

Clustering.

Unsupervised Learning

Maria-Florina Balcan

11/05/2018

Clustering, Informal Goals

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

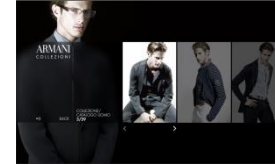
Question: When and why would we want to do this?

Useful for:

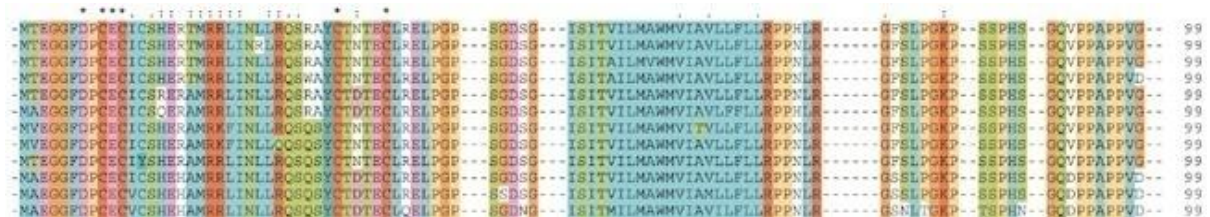
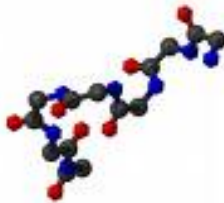
- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
 - Representing high-dimensional data in a low-dimensional space (e.g., for visualization purposes).

Applications (Clustering comes up everywhere...)

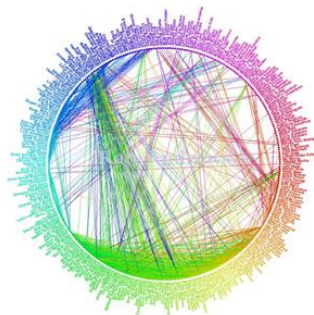
- Cluster news articles or web pages or search results by topic.



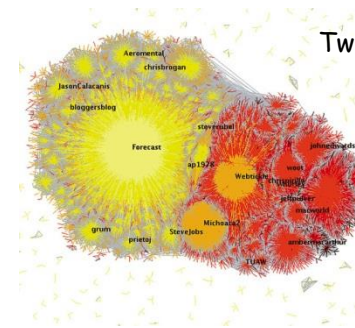
- Cluster protein sequences by function or genes according to expression profile.



- 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)



- And many many more applications....

Clustering

Today:

- Objective based clustering
 - K-means clustering
- Hierarchical clustering

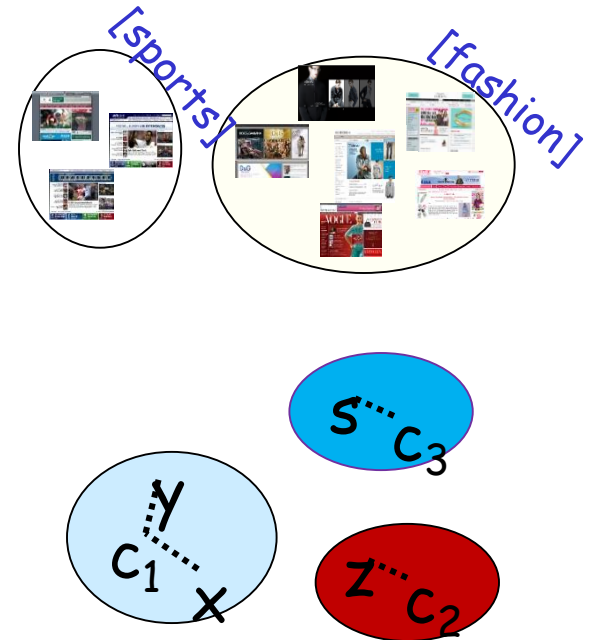
Objective Based Clustering

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 $\sum_{i=1}^n \min_{j \in \{1, \dots, k\}} d^2(x^i, c_j)$
- **k-median:** find center pts c_1, c_2, \dots, c_k to
minimize $\sum_{i=1}^n \min_{j \in \{1, \dots, k\}} d(x^i, c_j)$
- **K-center:** find partition to minimize the maxim radius



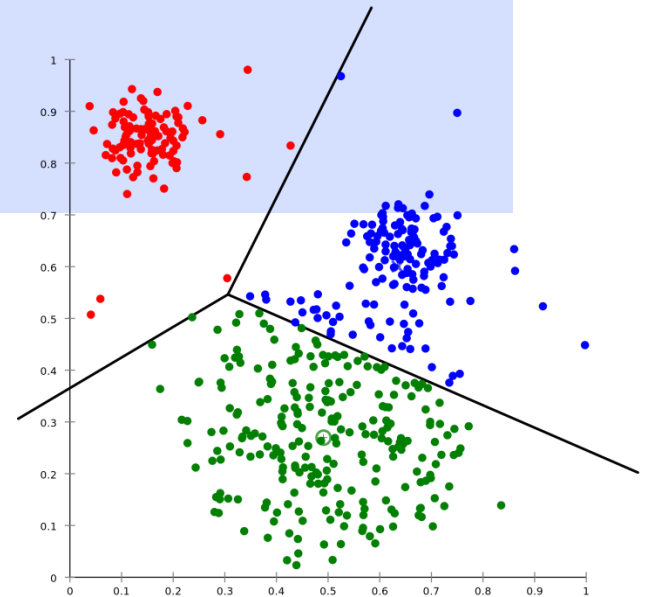
Euclidean k-means Clustering

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

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

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

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



Euclidean k-means Clustering

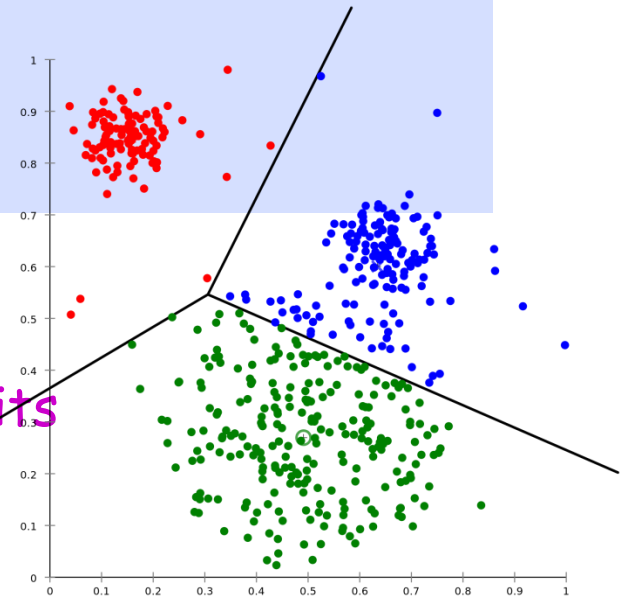
Input: A set of n datapoints $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ in \mathbb{R}^d
target #clusters k

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

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

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

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



Euclidean k-means Clustering

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

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

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

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

Computational complexity:

NP hard: even for $k = 2$ [Dagupta'08] or
 $d = 2$ [Mahajan-Nimbhorkar-Varadarajan09]

There are a couple of easy cases...



An Easy Case for k-means: $k=1$

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

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

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

Idea: bias/variance like decomposition

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

Avg k-means cost wrt \mathbf{c}

Avg k-means cost wrt $\boldsymbol{\mu}$

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

Another Easy Case for k-means: $d=1$

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

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

Extra-credit homework question

Hint: dynamic programming in time $O(n^2k)$.

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 \mathbb{R}^d

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

Repeat until there is no further change in the cost.

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

Common Heuristic in Practice: The Lloyd's method

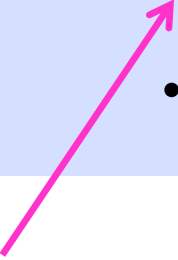
[Least squares quantization in PCM, Lloyd, IEEE Transactions on Information Theory, 1982]

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


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

Repeat until there is no further change in the cost.

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



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



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

Common Heuristic: The Lloyd's method

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

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

Repeat until there is no further change in the cost.

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

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)

Initialization for the Lloyd's method

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

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

Repeat until there is no further change in the cost.

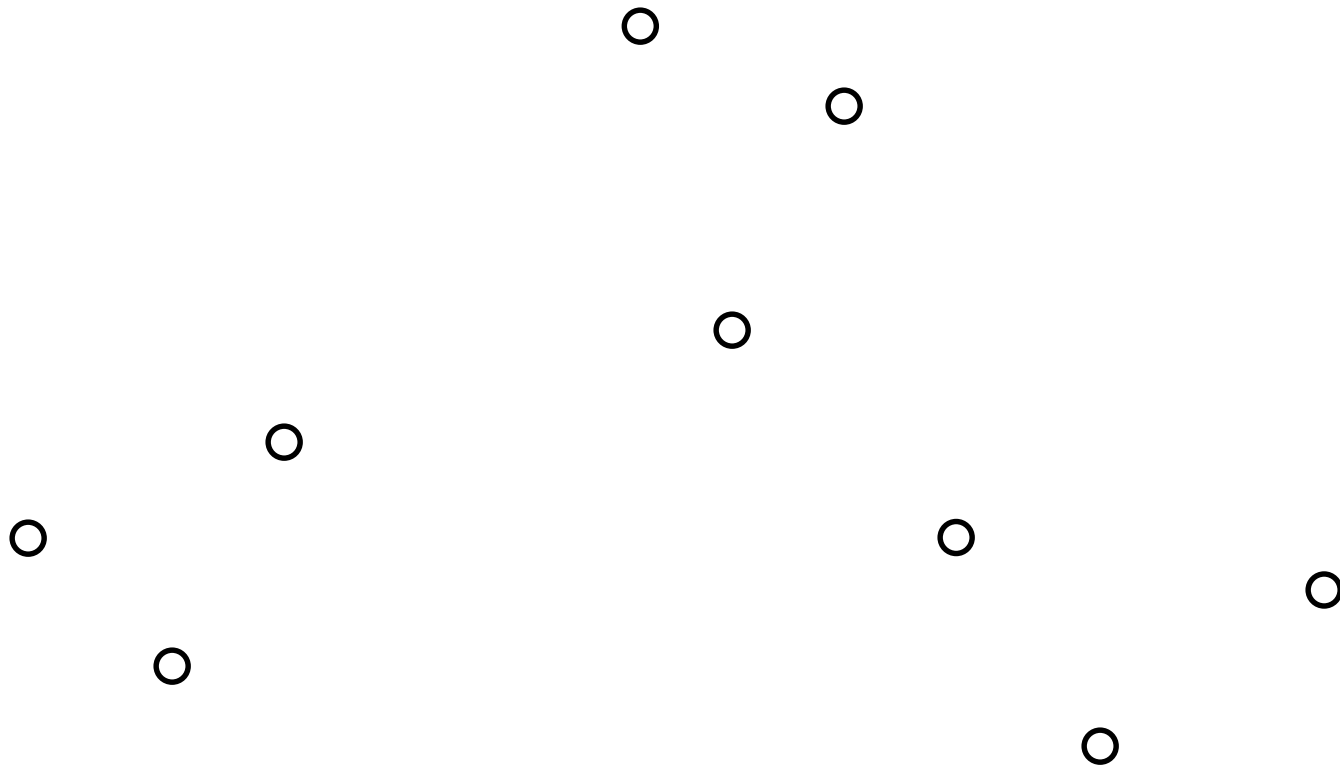
- For each j : $C_j \leftarrow \{x \in S \text{ whose closest center is } \mathbf{c}_j\}$
- For each j : $\mathbf{c}_j \leftarrow \text{mean of } C_j$

- **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

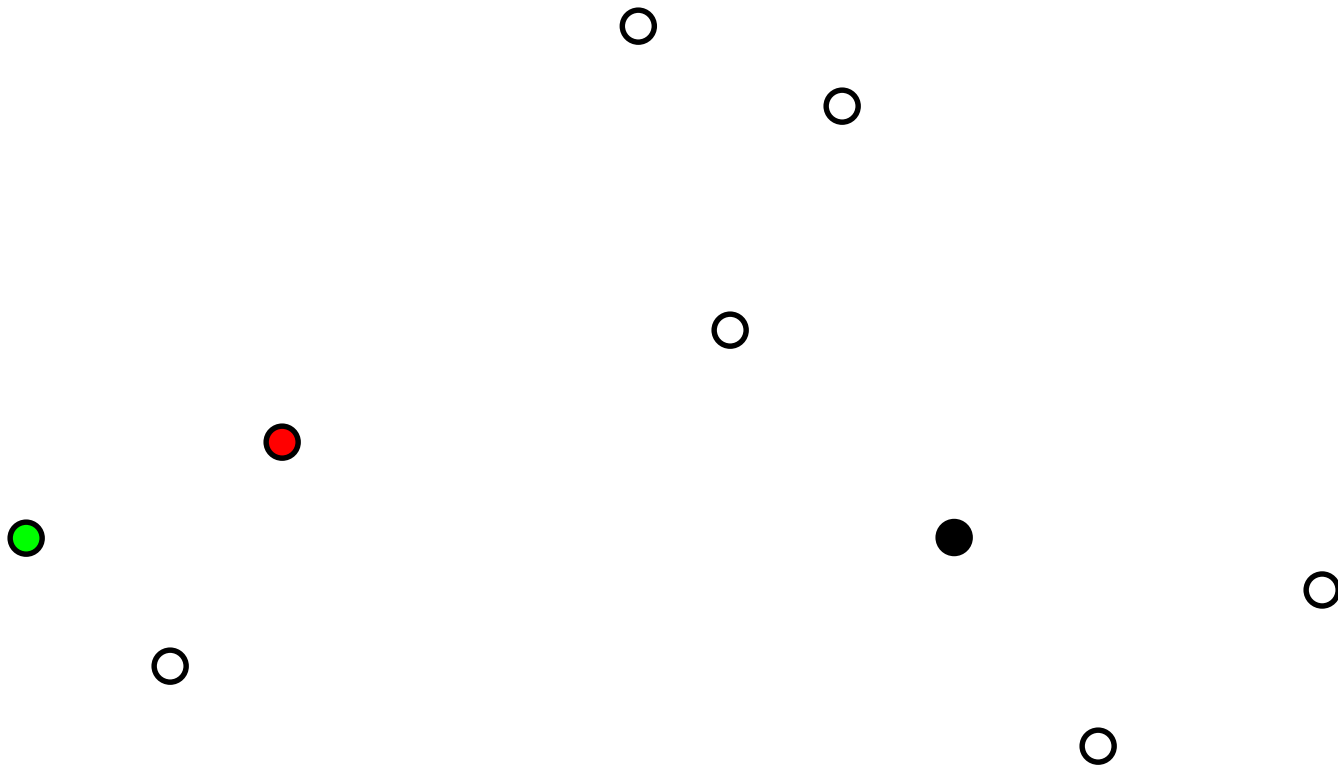
Lloyd's method: Random Initialization

Example: Given a set of datapoints



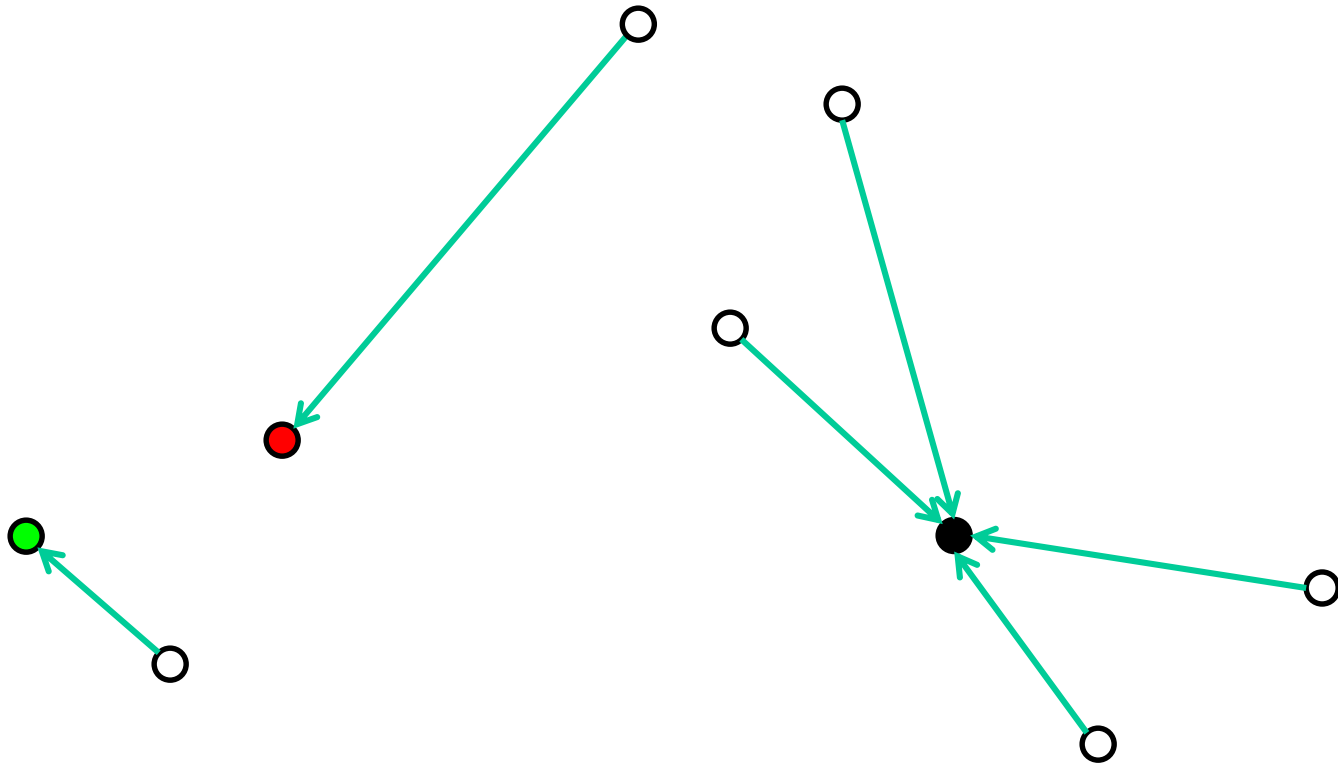
Lloyd's method: Random Initialization

Select initial centers at random



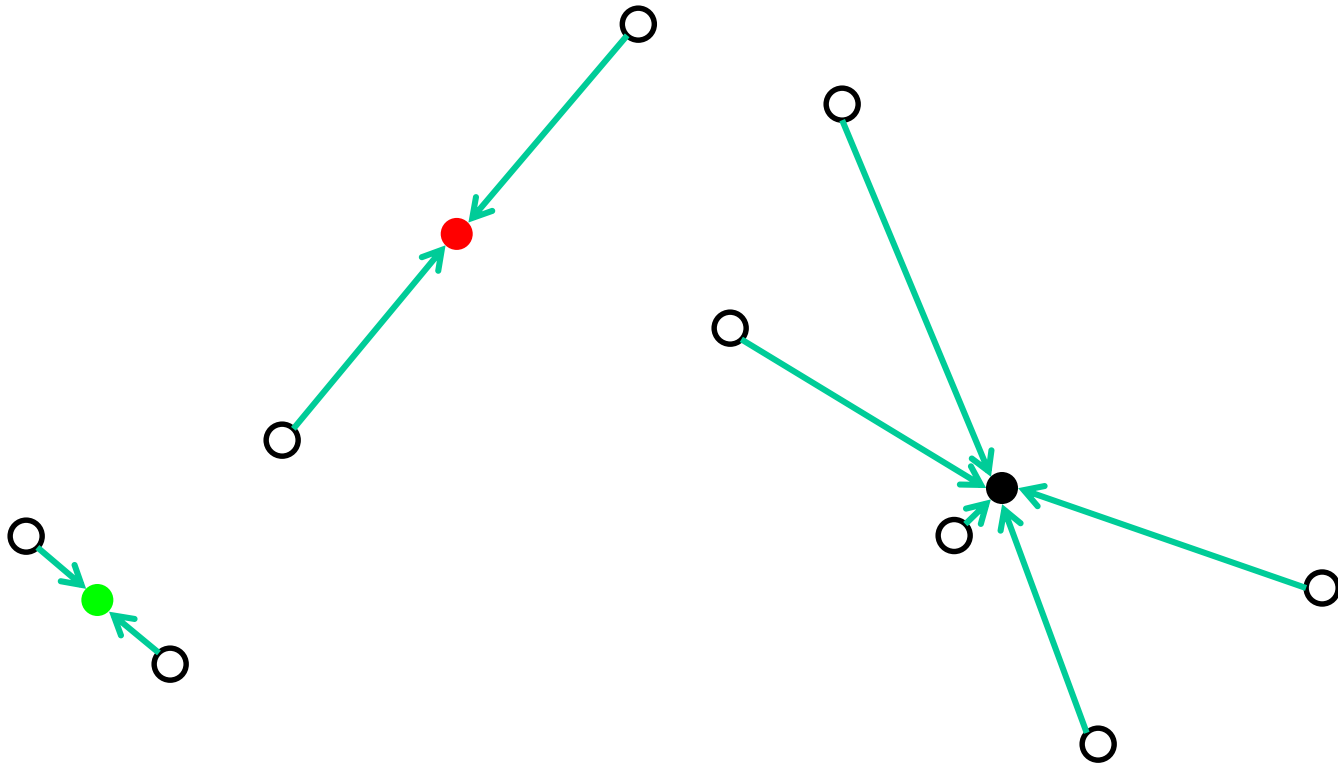
Lloyd's method: Random Initialization

Assign each point to its nearest center



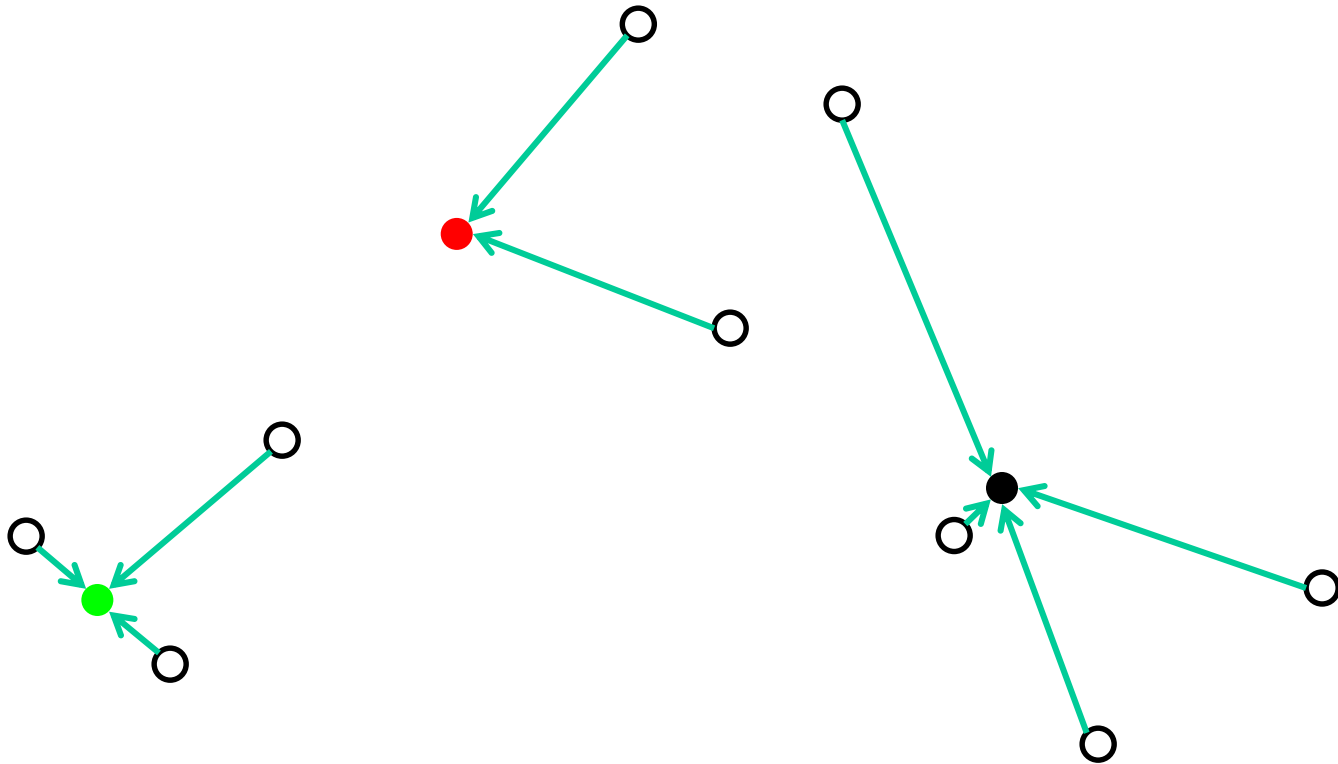
Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering



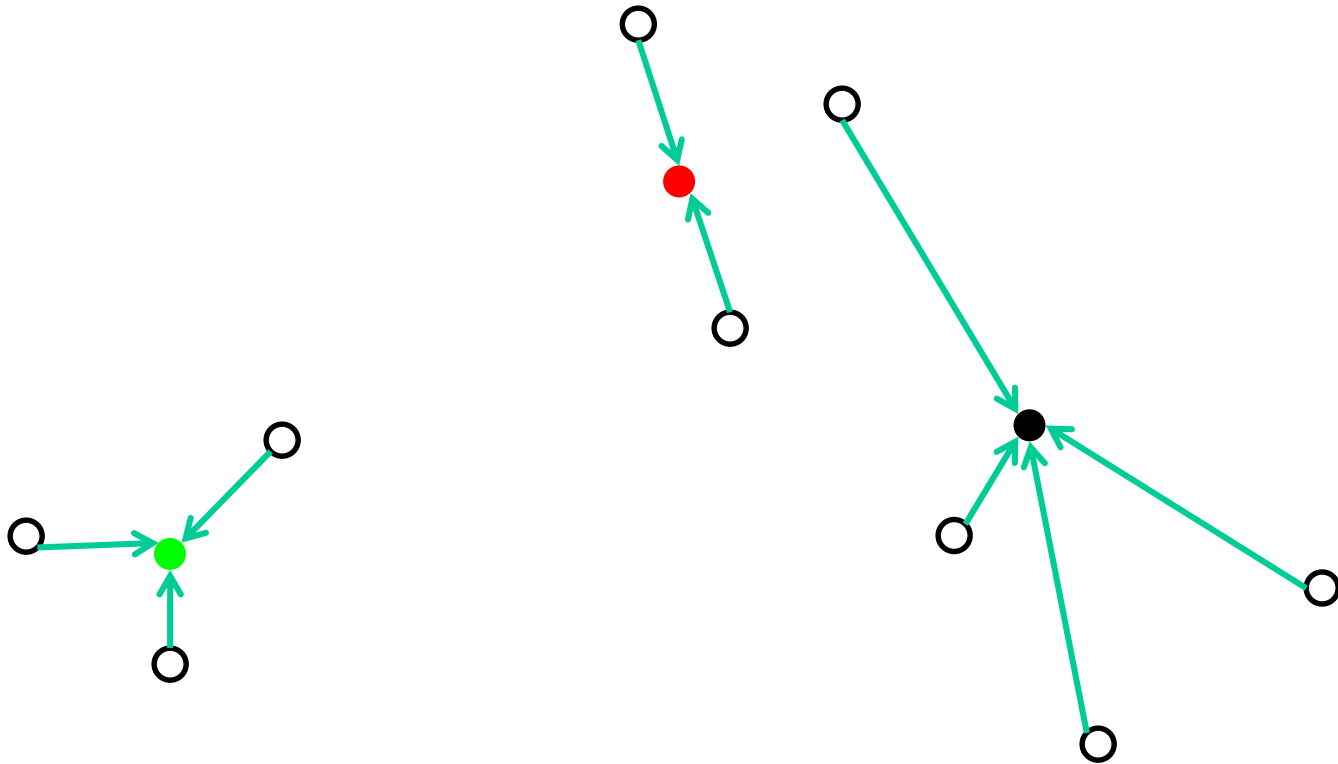
Lloyd's method: Random Initialization

Assign each point to its nearest center



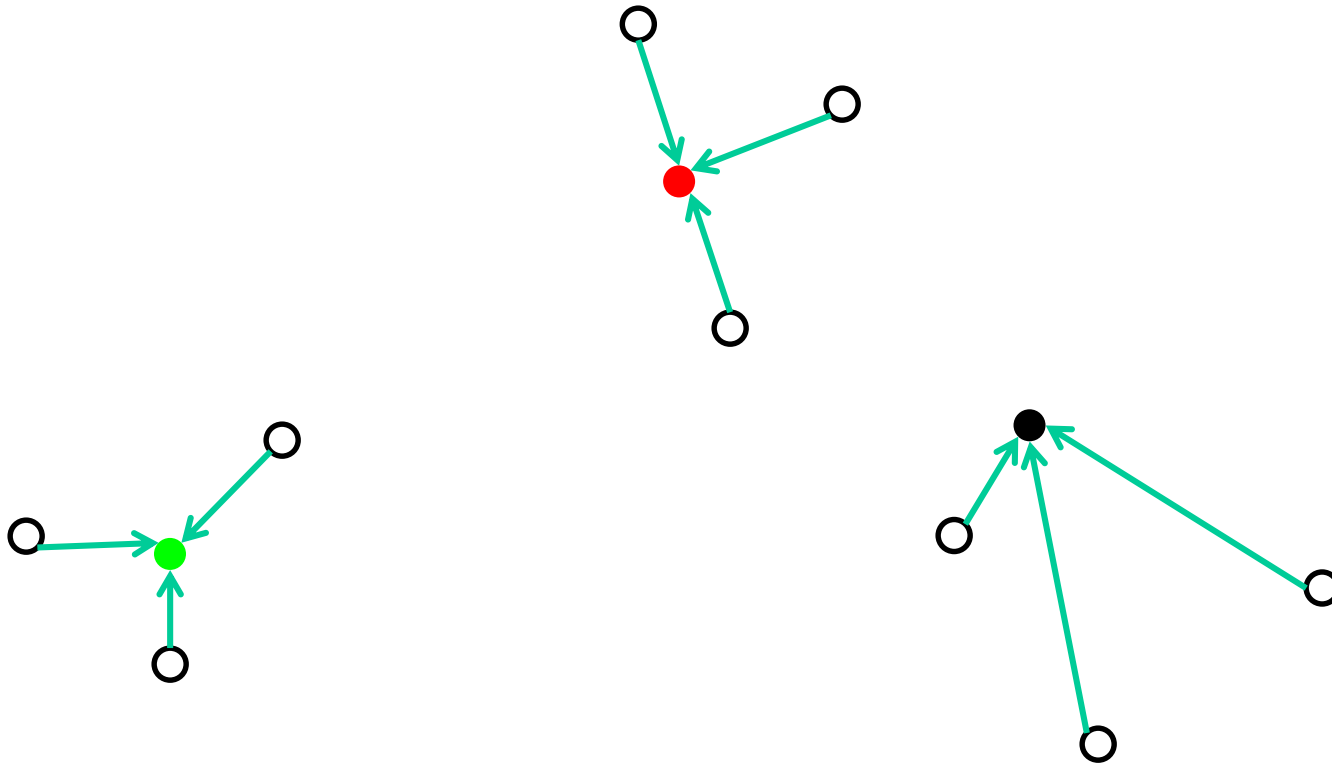
Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering



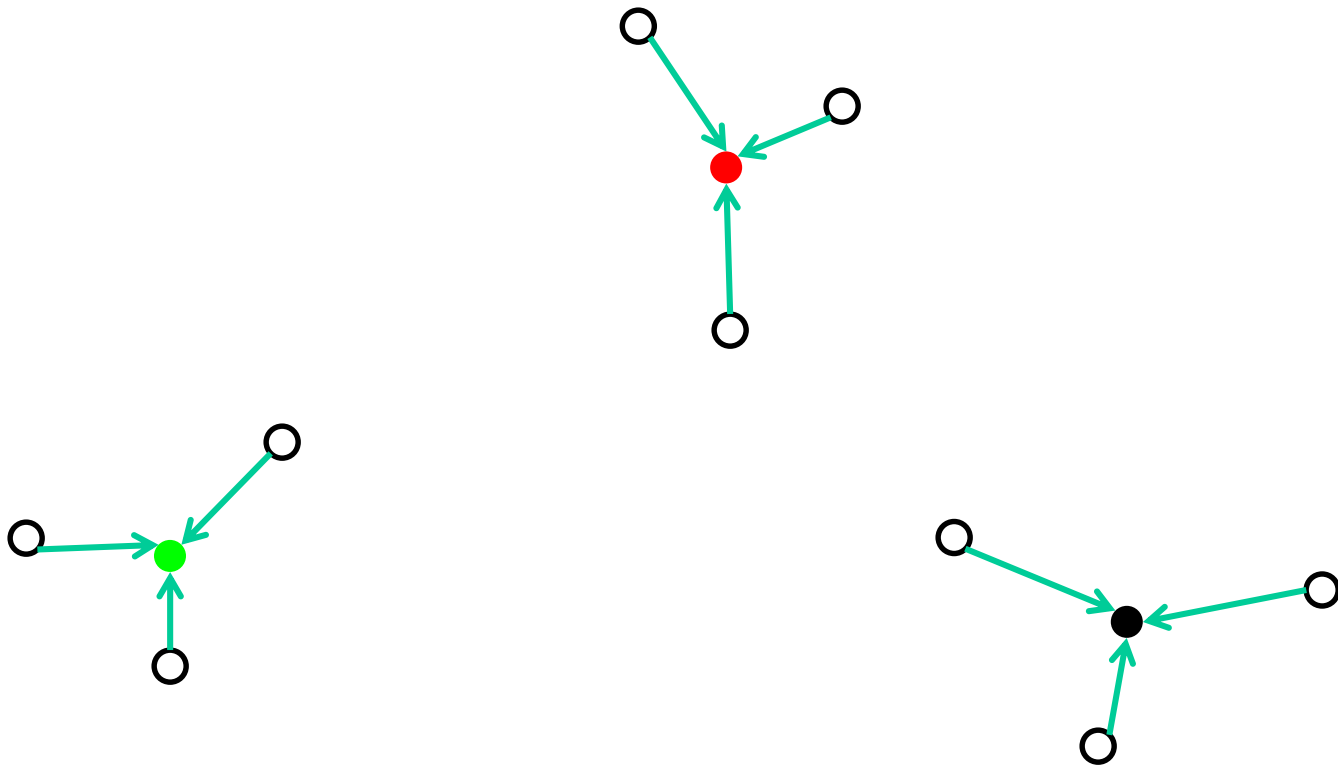
Lloyd's method: Random Initialization

Assign each point to its nearest center



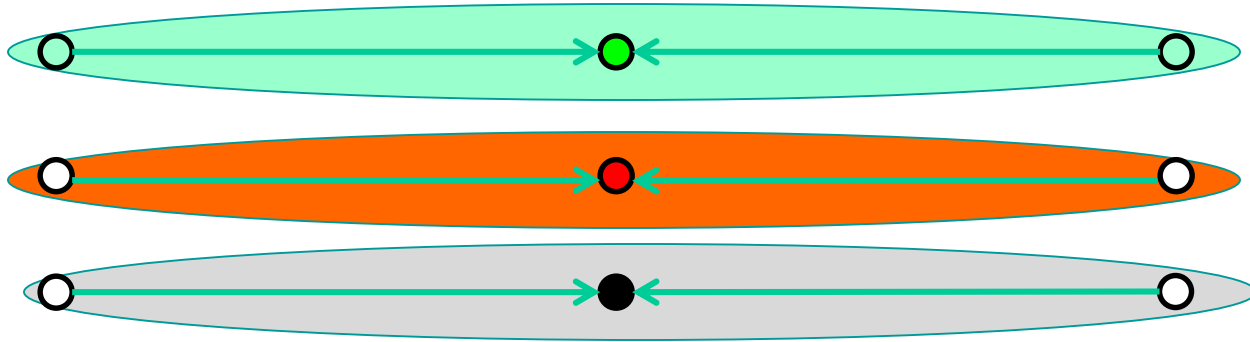
Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering



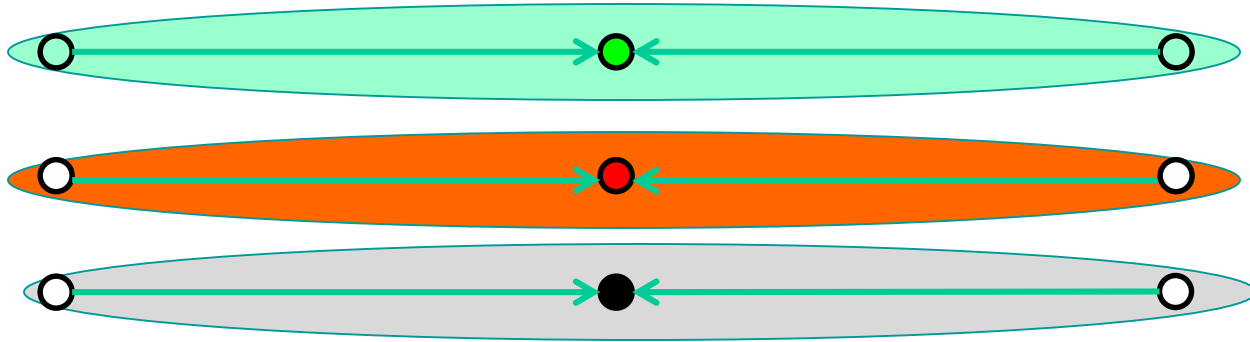
Get a good quality solution in this example.

Lloyd's method: Performance



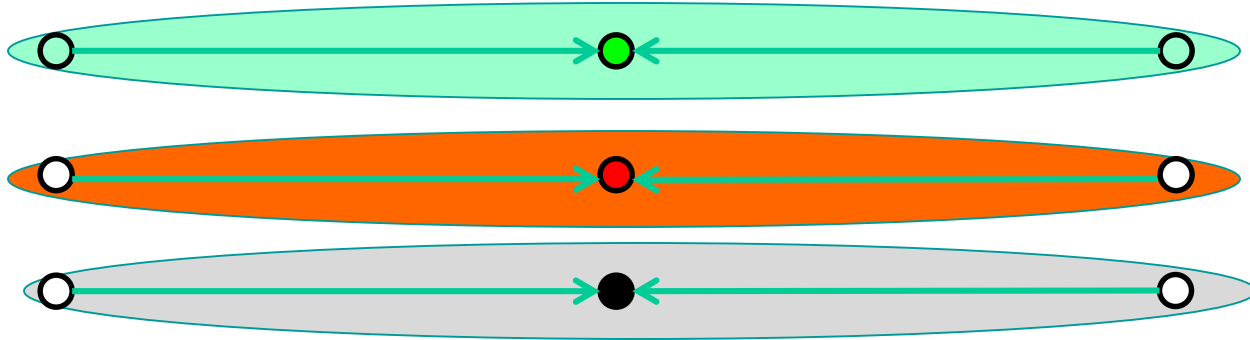
It always converges, but it may converge at a local optimum that is different from the global optimum, and in fact could be arbitrarily worse in terms of its score.

Lloyd's method: Performance

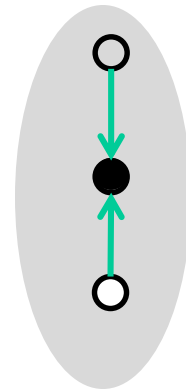
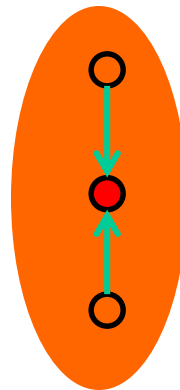
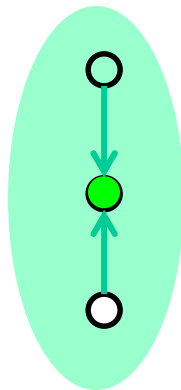


Local optimum: every point is assigned to its nearest center and every center is the mean value of its points.

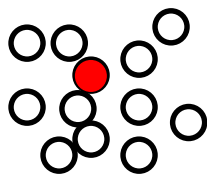
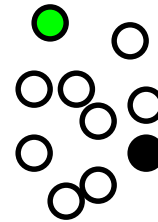
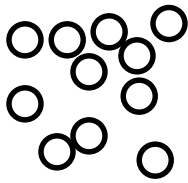
Lloyd's method: Performance



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

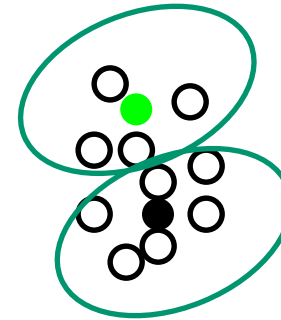
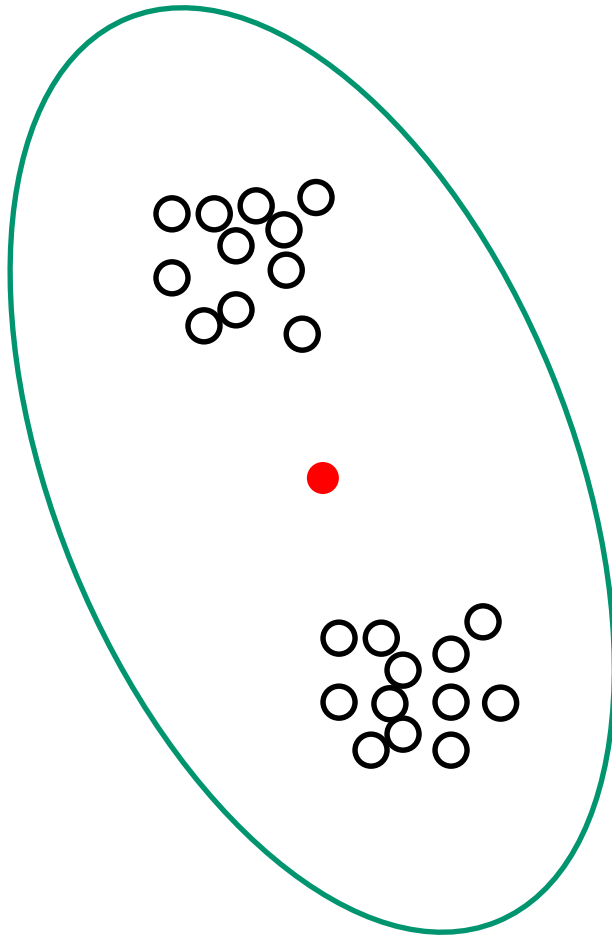


Lloyd's method: Performance



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).
 - For k equal-sized Gaussians, $\Pr[\text{each initial center is in a different Gaussian}] \approx \frac{k!}{k^k} \approx \frac{1}{e^k}$
 - Becomes unlikely as k gets large.

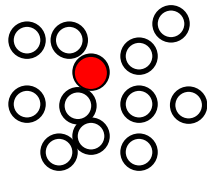
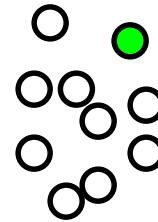
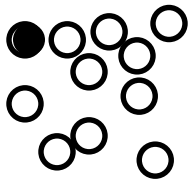
Another Initialization Idea: Furthest Point Heuristic

Choose \mathbf{c}_1 arbitrarily (or at random).

- For $j = 2, \dots, k$
 - Pick \mathbf{c}_j among datapoints $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^d$ that is farthest from previously chosen $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{j-1}$

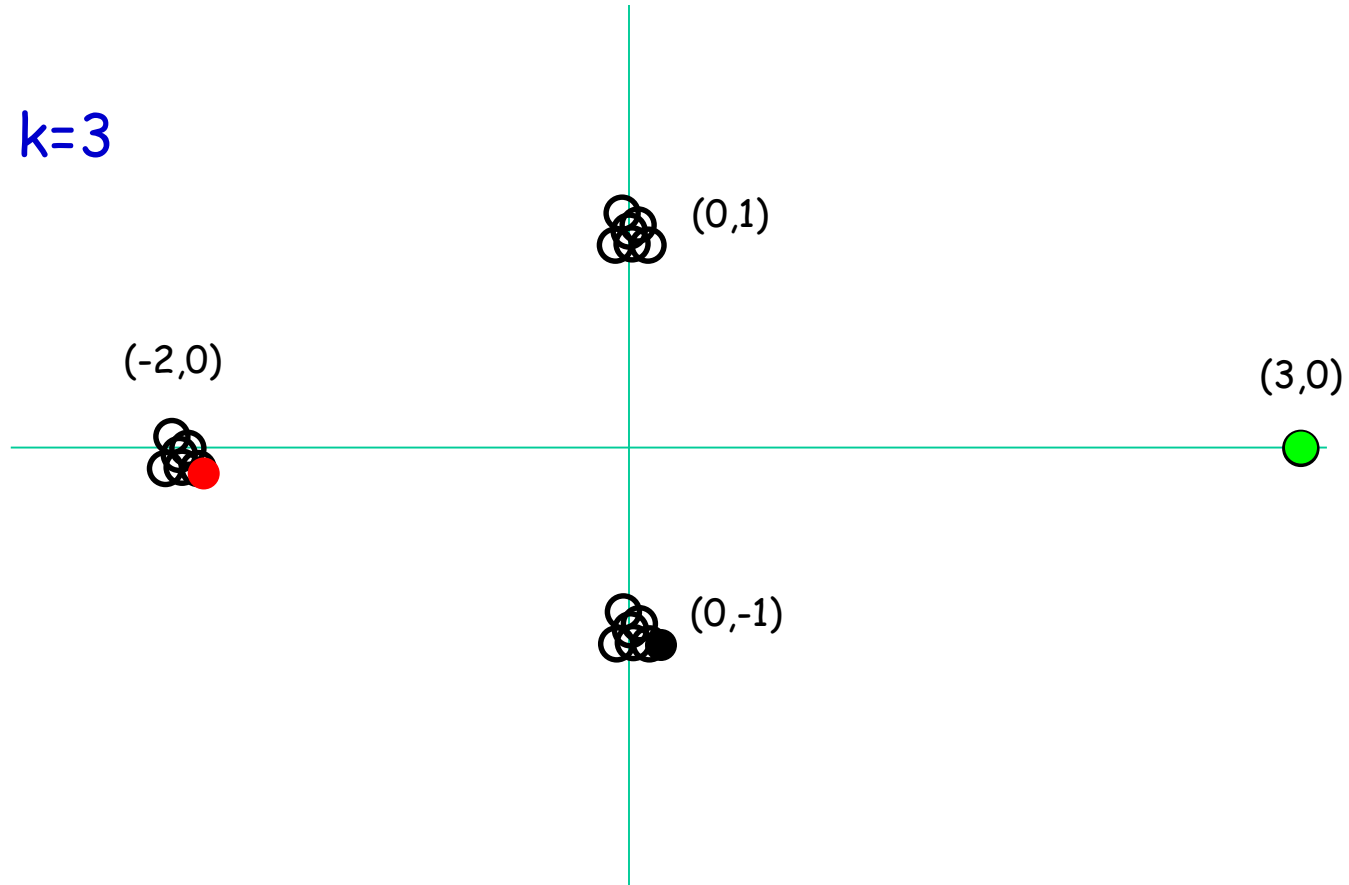
Fixes the Gaussian problem. But it can be thrown off by outliers....

Furthest point heuristic does well on
previous example



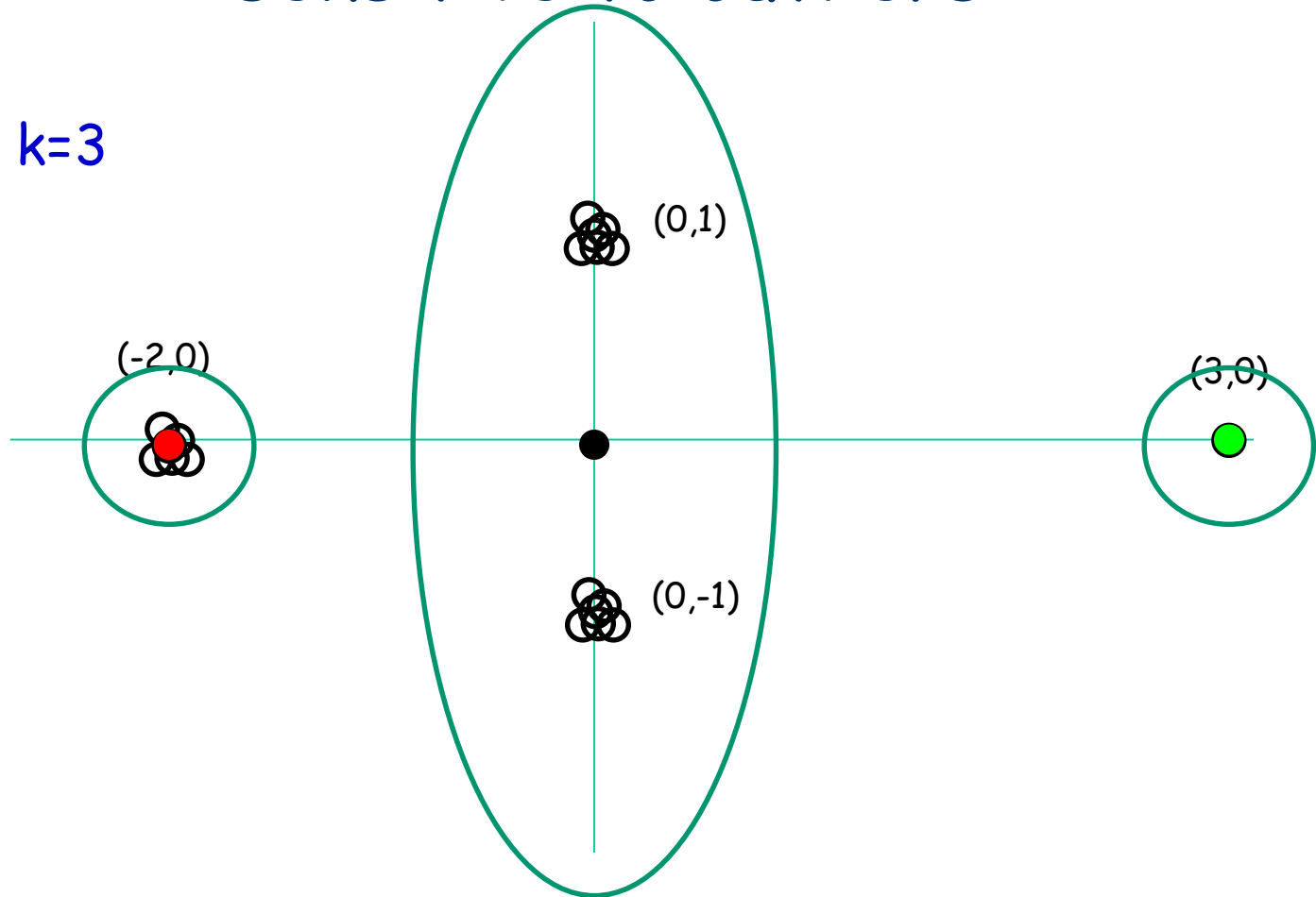
Furthest point initialization heuristic sensitive to outliers

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 $D^2(x)$.

- Choose c_1 at random.
- For $j = 2, \dots, k$
 - Pick c_j among x^1, x^2, \dots, x^d according to the distribution

$$\Pr(c_j = x^i) \propto \min_{j' < j} \|x^i - c_{j'}\|^2 D^2(x^i)$$

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 $D^\alpha(x)$.
- $\alpha = 0$, 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

Repeat until there is no change in the cost.

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

Each round takes time $O(nkd)$.

- Exponential # of rounds in the worst case [AV07].
- Expected polynomial time in the smoothed analysis (non worst-case) model!

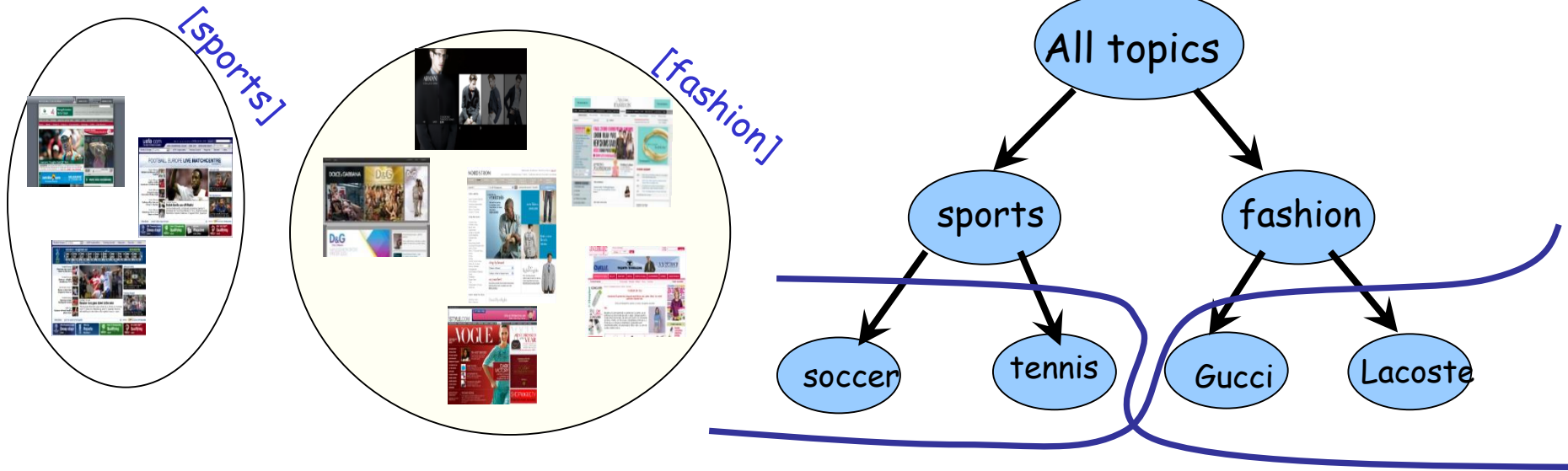
K-means++/ Lloyd's Summary

- 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.

What value of k ???

- Heuristic: Find large gap between $k-1$ -means cost and k -means cost.
- Hold-out validation/cross-validation on auxiliary task (e.g., supervised learning task).
- Try hierarchical clustering.

Hierarchical Clustering

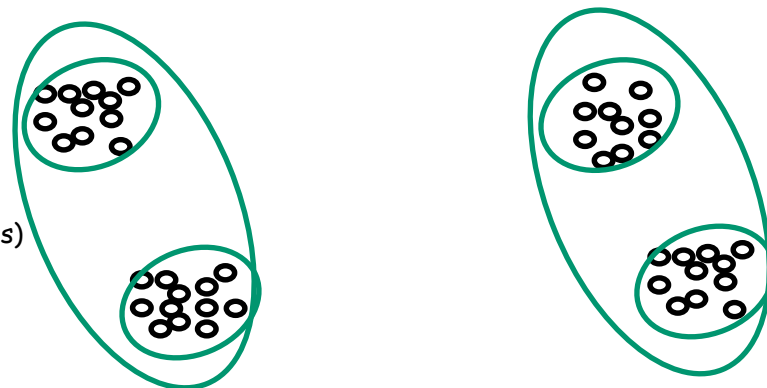


- A hierarchy might be more natural.
- Different users might care about different levels of granularity or even prunings.

Hierarchical Clustering

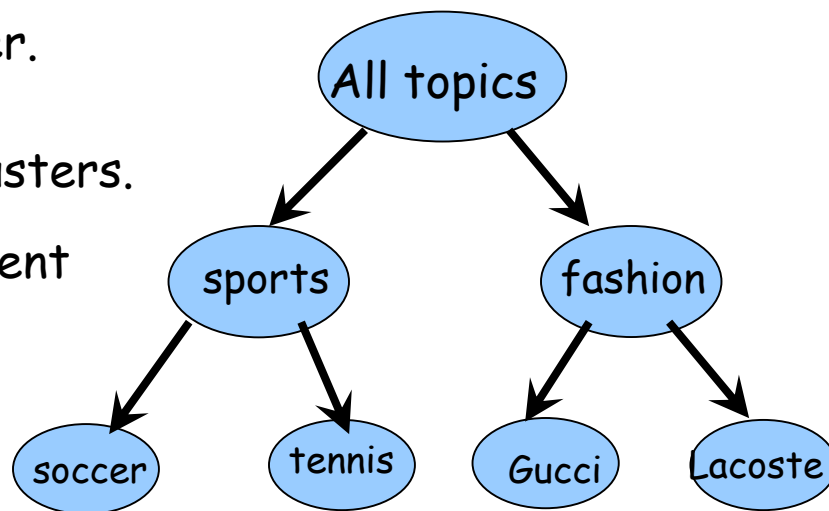
Top-down (divisive)

- 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.

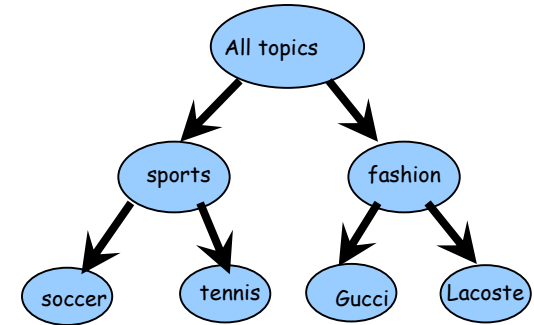


Bottom-Up (agglomerative)

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: $\text{dist}(A, B) = \min_{x \in A, x' \in B'} \text{dist}(x, x')$
- Complete linkage: $\text{dist}(A, B) = \max_{x \in A, x' \in B'} \text{dist}(x, x')$
- Average linkage: $\text{dist}(A, B) = \text{avg}_{x \in A, x' \in B'} \text{dist}(x, x')$

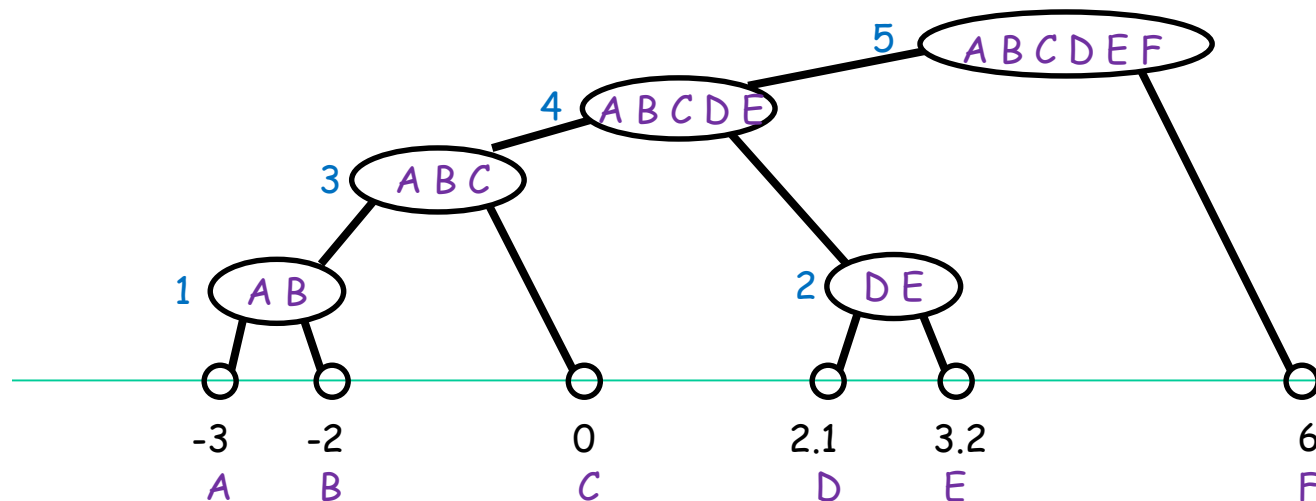
Single Linkage

Bottom-up (agglomerative)

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

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

Dendrogram



Single Linkage

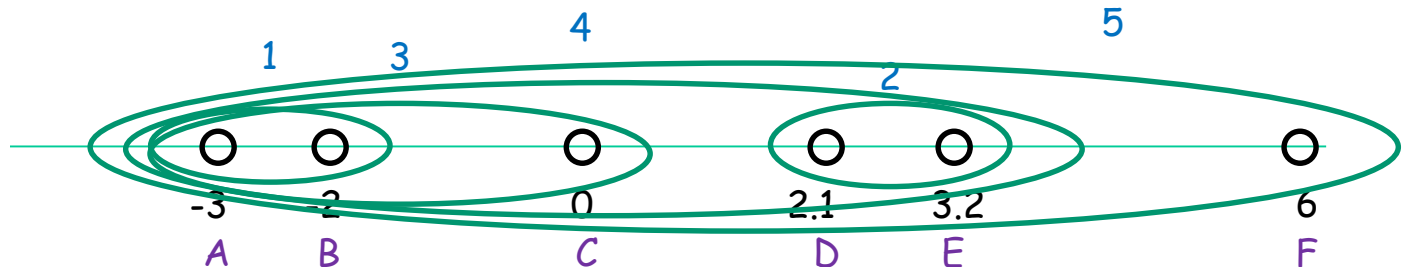
Bottom-up (agglomerative)

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

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

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 merge at value of r corresponding to values of r in different clusters).



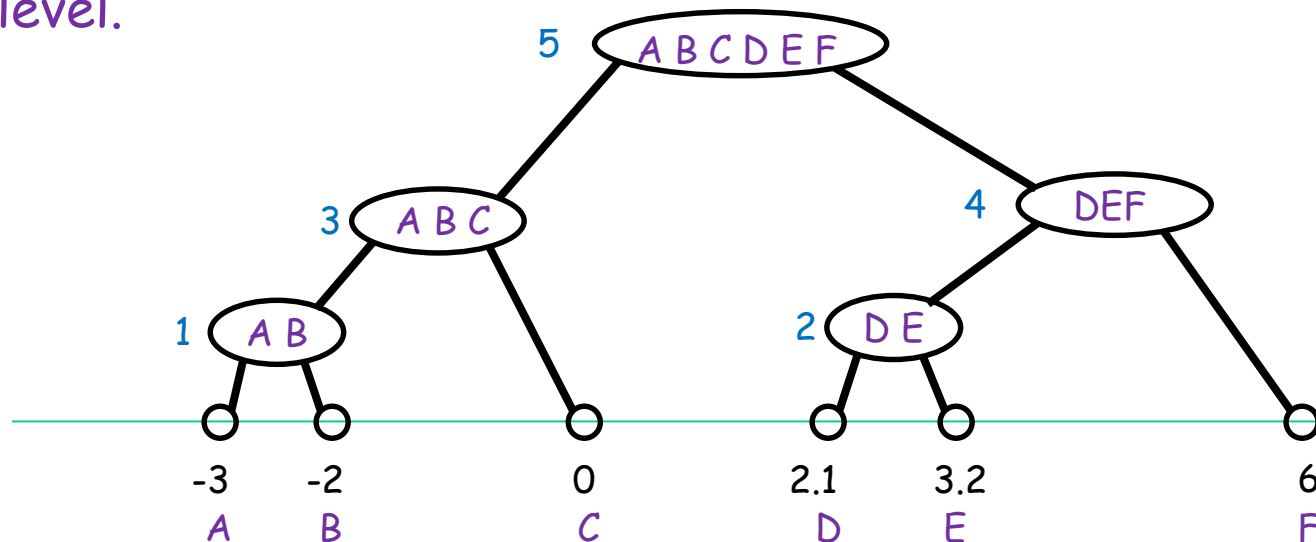
Complete Linkage

Bottom-up (agglomerative)

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

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

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



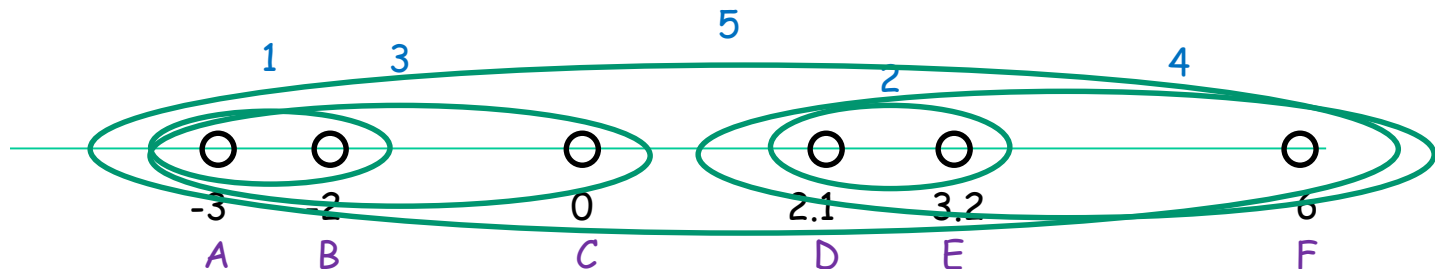
Complete Linkage

Bottom-up (agglomerative)

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

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

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



Running time

- Each algorithm starts with N clusters, and performs $N-1$ merges.
- For each algorithm, computing $\text{dist}(C, C')$ can be done in time $O(|C| \cdot |C'|)$. (e.g., examining $\text{dist}(x, x')$ 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 $O(N^2 \log N)$.

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

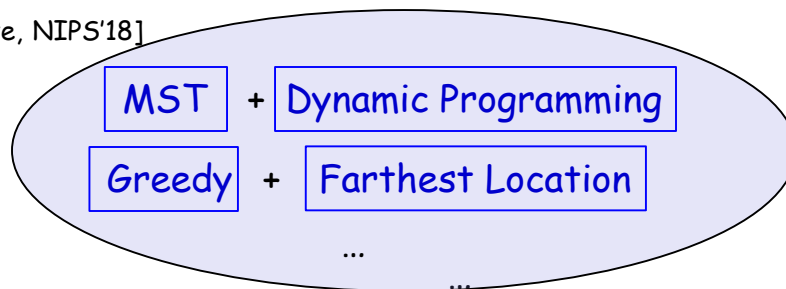
Data Driven Clustering

Goal: given family of algos \mathbf{F} , sample of typical instances from domain (unknown distr. \mathbf{D}), find algo that performs well on new instances from \mathbf{D} .

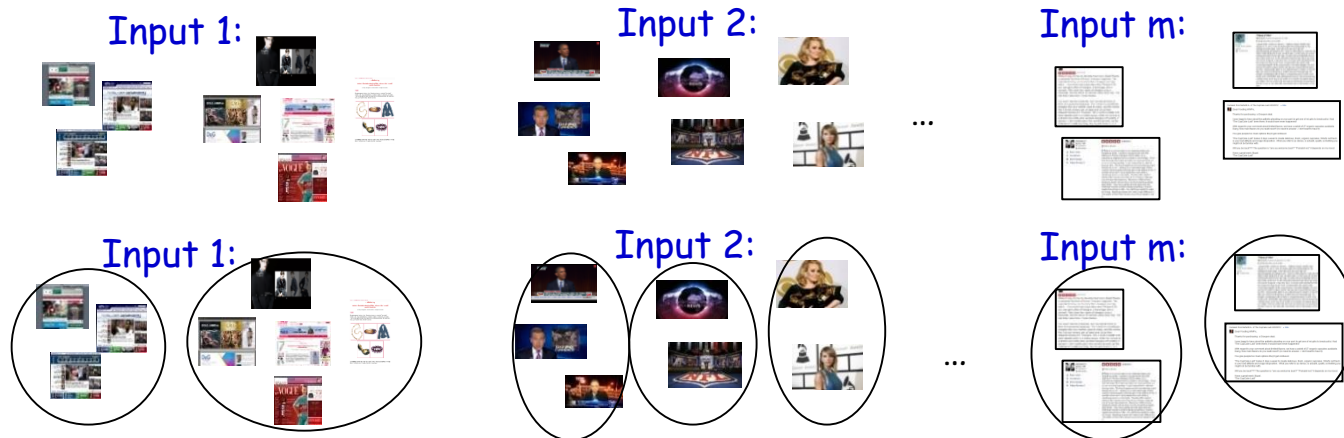
[Balcan-Nagarajan-Vitercik-White, COLT'17] [Balcan-Dick-White, NIPS'18]

Large family \mathbf{F} of algorithms

Sample of typical inputs



Clustering:



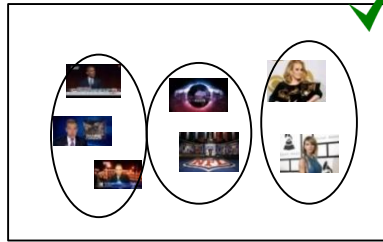
Data Driven Clustering

Goal: given family of algos \mathbf{F} , sample of typical instances from domain (unknown distr. \mathbf{D}), find algo that performs well on new instances from \mathbf{D} .

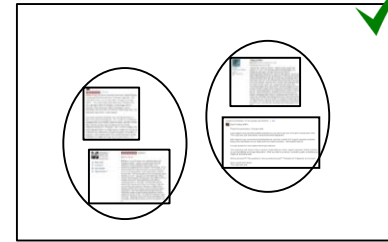
Approach: find $\hat{\mathbf{A}}$ near optimal algorithm over the set of samples.

Key Question: Will $\hat{\mathbf{A}}$ do well on future instances?

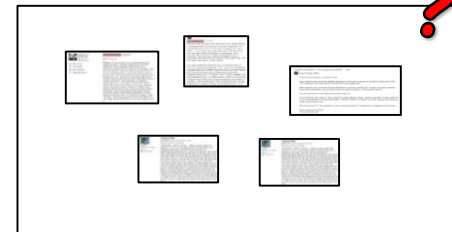
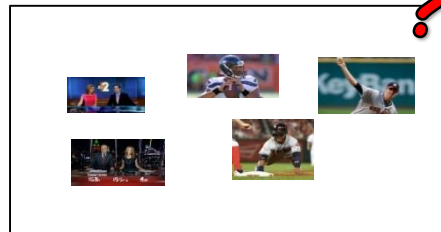
Seen:



...



New:



Sample Complexity: How large should our sample of typical instances be in order to guarantee good performance on new instances?

Data Driven Clustering

Goal: given family of algos \mathbf{F} , sample of typical instances from domain (unknown distr. \mathbf{D}), find algo that performs well on new instances from \mathbf{D} .

Approach: find $\hat{\mathbf{A}}$ near optimal algorithm over the set of samples.

Key Question: Will $\hat{\mathbf{A}}$ do well on future instances?

Key tool: uniform convergence, for any algo in \mathbf{F} , average performance over samples "close" to its expected performance.

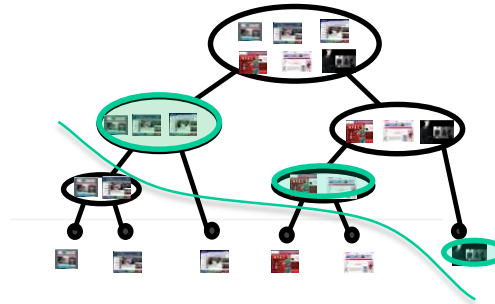
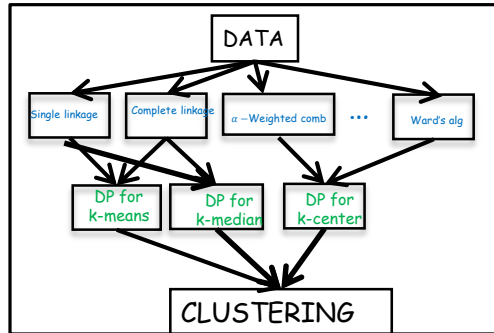
- Imply that $\hat{\mathbf{A}}$ has high expected performance.

Learning theory: $N = O(\dim(\mathbf{F}) / \epsilon^2)$ instances suffice for ϵ -close.

Data Driven Clustering

- Clustering: Linkage + Dynamic Programming

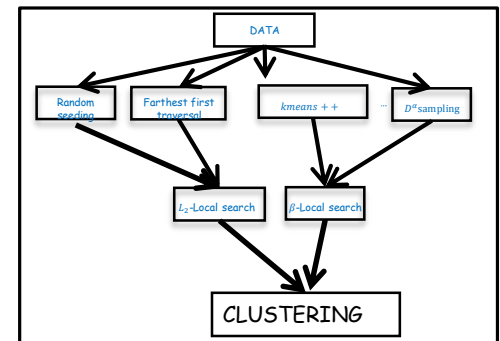
[Balcan-Nagarajan-Vitercik-White, COLT'17]



- Clustering: Greedy Seeding + Local Search

[Balcan-Dick-White, NIPS'18]

Parametrized Lloyd's methods



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 linkage