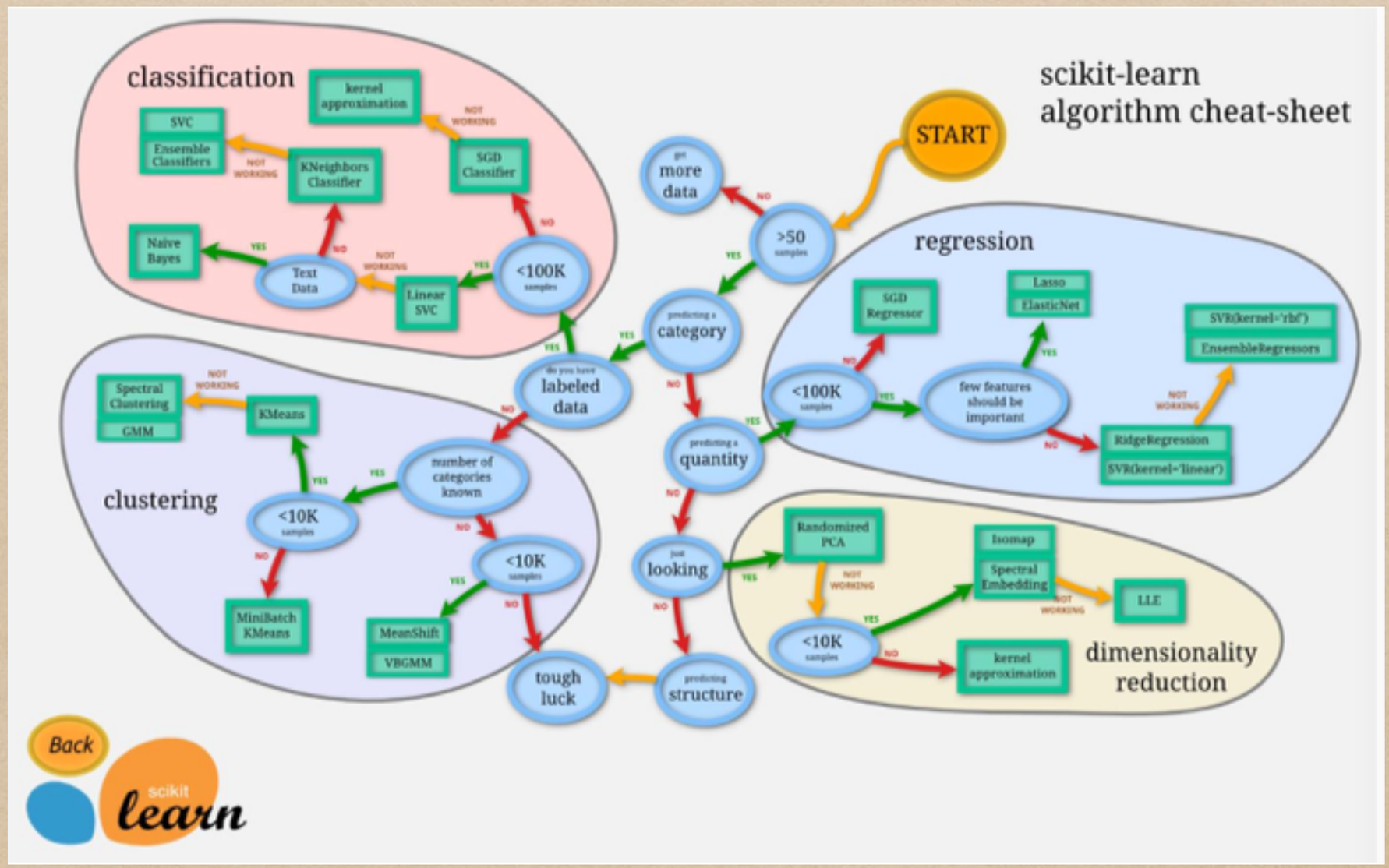
Clustering

Paradigms in ML

- **Supervised Learning**: classification, regression, etc.
- **Unsupervised Learning**: clustering, dimensionality reduction, etc.
- **Reinforcement Learning**

Paradigms in ML



Clustering, Informal Goals

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

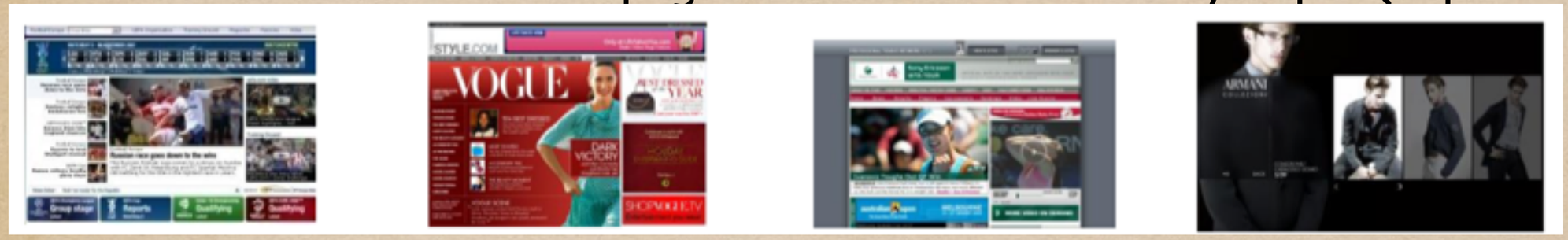
Question: When and why would we want to do this? **Useful for:**

- Automatically organizing data.
- Understanding hidden structure in data.
- Preprocessing for further analysis.
- Representing high-dimensional data in a low-dimensional space

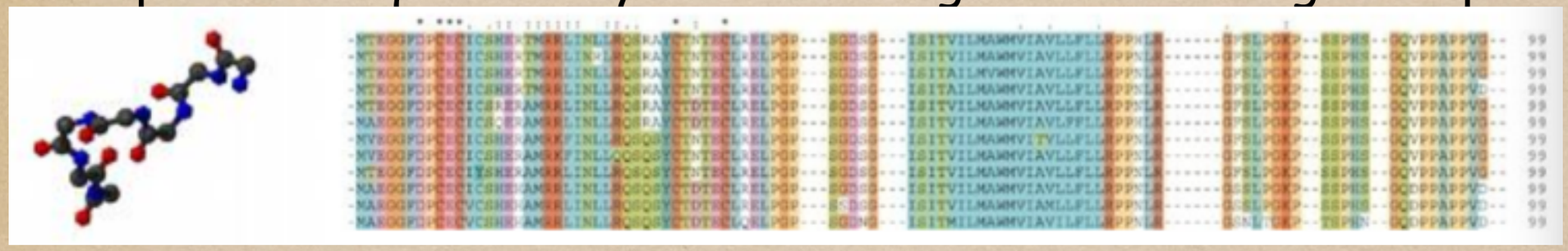
Applications

(Clustering comes up everywhere…)

Cluster news articles or web pages or search results by topic (topic modeling).



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



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

Applications

(Clustering comes up everywhere...)

Cluster customers according to purchase history(recommendation system).

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

- And many many more applications....

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 $c_1, c_2, \ldots, c_k$ to minimize

$$\sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} d^2(\mathbf{x}^i, \mathbf{c}_j)$$

k-median: find center pts $c_1, c_2, \ldots, c_k$ to minimize

$$\sum_{i=1}^{n} \min_{j\in\{1,\ldots,k\}} d(\mathbf{x}^i, \mathbf{c}_j)$$

K-center: find partition to minimize the maximum radius

Euclidean k-means Clustering

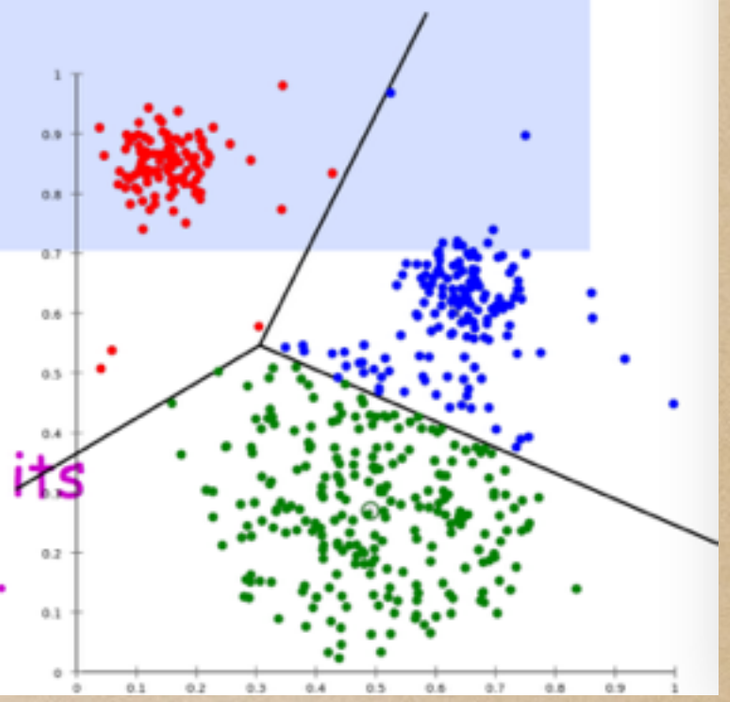
Input: A set of n datapoints $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ in $\mathrm{R^d}$
target #clusters k

Output: k representatives $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R^d}$

Objective: choose $\boldsymbol{c_1}, \mathbf{c_2}, \ldots, \boldsymbol{c_k} \in \mathrm{R^d}$ to minimize

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

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



The Lloyd's method

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

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

Repeat until there is no further change in the cost.

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

Holding $\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_k}$ fixed, pick optimal $\mathrm{C_1}, \mathrm{C_2}, \ldots, \mathrm{C_k}$

Holding $\mathrm{C_1}, \mathrm{C_2}, \ldots, \mathrm{C_k}$ fixed, pick optimal $\boldsymbol{c_1}, \boldsymbol{c_2}, \ldots, \boldsymbol{c_k}$

Initialization for the Lloyd's method

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

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

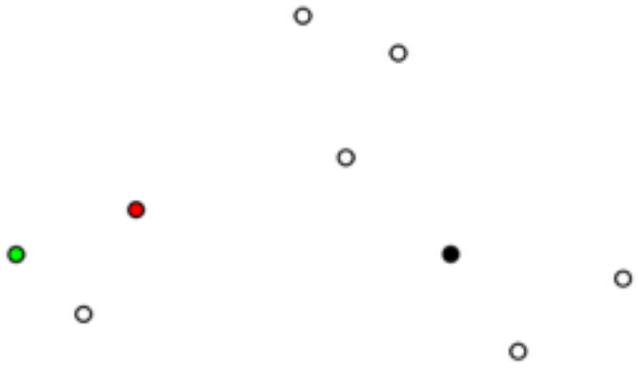
Repeat until there is no further change in the cost.

- 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

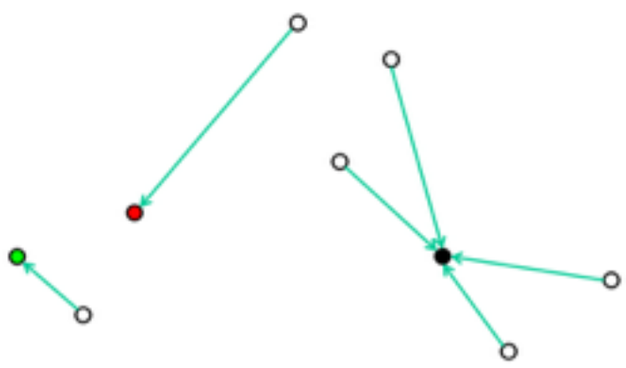
- Initialization is crucial (how fast it converges, quality of solution output)
- Discuss techniques commonly used in practice
  - Random centers from the datapoints (repeat a few times)
  - Furthest traversal
  - K-means ++ (works well and has provable guarantees)

Lloyd's method: Random Initialization

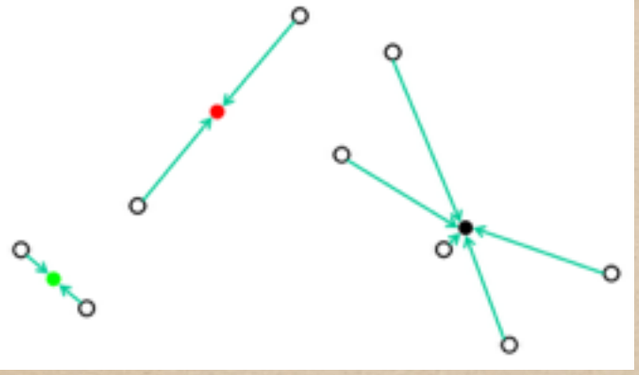
Select initial centers at random



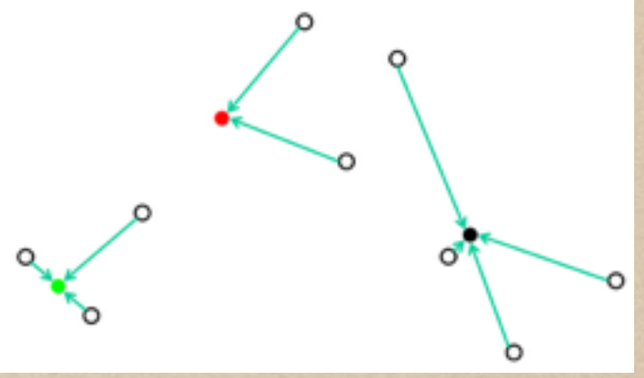
Assign each point to its nearest center



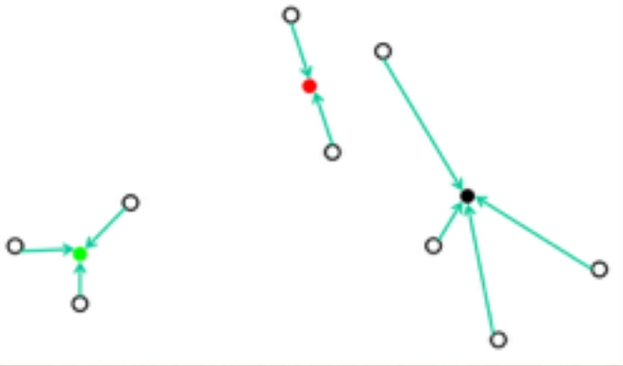
Recompute optimal centers given a fixed clustering



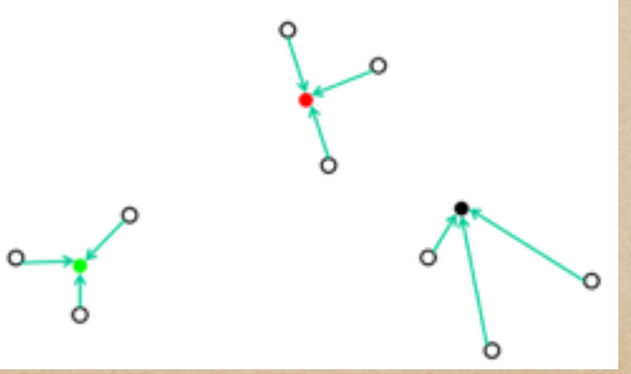
Assign each point to its nearest center



Recompute optimal centers given a fixed clustering

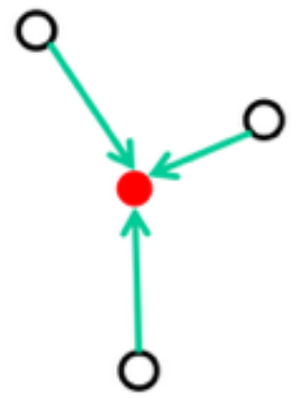


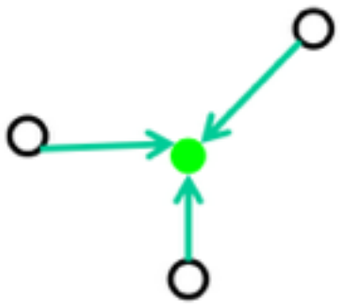
Assign each point to its nearest center

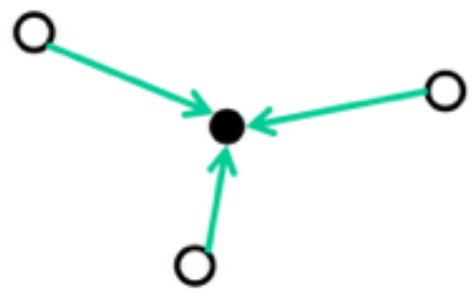


Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering

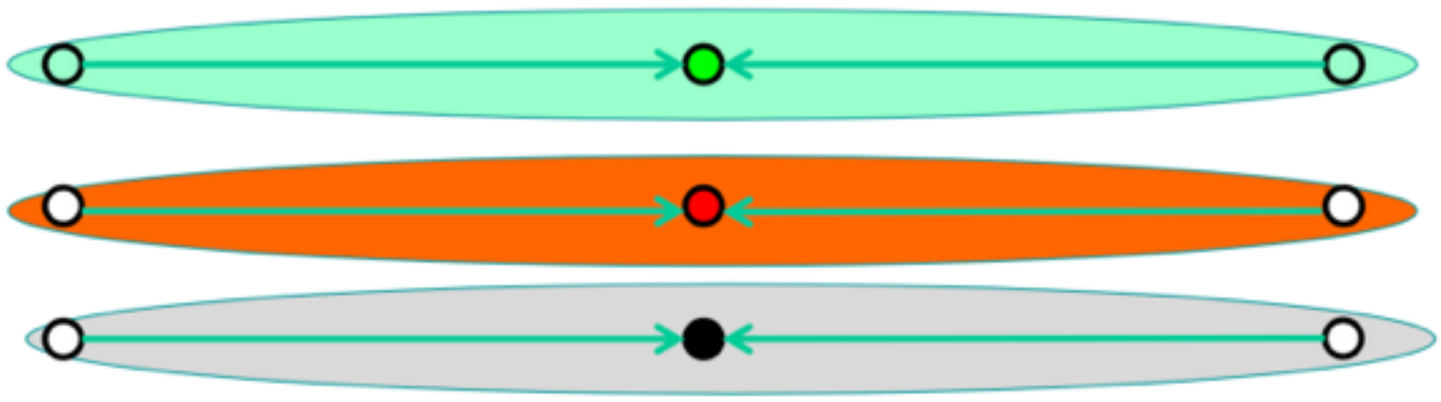




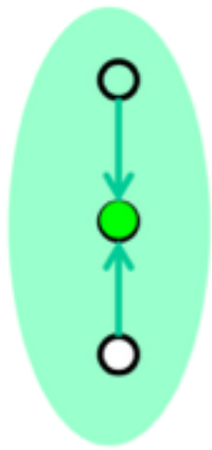


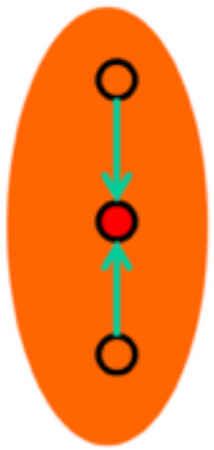
Get a good quality solution in this example.

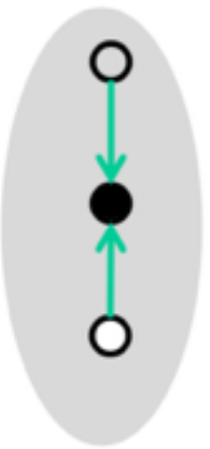
Lloyd's method: Random Initialization



It is arbitrarily worse than optimum solution....





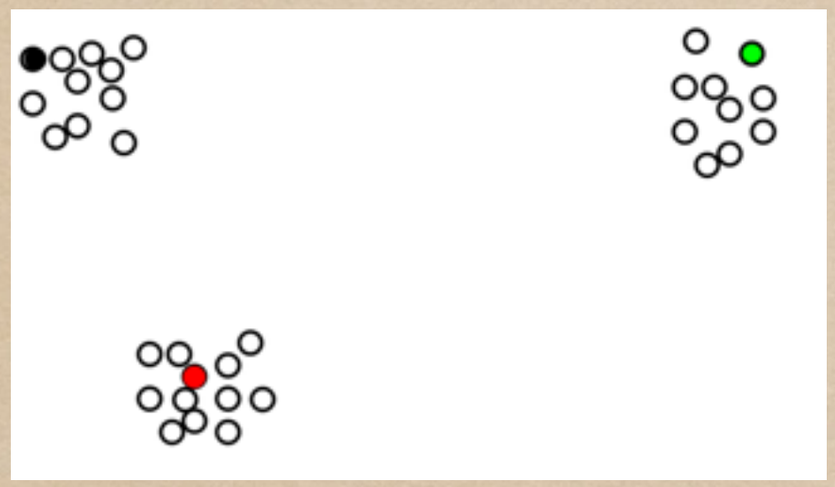


Furthest point heuristic does well on previous example

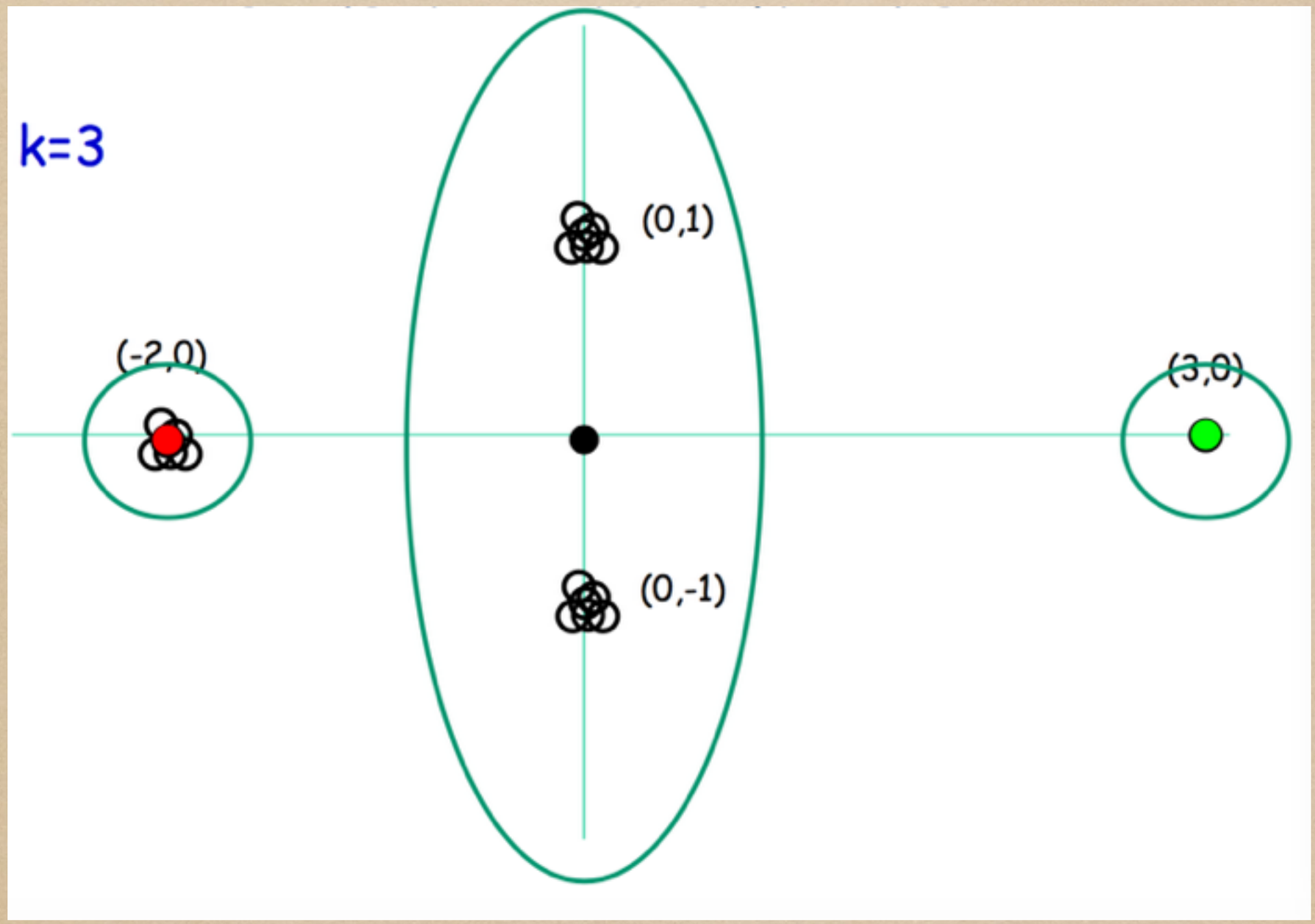
Choose $\mathbf{c_1}$ arbitrarily (or at random).

- For $\mathrm{j} = 2, \ldots, \mathrm{k}$
  - Pick $\mathbf{c_j}$ among datapoints $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ that is farthest from previously chosen $\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_{j-1}}$

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



Furthest point initialization heuristic sensitive to outliers



K-means++ Initialization: D2 sampling [AV07]

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

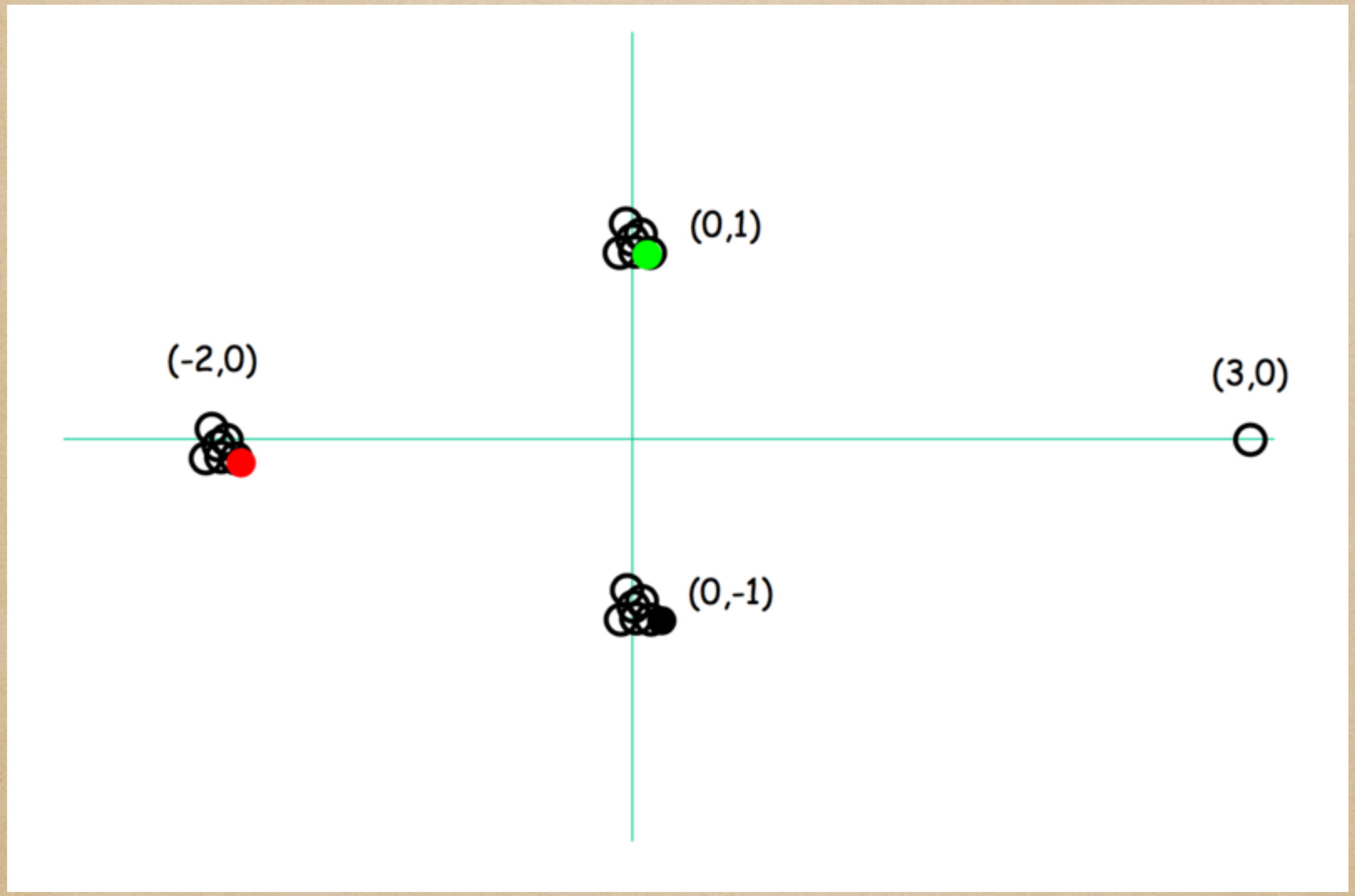
  - Choose $\mathbf{c_1}$ at random.
  - For $j = 2, \ldots, k$
    - Pick $\mathbf{c_j}$ among $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^n}$ according to the distribution

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

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

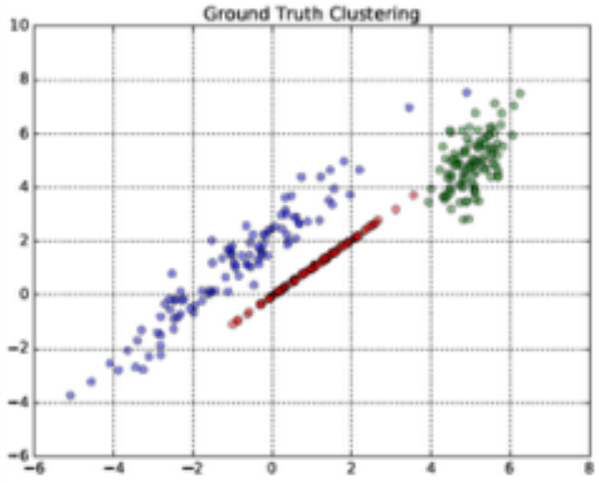
Each round takes time $O(nkd)$.

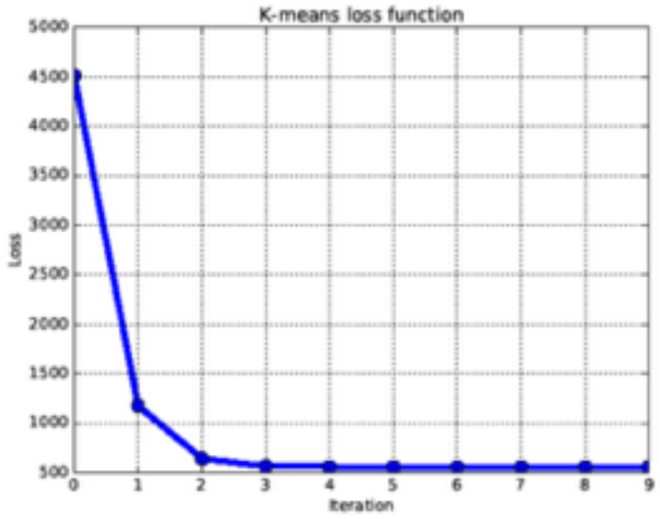
- Exponential # of rounds in the worst case [AV07].
- Expected polynomial time in the smoothed analysis (non worst-case) model!

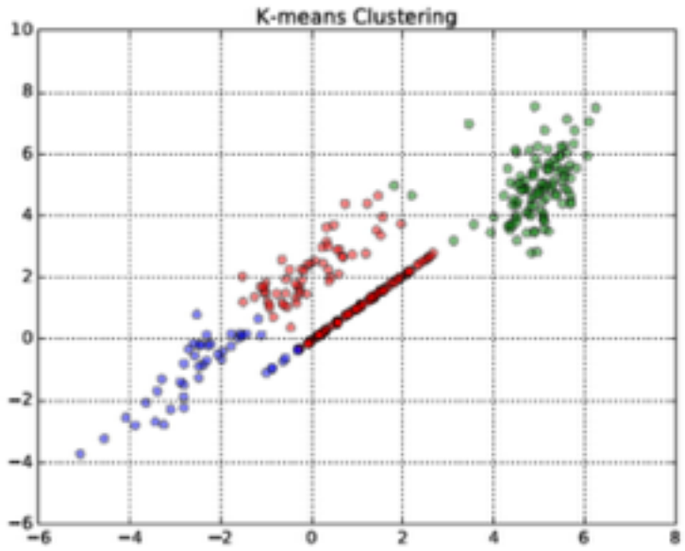
What value of k???

- Heuristic (Elbow's method): 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.

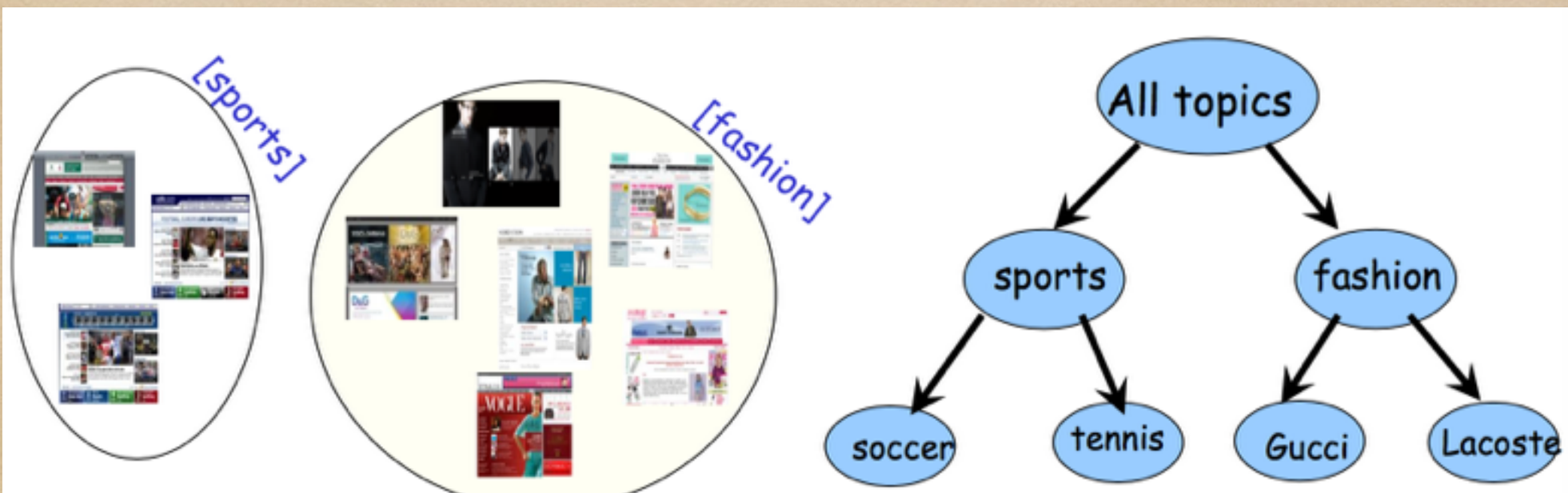
Kmeans Examples







Hierarchical Clustering



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

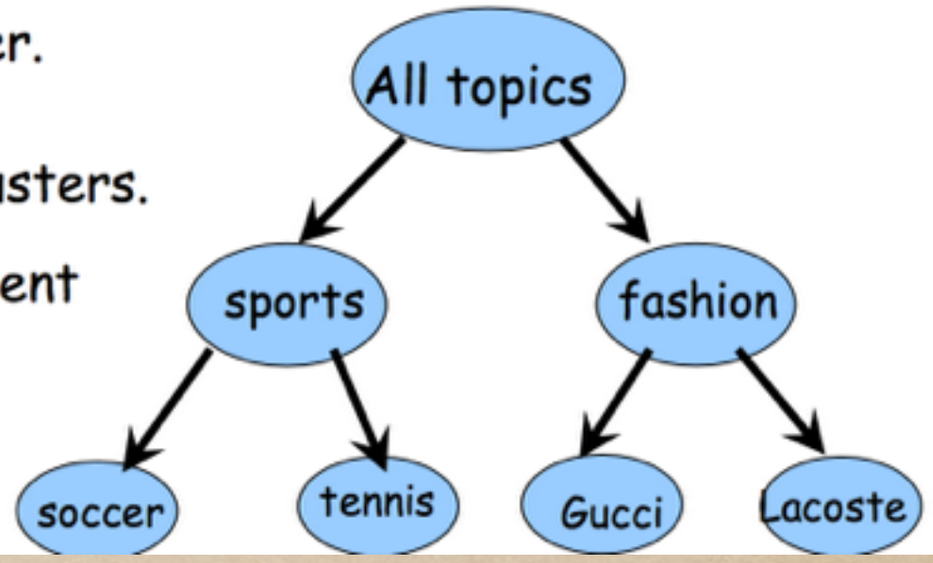
Hierarchical Clustering

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

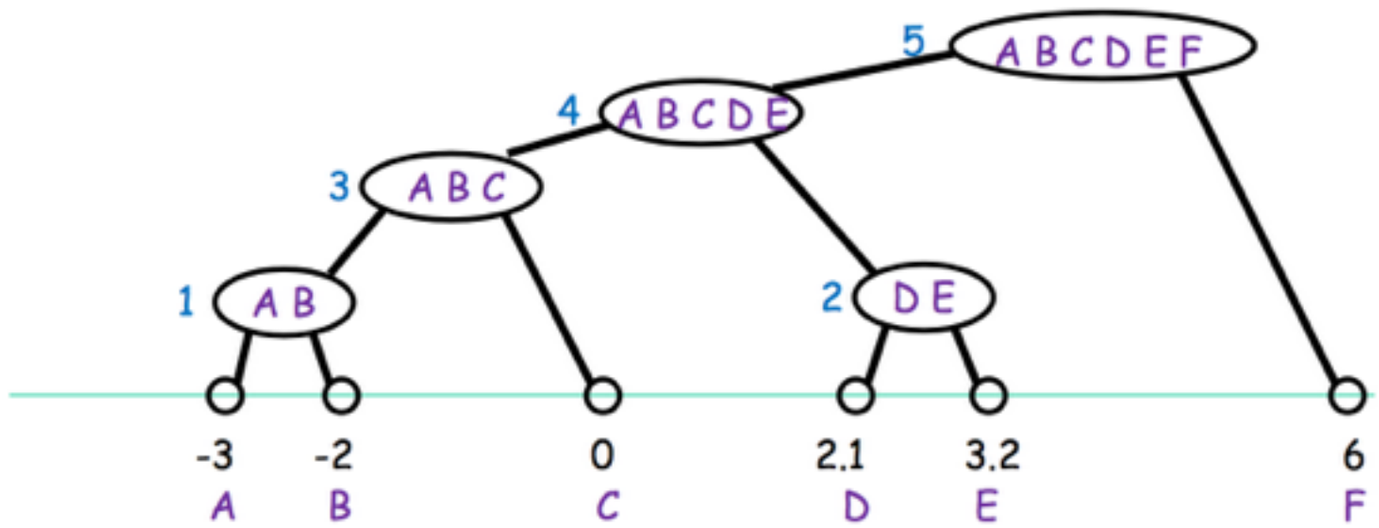
Single Linkage

Bottom-up (agglomerative)

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

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

Dendogram



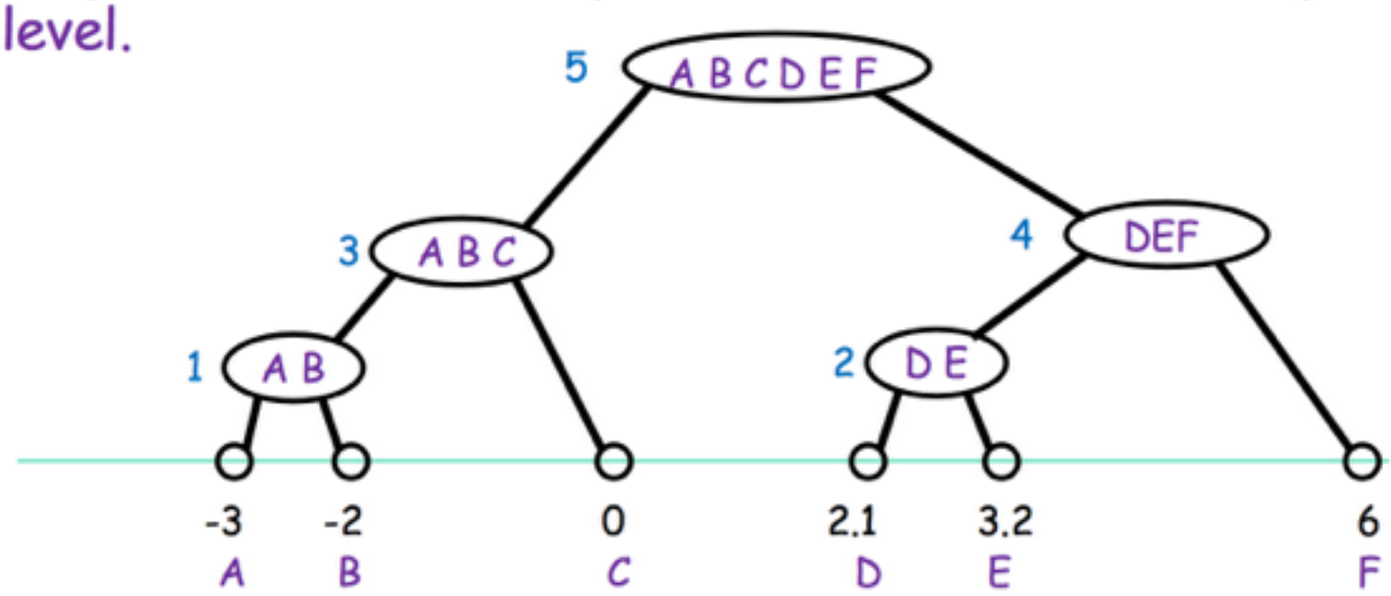
Complete Linkage

Bottom-up (agglomerative)

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

Complete linkage: $\text{dist}(S, T) = \max_{x \in S, x' \in T} \text{dist}(x, x')$

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



Running time for Single and Complete Linkage

- 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 $O(N^2 \log N)$.

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/

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