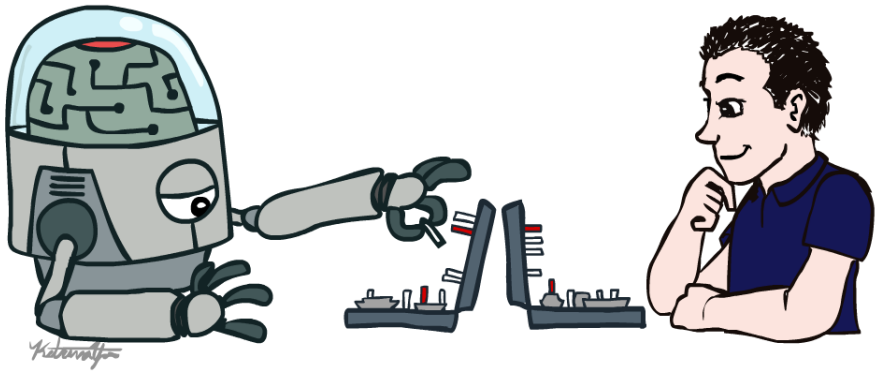
CSE 3521:
Introduction to Artificial Intelligence



[Many slides are adapted from the UC Berkeley. CS188 Intro to AI at UC Berkeley and previous CSE 3521 course at OSU.]

THE OHIO STATE UNIVERSITY

Clustering

THE OHIO STATE UNIVERSITY

Today

- An overview of unsupervised learning
  - Clustering
  - Generative models

- K-means clustering
- Agglomerative clustering
- Gaussian mixture models (GMM): probabilistic clustering & generative models

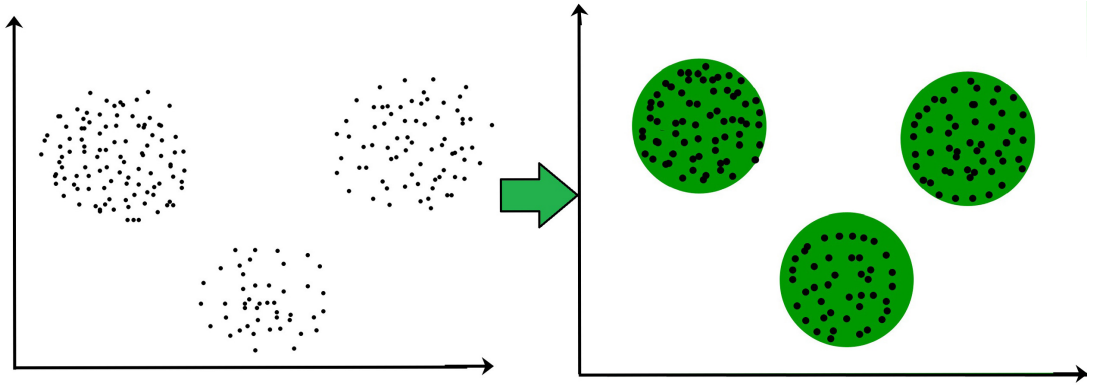
Unsupervised learning

- Data type: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$

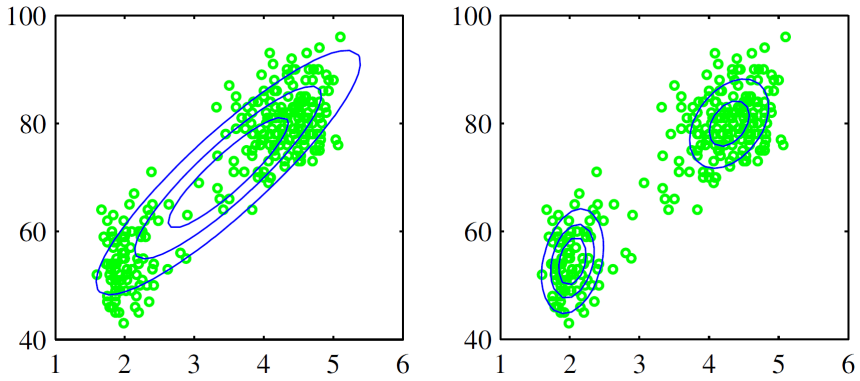
- Goal:
  - Discover the structure (e.g., clusters, groups, or classes) of the data instances
  - Estimate the distribution/density of the data instances
  - Learn to generate and sample data instances

Unsupervised learning

- Clustering:

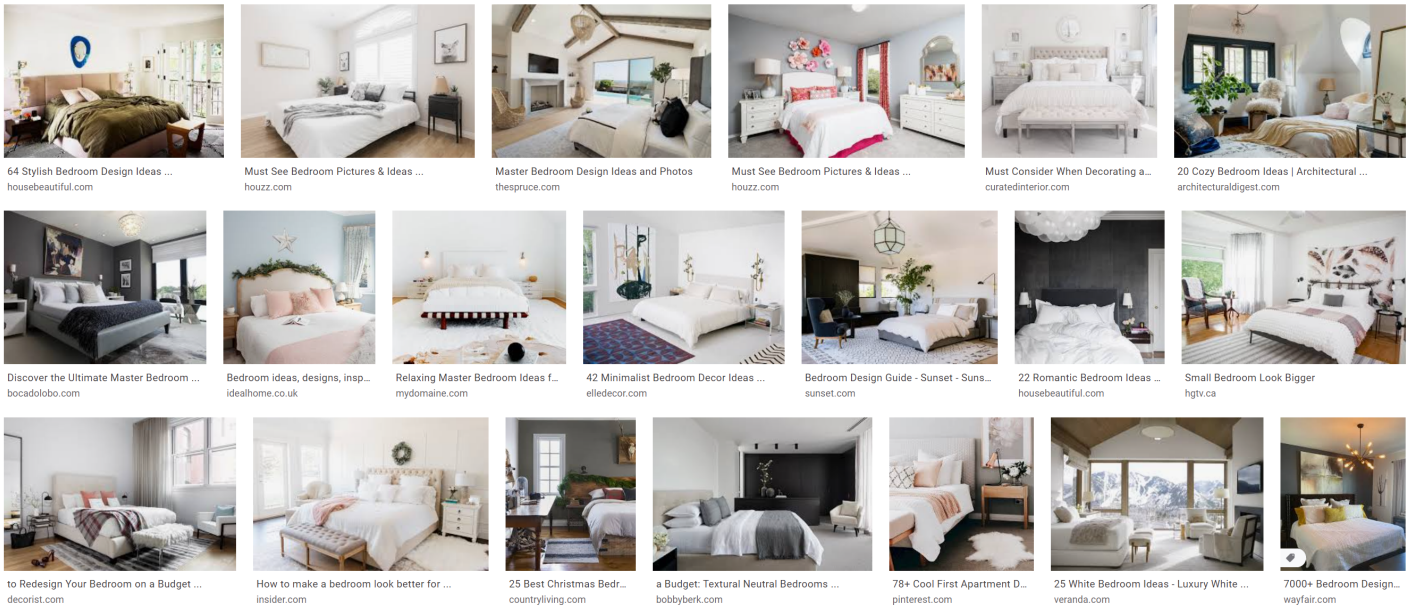


- Distribution/Density estimation
  - Can sample from the distribution to generate new data



Unsupervised learning

- Generative models
  - After seeing many data, can the model generate one?
  - Example: Google bedroom image search (real images)

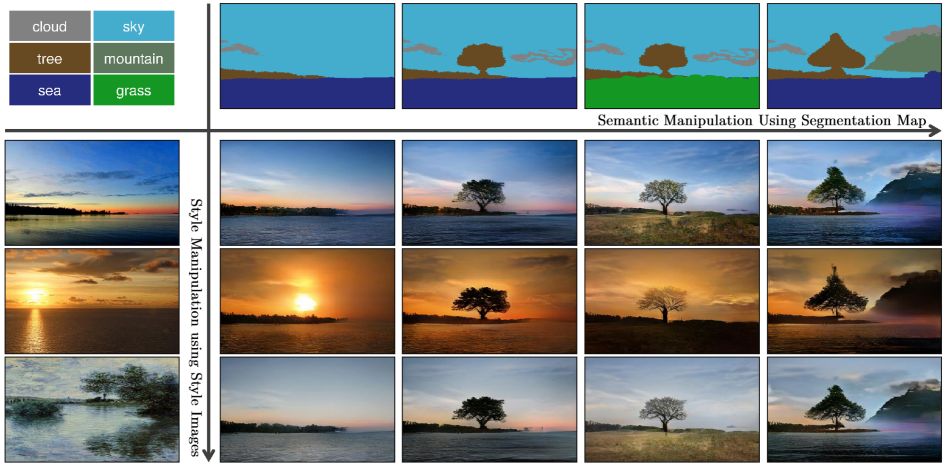


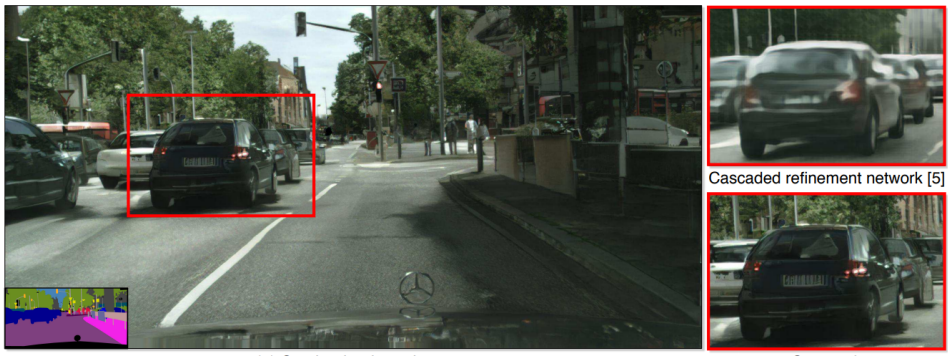
Unsupervised learning

- Generative models
  - After seeing many data, can the model generate one?
  - Example: Machine generated images [Radford et al., ICLR 2016]

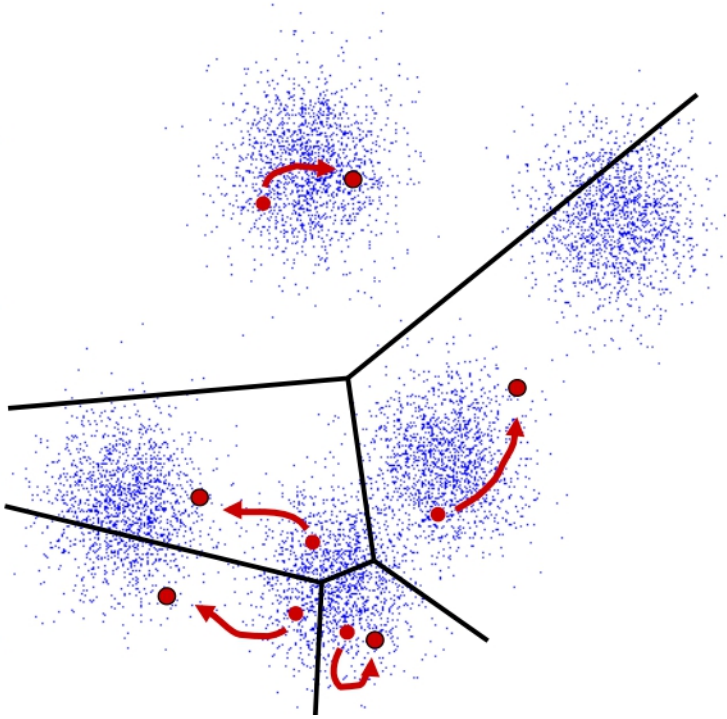
Unsupervised learning

- Generative models (variants)
  - Image “translation”
    - [Huang et al., ECCV 2018]
  - Conditional image generation
    - [Wang et al., CVPR 2018]
    - [Park et al., CVPR 2019]





Clustering



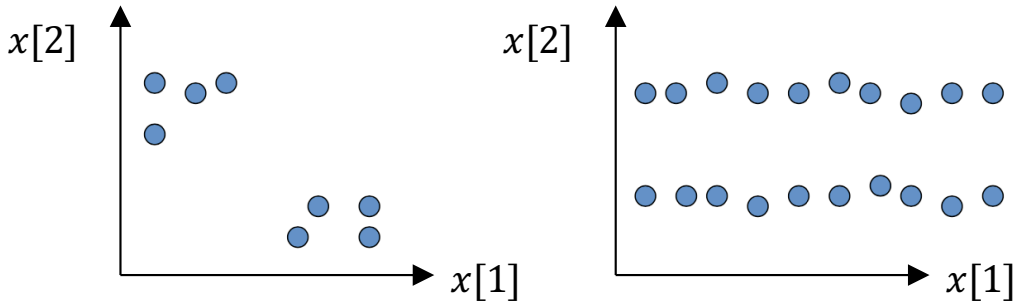
THE OHIO STATE UNIVERSITY

Clustering

- Clustering systems:
  - Unsupervised learning
  - Detect patterns in unlabeled data
    - E.g. group emails or search results
    - E.g. find categories of customers
    - E.g. detect anomalous program executions
  - Useful when we don't know what you are looking for
    - Requires data, but no labels
    - May get gibberish

Clustering

- Basic idea: group together similar instances
  - Example: 2D point patterns

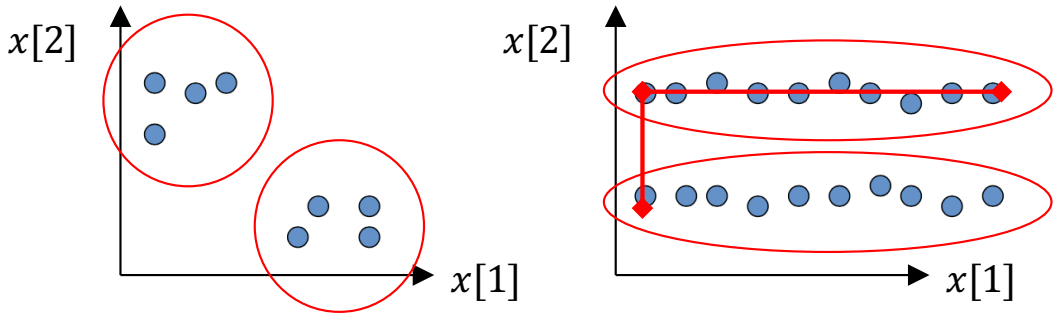


- What does "similar" mean?
  - One option: small (squared) Euclidean distance (i.e., L_2 distance)

$$dist(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 = (\boldsymbol{x} - \boldsymbol{x}')^T(\boldsymbol{x} - \boldsymbol{x}) = \sum_{d=1}^{D} (x[d] - x'[d])^2$$

Clustering

- Basic idea: group together similar instances
  - Example: 2D point patterns



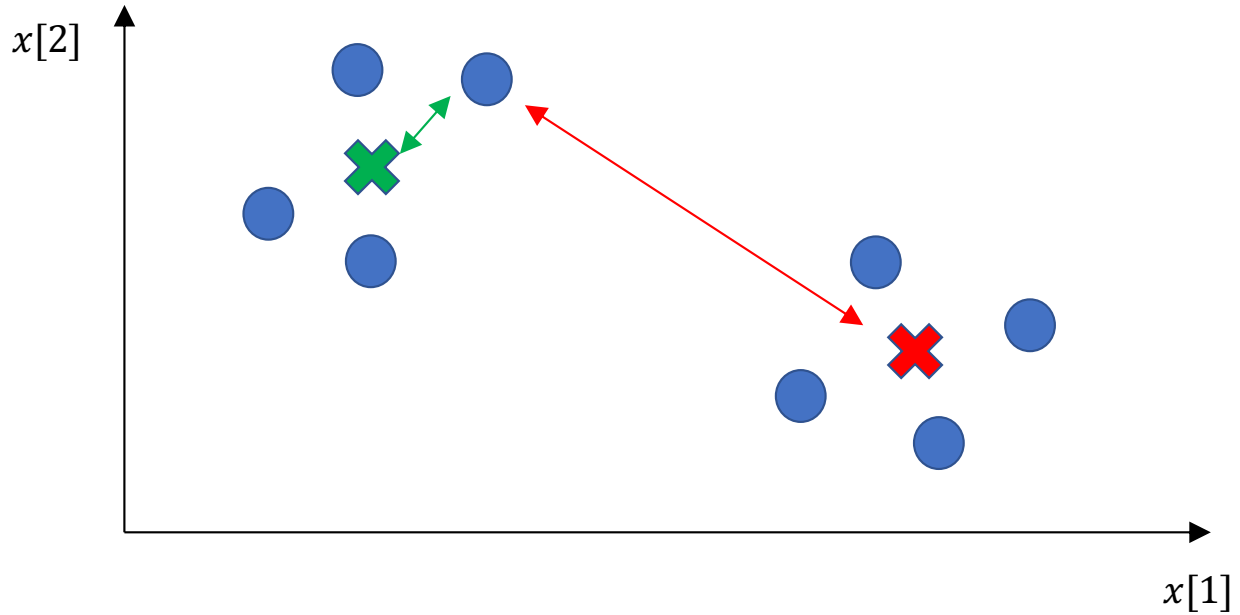
- Need other distance metrics
- Feature normalization
- Other algorithms

- What does "similar" mean?
  - One option: small (squared) Euclidean distance (i.e., L_2 distance)

$$dist(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2 = (\boldsymbol{x} - \boldsymbol{x}')^T(\boldsymbol{x} - \boldsymbol{x}) = \sum_{d=1}^{D} (x[d] - x'[d])^2$$

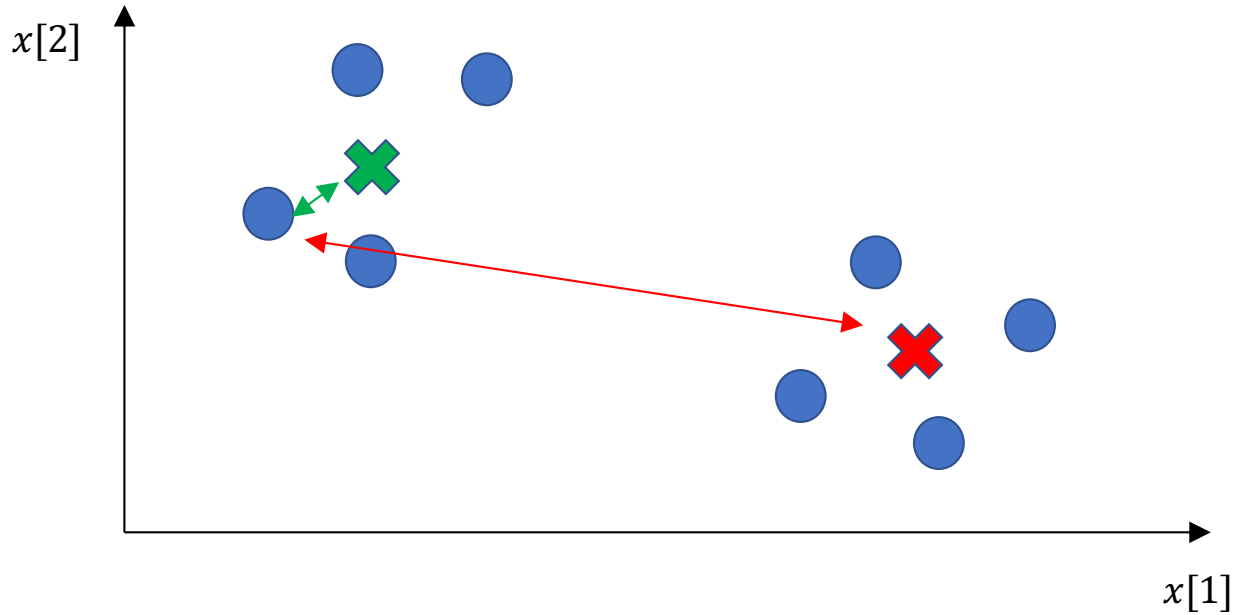
K-Means

- Find K centers and group data instances into the K centers
  - Example: K = 2
  - Assign each data instance to the closet (most similar, smallest distant) center



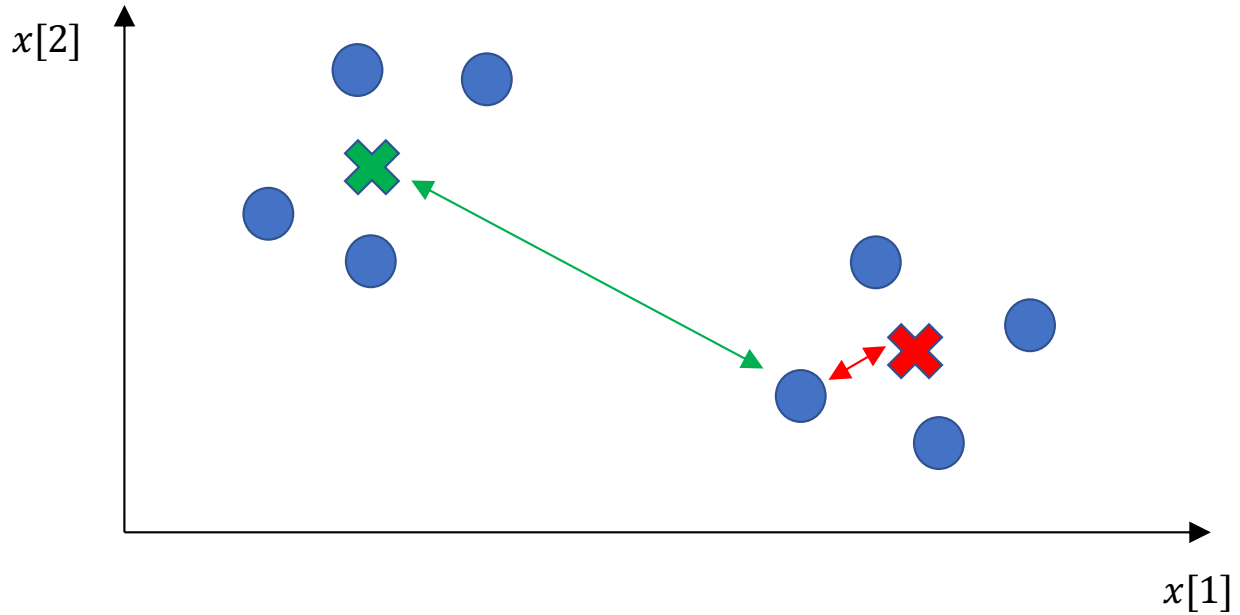
K-Means

- Find K centers and group data instances into the K centers
  - Example: K = 2
  - Assign each data instance to the closet (most similar, smallest distant) center



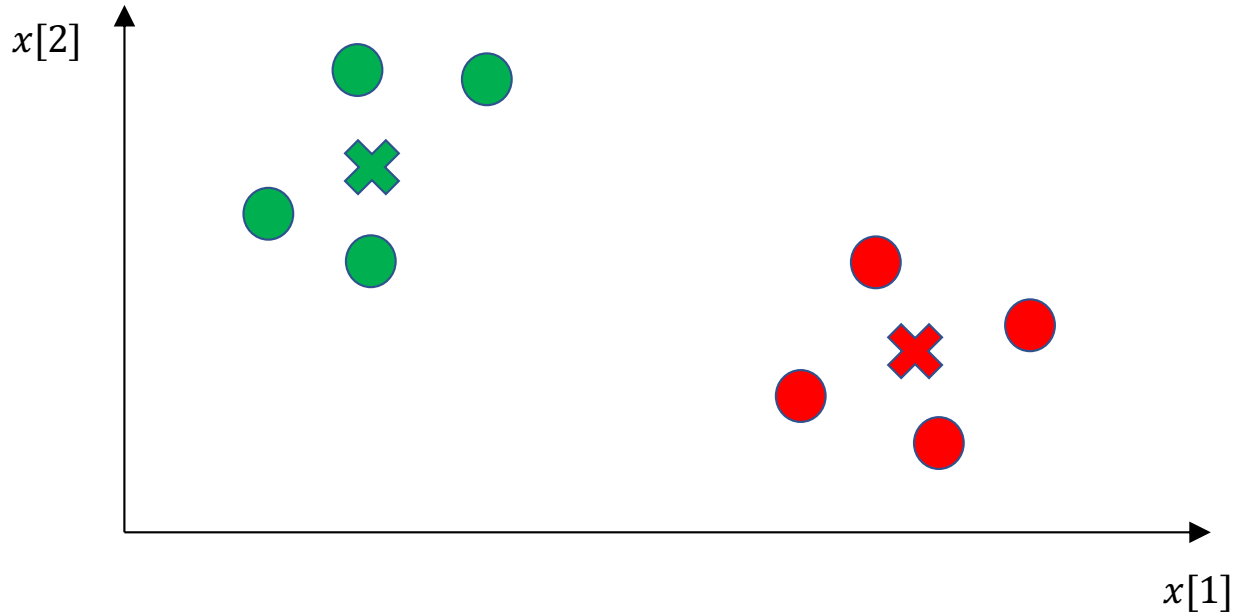
K-Means

- Find K centers and group data instances into the K centers
  - Example: K = 2
  - Assign each data instance to the closet (most similar, smallest distant) center



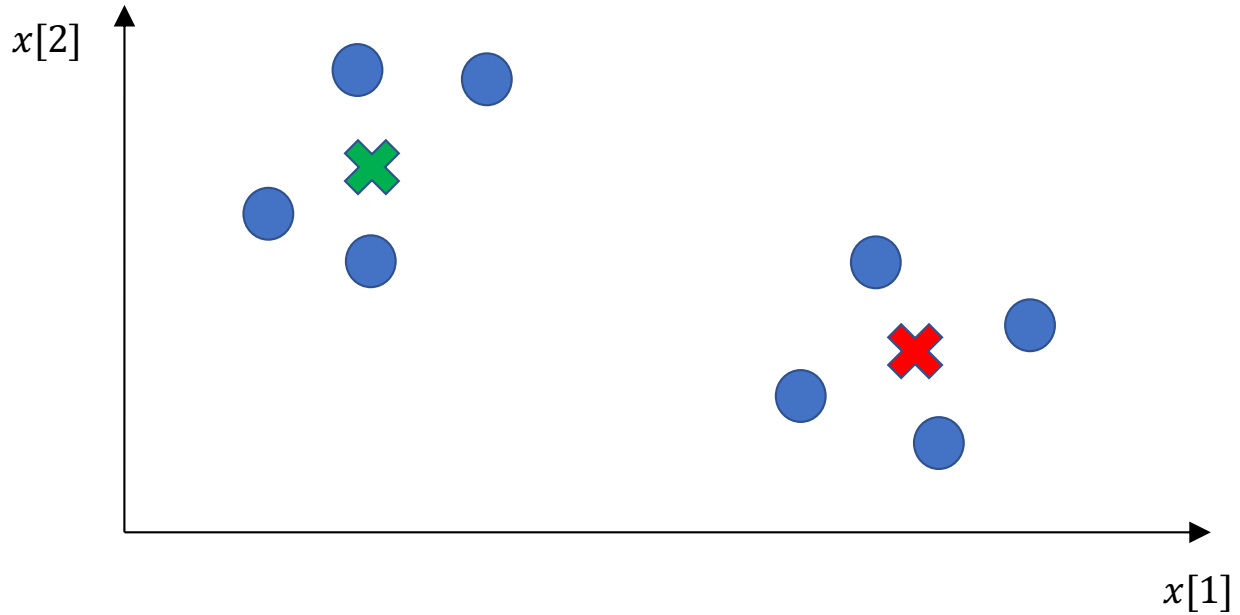
K-Means

- Find K centers and group data instances into the K centers
  - Example: K = 2
  - Assign each data instance to the closet (most similar, smallest distant) center

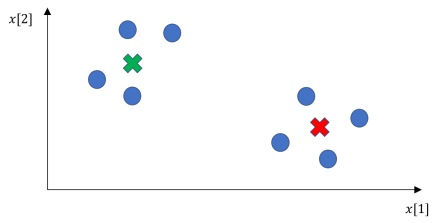


K-Means

- Find K centers and group data instances into the K centers simultaneously
  - Chicken-egg problem: need to have centers before assignments, but where are the centers?



K-Means



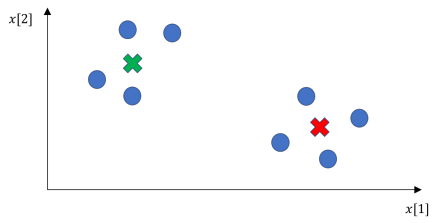
- Formulate as an optimization problem (total distance to the centers)

$$\underset{\{y_i \in \{1,\ldots,K\}\}_{i=1}^{N},\ \{\boldsymbol{c}_k \in \mathbb{R}^D\}_{k=1}^{K}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{\{i \mid y_i = k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

All data instances assigned to group k

  - To find centers (K of them) and group assignments (N data instances) simultaneously

K-Means



- Formulate as an optimization problem (total distance to the centers)

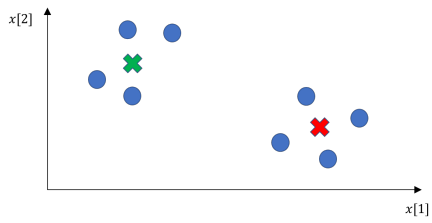
$$\underset{\{y_i \in \{1,\dots,K\}\}_{i=1}^{N},\, \{\boldsymbol{c}_k \in \mathbb{R}^D\}_{k=1}^{K}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{\{i \mid y_i = k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

All data instances assigned to group k

  - To find centers (*K* of them) and group assignments (*N* data instances) simultaneously
  - Given centers, then for each data point *i*, the assignment is independent

$$\underset{y_i \in \{1,\dots,K\}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i \mid y_i = k} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2 \Rightarrow y_i = \underset{k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

K-Means



- Formulate as an optimization problem (total distance to the centers)

$$\underset{\{y_i \in \{1,\ldots,K\}\}_{i=1}^{N},\, \{\boldsymbol{c}_k \in \mathbb{R}^D\}_{k=1}^{K}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{\{i \mid y_i = k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

All data instances assigned to group k

  - To find centers (K of them) and group assignments (N data instances) simultaneously
  - Given assignments, then finding the center for each group k is independent

$$\underset{\boldsymbol{c}_k \in \mathbb{R}^D}{\operatorname{argmin}} \sum_{\{i \mid y_i = k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2 \implies \boldsymbol{c}_k = \frac{1}{|\{i \mid y_i = k\}|} \sum_{\{i \mid y_i = k\}} \boldsymbol{x}_i$$

K-Means: an iterative clustering algorithm

- Pick *K* random points as cluster centers (means): $c_1 \ldots c_K$
- Alternate (the error is monotonically decreasing):
  - Assign each example x_i to the mean c_k that is closest to it

$$\underset{y_i \in \{1,\ldots,K\}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i|y_i=k} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2 \Rightarrow y_i = \underset{k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

  - Set each mean c_k to the average of its assigned points

$$\underset{\boldsymbol{c}_k \in \mathbb{R}^D}{\operatorname{argmin}} \sum_{\{i \mid y_i=k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2 \Rightarrow \boldsymbol{c}_k = \frac{1}{|\{i \mid y_i = k\}|} \sum_{\{i \mid y_i=k\}} \boldsymbol{x}_i$$

  - Stop when no points assignments change

K-Means

- Data: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$
- An iterative clustering algorithm
  - Pick K random cluster centers, $c_1 \ldots c_K$
  - For t=1 … T: [or, stop if assignments don t change]
    - for i = 1 ... N: [update cluster assignments]

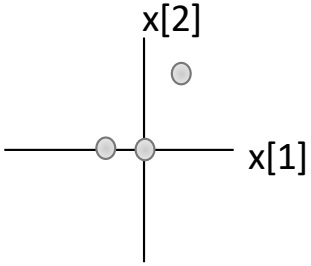
$$y_i = \underset{k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

    - for k= 1 … K: [update cluster centers]

$$\boldsymbol{c}_k = \frac{1}{|\{i \mid y_i = k\}|} \sum_{\{i \mid y_i = k\}} \boldsymbol{x}_i$$

K-Means

	x[1]	x[2]
i=1	-1	0
i=2	0	0
i=3	2	2



Random cluster means:
c_1=[-1,0], c_2=[0,0]
t = 0:

	i=1	i=2	i=3
c_1	0	1	13
c_2	1	0	8

- $y_1 = \text{argmin}_k \text{ dist}(x_1,c_k) = 1$
- $y_2 = \text{argmin}_k \text{ dist}(x_2,c_k) = 1$
- $y_3 = \text{argmin}_k \text{ dist}(x_3,c_k) = 2$

- $c_1 = (1/1) * [-1,0] = [-1,0]$
- $c_2 = ((1/2) * ([0,0]+[2,2]) = [1,1]$

c_1=[-1,0], c_2=[1,1]
t = 1 :

	i=1	i=2	i=3
c_1	0	1	13
c_2	4	4	18

- $y_1 = \text{argmin}_k \text{ dist}(x_1,c_k) = 1$
- $y_2 = \text{argmin}_k \text{ dist}(x_2,c_k) = 1$
- $y_3 = \text{argmin}_k \text{ dist}(x_3,c_k) = 2$

- $c_1 = (1/2) * ([-1,0]+[0,0]) = [-0.5,0]$
- $c_2 = (1/1) * ([2,2]) = [2,1]$

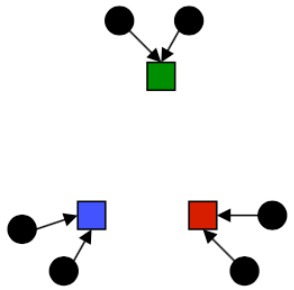
- t=2:
- Stop!!
- y_i won't change (verify!)

K-Means optimization: a second glance

- Consider the total distance to the centers:

$$\sum_{k=1}^{K} \sum_{\{i \mid y_i=k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

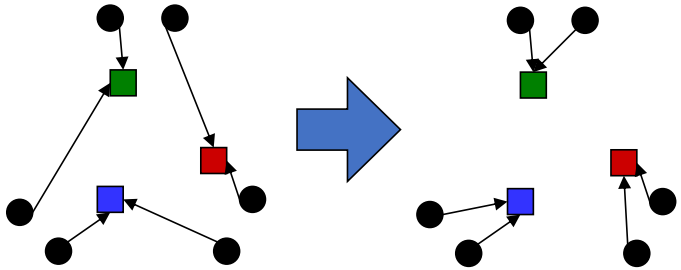
- Two stages each iteration:
  1. Update assignments: fix means, change assignments
  2. Update means: fix assignments, change means
- Will it converge?
  - Yes!, if you can argue that each update can decrease the error



Phase I: Update Assignments

- For each point, re-assign to closest mean:

$$y_i = \underset{k}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$



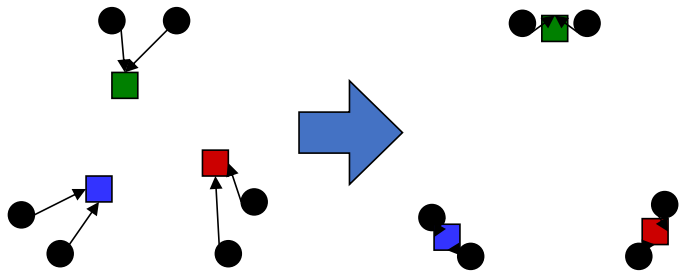
- Can only decrease the total distance

$$\sum_{k=1}^{K} \sum_{\{i \mid y_i = k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

Phase II: Update Means

- Move each mean to the average of its assigned points:

$$\boldsymbol{c}_k = \frac{1}{|\{i \mid y_i = k\}|} \sum_{\{i \mid y_i = k\}} \boldsymbol{x}_i$$

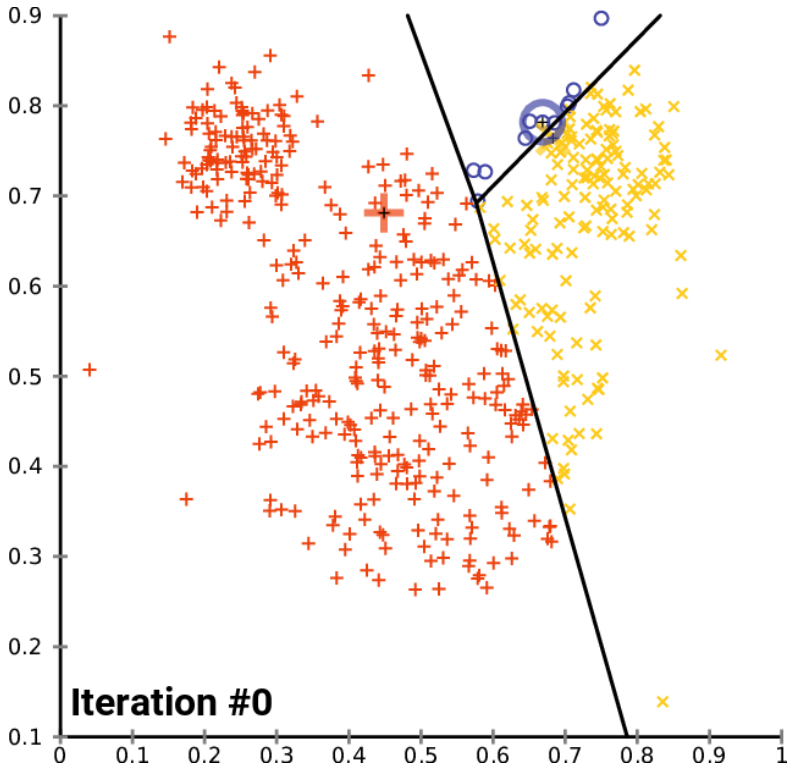


- Also can only decrease total distance… (Why?)

$$\sum_{k=1}^{K} \sum_{\{i \mid y_i = k\}} \|\boldsymbol{x}_i - \boldsymbol{c}_k\|_2^2$$

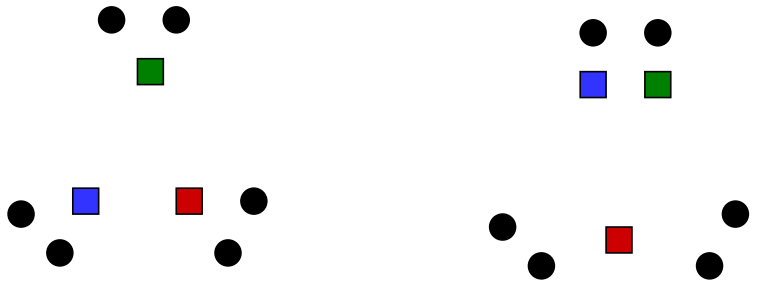
- The point c_k with the least squared Euclidean distance to points $\{x_i \mid y_i = k\}$ is their mean
  - Proof: take partial derivative w.r.t. c_k and set it to 0 to find c_k

K-means: animation



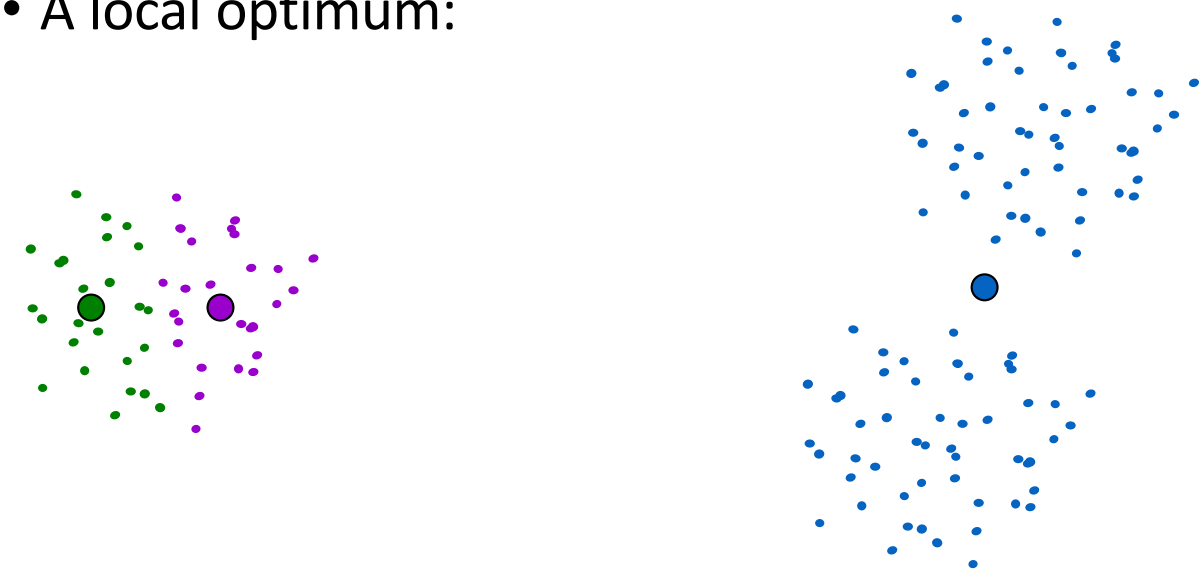
Initialization

- K-means is non-deterministic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
  - Various schemes for preventing this kind of thing:
    - Variance-based split / merge, initialization heuristics



K-Means Getting Stuck

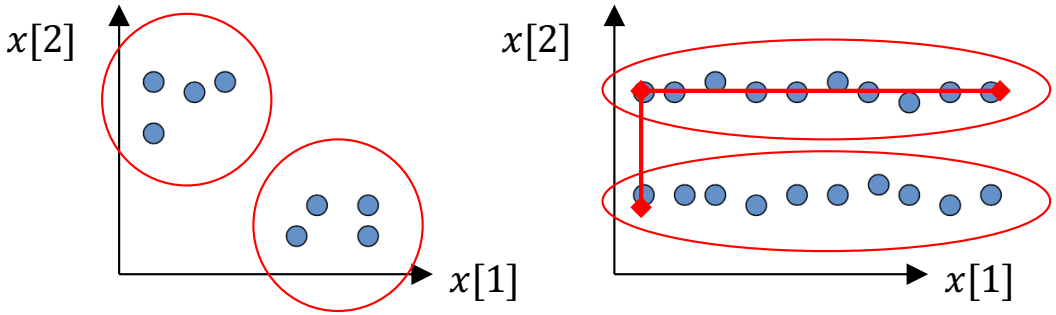
- A local optimum:



Ideally, we want the purple center to take half of the blue points, but the initial centers locations lead to a sub-optimal solutions.

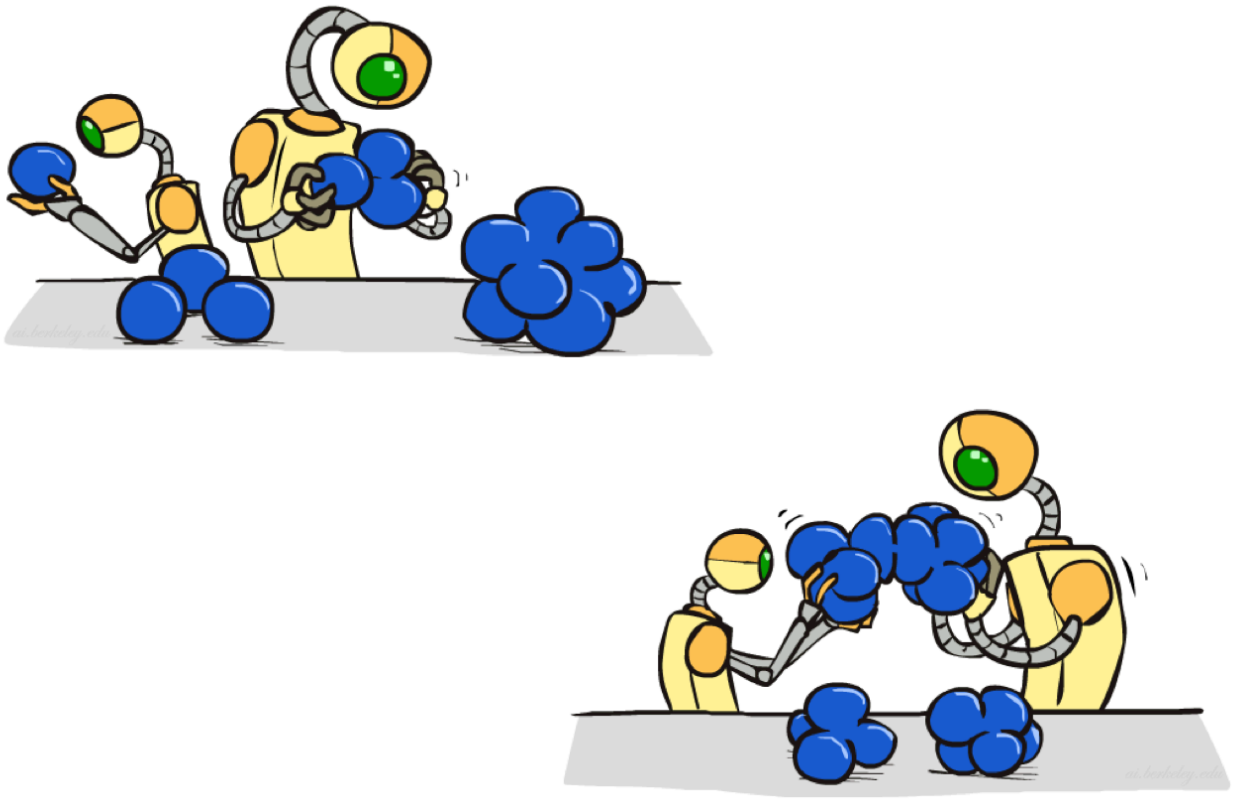
K-Means Questions

- What distance to use?
  - Square Euclidean: mean
  - L1 distance: medium
  - Perform feature transform before K-means



- Variants:
  - K-medoid: chooses data points as centers (medoids) and can be used with arbitrary distances

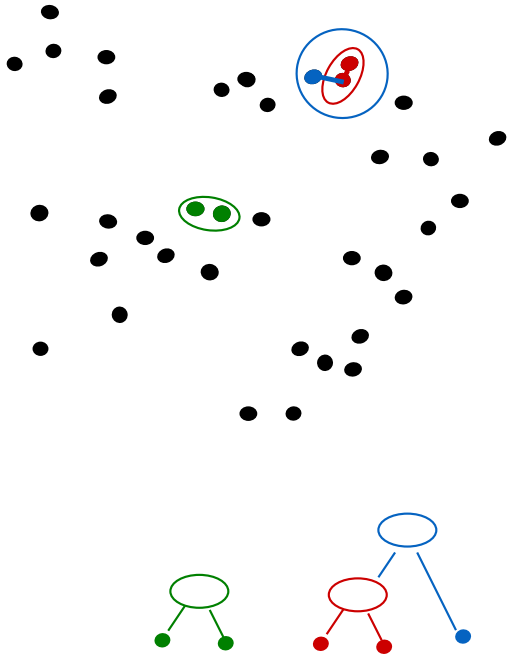
Agglomerative Clustering



Agglomerative Clustering

- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters

- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there is only one cluster left



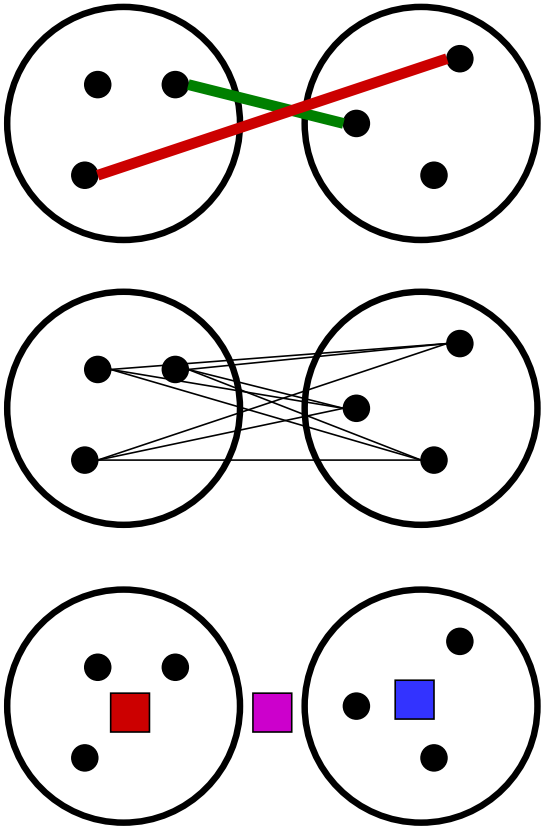
- Produces not one clustering, but a family of clusters represented by a dendrogram

Agglomerative Clustering

- How should we define “closest” for clusters with multiple elements?

- Many options
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
  - Ward’s method (min variance, like k-means)

- Different choices create different clustering behaviors



Agglomerative Clustering

complete link

single link

Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link

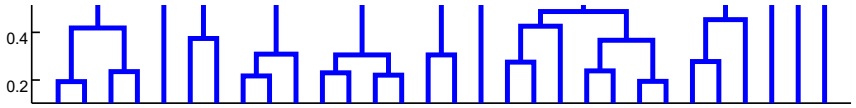
single link

Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link

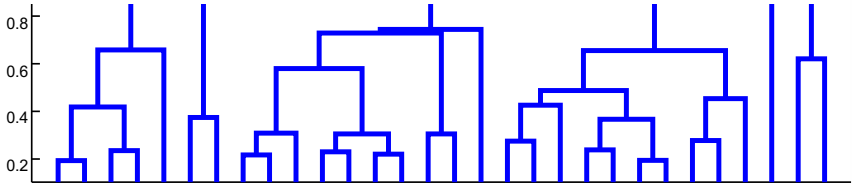
single link



Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link

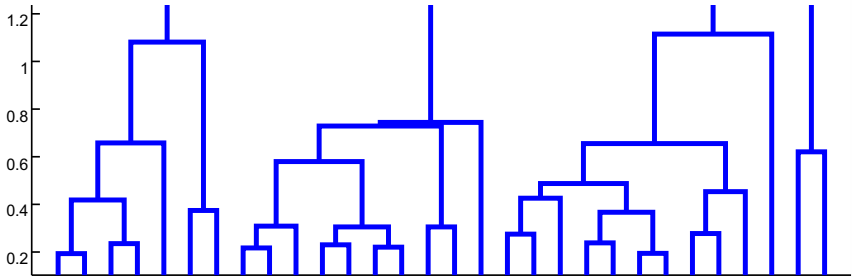


single link

Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link

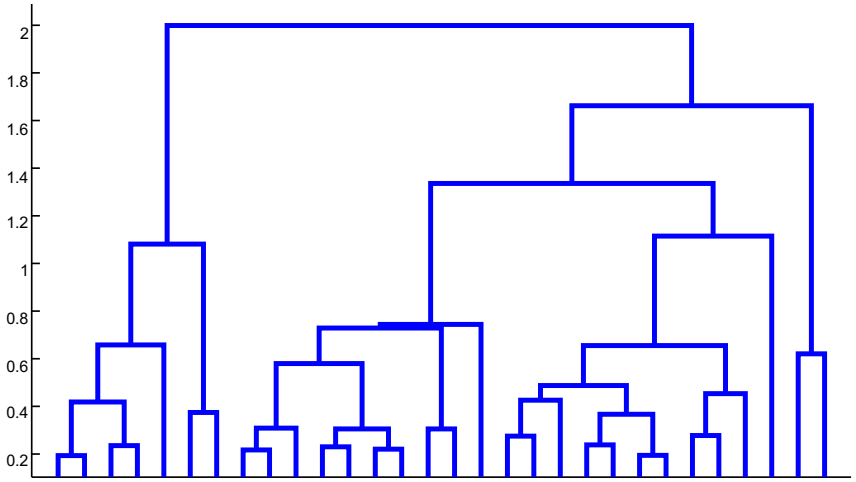


single link

Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link

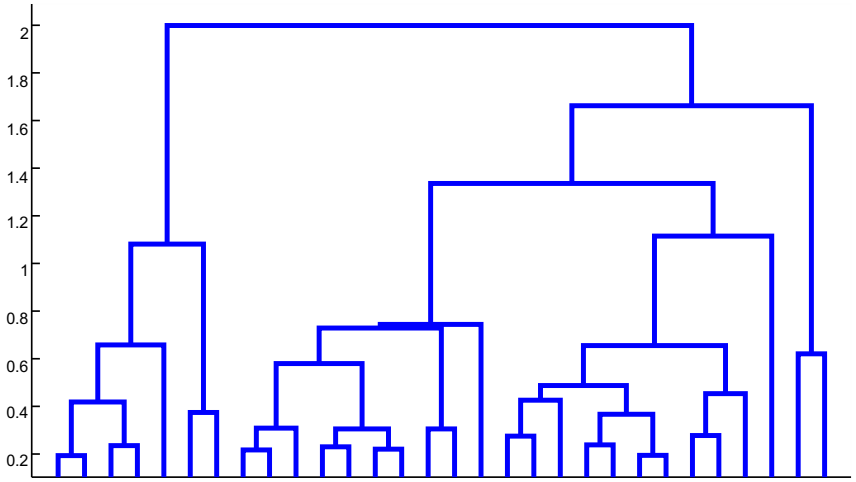


single link

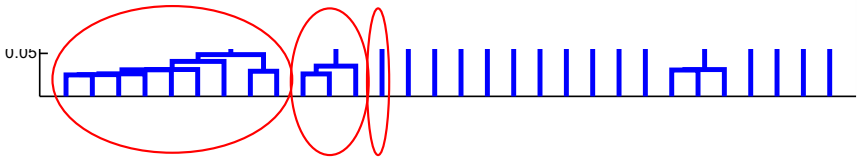
Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link



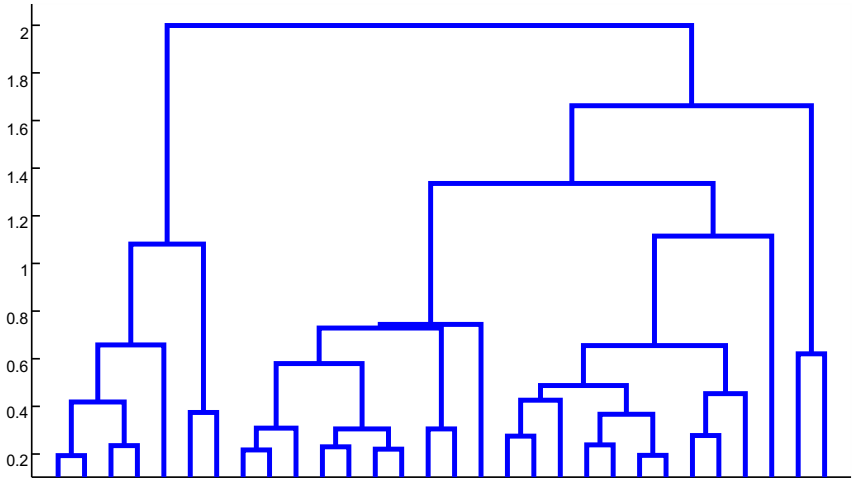
single link



Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link

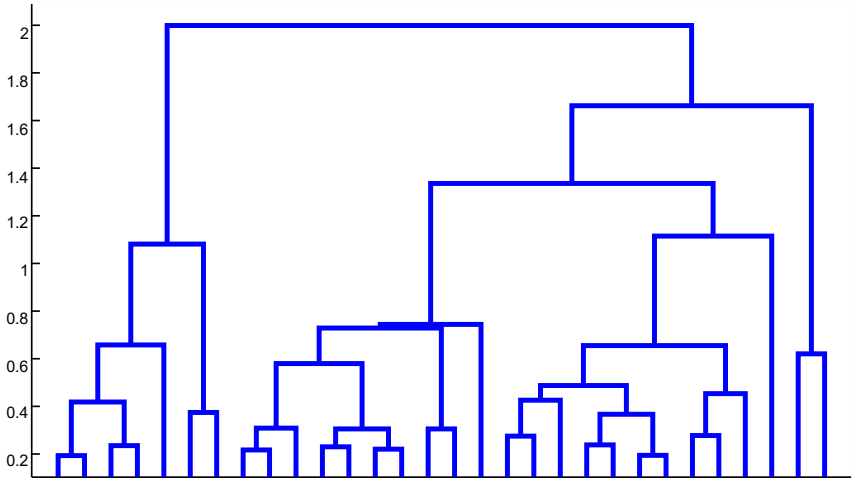


single link

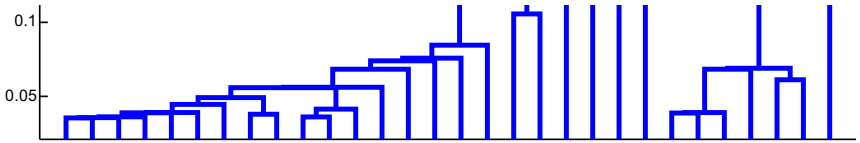
Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link



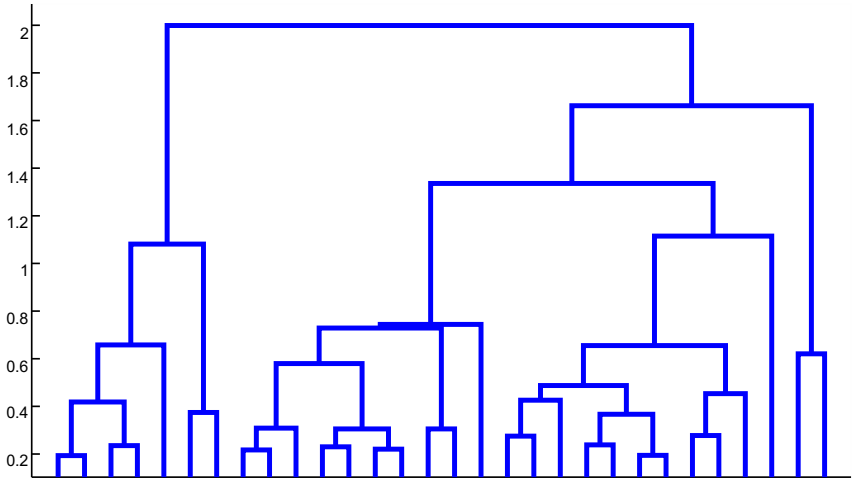
single link



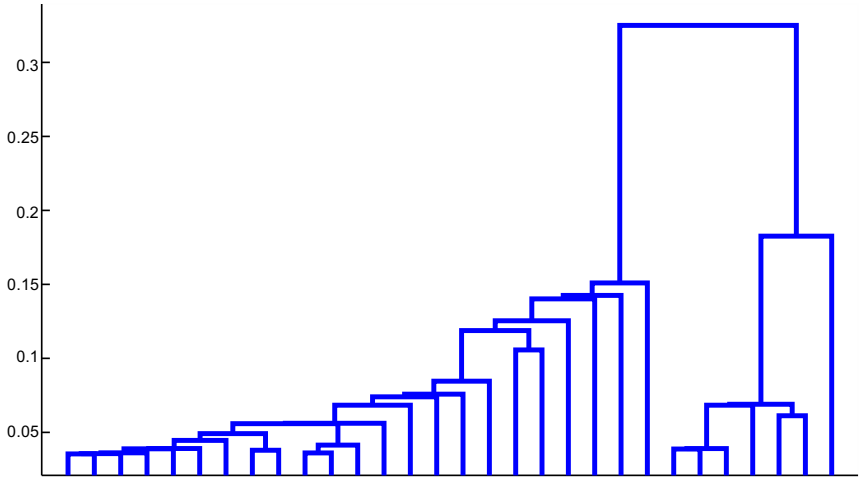
Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link



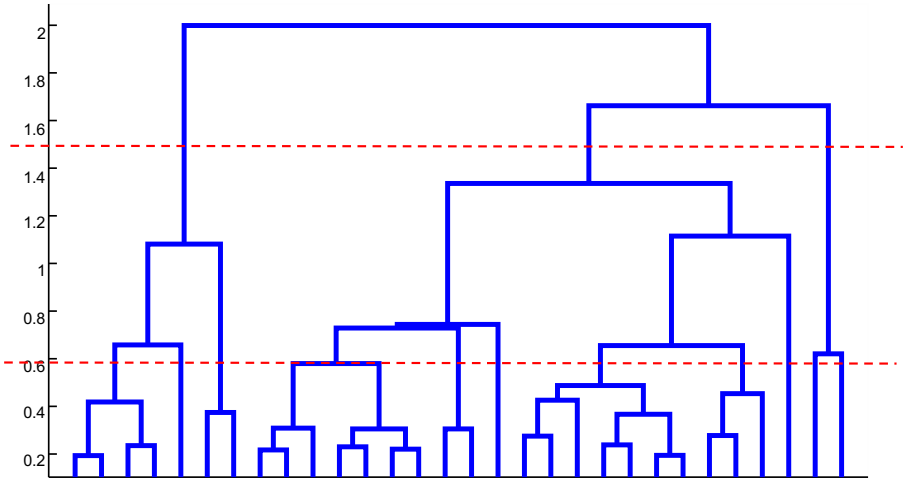
single link



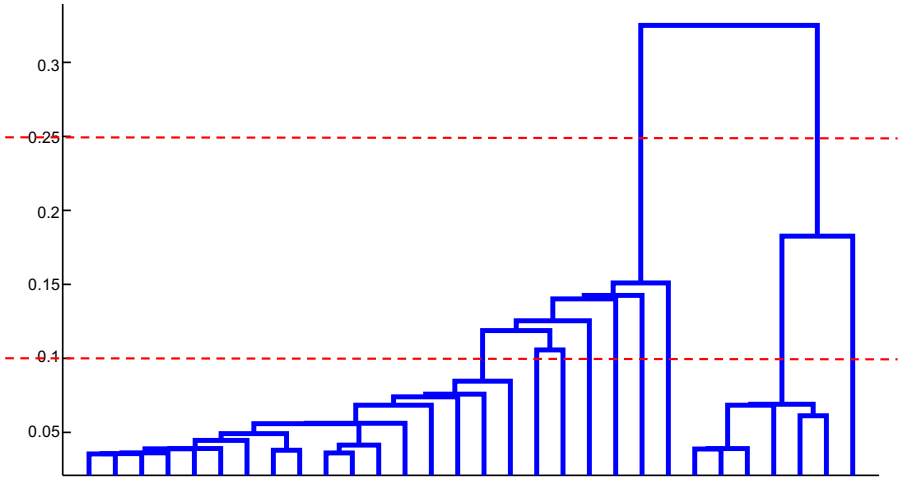
Each discrete value alone the x-axis is treated as one data instance.

Agglomerative Clustering

complete link



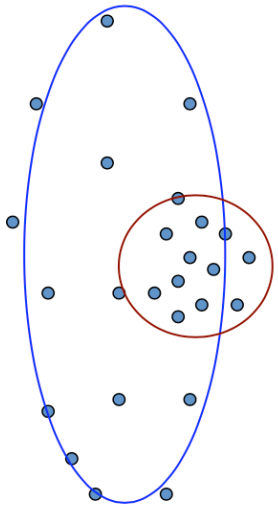
single link



Each discrete value alone the x-axis is treated as one data instance.

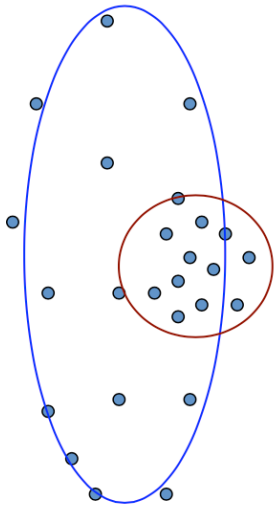
(One) bad case for “hard assignments”?

- Clusters may overlap
- Some clusters may be “wider” than others



(One) bad case for “hard assignments”?

- Clusters may overlap
- Some clusters may be “wider” than others
- *Distances can be deceiving!*

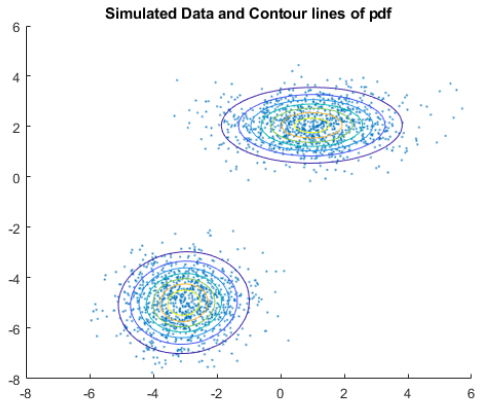


Probabilistic clustering / Generative models

THE OHIO STATE UNIVERSITY

Probabilistic Clustering

- We can use a probabilistic model!
  - Change from distance to likelihood: p(x|Y=0)p(Y=0) vs. p(x|Y=1)p(Y=1)
  - Allows overlaps, clusters of different size, etc.



- Can tell a *generative story* for data
  - P(X|Y)P(Y) is common

- Challenge:
  - Like K-means, we don't know y_i for each $\boldsymbol{x}_i$
  - We need to estimate model parameters without the labels

What model to use

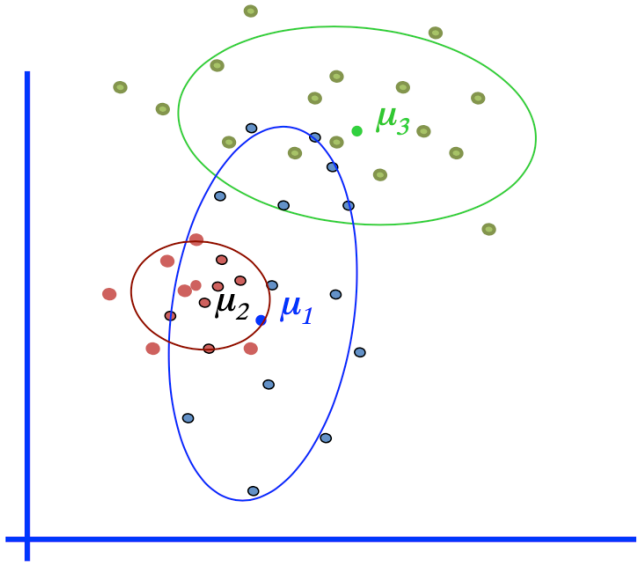
- Depends on X!
- Gaussian Naïve Bayes
  - Multinomial over cluster Y,
  - Gaussian over each X_i given Y

$$p(Y_i = y_k) = \theta_k$$

$$P(X_i = x \mid Y = y_k) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} \quad e^{\frac{-(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

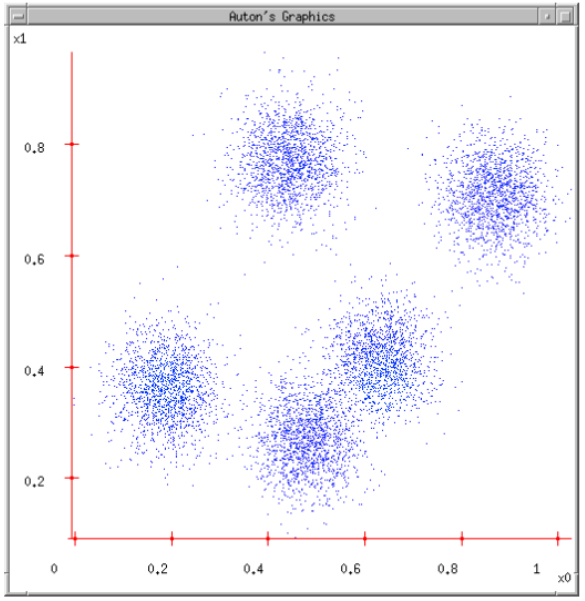
General GMM assumption

- P(Y): There are *K* components (classes)
- P(X|Y): Each component generates data from a Gaussian with mean μ_i and covariance matrix Σ_i
- Each data point is sampled from a *generative process*:
  1. Pick a component at random:
     Choose component *i* with probability *P(y=i)*
  2. Datapoint ~N(μ_i, Σ_i)



How to estimate parameter

- MLE:
  - $\text{argmax}_\theta \prod_j P(y^j, x^j; \theta)$
  - θ: all model parameters
    - e.g., class probability, means, and variance for naïve Bayes
- But we don t know y_j!!!
- Maximize *marginal likelihood*:
  - $\text{argmax}_\theta \prod_j P(x^j; \theta) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y^j = i, x^j; \theta)$



How do we optimize? Closed Form?

- Maximize *marginal likelihood*:
  - $\text{argmax}_\theta \prod_j P(x^j;\theta) = \text{argmax} \prod_j \sum_{i=1}^{k} P(y^j=i,x^j;\theta)$
- Almost always a hard problem!
  - Usually no closed form solution
  - Even when P(X, Y ; θ) is convex, P(X;θ) generally is not
  - For all but the simplest P(X;θ),
    - we will have to do gradient ascent, in a big messy space with lots of local optimum…