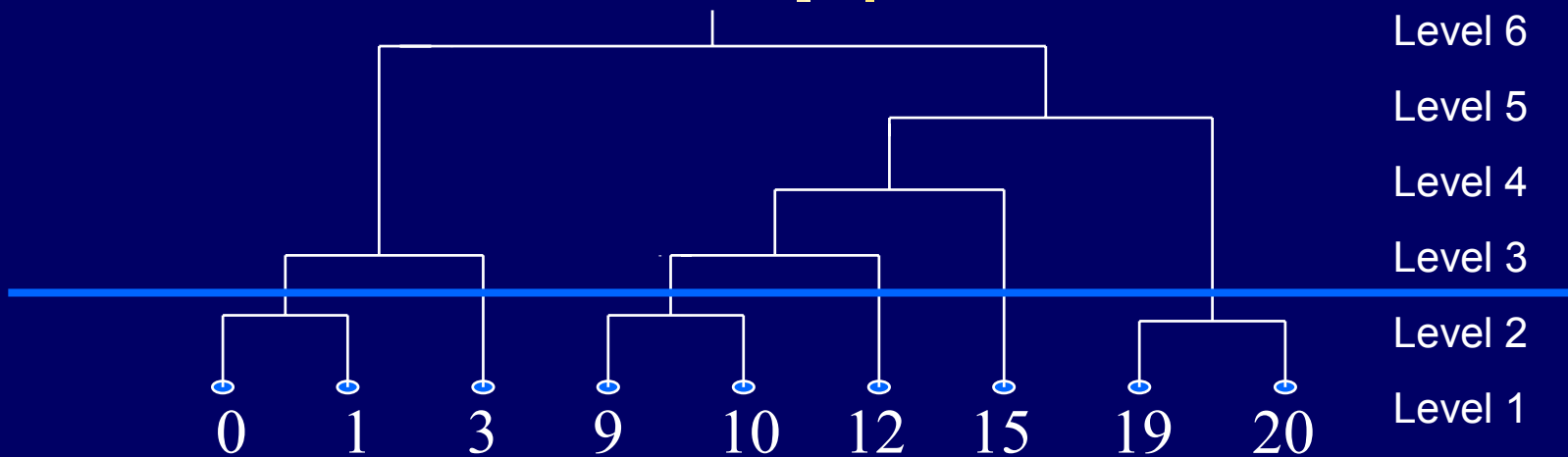
- ☒ *Output the best clustering* generated in this process, i.e., output the clustering that maximizes $D(C) - T(C)$.

Dendrogram

- ⌘ A *Dendrogram* shows how the clusters are merged hierarchically
- ⌘ A *clustering* of the data objects is obtained by *cutting* the dendrogram at the desired level, then each *connected component* forms a cluster



Dendrogram

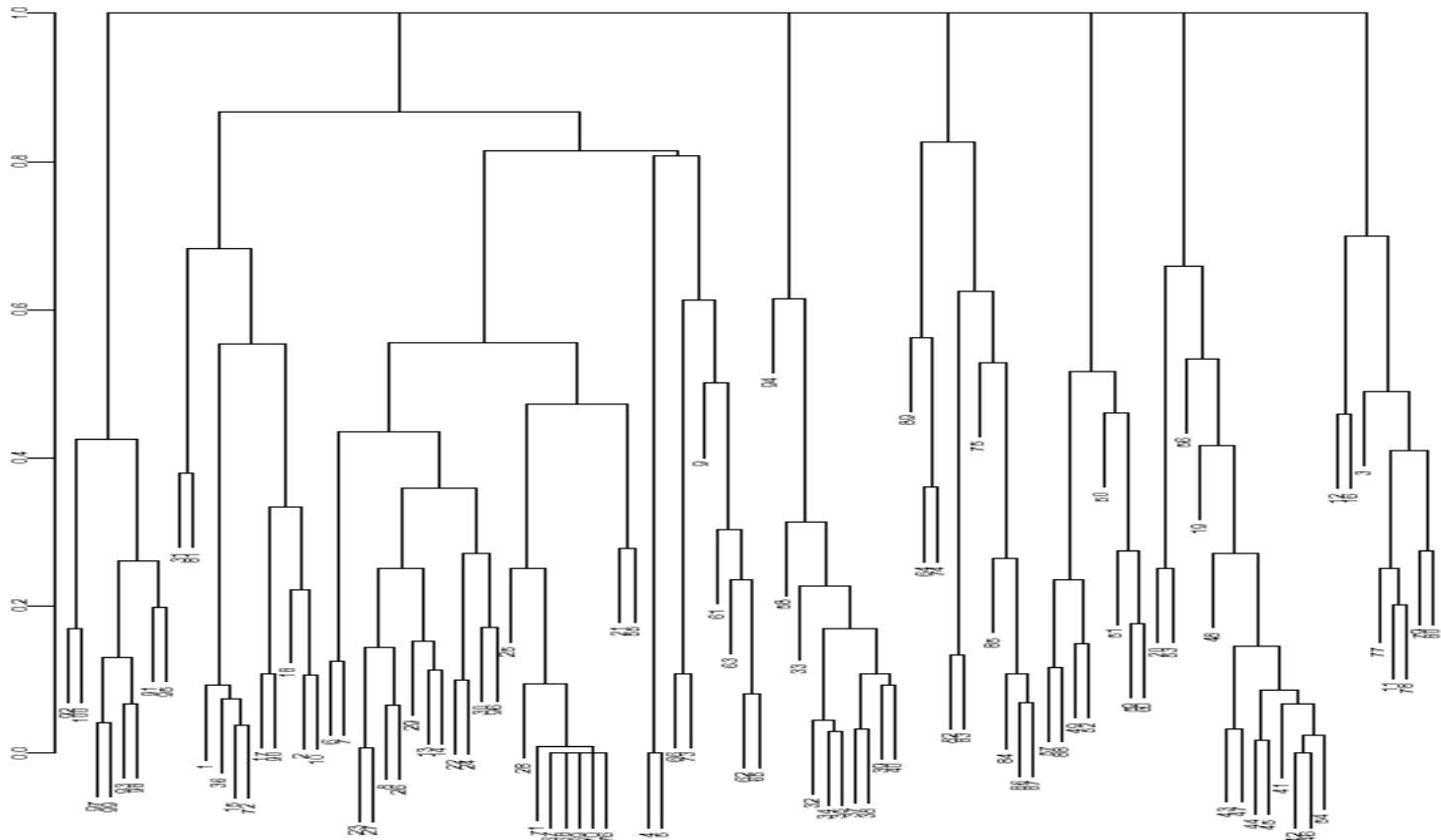


Level 1: $D - T = 2.5$

Level 2: $D - T = 8.4 - 0.5 = 7.9$

Level 3: $D - T = 8.0 - 1.75 = 6.25$

HAC still works..



Single Link Agglomerative

⌘ Use maximum similarity of pairs:

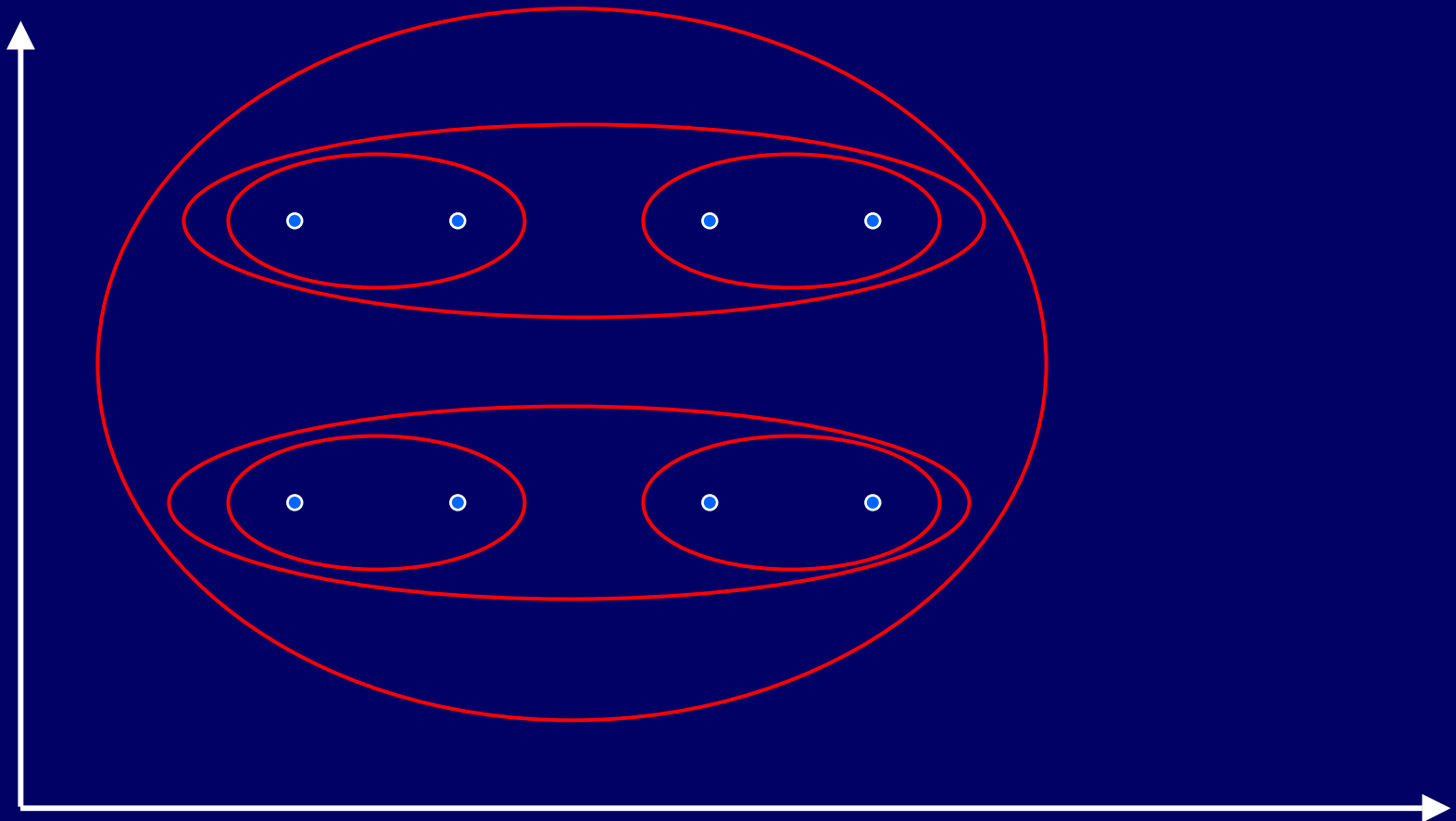
$$\text{sim}(c_i, c_j) = \max_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

⌘ Similarity = 1/distance

⌘ Can result in “straggly” (long and thin) clusters due to *chaining effect*.

☑ Appropriate in some domains, such as clustering islands.

Single Link Example



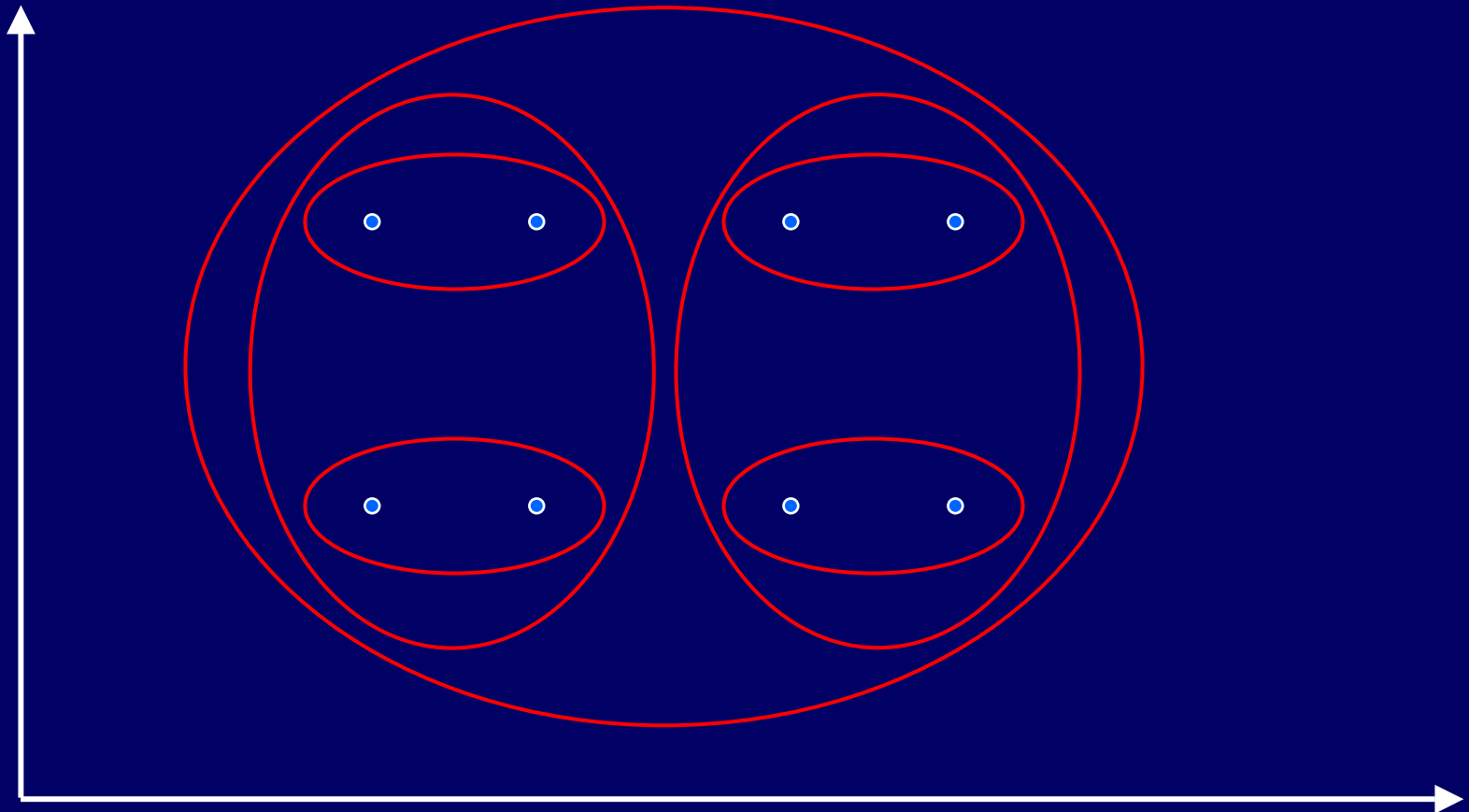
Complete Link Agglomerative

⌘ Use minimum similarity of pairs:

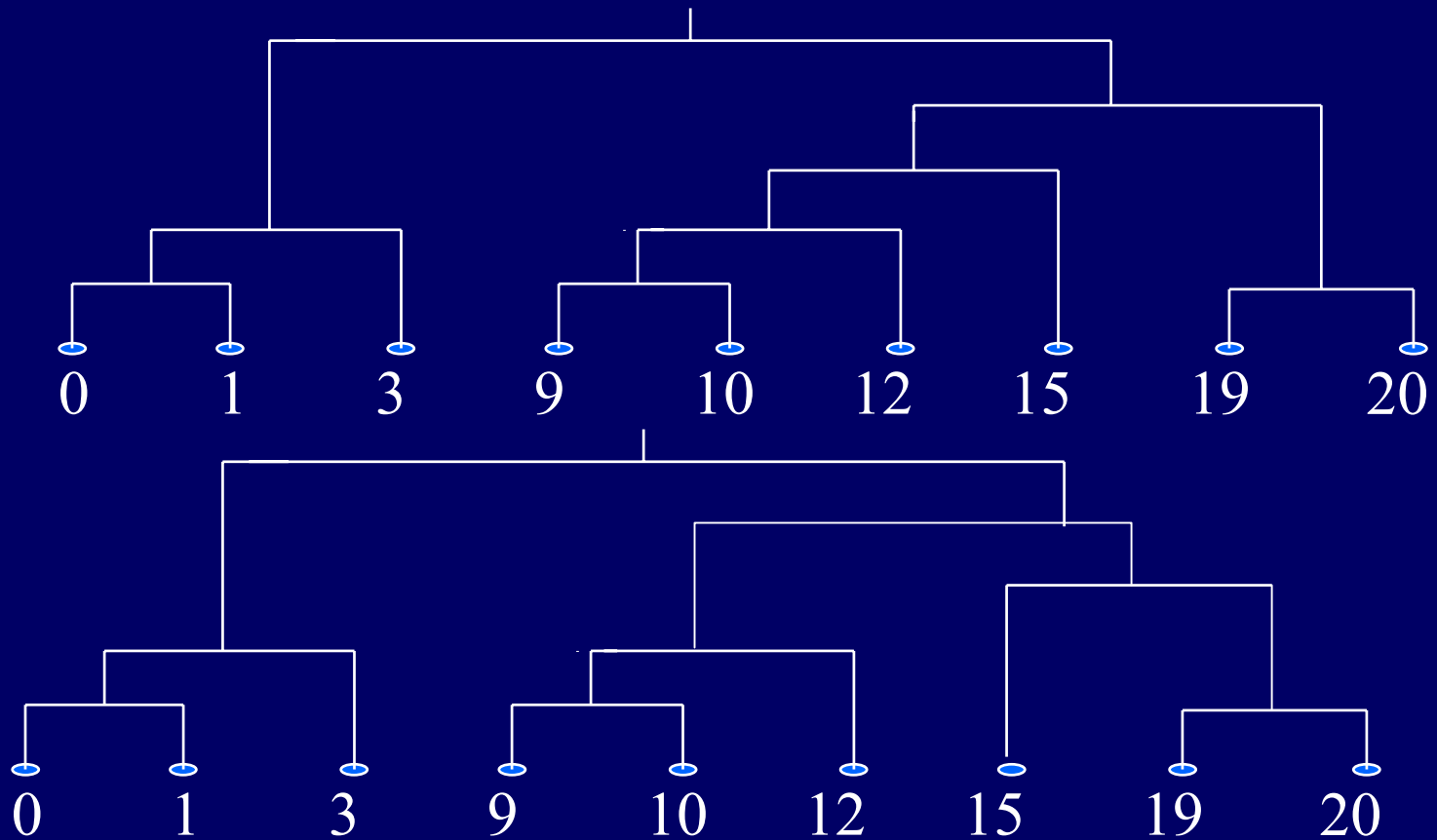
$$sim(c_i, c_j) = \min_{x \in c_i, y \in c_j} sim(x, y)$$

⌘ Makes more "tight," spherical clusters that are typically preferable.

Complete Link Example



Dendrogram



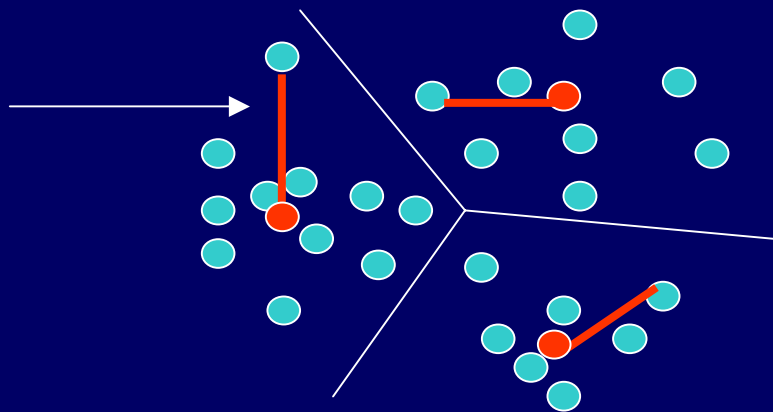
Good Things about HAC

- ⌘ Combines Inter-Intra
- ⌘ Don't need to know number of clusters k
- ⌘ However, if there is an optimum solution with k clusters, HAC might find more or less than k .
 - ☑ Finding optimum solution with minimum number of clusters k is NP-hard

Problems with HAC

- ⌘ *Running Time:* $O(n^2 * |\text{levels}|) = O(n^3)$
- ⌘ Imagine doing this for the 2-4 billion documents on the web.
- ⌘ Open question: How to optimize Inter-Intra on **a large data set** or **data stream**?
- ⌘ *Next:* Change the objective function
 - ⊞ Will not be as "general"
 - ⊞ Is very simple to state
 - ⊞ But can cluster large data sets/data streams

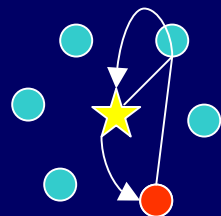
k-Center Problem Definition



Given n points in a metric space,
find k centers
so as to minimize the maximum distance
from a point to its nearest center (radius)

Discrete vs. Continuous k-Center

- ⌘ Suppose points in \mathbb{R}^d .
- ⌘ k centers in S (discrete) or \mathbb{R}^d (continuous) ?
- ⌘ *Claim*: The best Discrete solution is within a factor of 2 of the best Continuous solution.
- ⌘ *Proof*: (skip)
 - ⌘ Start with the best continuous solution
 - ⌘ Pick the points in S closest to the continuous solution
 - ⌘ Now bound radius



$$\text{dist}(\text{red circle}, \text{blue circle}) \leq \text{dist}(\text{blue circle}, \text{yellow star}) + \text{dist}(\text{yellow star}, \text{red circle}) \leq 2\text{OPT}$$

- ⌘ Assume Discrete k-Center from now on.

Naive Algorithm

The Naive Algorithm: Enumerate all possible k -clusterings and output the one that optimizes the clustering metric.

- This algorithm is inefficient because there are roughly k^n different k -clusterings of n points!
- So, naive won't cut it

k-Center is NP-hard

⌘ *Claim:* Dominating Set \leq k-Center

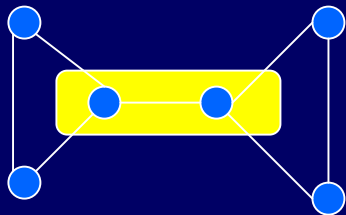
⌘ Dominating Set

⌘ Given a graph $G=(V,E)$ and a number k

⌘ Question: Does there exist $S \subset V$ such that

⌘ each vertex v is either in S or adjacent to S

⌘ $|S| \leq k$

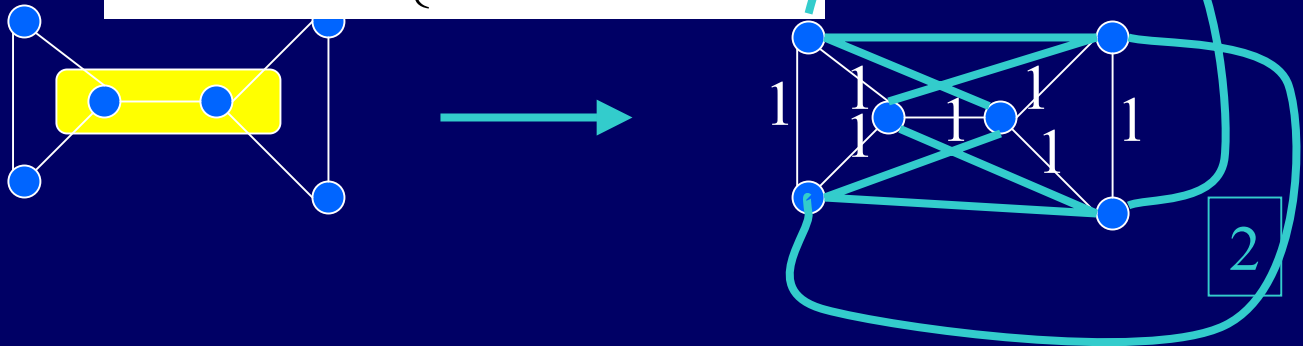


2-Dominating Set

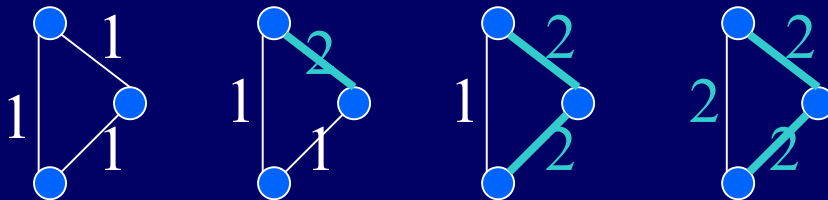
k-Center is NP-hard

⌘ Transform: G to a set of points $P=V$
 where

$$\text{dist}(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 2 & \text{otherwise} \end{cases}$$



• Note: Triangle inequality is preserved.



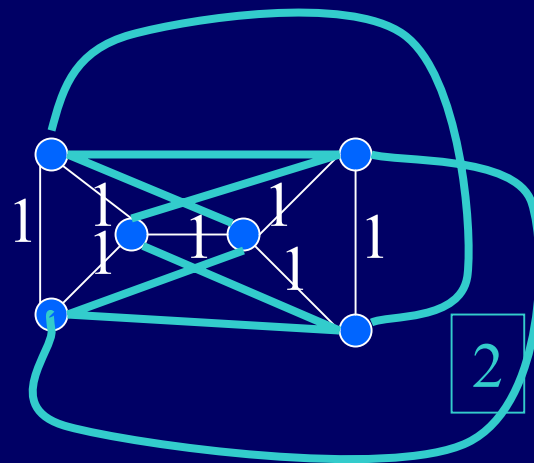
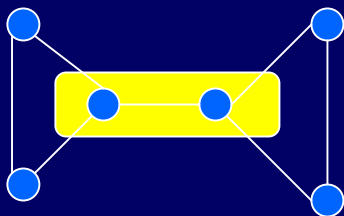
• Note 2 is the maximum allowable number if want to preserve triangle inequality

K-Center is NP-hard, contd

Claim: G has a k dominating set



P has a k -center clustering of cost 1



Proof:

→ If \exists k dominating set then \exists k center clustering with cost 1

← If no k dominating set then any k centers have cost 2

Approximation Algorithm

- ⌘ Unlikely that there is a poly-time algorithm for finding the k best centers.
- ⌘ Can we find k *"close to best"* centers?

$$\frac{\text{Cost of } k \text{ centers we find}}{\text{Cost of } k \text{ optimum centers}} \leq c$$

- ⌘ c -approximation algorithm.
- ⌘ c is some small constant
- ⌘ *Next:* Show a simple 2-approximation algorithm [Gonzalez]

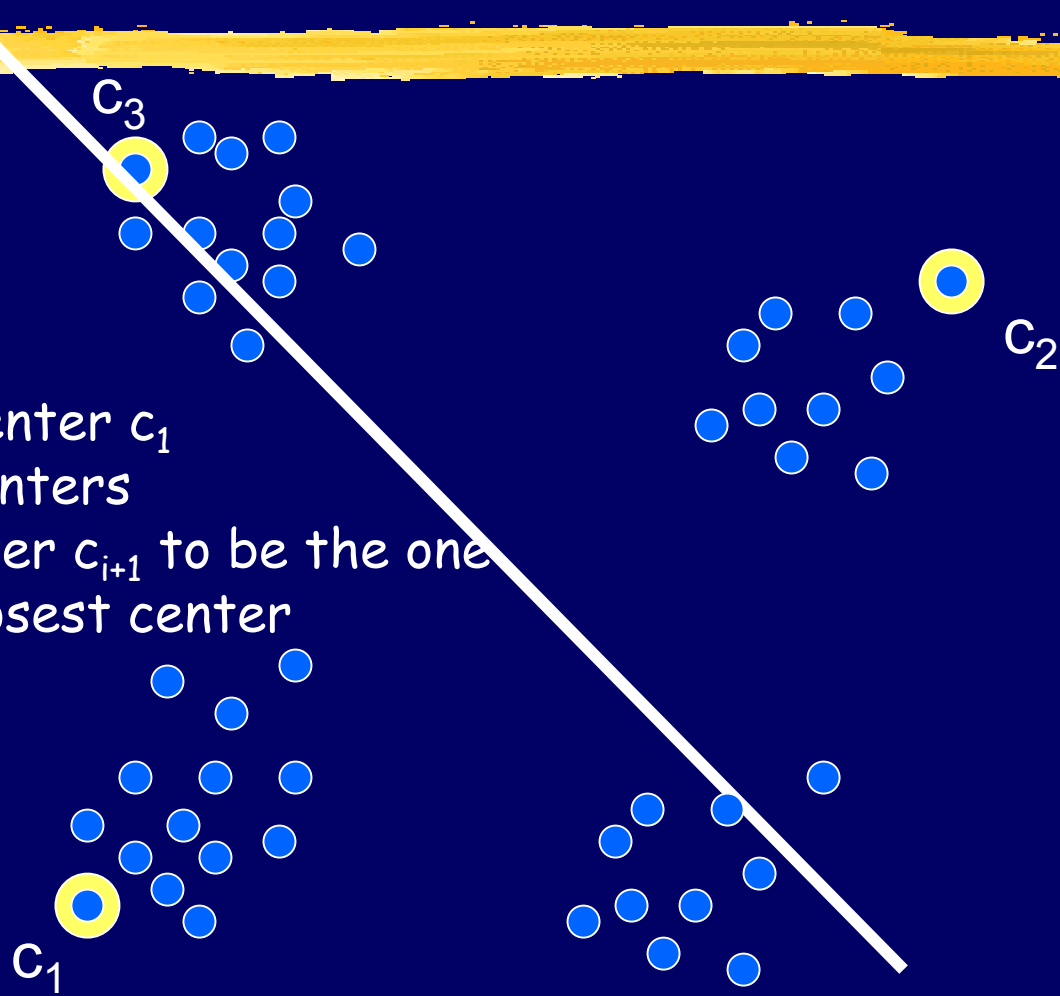
Farthest Point Clustering Algorithm, $k=4$

ALG:

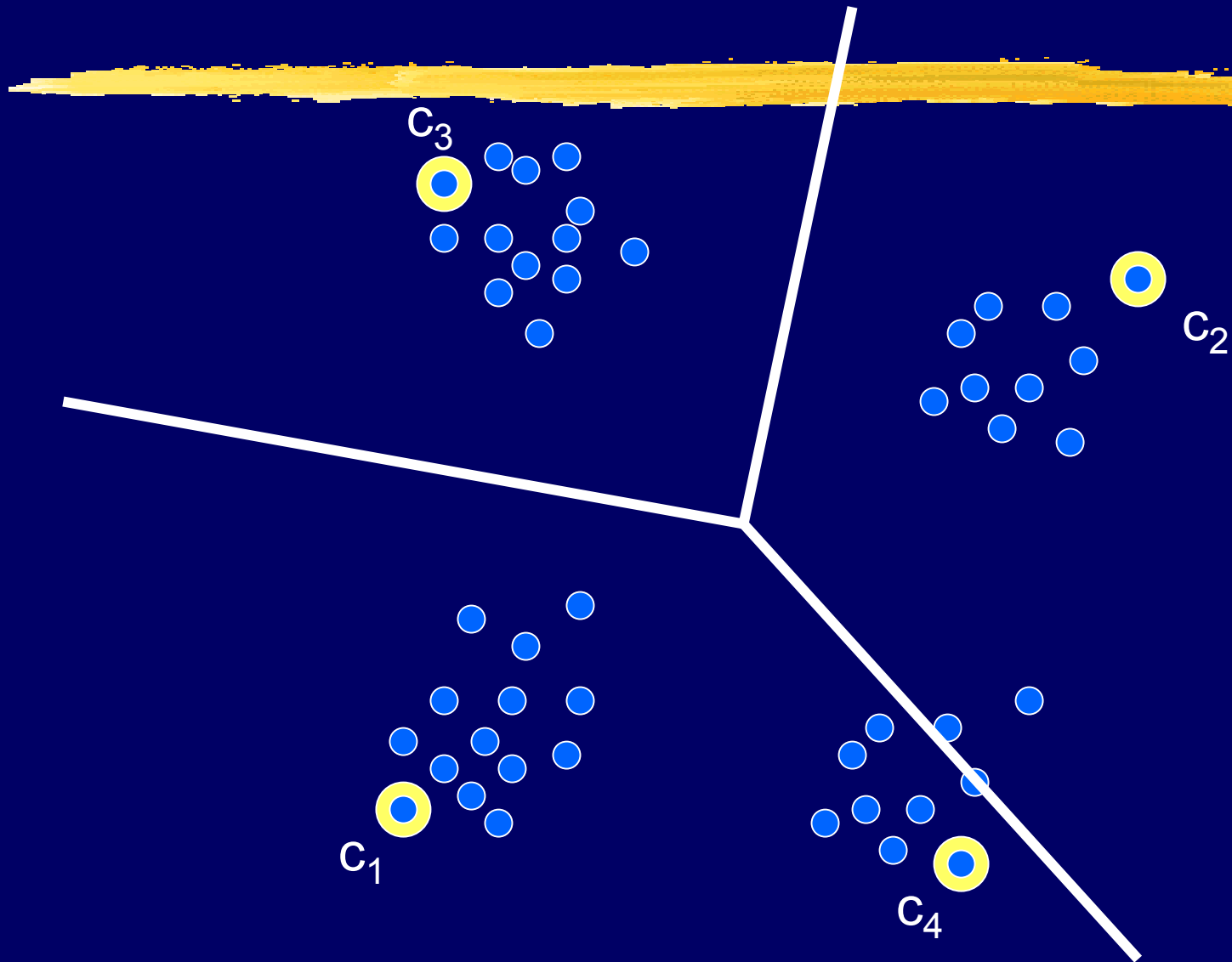
Select an arbitrary center c_1

Repeat until have k centers

Select the next center c_{i+1} to be the one farthest from its closest center



Farthest Point Clustering Algorithm, $k=4$

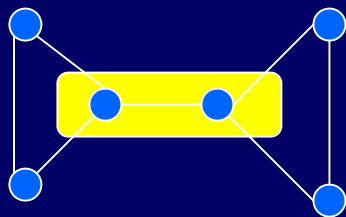


Can we do better than a 2 approximation?

⌘ Claim: There is no poly time algorithm that obtains a $(2-\epsilon)$ approximation to k -Center, unless $P=NP$.

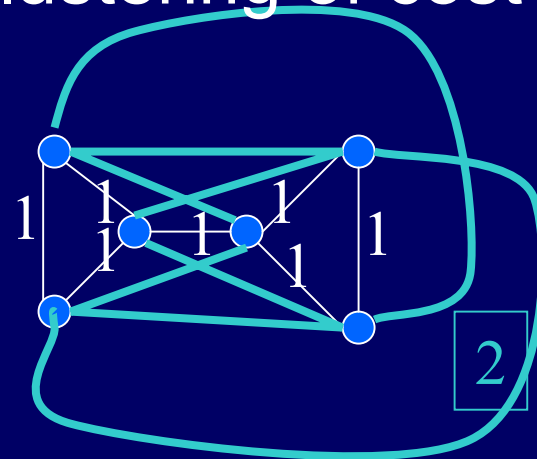
⊠ A $(2-\epsilon)$ approximation to k -Center can solve k -Dominating Set

Claim: G has a k dominating set



\leftrightarrow

P has a k -center clustering of cost 1

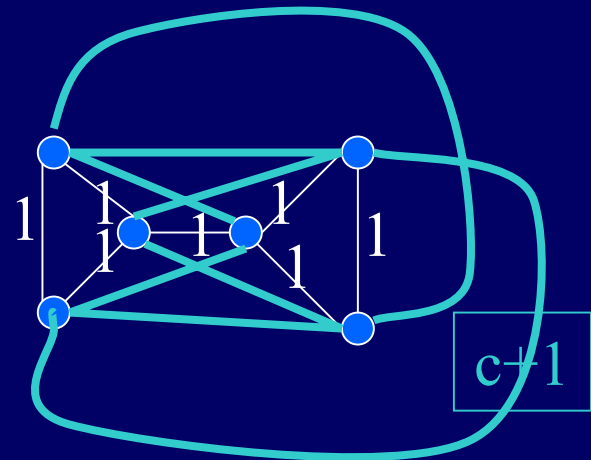
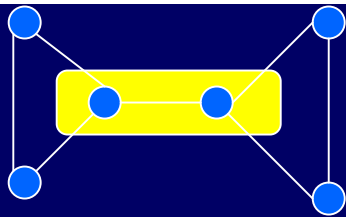


How important is metric space assumption?

⌘ Claim: Without metric assumption, k -center cannot be approximated to any constant c .

⌘ Transform: G to a set of points $P=V$ where

$$\text{dist}(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ c+1 & \text{otherwise} \end{cases}$$



- We've lost triangle inequality.
- A c -approximation to k -center implies an algorithm that can distinguish between k dominating set (cost between 1 and c) or not (cost between $c+1$ and $c(c+1)$).

Running Time

- ⌘ Running Time: $O(nk)$.
- ⌘ k passes through the data set, each pass takes $O(n)$ time.
- ⌘ What if our data is too large to fit in main memory?

Summary

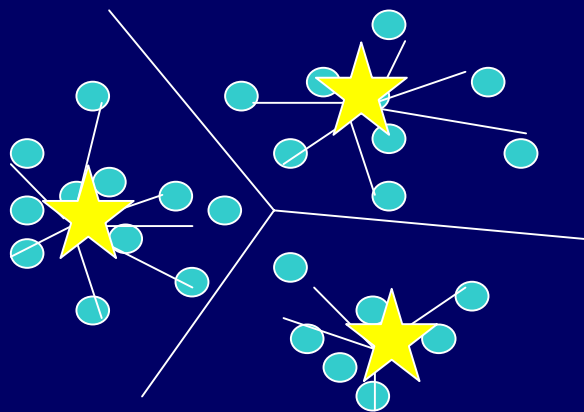
⌘ Maximizing Inter-Intra Cluster Quality

- ☑ Conditions under which HAC Algorithm finds optimum solution in polynomial time
- ☑ Algorithm does not scale

⌘ K-Center

- ☑ NP-hard
- ☑ Farthest Point Algorithm yields 2-approximation

K-Median Problem Definition



Given n points in a metric space,
choose k medians

so as to minimize the assignment cost:
the sum of (squared) distances from
points to nearest centers.

Single-Pass Algorithm

⌘ 0,1,3,9,10,12,15,19,20

⌘ Threshold = 2

⌘ Sort the data, if have not been

⌘ {0}, {0,1}, ({0,1}, {3}) ({0,1},{3},{9})

{0,1},{3},{9,10})...

{0,1},{3},{9,10},{12},{15},{19,20}

Single-Pass Algorithm

	T1	T2	T3	T4	T5
#1	1	2	0	0	1
#2	3	1	2	3	0
#3	3	0	0	0	1
#4	2	1	0	3	0
#5	2	2	1	5	1

Threshold setting is most critical

Outline

⌘ Motivation

⌘ Algorithms

☐ Inter-Intra

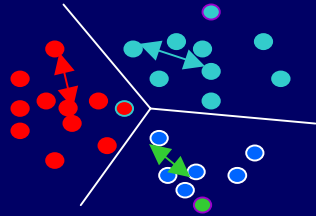
☐ K-Center

☐ K-median

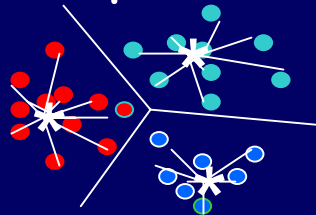
☐ TSVQ

Clustering Quality Measures

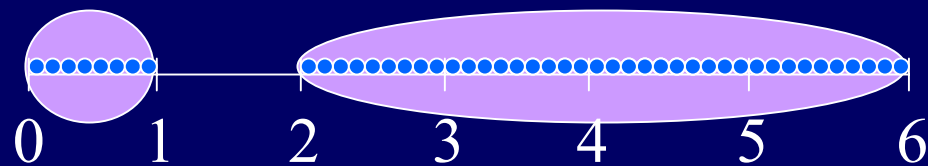
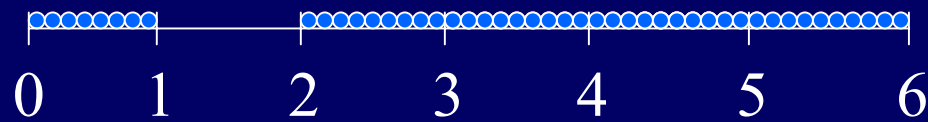
- Given a set S of n points in R^d
 - *K-Center*: Find k centers such that the maximum radius of a cluster is minimized.



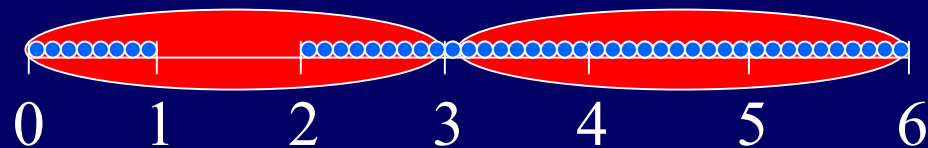
- *k-Median*: Find k centers that Minimize the Average Distance from a point to its nearest center



K-Center not always the "right" clustering measure



Max radius = 2

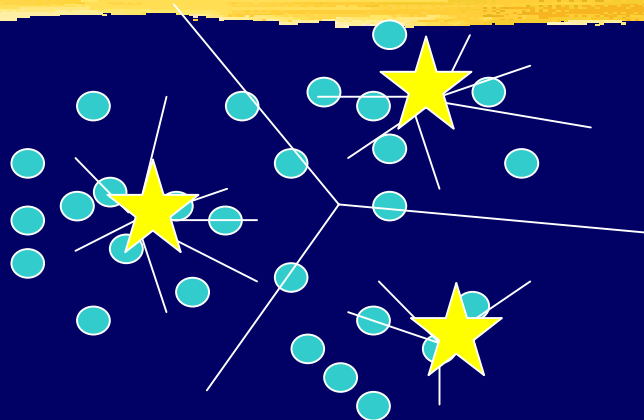


Max radius = 1.5

The K-Median Problem = Facility Placement Problem

- ⌘ We want the facilities to be as efficient as possible, thus we want to minimize the distance from each client to its closest facility.
- ⌘ There can be a cost associated with creating each facility that also must be minimized
 - ☑ Otherwise if we were not limited to k facilities, all points could be facilities

K-Median Problem Definition



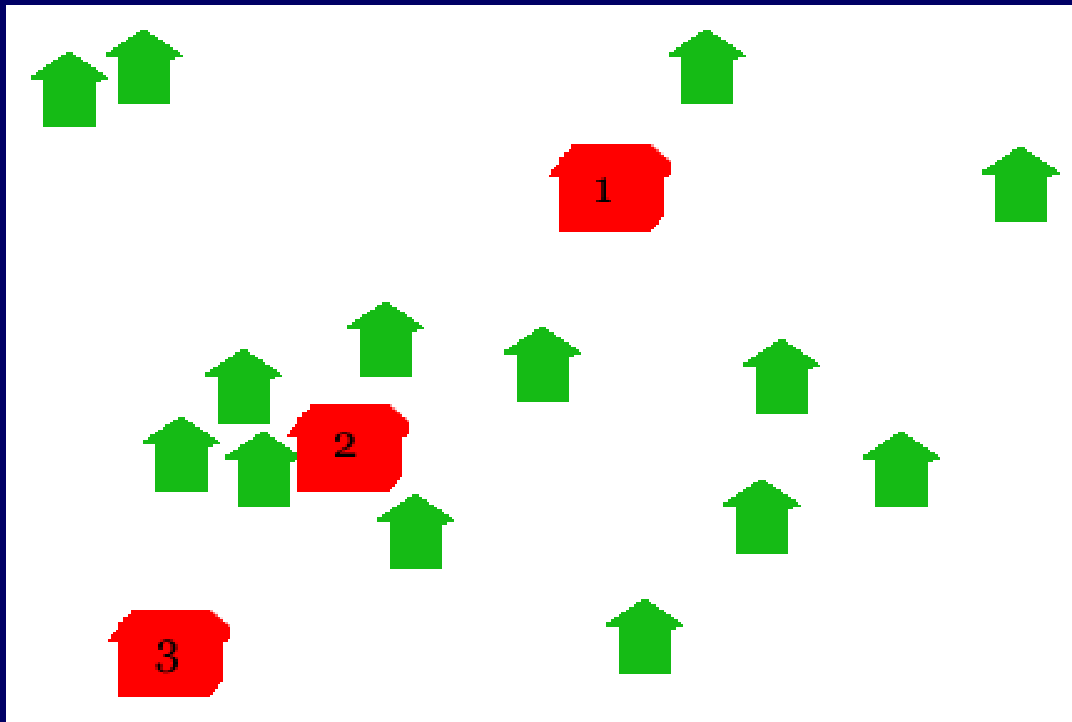
Given D a set of n points in a metric space
choose k medians

so as to minimize the assignment cost:
the sum of (squared) distances from
points to nearest centers.

$$\sum_x \min_i \text{dist}(x, c_i)$$

Local Search / K-Median

Where do we place our k facilities?

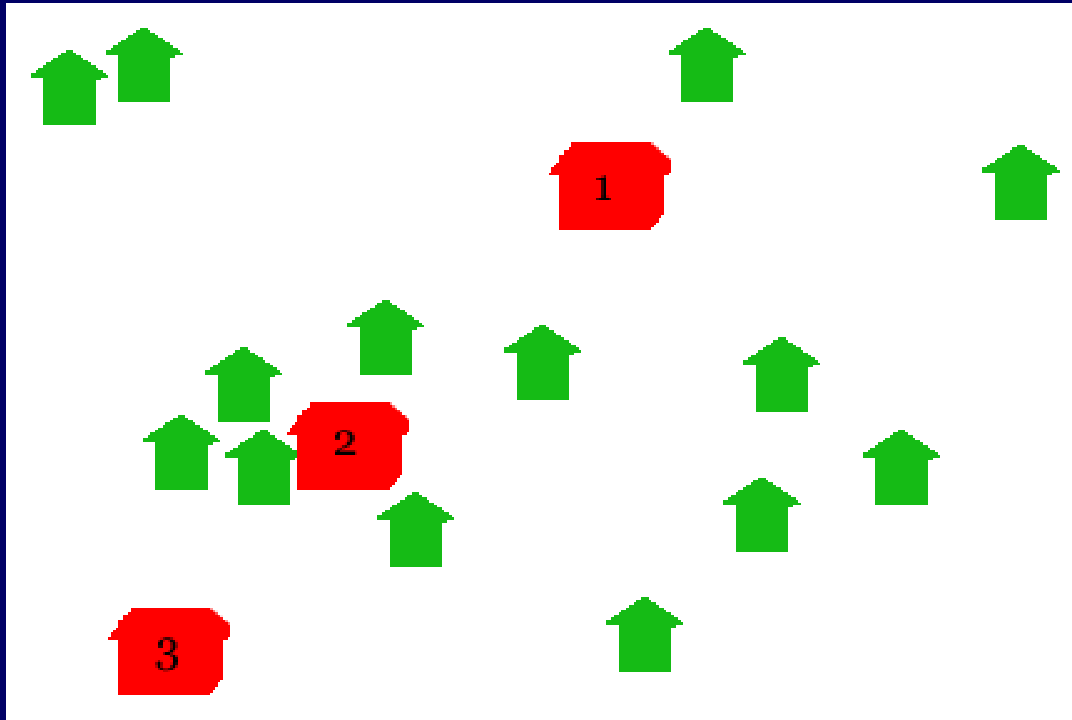


The Algorithm:

Choose some initial K points to be facilities, and calculate your cost

Initial points can be chosen by first choosing a random point, then successively choosing the point farthest from the current group of facilities until you have your initial K

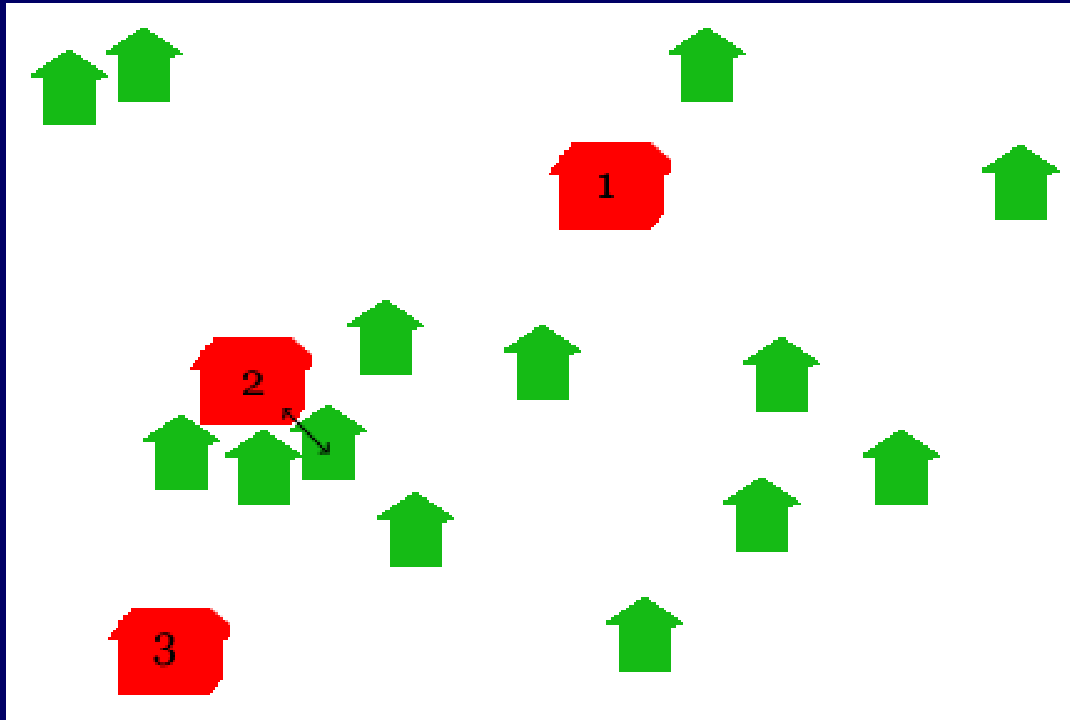
Local Search / K-Median



Now we swap

While there exists a swap between a current facility location and another vertex which improves our current cost, execute the swap

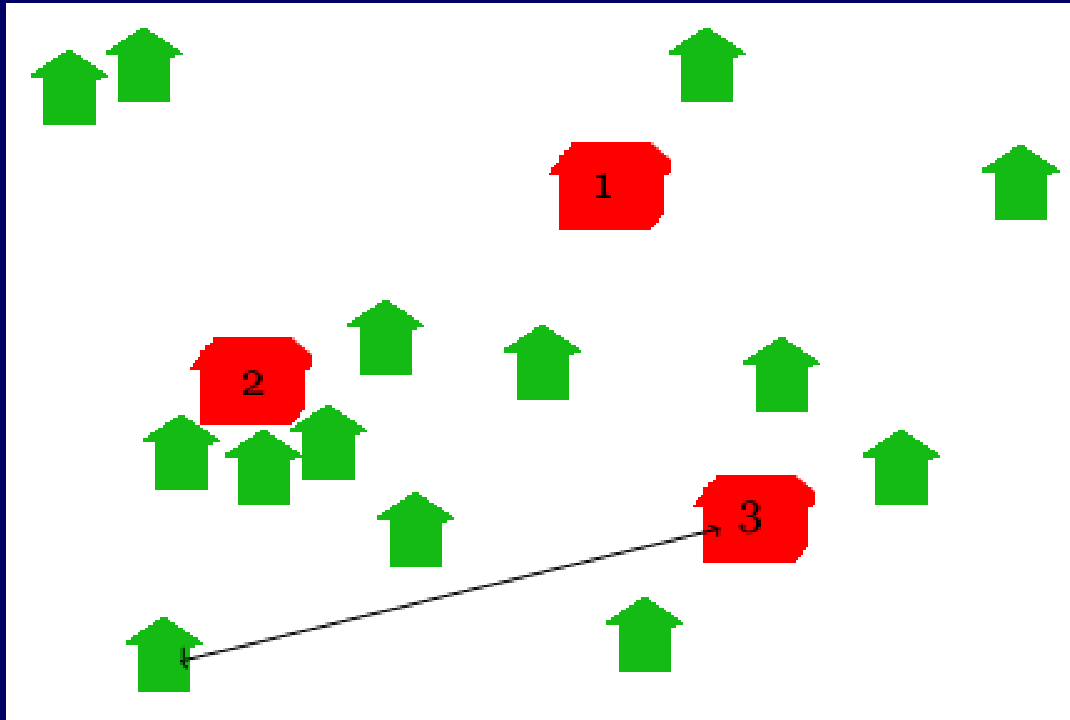
Local Search / K-Median



Now we swap

While there exists a swap between a current facility location and another point which improves our current cost, execute the swap

Local Search / K-Median



Now we swap

While there exists a swap between a current facility location and another point which improves our current cost, execute the swap

Etc.

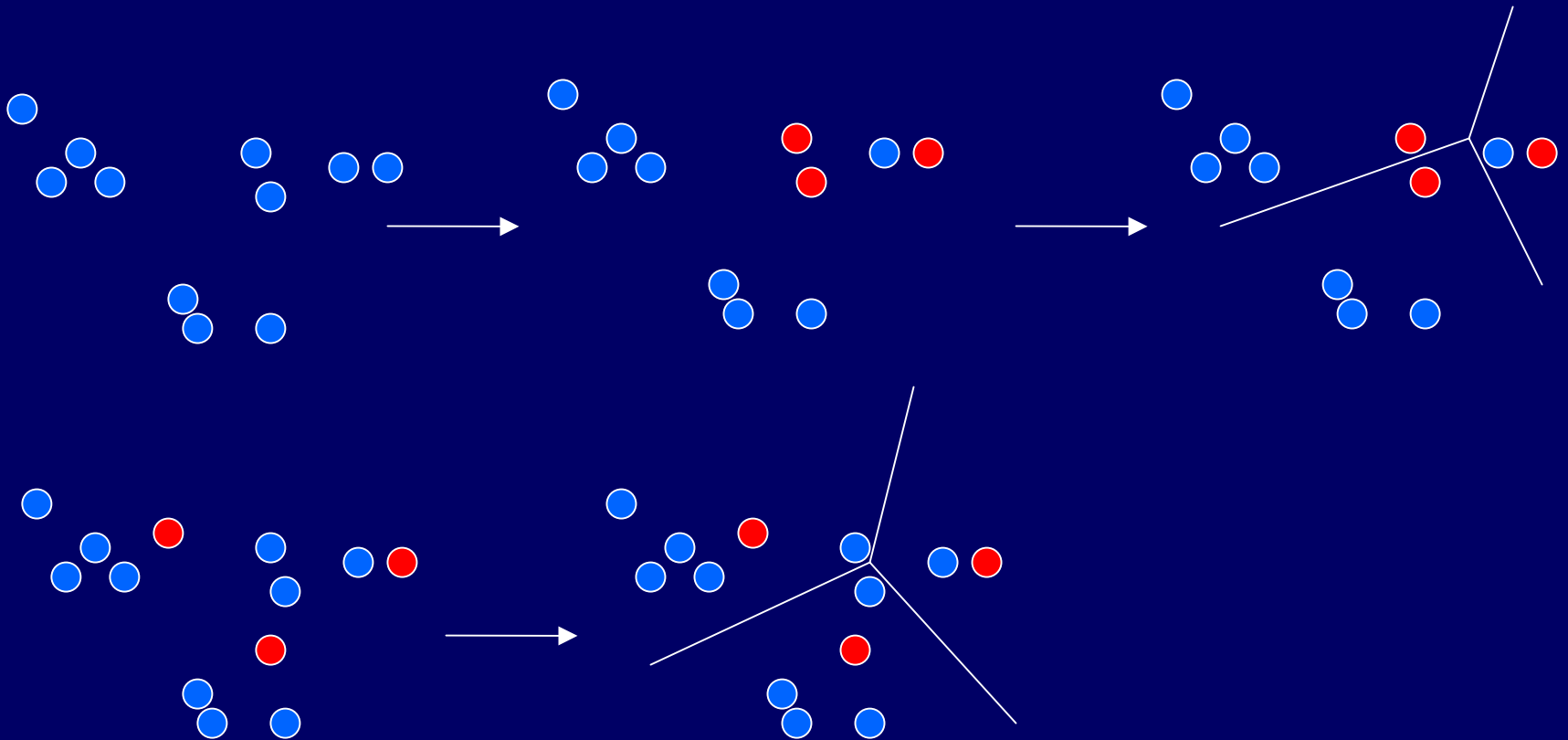
Local Search / K-Median

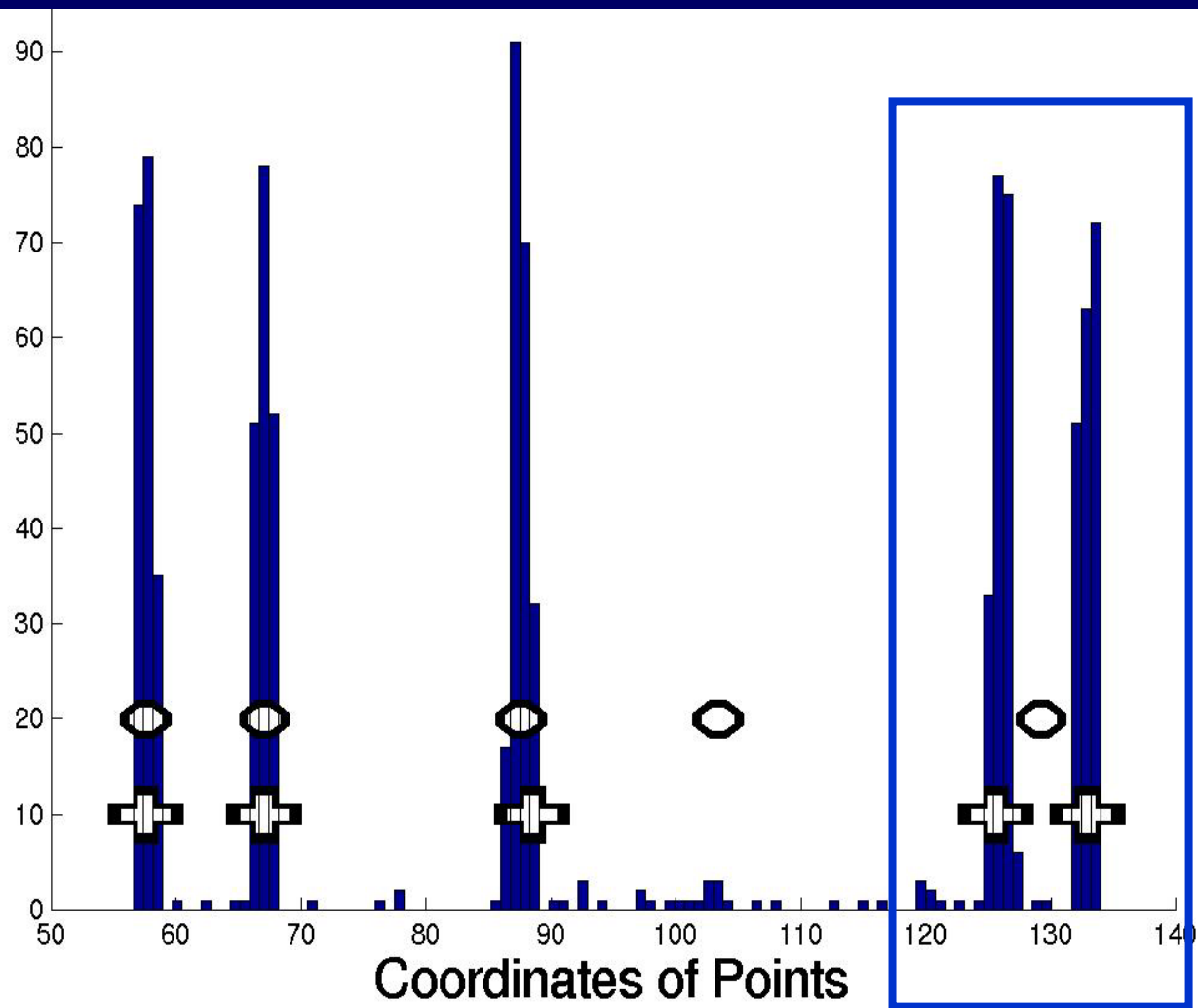
- ⌘ It is possible to do multiple swaps at one time
- ⌘ In the worst case, this solution will produce a total cost of $(3 + 2/p)$ times the optimal cost, where p is the number of swaps that can be done at one time

K-Means Algorithm [MacQueen, 1967]

- ⌘ Select k centers somehow (e.g., choose k points in \mathbb{R}^d)
- ⌘ Repeat until k centers don't change (or change very little)
 - ⌘ Partition the data according to the k centers
 - ⌘ Use the means of the cluster to find k new centers

K-Means

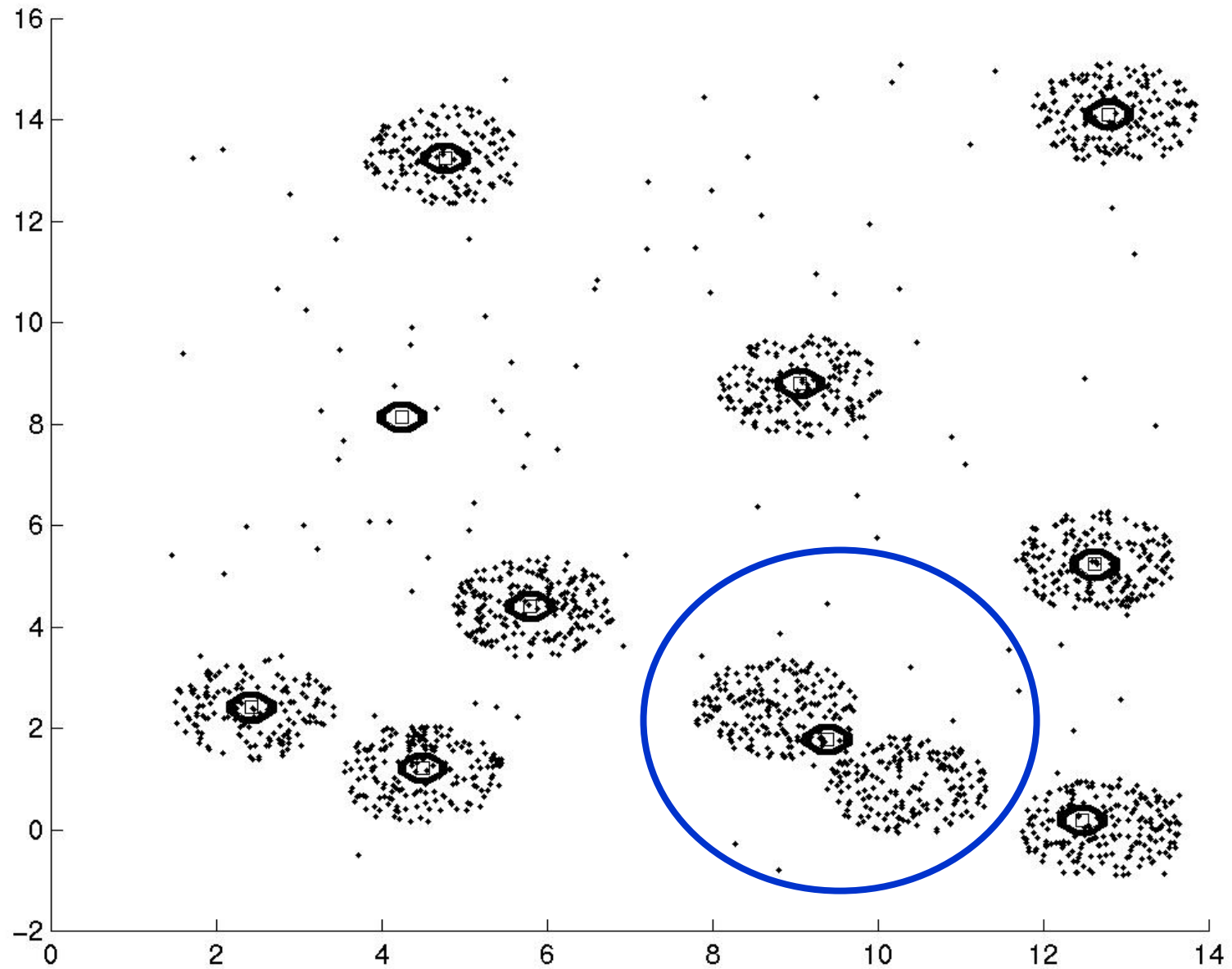




← K-means

← OPT

Best k-Means Centers



K-Means Issues

- ⌘ A dense local area can trap a center
- ⌘ Clustering quality depends on initial K points
 - ⌘ How do we know K ?

More Generally

$$\text{⌘ Perf}(X, M) = \sum_{x \in X} d(x, M)$$

⌘ K-Means

$$\text{⌘ } d(x, M) = \min \{ ||x - m||^2 \mid m \in M \}$$

⌘ EM

$$\text{⌘ } d(x, M) = -\log \left(\sum_{m \in M} p_m \times G_m(x) \right)$$

$$\text{⌘ } G_m(x) = \exp(-||x - m||^2)$$

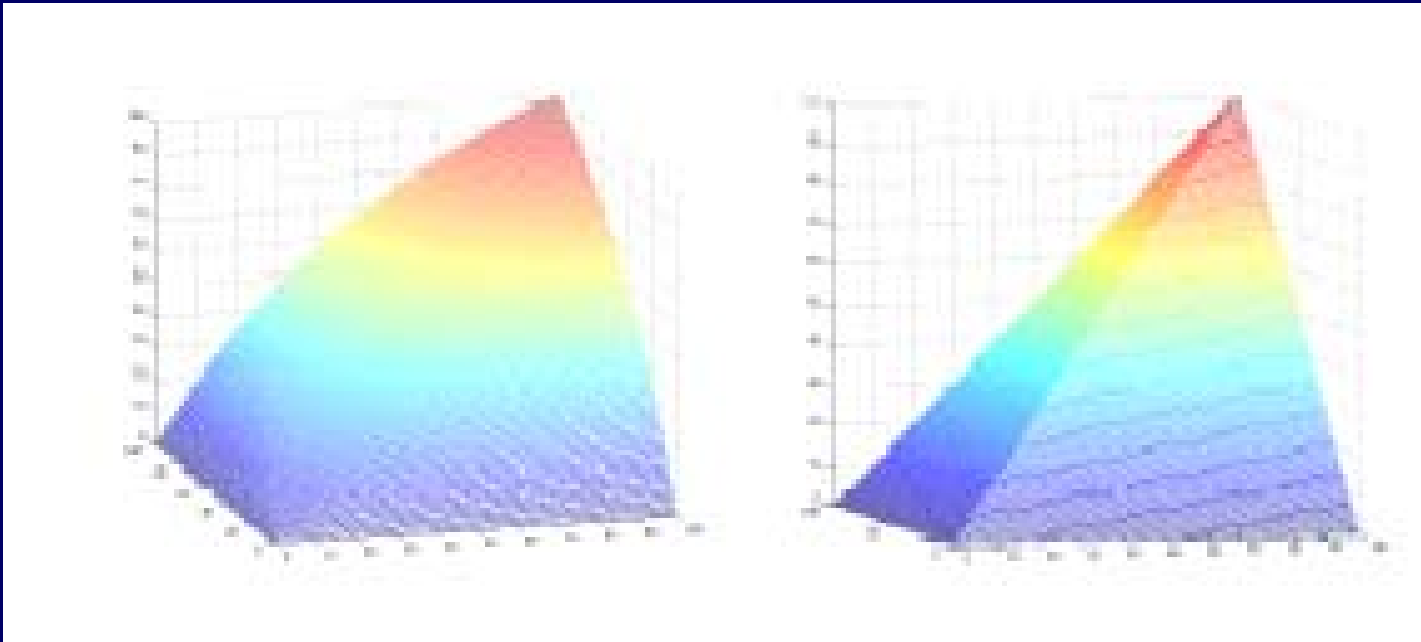
⌘ K-Harmonic Means

$$\text{⌘ } d(x, M) = |M| / \sum_{m \in M} 1 / ||x - m||^2$$

HA (x,y) vs. MIN (x,y)

⌘ K-Harmonic Means

$$\triangle d(x,M) = |M| / \sum_{m \in M} 1 / ||x-m||^2$$



$$\triangle d(x,M) = |M| / \sum_{m \in M} 1 / ||x-m||^p$$

EM (restricted)

⌘ EM with linear mixing of K spherical Gaussians

⌘ EM: recursive E and S steps

⌘ E-Step

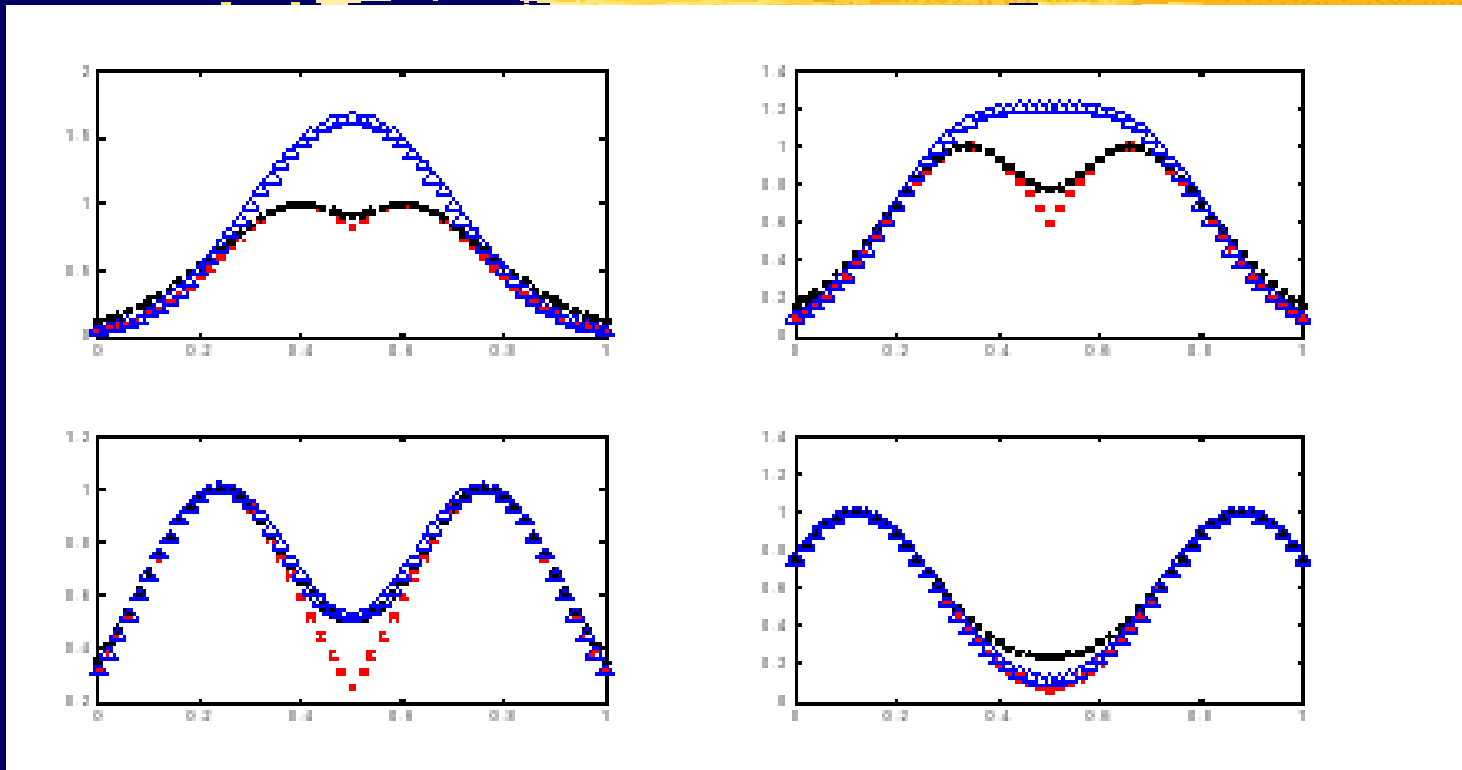
$$p(m_k | x_i) = p(x_i | m_k) \times p(m_k) / \sum_{i=1, N} p(x_i | m_k) \times p(m_k)$$

⌘ M-Step

$$m_k = \sum_{i=1, N} p(m_k | x_i) \times x_i / \sum_{i=1, N} p(m_k | x_i)$$

$$p(m_k) = 1/N \sum_{i=1, N} p(m_k | x_i)$$

Comparison



EM: Red

KH: Black

KM: Blue

Comparison

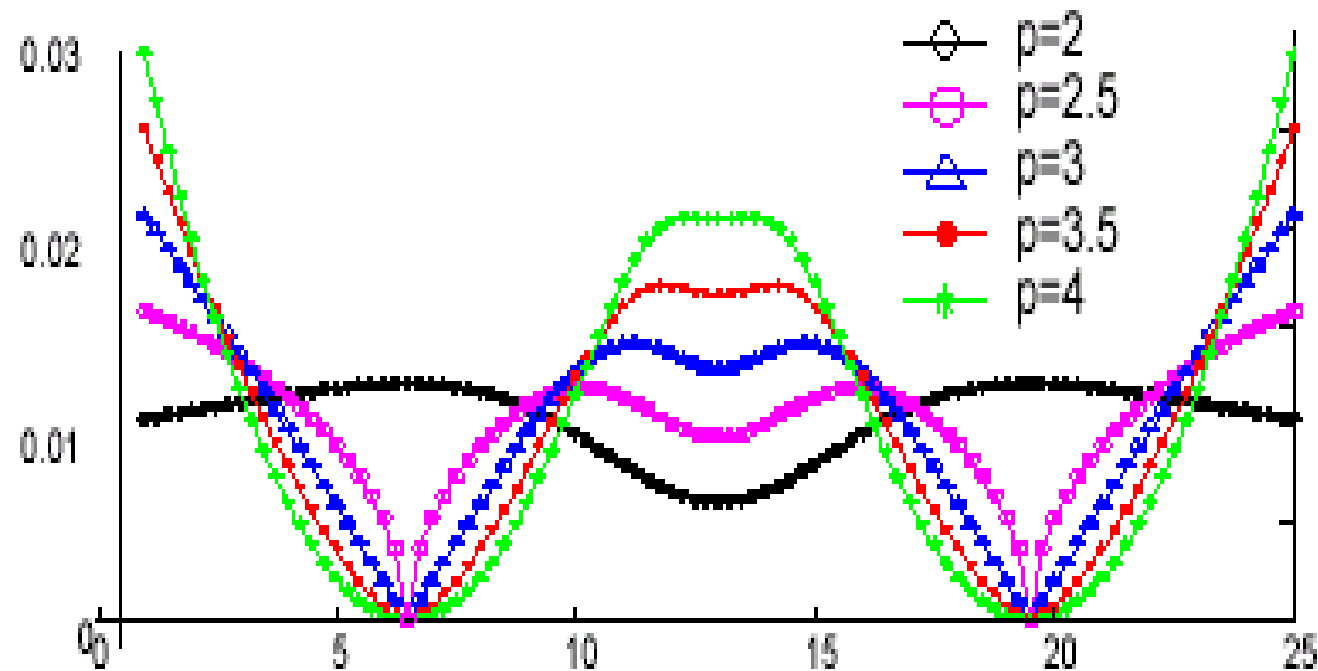


Fig. 3. A Plot of $a(x)$ for K -Harmonic Means with two centers in one-dimensional space. The two centers are located at 6.5 and 19.5.

VQ

⌘ Linde-Buzo-Gray (LBG) algorithm

Let the codebook size be K and the training vectors be $\{x(n) \mid n=1, \dots, M\}$

Step 1 Let the initial codebook $C=\{y(i) \mid i=1, \dots, K\}$ be randomly selected from $\{x(n) \mid n=1, \dots, M\}$

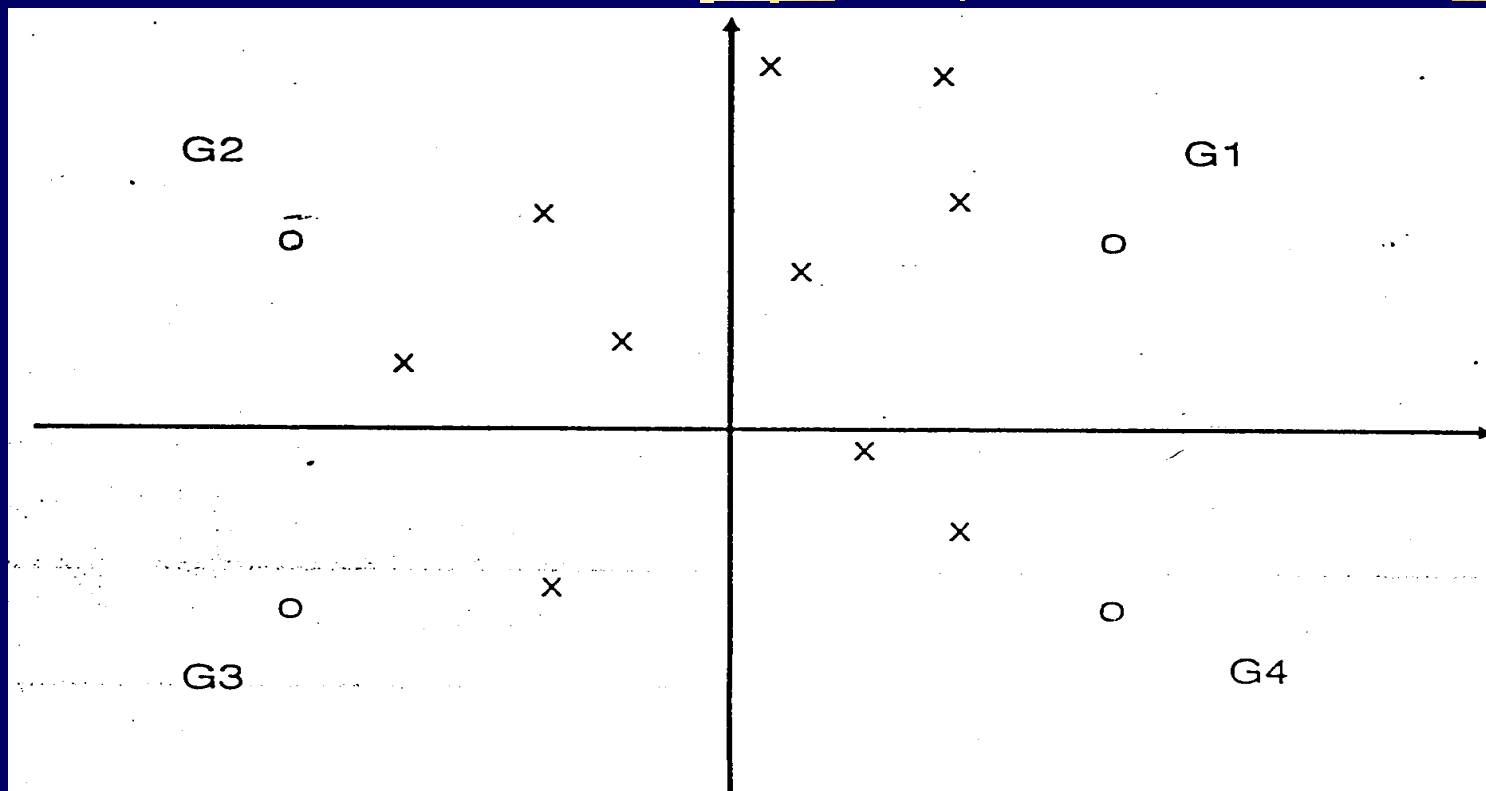
Step 2 Cluster the training vectors into K groups $G(i)$, $i=1, \dots, K$, where $G(i)=\{x(a) \mid d(x(a), y(i)) < d(x(a), y(j)); j \neq i \text{ and } d(p, q) \text{ denotes the distance between } p \text{ and } q\}$

Step 3 If the **distortion** decreases, then go to Step 4
Else stop

Step 4 New $y(i) = \frac{1}{|G(i)|} \sum_{x(a) \in G(i)} x(a)$, where $|G(i)|$ = the number of

vector in $G(i)$; go to Step 2

Codebook Generation

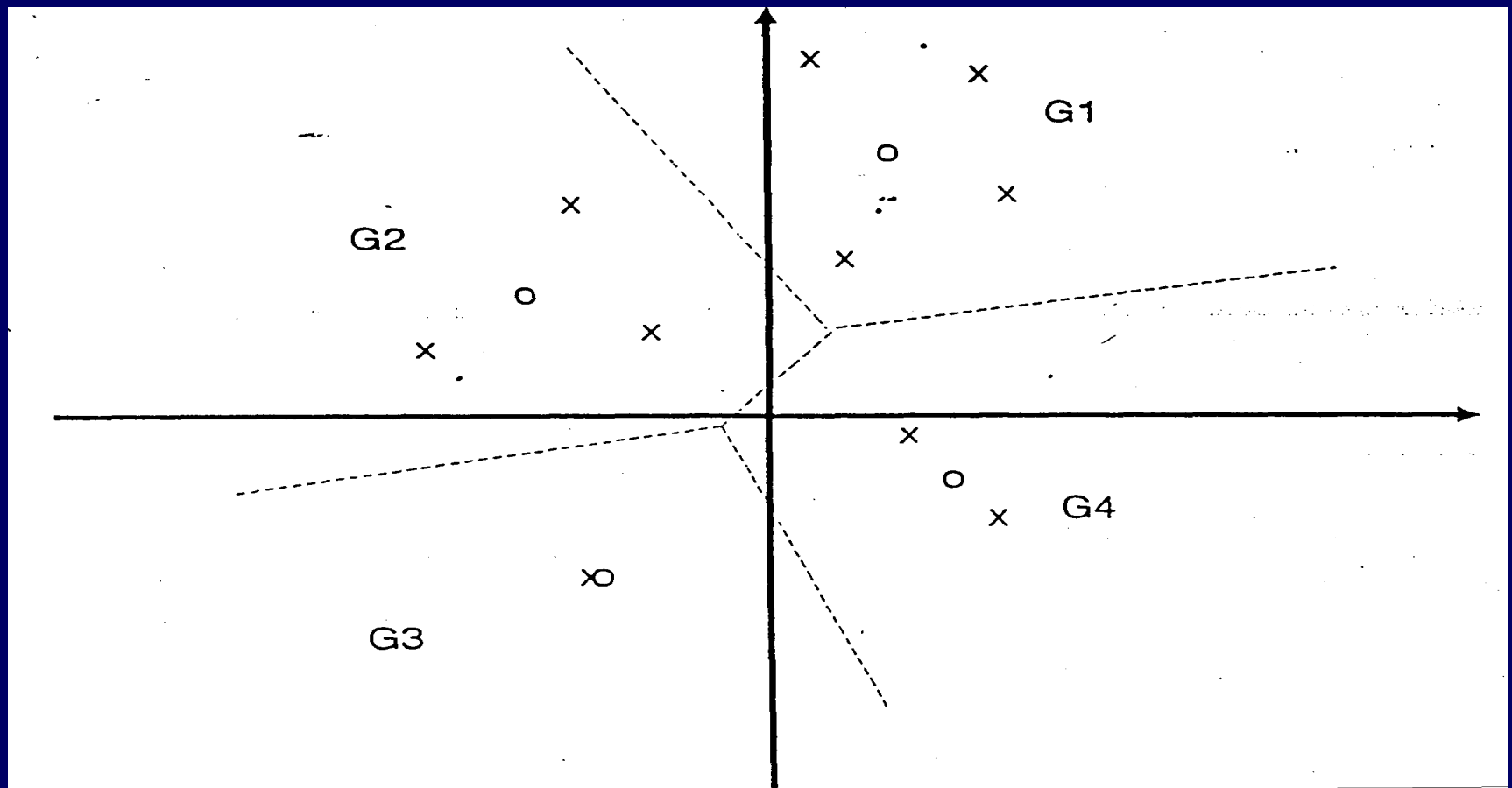


x = training codevectors

O = codewords in the codebook

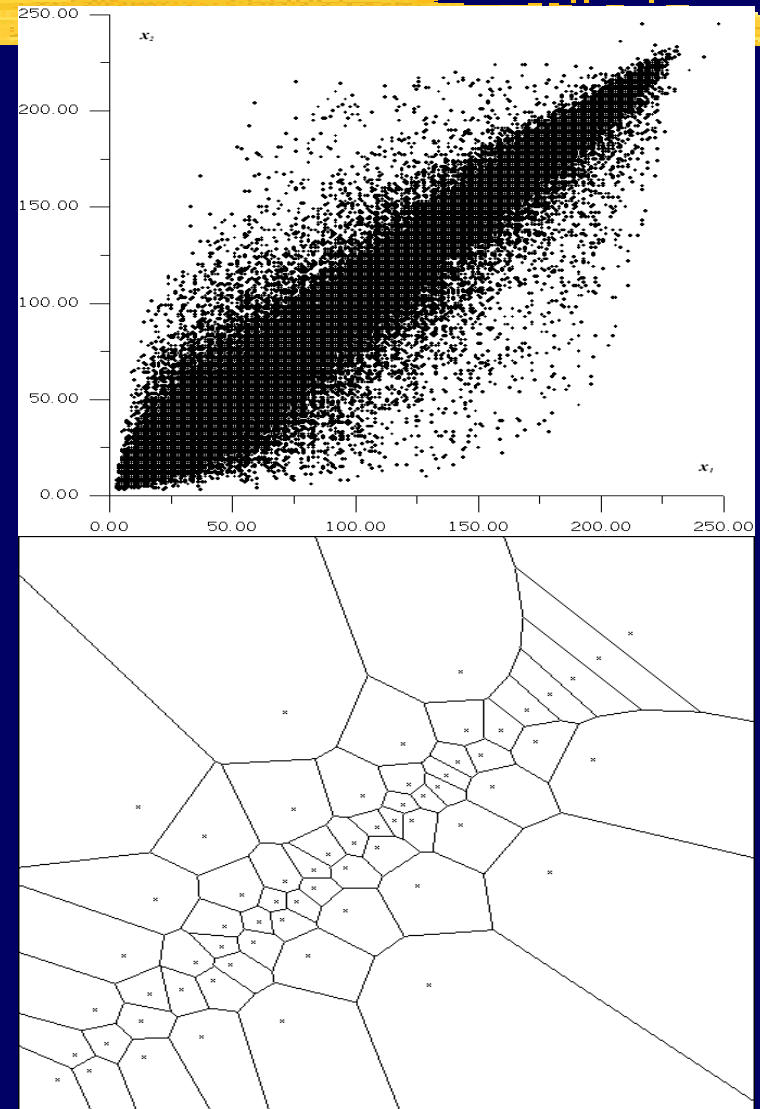
G_i = region encoded into codeword i

1. *Journal of the American Medical Association*, 1997; 277: 1039-1043.



Any finite set S of n points on the plane gives rise to a cellular decomposition of the plane called the **Voronoi partition**

- ◆ For a codebook \mathcal{Y} which contains k codevectors, it is equivalent to a Voronoi partition of $\mathbb{R}^D : r_1, r_2, r_3, \dots, r_k$, where $r_i = \{x \in \mathbb{R}^D : Q(x) = y_i\}$
- ◆ With this definition, it follows that $\bigcup_{i=1}^k r_i = \mathbb{R}^D$ and $r_i \cap r_j = \emptyset$, for $i \neq j$



Codebook Generation

- ⌘ What LBG generate is a local optimal codebook
- ⌘ The following methods are used to improve the codebook
 - ⌘ Perturbation
 - ⌘ Simulated annealing
 - ⌘ Genetic algorithm
 - ⌘ Etc.

Codebook Generation

⌘ Codebook Generation

⌘ Initial codebook

- ⌘ random code
- ⌘ splitting method
- ⌘ pairwise nearest neighbor clustering

⌘ Training codebook

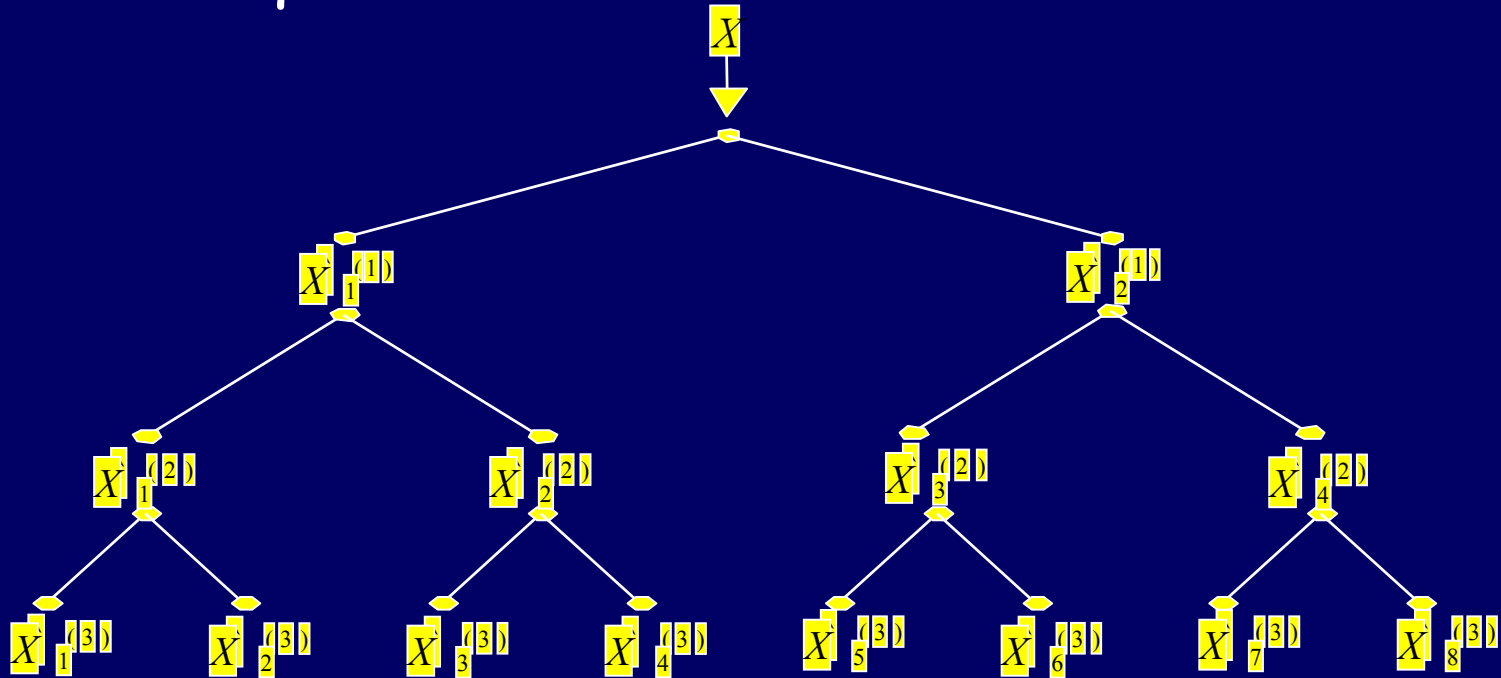
⌘ How to measure the quality of codebook

⌘ Entropy

⌘ SNR

Tree Structured VQ

⌘ TSVQ reduces the quantizer search complexity by replacing full search encoding with a sequence of tree decisions.



TSVQ

⌘ Other designs

☒ Tapered tree structured

- ☒ Number of branches increases as level increases
- ☒ Shown to have better performance than tree with fixed m branches

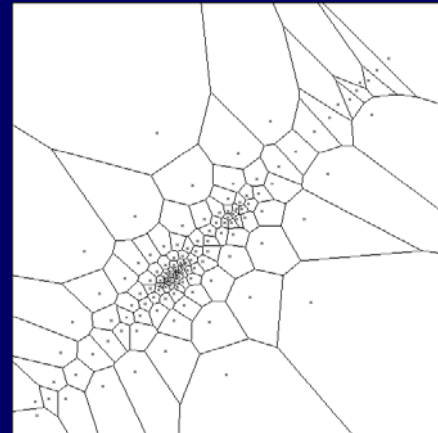
☒ Pruned tree

- ☒ Start from a complete balance tree, branches that contribute the least to the signal fidelity are pruned away
- ☒ Shown to have better performance than full search for speech signals

TSVQ Wrap

⌘ $O(N \log K)$

⌘ Refer to *Vector Quantization and Signal Compression*, by Allen Gersho and Robert M. Gray, Kluwer Academic Pub., 1992 for an extensive study of VQ.



Conclusion

⌘ Algorithms

- ⌘ Inter-Intra

- ⌘ K-Center

- ⌘ K-median

- ⌘ TSVQ

⌘ Key Components

- ⌘ Clustering quality measurement

- ⌘ Inter/intra-cluster distance

- ⌘ K ?