Clustering.
Unsupervised Learning

Maria-Florina Balcan

04/06/2015

Reading:
- Chapter 14.3: Hastie, Tibshirani, Friedman.

Additional resources:
- Center Based Clustering: A Foundational Perspective. Awasthi, Balcan. Handbook of Clustering Analysis. 2015.

Clustering, Informal Goals

Goal: Automatically partition unlabeled data into groups of similar datapoints.

Question: When and why would we want to do this?

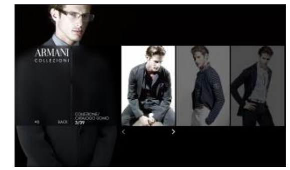
Useful for:

- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
  - Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).

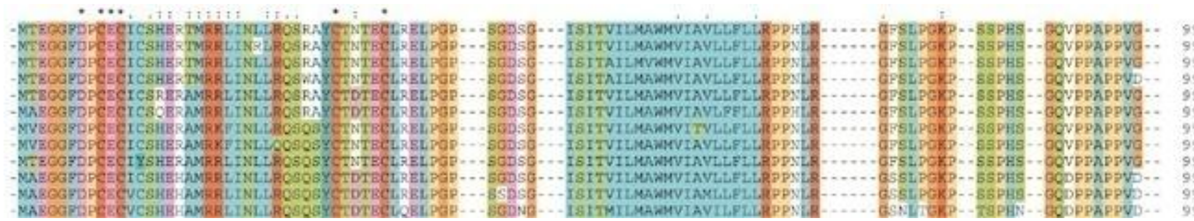
Applications (Clustering comes up everywhere…)

- Cluster news articles or web pages or search results by topic.

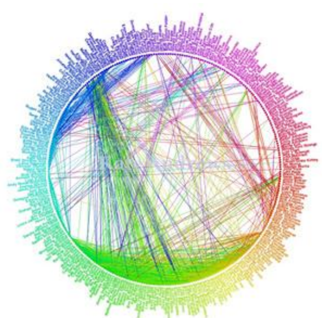


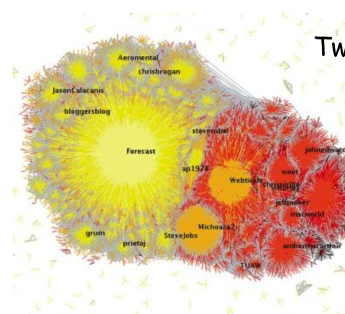
- Cluster protein sequences by function or genes according to expression profile.

- Cluster users of social networks by interest (community detection).



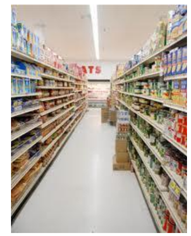
Facebook network



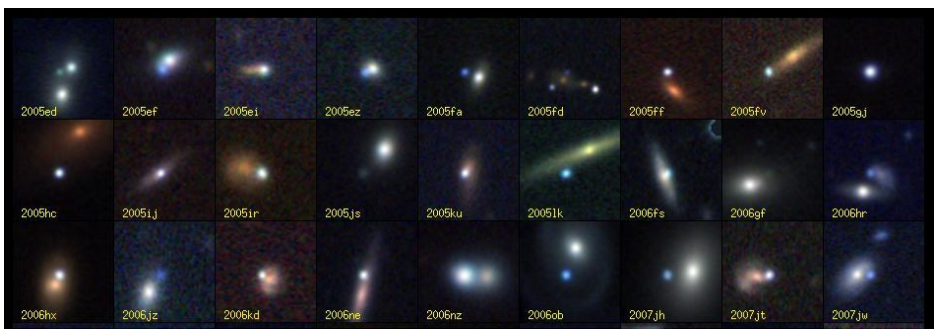
Twitter Network

Applications (Clustering comes up everywhere…)

- Cluster customers according to purchase history.



- Cluster galaxies or nearby stars (e.g. Sloan Digital Sky Survey)



- Image segmentation (clustering pixels)

Clustering

Today:

Flat

- hard Objective based clustering
- Hierarchical clustering Tree
- soft Mention overlapping clusters

[March 4th: EM-style algorithm for clustering for mixture of Gaussians (specific probabilistic model).]

distance metric learning $d(x_1, x_2) = (x_1-x_2)^T A (x_1-x_2)$ $A = L^T L$

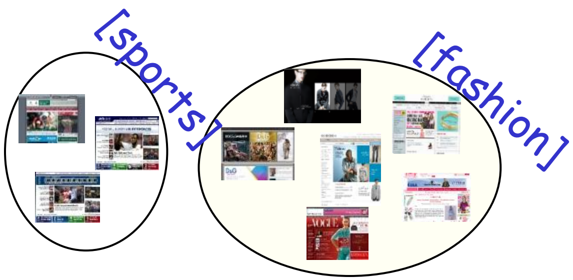
$= \|L(x_1-x_2)\|_2^2$

Objective Based Clustering

Input: A set S of n points, also a distance/dissimilarity measure specifying the distance d(x,y) between pairs (x,y).

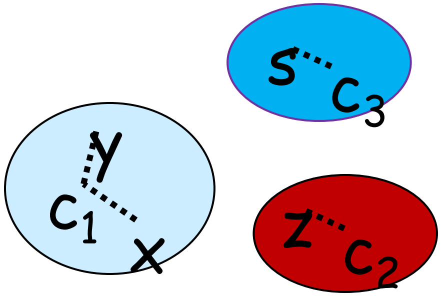
E.g., # keywords in common, edit distance, wavelets coef., etc.

Goal: output a partition of the data.



- k-means: find center pts $c_1, c_2, \dots, c_k$ to
  minimize $\sum_{i=1}^{n} \min_{j\in\{1,\dots,k\}} d^2(\mathbf{x}^i, \mathbf{c}_j)$

- k-median: find center pts $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k$ to
  minimize $\sum_{i=1}^{n} \min_{j\in\{1,\dots,k\}} d(\mathbf{x}^i, \mathbf{c}_j)$



- K-center: find partition to minimize the maxim radius

Euclidean k-means Clustering

Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ in R^d
target #clusters k

Output: k representatives $c_1, \mathbf{c}_2, \ldots, c_k \in \mathrm{R}^d$

Objective: choose $c_1, \mathbf{c}_2, \ldots, c_k \in \mathrm{R}^d$ to minimize

$$\min_{c} \underbrace{\sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \left\|\mathbf{x}^i - \mathbf{c}_j\right\|^2}_{W}$$

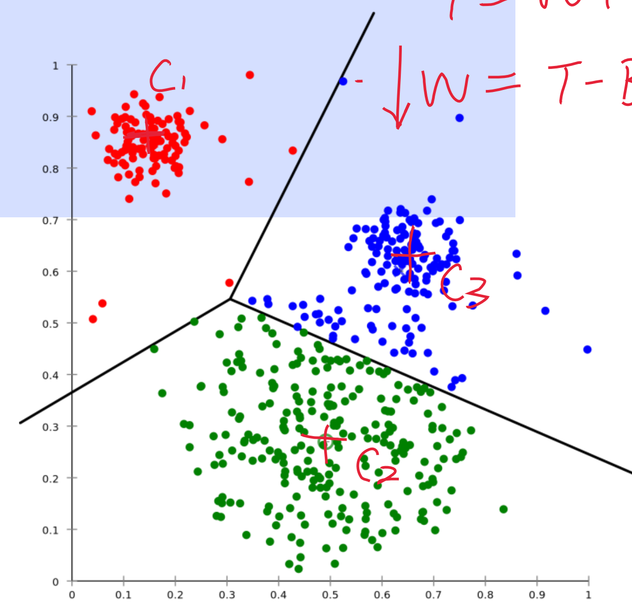
within-cluster cov. (W)

between-cluster cov. (B)

total cov. (T)

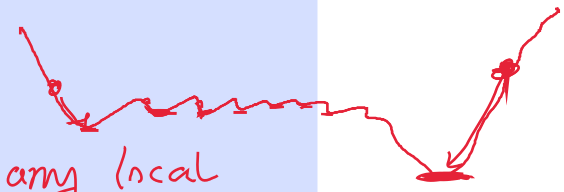
T = W + B

↓W = T − B↑



Euclidean k-means Clustering

Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ in R^d
target #clusters k

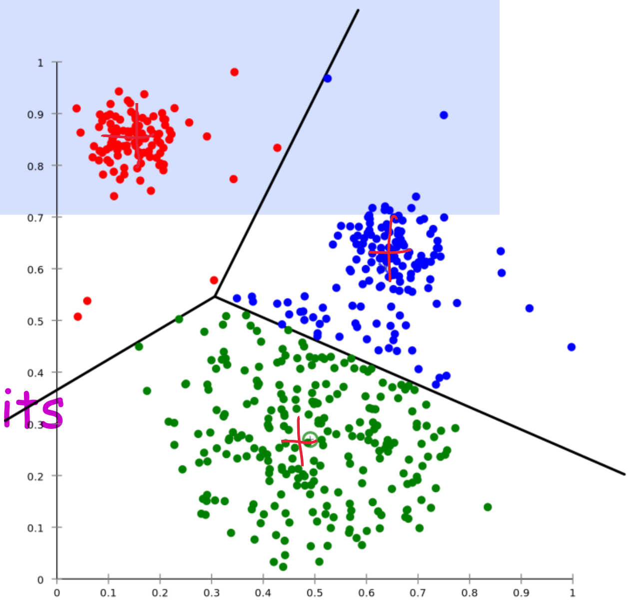


Output: k representatives $\boldsymbol{c}_1, \mathbf{c}_2, \dots, \boldsymbol{c}_k \in R^d$

Objective: choose $\boldsymbol{c}_1, \mathbf{c}_2, \dots, \boldsymbol{c}_k \in R^d$ to minimize

non-convex

$$\sum_{i=1}^{n} \min_{j\in\{1,\dots,k\}} \left\|\mathbf{x}^i - \mathbf{c}_j\right\|^2$$



Natural assignment: each point assigned to its closest center, leads to a Voronoi partition.

Euclidean k-means Clustering

$$\min_{C} \sum_{i=1}^{n} \min_{j=1,2,\dots,k} \|x_i - c_j\|_2^2$$

$$\Updownarrow$$

$$\min_{R,C} \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \|x_i - c_j\|_2^2,$$

s.t. $r_{ij} \in \{0,1\}$, $R\mathbb{1}_k = \mathbb{1}_n$

$$\Updownarrow$$

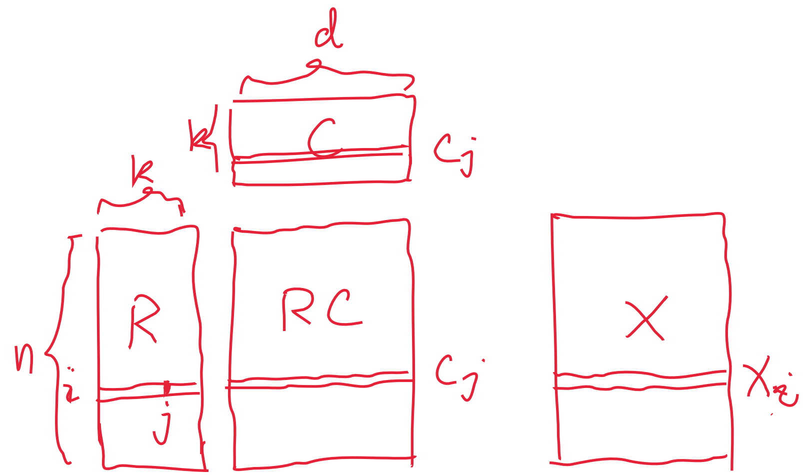
$$\min_{R,C} \|X - RC\|_F^2$$

s.t. $R \in \{0,1\}^{n\times k}$, $R\cdot\mathbb{1}_k = \mathbb{1}_n$

$\|a\|_2^2 = \langle a, a\rangle = a^T a$

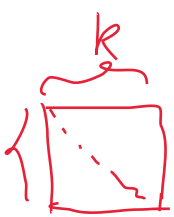
$\|A\|_F^2 = \langle A, A\rangle = Tr(A^TA)$
$= Tr(AA^T)$

$$r_{ij} = \begin{cases} 1, & x_i \in C_j \\ 0, & \text{otherwise.} \end{cases}$$



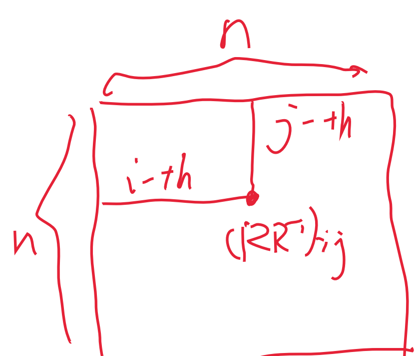
R: indicator matrix

R^TR:



(diagonal)

RR^T:



Euclidean k-means Clustering

Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ in R^d

target #clusters k

Output: k representatives $\boldsymbol{c}_1, \mathbf{c}_2, \ldots, \boldsymbol{c}_k \in \mathrm{R}^\mathrm{d}$

Objective: choose $\boldsymbol{c}_1, \mathbf{c}_2, \ldots, \boldsymbol{c}_k \in \mathrm{R}^\mathrm{d}$ to minimize

$$\sum_{\mathrm{i}=1}^{\mathrm{n}} \min_{\mathrm{j} \in \{1,\ldots,\mathrm{k}\}} \left|\left|\mathbf{x}^\mathbf{i} - \mathbf{c}_\mathbf{j}\right|\right|^2$$

Computational complexity:

NP hard: even for $\mathrm{k} = 2$ [Dagupta'08] or

$\mathrm{d} = 2$ [Mahajan-Nimbhorkar-Varadarajan09]

There are a couple of easy cases…

An Easy Case for k-means: k=1

Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ in R^d

Output: $\boldsymbol{c} \in \mathrm{R}^\mathrm{d}$ to minimize $\sum_{i=1}^{n} \left\|\mathbf{x}^\mathbf{i} - \mathbf{c}\right\|^2$

Solution: The optimal choice is $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^\mathbf{i}$

Idea: bias/variance like decomposition

$$\frac{1}{n}\sum_{i=1}^{n} \left\|\mathbf{x}^\mathbf{i} - \mathbf{c}\right\|^2 = \left\|\boldsymbol{\mu} - \mathbf{c}\right\|^2 + \frac{1}{n}\sum_{i=1}^{n} \left\|\mathbf{x}^\mathbf{i} - \boldsymbol{\mu}\right\|^2$$

Quiz

Avg k-means cost wrt c

Avg k-means cost wrt μ

So, the optimal choice for $\mathbf{c}$ is $\boldsymbol{\mu}$.

Common Heuristic in Practice: The Lloyd's method

[Least squares quantization in PCM, Lloyd, IEEE Transactions on Information Theory, 1982]

Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ in R^d

Initialize centers $\boldsymbol{c}_1, \mathbf{c}_2, \dots, \boldsymbol{c}_k \in R^d$ and clusters $C_1, C_2, \dots, C_k$ in any way.

Repeat until there is no further change in the cost. $\|X - RC\|_F^2$

E-step • For each $\mathbf{j}$: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c_j}\}$ R

M-step • For each $\mathbf{j}$: $\mathbf{c_j} \leftarrow$ mean of C_j C

Common Heuristic in Practice: The Lloyd's method

[Least squares quantization in PCM, Lloyd, IEEE Transactions on Information Theory, 1982]

Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ in R^d

Initialize centers $\boldsymbol{c}_1, \mathbf{c}_2, \ldots, \boldsymbol{c}_k \in R^d$ and clusters $C_1, C_2, \ldots, C_k$ in any way.

$\|X - RC\|_F^2$

Repeat until there is no further change in the cost.

- For each j: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c}_j\}$
- For each j: $\mathbf{c}_j \leftarrow$ mean of C_j $\leftarrow c_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$

$C_j = \{ x_i \mid j = \arg\min_{k=1,2,\ldots,k} \|x_i - c_k\|_2^2, \; i = 1, 2, \ldots, n \}$

Holding $\boldsymbol{c}_1, \mathbf{c}_2, \ldots, \boldsymbol{c}_k$ fixed, pick optimal $C_1, C_2, \ldots, C_k$

Holding $C_1, C_2, \ldots, C_k$ fixed, pick optimal $\boldsymbol{c}_1, \mathbf{c}_2, \ldots, \boldsymbol{c}_k$

Common Heuristic: The Lloyd's method

Input: A set of n datapoints $\mathbf{x^1, x^2, \ldots, x^n}$ in $\mathrm{R^d}$

Initialize centers $\mathbf{c_1, c_2, \ldots, c_k} \in \mathrm{R^d}$ and clusters $\mathrm{C_1, C_2, \ldots, C_k}$ in any way.

Repeat until there is no further change in the cost.

1. R • For each j: $\mathrm{C_j} \leftarrow \{x \in S$ whose closest center is $\mathbf{c_j}\}$

2. C • For each j: $\mathbf{c_j} \leftarrow$ mean of $\mathrm{C_j}$

Note: it always converges.

- the cost always drops and
- there is only a finite #s of Voronoi partitions

  (so a finite # of values the cost could take)

Common Heuristic: The Lloyd's method

$$\min_{R,C} \underbrace{\|X - RC\|_F^2}_{} = Q(R,C).$$

$$\text{s.t.} \quad R \in \{0,1\}^{n \times k}, \quad R \cdot \mathbb{1}_k = \mathbb{1}_n$$

$$\mathbb{1}_k = \left.\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}\right\} k, \quad \mathbb{1}_n = \left.\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}\right\} n.$$

---

t-th iteration.

$$R^{(t-1)}, \ C^{(t-1)}$$

1. $R^{(t)} = \arg\min_R Q(R, C^{(t-1)})$

$$Q(R^{(t)}, C^{(t-1)}) \le Q(R^{(t-1)}, C^{(t-1)})$$

2. $C^{(t)} = \arg\min_C Q(R^{(t)}, C)$

$$Q(R^{(t)}, C^{(t)}) \le Q(R^{(t)}, C^{(t-1)})$$

$$\Rightarrow \quad Q(R^{(t)}, C^{(t)}) \le Q(R^{(t-1)}, C^{(t-1)})$$

Initialization for the Lloyd's method

Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ in R^d

Initialize centers $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k \in R^d$ and clusters $C_1, C_2, \dots, C_k$ in any way.

Repeat until there is no further change in the cost.

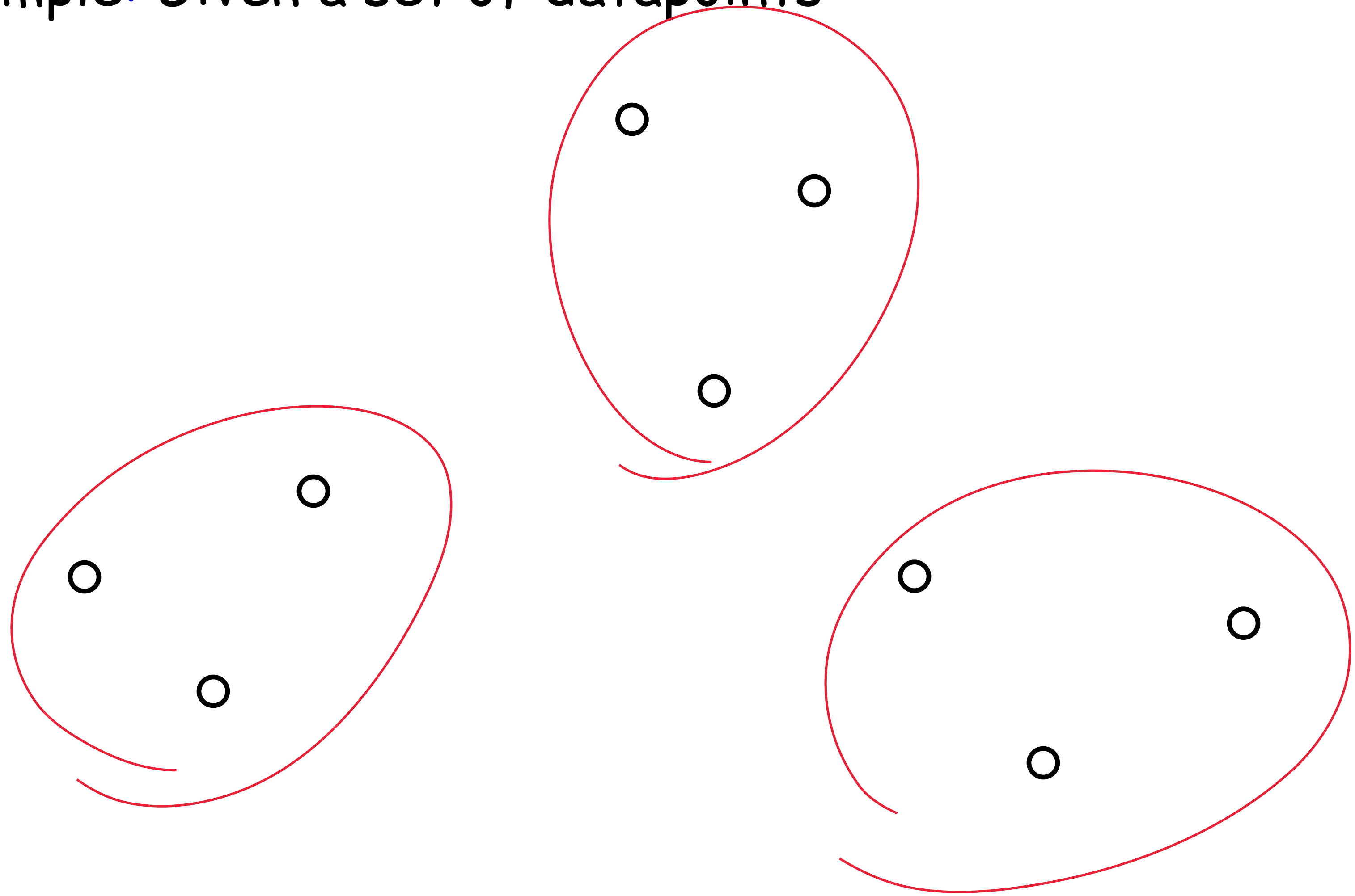
- For each $\mathbf{j}$: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c_j}\}$
- For each $\mathbf{j}$: $\mathbf{c_j} \leftarrow$mean of C_j

- Initialization is crucial (how fast it converges, quality of solution output)
- Discuss techniques commonly used in practice
  - Random centers from the datapoints (repeat a few times)
  - Furthest traversal
  - K-means ++ (works well and has provable guarantees)

Lloyd's method: Random Initialization

Lloyd's method: Random Initialization

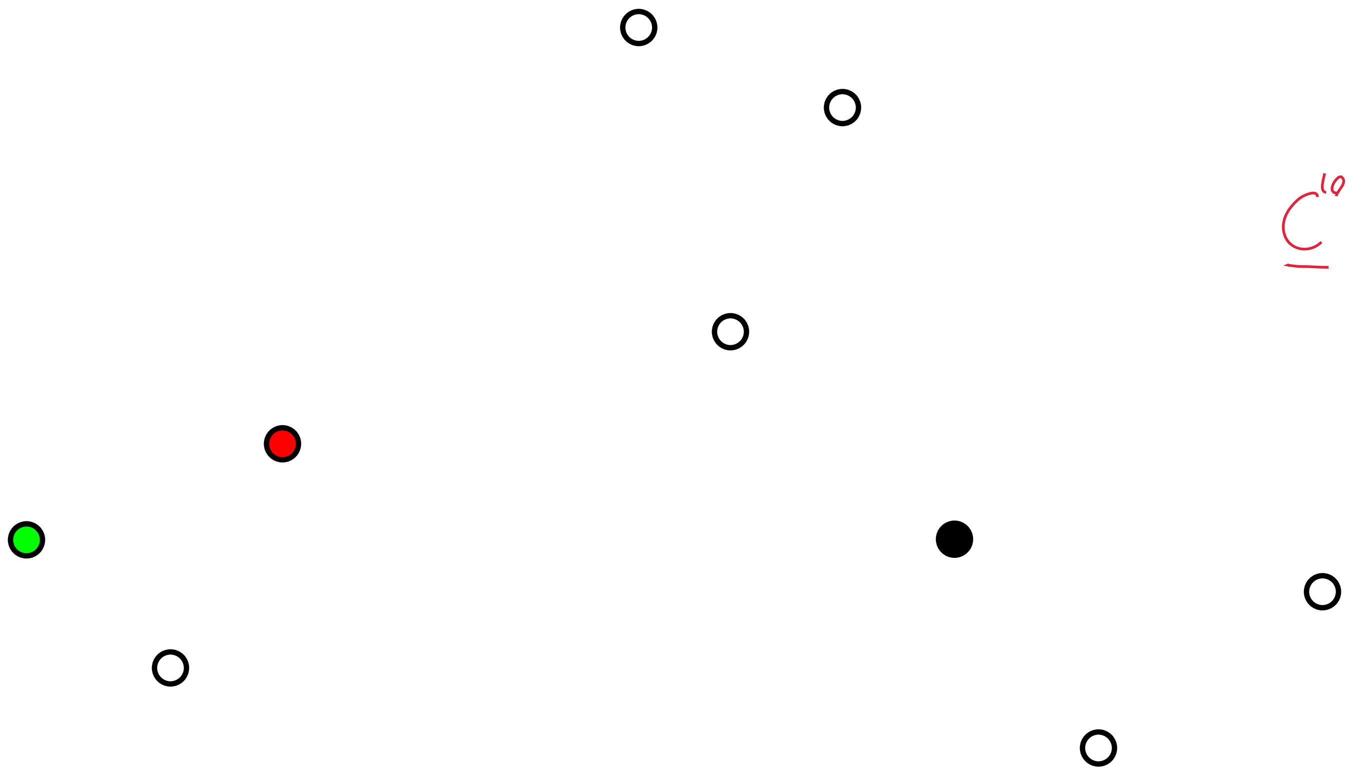
Example: Given a set of datapoints



Lloyd's method: Random Initialization

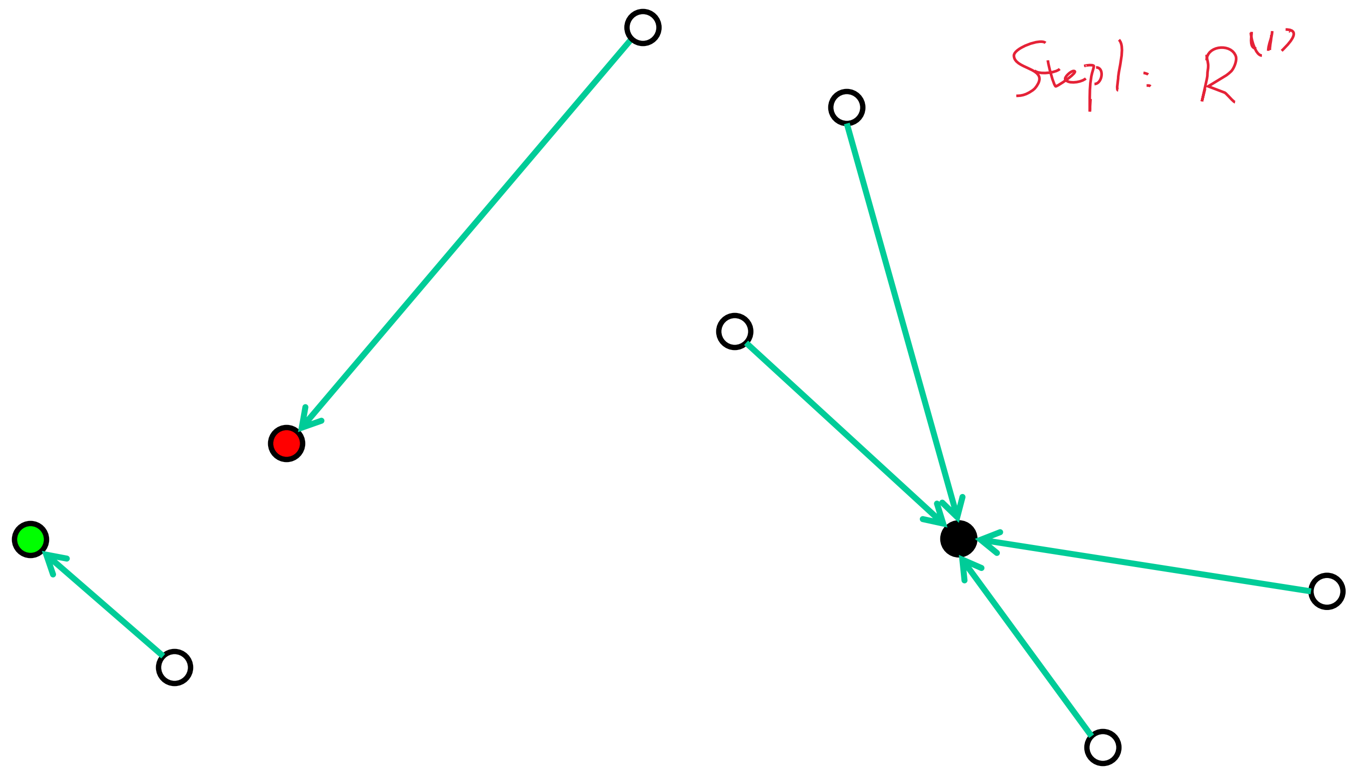
Select initial centers at random

$\underline{C}^{(0)}$



Lloyd's method: Random Initialization

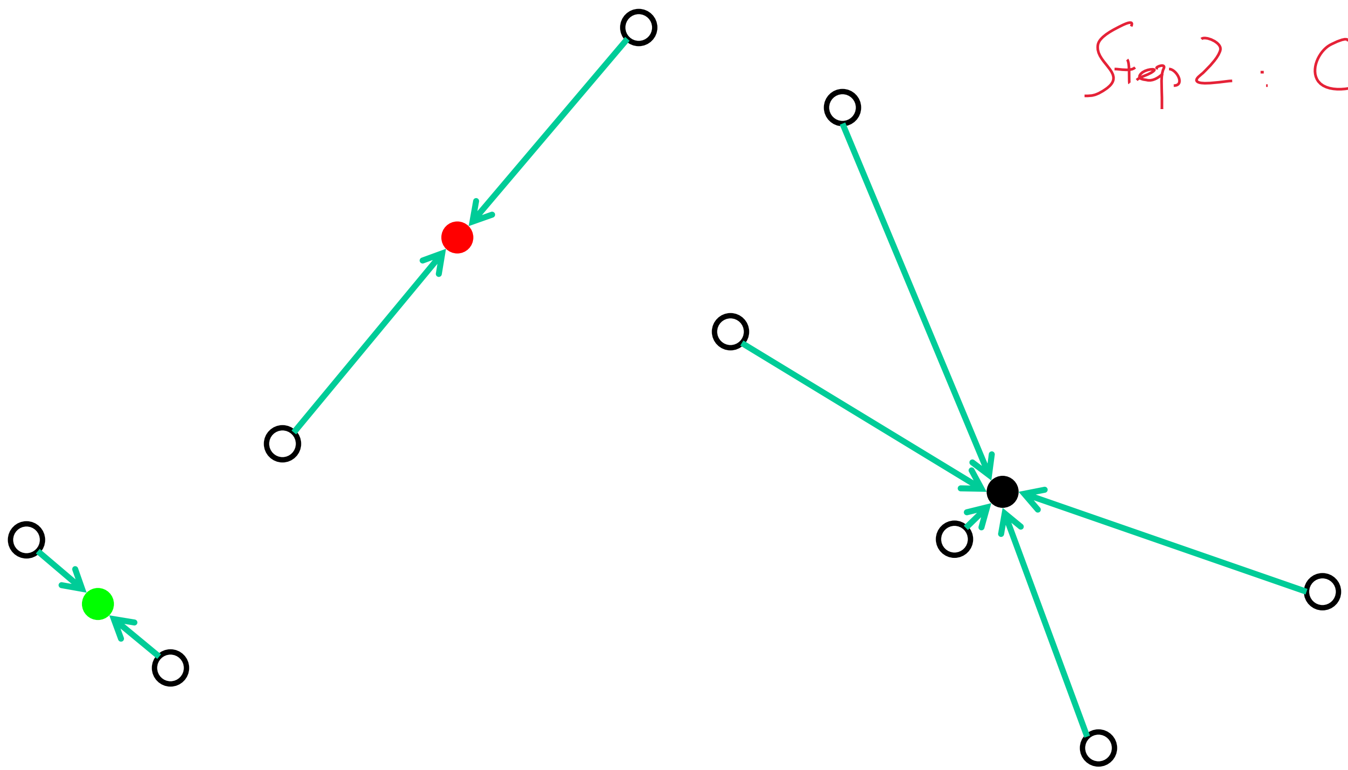
Assign each point to its nearest center



Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering

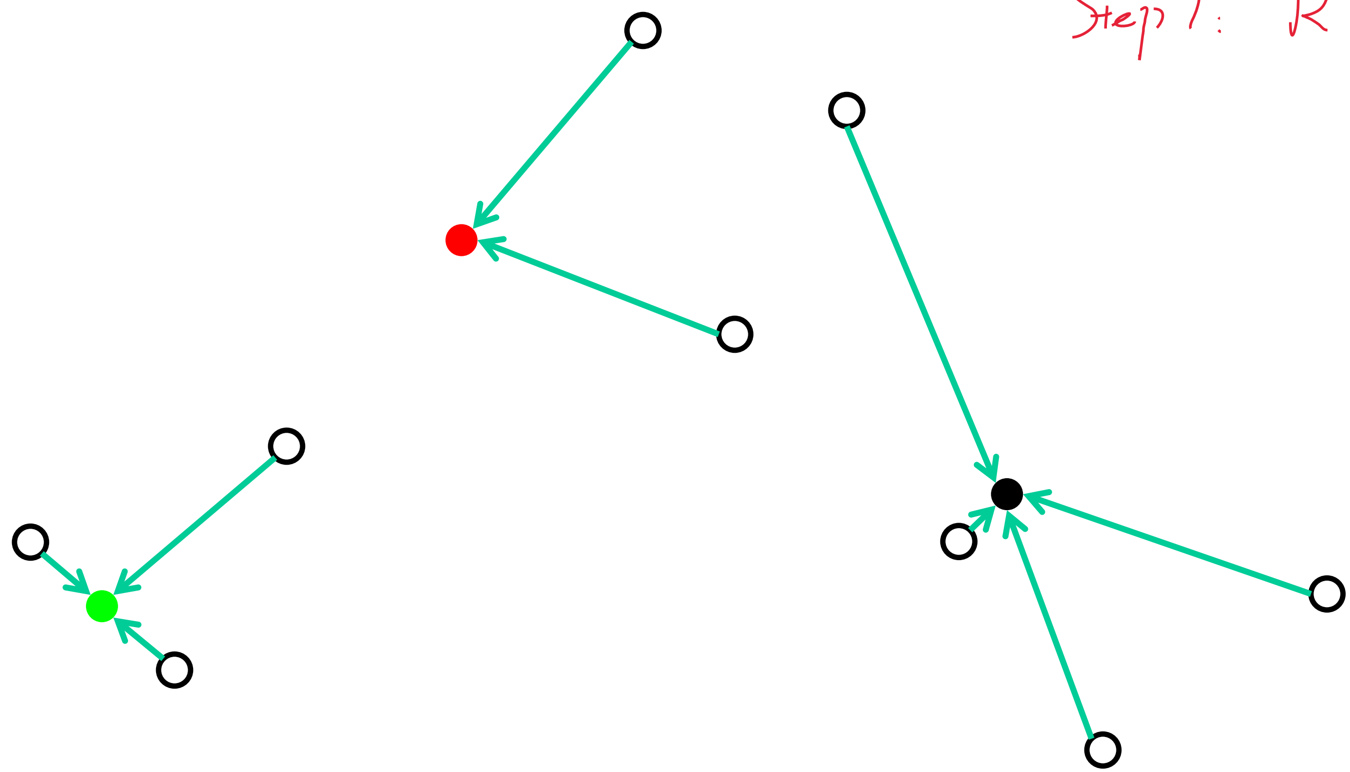
Step 2 : $C^{(1)}$



Lloyd's method: Random Initialization

Assign each point to its nearest center

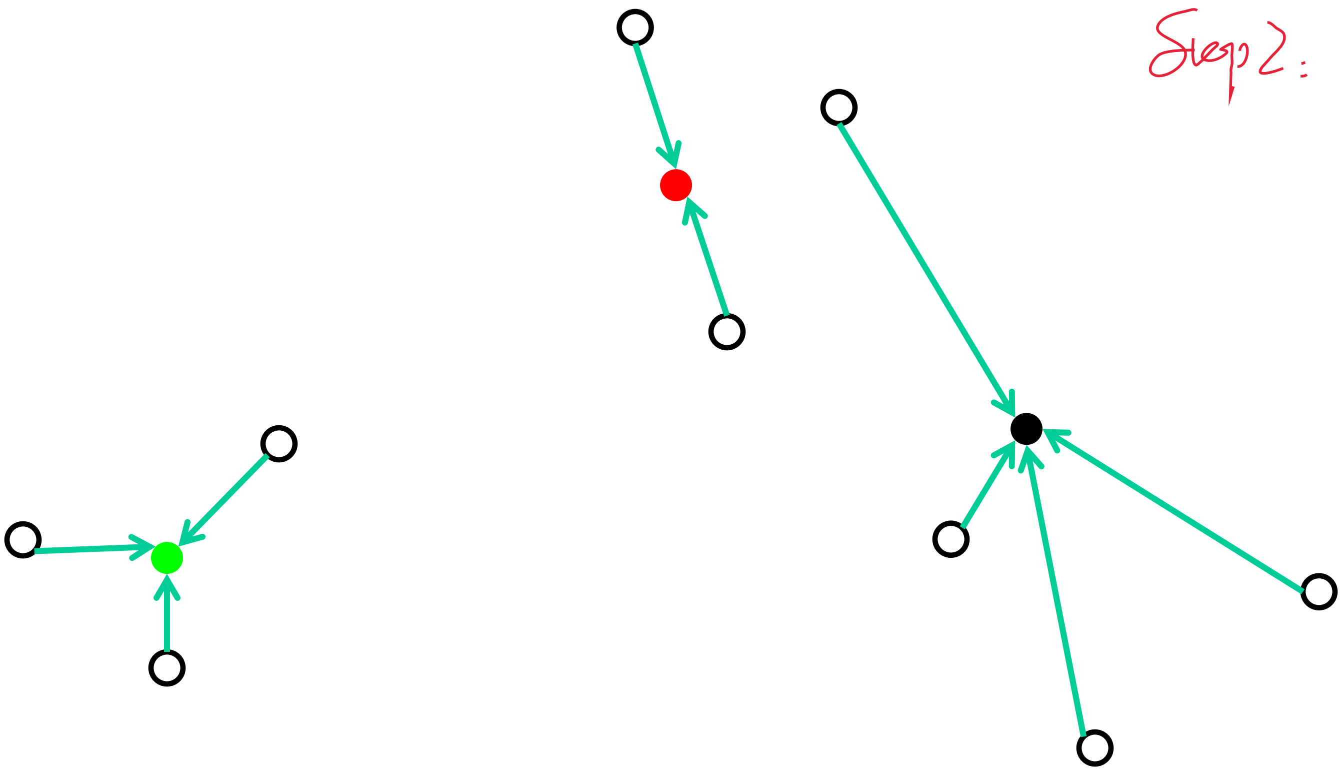
Step 1: $R^{(2)}$



Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering

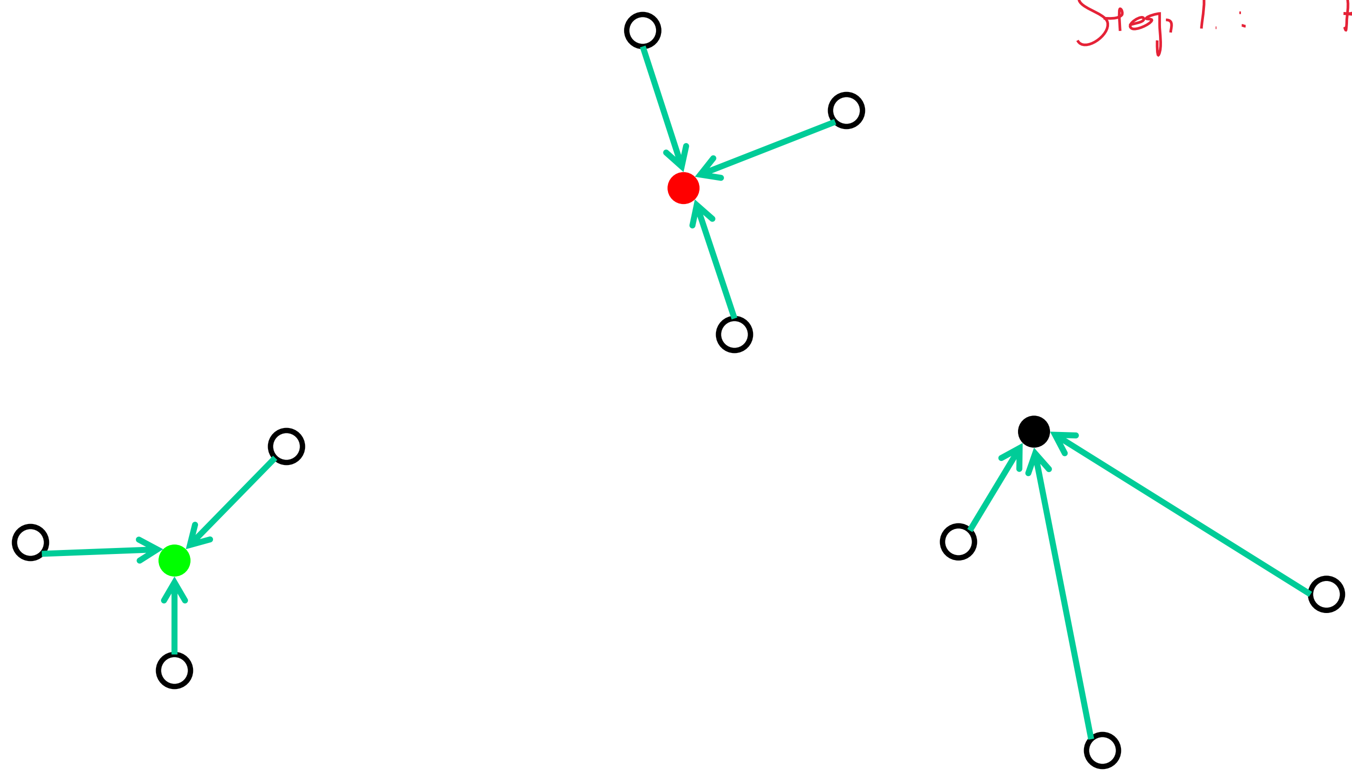
Step 2: $C^{(2)}$



Lloyd's method: Random Initialization

Assign each point to its nearest center

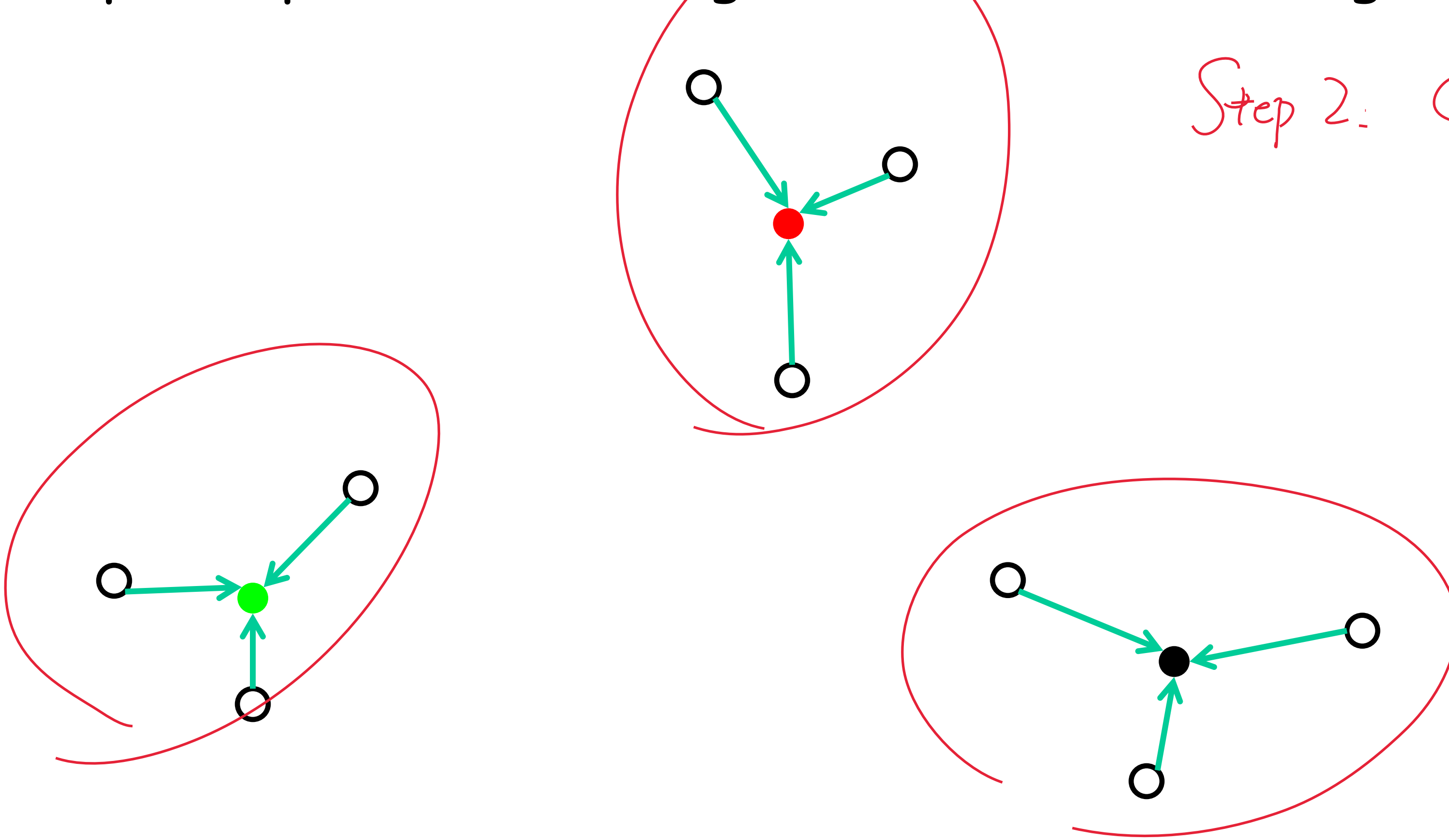
Step 1: $R^{(3)}$



Lloyd's method: Random Initialization

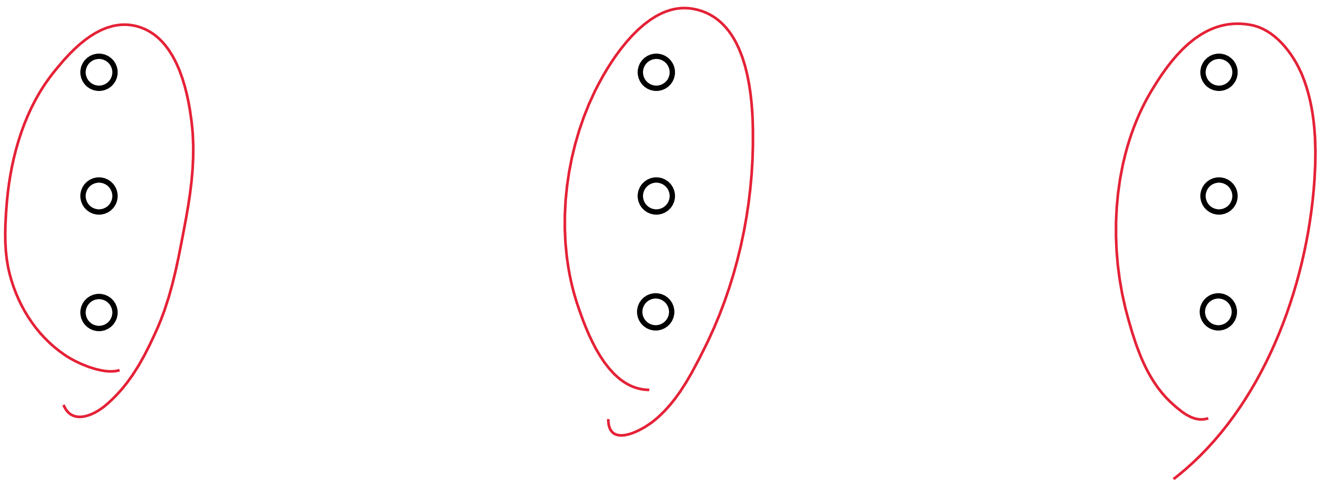
Recompute optimal centers given a fixed clustering

Step 2: $C^{(3)}$



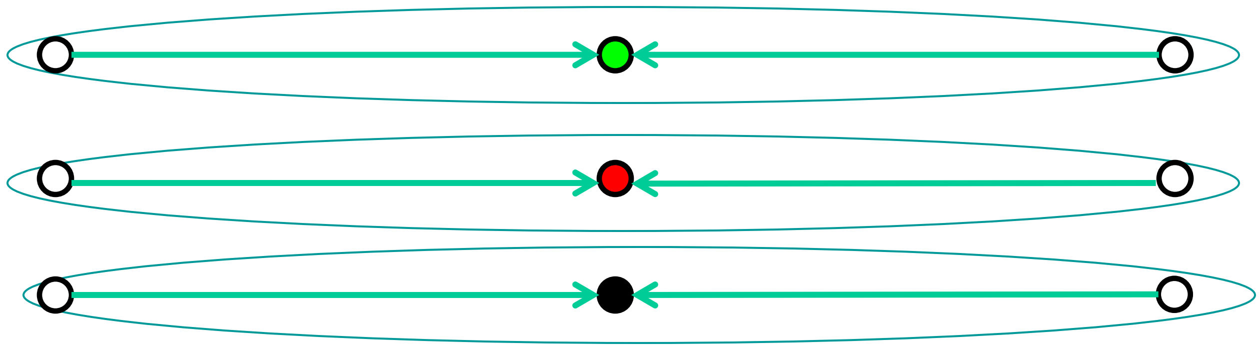
Get a good quality solution in this example.

Lloyd's method: Performance



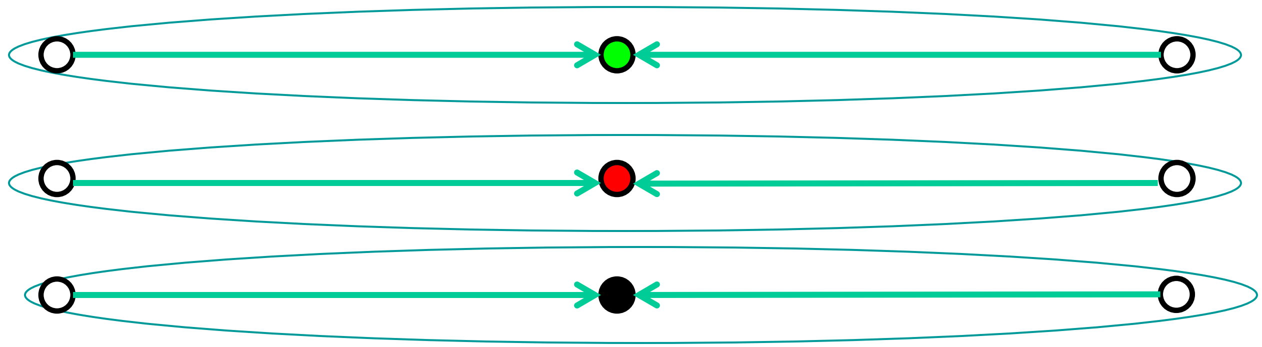
It always converges, but it may converge at a local optimum that is different from the global optimum, and in fact could be arbitrarily worse in terms of its score.

Lloyd's method: Performance

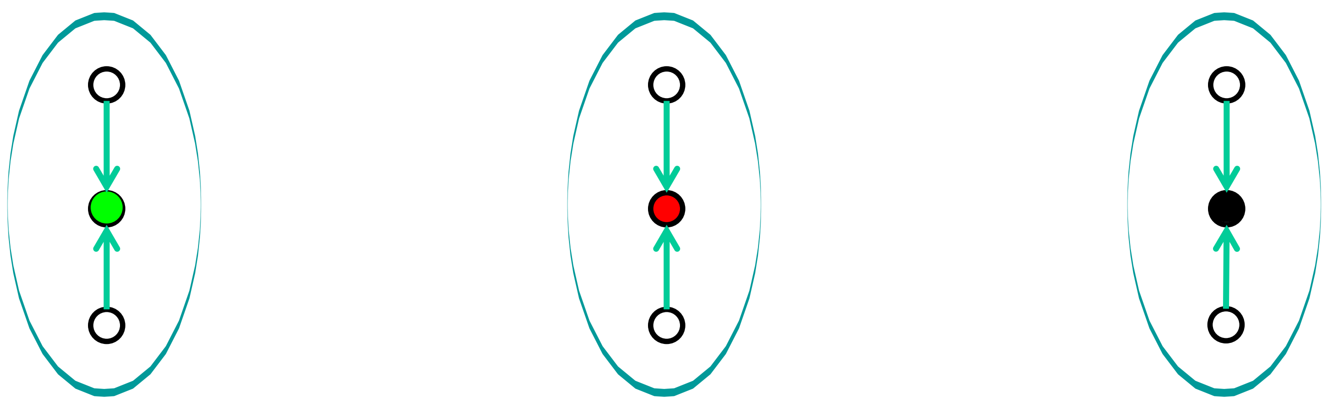


Local optimum: every point is assigned to its nearest center and every center is the mean value of its points.

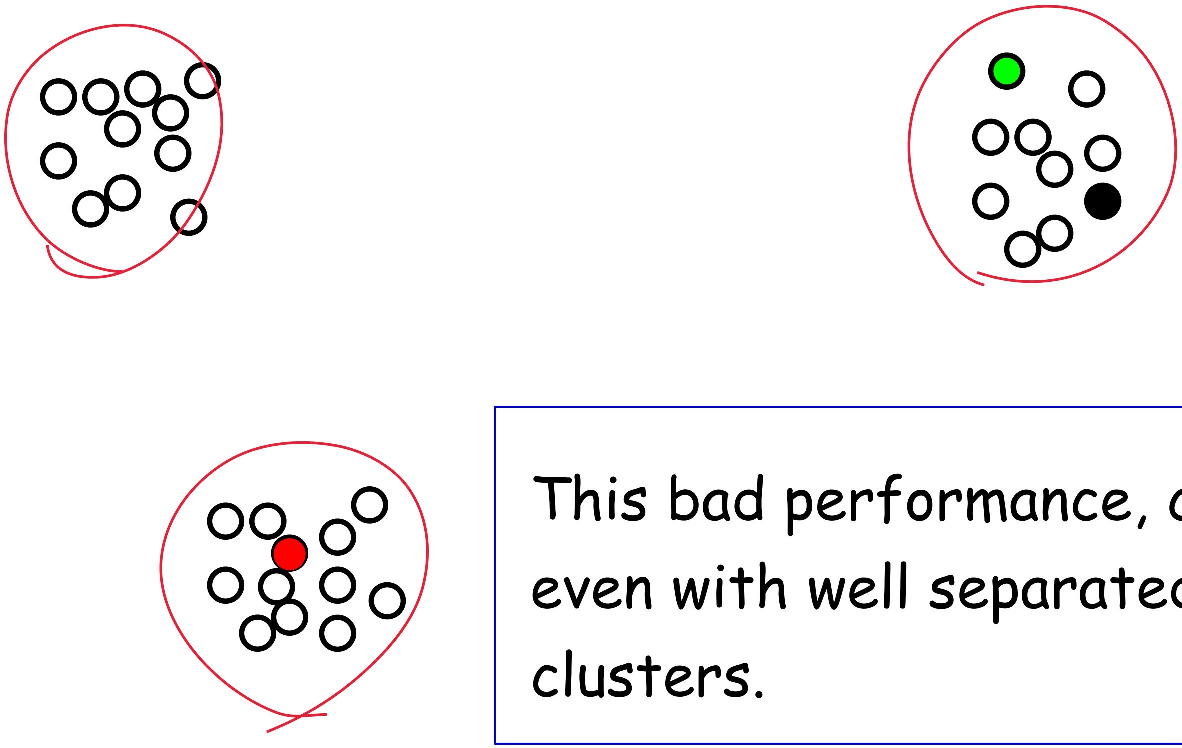
Lloyd's method: Performance



.It is arbitrarily worse than optimum solution.... 

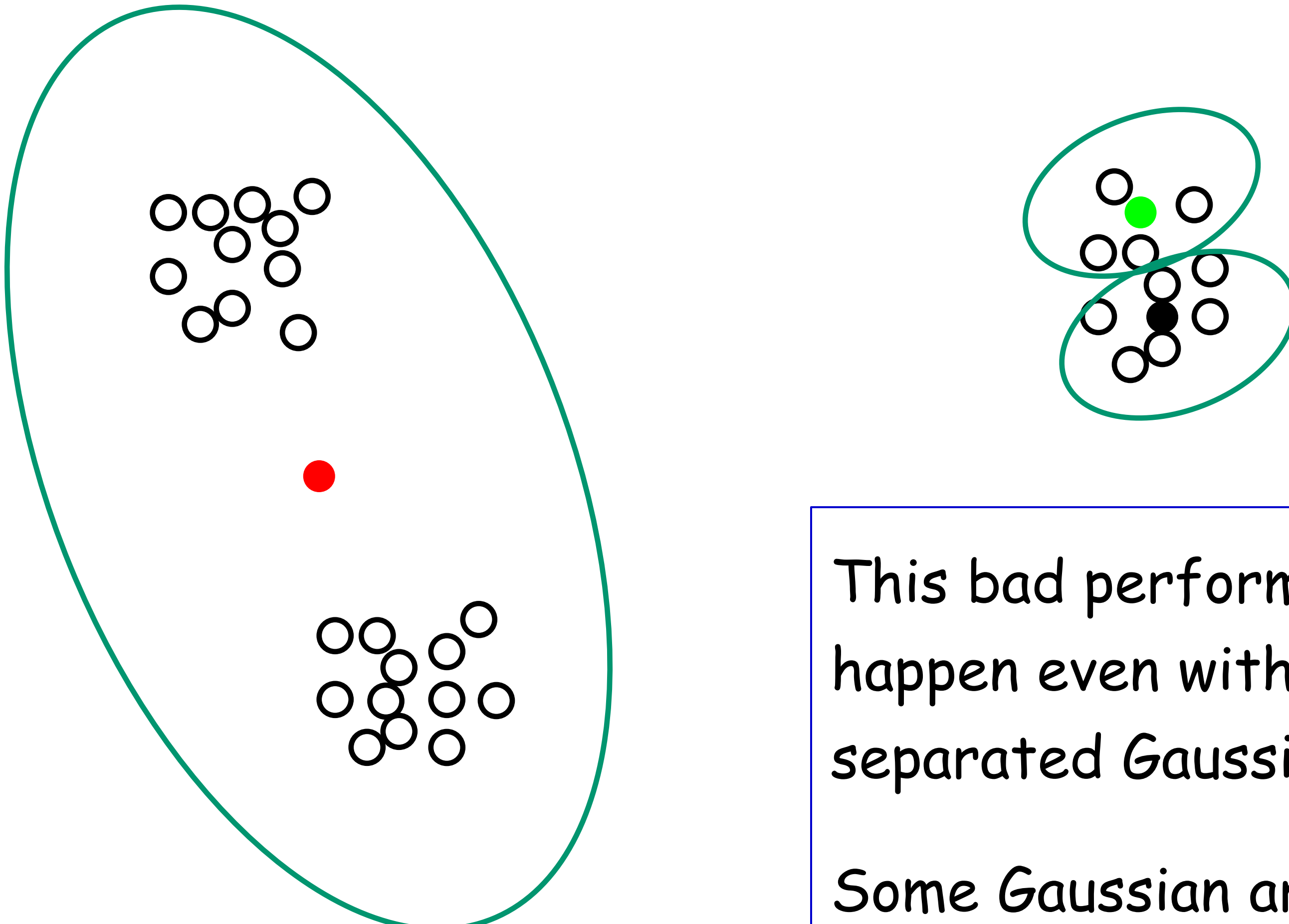


Lloyd's method: Performance



This bad performance, can happen even with well separated Gaussian clusters.

Lloyd's method: Performance



This bad performance, can happen even with well separated Gaussian clusters.

Some Gaussian are combined.....

Lloyd's method: Performance

- If we do random initialization, as k increases, it becomes more likely we won't have perfectly picked one center per Gaussian in our initialization (so Lloyd's method will output a bad solution).

$k \cdot (k-1) \cdot (k-2) \cdots 2 \cdot 1 = k!$

  - For k equal-sized Gaussians, Pr[each initial center is in a different Gaussian] $\approx \frac{k!}{k^k} \approx \frac{1}{e^k}$

$\binom{k+k-1}{k-1} = O(k^k)$

  - Becomes unlikely as k gets large.

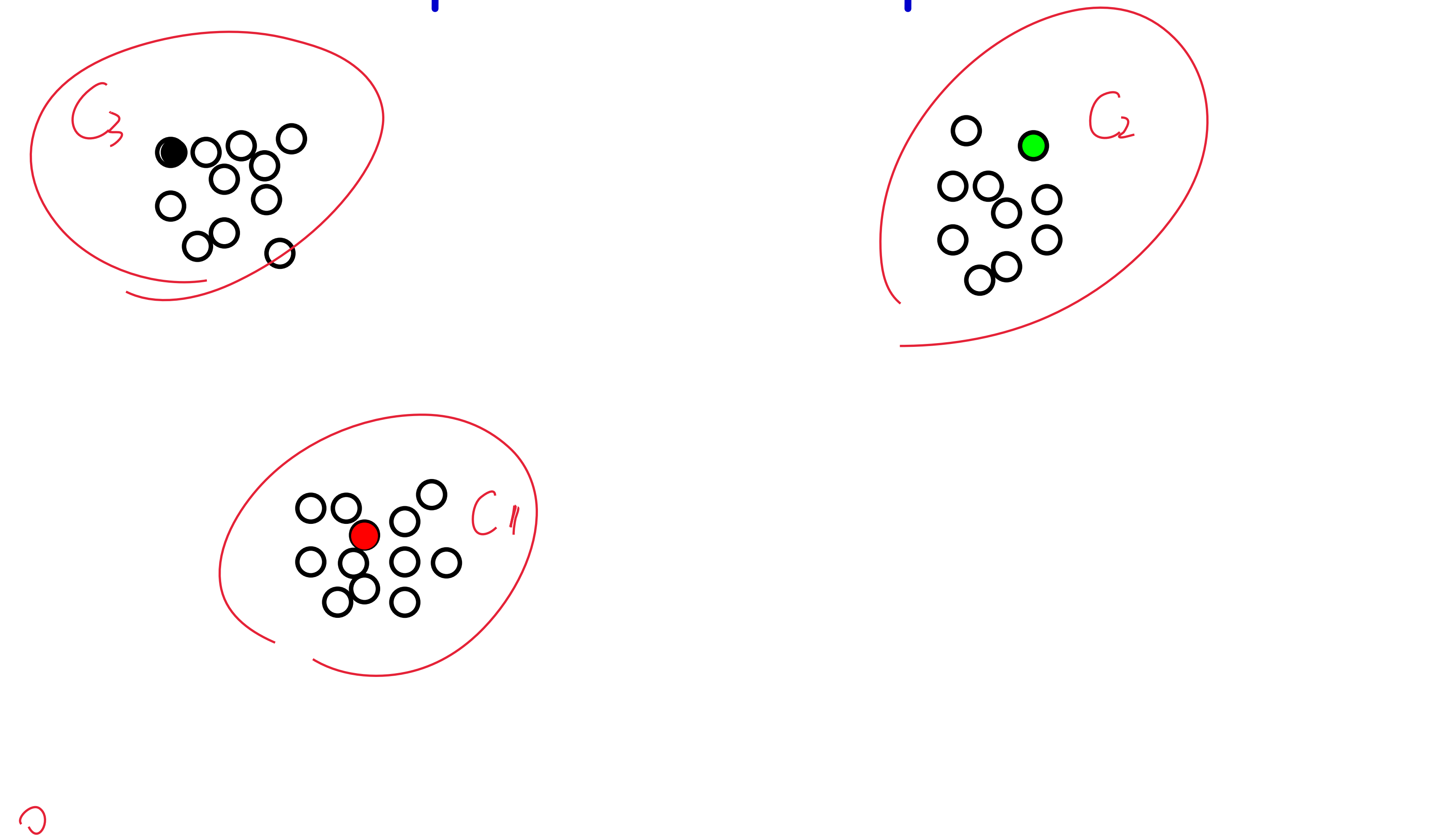
Another Initialization Idea: Furthest Point Heuristic

Choose $\mathbf{c_1}$ arbitrarily (or at random).

- For $\mathrm{j} = 2, \dots, \mathrm{k}$
  - Pick $\mathbf{c_j}$ among datapoints $\mathbf{x^1}, \mathbf{x^2}, \dots, \mathbf{x^d}$ that is farthest from previously chosen $\mathbf{c_1}, \mathbf{c_2}, \dots, \mathbf{c}_{\boldsymbol{j-1}}$

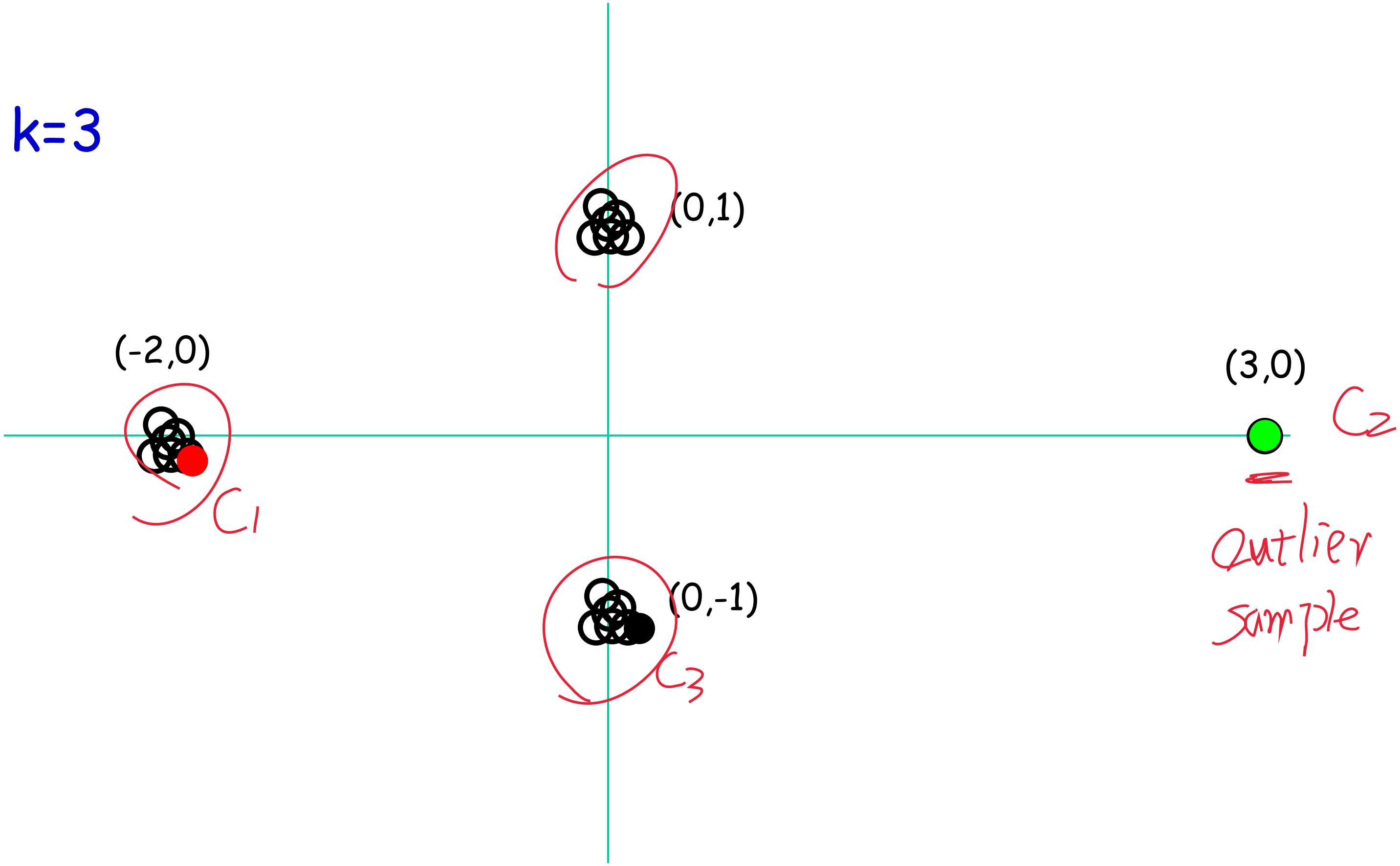
Fixes the Gaussian problem. But it can be thrown off by outliers….

Furthest point heuristic does well on previous example



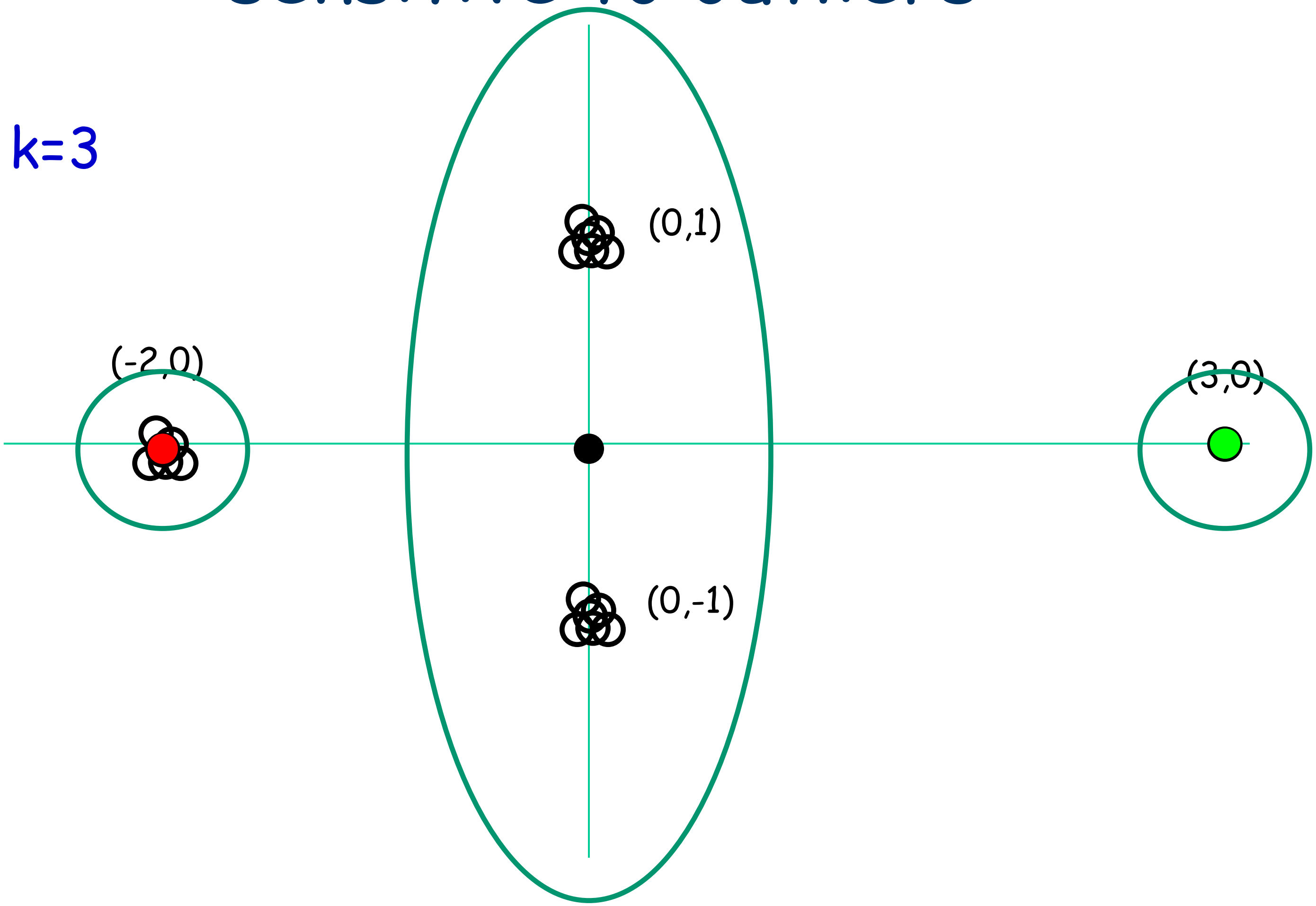
Furthest point initialization heuristic sensitive to outliers

Assume k=3



Furthest point initialization heuristic sensitive to outliers

Assume k=3



K-means++ Initialization: D^2 sampling [AV07]

- Interpolate between random and furthest point initialization
- Let $D(x)$ be the distance between a point x and its nearest center. Chose the next center proportional to $\mathrm{D}^2(\mathbf{x})$.

- Choose $\mathbf{c_1}$ at random.
- For $\mathrm{j} = 2, \ldots, \mathrm{k}$
  - Pick $\mathbf{c_j}$ among $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{\mathrm{n}}$ according to the distribution

$$\mathbf{Pr}(\mathbf{c_j} = \mathbf{x^i}) \propto \mathbf{min}_{\mathbf{j'<j}} \left|\left|\mathbf{x^i} - \mathbf{c_{j'}}\right|\right|^2 \quad \mathrm{D}^2(\mathbf{x^i})$$

Theorem: K-means++ always attains an $O(\log k)$ approximation to optimal k-means solution in expectation.

Running Lloyd's can only further improve the cost.

K-means++ Idea: D^2 sampling

- Interpolate between random and furthest point initialization
- Let D(x) be the distance between a point x and its nearest center. Chose the next center proportional to $\mathrm{D}^{\alpha}(\mathbf{x})$.

ℓ_α-norm

$$\|a\|_\alpha = \left(\sum_{j=1}^{d} |a_j|^\alpha\right)^{\frac{1}{\alpha}}$$

$$\|a\|_0 = \sum_{j=1}^{d} \mathbb{I}(a_j \neq 0)$$

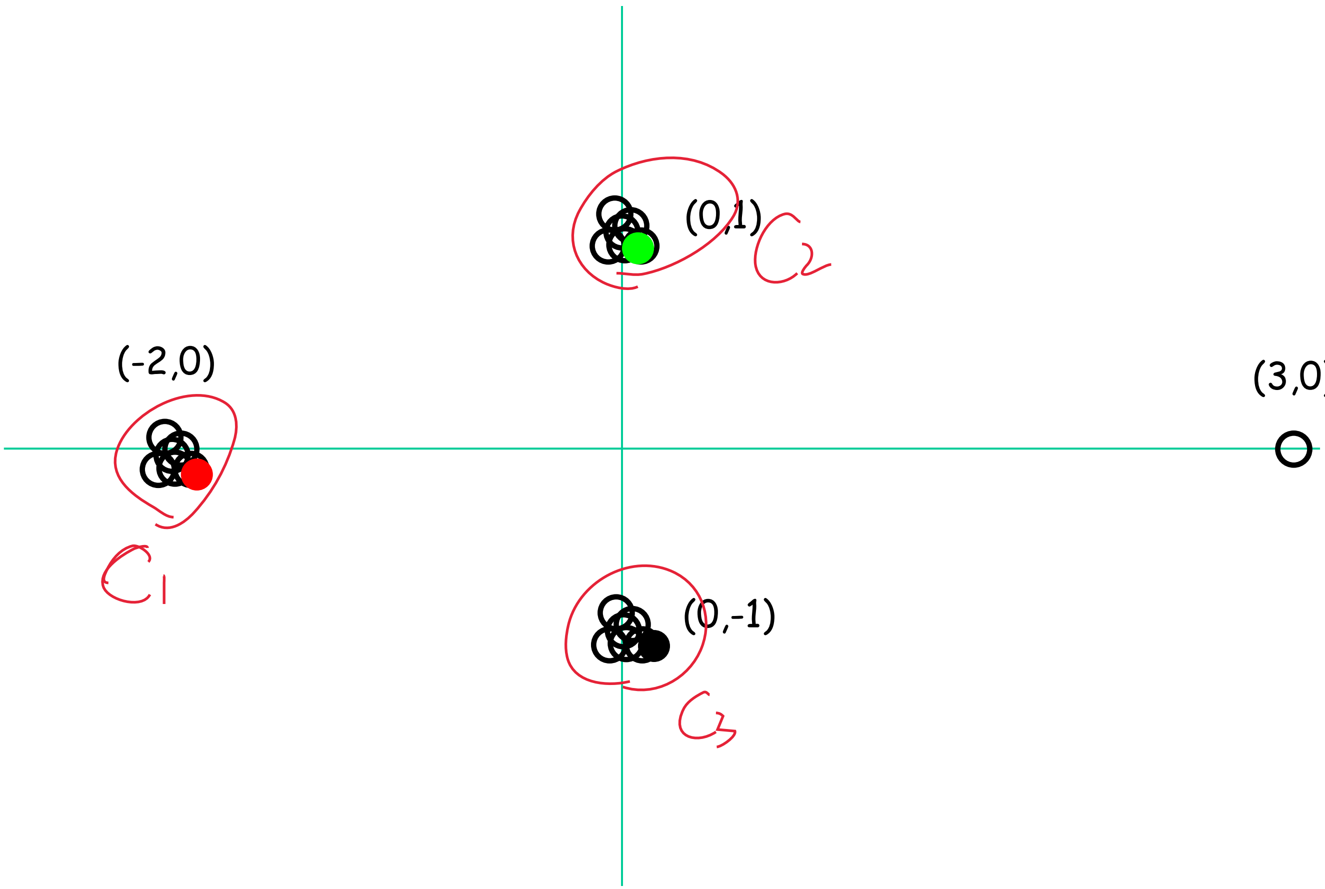
$$\|a\|_1 = \sum_{j=1}^{d} |a_j|$$

$$\|a\|_\infty = \max_{j=1,2,\dots,d} |a_j|$$

  - $\alpha = 0$, random sampling
  - $\alpha = \infty$, furthest point (Side note: it actually works well for k-center)
  - $\alpha = 2$, k-means++

Side note: $\alpha = 1$, works well for k-median

K-means ++ Fix



K-means++/ Lloyd's Running Time

- K-means ++ initialization: O(nd) and one pass over data to select next center. So O(nkd) time in total.

- Lloyd's method

$Q(R,C) = \|X - RC\|_F^2$

Repeat until there is no change in the cost.

Step 1 • For each j: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c_j}\}$

Step 2 • For each j: $\mathbf{c_j} \leftarrow$mean of C_j

Each round takes time O(nkd). (T: # iter.)

$O(Tnkd)$.

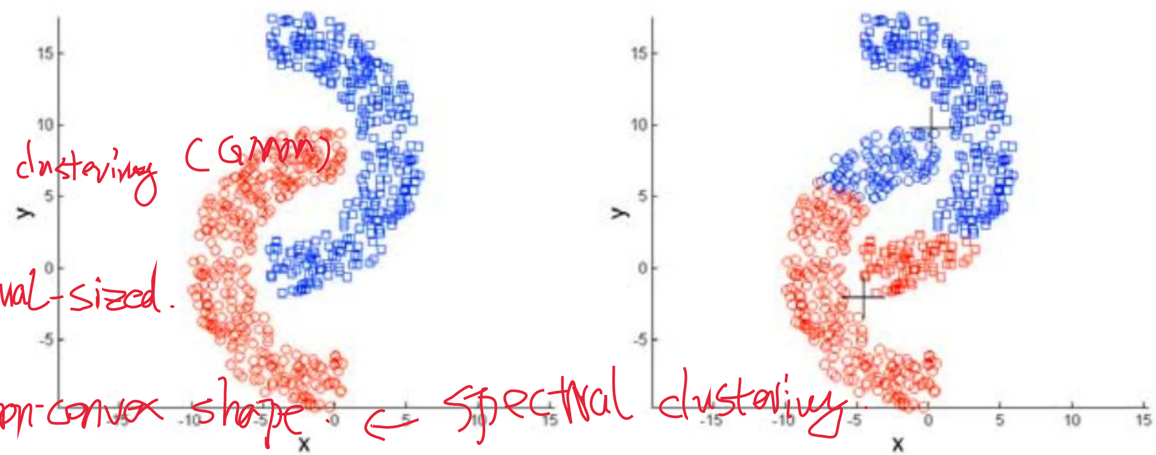
In total: $O(nkd) + O(Tnkd)$

- Exponential # of rounds in the worst case [AV07].

- Expected polynomial time in the smoothed analysis model!

K-means++/ Lloyd's Summary

- K-means++ always attains an O(log k) approximation to optimal k-means solution in expectation.
- Running Lloyd's can only further improve the cost.
- Exponential # of rounds in the worst case [AV07].
- Expected polynomial time in the smoothed analysis model!
- Does well in practice.



Limitations:

① hard clustering ← soft ← prob. clustering (GMM)

② work well only clusters are equal-sized.

③ work badly if clusters have non-convex shape. ← spectral clustering.

Kernel K-means

kernel trick

$K(\langle x_i, x_j\rangle) = \langle \phi(x_i), \phi(x_j)\rangle$

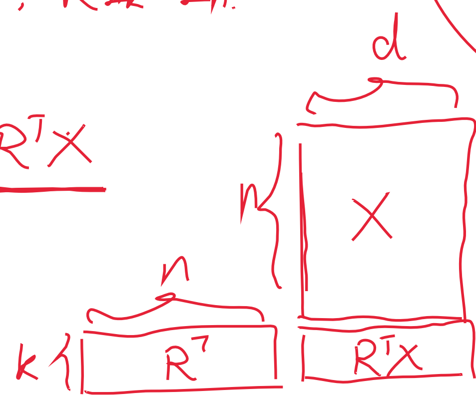
$$\min_{R,C} \|X - RC\|_F^2 = Q(R,C).$$

$$\text{s.t. } R \in \{0,1\}^{n \times k},\ R\mathbb{1}_k = \mathbb{1}_n.$$

$$\|A\|_F^2 = \langle A, A\rangle = Tr(A^T A) = Tr(AA^T)$$

$$Tr(ABC) = Tr(CAB) = Tr(BCA)$$

$$C = (R^T R)^{-1} R^T X = D R^T X$$



$$R D \underbrace{R^T R}_{D^{-1}} D R^T = R D R^T$$

$$\min_R Q(R) = \|X - RDR^T X\|_F^2$$

$$= \langle X - RDR^T X, X - RDR^T X\rangle$$

$$= \langle X, X\rangle - 2\langle X, RDR^T X\rangle + \langle RDR^T X, RDR^T X\rangle$$

$$= Tr(X^T \underbrace{RDR^T RDR^T}_{RDR^T} X)$$

$$= \langle X, X\rangle - \max_R \langle X, RDR^T X\rangle$$

$$\max_R \langle X, RDR^T X\rangle \quad \text{s.t. } R \in \{0,1\}^{n \times k},\ R\mathbb{1}_k = \mathbb{1}_n$$

$$\langle X, RDR^T X\rangle = Tr(X^T RDR^T X) = Tr(X^T F F^T X) = Tr(F^T X X^T F)$$

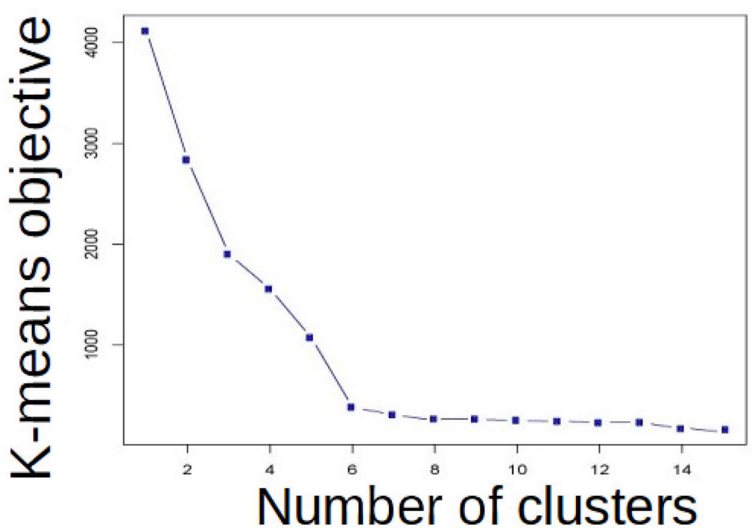
$(F = R(R^T R)^{-\frac{1}{2}}$: weighted indicator$)$

Gram

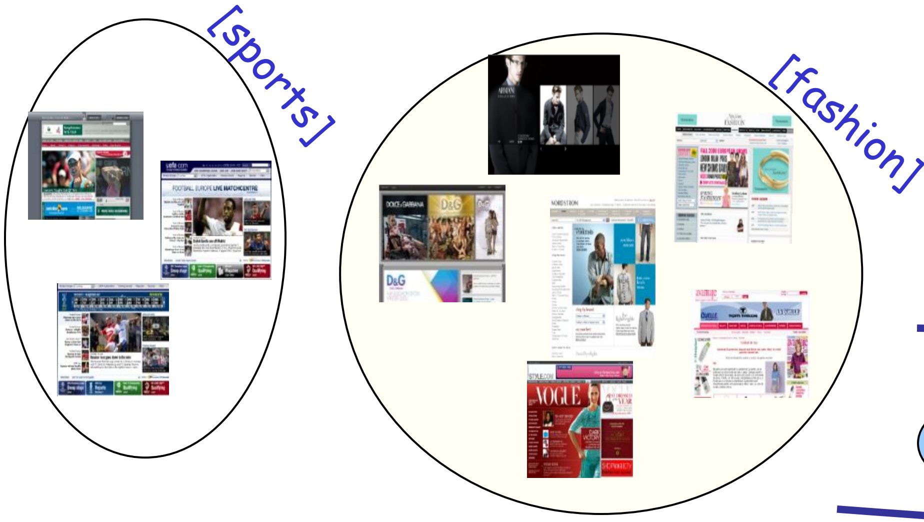
$$\max_F Tr(F^T X X^T F) \quad \text{s.t. } F^T F = I$$

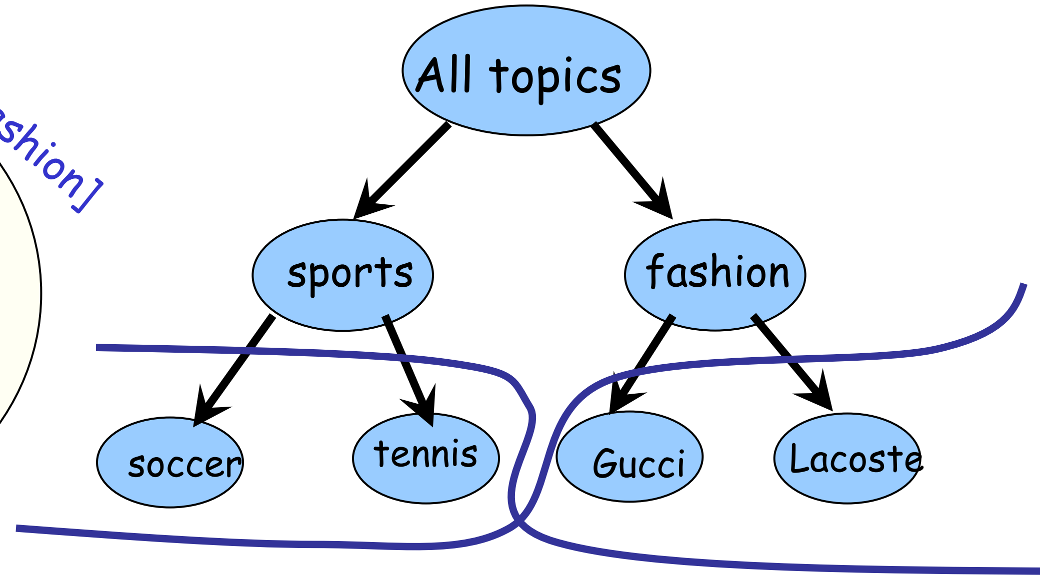
What value of k???

- Heuristic: Find large gap between k -1-means cost and k-means cost.
- According to information criteria (AIC, BIC, etc.)
- Try hierarchical clustering.



Hierarchical Clustering



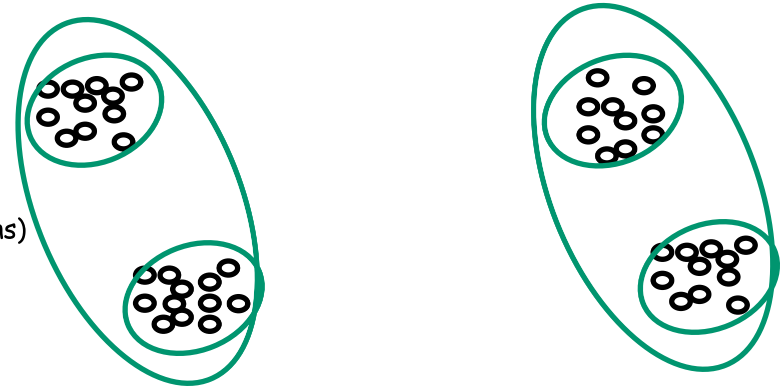


- A hierarchy might be more natural.
- Different users might care about different levels of granularity or even prunings.

Hierarchical Clustering

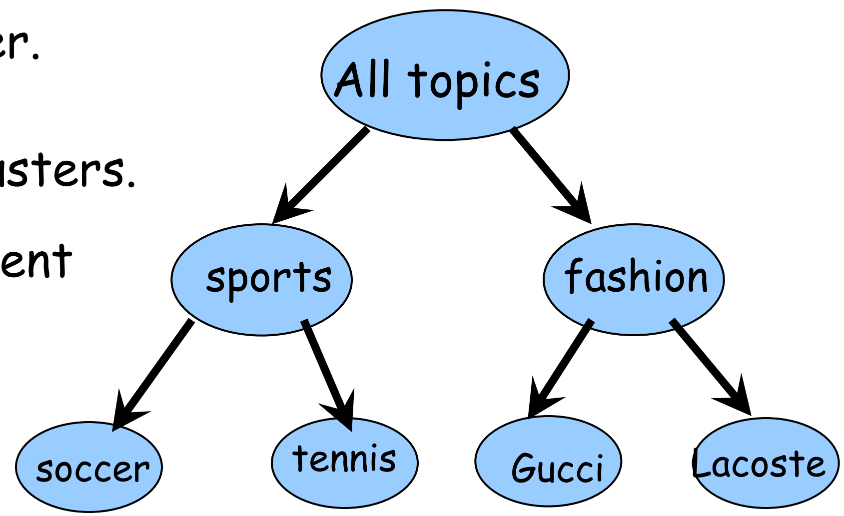
Top-down (divisive)

- Partition data into 2-groups (e.g., 2-means)
- Recursively cluster each group.



Bottom-Up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the “closest” two clusters.
- Different defs of “closest” give different algorithms.

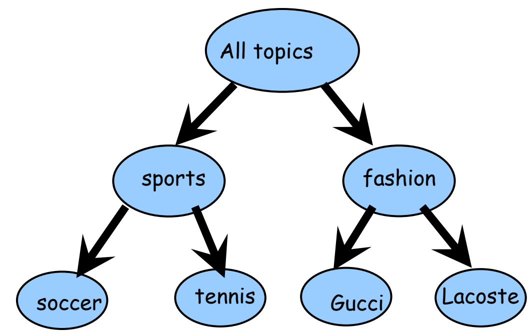


Bottom-Up (agglomerative)

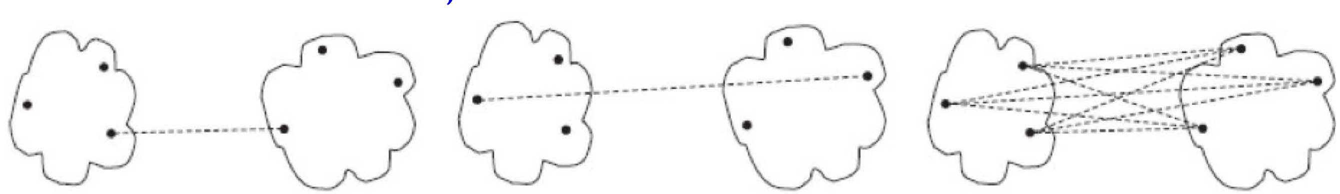
Have a distance measure on pairs of objects.

d(x,y) – distance between x and y

E.g., # keywords in common, edit distance, etc



- Single linkage: $$\mathrm{dist}(A,B) = \min_{x \in A, x' \in B'} \mathrm{dist}(x, x')$$
- Complete linkage: $$\mathrm{dist}(A,B) = \max_{x \in A, x' \in B'} \mathrm{dist}(x, x')$$
- Average linkage: $$\mathrm{dist}(A,B) = \operatorname*{avg}_{x \in A, x' \in B'} \mathrm{dist}(x, x')$$
- Ward's method



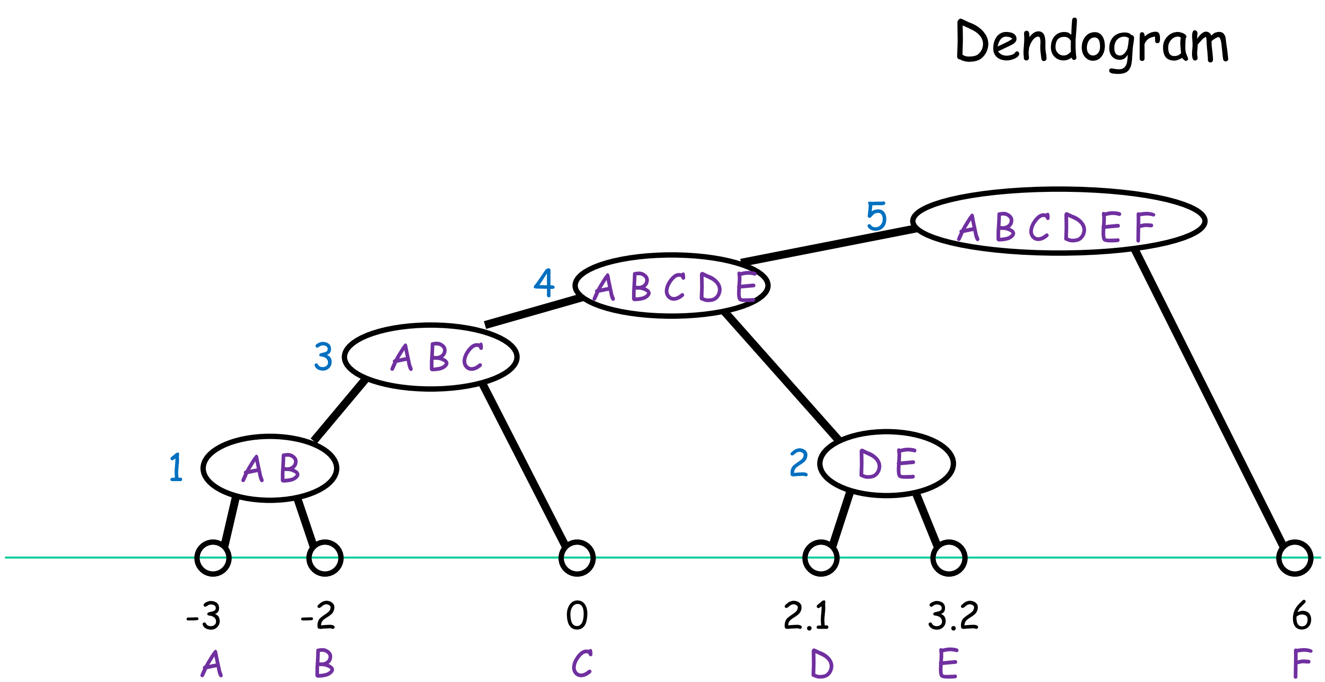
(a) MIN (single link.) (b) MAX (complete link.) (c) Group average.

Single Linkage

Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Single linkage: $\mathrm{dist}(\mathrm{A}, B) = \min_{\mathrm{x} \in \mathrm{A}, \mathrm{x}' \in B} \mathrm{dist}(\mathrm{x}, \mathrm{x}')$



Single Linkage

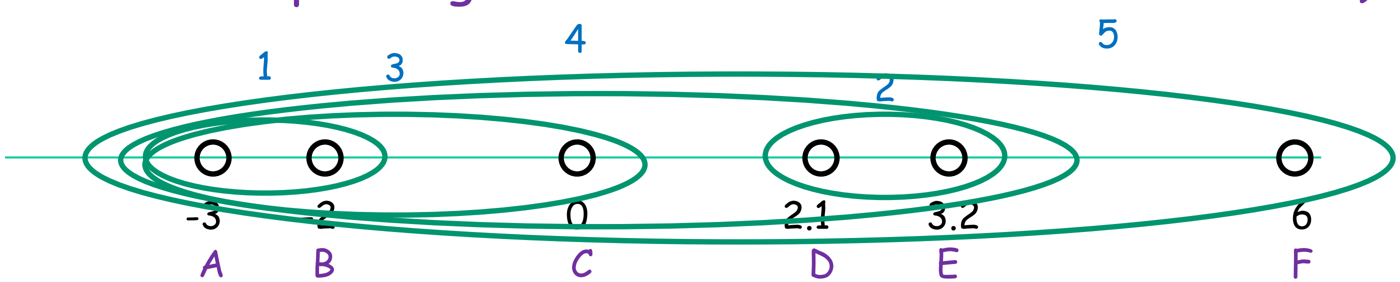
Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Single linkage: $\mathrm{dist}(A, B) = \min_{x \in A, x' \in B} \mathrm{dist}(x, x')$

One way to think of it: at any moment, we see connected components of the graph where connect any two pts of distance < r.

Watch as r grows (only n-1 relevant values because we only we merge at value of r corresponding to values of r in different clusters).



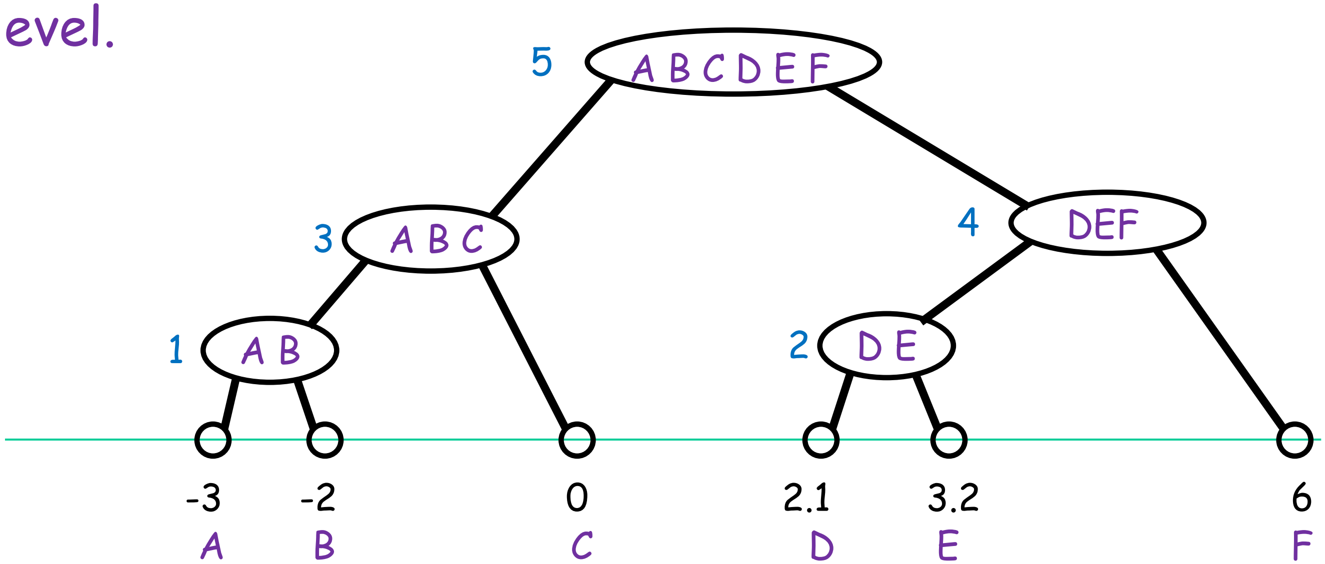
Complete Linkage

Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Complete linkage: $\mathrm{dist}(A,B) = \max_{x \in A, x' \in B} \mathrm{dist}(x, x')$

One way to think of it: keep max diameter as small as possible at any level.



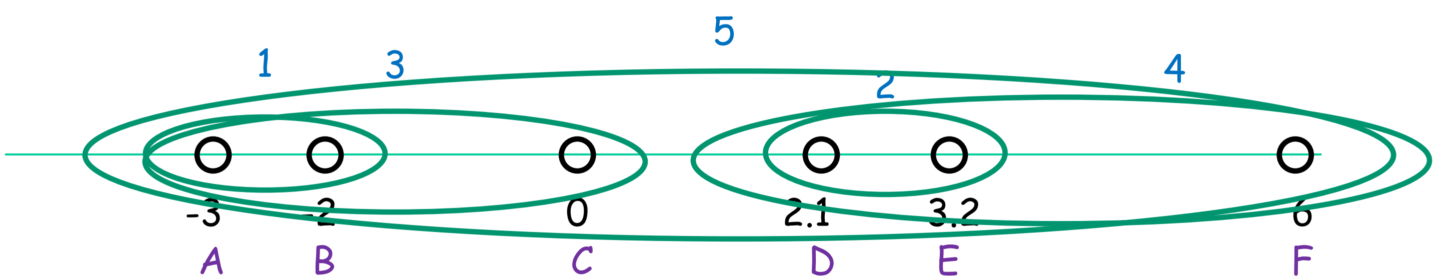
Complete Linkage

Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Complete linkage: $\mathrm{dist}(A, B) = \max_{x \in A, x' \in B} \mathrm{dist}(x, x')$

One way to think of it: keep max diameter as small as possible.



Ward's Method

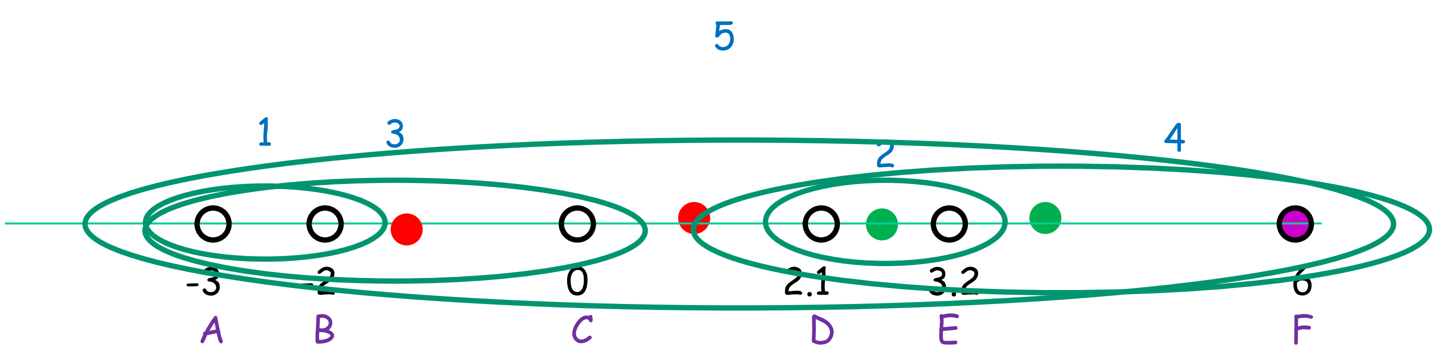
Bottom-up (agglomerative)

- Start with every point in its own cluster.
- Repeatedly merge the "closest" two clusters.

Ward's method: $\mathrm{dist}(C, C') = \frac{|C| \cdot |C'|}{|C| + |C'|} \|\mathrm{mean}(C) - \mathrm{mean}(C')\|^2$

Merge the two clusters such that the increase in k-means cost is as small as possible.

Works well in practice.



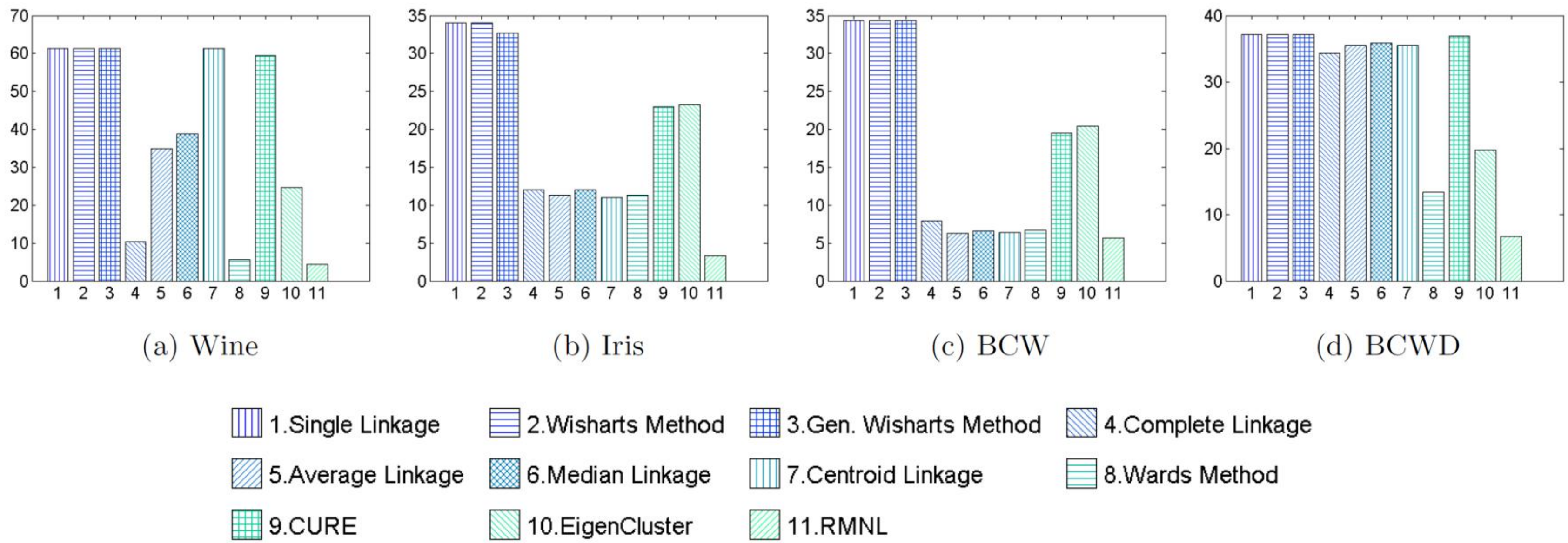
Running time

- Each algorithm starts with N clusters, and performs N-1 merges.
- For each algorithm, computing $dist(C, C')$ can be done in time $O(|C| \cdot |C'|)$. (e.g., examining $dist(x, x')$ for all $x \in C, x' \in C'$)
- Time to compute all pairwise distances and take smallest is $O(N^2)$.
- Overall time is $O(N^3)$.

In fact, can run all these algorithms in time $O(N^2 \log N)$.

See: Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. http://www-nlp.stanford.edu/IR-book/

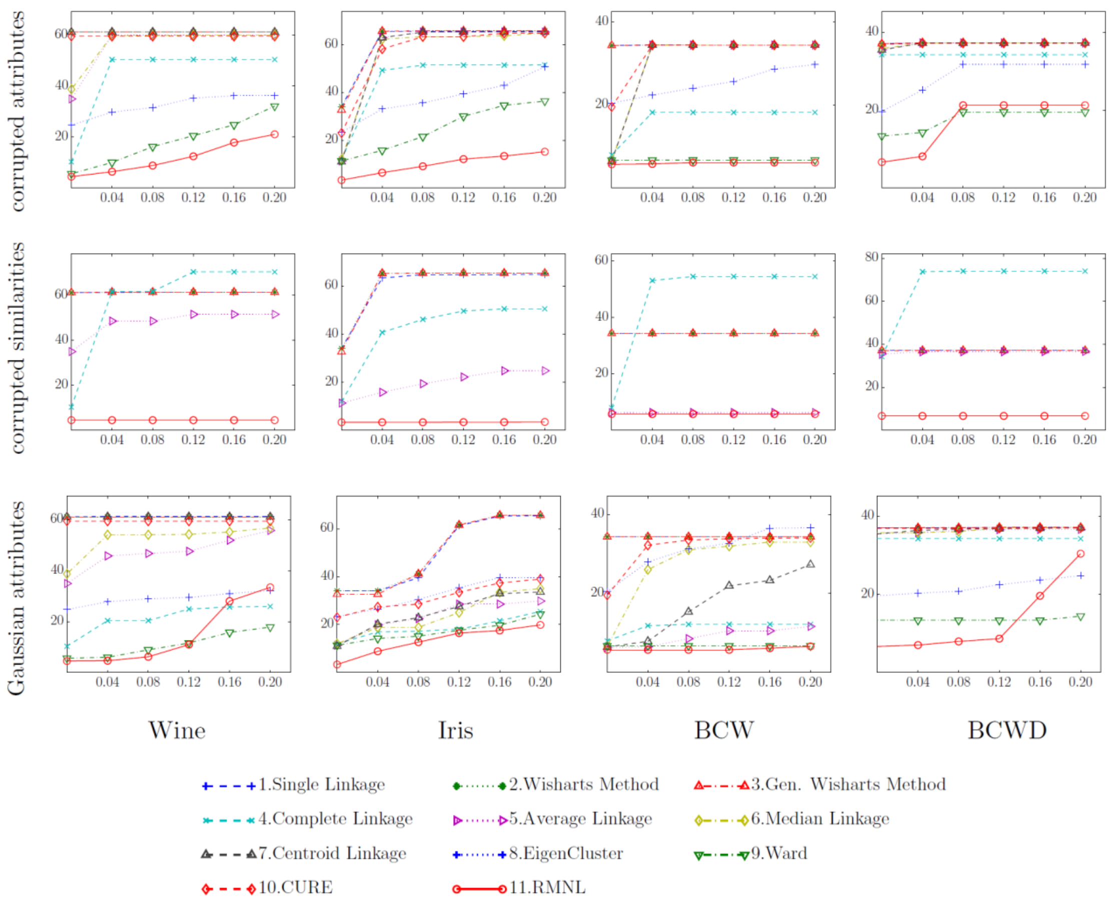
Hierarchical Clustering Experiments
[BLG, JMLR'15]



Ward's method does the best among classic techniques.

Hierarchical Clustering Experiments

[BLG, JMLR'15]



Ward's method does the best among classic techniques.

What You Should Know

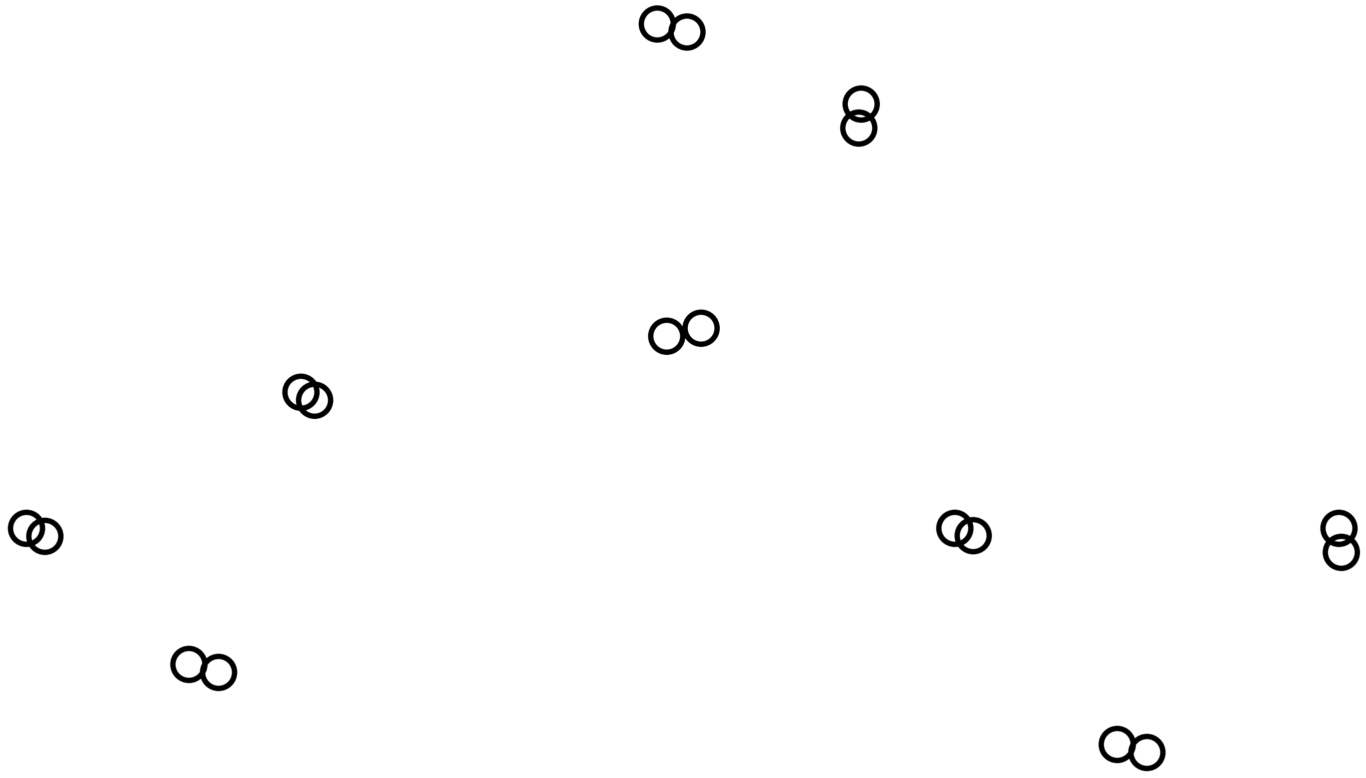
- Partitional Clustering. k-means and k-means ++
  - Lloyd's method
  - Initialization techniques (random, furthest traversal, k-means++)

- Hierarchical Clustering.
  - Single linkage, Complete Linkge, Ward's method

Additional Slides

Smoothed analysis model

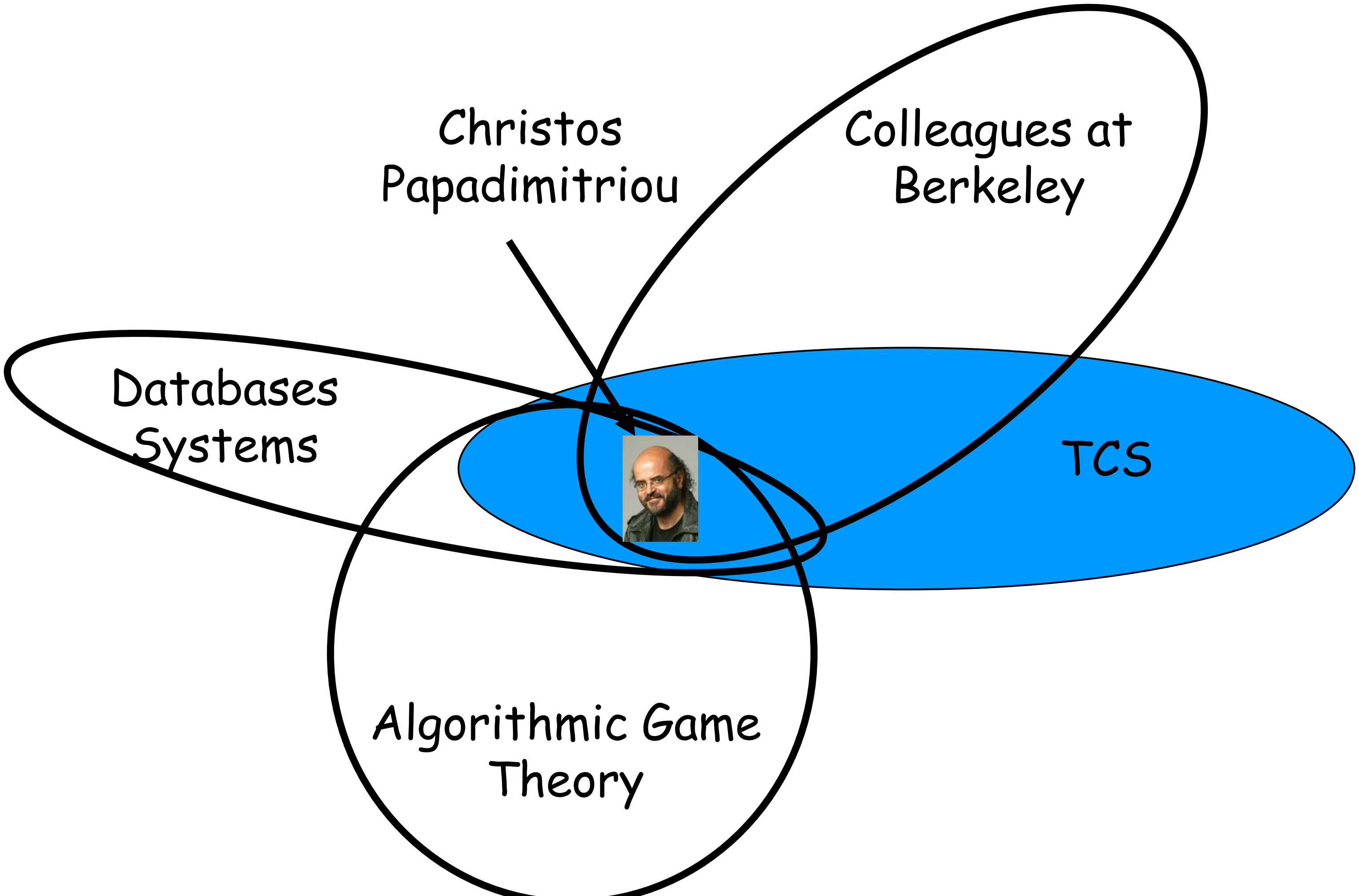
- Imagine a worst-case input.
- But then add small Gaussian perturbation to each data point.



Smoothed analysis model

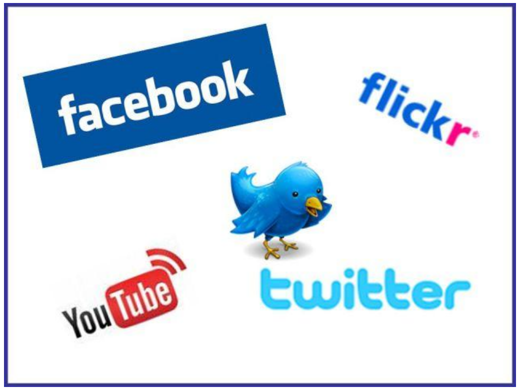
- Imagine a worst-case input.
- But then add small Gaussian perturbation to each data point.
- Theorem [Arthur-Manthey-Roglin 2009]:
  - E[number of rounds until Lloyd's converges] if add Gaussian perturbation with variance σ^2 is polynomial in n, $1/\sigma$.
  - The actual bound is : $O\left(\frac{n^{34}k^{34}d^8}{\sigma^6}\right)$
- Might still find local opt that is far from global opt.

Overlapping Clusters: Communities



Overlapping Clusters: Communities

- Social networks



- Professional networks

- Product Purchasing Networks, Citation Networks, Biological Networks, etc

Overlapping Clusters: Communities