Clustering.
Unsupervised Learning

Maria-Florina Balcan

04/08/2019

Clustering, Informal Goals

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

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

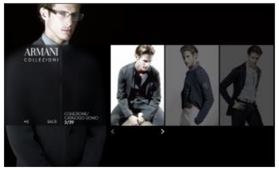
Useful for:

- 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…)

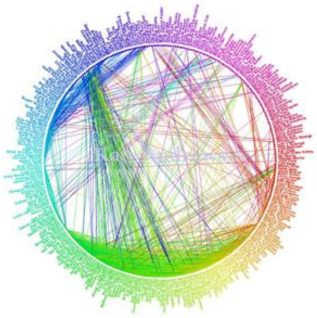
- Cluster news articles or web pages or search results by topic.



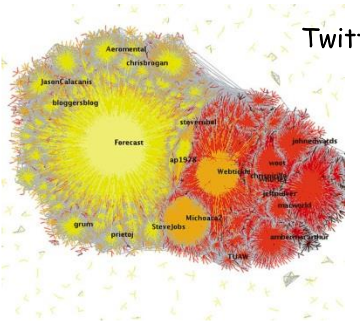



- Cluster protein sequences by function or genes according to expression profile.


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



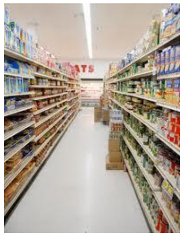
Facebook network



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:

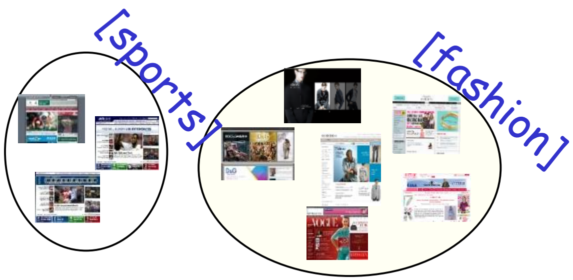
- Objective based clustering
  - K-means clustering
- Hierarchical clustering

Objective Based Clustering

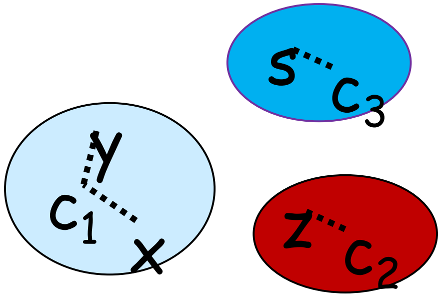
Input: A set S of n points, also a distance/dissimilarity measure specifying the distance d(x,y) between pairs (x,y).

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

Goal: output a partition of the data.



- k-means: find center pts $c_1, c_2, \ldots, c_k$ 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 $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ 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

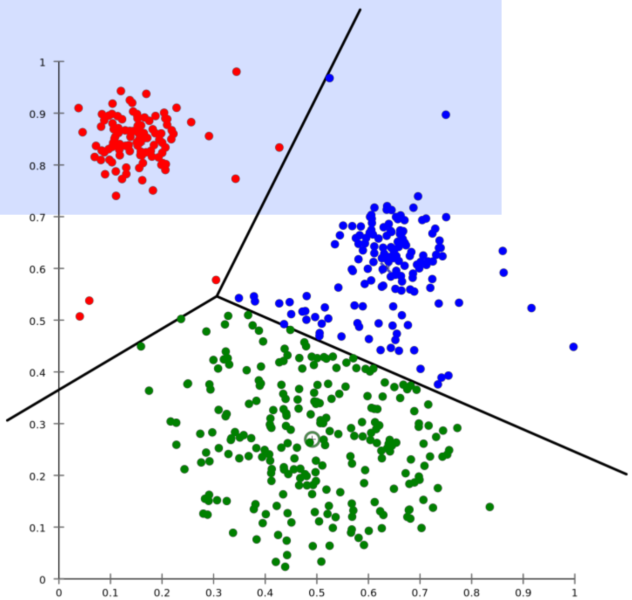
Input: A set of n datapoints $\mathbf{x}^{\mathbf{1}}, \mathbf{x}^{\mathbf{2}}, \dots, \mathbf{x}^{\boldsymbol{n}}$ in R^{d}

target #clusters k

Output: k representatives $\boldsymbol{c}_{\mathbf{1}}, \mathbf{c}_{\mathbf{2}}, \dots, \boldsymbol{c}_{\boldsymbol{k}} \in \mathrm{R}^{\mathrm{d}}$

Objective: choose $\boldsymbol{c}_{\mathbf{1}}, \mathbf{c}_{\mathbf{2}}, \dots, \boldsymbol{c}_{\boldsymbol{k}} \in \mathrm{R}^{\mathrm{d}}$ to minimize

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



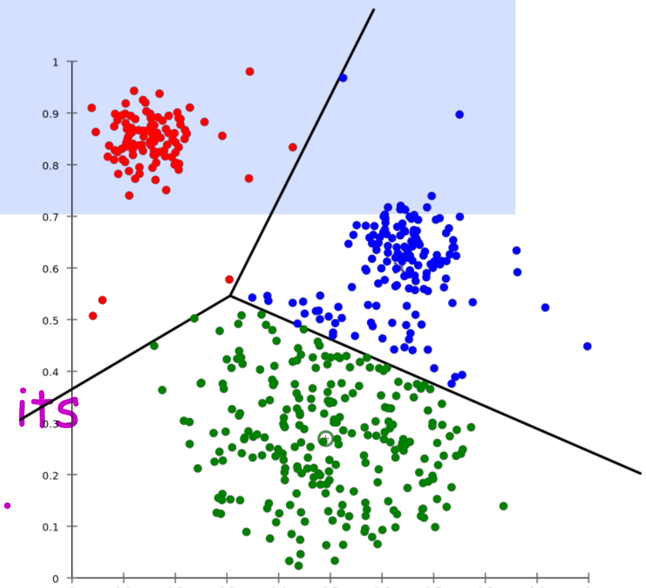
Euclidean k-means Clustering

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

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



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

Euclidean k-means Clustering

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

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

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

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

Idea: bias/variance like decomposition

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

Avg k-means cost wrt c

Avg k-means cost wrt μ

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

Another Easy Case for k-means: d=1

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

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

Extra-credit homework question

Hint: dynamic programming in time $\mathrm{O}(\mathrm{n}^2\mathrm{k})$.

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 $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^\mathbf{n}$ in R^d

Initialize centers $\boldsymbol{c}_1, \mathbf{c}_2, \dots, \boldsymbol{c}_{\boldsymbol{k}} \in \mathrm{R}^\mathrm{d}$ and clusters $\mathrm{C}_1, \mathrm{C}_2, \dots, \mathrm{C}_\mathrm{k}$ in any way.

Repeat until there is no further change in the cost.

- For each j: $\mathrm{C}_\mathrm{j} \leftarrow \{x \in S$ whose closest center is $\mathbf{c}_\mathbf{j}\}$
- For each j: $\mathbf{c}_\mathbf{j} \leftarrow$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 $\mathbf{x}^{\mathbf{1}}, \mathbf{x}^{\mathbf{2}}, \ldots, \mathbf{x}^{\mathbf{n}}$ in R^{d}

Initialize centers $\boldsymbol{c}_{\mathbf{1}}, \mathbf{c}_{\mathbf{2}}, \ldots, \boldsymbol{c}_{\boldsymbol{k}} \in \mathrm{R}^{\mathrm{d}}$ and clusters $\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_{\mathrm{k}}$ in any way.

Repeat until there is no further change in the cost.

- For each j: $\mathrm{C}_{\mathrm{j}} \leftarrow \{x \in S$ whose closest center is $\mathbf{c}_{\mathbf{j}}\}$
- For each j: $\mathbf{c}_{\mathbf{j}} \leftarrow$ mean of C_{j}

Holding $\boldsymbol{c}_{\mathbf{1}}, \mathbf{c}_{\mathbf{2}}, \ldots, \boldsymbol{c}_{\boldsymbol{k}}$ fixed, pick optimal $\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_{\mathrm{k}}$

Holding $\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_{\mathrm{k}}$ fixed, pick optimal $\boldsymbol{c}_{\mathbf{1}}, \mathbf{c}_{\mathbf{2}}, \ldots, \boldsymbol{c}_{\boldsymbol{k}}$

Common Heuristic: 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 $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k \in \mathrm{R}^d$ 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

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 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.

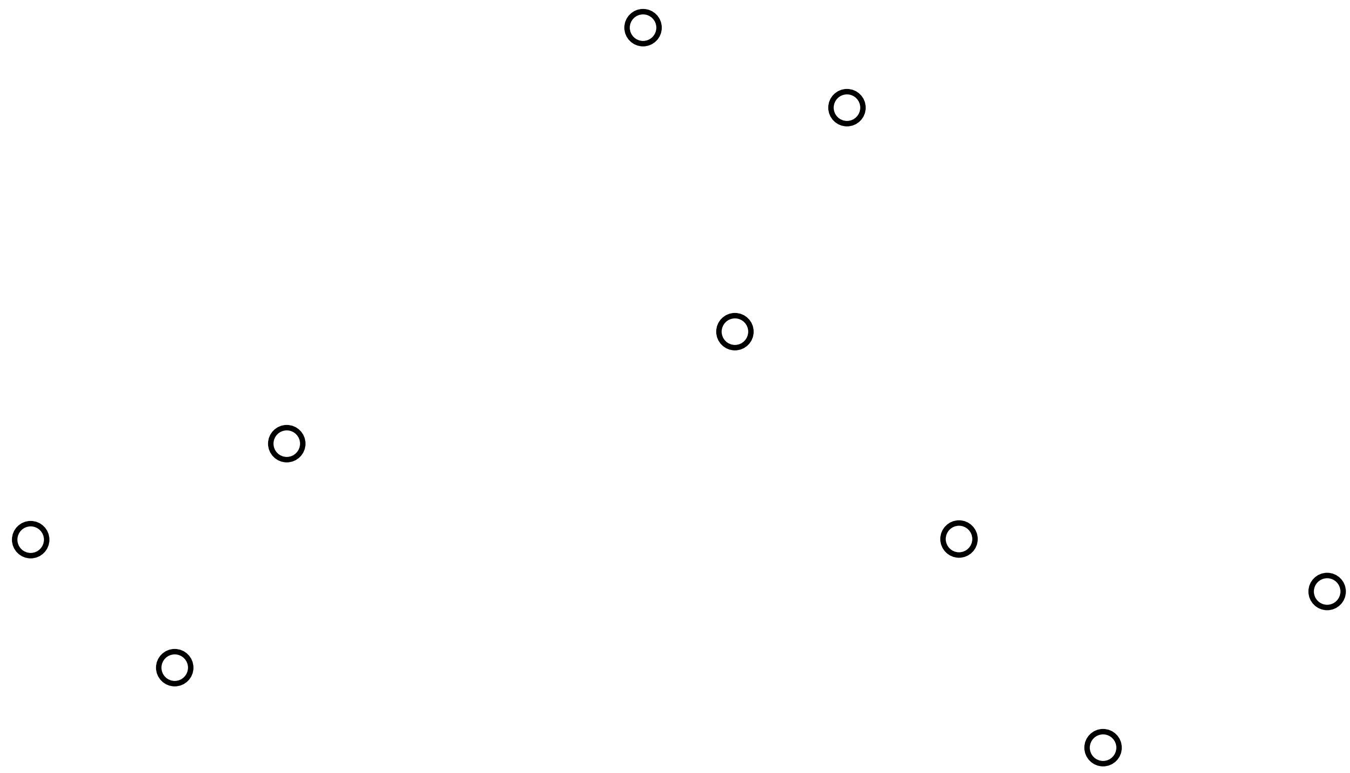
- For each j: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c}_j\}$
- For each j: $\mathbf{c}_j \leftarrow$mean of C_j

- 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

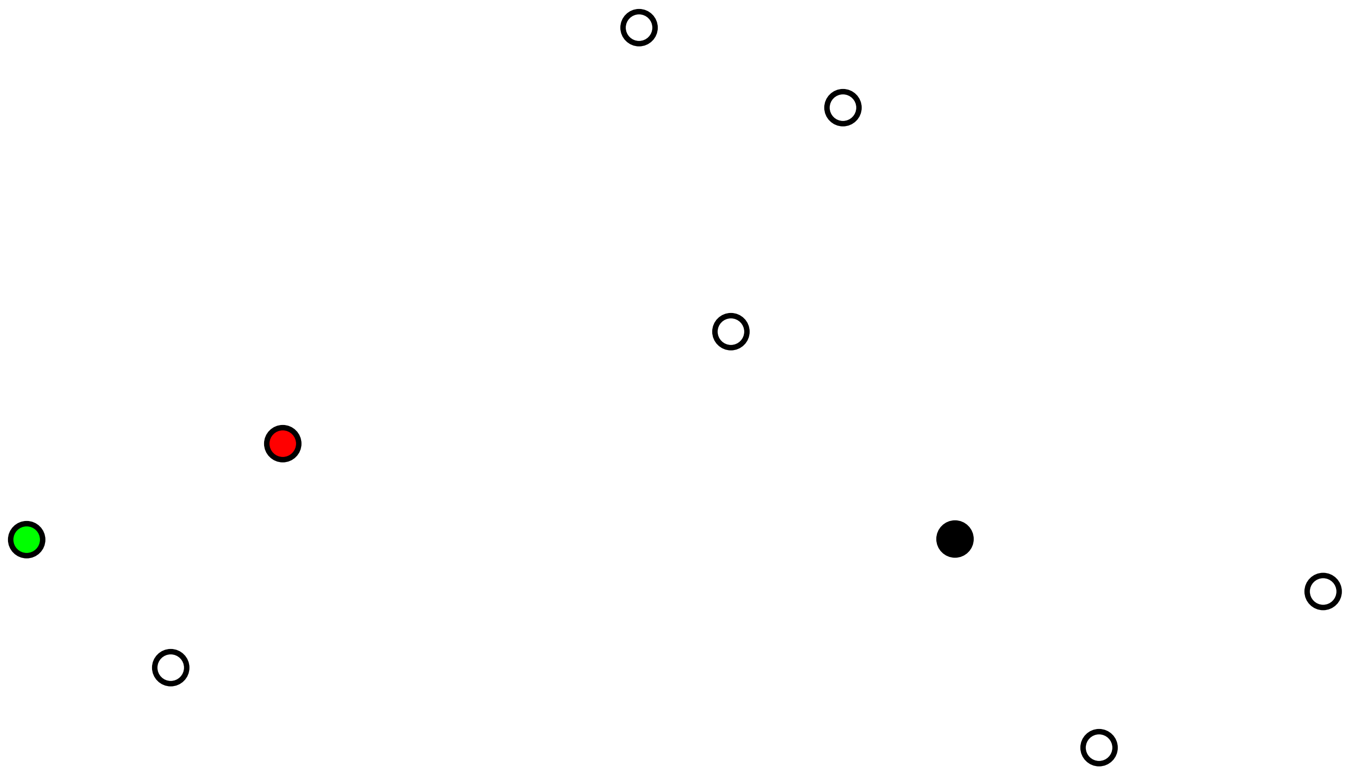
Lloyd's method: Random Initialization

Example: Given a set of datapoints



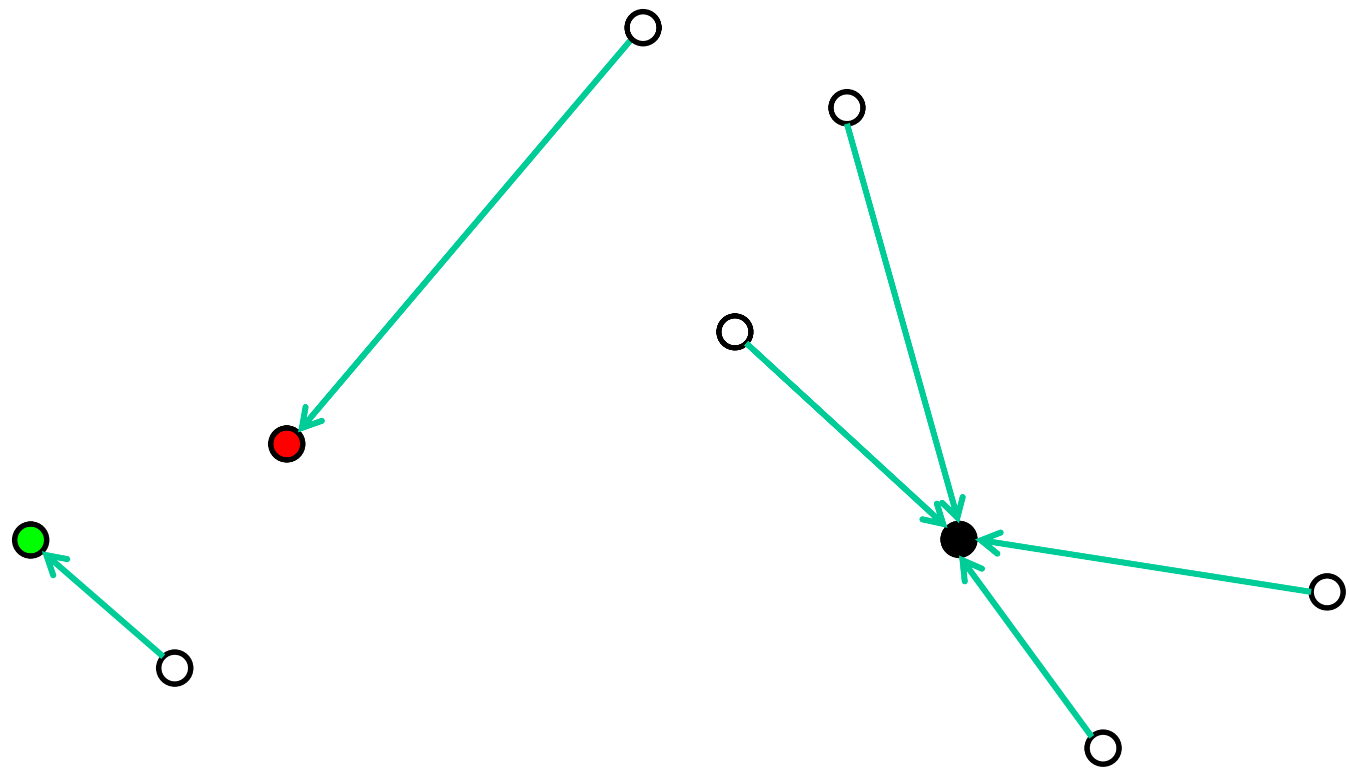
Lloyd's method: Random Initialization

Select initial centers at random



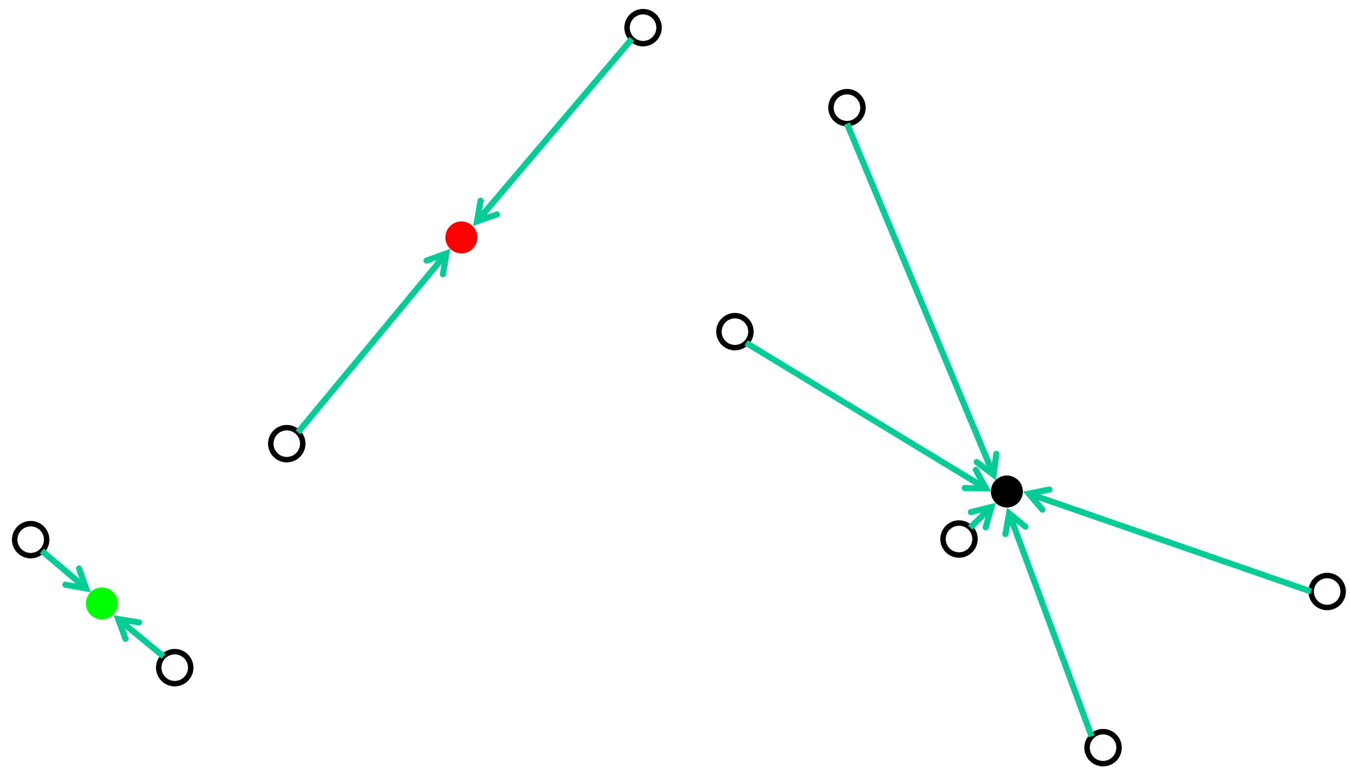
Lloyd's method: Random Initialization

Assign each point to its nearest center



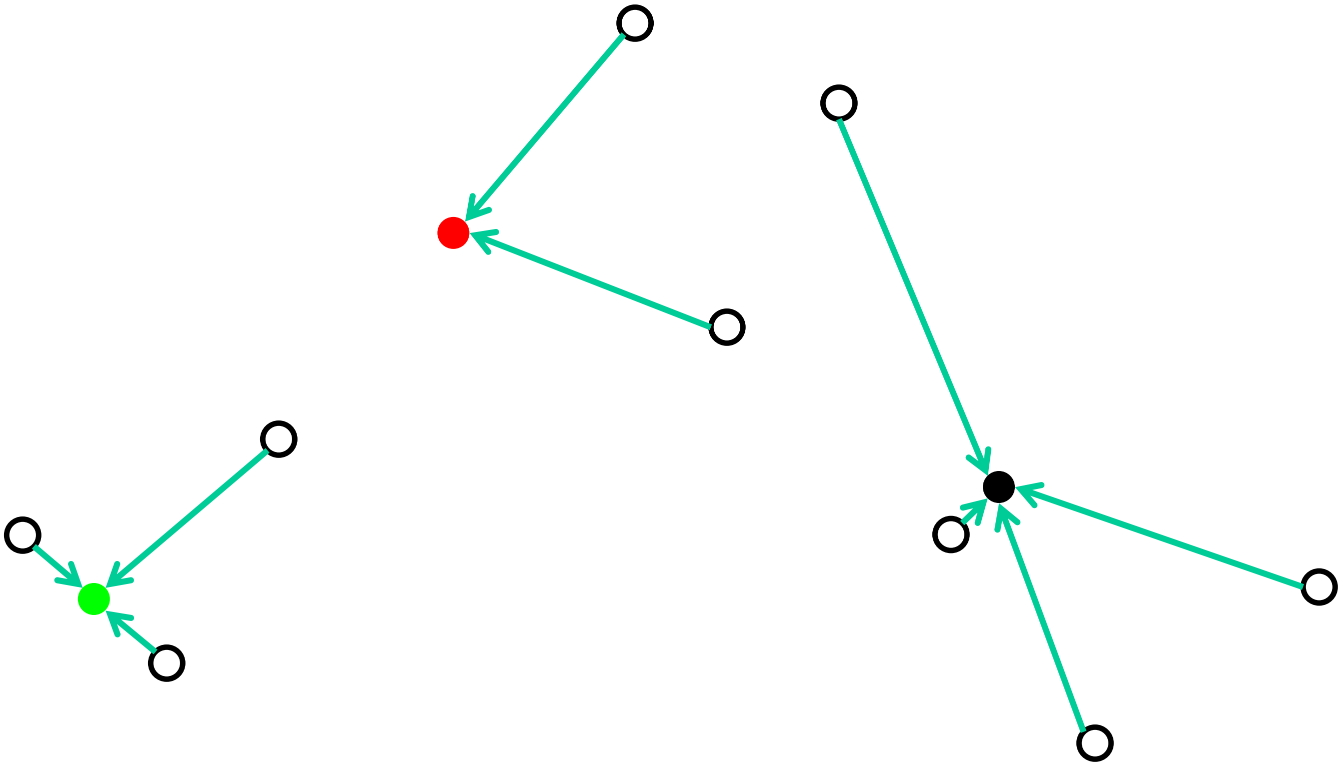
Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering



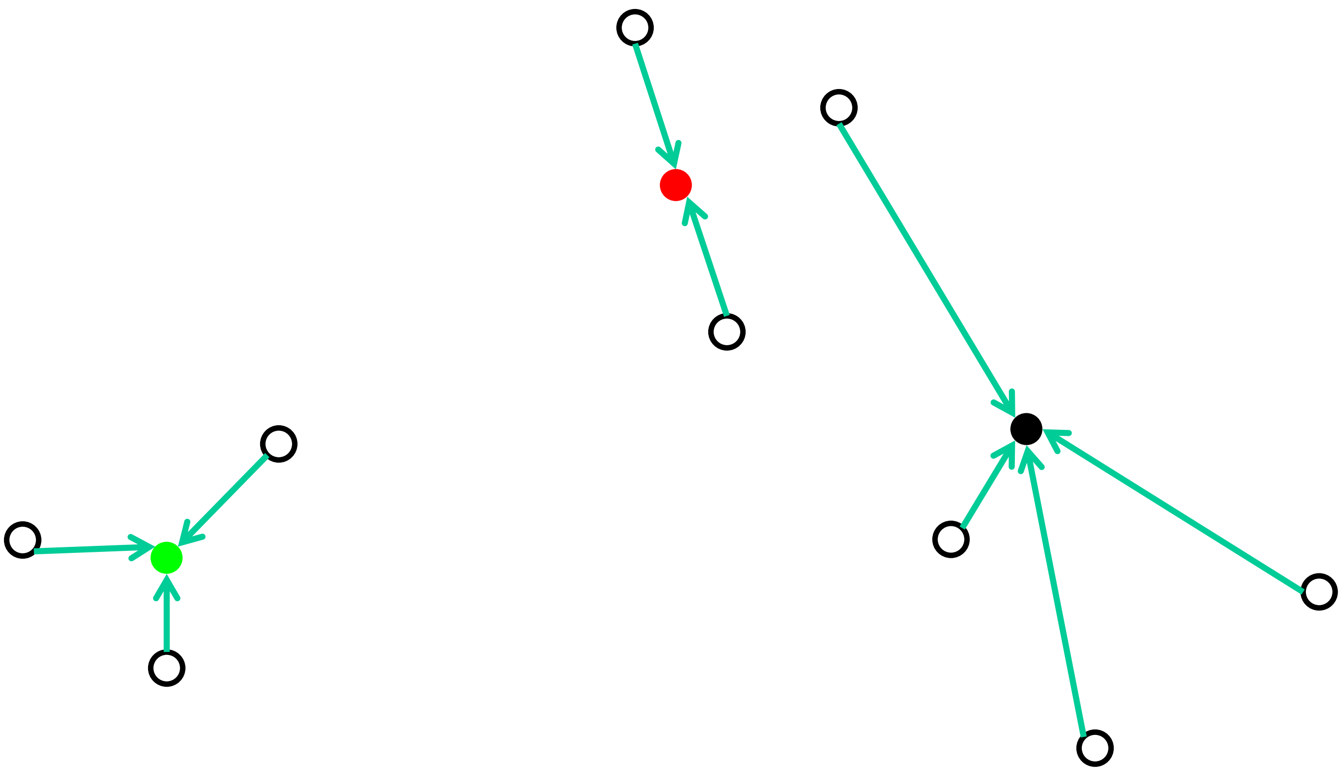
Lloyd's method: Random Initialization

Assign each point to its nearest center



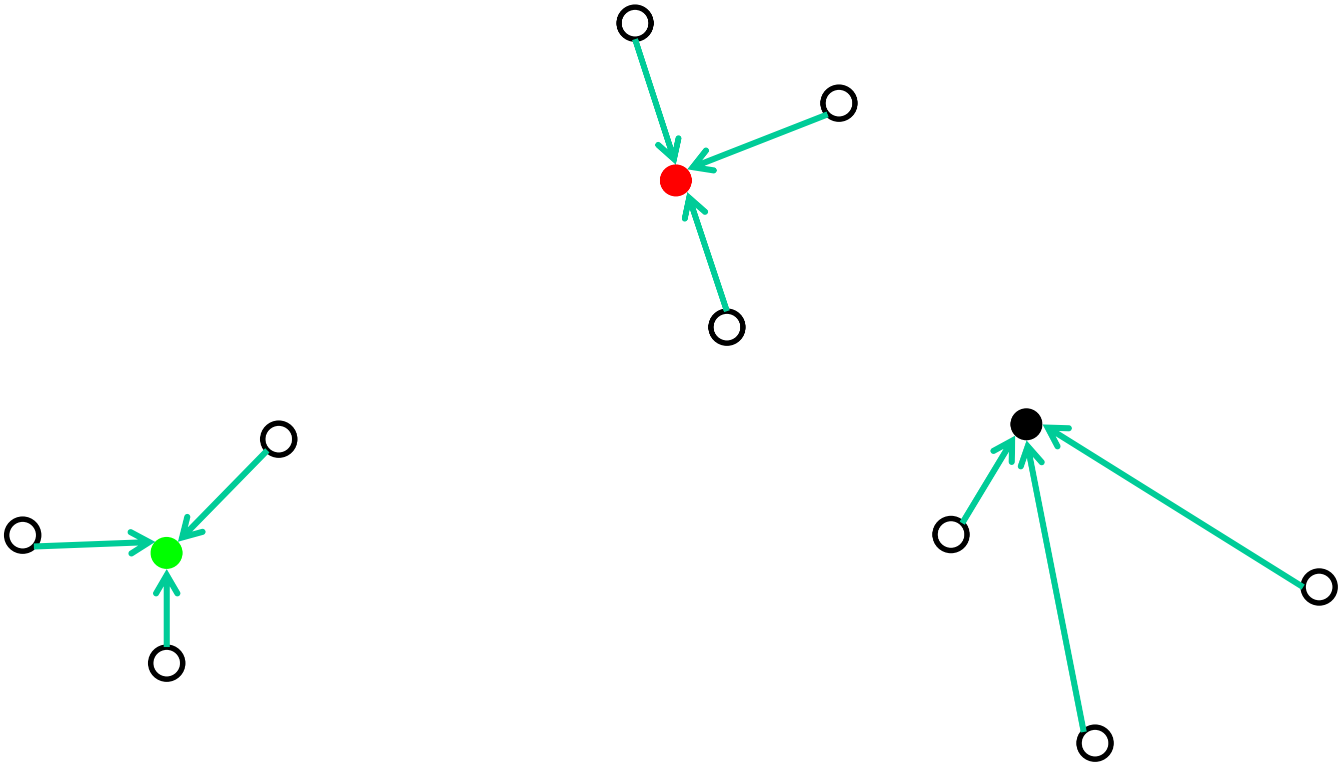
Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering



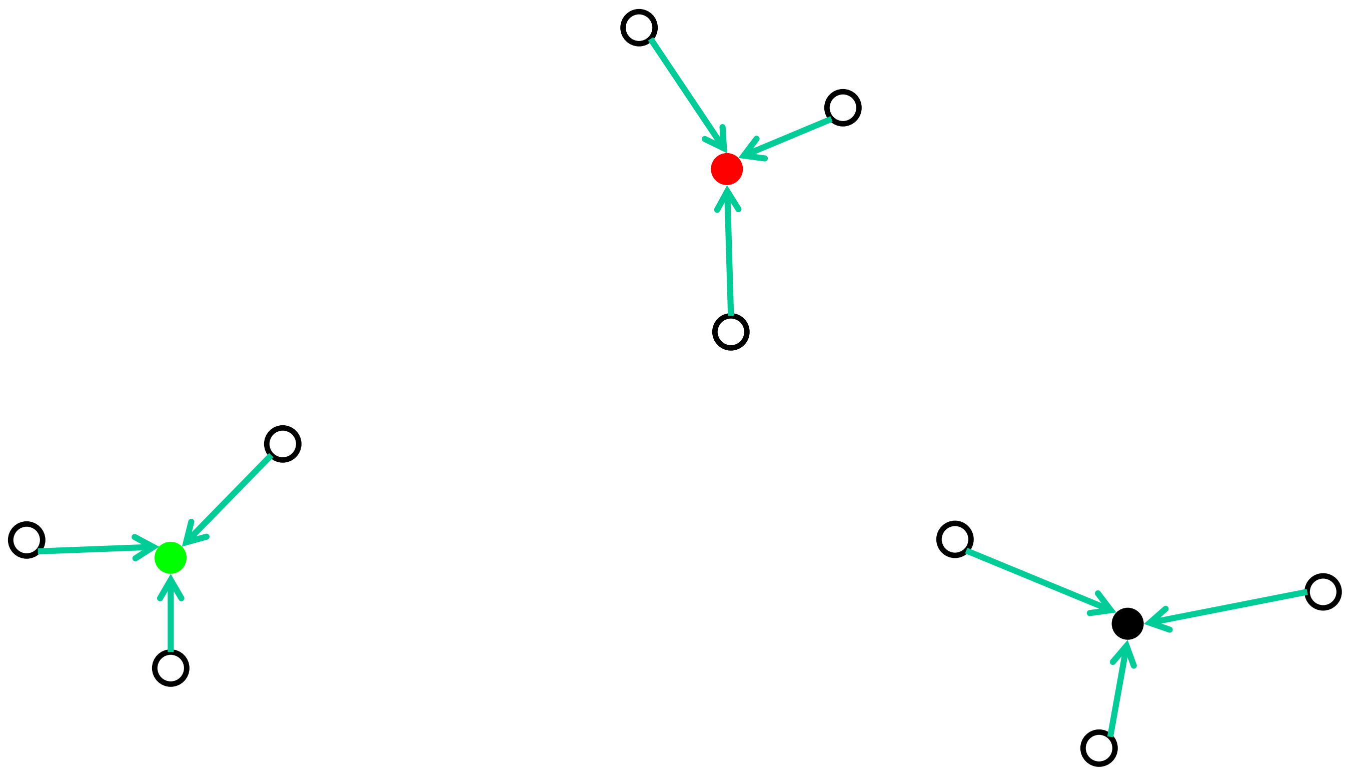
Lloyd's method: Random Initialization

Assign each point to its nearest center



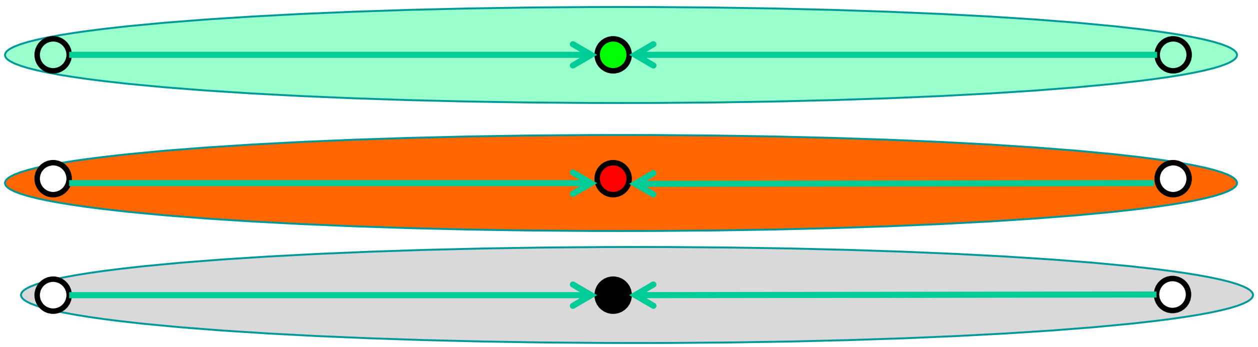
Lloyd's method: Random Initialization

Recompute optimal centers given a fixed clustering



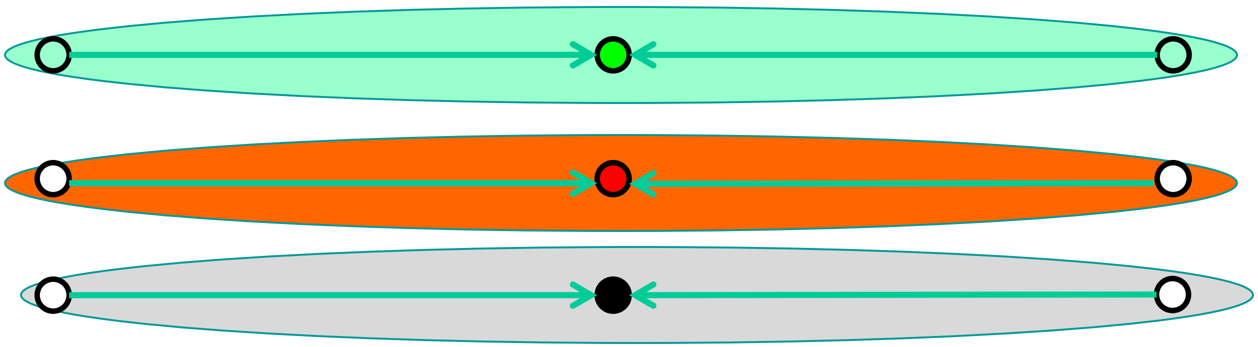
Get a good quality solution in this example.

Lloyd's method: Performance



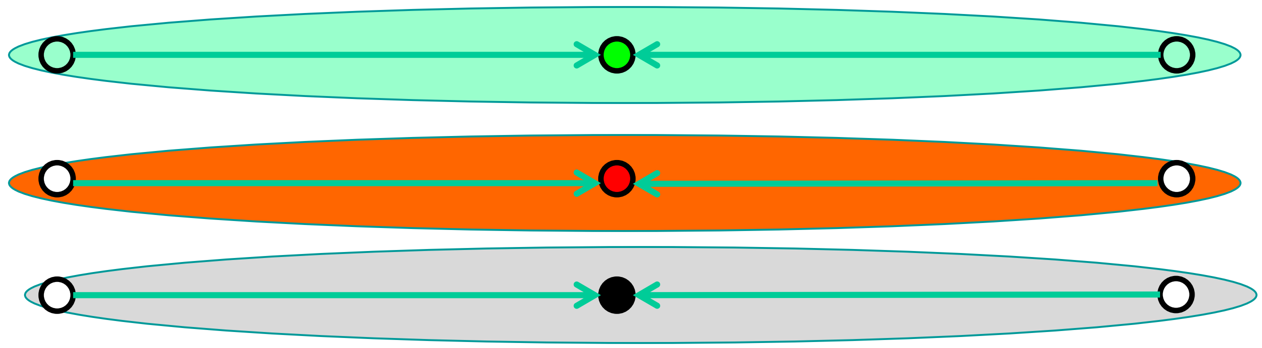
It always converges, but it may converge at a local optimum that is different from the global optimum, and in fact could be arbitrarily worse in terms of its score.

Lloyd's method: Performance

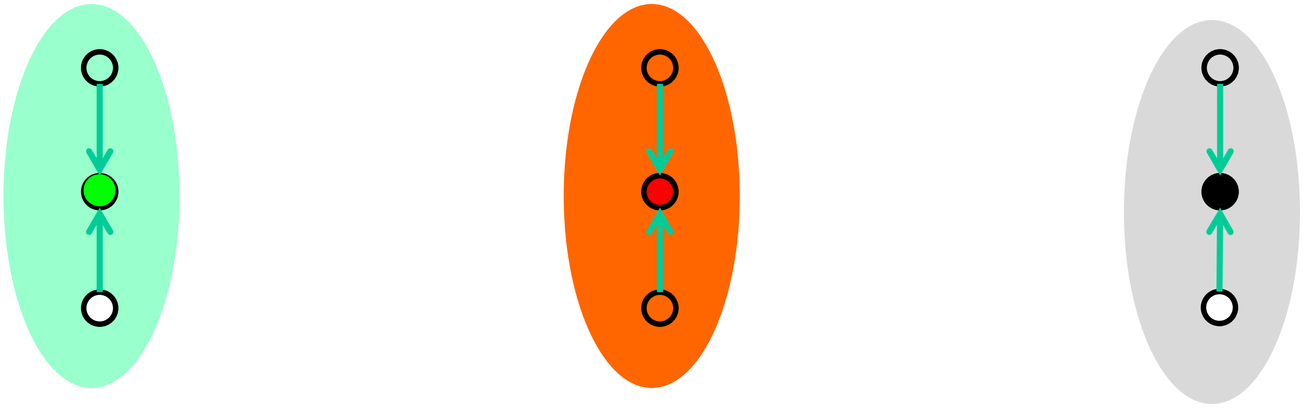


Local optimum: every point is assigned to its nearest center and every center is the mean value of its points.

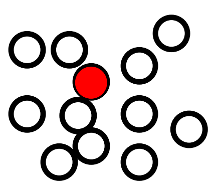
Lloyd's method: Performance



.It is arbitrarily worse than optimum solution….

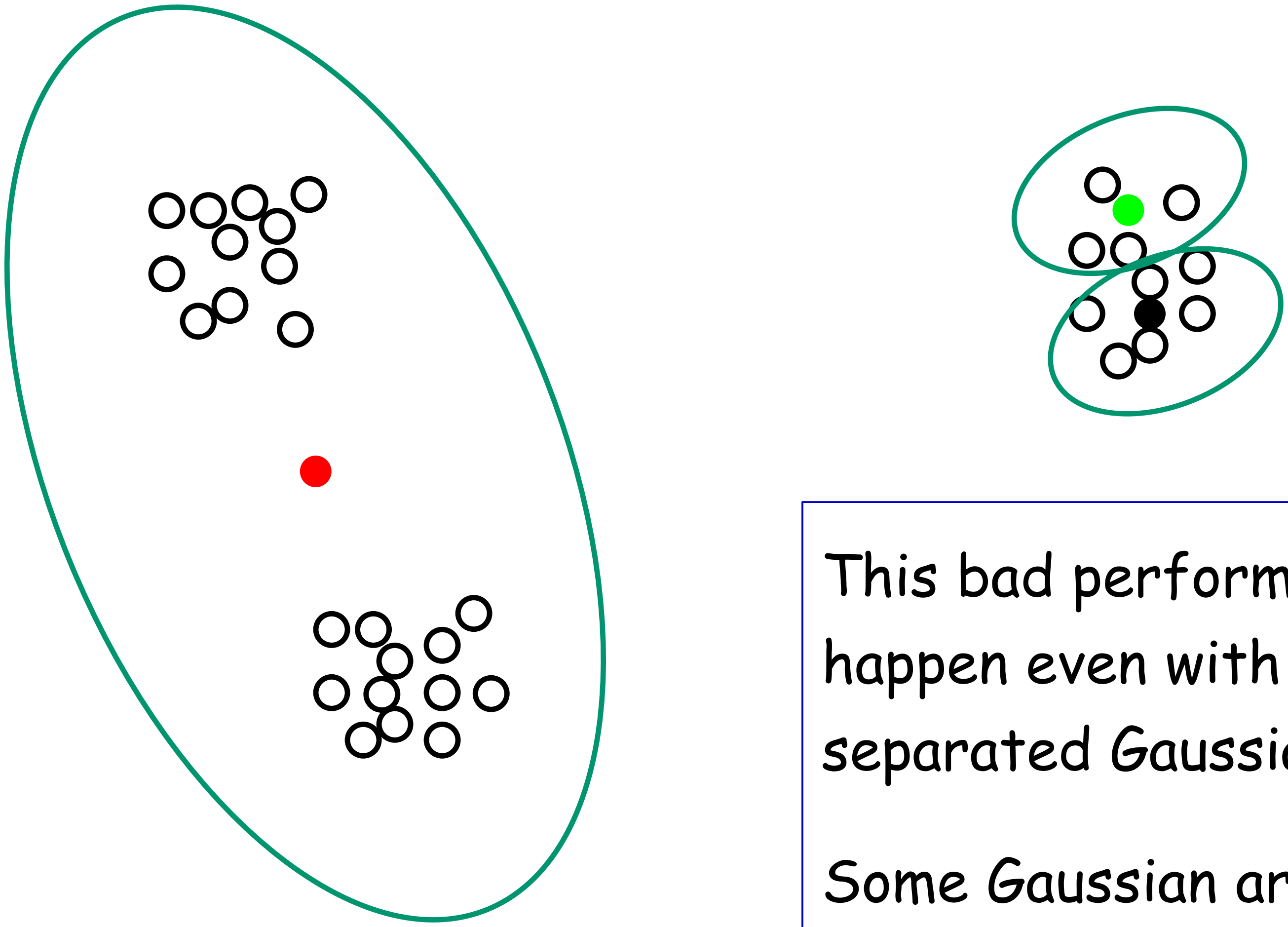


Lloyd’s method: Performance



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[each initial center is in a different Gaussian] $\approx \frac{k!}{k^k} \approx \frac{1}{e^k}$

  - Becomes unlikely as k gets large.

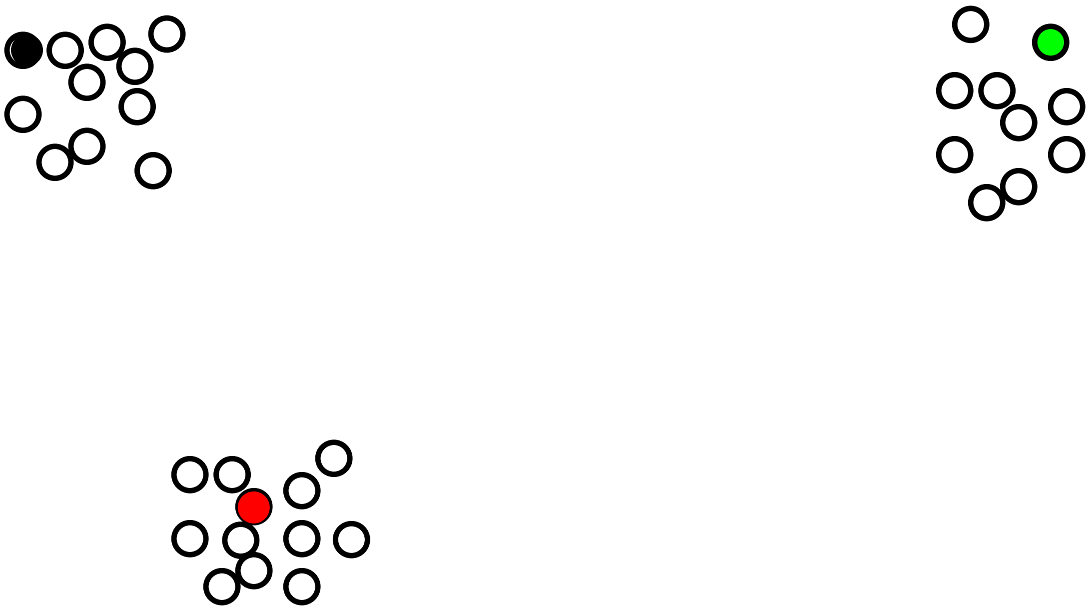
Another Initialization Idea: Furthest Point Heuristic

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

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

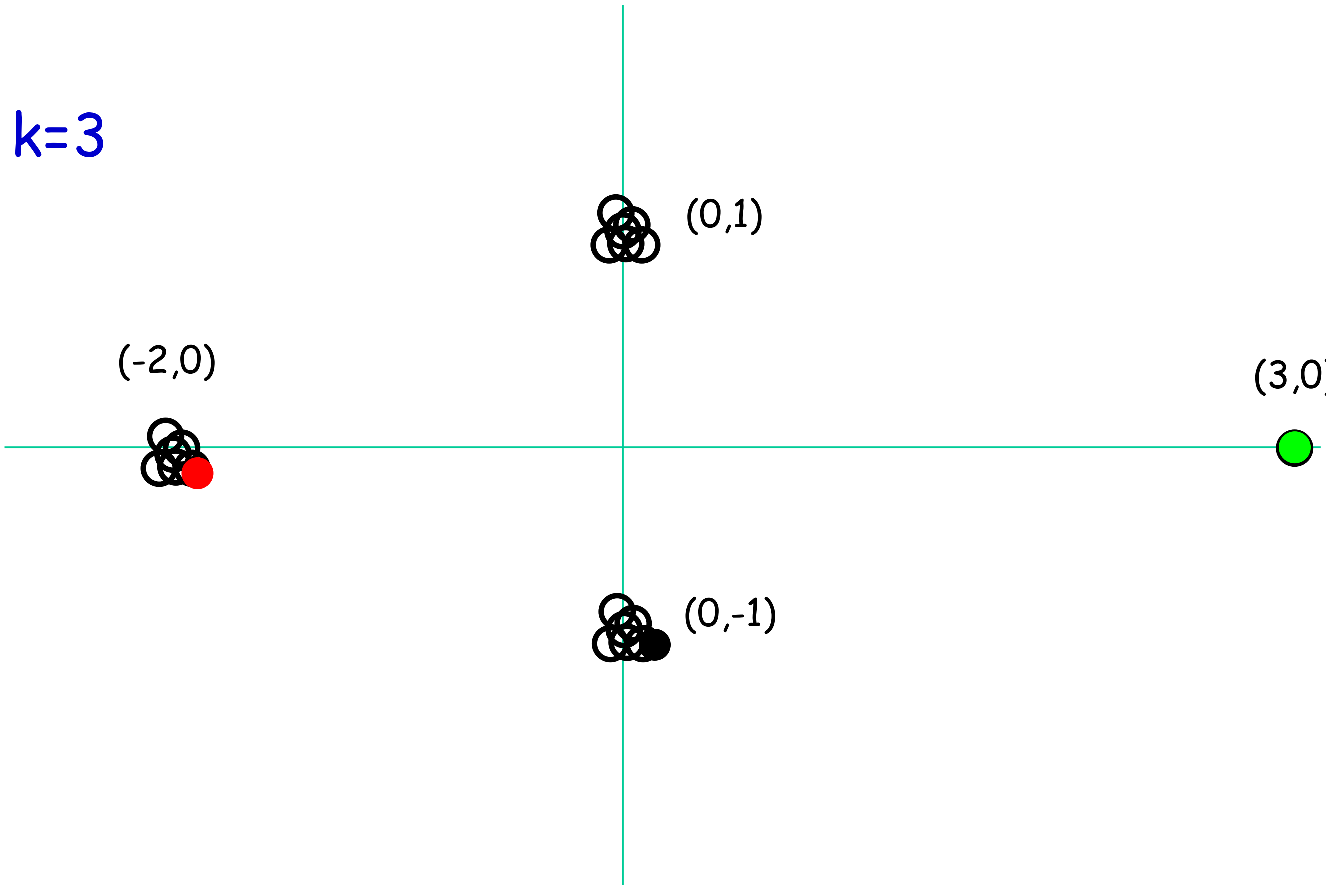
Fixes the Gaussian problem. But it can be thrown off by outliers….

Furthest point heuristic does well on previous example



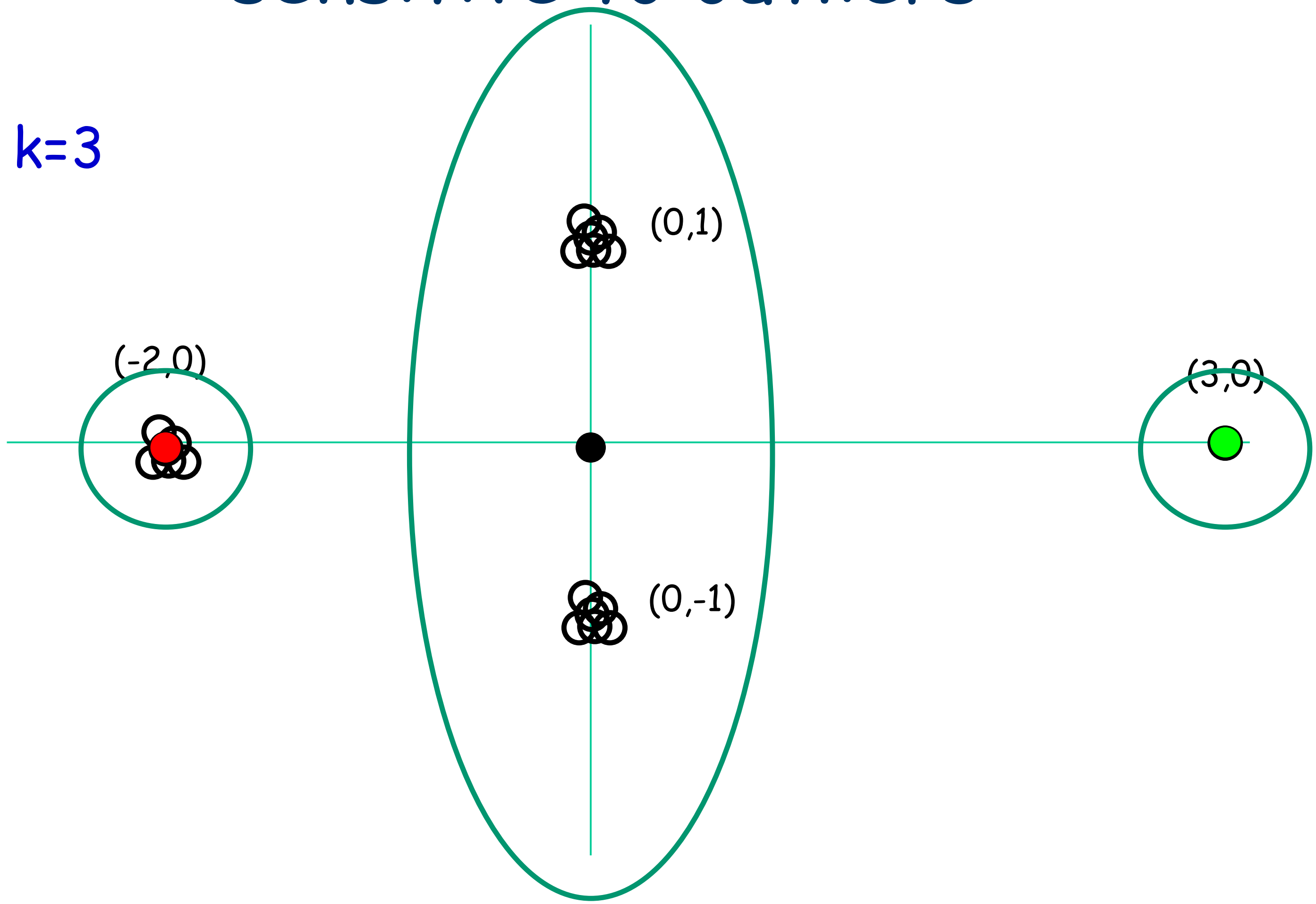
Furthest point initialization heuristic sensitive to outliers

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 $\mathrm{D}^2(\mathbf{x})$.

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

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

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

  - $\alpha = 0$, 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: $C_j \leftarrow \{x \in S$ whose closest center is $\mathbf{c_j}\}$
  - For each j: $\mathbf{c_j} \leftarrow$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!

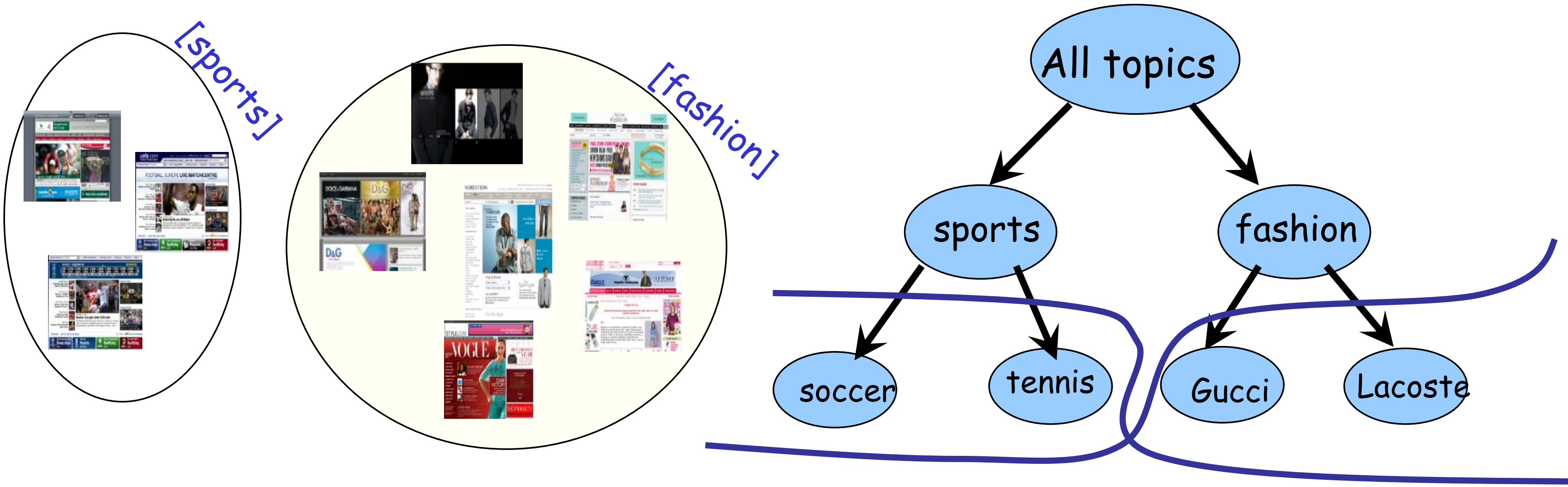
K-means++/ Lloyd's Summary

- 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.

What You Should Know

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