syde675 pattern recognition

Naïve Bayes

J.Zelek 2022

Recall: Bayes Theorem & Statistical Pattern Recognition

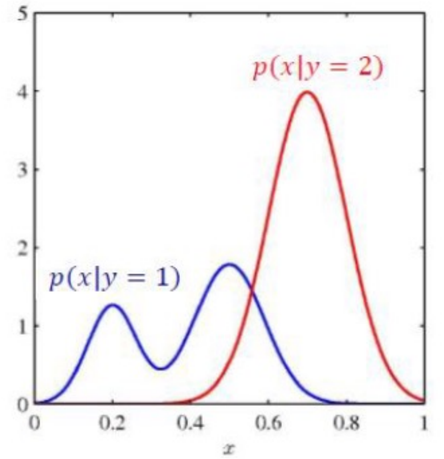
- Each term in the **Bayes Theorem** has a special name, which **you should be familiar with.**

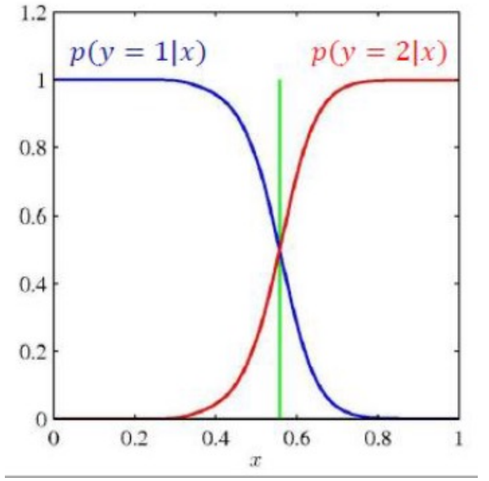
$$P(c_k|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|c_k)P(c_k)}{\sum_{i=1}^{N_C} P(\boldsymbol{x}|c_k)P(c_k)} = \frac{P(\boldsymbol{x}|c_k)P(c_k)}{p(\boldsymbol{x})}$$

- $\boldsymbol{P(c_k)}$: **Prior probability** (Probability of class c_k)
- $\boldsymbol{P(\boldsymbol{x}|c_k)}$: **Likelihood** (Conditional probability of observation $\boldsymbol{x}$ given class c_k)
- $\boldsymbol{P(c_k|\boldsymbol{x})}$: **Posterior Probability** (Probability of class c_k given the observation $\boldsymbol{x}$)
- $p(\boldsymbol{x})$: A **normalization factor** which is constant and doesn't affect the decision.

Generative vs. Discriminative Aproaches

- **Generative approaches**: In these approaches, first a joint model $\boldsymbol{P(x;\ C_k)}$ is created and then $P(C_k\ |\boldsymbol{x})$ is derived.
- Approaches that explicitly or implicitly model the distribution of inputs as well as outputs, here $P(\boldsymbol{x};\ C_k)$, are known as generative models [Bishop]
  - because by sampling from $P(\boldsymbol{x};\ C_k)$ it is possible to generate synthetic data points in the input space
- **Discriminative approach**: These approaches directly create a model of the form of $\boldsymbol{P(C_k|x)}$.
- Approaches that model the posterior probabilities directly are called discriminative models





note

Bayes is optimal, so why do we even think about other classifiers?

Bayes Approach Issue

- With many attributes, it is computationally expensive to compute $P(\boldsymbol{x}|c_k)$

$$P(\boldsymbol{x}|c_k) = P(x_1, x_2, \ldots, x_n|c_k)$$

- # of parameters for modeling $P(x_1, x_2, \ldots, x_n|c_k)$:
  - Let's say 2 class labels and each feature is binary
    - $2(2^n-1)$

Example

- Dataset
- We want to classify a Red Domestic SUV
- No example of a Red Domestic SUV in our data

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

http://www.inf.u-szeged.hu/~ormandi/ai2/06-naiveBayes-example.pdf

Naïve Bayes

- Conditional Independence Assumption:
- All features are independent **given the class label** c_k

$$P(x_1, x_2, \ldots, x_n | c_k) = P(x_1|c_k) \times P(x_2|c_k) \times \ldots \times P(x_n|c_k)$$

- # of parameters for modeling $P(x_1|c_k) \times P(x_2|c_k) \times \ldots \times P(x_n|c_k)$:
  - 2n

Naïve Bayes

- Training in Naïve Bayes is **easy**:
  - Estimate $P(C = c_k)$ as the fraction of records with $C = c_k$

$$P(C = c_k) = \frac{count(C = c_k)}{\neq record}$$

  - Estimate $P(x_i = u | C = c_k)$ as the fraction of records with $C = c_k$ for which $x_i = u$

$$P(x_i = u | C = c_k) = \frac{count(x_i = u,\ C = c_k)}{\neq record(C = c_k)}$$

- This corresponds to Maximum Likelihood estimation of model parameters
- In case of limited training data it is better to use more sophisticated m-estimator

Log Sum-EXP Trick

- If one of the $P(x_k|c_k)$ is very small and close to zero:

$$P(x_1|c_k) \times P(x_2|c_k) \times \ \ldots \times P(x_n|c_k) \approx 0$$

- But the goal is to find best class

$$c_{map} = argmax\ P(\boldsymbol{x}|c_k)P(c_k)$$

- Then we can use $\log(\quad)$ to normalize the values

$$c_{map} = argmax\ [\log P(c_k) + \sum_{1 \le k \le n} \log P(x_k|c_k)]$$

Example

- Dataset:

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

http://www.inf.u-szeged.hu/~ormandi/ai2/06-naiveBayes-example.pdf

Example

- We want to classify a Red Domestic SUV
- No example of a Red Domestic SUV in our data
- Naïve Bayes (conditional independence assumption!):

$$c_{map} = argmax\ P(\boldsymbol{x}|c_k)P(c_k)$$
$$= argmax\ P(x_1|c_k) \times P(x_2|c_k) \times \ \ldots \times P(x_n|c_k)P(c_k)$$
$$= argmax\ P(c_k) \prod_i P(x_i|c_k)$$

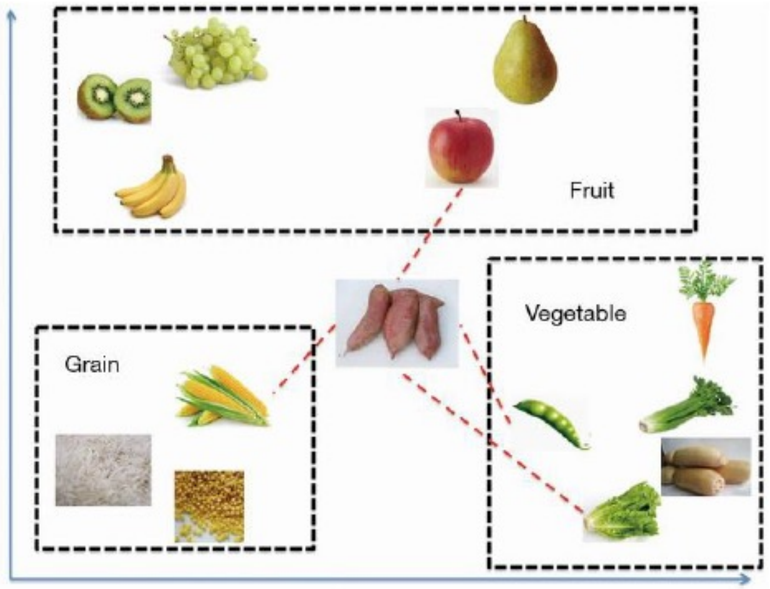
- P(Red|Yes), P(SUV|Yes), P(Domestic|Yes)
- P(Red|No) , P(SUV|No), and P(Domestic|No)

Example

- P(Red|Yes) = 3/5
- P(SUV|Yes) =1/5
- P(Domestic|Yes) = 2/5
- P(Red|No) = 2/5
- P(SUV|No) = 3/5
- P(Domestic|No) = 3/5
- P(Yes) = 5/10
- P(No) = 5/10

- P(Red,SUV,Domestic|Yes) = 0.5x(0.6x0.2x0.4)
- P(Red,SUV,Domestic|No) = 0.5x(0.4x0.6x0.6)

Instance based Learning



Instance-based Learning

Instance based Learning

- Instance-based Learning other names:
  - Memory-based learning
  - Exemplar-based
  - Case-based
  - Lazy learning

- Lazy learning:
  - There is no training step.
  - All training examples (x_i, y_i) are stored instead of explicit representation of target function.
  - All the work is done at the test step when a new sample must be classified .

Lazy vs. Eager Learning

- Eager learning:
  - Build a model based on training data, e.g. SVM, logist reg., Neural Net., Decision tree
  - Discard training data after constructing the model
  - Classify new samples based on the learned model

- Trade-offs:
  - Computational cost of eager algorithms is higher than lazy algorithms during training
  - Storage requirements and computational cost of lazy algorithms are higher than eager learners during test.

k-Nearest-Neighbor (kNN)

- KNN classifies a new example x based on its similarity to training examples

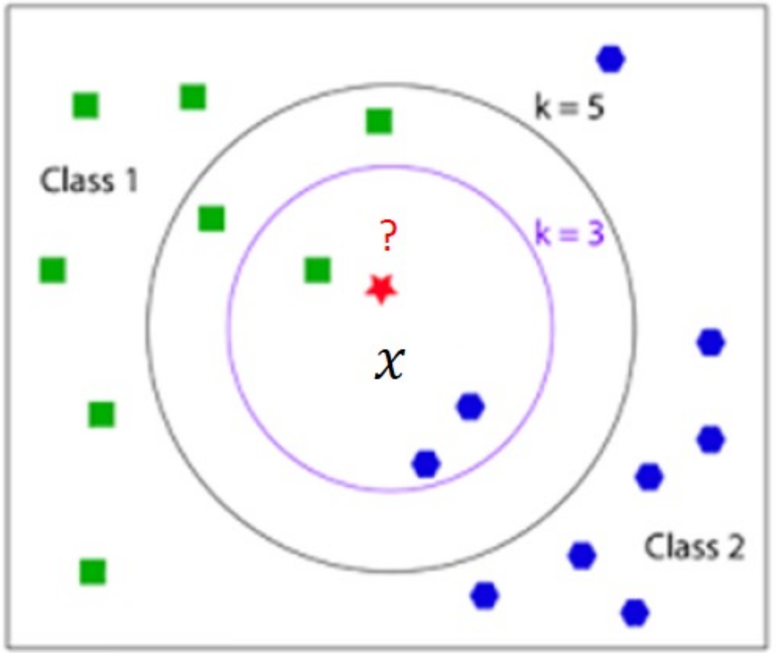
- KNN finds k nearest neighbors of the unlabeled example x and assign the class label via majority voting of the label of the k nearest neighbors

- KNN only requires:
  - The value of K
  - A set of labeled examples (training data)
  - A similarity metric that measures closeness

KNN Example 1

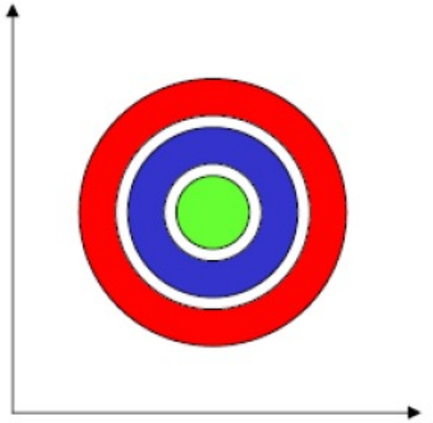
- Suppose:
  - A 2D three classes problem
  - We use a Euclidean distance to measure similarity

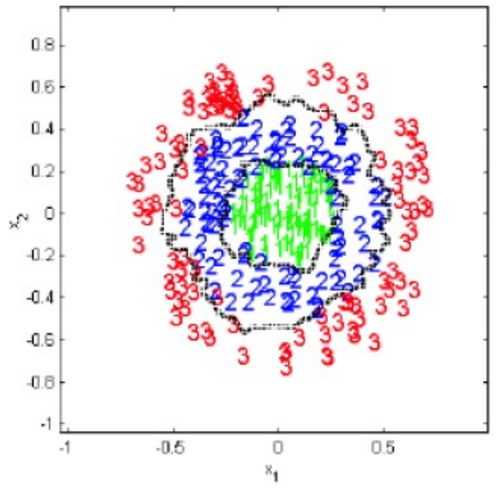
- What is the class label of the new sample x ?

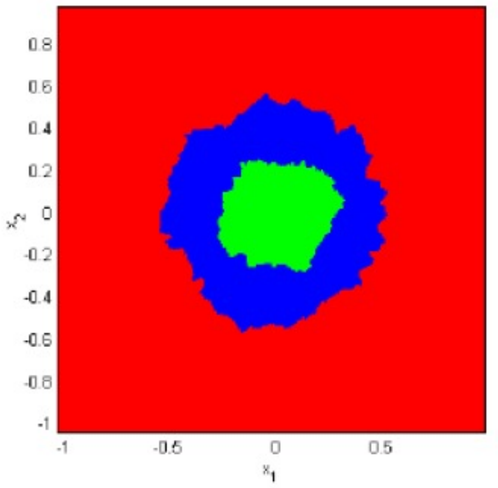


KNN Example 2

- Suppose:
  - A 2D three classes problem
  - We use a Euclidean distance to measure similarity
  - K=5
- The resulting decision boundaries and decision regions are:







Samples from [R. Gutierrez-Osuna]

KNN Algorithm

- Input: unlabeled sample x, and

  training set $D = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})\}$, The value of k

1. KNNs = Find k nearest neighbors to x in D
2. Among the KNNs, identify the #samples k_j that belong to each class C_j $(\forall j = 1 \ldots m)$
3. Assign x to the class C_{j^*} where $j^* = argmax(k_j)$

- Output: class label of x

KNN Probabilistic Perspective

- It can be said that the KNN classifier approximates the posterior:

$$P(C_j|x) \approx \frac{k_j}{K}$$

  - where k_j shows the number of examples belong to class C_j among K nearest neighbors

- Therefore, KNN approximate a Bayesian decision rule for assigning label

KNN: Error Bound

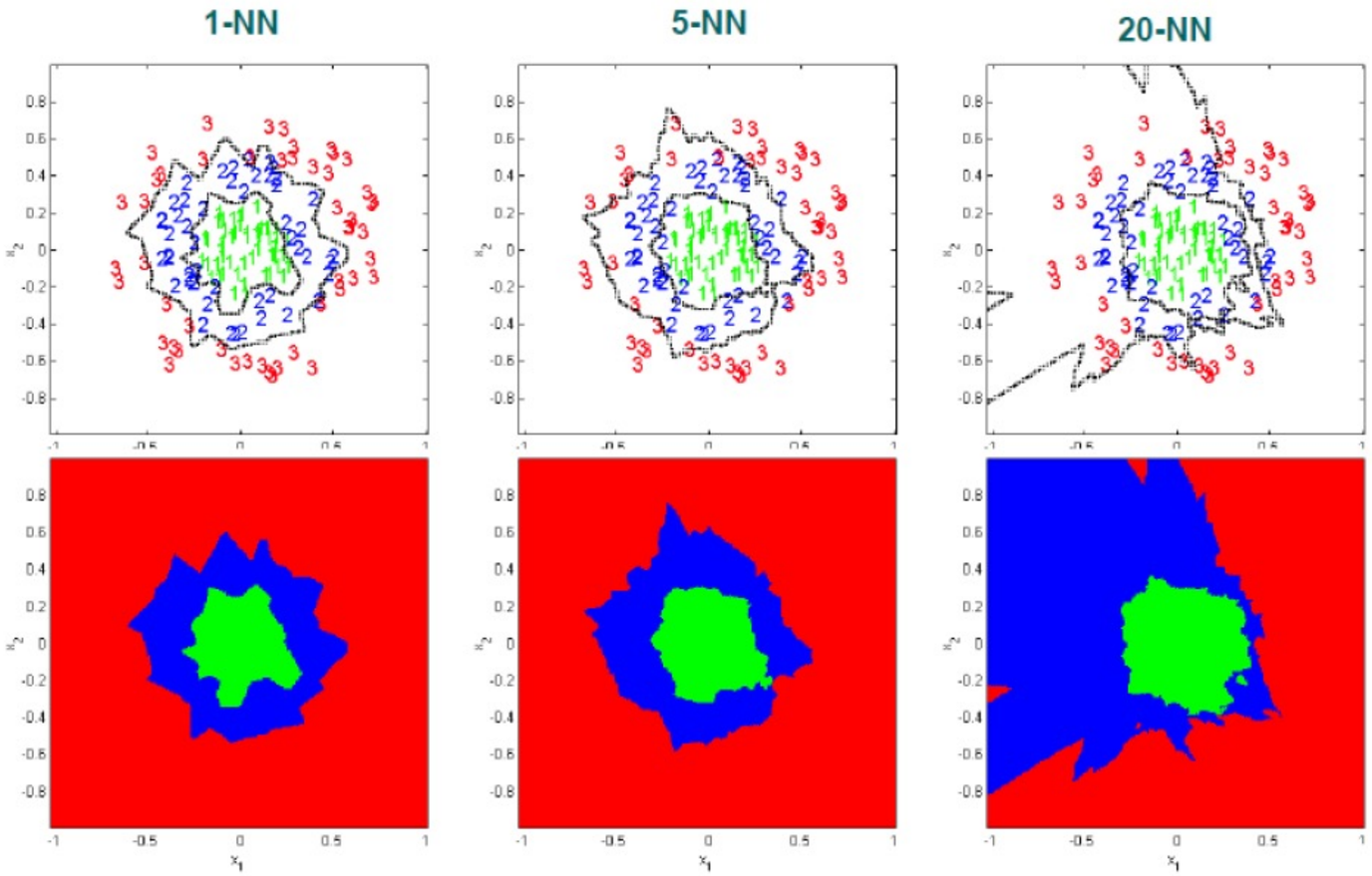
- For 1NN classifier, it can be shown that [Cover&Hart67]:
  - R^* = The optimal Bayes risk, i.e. $P(error)_{Bayes}$

$$R^* \leq R^{1NN} \leq 2R^*(1 - R^*) \leq 2R^*$$

- The error of 1NN is less than $2R^*$ (2*the optimal Bayes risk)

The effect of K in KNN

- Large value of K:
  - Advantages:
    - Provides smoother decision boundaries
    - Provides more robust probabilistic information
  - Disadvantages:
    - Misses the locality of the estimation
    - Increases the computational cost
- Commonly k={1,3,5} is suitable



[R. Gutierrez-Osuna]

Reducing Storage Requirements of KNN

- Basic KNN stores all training samples leads to high storage and computational cost
- Almost all of the information for classification purposes is located around the decision boundaries
- Some approaches to reduce the storage requirements:
  - Classify all training examples and remove samples that are classified correctly
    - Attempt to define the boundaries by eliminating points in the interior of the regions
  - Reducing training samples by prototype selection
    - It is performed using clustering methods

KNN Properties

- Advantages:
  - Simple implementation
  - Easy to be implemented in parallel
  - Nearly optimal in the large samples
    - The error of 1NN is less than 2 (2*the optimal Bayes risk)
  - Use local information

- Disadvantages:
  - Lazy algorithm
  - Large storage requirements
  - High computational cost
  - Prone to the curse of dimensionality problem
  - Sensitive to noise if features were in different scale

- Note: make sure the feature are on similar scale to avoid scaling problem

Weighted KNN Classification

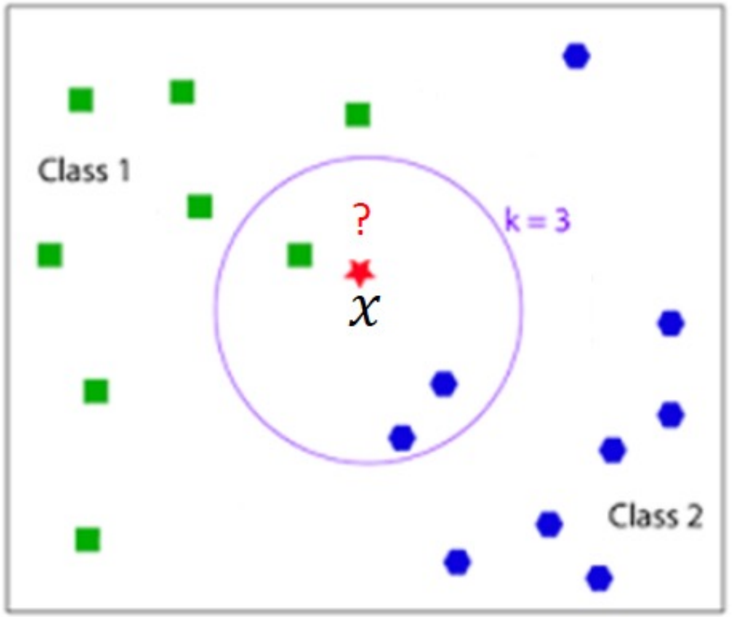
Weighted KNN Classification

- Weighted KNN assigns a weight to the samples according to their distances to the test sample.
- Nearer neighbors are taken more weights:

$$\hat{y} = f(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\operatorname{argmax}} \sum w_j(\boldsymbol{x}) \times I(c = y^{(j)})$$

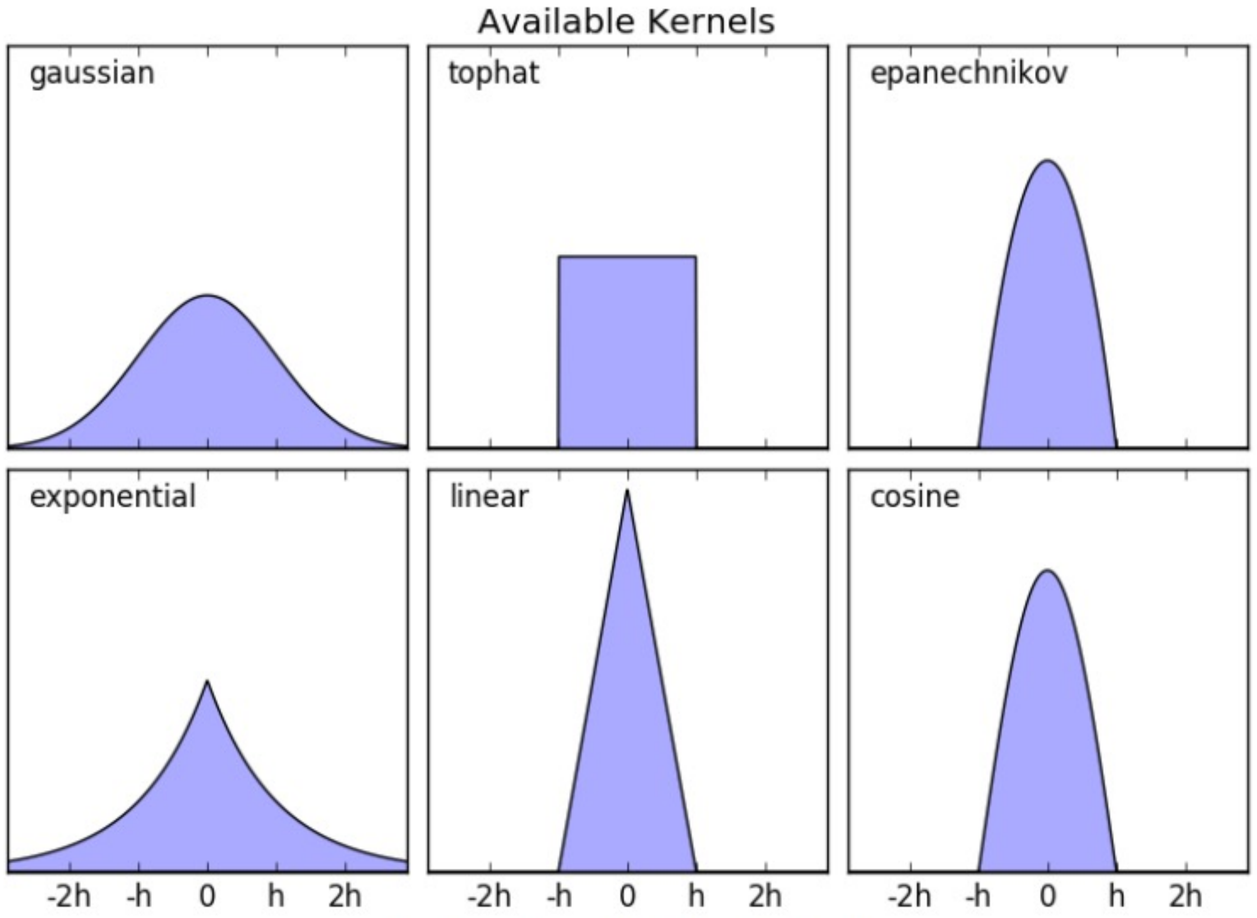
$$w_j(\boldsymbol{x}) = \frac{1}{\|\boldsymbol{x} - \boldsymbol{x}^{(j)}\|^2}$$

An example of weighting function



- In weighted KNN, all samples can be used instead of K nearest neighbor

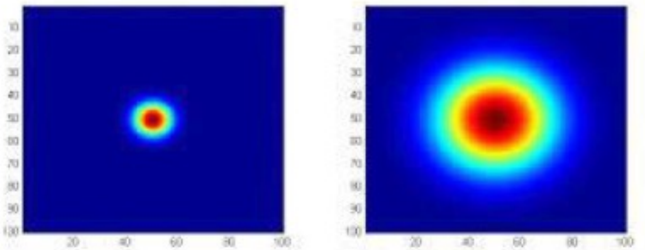
Weighted KNN Classification



[Figure by Jiancheng Li]

- Weights can be found using a kernel function $w_i = K(x, x^{(i)})$
  - For example: Gaussian kernel

$$K\left(x, x^{(i)}\right) = \exp\left(-\frac{dist\left(x, x^{(i)}\right)}{\sigma^2}\right)$$



small σ *large* σ

Distance Metrics

- Minkowski metric (L_k norm)

$$\|x - y\|_k = \left(\sum_{i=1}^{D} |x_i - y_i|^k \right)^{1/k}$$

- Manhattan or city-block distance (L_1 norm)

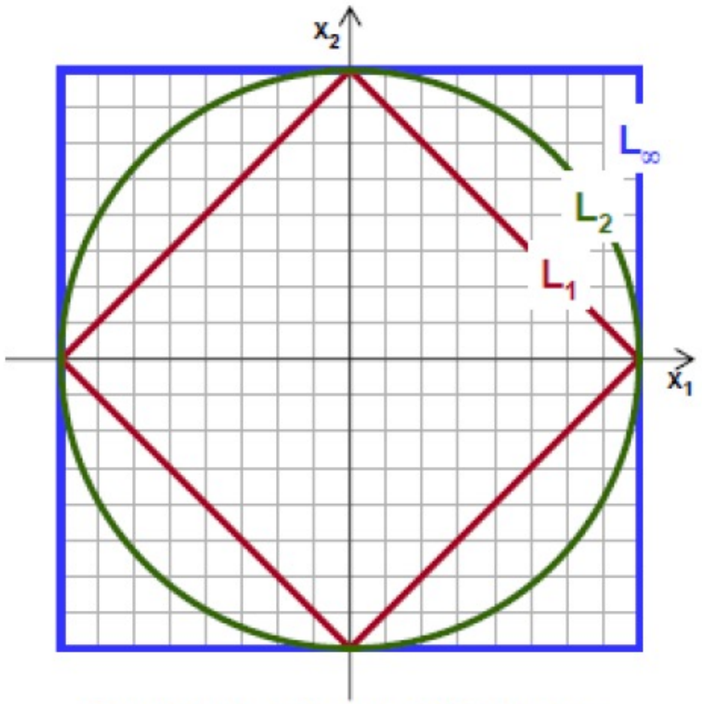
$$\|x - y\|_{c\text{-}b} = \sum_{k=1}^{D} |x_k - y_k|$$

- Euclidean norm (L_2 norm)

$$\|x - y\|_e = \left(\sum_{k=1}^{D} |x_k - y_k|^2 \right)^{1/2}$$

- Chebyshev distance (L_∞ norm)

$$\|x - y\|_c = \max_{1 \le i \le D} |x_i - y_i|$$



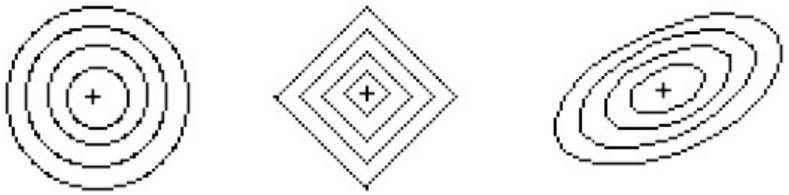
Contours of equal distance

Distance Metrics

- Mahalanobis distance
  - P is the covariance matrix $cov(x, y)$

$$d_P(x, y) = \sqrt{(x - y)^T P^{-1} (x - y)}$$

  - Mahalanobis distance considers the distribution of the data along each
  - Matrix P has a role of normalization Factor
    - In Euclidean distance the matrix P is an identity matrix, i.e., equal weigh to each dimension



Supervised vs. non supervised clustering

knn is supervised

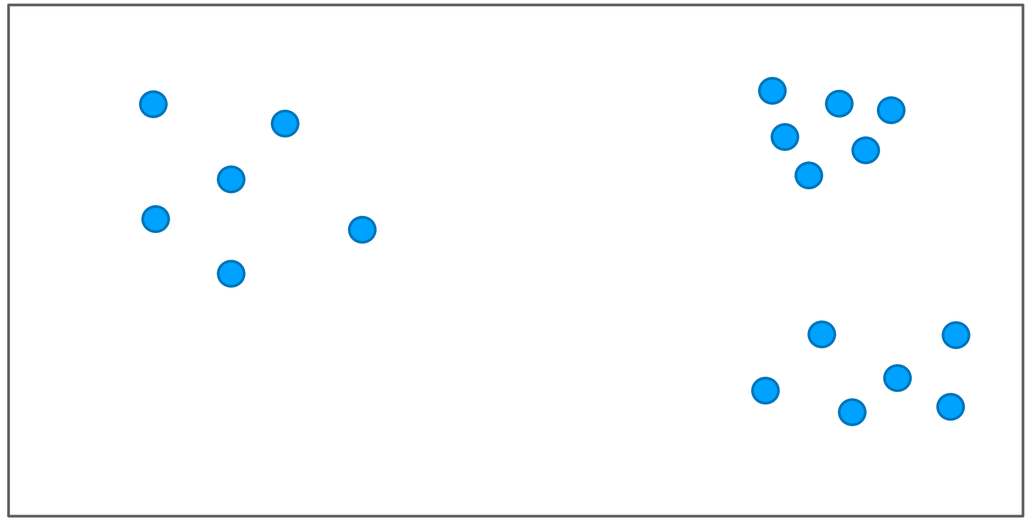
Outline

- Background

- Clustering algorithms
  - K-mean
  - EM Clustering
  - Hierarchy

Clustering

- Unsupervised Learning
- No Ground Truth
- Investigate data structure by grouping them into distinct groups.
- Advantage: useful when don’t know what to look for
- Disadvantage: Subjective



Applications

- Find set of representative that provide coverage of the data
  - Identify fake news based on the content
  - Group emails

- Exploratory Insight
  - Customer shopping patterns
  - Groups of genes/proteins with similar function

Clustering

Basic idea: group similar instances together

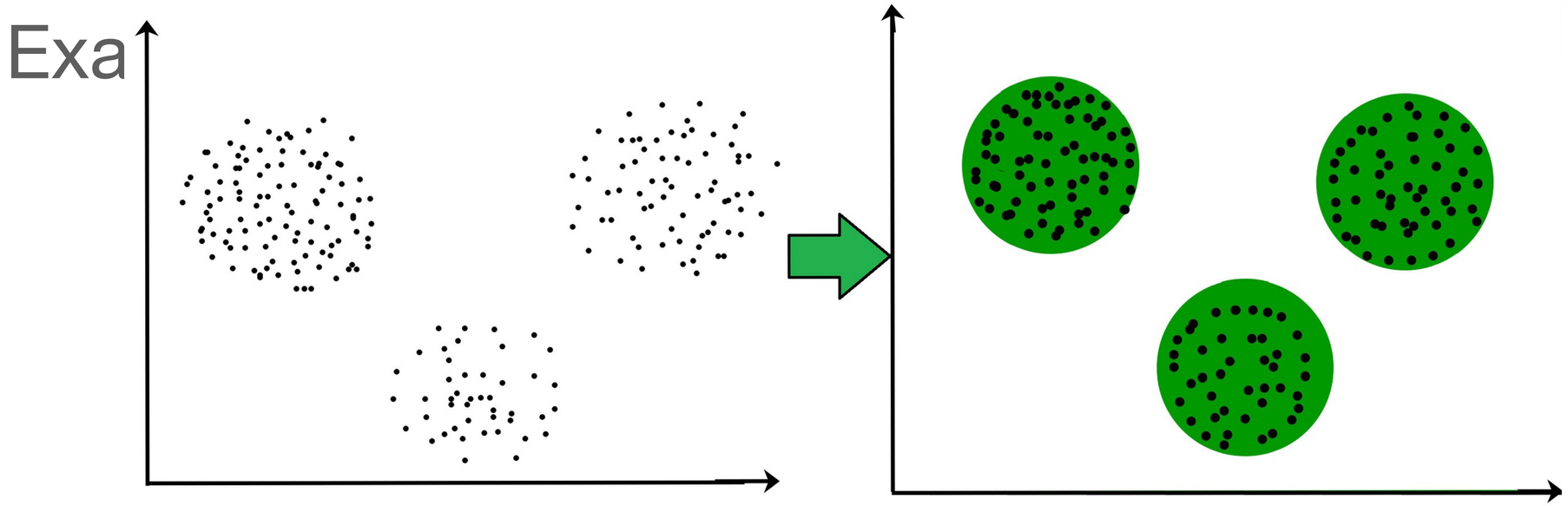


Image taken from https://www.geeksforgeeks.org/clustering-in-machine-learning/

Clustering

Basic idea: group similar instances together

Exa

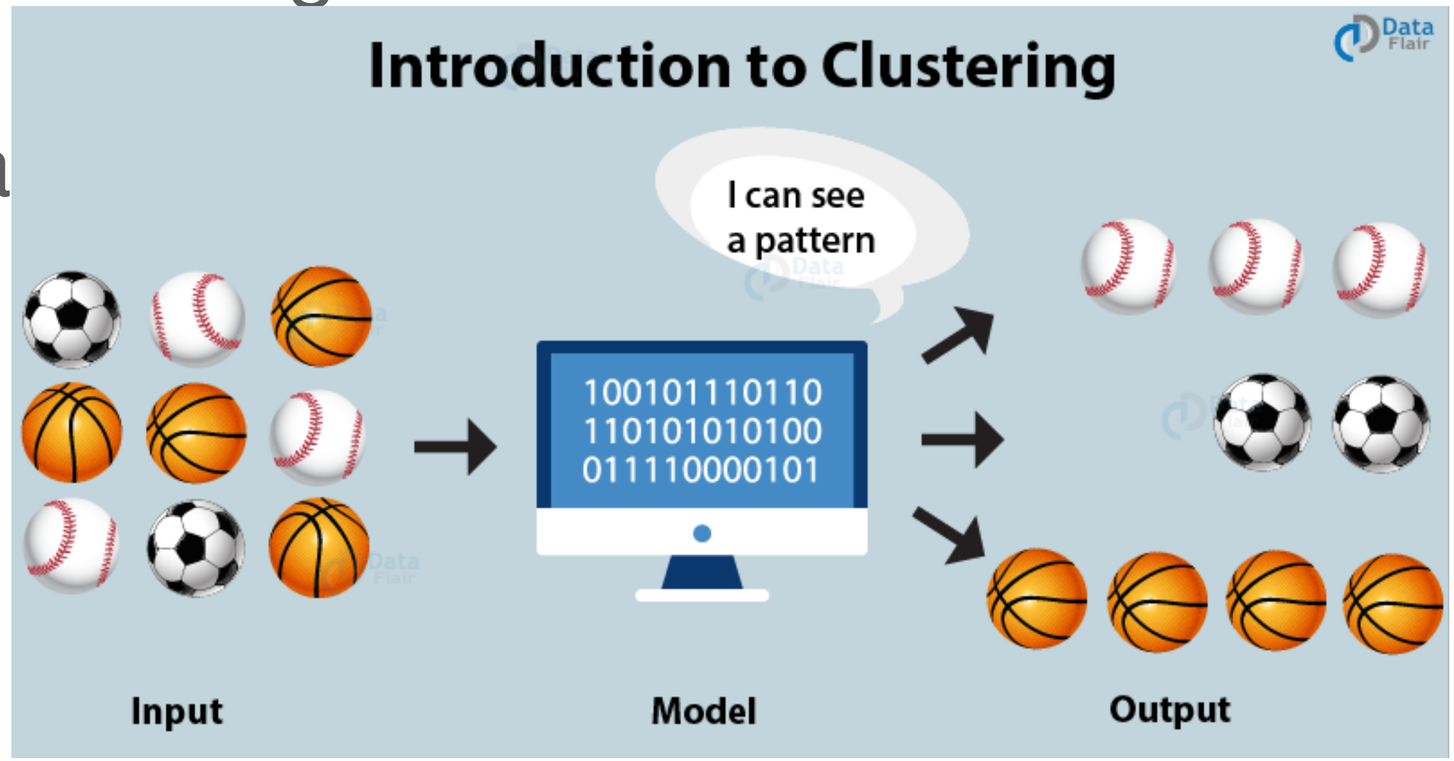
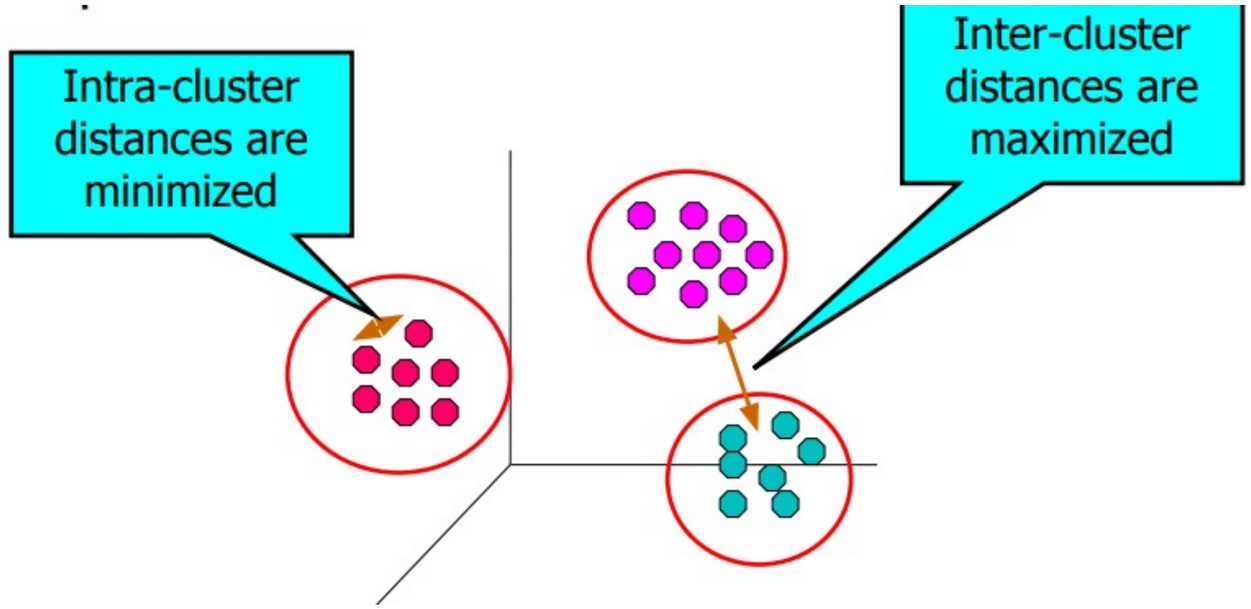


Image taken from https://data-flair.training/blogs/clustering-in-machine-learning/

Clustering

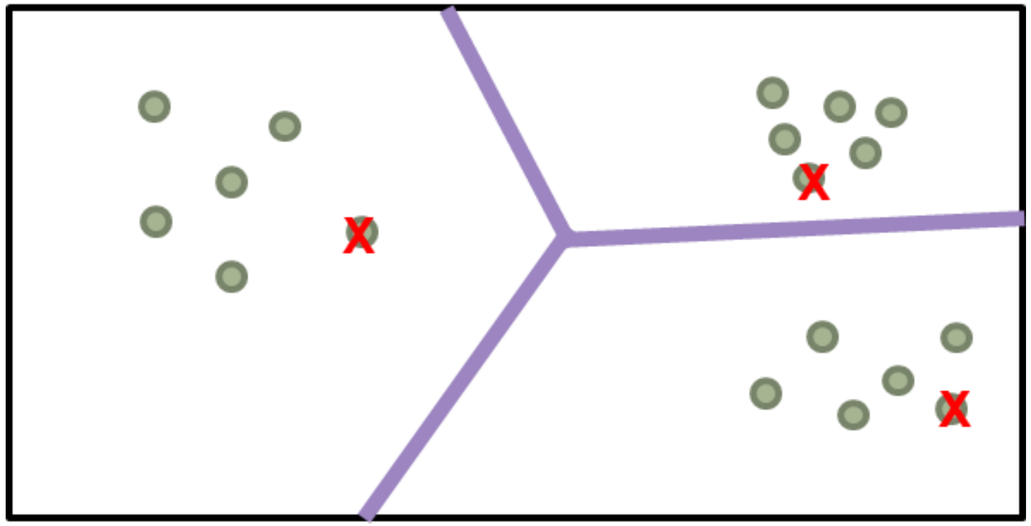
- Basic idea: group similar instances together
  - What does "similar" mean?
    - Based on Euclidean



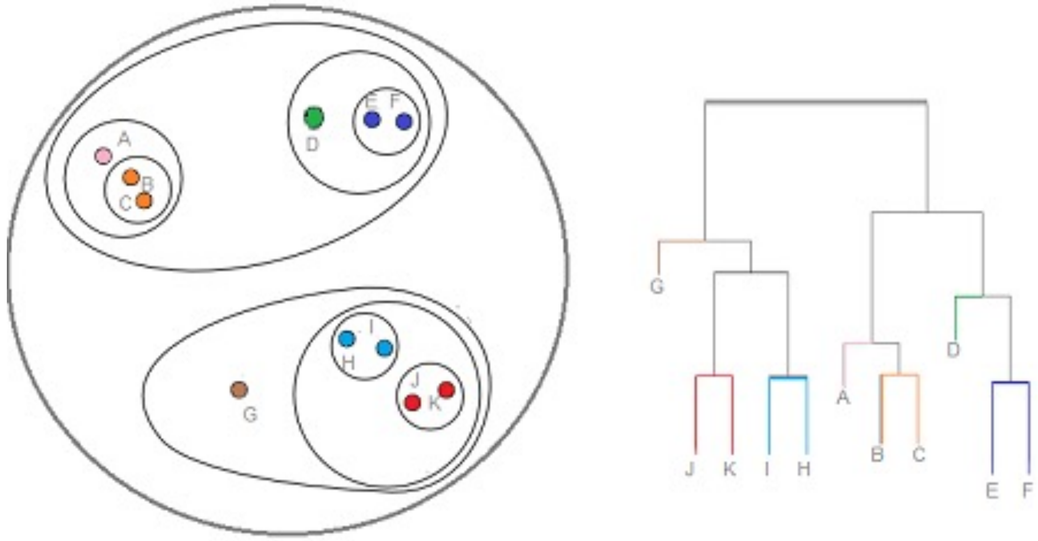
Clustering

- Partitional - data in non-overlapping subsets. One data object is in one subset
- Hierarchy - data are in nested clusters, organized in a hierarchical tree

Partitional



Hierarchy



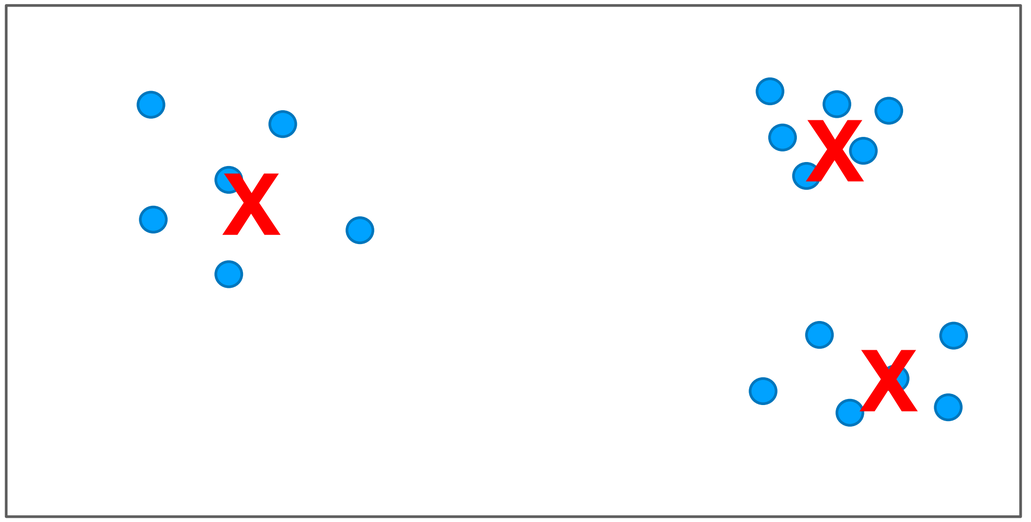
Partitional Clustering - Kmean

Also known as Lloyd's algorithm

Input: $x_1, x_2, \ldots, x_n$, given $x \in R^d$

Output: 'Centers', $\mu_1, \mu_2, \ldots, \mu_k$, given $\mu \in R^d$

Goal: Minimize average square distance between point and the closest 'center'

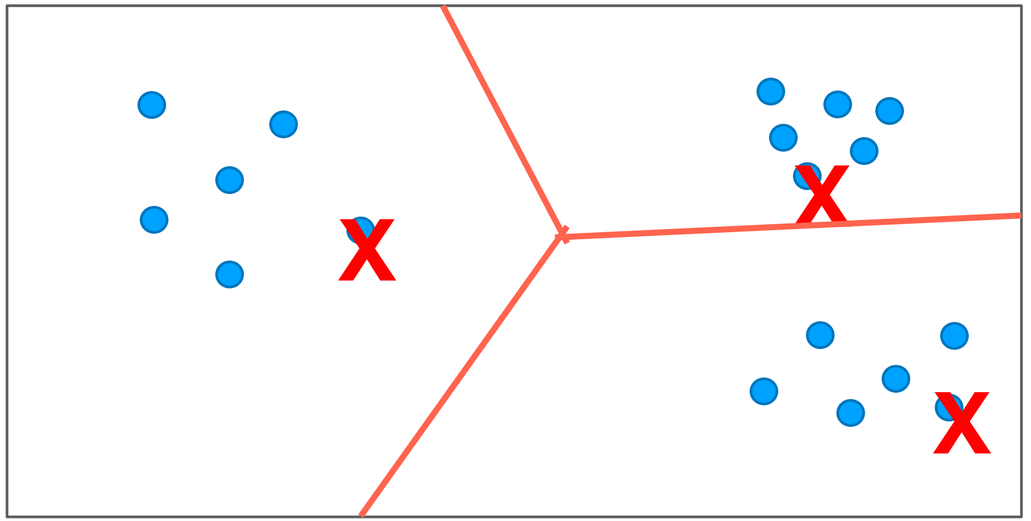


$$cost(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \min_j \left| x_i - \mu_j \right|^2$$

Euclidean distance

K-mean

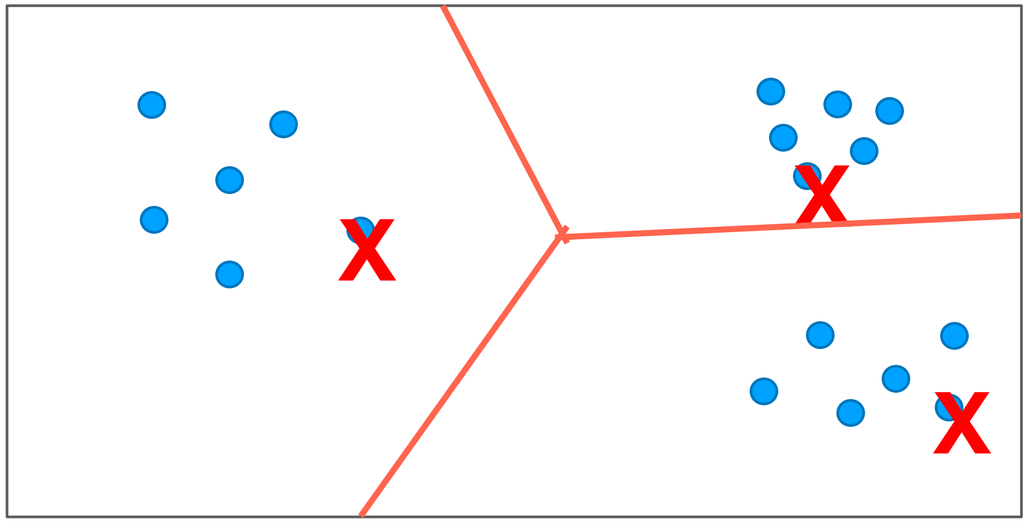
Say we want three clusters, first initialize random center



Generate optimal partition

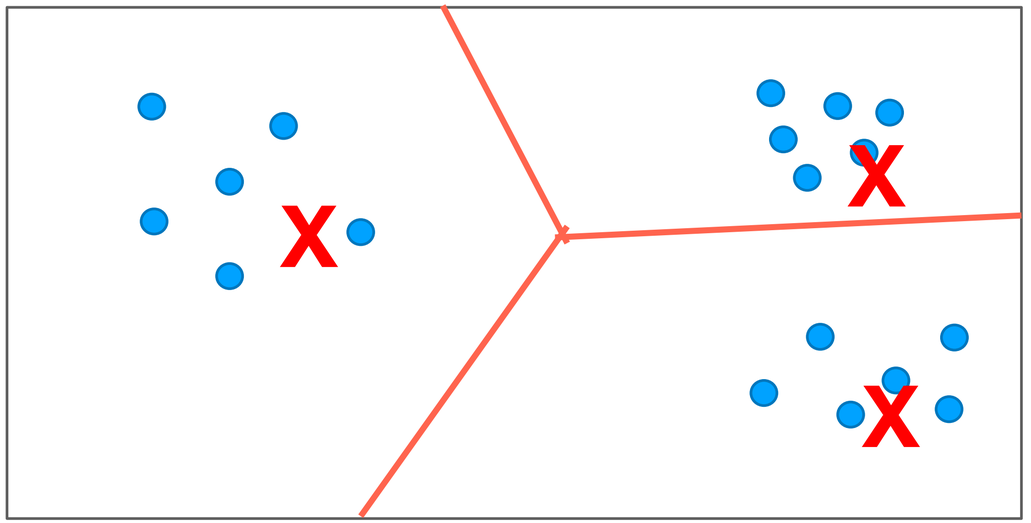
Compute Euclidean distance between center and data points

K-mean



Update the center by computing the mean of coordinate in their respective region.

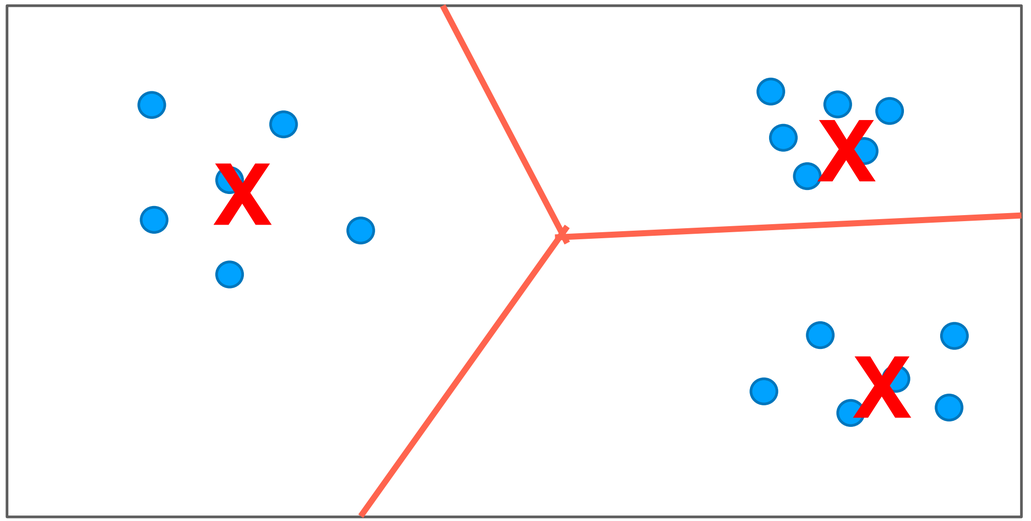
K-mean



Updated center

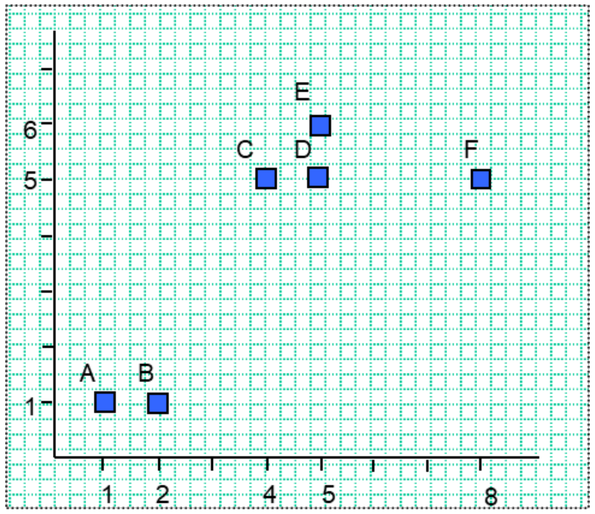
Compute Euclidean distance between points to the closest centers.

K-mean



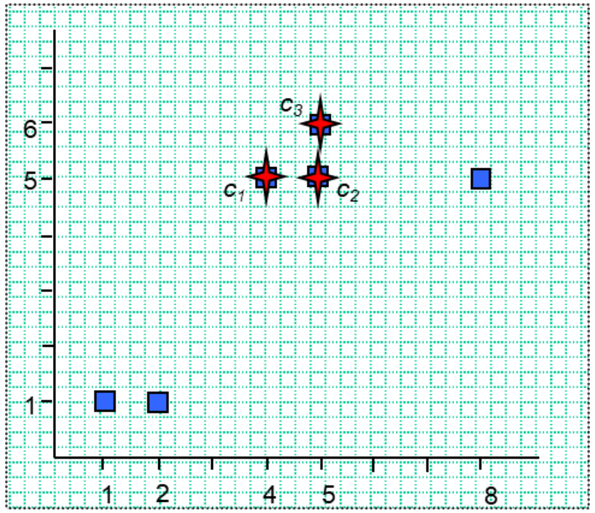
Center no longer change (converge). Now, we have identify the three clusters.

K-mean



Data set

X: a set of *N* data vectors
N = 6

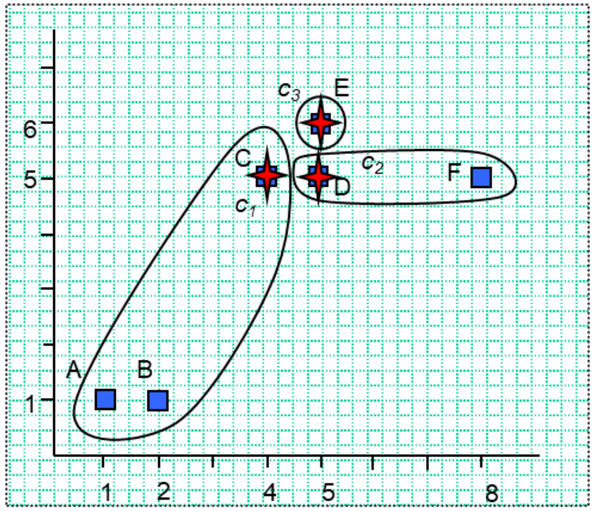


Number of clusters
Random initial centroids

Initial codebook:
$c_1 = C$, $c_2 = D$, $c_3 = E$

C_I: initialized *k* cluster centroids
k = 3

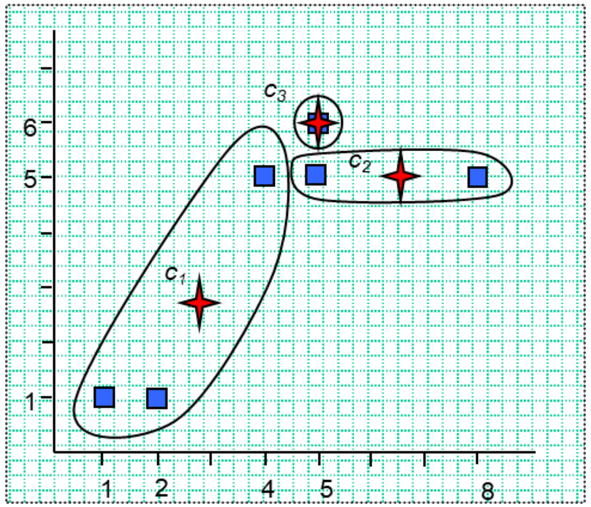
K-mean



Generate optimal partitions

Distance matrix (Euclidean distance)

	A	B	C	D	E	F
c_1	***5***	***4,5***	***0***	1	1,4	4
c_2	5,7	5	1	***0***	1	***3***
c_3	6,4	5,8	1,4	1	***0***	3,2



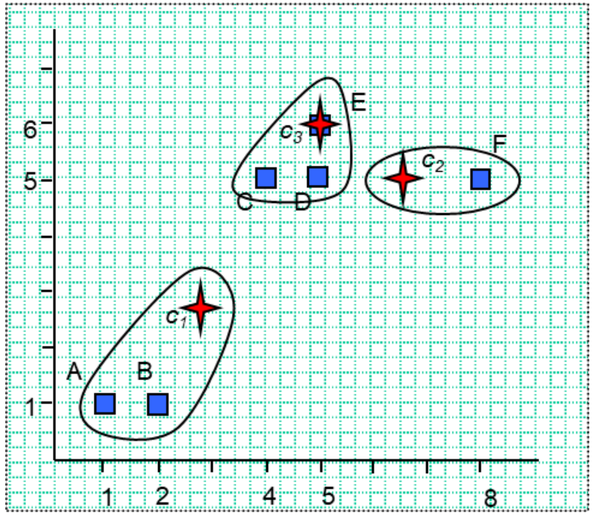
Generate optimal centroids.

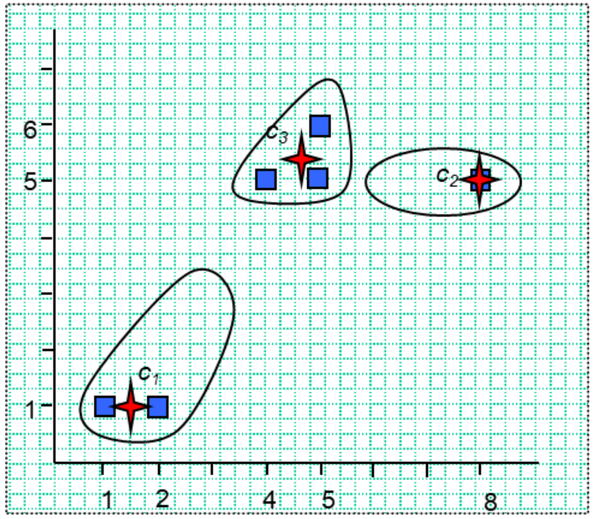
$$c_1 = \left(\frac{1+2+4}{3}, \frac{1+1+5}{3}\right) = (2.3, 2.3)$$

$$c_2 = \left(\frac{5+8}{2}, \frac{5+5}{2}\right) = (6.5, 5)$$

$$c_3 = (5, 6)$$

K-mean





Generate optimal partitions

Distance matrix (Euclidean distance)

	A	B	C	D	E	F
c_1	***1,9***	***1,4***	3,1	3,8	4,5	6,3
c_2	6,8	6	2,5	1,5	1,8	***1,5***
c_3	6,4	5,8	***1,4***	***1***	***0***	3,2

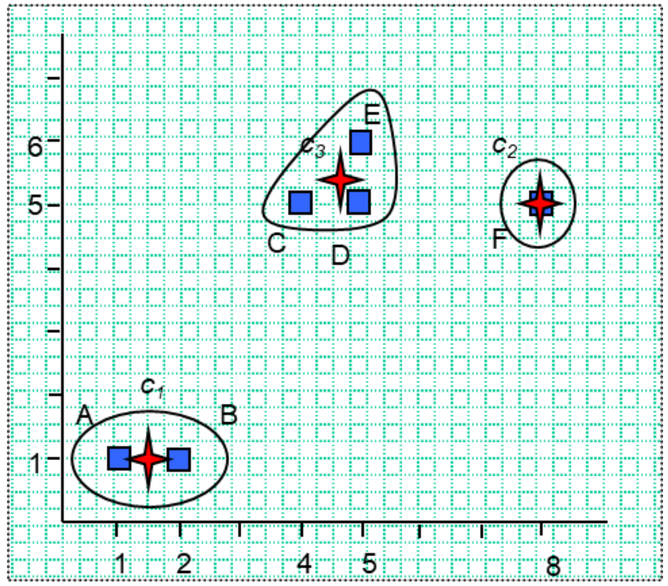
Generate optimal centroids

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = (1.5,1)$$

$$c_2 = (8,5)$$

$$c_3 = \left(\frac{4+5+5}{3}, \frac{5+5+6}{3}\right) = (4.7,5.3)$$

K-mean



Generate optimal partitions

Distance matrix (Euclidean distance)

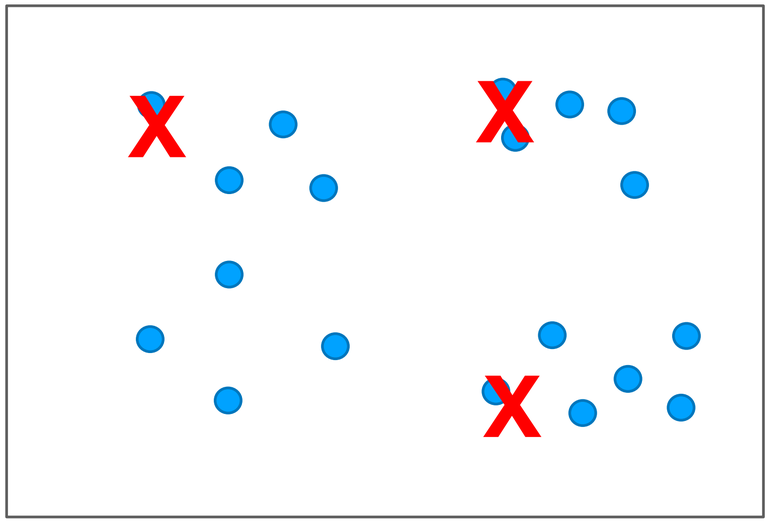
	A	B	C	D	E	F
c_1	***0,5***	***0,5***	4,7	5,3	6,1	7,6
c_2	8,1	7,2	4	3	3,2	***0***
c_3	5,7	5,1	***0,7***	***0,5***	***0,7***	3,3

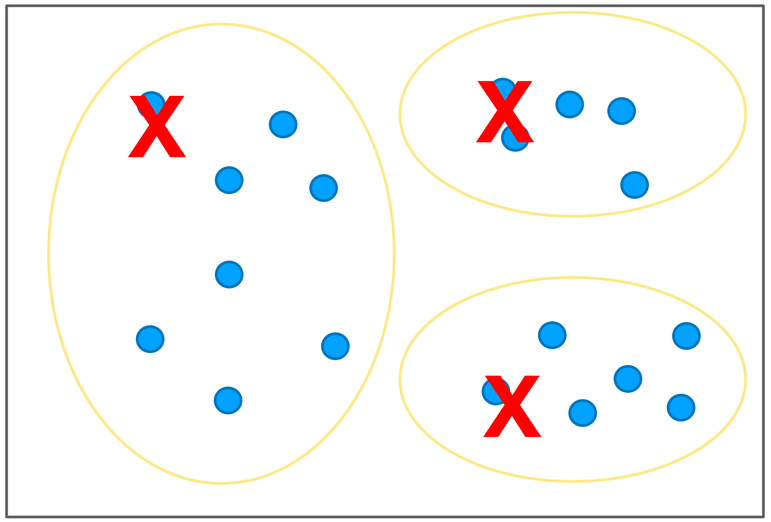
After 3rd iteration: MSE = 0.31

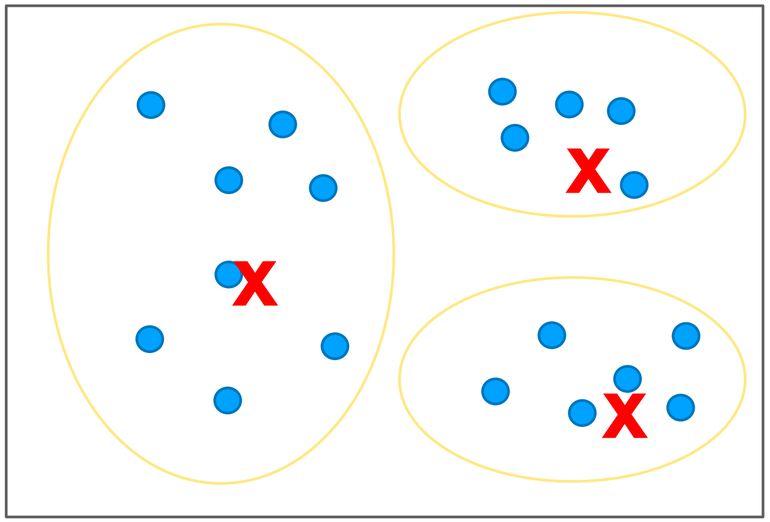
Converge. Centroid no longer change.

K-mean

What if we initialize in different way. Example 1

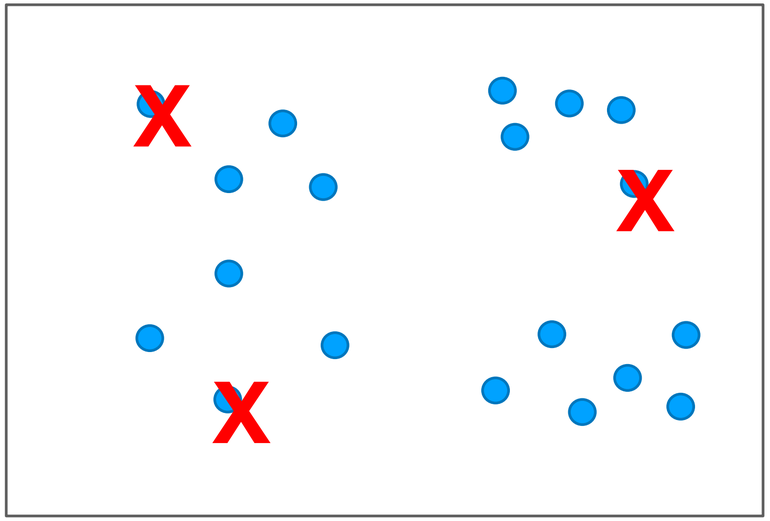


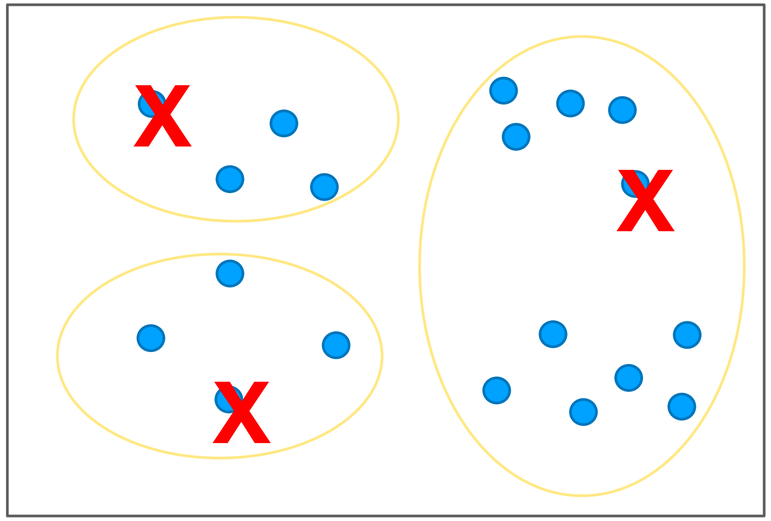


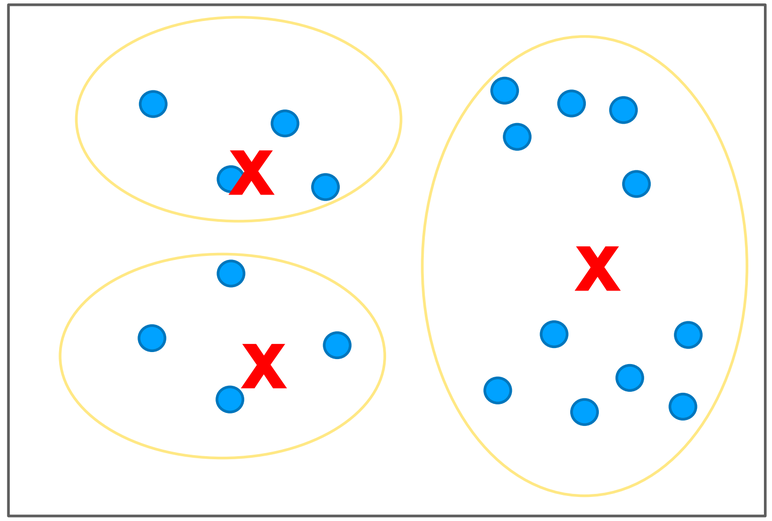


K-mean

What if we initialize in different way. Example 2



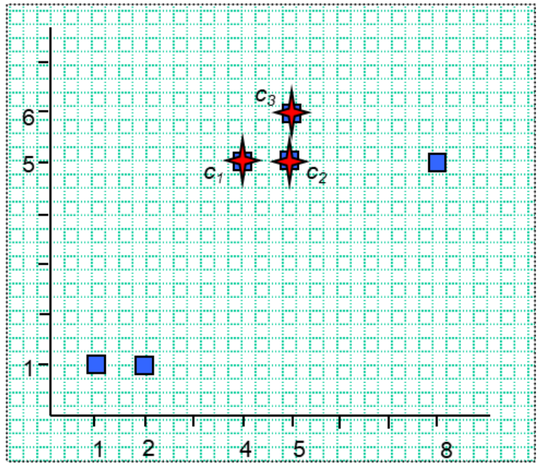


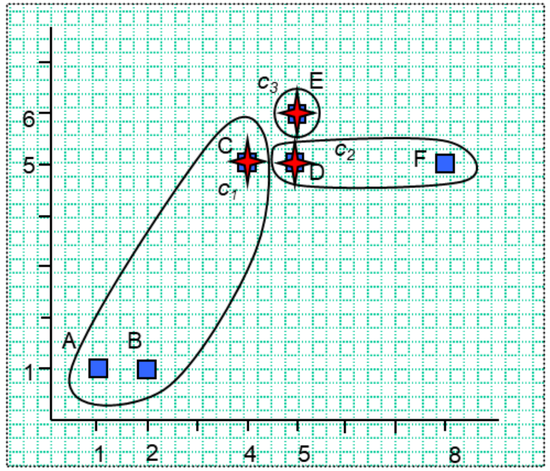


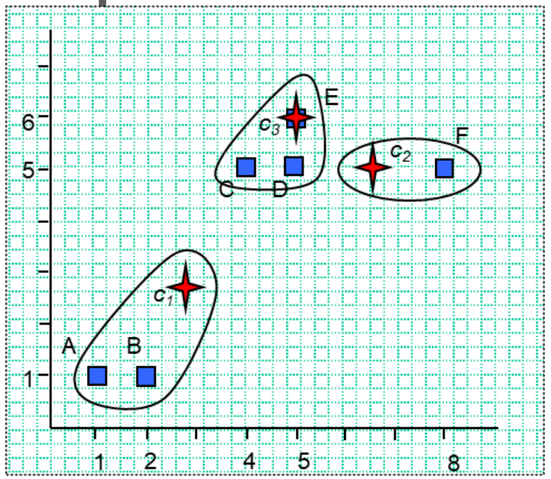
We may get different results.

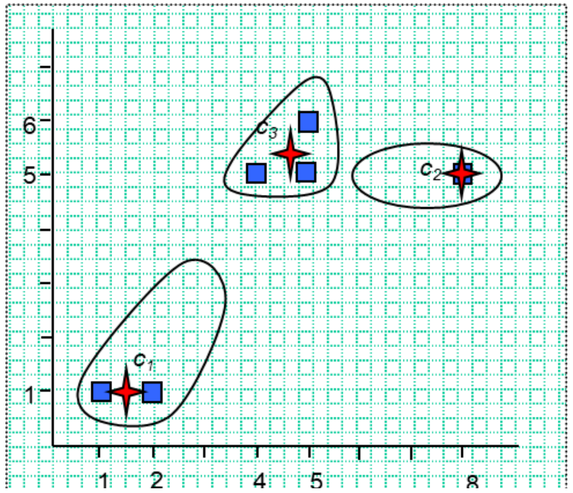
K-mean

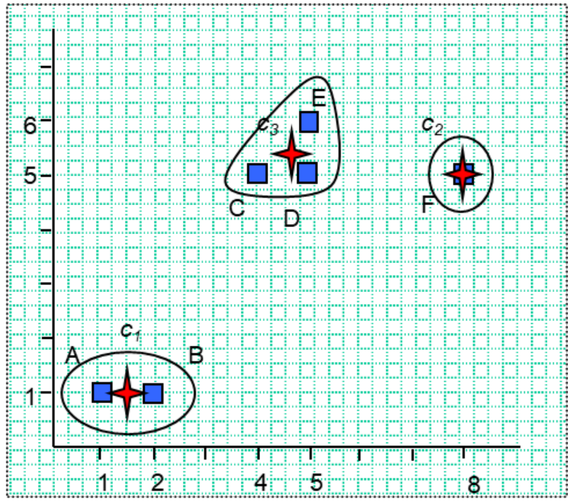
What if we initialize in different way. Example 3







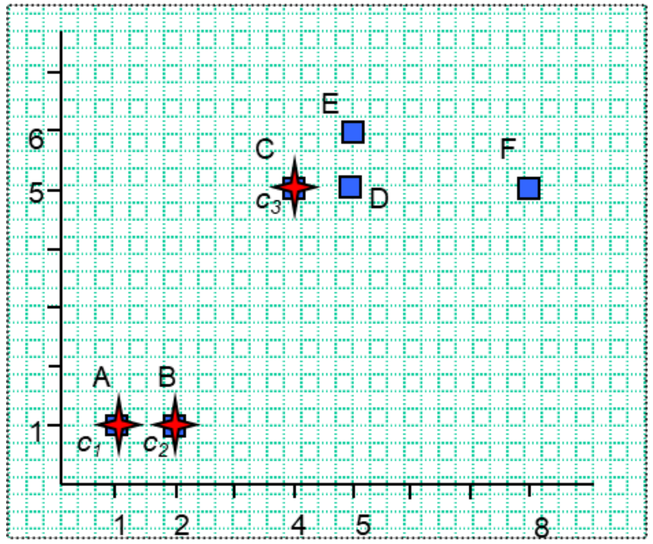


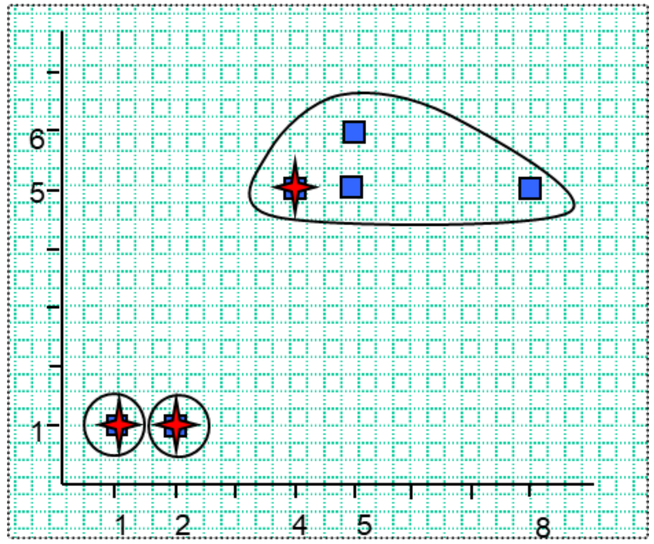


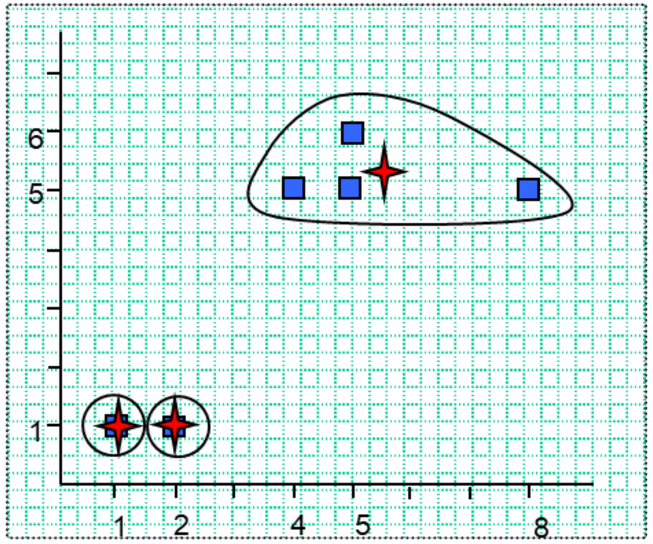
We may get different results.

K-mean

What if we initialize in different way. Example 4







We may get different results.

Kmean++

A good initializer, kmean ++

- Choose a data point, *x* randomly as center. Let C={x}.
- Choose the second center that is far from the first center.

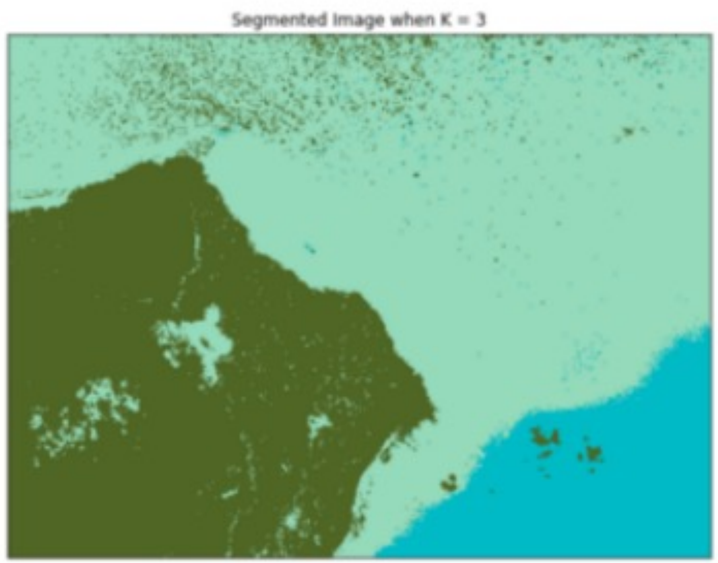
$$\Pr(x) \propto dist(x, C)^2$$

where $dist(x, C) = min_{z \in C} ||\mathrm{x} - \mathrm{z}||$

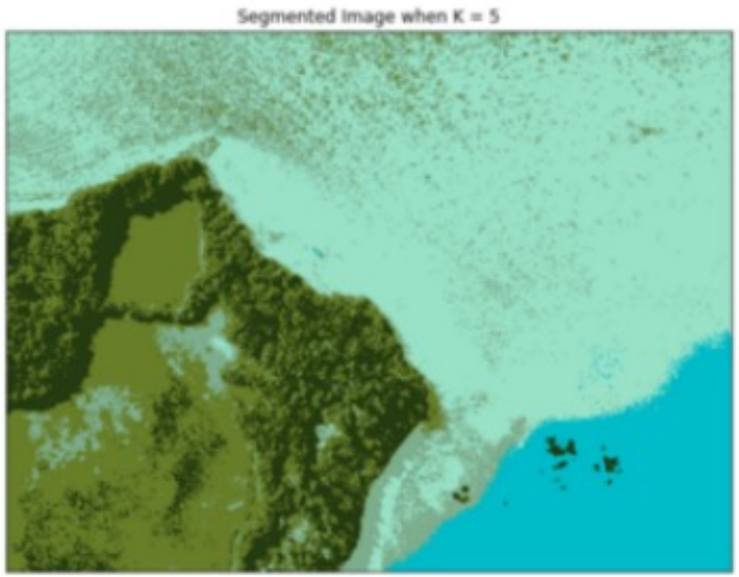
- Repeat until desired number of center is attained.

This is to ensure centers are far apart

Kmean for Segmentation



K=3



K=5

Used case

1. **Document Classification** – cluster the document according to the terms used. Group similar document together.

2. **Delivery store optimization** - use a combination of k-means to find the optimal number of launch locations and a genetic algorithm to solve the truck route as a traveling salesman problem. Paper here.
3. **Identifying Crime Localities**

4. **Customer Segmentation** - how telecom providers can cluster pre-paid customers to identify patterns in terms of money spent in recharging, sending sms, and browsing the internet. the classification would help the company target specific clusters of customers for specific campaigns. Paper here.

https://dzone.com/articles/10-interesting-use-cases-for-the-k-means-algorithm

Summary - Kmean

A particularly simple method for clustering is K-means, which is identical to the generalized Lloyd algorithm we know from vector quantization, just applied to clustered data.

The idea is to represent each cluster k by a center point c_k and assign each data point x_n to one of the clusters k, which can be written in terms of index sets C_k.

The center points and the assignment are then chosen such that the mean squared distance between data poin

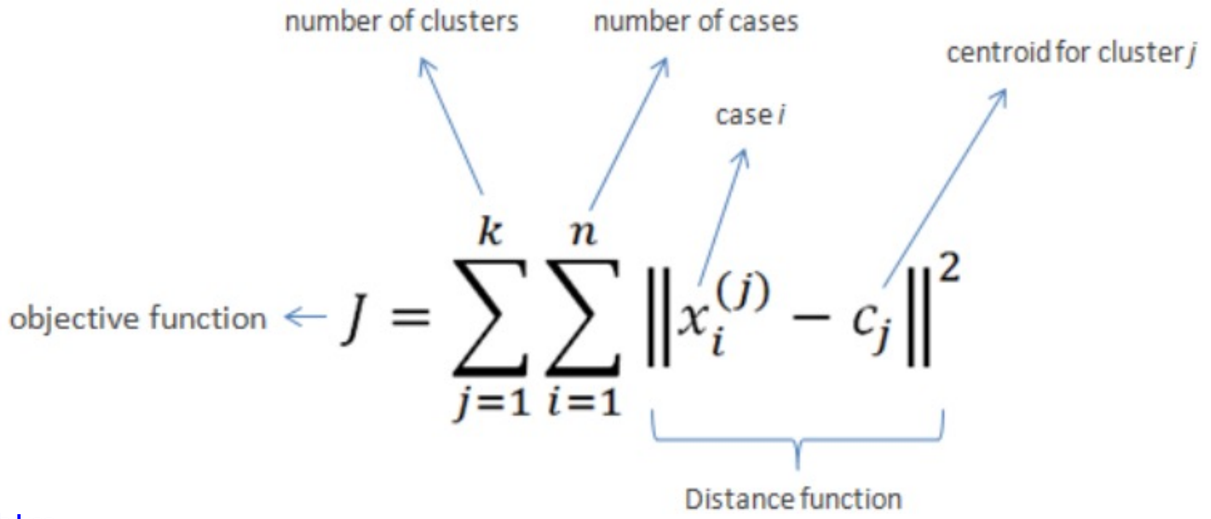
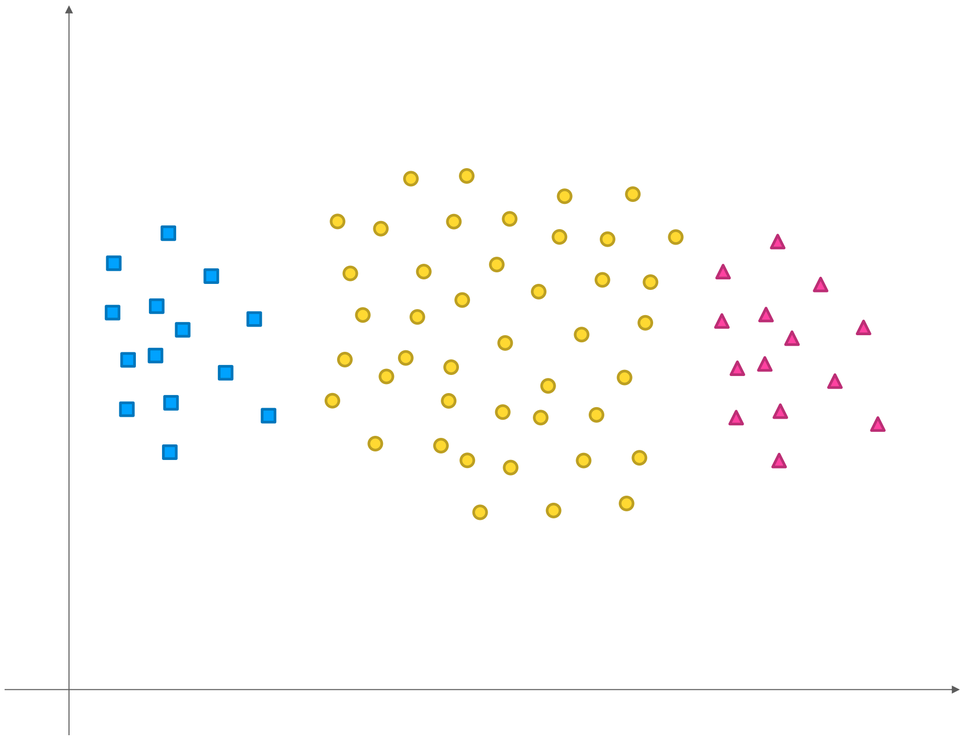


Image from https://www.saedsayad.com/clustering_kmeans.htm

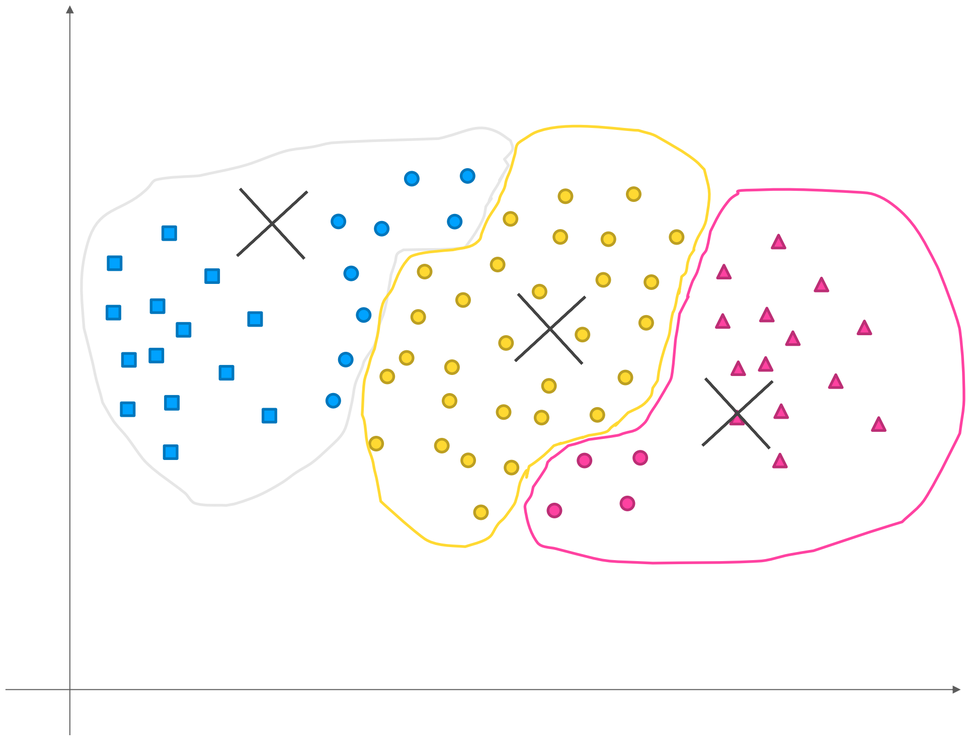
Limitation

- K-means has problems when clusters are of different
  - Sizes
  - Densities
  - Non-globular shapes
- K-means has problems when the data contains outliers.

Limitation – Different Size

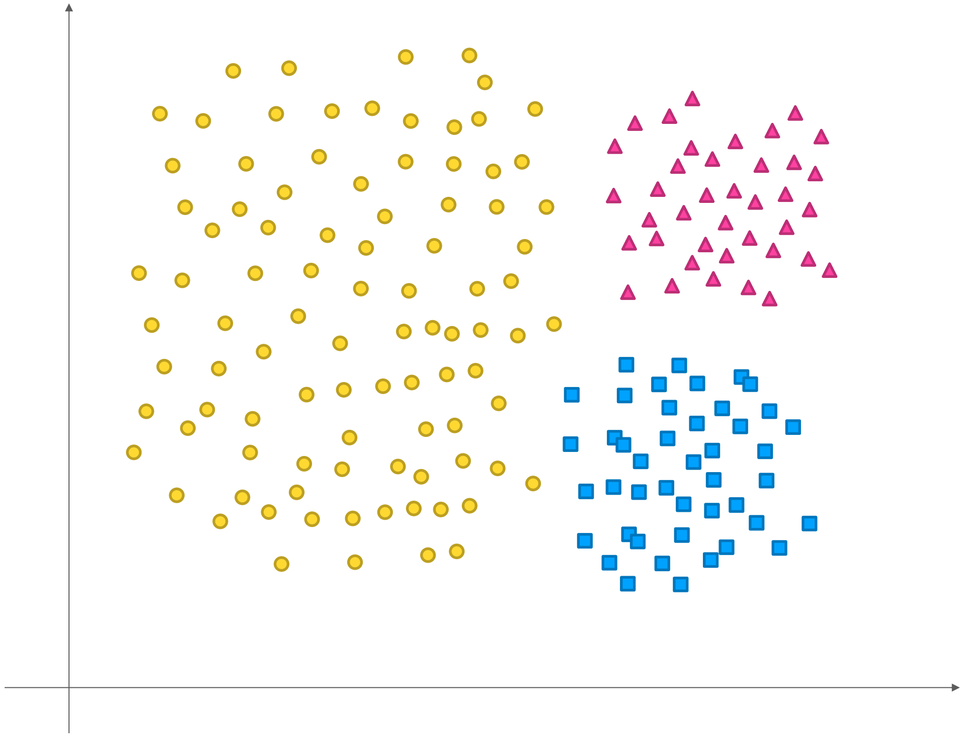


Data point

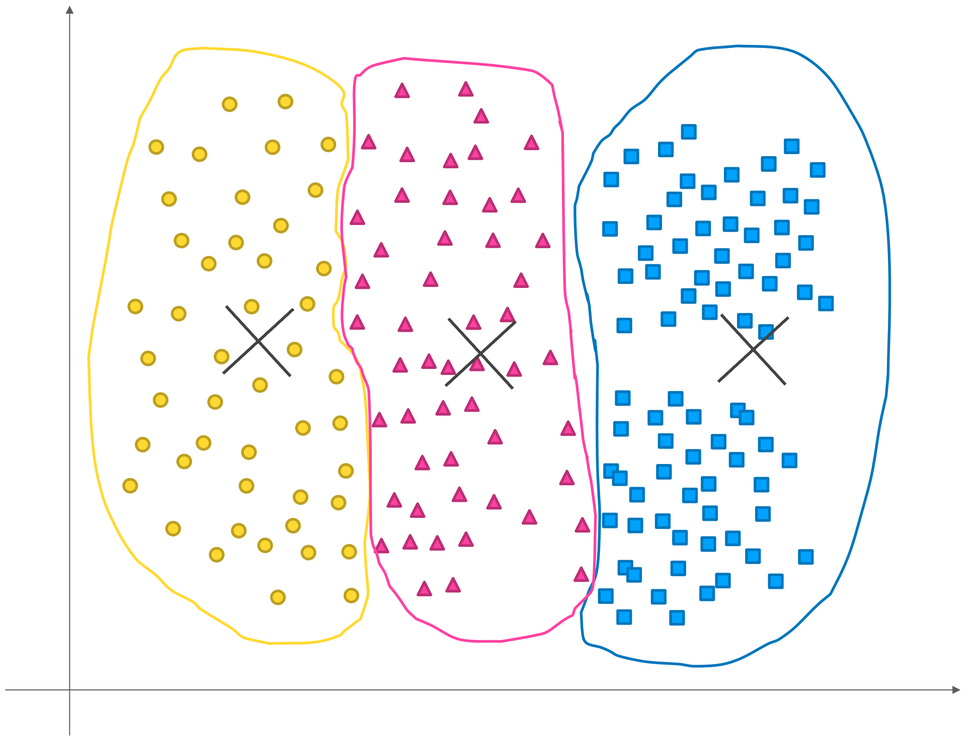


K = 3

Limitation – Different Density

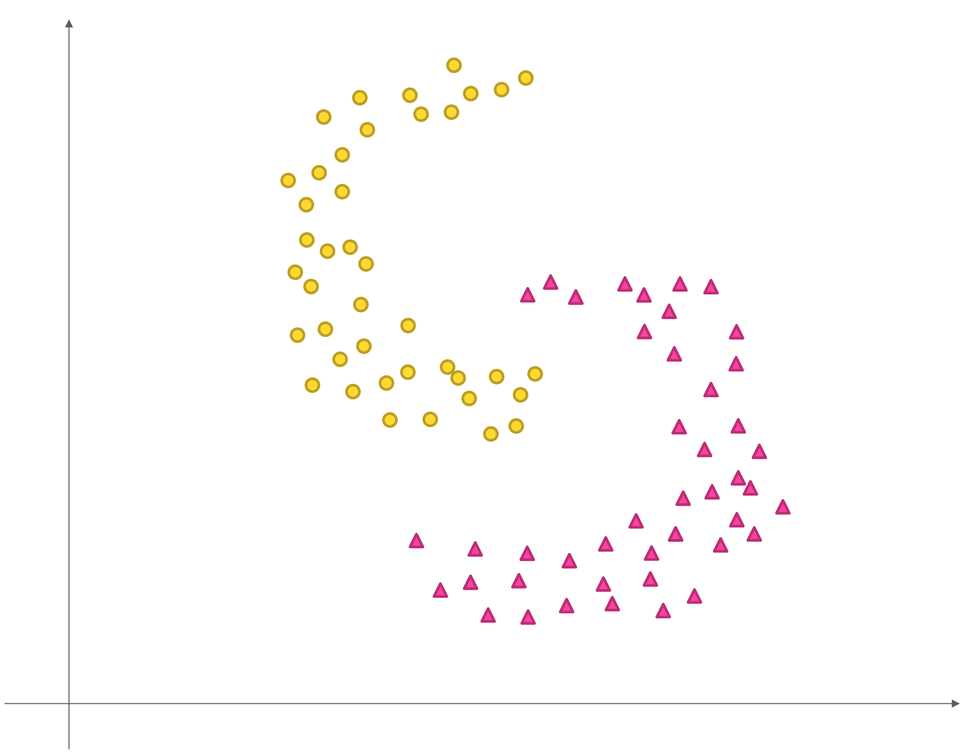


Data point

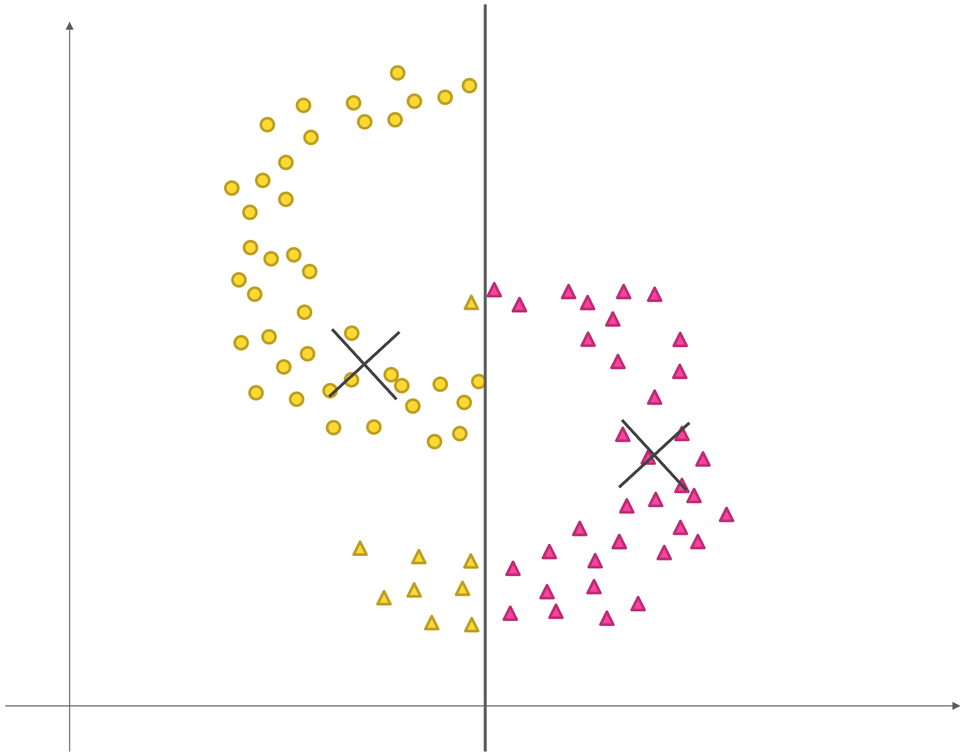


K = 3

Limitation – non globular shape



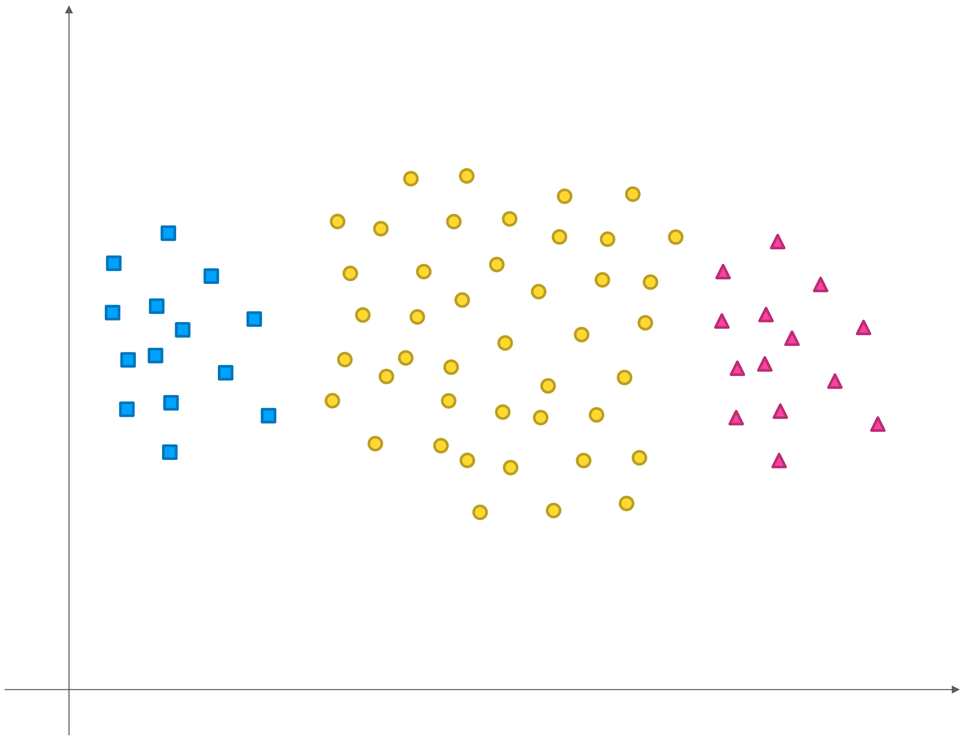
Data point



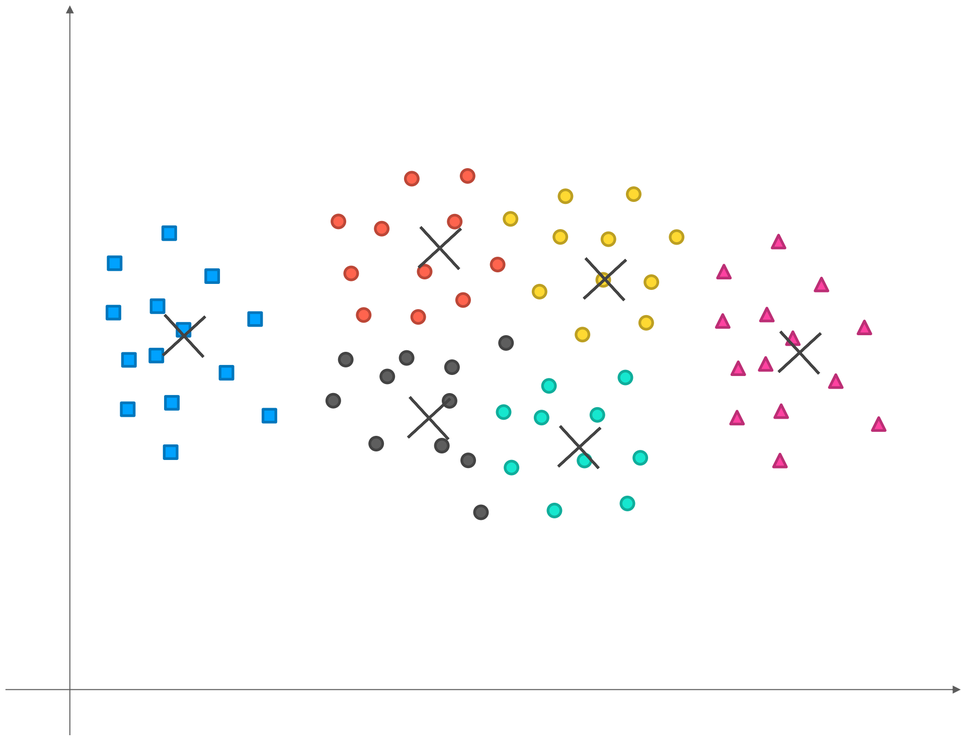
K = 2

To overcome k-mean limitation

Generate more cluster. Then combine similar cluster



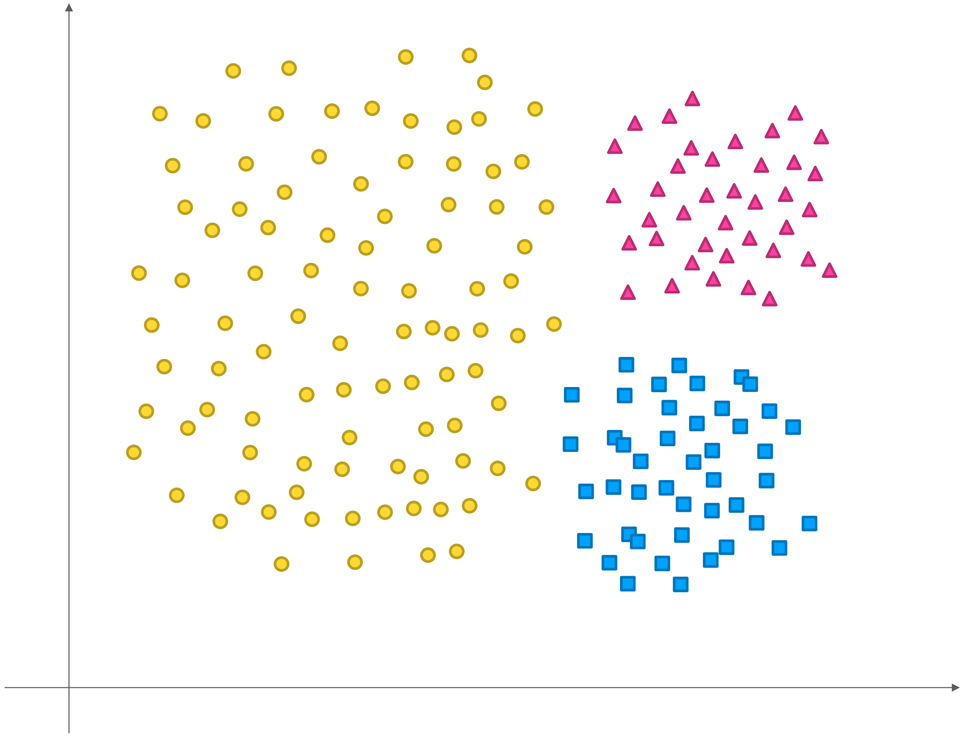
Data point



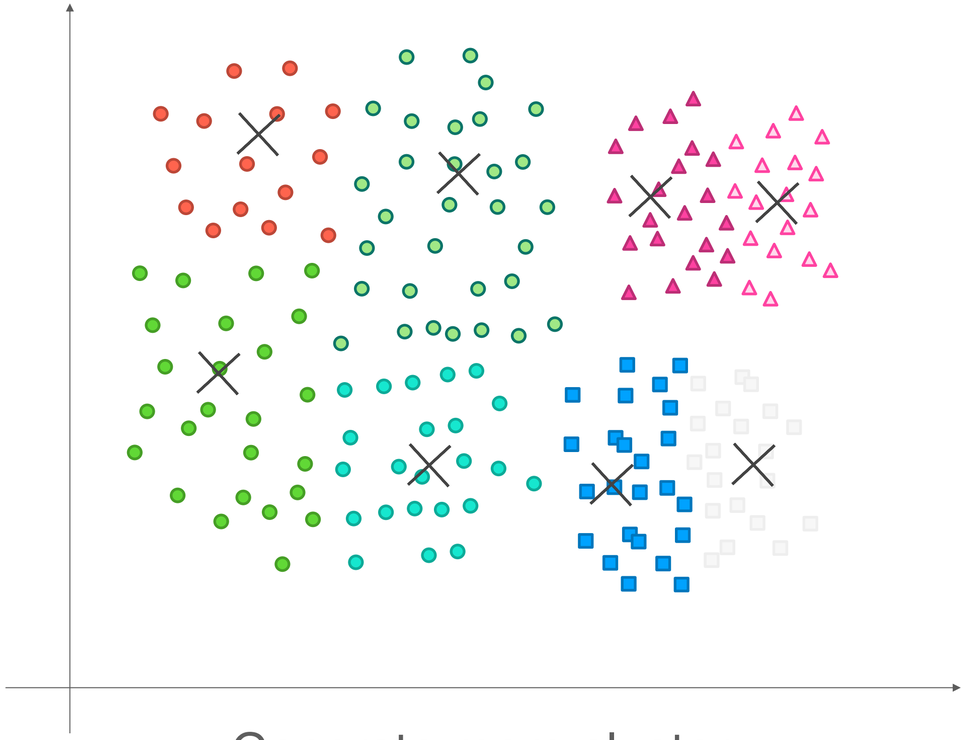
Generate more cluster

To overcome k-mean limitation

Generate more cluster. Then combine similar cluster



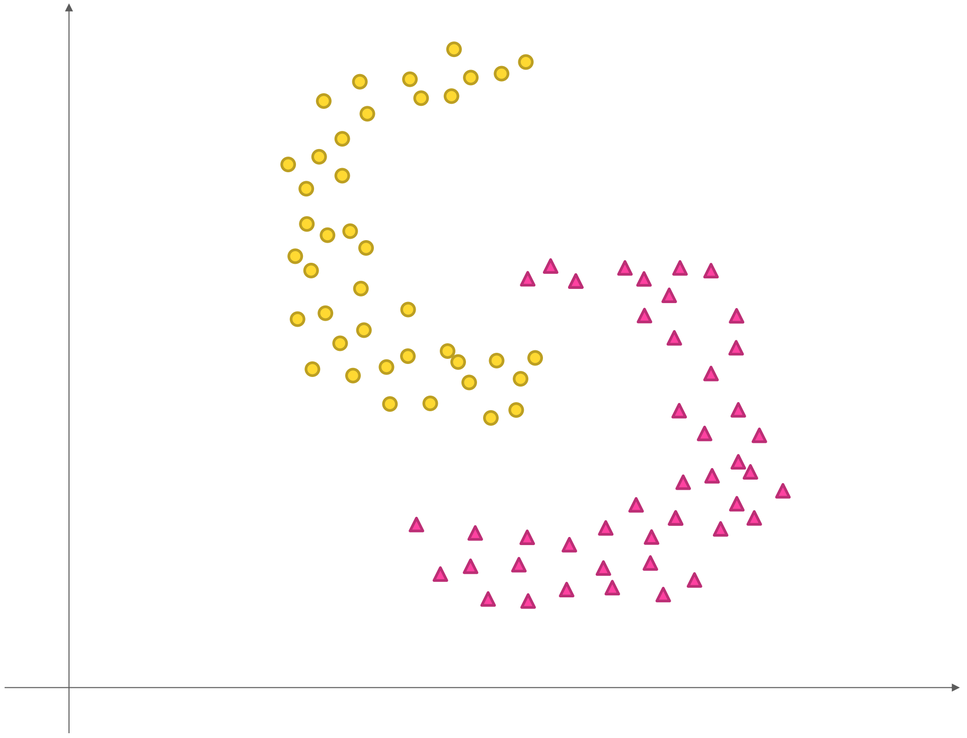
Data point



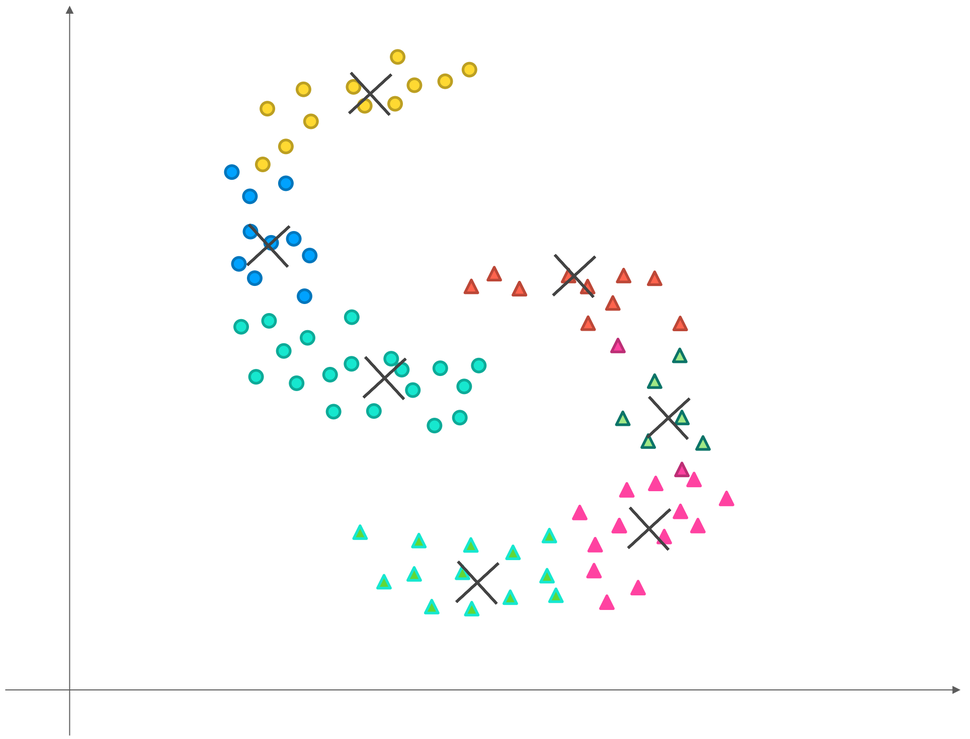
Generate more cluster

To overcome k-mean limitation

Generate more cluster. Then combine similar cluster



Data point

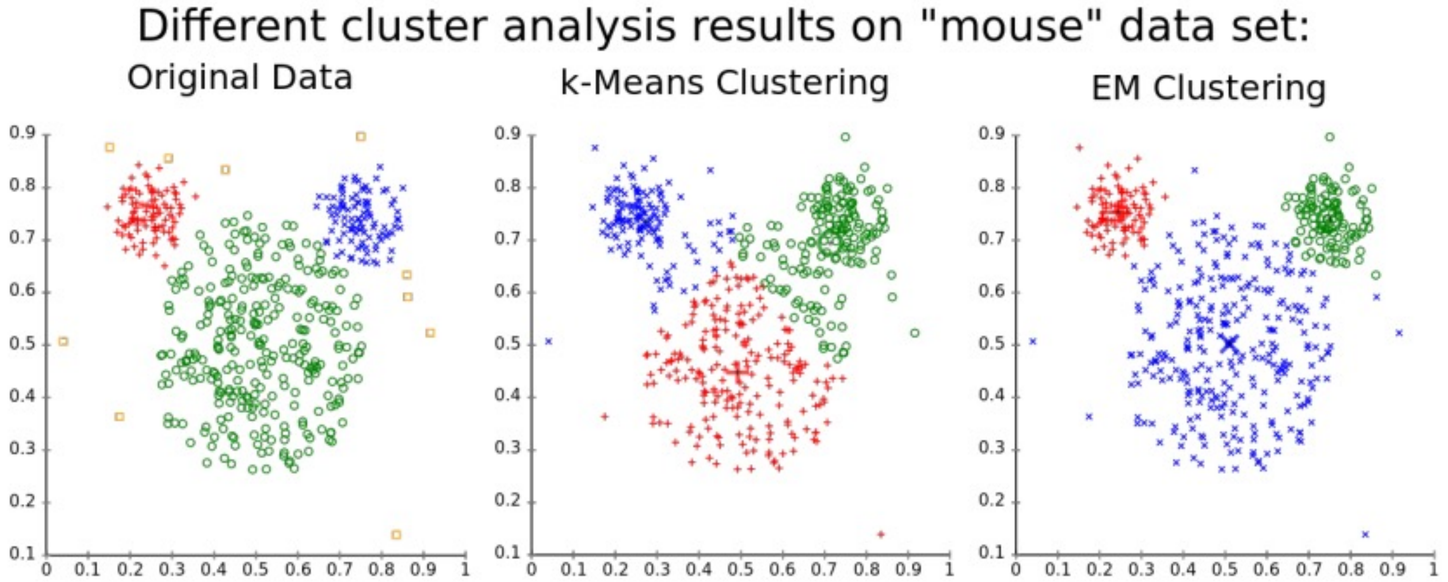


Generate more cluster

To overcome k-mean limitation

- Generate more cluster. Then combine similar cluster

- EM clustering using Gaussian Mixture Model
- Intuition: Assume that the data generating process is a mixture of Multivariate Normals

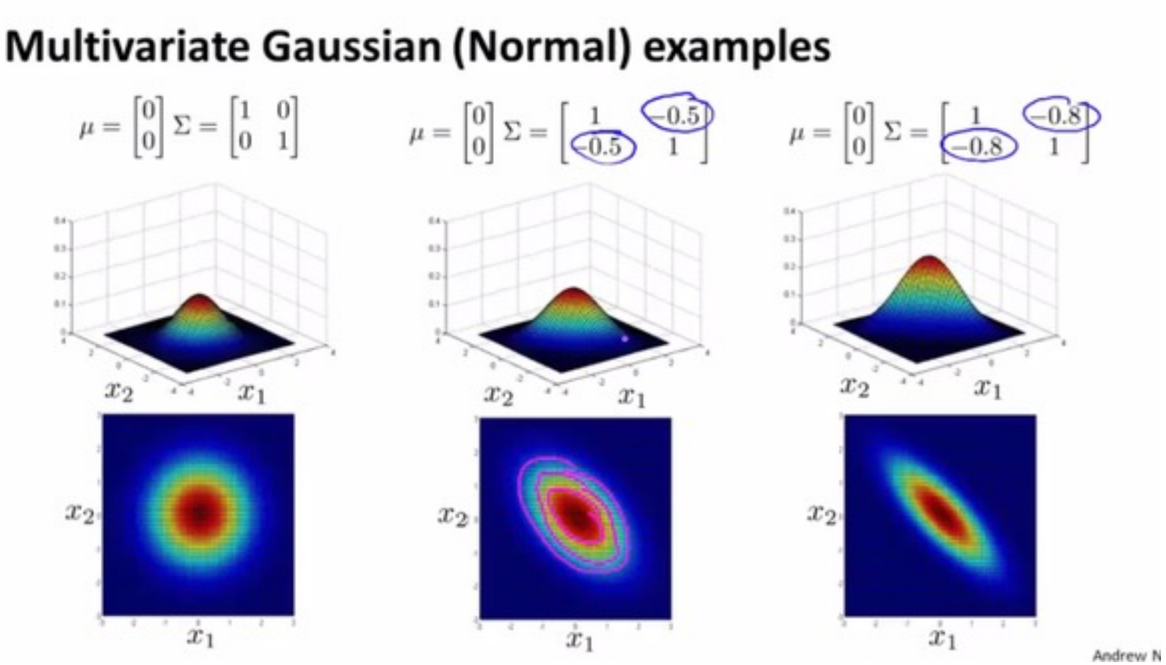


[source: Wikipedia, Public Domain image]

EM clustering

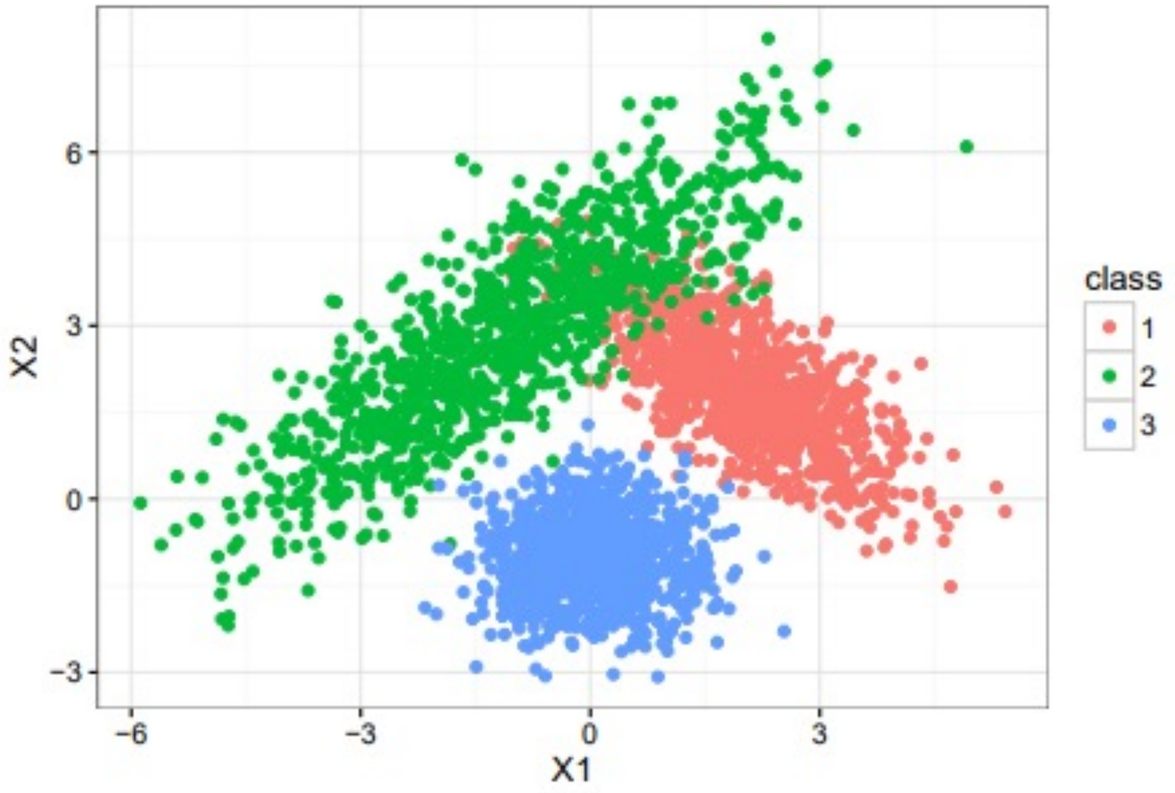
- Assume each observation has probability π_k of coming from cluster *k*

- Assume that all observations from cluster *k* a drawn randomly from a Multivariate Variable Normal (μ_k, Σ_k) distribution

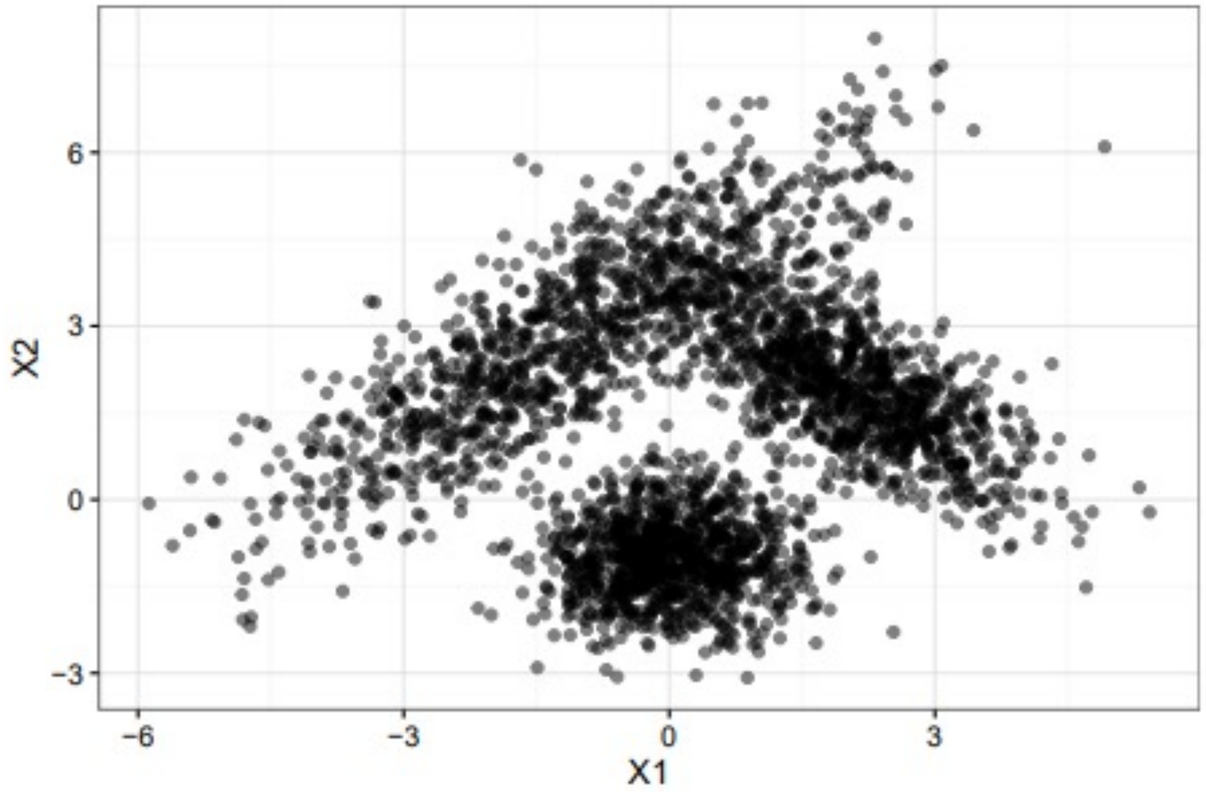


EM clustering

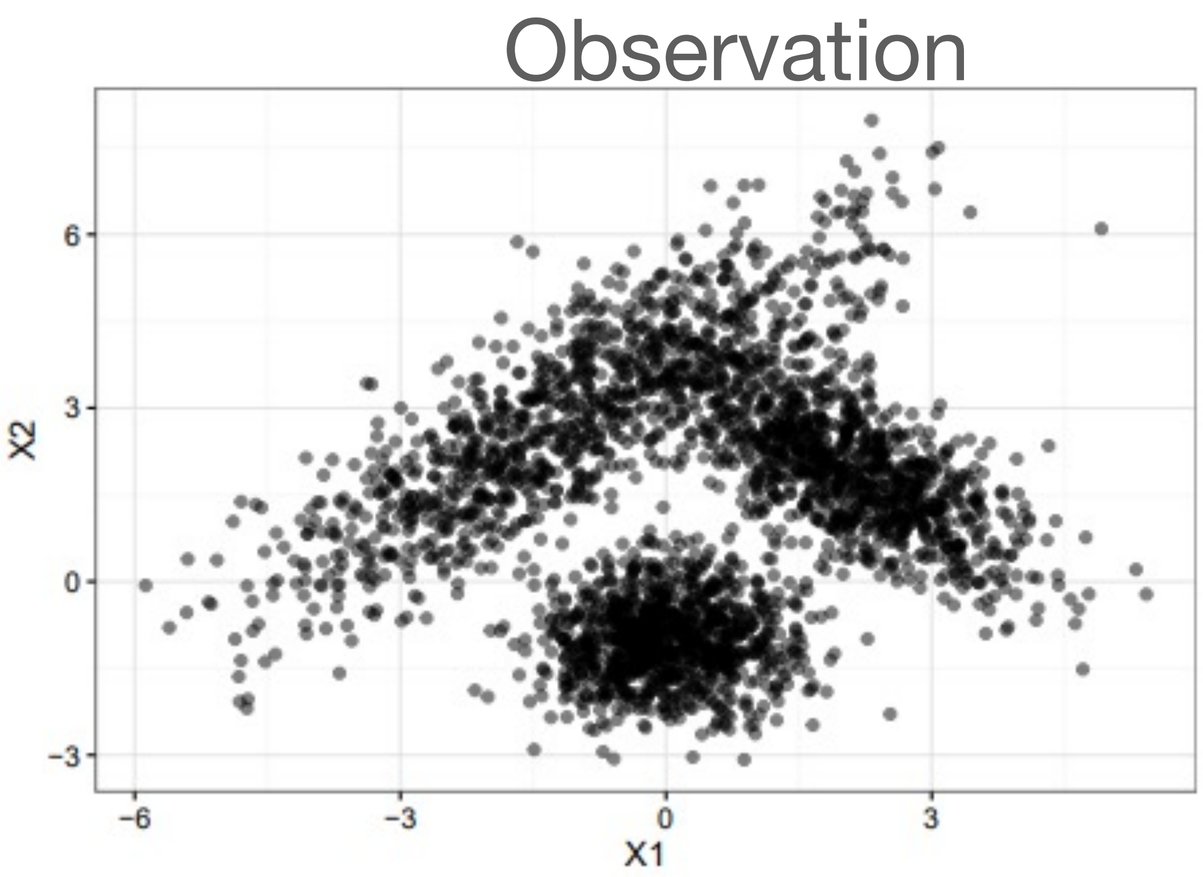
Ground Truth



Observation



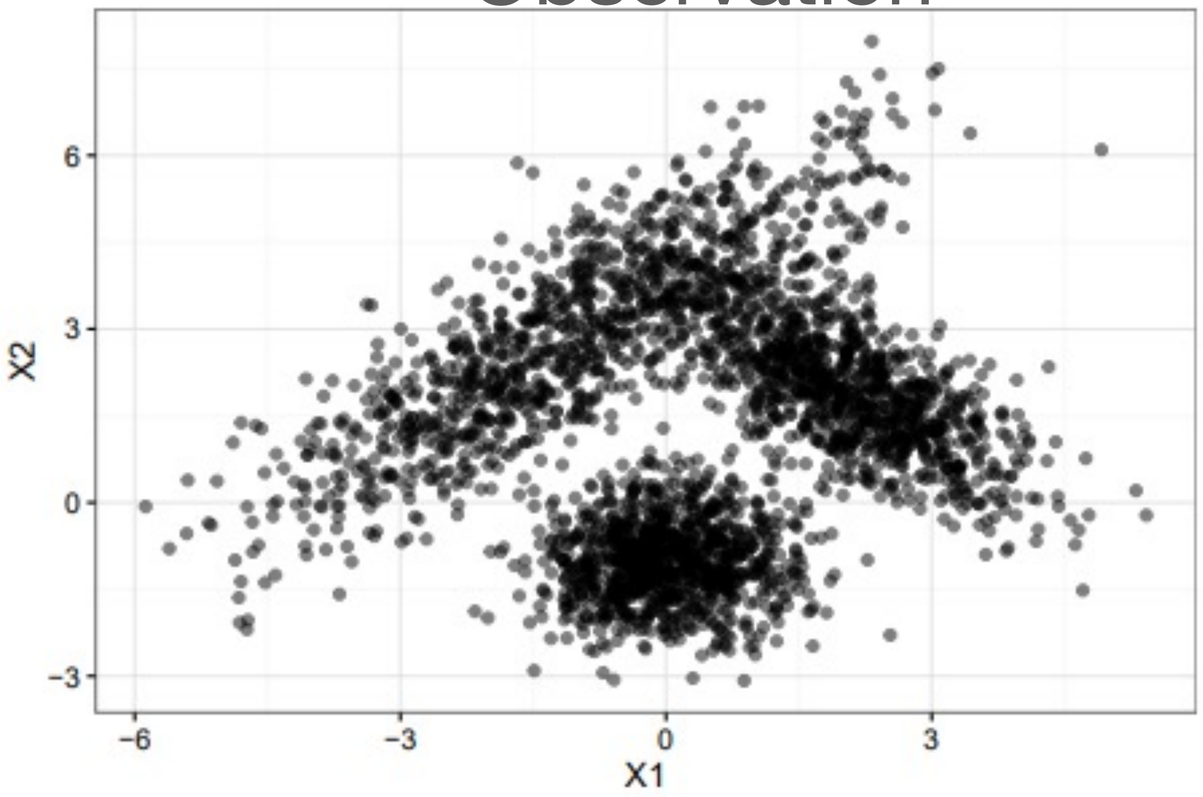
EM clustering



- We don’t know what the mixture components look like ahead of time
- We’ll use an iterative algorithm that feels kind of like K-means
  - Start by fixing K at some value
- **E-step**: Now, if we have estimates $\hat{\mu}_k, \hat{\Sigma}_k, \hat{\pi}_k$, we can calculate the posterior probability that observation *i* comes from class *k*
- **M-step:** If we have the posterior probabilities, we can get updated estimates of $\hat{\mu}_k, \hat{\Sigma}_k, \hat{\pi}_k$

EM clustering

Observation



- Take initial guesses for the parameters $\hat{\mu}_k, \hat{\Sigma}_k, \hat{\pi}_k$

- Expectation step: Compute the posterior probability (responsibilities) over z given our current model - i.e. how much do we think each Gaussian generates each datapoint.

$$\gamma_{ik} = \frac{\pi_k \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j)}$$

- Maximization step: Assuming that the data really was generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} \mathbf{x}_i \\
\Sigma_k &= \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top \\
\pi_k &= \frac{N_k}{N} \quad \text{where } N_k = \sum_{i=1}^{N} \gamma_{ik}
\end{aligned}$$

- Repeat until convergence

EM clustering

Data: $x = (x_1, x_2, \ldots, x_N)$

OBJECTIVE: Fit mixture of Gaussian model with K=2 components

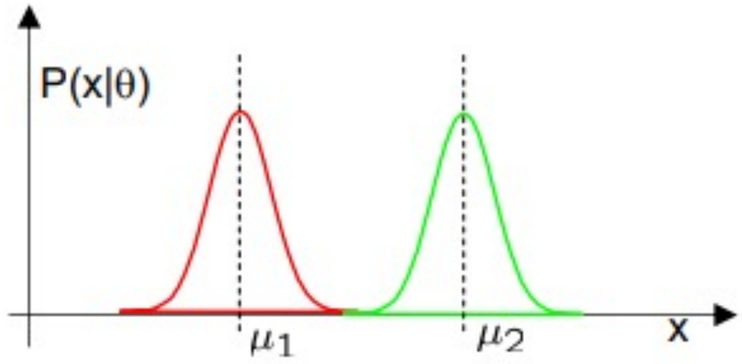
Model:

$$p(x_i|\theta) = \sum_k \pi_k \, N(x_i|\mu_k, \sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1$$

Parameters: $\theta = \{\pi, \mu, \sigma\}$

keep π, σ fixed

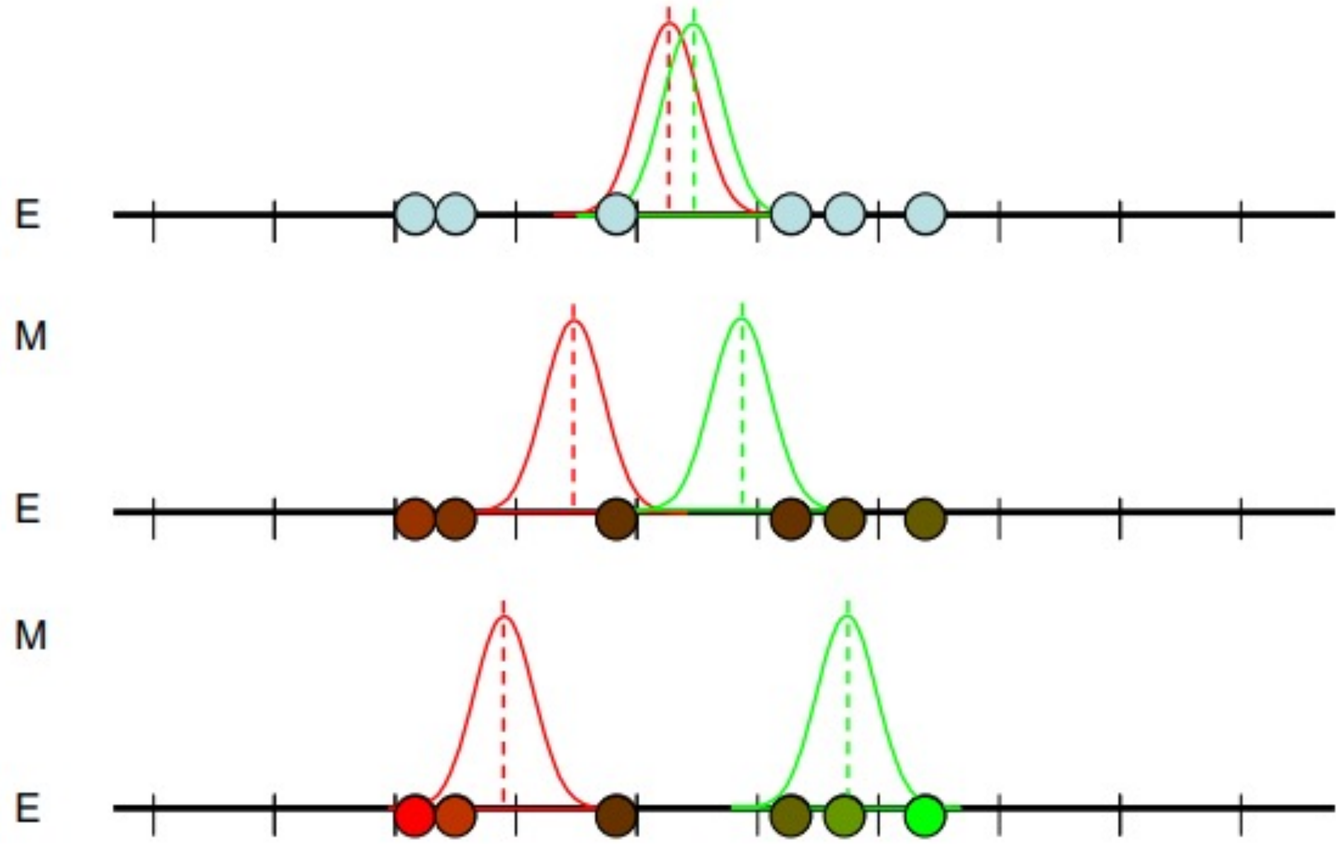
i.e. only estimate μ



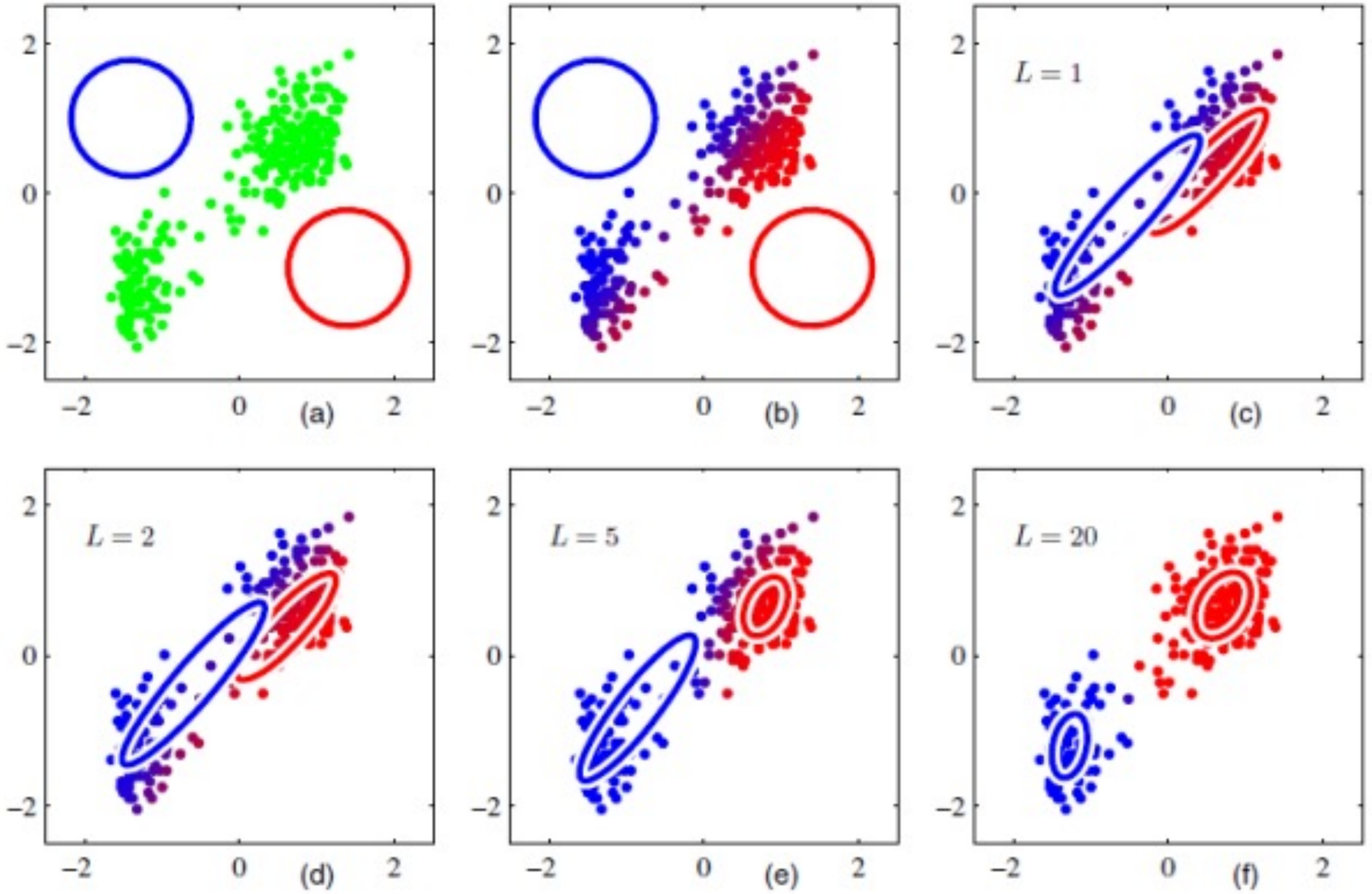
EM clustering intuition

E-step: Compute soft assignment of the points, using current parameters

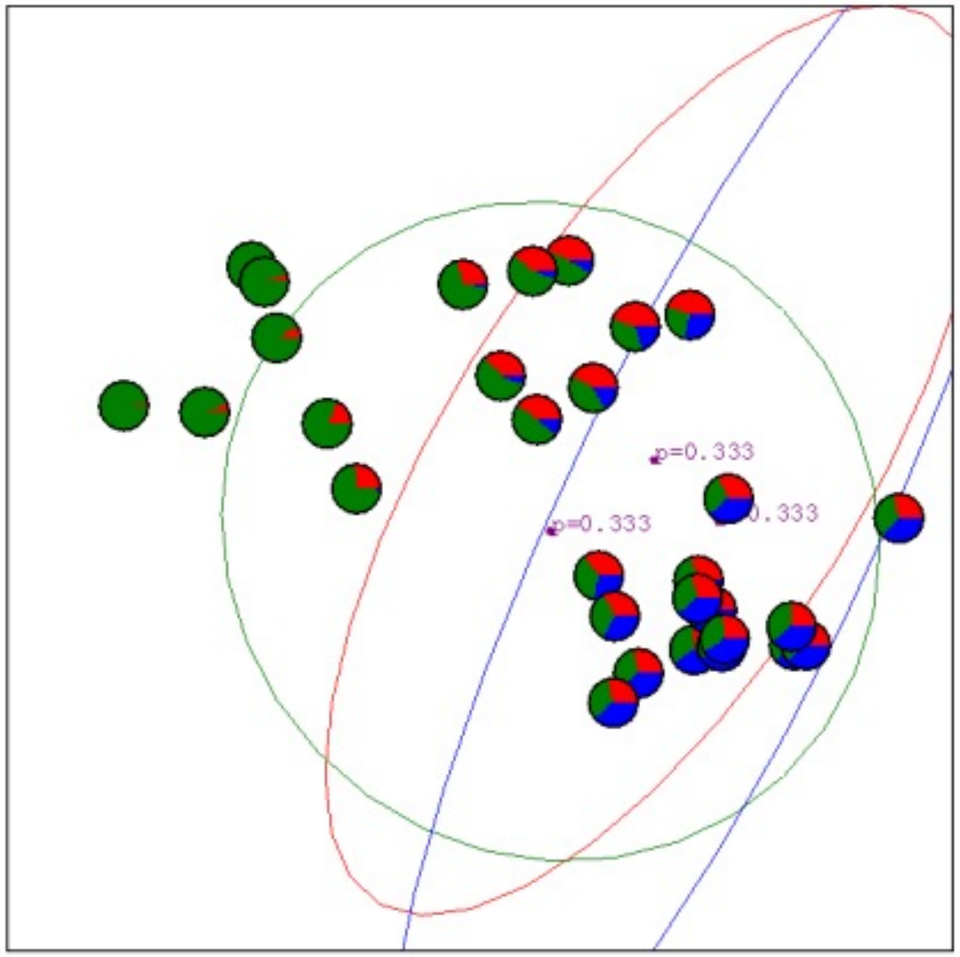
M-step: Update parameters using current responsibilities



EM clustering

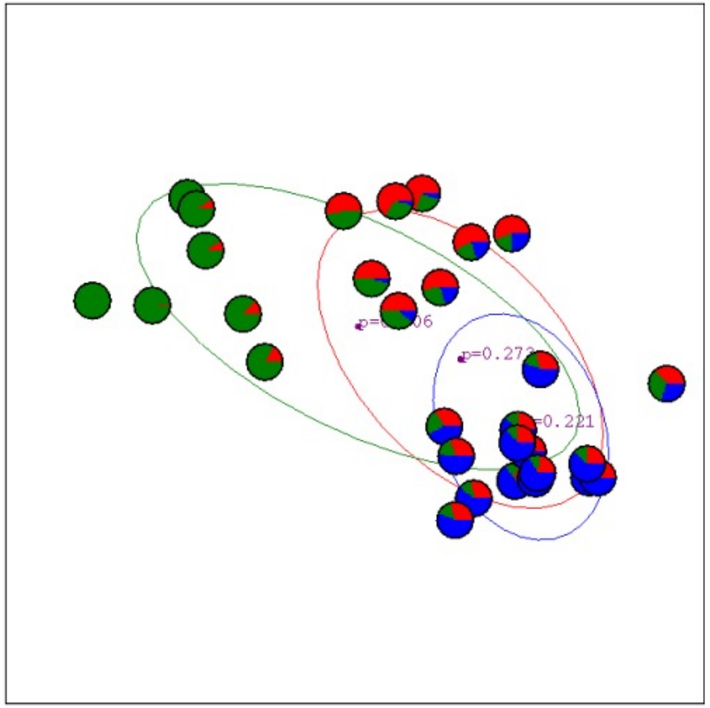


EM clustering

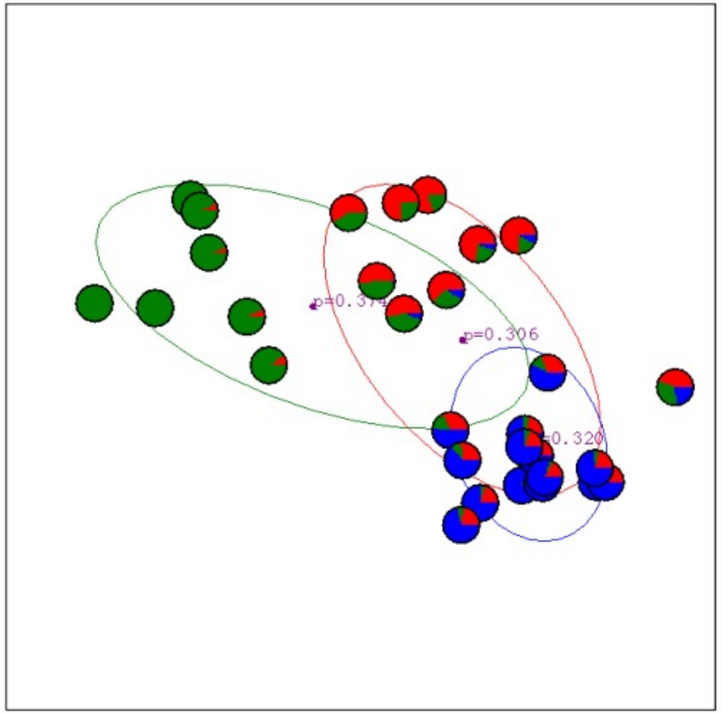


- Each circle (mini pie-chart) is an observation
- Large ovals in the background represent initial $\hat{\mu}_k$, $\hat{\Sigma}_k$.
  $\hat{\pi}_k = 1/3$ for all 3 classes
- Pie chart segments correspond to responsibilities estimates from current $\hat{\mu}_k$, $\hat{\Sigma}_k$, $\hat{\pi}_k$.

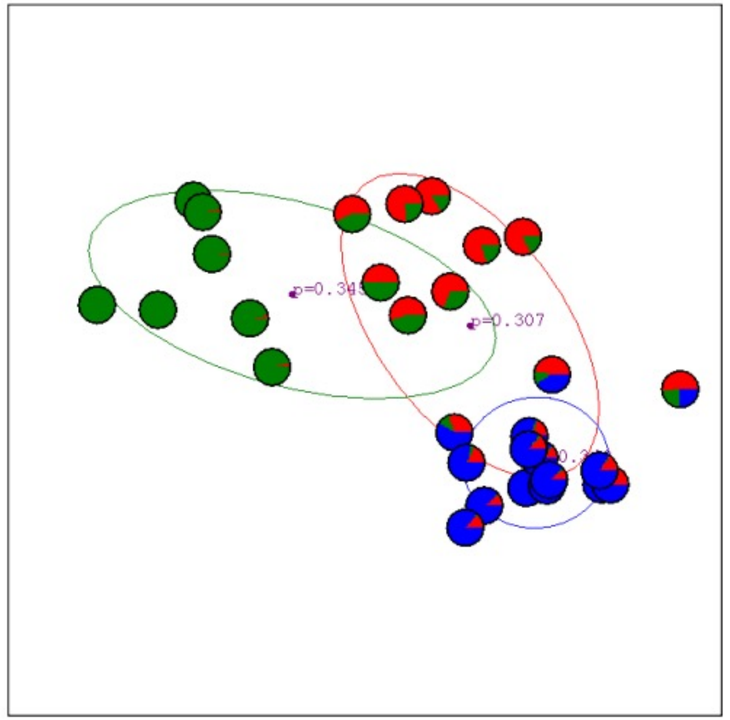
EM clustering



1 iteration of both the E-step and M-step

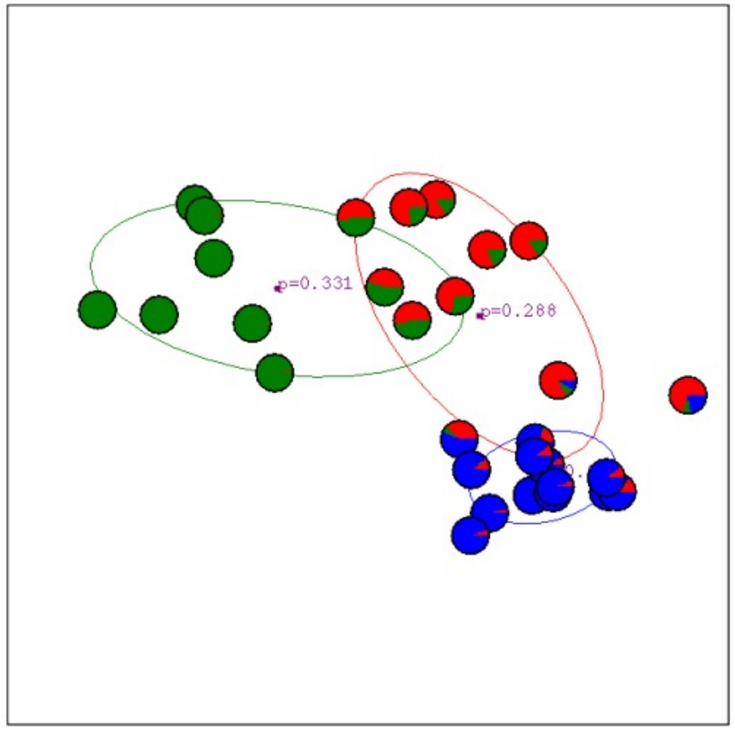


another iteration of both the E-step and M-step

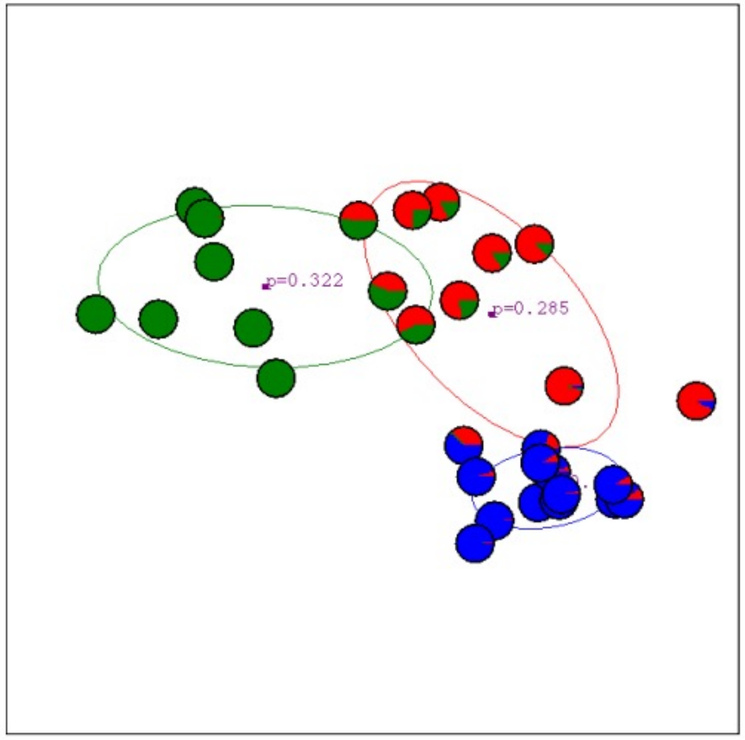


another iteration of both the E-step and M-step.

EM clustering

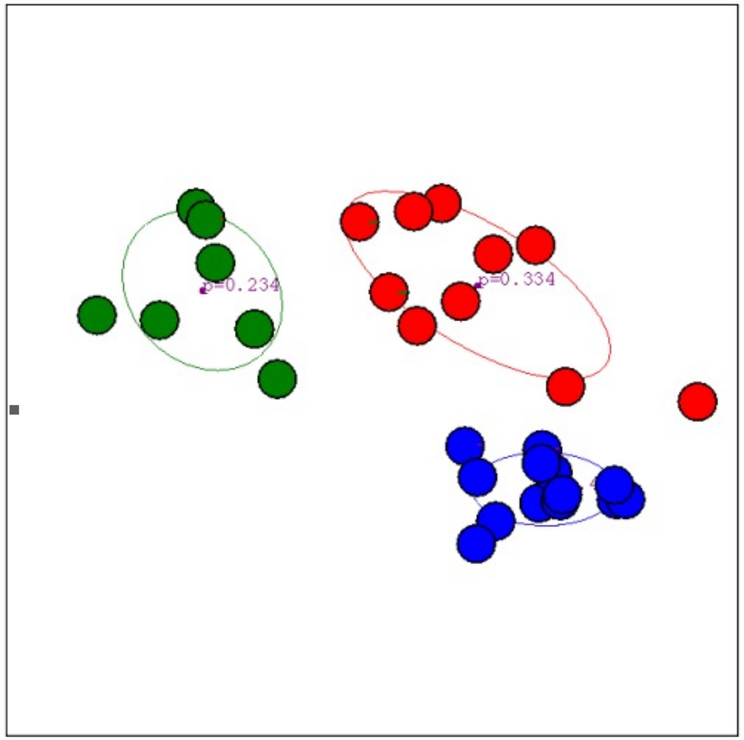


another iteration of both the E-step and M-step..



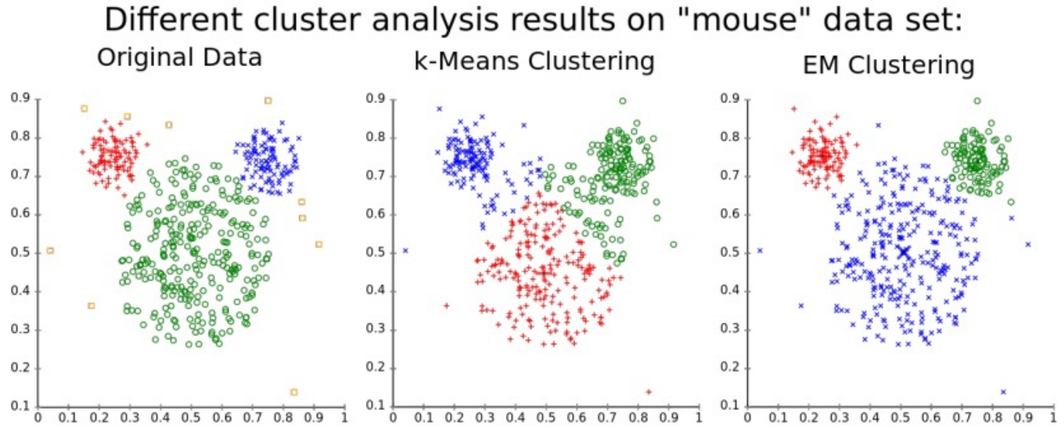
another iteration of both the E-step and M-step...

...



Final picture: algorithm has converged

EM clustering



[source: Wikipedia, Public Domain image]

- GMM's do better on this example because they essentially allow for a data-adaptive notion of distance when assigning points to centroids

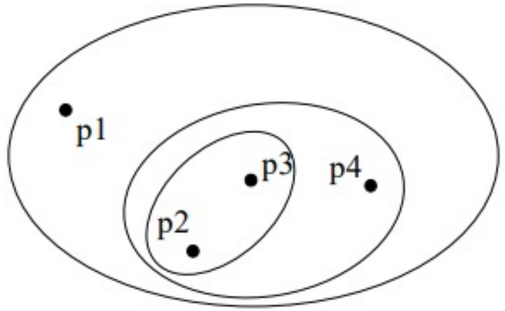
- i.e., In the original data, we have 2 clumps with small variance, and one clump with large variance

- K-means can't capture this added information

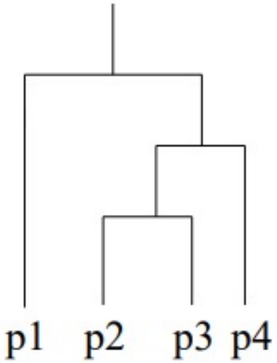
- GMM's say: An observation belongs to C_k if its variance-adjusted (i.e., Σ_k-adjusted) distance to μ_k is small

Hierarchy Clustering

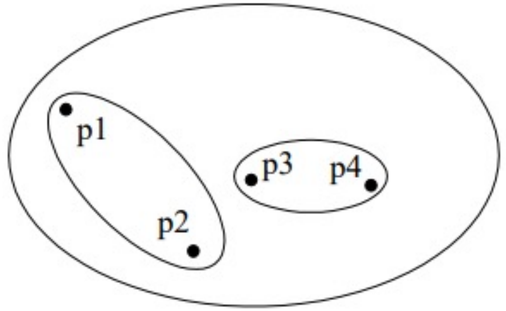
Data are in nested clusters, organized in a hierarchical tree



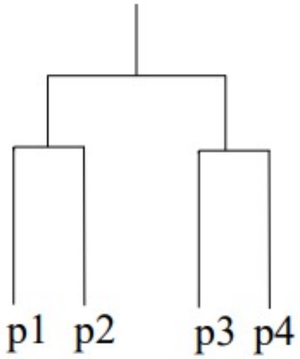
Traditional Hierarchical Clustering



Traditional Dendrogram



Non-traditional Hierarchical Clustering



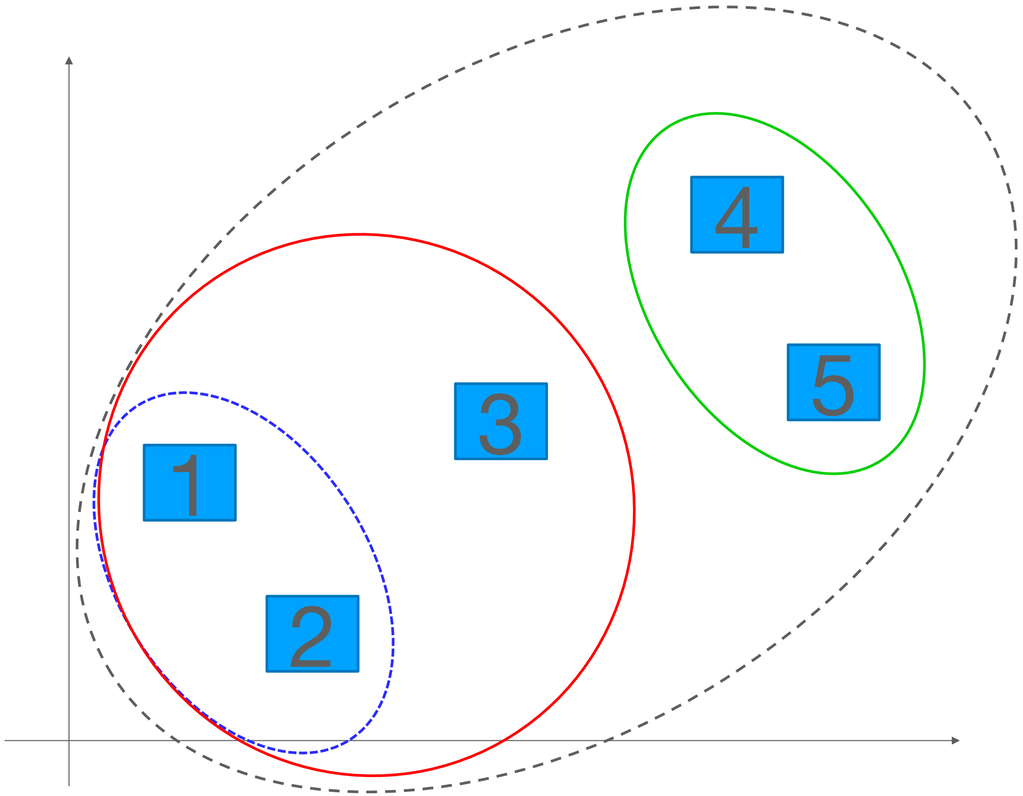
Non-traditional Dendrogram

Hierarchy Clustering

- **Agglomerative** (bottom up): Initially, each point is a cluster repeatedly combine the two “nearest” clusters into one

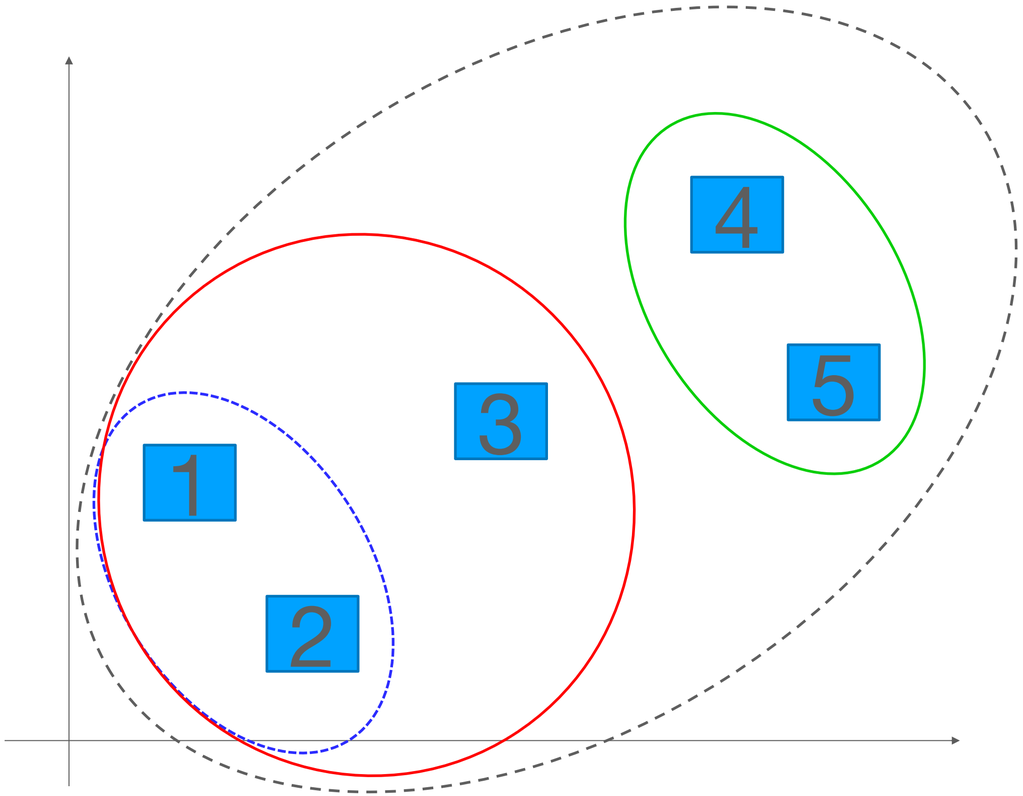
- **Divisive** (top down): Start with one cluster and recursively split

Agglomerative (bottom up)

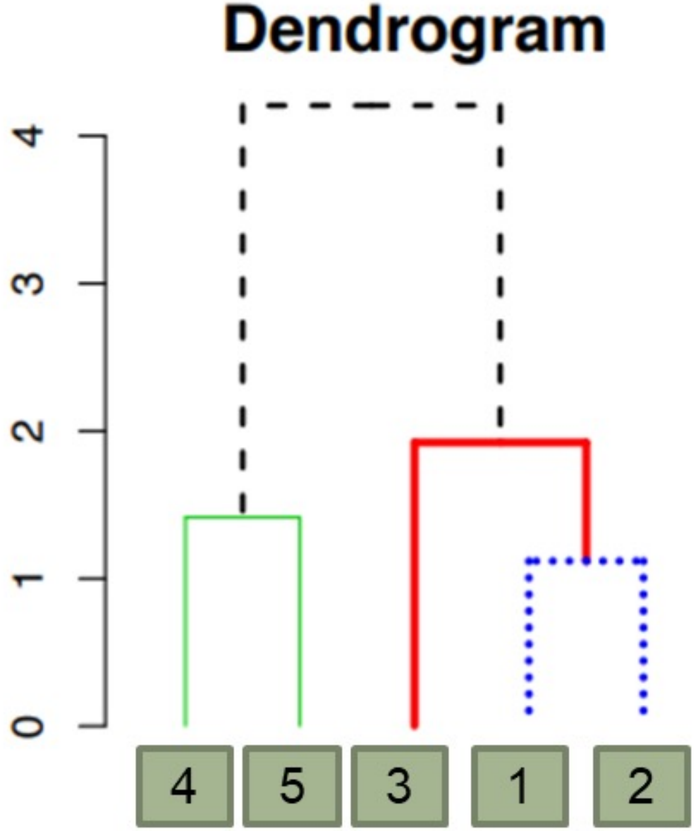


1. Each point starts as its own cluster
2. Merge the two clusters (points) that are closet to each other
3. Merge the two other clusters (points) that are closet to each other
4. Merge the next two close clusters
5. Until at last all of the points are all in a single cluster

Agglomerative (bottom up)



Each point starts as its own cluster



y-axis on dendrogram is (proportional to) the distance between the clusters that got merged at that step

Compute dissimilarity

- Let $d_{ij} = d(x_i, x_j)$denote the dissimilarity (distance) between observation x_i and x_j

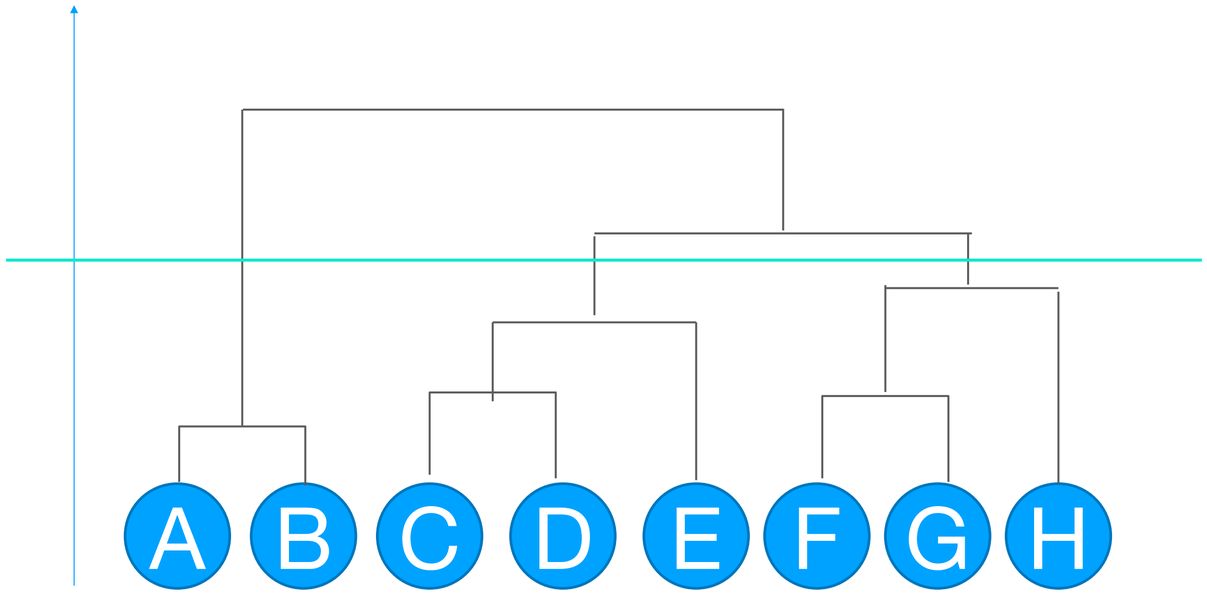
- Step 1: Each cluster is a single point, so we start by merging the two observations that have the lowest dissimilarity.

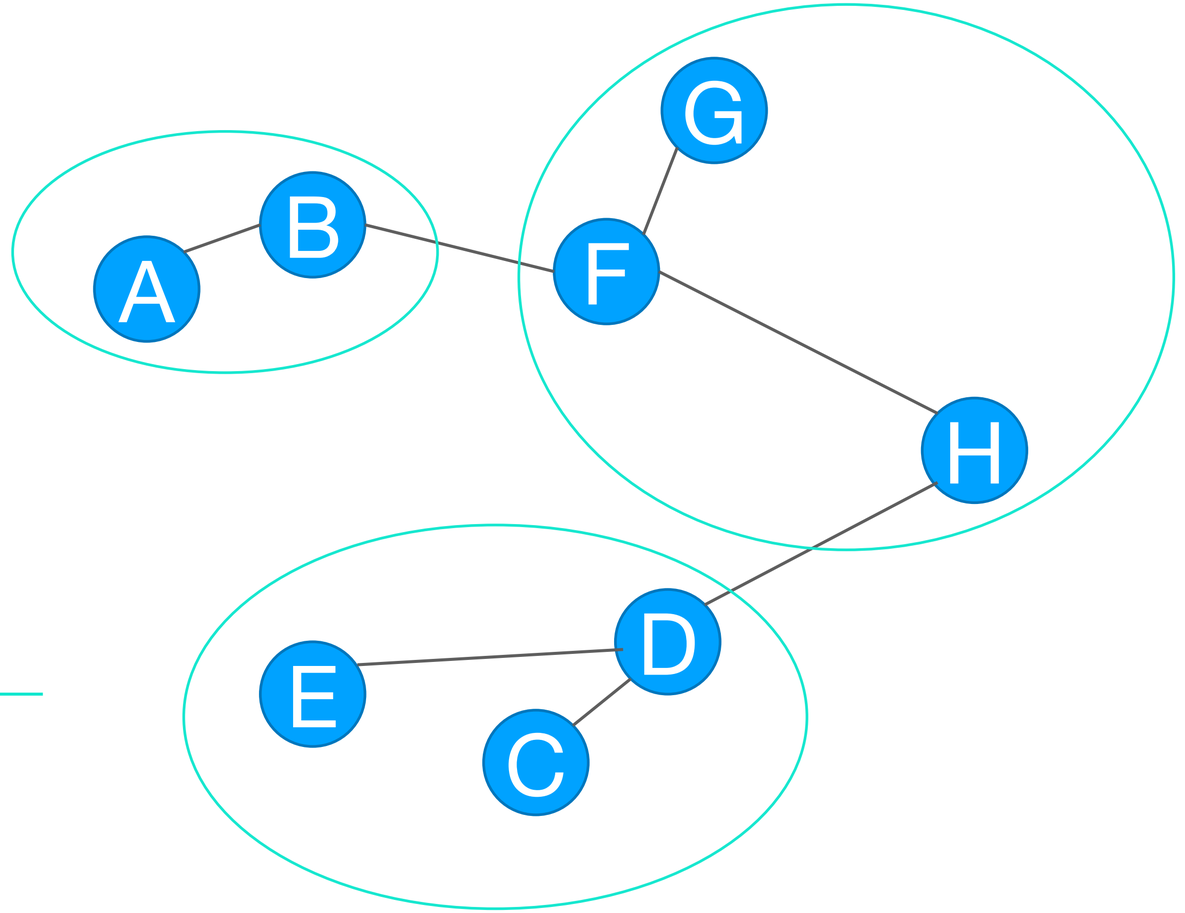
- Step 2: We compute distance between cluster, not between points.

  - The dissimilarity between two clusters is called the linkage
  - i.e., Given two clusters, Q and W, a linkage is a dissimilarity measure $d(Q, W)$ telling us how different the points in these sets are

Single Linkage

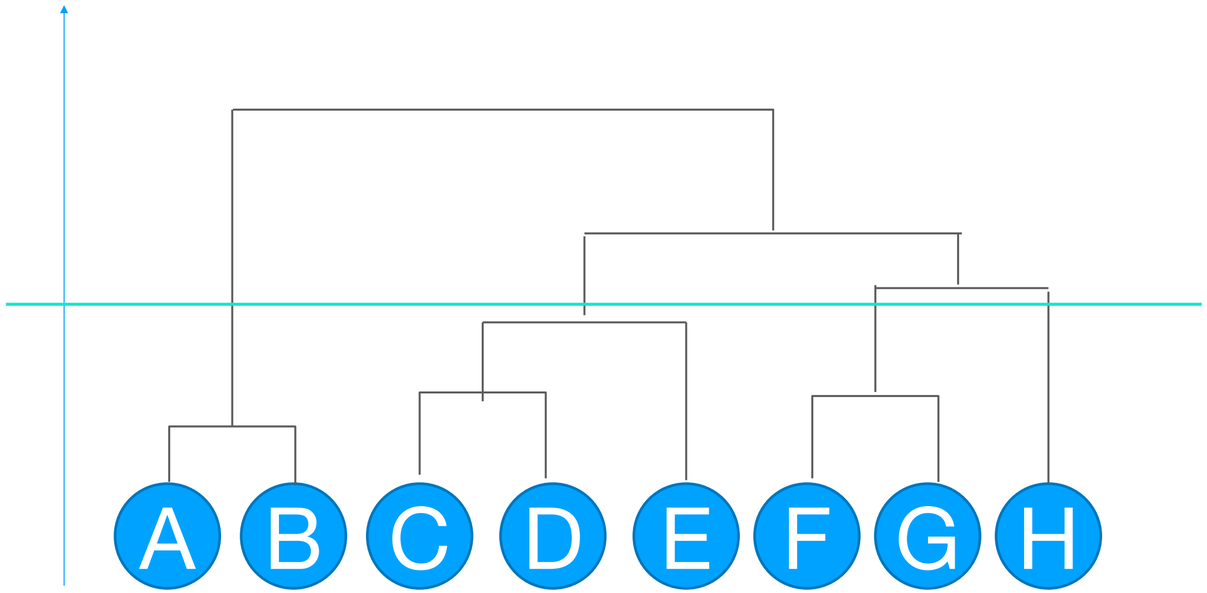
distance between the clusters that got merged at that step

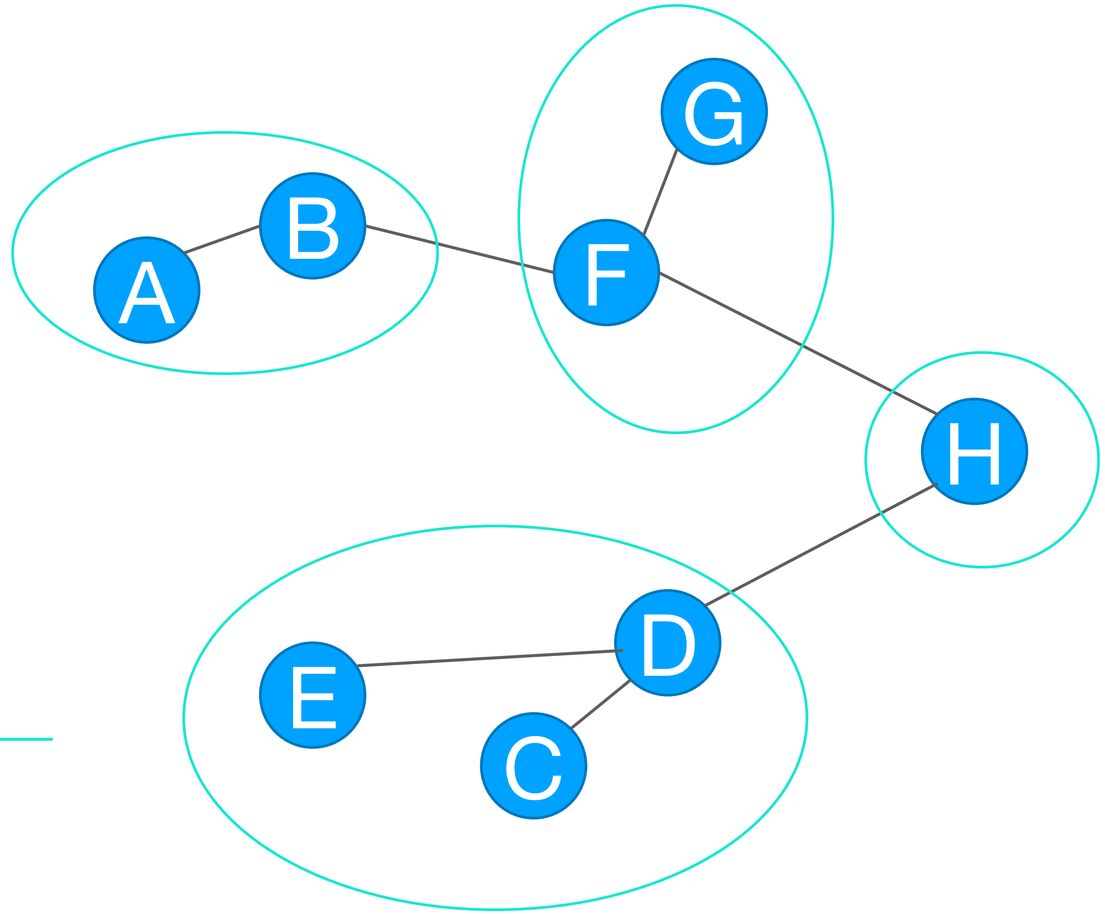




Single Linkage

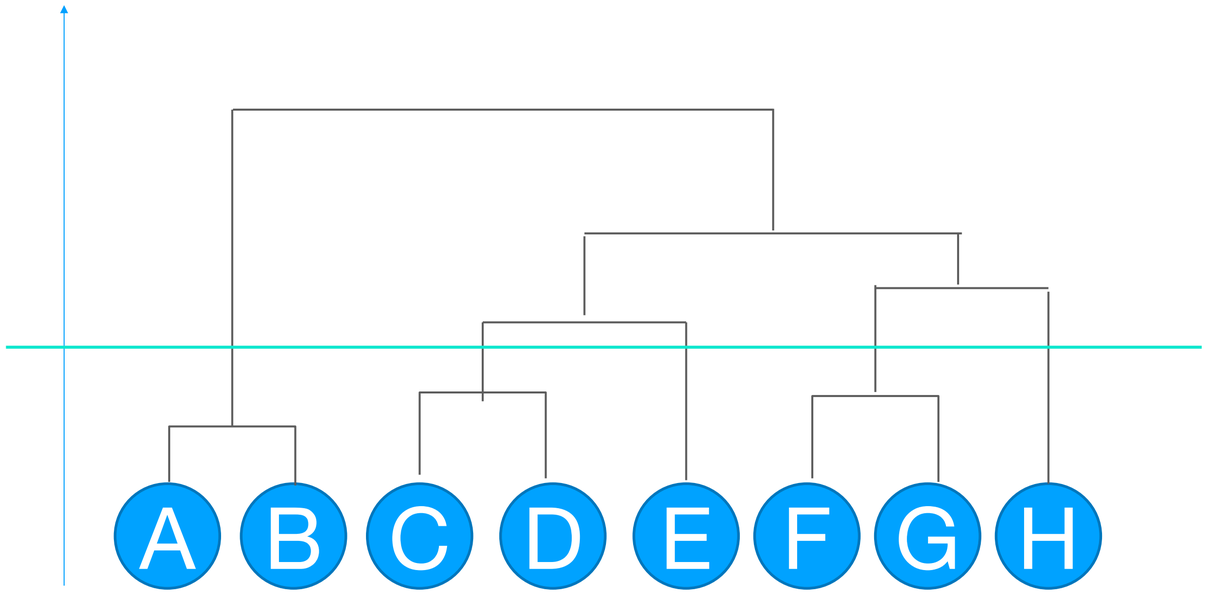
distance between the clusters that got merged at that step

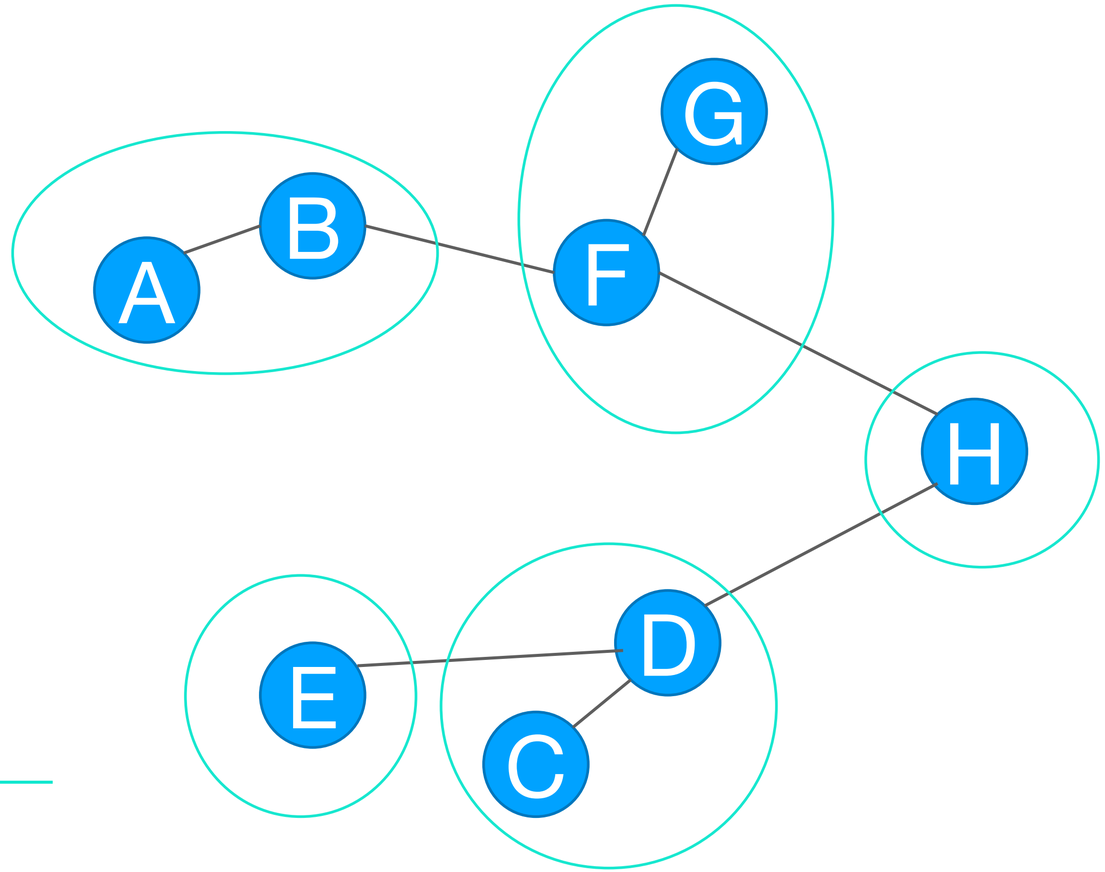




Single Linkage

distance between the clusters that got merged at that step

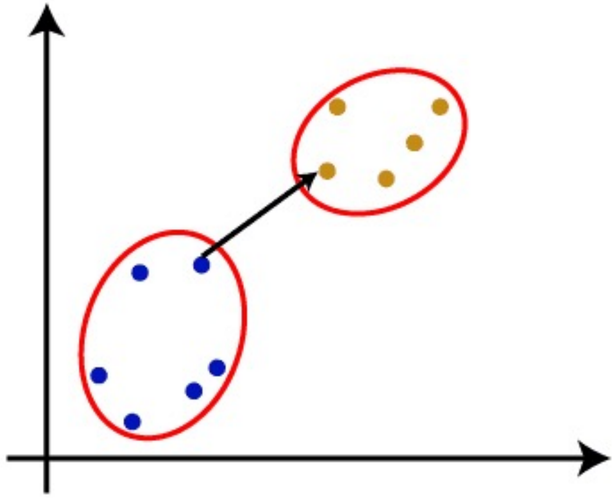




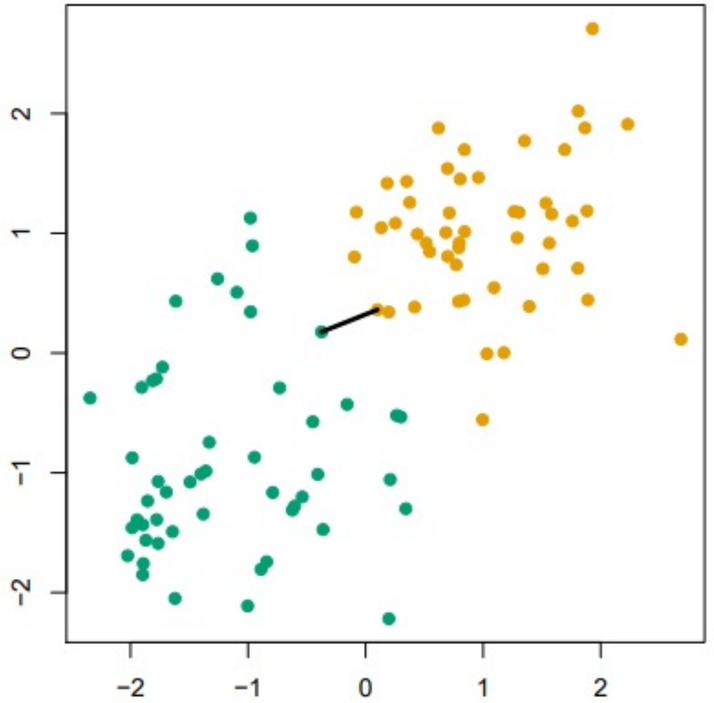
Single Linkage

In single linkage (i.e., nearest-neighbor linkage), the dissimilarity between G, H is the smallest dissimilarity between two points in different groups:

$$d_{\text{single}}(G, H) = \min_{i \in G,\, j \in H} d(x_i, x_j)$$



Example (dissimilarities d_{ij} are distances, groups are marked by colors): single linkage score $d_{\text{single}}(G, H)$ is the distance of the closest pair

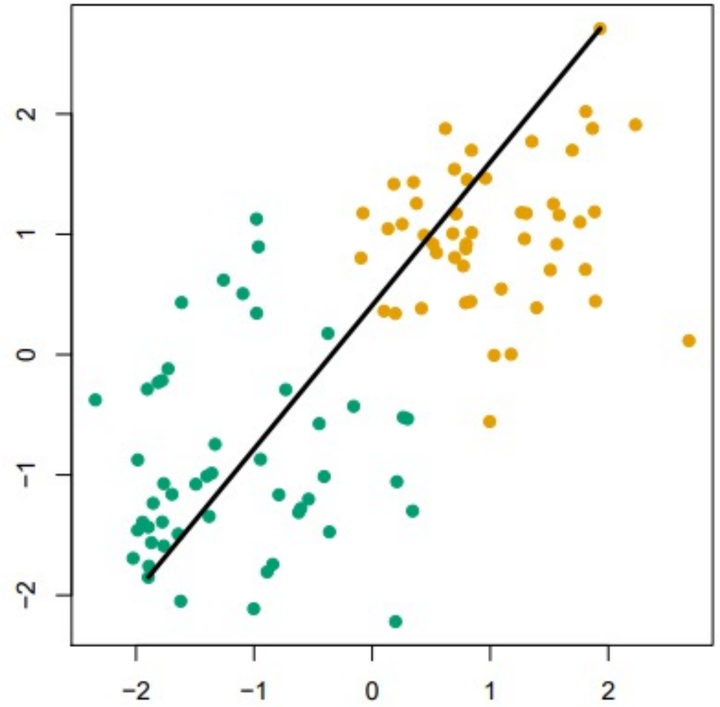


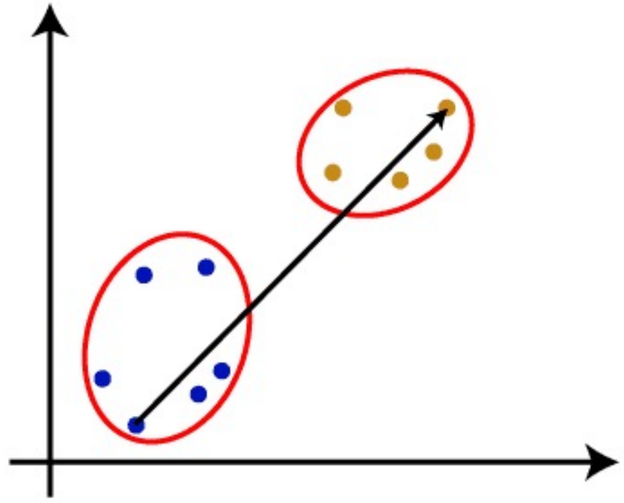
Complete Linkage

In complete linkage (i.e., furthest-neighbor linkage), dissimilarity between G, H is the largest dissimilarity between two points in different groups:

$$d_{\text{complete}}(G, H) = \max_{i \in G,\, j \in H} d(x_i, x_j)$$

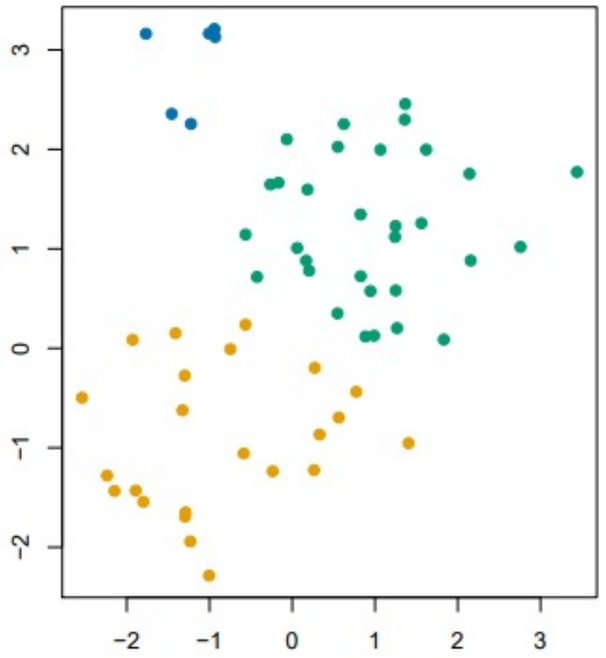
Example (dissimilarities d_{ij} are distances, groups are marked by colors): complete linkage score $d_{\text{complete}}(G, H)$ is the distance of the furthest pair

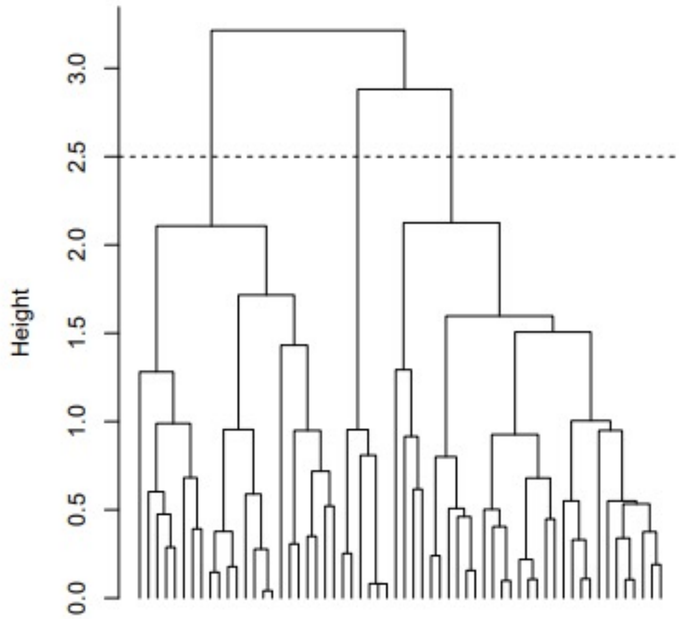


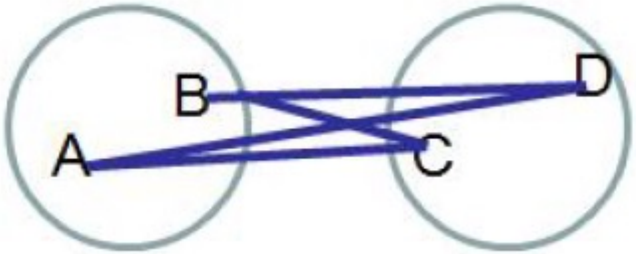


Average Linkage

Same data as before. Cutting the tree at $h = 2.5$ gives clustering assignments marked by the colors







(c)

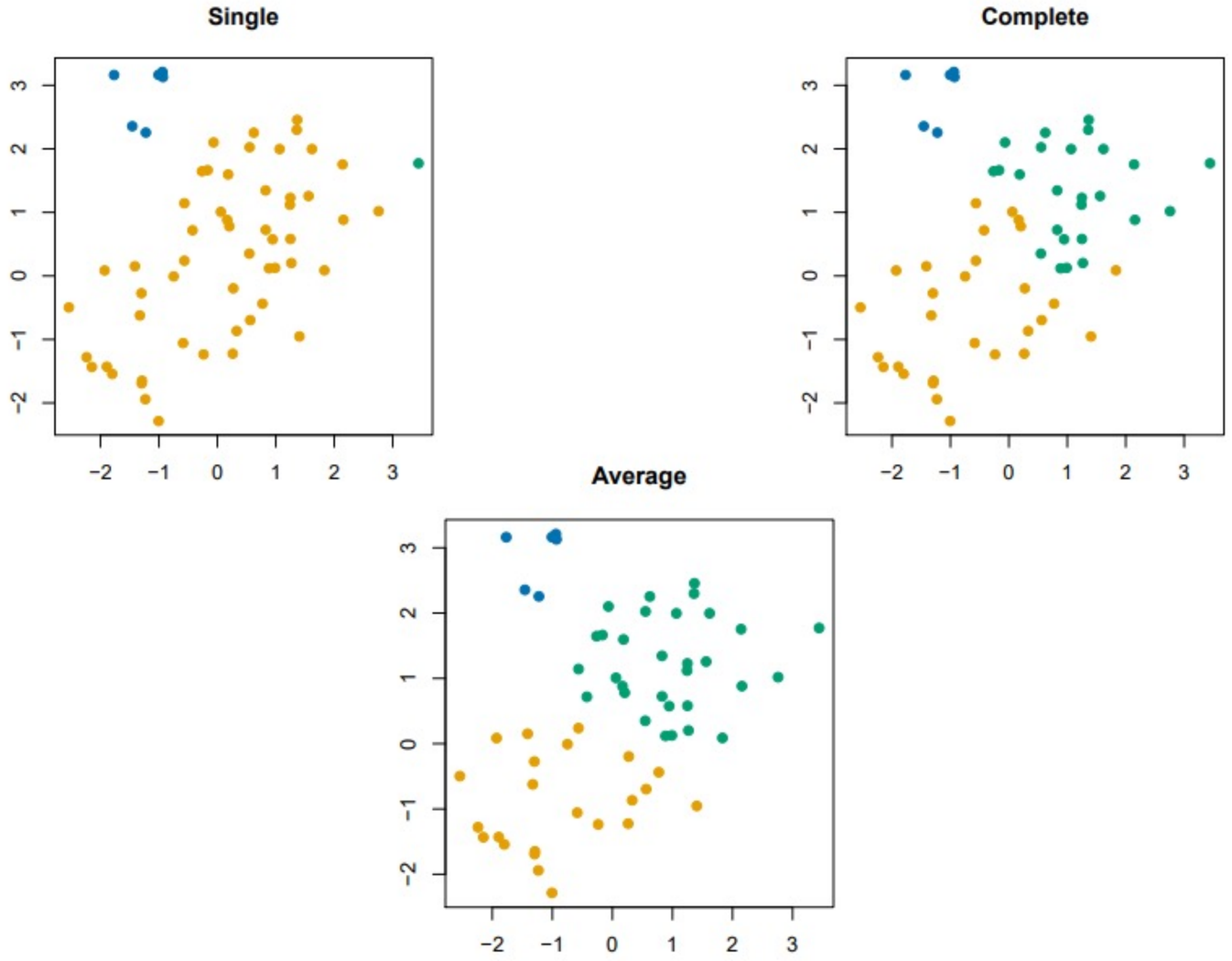
Shortcoming of Single and Complete Linkage

Single and complete linkage have some practical problems:

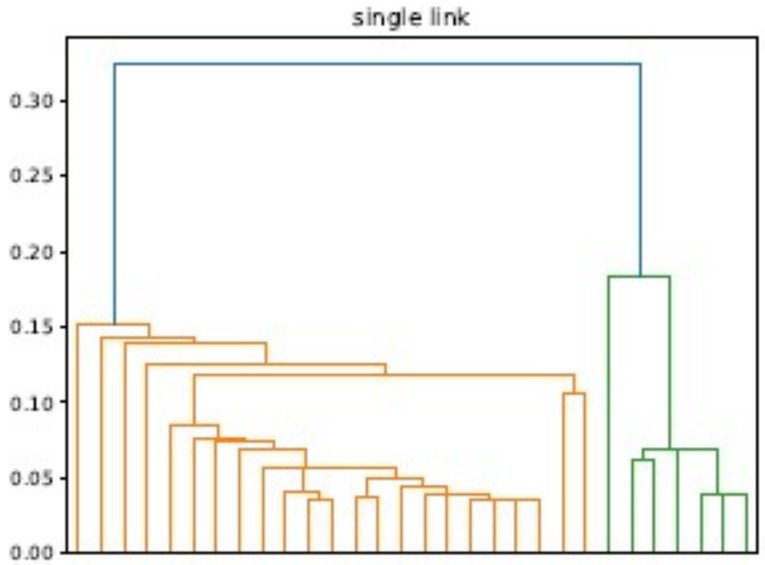
- Single linkage suffers from chaining.
  - In order to merge two groups, only need one pair of points to be close, irrespective of all others. Therefore *clusters can be too spread out*, and not compact enough.
- Complete linkage avoids chaining, but suffers from crowding.
  - Because its score is based on the worst-case dissimilarity between pairs, *a point can be closer to points in other clusters than to points in its own cluster*. Clusters are compact, but not far enough apart.

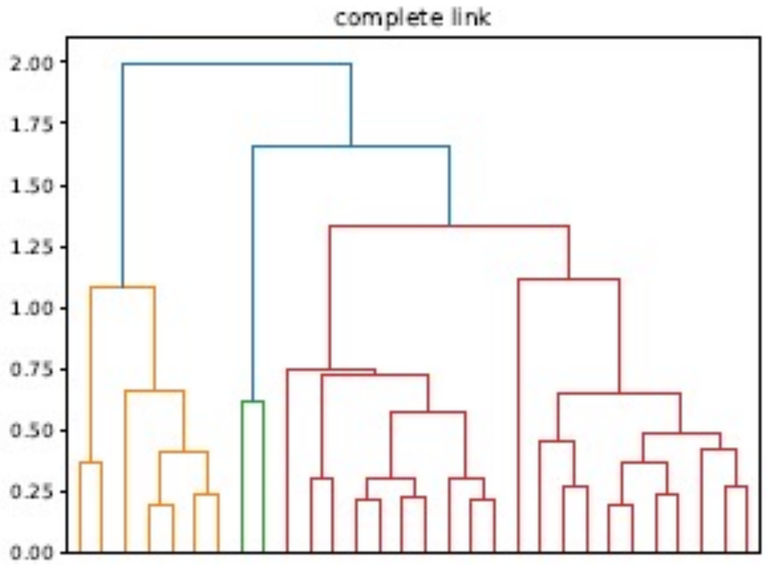
Average linkage tries to strike a balance. It uses average pairwise dissimilarity, so clusters tend to be relatively compact and relatively far apart

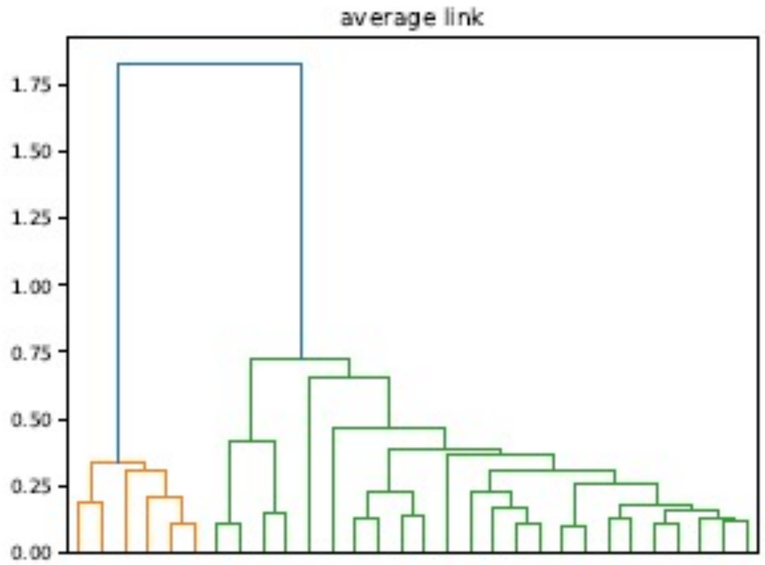
Shortcoming of Single and Complete Linkage



Shortcoming of Single and Complete Linkage







Dissimilarity measures

- The choice of linkage can greatly affect the structure and quality of the resulting clusters
- The choice of dissimilarity (equivalently, similarity) measure is *arguably* even more important
- To come up with a similarity measure, you may need to think carefully and use your intuition about what it means for two observations to be similar. E.g.,
  - What does it mean for two people to have similar *purchasing behaviour*?
  - What does it mean for two people to have similar music listening habits?
- You can apply hierarchical clustering to any similarity measure $s(x_i, x_j)$ you come up with. The difficult part is coming up with a good similarity measure in the first place.

Dissimilarity measures

Name of function	Mathematical formula		Comment
Minkowski Distance	$dis(X,Y)=(\sum_{i=1}^{d}\lvert x_i-y_i\rvert^{q})^{\frac{1}{q}}$	(1)	q is positive real number
Euclidean Distance	$dis(X,Y)=\sqrt{\sum_{i=1}^{d}(x_i-y_i)^2}$	(2)	equals to minkowski diatance where q=2
Average distance	$dis(X,Y)=\sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i-y_i)^2}$	(3)	is modified version of Euclidean distance
Manhattan Distance	$dis(X,Y)=\sum_{i=1}^{d}\lvert x_i-y_i\rvert$	(4)	equals to minkowski diatance where q=1, sensitive to outliers
Maximum Distance	$dis(X,Y)=\max_{i=1}^{d}\lvert x_i-y_i\rvert$	(5)	equals to minkowski diatance where $q\rightarrow\infty$
Pearson correlation	$dis(X,Y)=\frac{1}{2}(1-\frac{\sum_{i=1}^{d}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{d}(xi-\bar{x})^2\sum_{i=1}^{d}(yi-\bar{y})^2}})$	(6)	used in gene expression clustering
Mahalanobis distance	$dis(X,Y)=(x-y)\ S^{-1}(x-y)^T$	(7)	used in hyper ellipsoid clusters, S is covariance matrix
Cosine similarity	$sim(X,Y)=\cos\alpha=\frac{x^T y}{\sqrt{\sum_{i=1}^{d}x_i^2}\sqrt{\sum_{i=1}^{d}y_i^2}}$	(8)	used in document clustering
Chord distance	$dis(X,Y)=\sqrt{2-2\frac{\sum_{i=1}^{d}x_i y_i}{\sqrt{\sum_{i=1}^{d}x_i^2}\sqrt{\sum_{i=1}^{d}y_i^2}}}$	(9)	used for normalized attributes

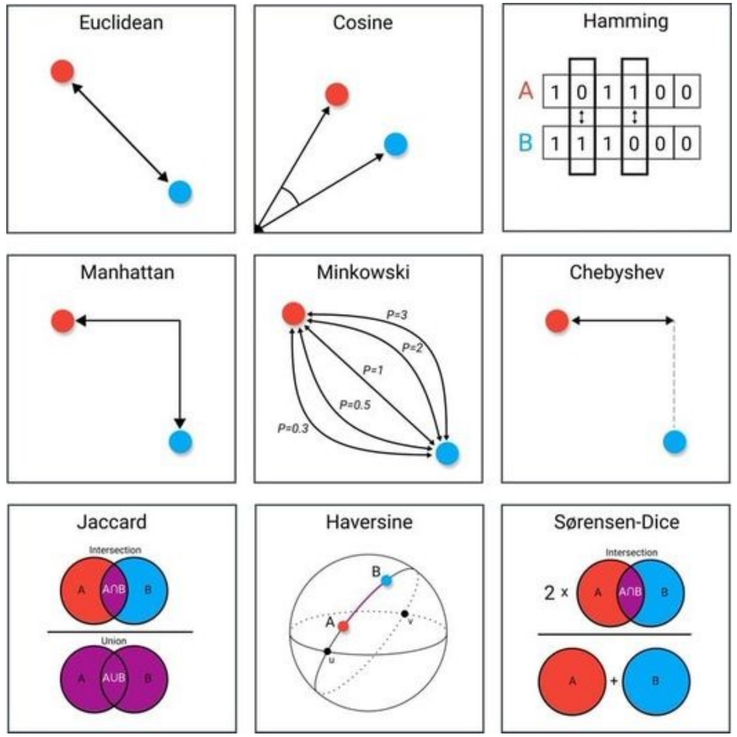


Image from
https://www.facebook.com/photo/?fbid=10224815831330636&set=a.10200946882861842
https://www.researchgate.net/publication/328379237_Homogeneous_Densities_Clustering_Algorithm/figures?lo=1&utm_source=google&utm_medium=organic

Kmean vs Hierarchy

- K-means:
  - ☺ Low memory usage
  - ☺ Essentially $O(n)$ compute time
  - ☹ Results are sensitive to random initialization
  - ☹ Number of clusters is pre-defined
  - ?? Awkward with categorical variables

- Hierarchical clustering:
  - ☺ Deterministic algorithm
  - ☺ Dendrogram shows us clusterings for various choices of K
  - ☺ Requires only a distance matrix, quantifying how dissimilar observations are from one another
    - We can use a dissimilarity measure that gracefully handles categorical variables, missing values, etc
  - ☹ Memory-heavy, more computationally intensive than K-means

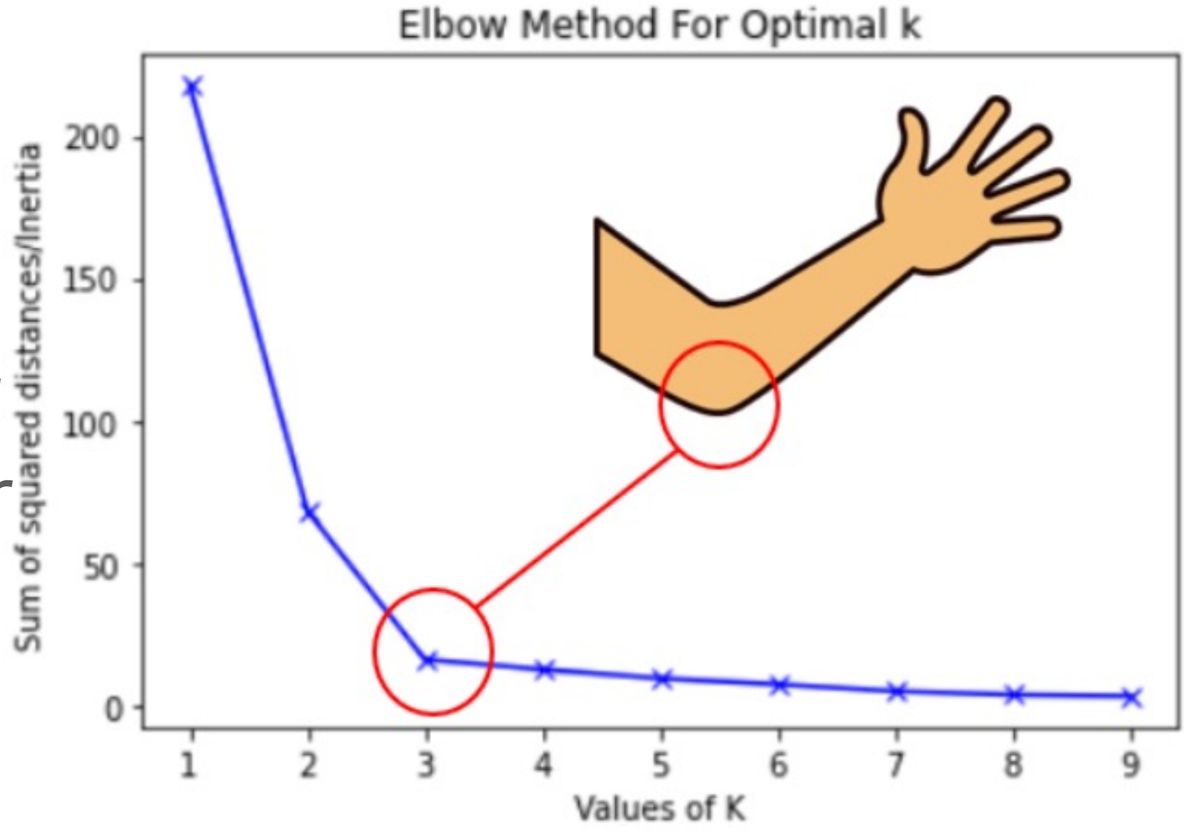
Choosing the number of K

- Elbow curve method
- Silhouette analysis

Elbow curve method

Basic idea behind partitioning methods, such as k-means clustering, is to define clusters such that the total intra-cluster variation [or total within-cluster sum of square (WSS)] is minimized.

Y-axis plots the total WSS.



Line plot between K and inertia

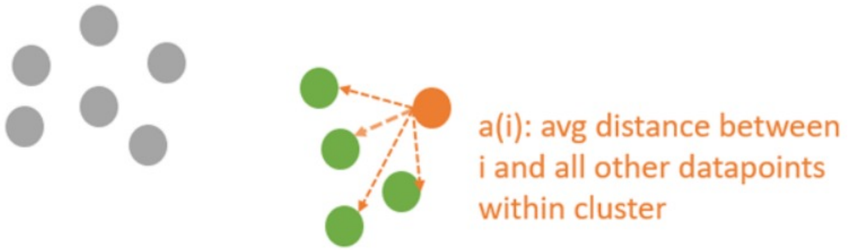
Image from https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

Silhouette analysis

The silhouette coefficient is a measure of how similar a data point is within-cluster (cohesion) compared to other

The equation for calculating the silhouette coefficient for a particular data point:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$



- S(i) is the silhouette coefficient of the data point i.
- a(i) is the average distance between i and all the other data points in the cluster to which i belongs.
- b(i) is the average distance from i to all clusters to which i does not belong.

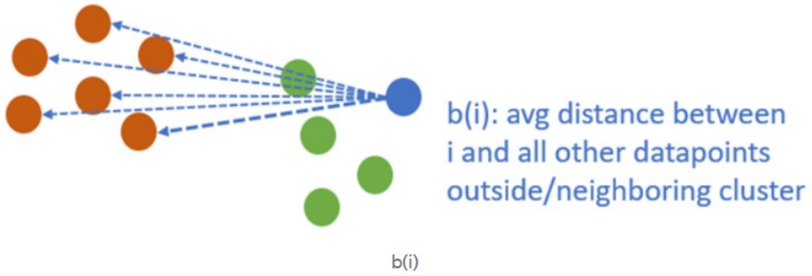


Image from https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

Silhouette analysis

- The value of the silhouette coefficient is between [-1, 1].
- A score of 1 denotes the best meaning that the data point i is very compact within the cluster to which it belongs and far away from the other clusters.
- The worst value is -1. Values near 0 denote overlapping clusters.

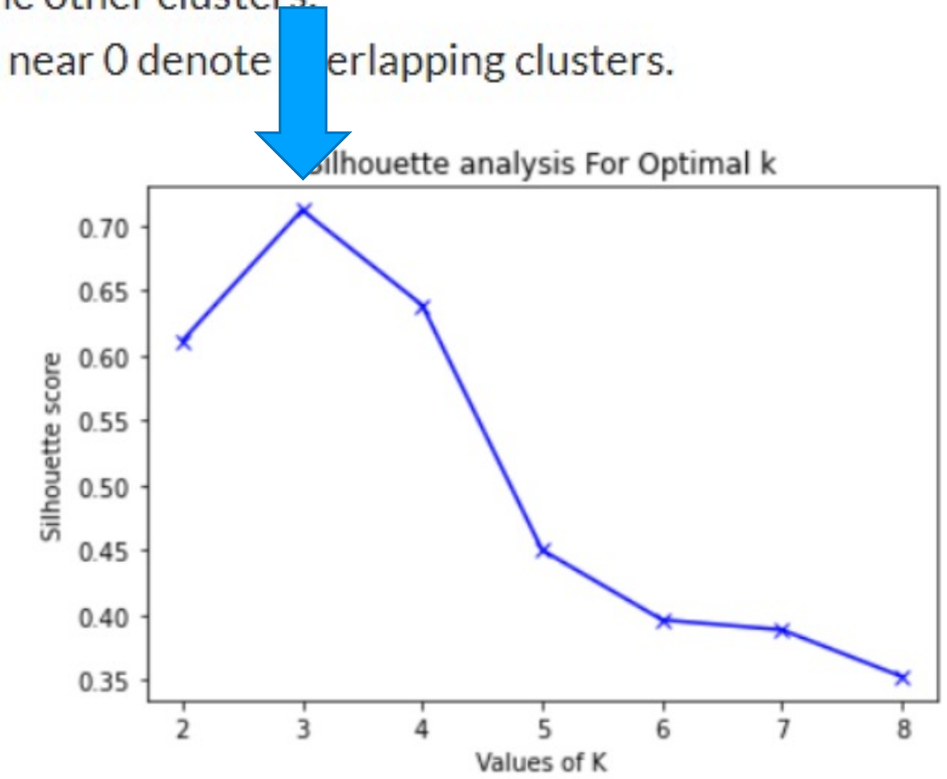
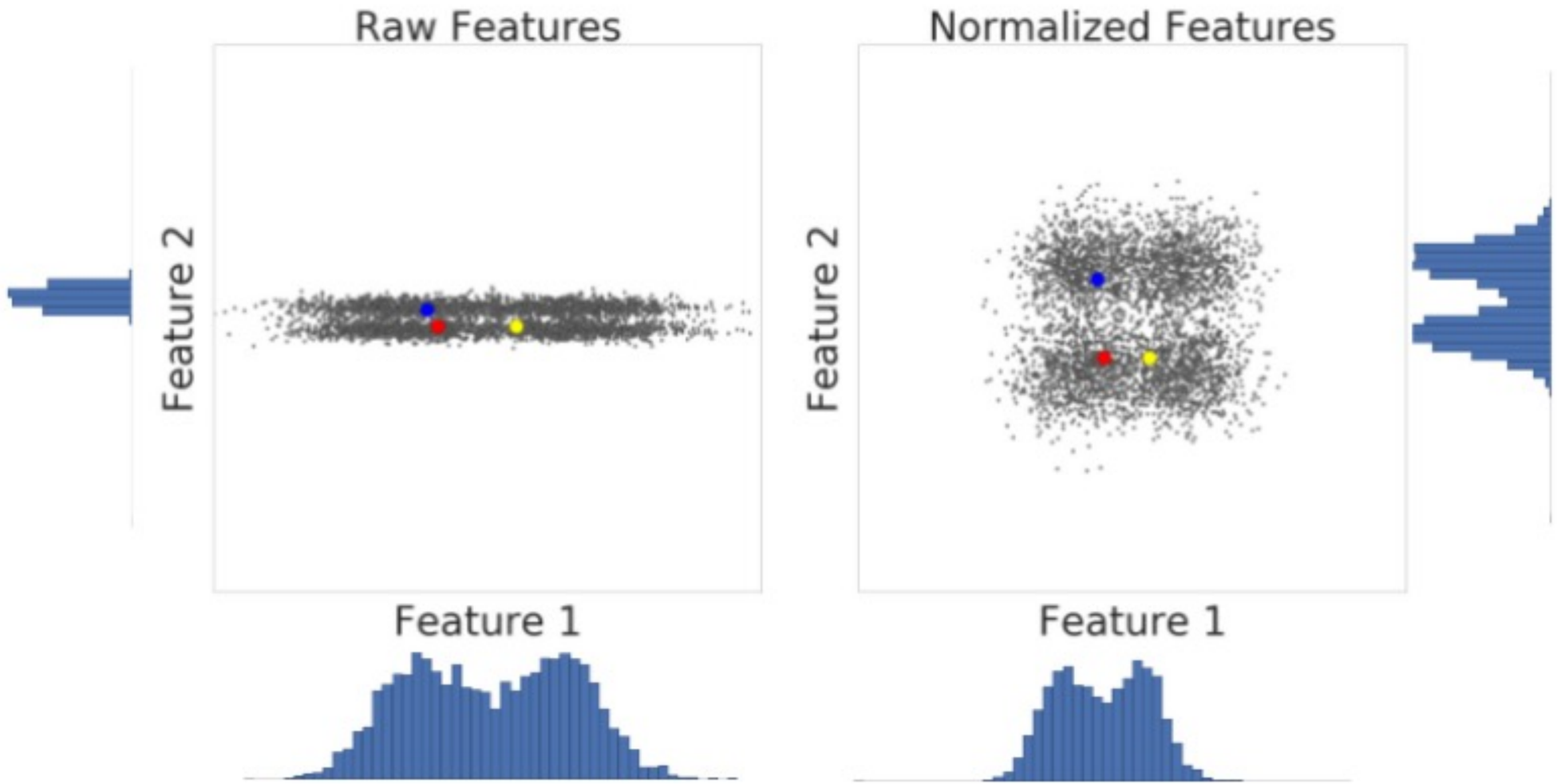


Image from https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/

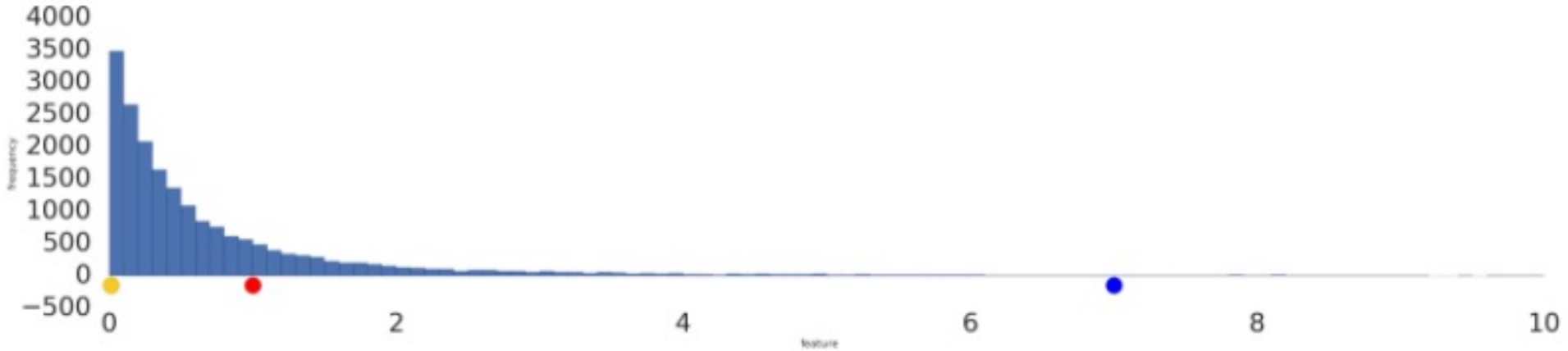
Data Preprocessing before Clustering

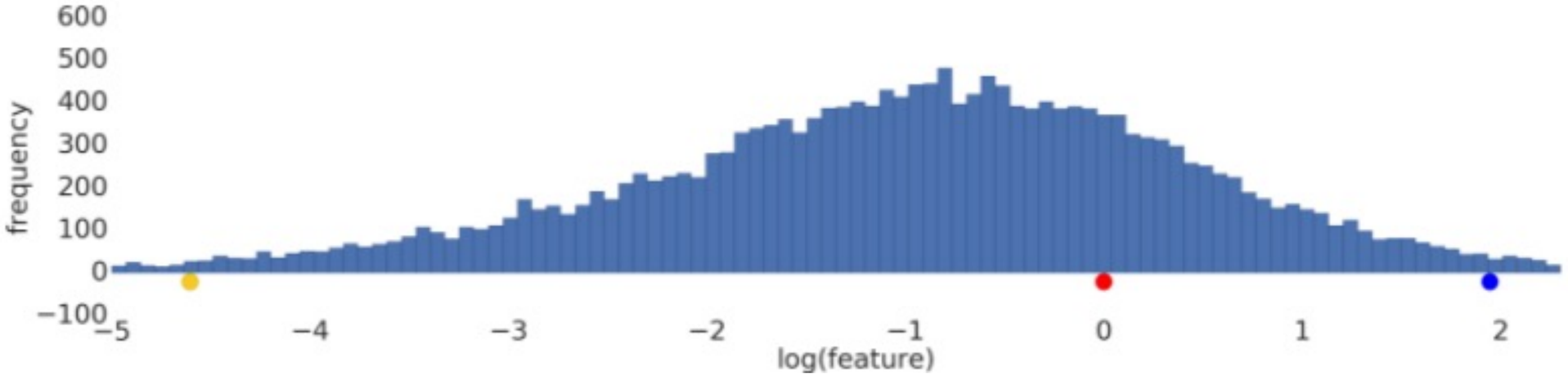
- Normalization – if your data has normal distribution
- Log-transform – if your data has power distribution
- Quantile normalization - otherwise

Normalization



Log-transform





Quantile


2500
2000
1500
1000
500
0
−500
frequency
0
2
4
6
8
10
12
feature


3000
2500
2000
1500
1000
500
0
−500
frequency

Credit to the following sources:

1. https://www.ini.rub.de/PEOPLE/wiskott/Teaching/Material/Clustering-LectureNotesPublicVideoAnnotated.pdf

2. https://www.aryabhattacollege.ac.in/samplepaper/637202999310007925DataMining1.pdf

3. https://www.andrew.cmu.edu/user/achoulde/95791/lectures/lecture07/lecture07_95791.pdf

4. https://inblog.in/Hierarchical-Clustering-and-Density-Based-Spatial-Clustering-of-Applications-with-Noise-DBSCAN-DbGMBttoOD

5. https://www.robots.ox.ac.uk/~az/lectures/ml/2011/lect8.pdf