Clustering

Lecture 8:

Clustering: Hard and Soft Clustering

(K-Means, Gaussian Mixture Models)

ECE/CS 498 DS

Professor Ravi K. Iyer

Department of Electrical and Computer Engineering

University of Illinois

Announcements

- HW 1 due **tonight Feb 17th @ 11:59 PM** on Compass2G
  - To be done individually
- MP 1 final checkpoint due Thu Feb 20th @ 11:59 PM on Compass2G
  - One submission per group, consisting of
    - Single ipynb for all tasks
    - Single PDF with results for all tasks (template has been provided)
  - **Presentation (2/21) signup link is live**: https://docs.google.com/spreadsheets/d/14braJUAaud3y4kcg6I1N1fTRBxZBis6sutxqxTpWsKU/edit#gid=0
- Discuss section this week (2/21) is cancelled due to MP 1 presentations
- **For Grad Students**: Final project information has been released
  - Initial project ideas due Wed Feb 26
- Midterm exam: **Wed March 11th**

Looking for patterns and relationships in the data

- Clustering
  - Finding groupings in the data
  - Groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters
- Linear and non-linear regression
  - Finding relationships between different variables/features in the data
- Principal component analysis
  - Rotating the axes to simplify data visualization/description
  - Dimensionality reduction

An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters
- “Similar” data points are more likely to belong to the same cluster compared to “dissimilar” data points

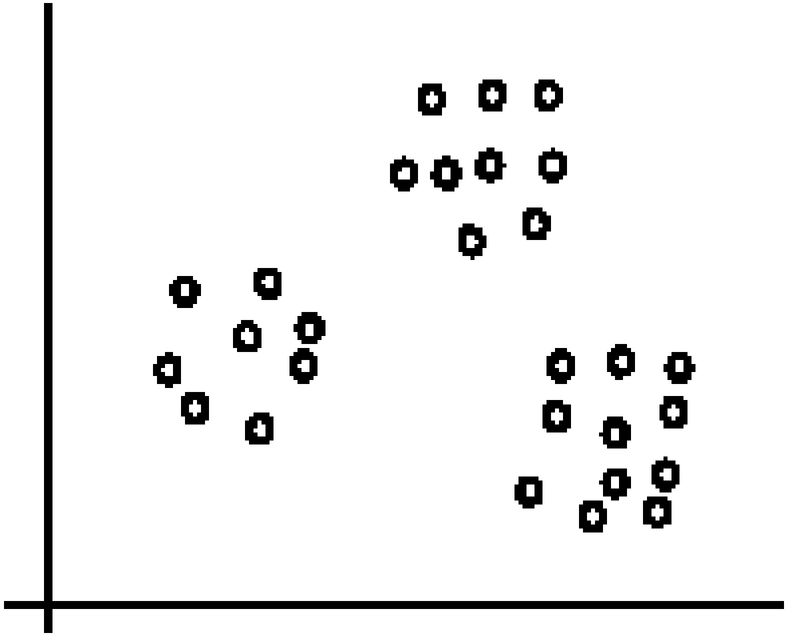
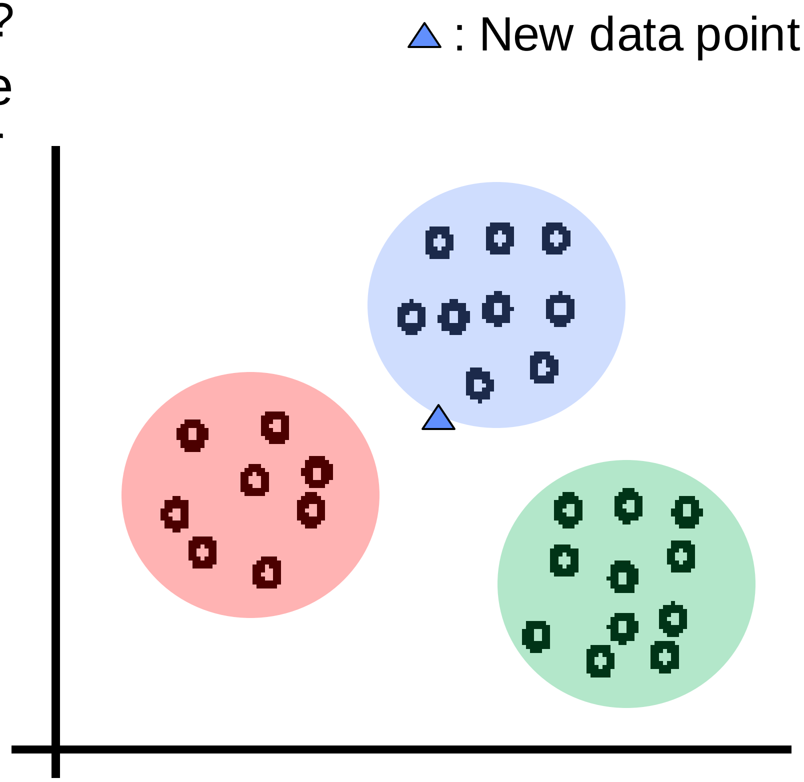


Image source: CS583, Bing Liu, UIC

Aspects of Clustering Algorithms

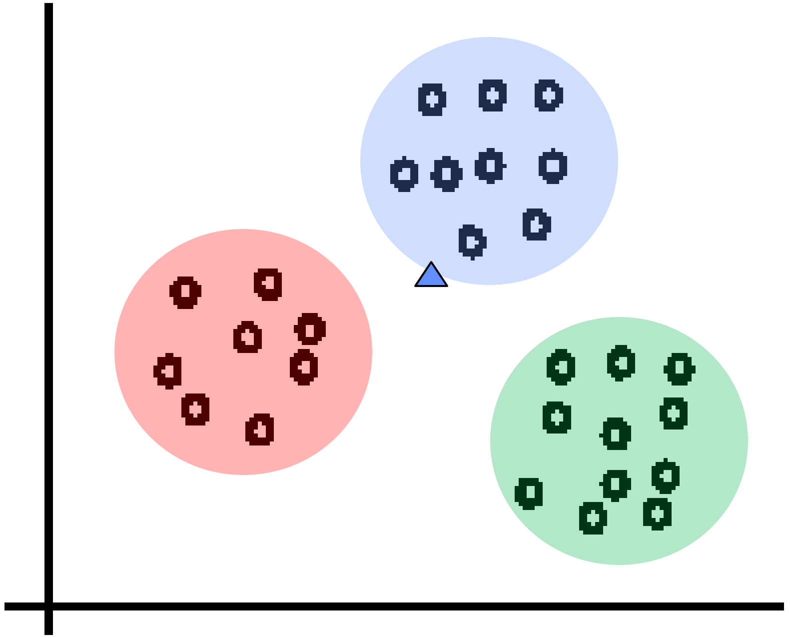
- Question:
  - How to find the clusters?
  - How to assign a point to a cluster?
- Want clusters of instances that are similar to each other but dissimilar to others

- Examples of methods:
  - Hard clustering:
    - K-means clustering
    - Hierarchical clustering
  - Soft clustering
    - Gaussian Mixture Models



Aspects of Clustering: Distance function

- Question:
  - How similar is a point to a cluster or to the other points in a cluster?
- Need a similarity measure/metric that measures how close a point is to a cluster or to another point



Aspects of Clustering: Distance function

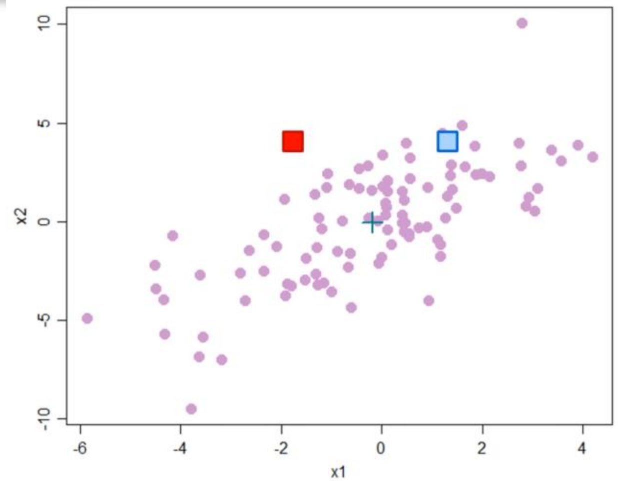
- **Euclidean distance** (compact isolated clusters)

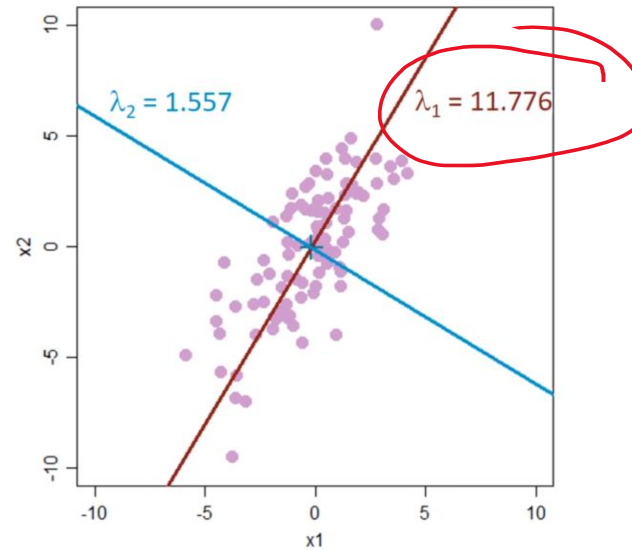
$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$$

  - Simple to understand
  - Limited interpretation when features are correlated
    - Both red and blue box in figure are equidistant from mean ('+') in terms of Euclidean distance
    - **To correctly recognize red box as outlier, its distance from mean should reflect its higher/more extreme**
- **Mahalanobis distance** alleviates problems with correlation using covariance matrix Σ

$$[d(\boldsymbol{x}_i, \boldsymbol{x}_j)]^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)\Sigma^{-1}(\boldsymbol{x}_i - \boldsymbol{x}_j)^T$$

  - Intuitively, it
    - (i) transforms features to remove any covariance
    - (ii) rescales each feature so that it has variance of 1
    - (iii) calculates Euclidean distance
  - Distance is reported in **in units of standard deviations**
  - Requires computation of the Σ^{-1} matrix

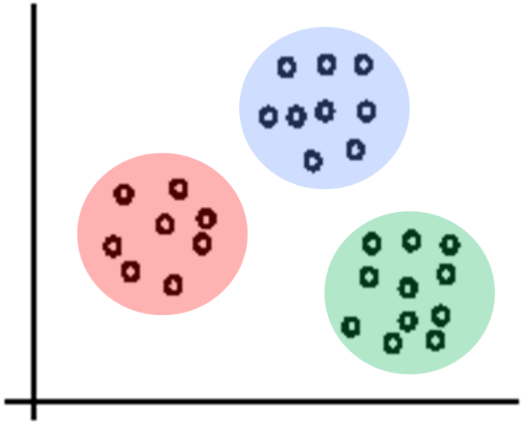




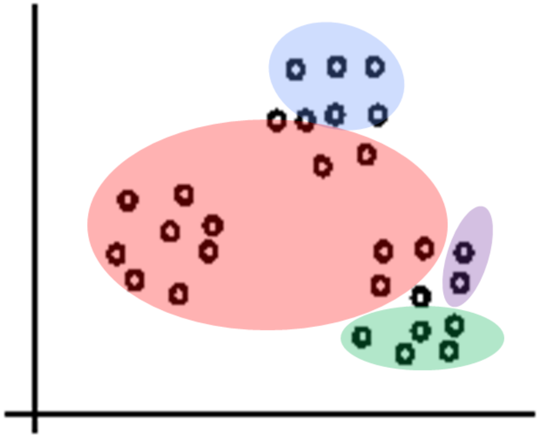
https://www.youtube.com/watch?v=spNpfmWZBmg

Aspects of Clustering: Cluster quality

- Clustering quality:
  - How "good" are the clusters?
- Examples of criteria:
  - Inter-cluster distance $\Rightarrow$ maximized
  - Intra-cluster distance $\Rightarrow$ minimized
- The quality of a clustering result depends on the
  - clustering algorithm
  - distance function (Euclidean, Mahalanobis, etc.)
  - data

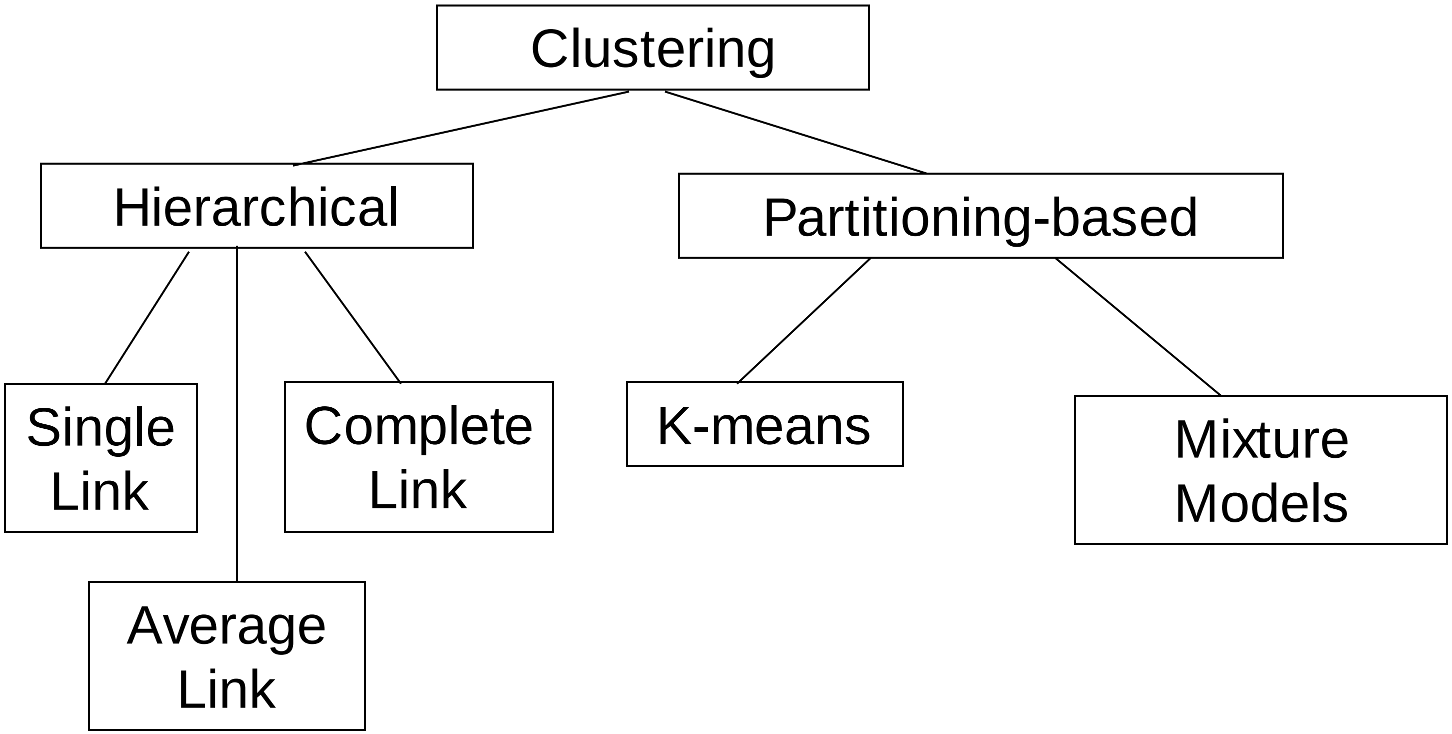


"Good" clusters



"Not-so-good" clusters

Clustering Techniques



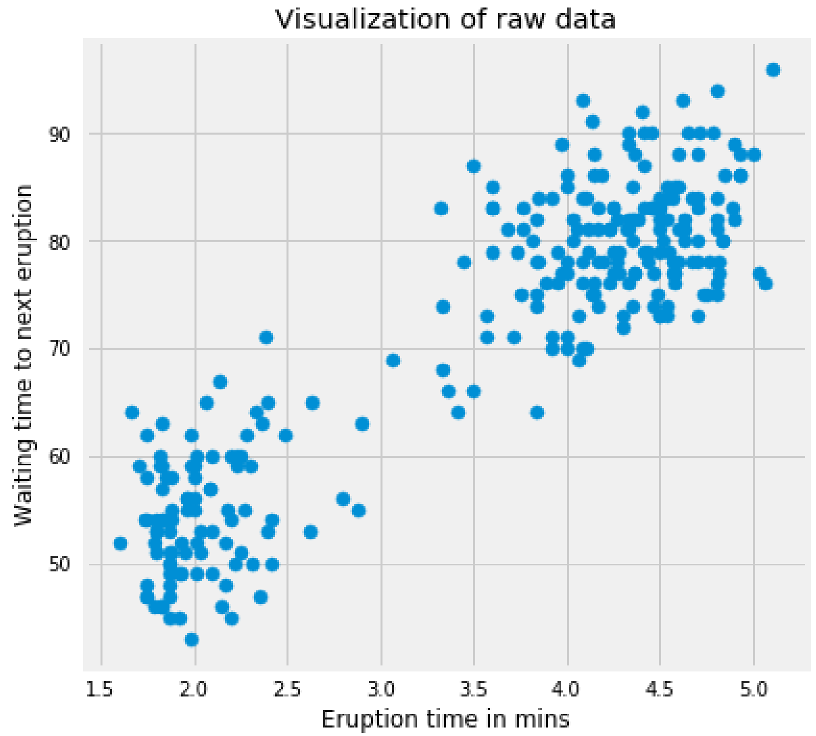
Clustering Example: Geyser Eruptions

- **Geysers** are vents in the Earth's crust that periodically release bursts of hot water and steam
- "Old Faithful" (pictured), found at Yellowstone National Park, is one of the world's most famous geysers[1]
- 2-D observations about Old Faithful eruptions were made, each consisting of the following information[2]:
  - **Eruption length** (in minutes)
  - **Time until next eruption** (in minutes)

[1] https://yellowstone.net/geysers/old-faithful/
[2] https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

Clustering Example: Geyser Eruptions

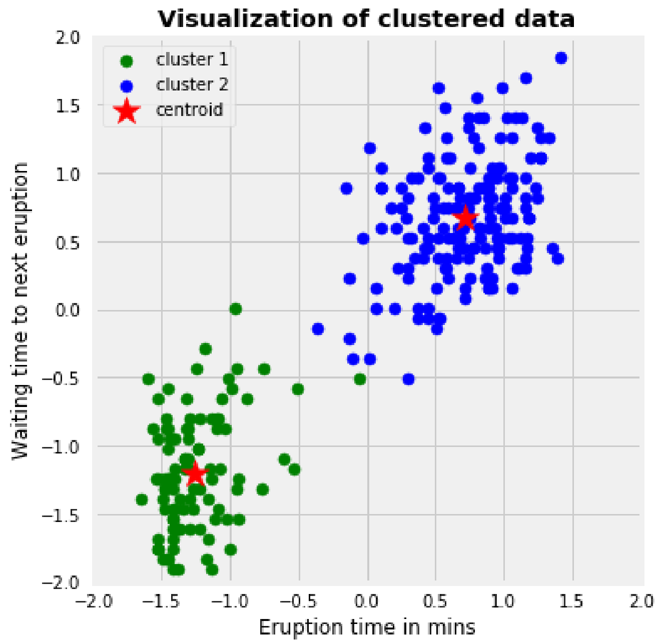
- After plotting the raw data below, **it seems like there are two ”clusters”,** or natural groupings of the data



https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

Clustering Example: Geyser Eruptions

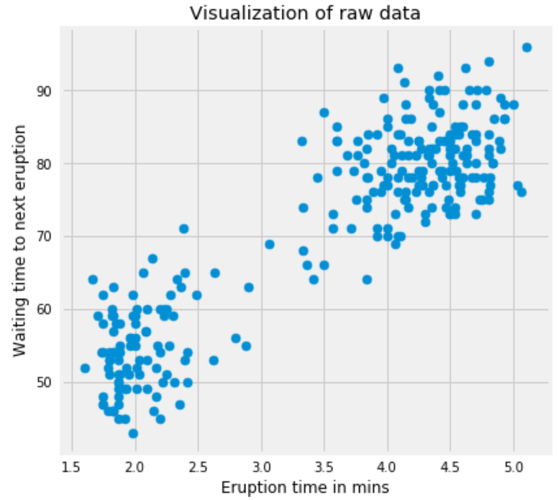
- With a simple clustering algorithm called ***k*-means** (which we will learn about next), we assign each standardized data point into one of the two clusters
  - Each cluster has a geometric center of mass called a *centroid*

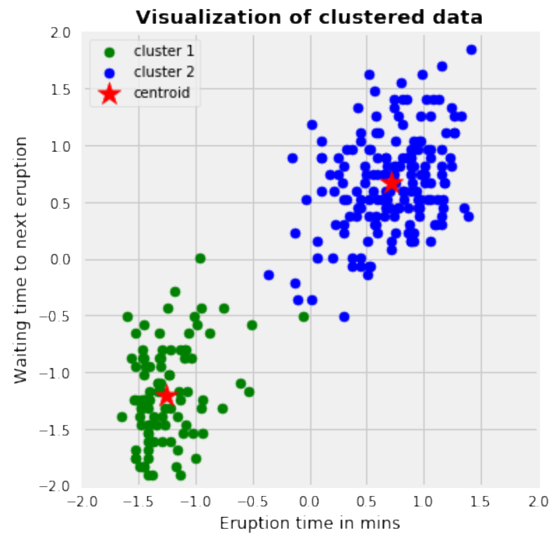


https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a

Clustering Example: Geyser Eruptions

- From this clustering analysis, **we can derive some basic insights** about the behavior of Old Faithful
- A cluster towards the top right of the chart suggests that if an eruption is longer, then we wait longer until the next eruption happens
  - This make sense geologically as we need to wait longer for the additional steam/water to build up
- Similarly, a cluster towards the bottom left of the chart suggests that shorter eruptions imply shorter intervals between eruptions





K-Means Clustering

K-means clustering

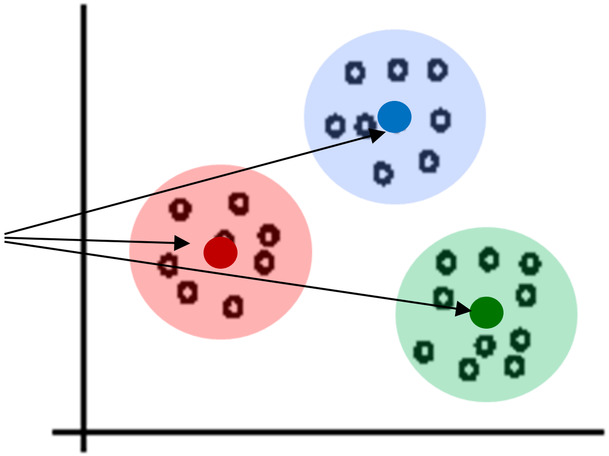
- K-means is a partition-based clustering algorithm

- Let the set of data points (or instances) *D* be

  $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_n\}$,

  where $\boldsymbol{x}_i = (x_{i_1}, x_{i_2}, \dots, x_{i_d})$ is a vector in a real-valued space $\mathbb{R}^d$, and d is the number of feature (dimensions) in the data

- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**
  - *k* is specified by the user



K-means clustering with $k = 3$

K-means algorithm

Given *k*, the *k-means* algorithm works as follows:

1) Randomly choose *k* data points (seeds) to be the initial centroids i.e., cluster centers
2) Assign each data point to the closest centroid
3) Re-compute the centroids using the current cluster memberships.
4) If a convergence criterion is not met, go to 2).

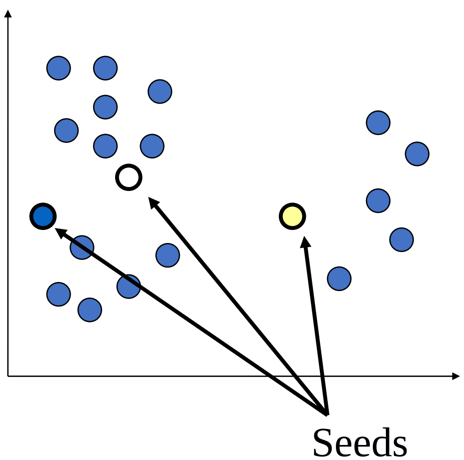
Stopping/Convergence criterion

1. No (or minimal) re-assignments of data points to different clusters,
2. No (or minimal) change of centroids, or
3. Minimal decrease in the **sum of squared error** (SSE),

$$SSE = \sum_{j=1}^{k} \sum_{\boldsymbol{x} \in C_j} dist(\boldsymbol{x}, \boldsymbol{m_j})^2 \qquad \boldsymbol{m_j} = \frac{1}{n_j} \sum_{\boldsymbol{x} \in C_j} \boldsymbol{x}$$

- C_j is the j^{th} cluster, $\boldsymbol{m}_j$ is the centroid of cluster C_j (the mean of all the data points belonging to C_j)
- n_j is the number of points in cluster C_j
- $dist(\boldsymbol{x}, \boldsymbol{m}_j)$ is the distance between data point $\boldsymbol{x}$ and centroid $\boldsymbol{m}_j$.

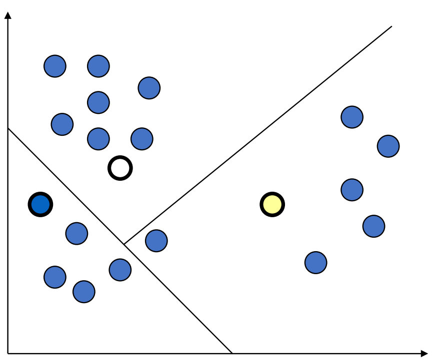
K-Means Example



(a)

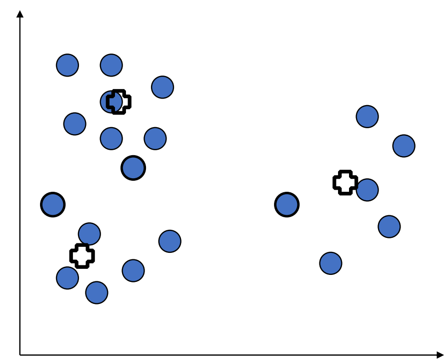
Predetermined number of clusters

Start with seed clusters of one element



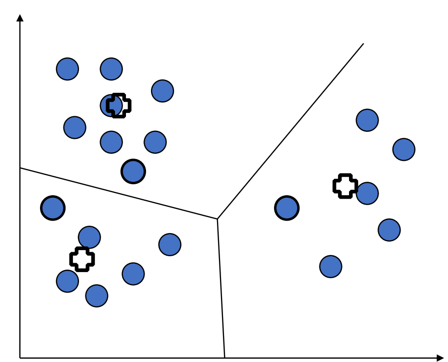
(b)

Assign Instances to Clusters



(c)

Find new centroids



(d)

Assign instances to new cluster

Why k-means is Popular

- **Strengths:**
  - Simple: Easy to understand and to implement
  - Efficient: Time complexity = $O(tkn)$, where

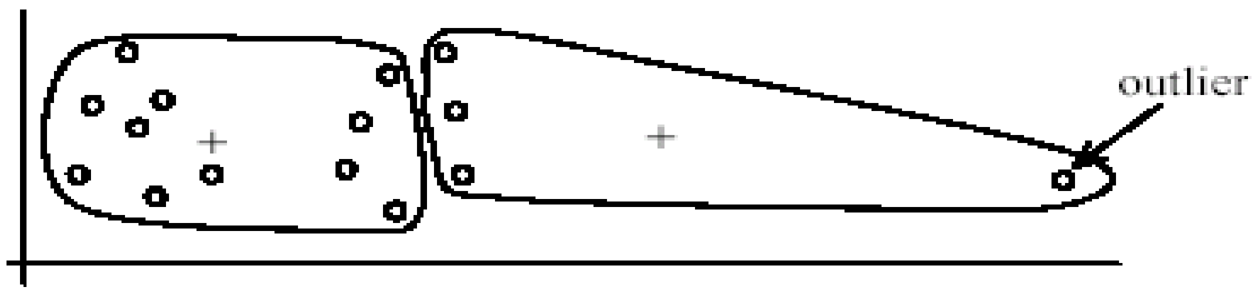
    n is the number of data points, k is the number of clusters, and t is the number of iterations.
    - In a given iteration, assigning each of the n data points to a cluster requires calculating their distance to each of k centroids – *$O(kn)$ work*
    - Over t trials, this is $O(tkn)$ work
  - Since both k and t are generally small, k-means is considered a linear algorithm

- Note: The algorithm can converge to a local optimum if SSE is used. The global optimum is difficult to find due to complexity.

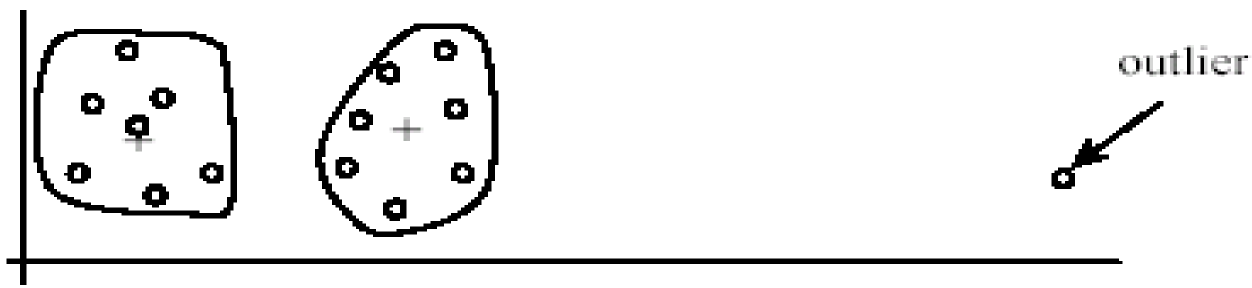
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined
  - For categorical data, *k*-mode - the centroid is represented by most frequent values
- Can be sensitive to seeds (choice of the initial k centroids)
- The user needs to specify *k*
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points
  - Outliers could be errors in the data recording or some special data points with very different values

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



(B): Ideal clusters

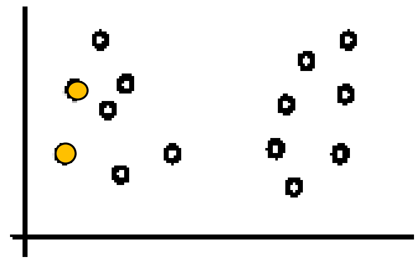
Weaknesses of k-means: To deal with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them

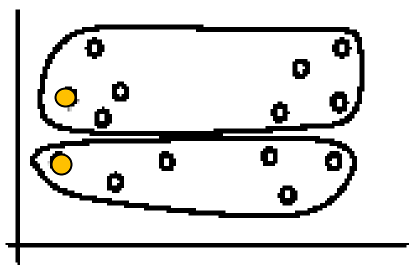
- Another method is to perform random sampling: Since sampling chooses a small subset of the data, the chance of selecting an outlier is small
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

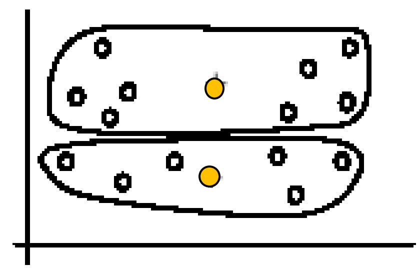
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



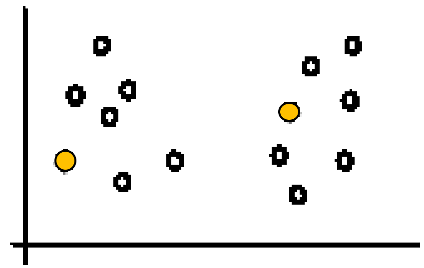
(B). Iteration 1



(C). Iteration 2

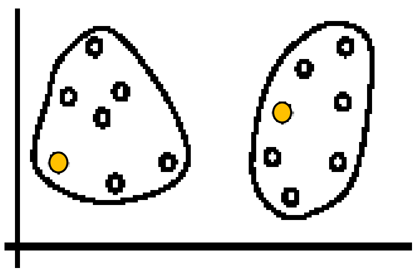
Weaknesses of k-means (cont …)

- Use different seeds: Good results

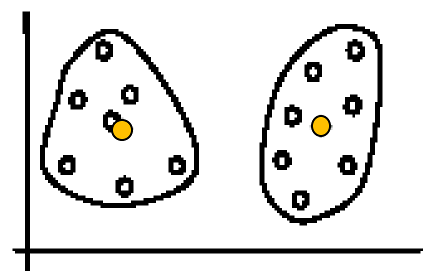


(A). Random selection of k seeds (centroids)

There are some methods to help choose good seeds



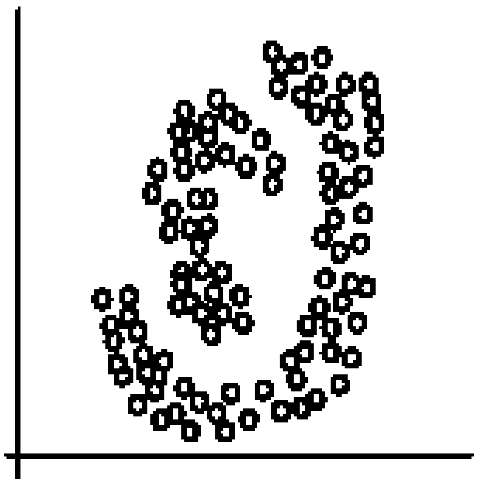
(B). Iteration 1



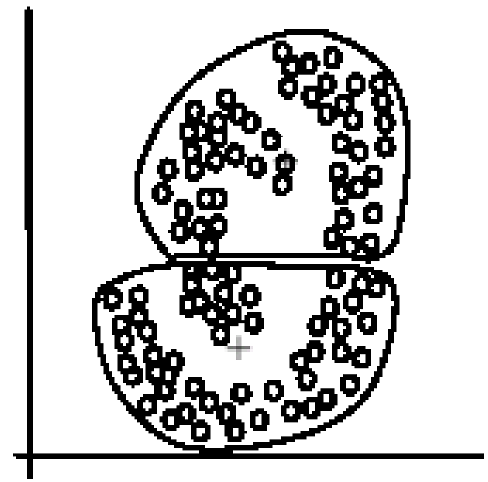
(C). Iteration 2

Weaknesses of k-means (cont …)

- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres)



(A): Two natural clusters



(B): *k*-means clusters

K-means summary

- Despite the weaknesses, *k*-means is a very useful algorithm due to its simplicity and efficiency
  - Other clustering algorithms have their own lists of weaknesses.
- No clear evidence that any other clustering algorithm performs better in general
  - Although they may be more suitable for some specific types of data or applications
- Comparing different clustering algorithms is a difficult task
  - Problem dependent insights are very useful

Gaussian Mixture Models

Expectation-Maximization: Motivation

- Clustering data points using MAP or maximum likelihood rules is very difficult when there are **latent (hidden / unobservable) variables**
  - Latent variables interact with the dataset but are not directly observed/known
  - In clustering, these latent variables are usually parameters of the clusters we are trying to determine (e.g. centroid locations in k-means, mean and standard deviation in Gaussian clustering)
- **Expectation Maximization** is an iterative solution to this problem
  - General procedure:
    - (1) Initialization Step: “guess” latent variables (e.g. cluster parameters)
    - (2) Expectation Step: optimize model to fit the data using the currently known latent variables
    - (3) Maximization Step: optimize the parameters using the current model
    - (4) Repeat steps (2)-(3) until convergence

https://machinelearningmastery.com/expectation-maximization-em-algorithm/

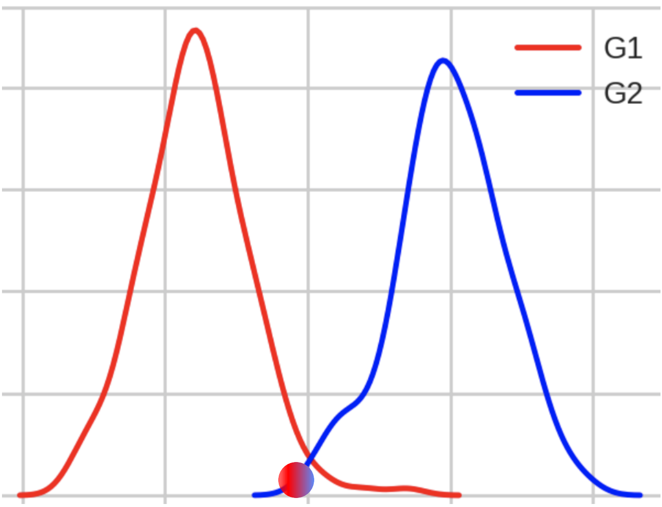
Soft Clustering: Mixture Model: Clusters may Overlap

Given:

- Data points/observations: $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots$

Model:

- There is a set of *K* probability distributions
  - Each distribution represents a cluster
  - Each distribution is described by certain parameters
  - Clusters may overlap
  - Find strengths of association between clusters and data instances
  - Discover the parameters of the distribution e.g. mean and variance
- Each data point is sampled from one of several distributions
  - $p(\boldsymbol{x_i}|b)$: Likelihood probability (density) that an instance $\boldsymbol{x_i}$ takes certain feature values given that it is from cluster b
  - $P(b|\boldsymbol{x_i})$: Posterior probability that an instance belongs to cluster b given that its features are $\boldsymbol{x_i}$



Problem:

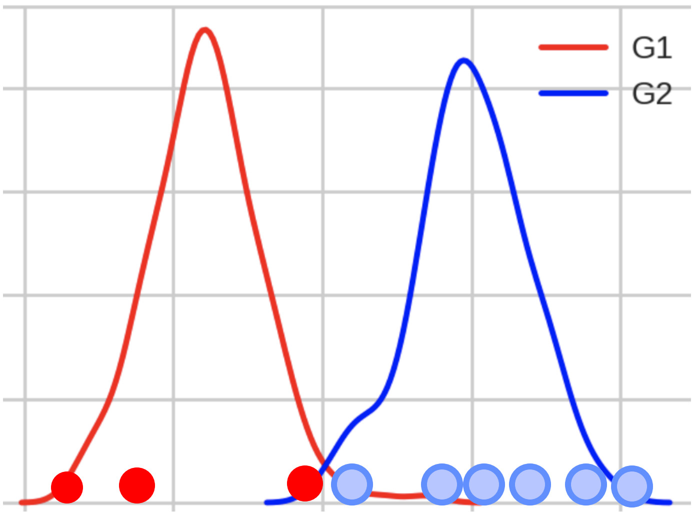
- Find parameters of the *K* distributions
- Find the posterior probabilities for each point

Expectation Maximization

- Automatically discover all the parameters for the K sources

GMM Example: Find parameters

- Observations: $x_1, x_2, \ldots, x_N$
  - Each observation has 1 feature (1-dimension)
- Data is sampled from one of two Gaussian distributions (K=2)
  - Cluster r: (μ_r, σ_r^2)
  - Cluster b: (μ_b, σ_b^2)



- **Estimation**: If source (cluster) of each observation is known, it is trivial to estimate (μ_r, σ_r^2) and (μ_b, σ_b^2)

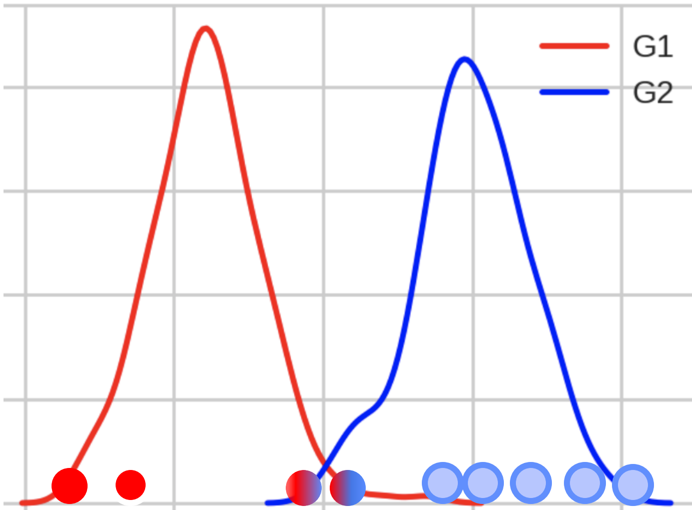
$$\mu_r = \frac{\sum_{i=1}^{N} x_i \mathbb{I}\{x_i \sim r\}}{\sum_{i=1}^{N} \mathbb{I}\{x_i \sim r\}} \qquad \sigma_r^2 = \frac{\sum_{i=1}^{N} (x_i - \mu_a)^2 \mathbb{I}\{x_i \sim r\}}{\sum_{i=1}^{N} \mathbb{I}\{x_i \sim r\}}$$

$$\mu_b = \frac{\sum_{i=1}^{N} x_i \mathbb{I}\{x_i \sim b\}}{\sum_{i=1}^{N} \mathbb{I}\{x_i \sim b\}} \qquad \sigma_b^2 = \frac{\sum_{i=1}^{N} (x_i - \mu_b)^2 \mathbb{I}\{x_i \sim b\}}{\sum_{i=1}^{N} \mathbb{I}\{x_i \sim b\}}$$

where $\mathbb{I}\{x_i \sim r\} = 1$ if x_i was sampled from cluster r and 0 otherwise.

GMM Example: Find posterior

- Observations: $x_1, x_2, \ldots, x_N$
  - Each observation has 1 feature (1-dimension)
- Data is sampled from one of two Gaussian distributions (K=2)
  - Cluster a: (μ_a, σ_a^2)
  - Cluster b: (μ_b, σ_b^2)



- If the distribution and its parameters are known, estimate where the point is likely to come from using Bayes rule

$$P(b|x_i) = \frac{p(x_i|b)P(b)}{p(x_i|b)P(b) + p(x_i|r)P(r)}$$

Posterior probability of distribution b given sample x_i

$$p(x_i|b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

Probability density of observing x_i when sampled from distribution b

Expectation Maximization

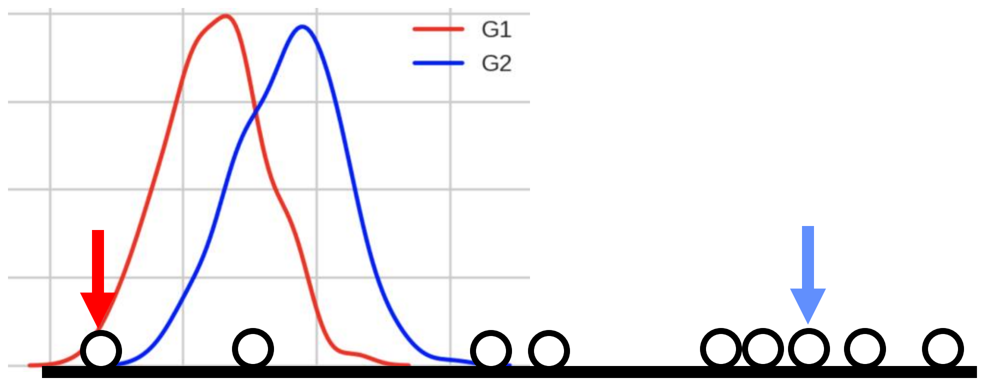
- **What if neither the source nor the distribution parameters are known?**

- **Chicken and Egg problem**
  - Need (μ_b, σ_b^2) and (μ_r, σ_r^2) to guess source of points
  - Need to know source to estimate (μ_b, σ_b^2) and (μ_r, σ_r^2)
  - Use **Expectation Maximization (EM)** algorithm

- **EM Algorithm**
  - Start with **two randomly placed Gaussians** (μ_b, σ_b^2) and (μ_r, σ_r^2)
  - For each x_i, calculate $P(b|x_i)$ and $P(r|x_i) = 1 - P(b|x_i)$
  - **Remember it does not assign the point but says here is the probability that it came from the red cluster or from the blue cluster (Soft assignment)**
  - Adjust (μ_b, σ_b^2) and (μ_r, σ_r^2) to fit points most likely belonging to them

GMM Example: EM in action

- Start with **two randomly placed Gaussians** (μ_b, σ_b^2) and (μ_r, σ_r^2)
- **Expectation step (E)**: Assign posterior probabilities to each sample x_i
- Let b_i be the posterior probability of sample x_i belonging to cluster b



$P(r|x) > P(b|x)$ $P(b|x) > P(r|x)$

$$b_i = P(b|x_i) = \frac{p(x_i|b)P(b)}{p(x_i|b)P(b) + p(x_i\,|r)P(r)}$$

$$p(x_i|b) = \frac{1}{\sqrt{2\pi\sigma_b^2}} \exp\left(-\frac{(x_i - \mu_b)^2}{2\sigma_b^2}\right)$$

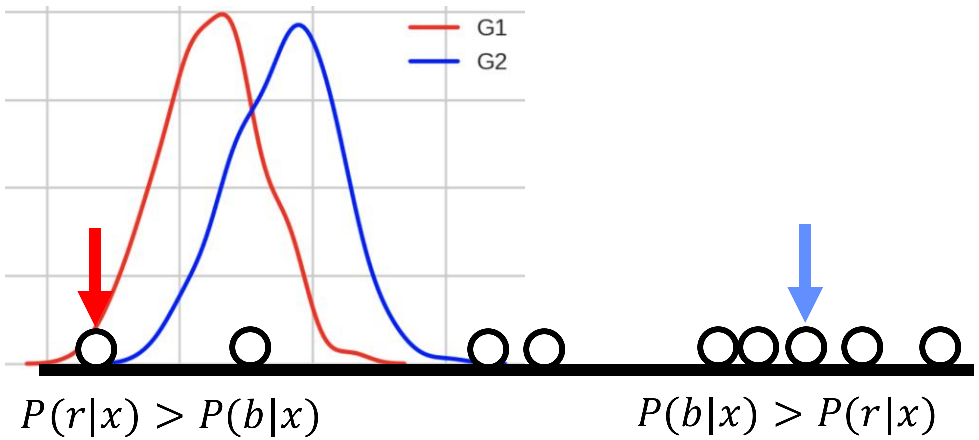
Probability density of observing x_i when sampled from distribution b

- Similarly, let r_i be the posterior probability of sample x_i belonging to cluster r

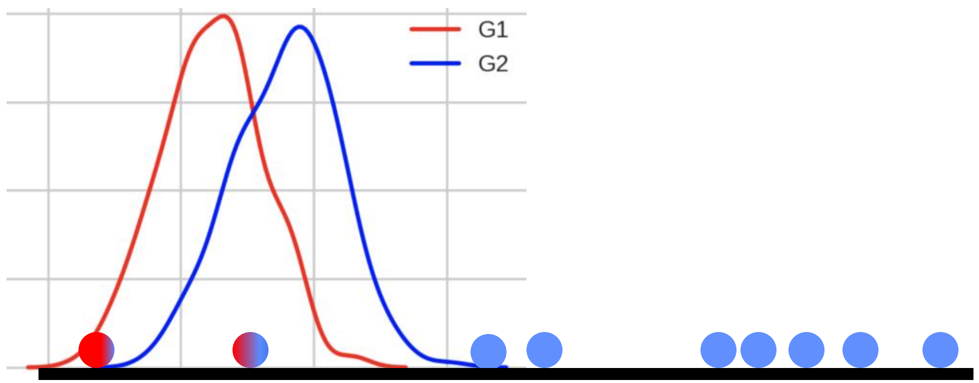
$$r_i = 1 - b_i$$

GMM Example: EM in action

Before assigning posterior probabilities b_i and r_i



After assigning posterior probabilities b_i and r_i



GMM Example: EM in action

- **Maximization step (M)**: Update the distribution parameters (re-estimation)
- Take **weighted average of the samples**
  - Weight is the posterior probability of that sample
- Similar to previous estimation, but with $\mathbb{I}\{x_i \sim b\}$ replaced by $P(b|x_i)$
  - $P(b|x_i)$ gives how likely it is that the cluster is b given the sample x_i
  - Therefore, x_i's contribution in re-estimating the parameters for b is $b_i = P(b|x_i)$

$$\mu_b = \frac{b_1 x_i + b_2 x_2 + \cdots + b_N x_N}{b_1 + b_2 + \cdots + b_N} = \frac{\sum_{i=1}^{N} b_i x_i}{\sum_{i=1}^{N} b_i}$$

Mean is simply weighted average of samples

$$\sigma_b^2 = \frac{b_1(x_1 - \mu_b)^2 + b_2(x_2 - \mu_b)^2 + \cdots + b_N(x_N - \mu_b)^2}{b_1 + b_2 + \cdots + b_N}$$

$$= \frac{\sum_{i=1}^{N} b_i (x_i - \mu_b)^2}{\sum_{i=1}^{N} b_i}$$

Variance is weighted sum of square distances of samples from the distribution mean

$$P(b) = \frac{b_1 + b_2 + \cdots + b_N}{N} = \frac{\sum_{i=1}^{N} b_i}{N}$$

Class prior is normalized sum of sample posteriors

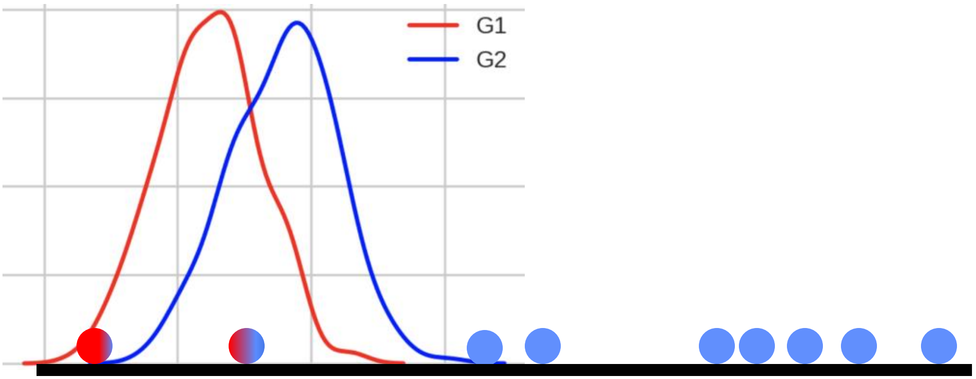
$$\mu_r = \frac{\sum_{i=1}^{N} r_i x_i}{\sum_{i=1}^{N} r_i}$$

$$\sigma_r^2 = \frac{\sum_{i=1}^{N} r (x_i - \mu_r)^2}{\sum_{i=1}^{N} r_i}$$

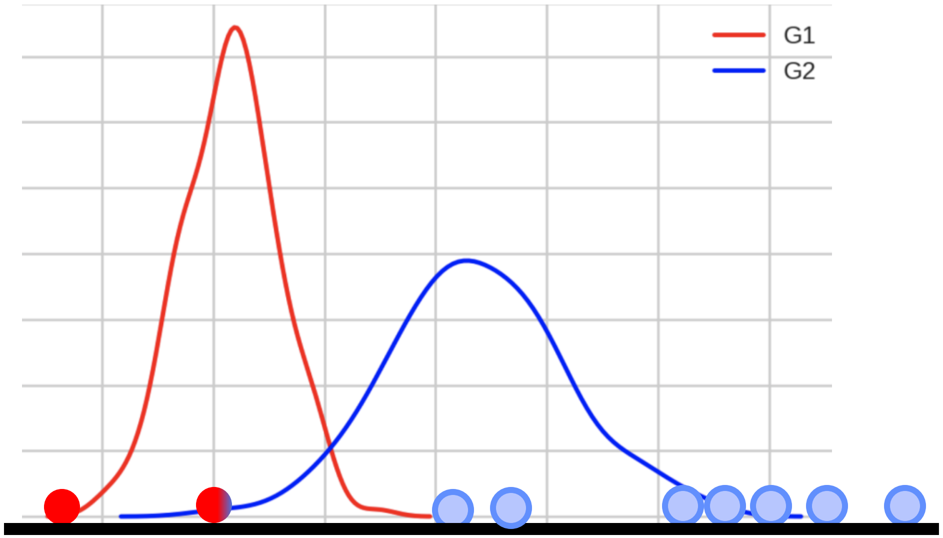
$$P(r) = \frac{\sum_{i=1}^{N} r_i}{N}$$

GMM Example: EM in action

Distributions before updating their parameters

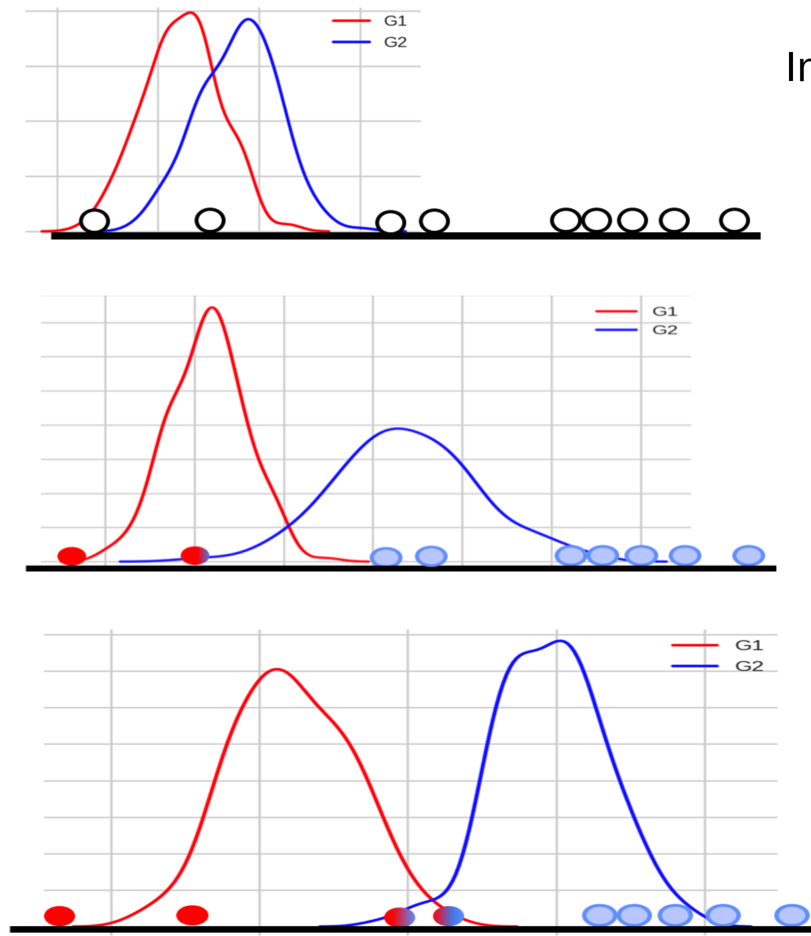


Distributions after updating their parameters using the posteriors



GMM Example: EM in action

- Repeat the E and M steps iteratively till convergence
- Convergence: When M step gives the same parameters that were used in E



Initialization

1 iteration

Convergence (n iteration)

GMM: Multi-dimensional features (1)

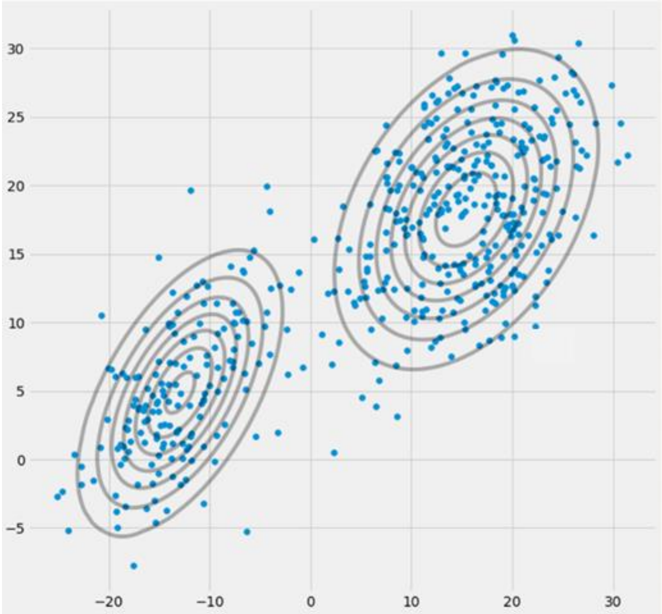
- Data with d features i.e., $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N} \in \mathbb{R}^d$ from K sources
- Each source $c \in \{1, \ldots, K\}$ has a Gaussian distribution, i.e., $\mathcal{N}(\mu_c, \Sigma_c)$ where $\mu_c \in \mathbb{R}^d$ and $\Sigma_c \in \mathbb{R}^{d \times d}$
- Iteratively estimate parameters
  - Prior: What fraction of instances came from source cluster c

$$P(c) = \frac{1}{N} \sum_{i=1}^{N} P(c|\boldsymbol{x_i})$$

  - Mean: Expected value of feature j from source cluster c:

$$\mu_{c,j} = \sum_{i=1}^{N} \left(\frac{P(c|\boldsymbol{x_i})}{N\, P(c)} \right) x_{i,j}$$

    - Similar to 1D case, but with extra index j to access specific feature from input vector



Source: https://www.python-course.eu/expectation_maximization_and_gaussian_mixture_models.php

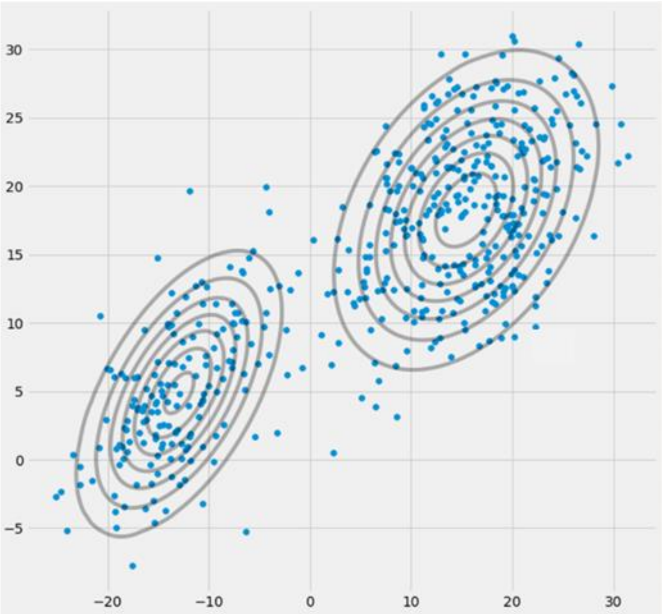
GMM: Multi-dimensional features (2)

- Data with d features i.e., $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_N} \in \mathbb{R}^d$ from K sources
- Each source $c \in \{1, \ldots, K\}$ has a Gaussian distribution, i.e., $\mathcal{N}(\mu_c, \Sigma_c)$ where $\mu_c \in \mathbb{R}^d$ and $\Sigma_c \in \mathbb{R}^{d \times d}$
- Iteratively estimate parameters
  - Covariance: How related are features j and k in source c:

$$(\Sigma_c)_{j,k} = \sum_{i=1}^{N} \left(\frac{P(c|\boldsymbol{x_i})}{NP(c)} \right) (x_{i,j} - \mu_{c,j})(x_{i,k} - \mu_{c,k})$$

  - Assignment: Based on our guess of the source for each instance

$$P(c|\boldsymbol{x_i}) = \frac{p(\boldsymbol{x_i}|c)P(c)}{\sum_{c'=1}^{K} p(\boldsymbol{x_i}|c')P(c')}$$



Source: https://www.python-course.eu/expectation_maximization_and_gaussian_mixture_models.php

Picking *K* - Gaussian Components

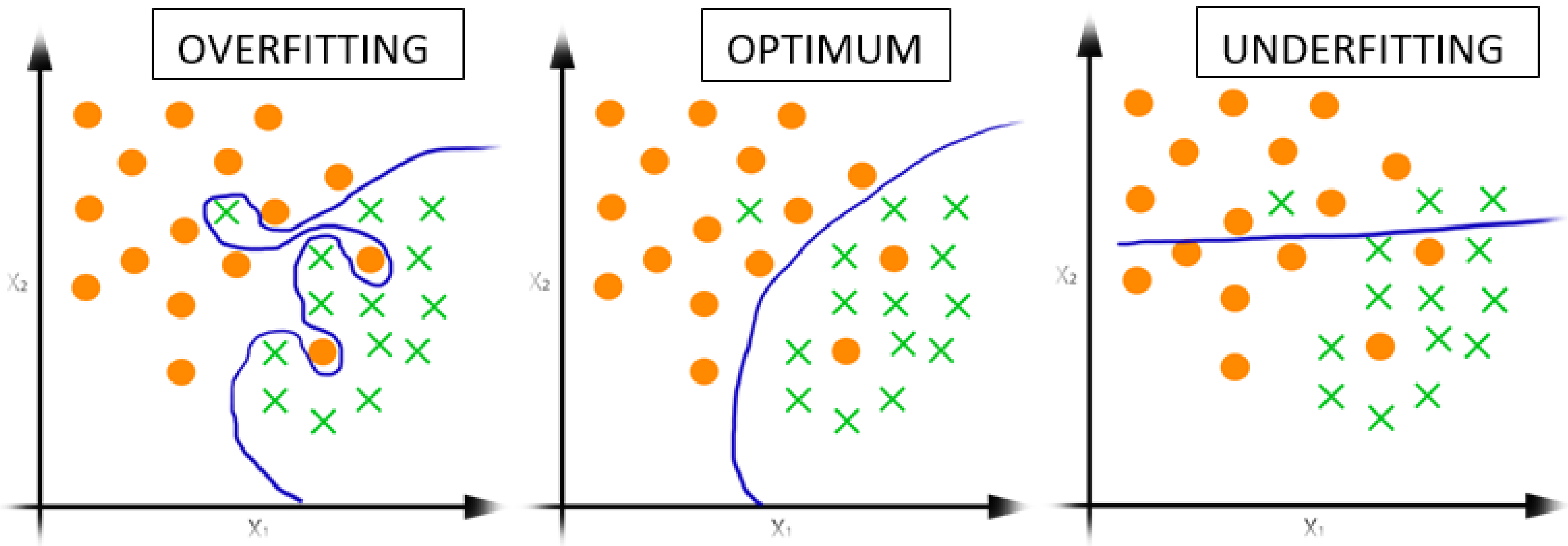
- Maximize the log likelihood of the data given the model

$$L = \log P(x_i, .., x_n) = \sum_{i=1}^{N} \log \sum_{k=1}^{K} p(x_i|k)P(k)$$

- Pick *K* that makes *L* as large as possible
  - *K* = *N*: each data point has its own source => overfitting
    - Unlikely to yield meaningful results for new (previously unseen) data points
    - Need to constrain (or regularize) to avoid overfitting

$$K^* = \underset{k \in \{1,..,K\}}{\operatorname{argmax}} L$$

Overfitting



Source: https://medium.com/@srjoglekar246/overfitting-and-human-behavior-5186df1e7d19

Picking *K* - Gaussian Components

Possible to deal with overfitting using the following two ways:

- Split points into training set *T* and validation set *V*
  - For each *K,* fit parameters on *T* and measure likelihood of *V*

- **Occam's Razor**: Pick "simplest" of all models that fit
  - Bayes Inference Criterion (BIC):
    - $(\log(N)\,K\ - 2\log L)$, where K is clusters, L: log likelihood [Fraley et. al , 2002]
    - When picking from several models, the one with the lowest BIC is preferred
    - BIC introduces a penalty term for adding parameters (i.e., #clusters)
- Cross Validation

Comparing *K*-Means and GMM

Similarity between GMM and K-means

- GMM
  - Given K

1. Randomly place K Gaussians distributions
2. Calculate posterior probability for each data point for each Gaussian (soft clustering)
3. Recompute mean and variance parameters of Gaussian distributions
4. Repeat 2 & 3 until convergence

- K-means algorithm
  - Given K

1. Randomly choose K data points (seeds) to be the initial centroids i.e. cluster centers
2. Assign each data point to the closest centroid (hard clustering)
3. Recompute the centroids using the current cluster memberships
4. Repeat 2 & 3 until convergence

Calculating centroid (mean) in k-means

Say you have two clusters (K=2) and six data points (x_1, x_2, … x_6). Assume that (x_1, x_4, x_5) belong to cluster 'a' and (x_2, x_3, x_6) belong to cluster 'b'

For k-means, centroid of cluster a:

$$centroid_a = \frac{x_2 + x_4 + x_5}{3} = \frac{x_1\,(0) + x_2(1) + x_3(0) + x_4(1) + x_5(1) + x_6(0)}{1(0)+1(1)+1(0)+1(1)+1(1)+1(0)}$$

- In the rightmost expression, x_i is multiplied with 1 if x_i belongs to cluster a and 0 if it does not.

Calculating mean in GMM

- If we were doing GMM, then the mean of cluster a (μ_a) is

$$\mu_a = \frac{x_1 P(a|x_1) + x_2 P(a|x_2) + x_3 P(a|x_3) + \ldots + x_6 P(a|x_6)}{1(P(a|x_1)) + 1(P(a|x_2)) + 1(P(a|x_3)) + \ldots + 1(P(a|x_6))}$$

Comparing formulae for means

- Notice the similarity between

$$\frac{x_1\,(0) + x_2(1) + x_3(0) + x_4(1) + x_5(1) + x_6(0)}{1(0) + 1(1) + 1(0) + 1(1) + 1(1) + 1(0)}$$

And

$$\frac{x_1 P(a|x_1) + x_2 P(a|x_2) + x_3 P(a|x_3) + \ldots + + x_6 P(a|x_6)}{1(P(a|x_1)) + 1(P(a|x_2)) + 1(P(a|x_3)) + \ldots + + 1(P(a|x_6))}$$

- Calculation of the mean involves:
  - Multiplying by 0 or 1 in k-means (hard clustering)
  - Multiplying by posterior probability (between 0 and 1) in GMM (soft clustering)

Summary

- K-means is a hard-clustering whereas GMMs is a soft-clustering method

- GMMs and K-means: Similarity
  - Sensitive to starting point, converges to local maximum
  - Convergence: When change in $P(x_1, x_2, \ldots, x_n)$ is sufficiently small
  - Cannot discover k easily

- Can make GMMs to behave as K-means
  - Fix variance to be 1
  - Uniform priors

Distances with Categorical Variables

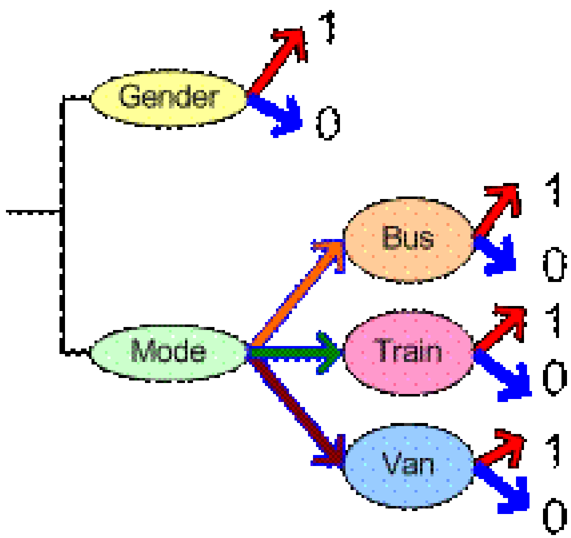
Nominal Distance

- **Nominal** variables are those where the values cannot be meaningfully measured or ordered (i.e. gender $\in \{0 = male, 1 = female\}$)
- We can find the distance between two samples with nominal features by doing the following:
  - 1. Split multinomial variables into multiple binary variables
  - 2. Redefine the datapoints using the new binary variables
  - 3. Calculate the distance between the redefined samples using a metric such as Hamming Distance
- Example: We have two variables:
  - **Gender** $\in \{0 = male, 1 = female\}$
  - **Transportation** $\in \{0 = bus, 1 = train, 2 = van\}$
- Suppose we have three samples:
  - **Alex** is male and takes the bus
  - **Brian** is male and takes the van
  - **Cherry** is female and takes the bus

https://people.revoledu.com/kardi/tutorial/Similarity/NominalVariables.html#Method1

Nominal Distance

- 1.: Split multinomial variables into multiple binary variables
  - Gender is binary already, so we don't need to do anything
  - Transportation can take on 3 values, so we split it into 3 binary variables: **Bus**, **Train**, and **Van**



https://people.revoledu.com/kardi/tutorial/Similarity/NominalVariables.html#Method1

Nominal Distance

- 2.: Redefine the datapoints using the new binary variables
  - **Alex** is male and takes the bus ⇒ $\boldsymbol{Alex} = (0, (1, 0, 0))$
  - **Brian** is male and takes the van ⇒ $\boldsymbol{Brian} = (0, (0, 0, 1))$
  - **Cherry** is female and takes the bus ⇒ $\boldsymbol{Cherry} = (1, (1, 0, 0))$

https://people.revoledu.com/kardi/tutorial/Similarity/NominalVariables.html#Method1

Nominal Distance

- 3.: Calculate the distance between the redefined samples using a metric such as Hamming Distance
- Hamming Distance: length of different digits
  - Hdistance(Alex, Brian) = Hdistance(0, (1, 0, 1)) = 0+1+0+1 = 2
  - Hdistance(Alex, Cherry) = Hdistance(1, (0, 0, 0)) = 1+0+0+0 = 1
  - Hdistance(Brian, Cherry) = Hdistance(1, (1, 0, 1)) = 1 + 1 + 0 + 1 = 3

- Ratio Distance: ratio of number of unmatched and total dummy variables
  - There are **2** total features, and the transportation feature can take **3** values
  - Rdistance(Alex, Brian) = Rdistance(0, (1, 0, 1)) = (0 + (1+0+1)/**3**)/**2** = 1/3
  - Rdistance(Alex, Cherry) = Rdistance(1, (0, 0, 0)) = (1 + (0+0+0)/**3**)/**2** = 1/2
  - Rdistance(Brian, Cherry) = Rdistance(1, (1, 0, 1)) = (1 + (1+0+1)/**3**)/**2** = 5/6

https://people.revoledu.com/kardi/tutorial/Similarity/NominalVariables.html#Method1

Vector Distances

- When some (or all) features are categorical, we can calculate distances between multi-dimensional data points using vectors
- Consider two features:
  - Categorical**: Gender** = {1 (Male), 2 (Female)}
  - Continuous: **Age** ∈ $[0, 100]$
- Three data points
  - John: Male, 20 years old ⇒ $\boldsymbol{J} = <1, 20>$
  - Kevin: Male, 25 years old ⇒ $\boldsymbol{K} = <1, 25>$
  - Lea: Female, 40 years old ⇒ $\boldsymbol{L} = <2, 40>$

https://stackoverflow.com/questions/29771355/how-can-we-measure-the-similarity-distance-between-categorical-data

Vector Distances

1. **Shift and Scale**: For each feature, shift the minimum value to 0 and scale the result so that it lies in a given range (say, [0, 100])
   1. Offset = (-1) * minimum value
   2. Scale Factor = 100 / (maximum value - minimum value)

   Gender: offset = -1, scale = (100 / 1) = 100

   Age: offset = 0, scale = (100/100) = 1

$$\boldsymbol{J} \rightarrow <\left(1+offset_{gender}\right) * scale_{gender}, \left(20+offset_{age}\right) * scale_{age} >=< 0,20 >$$

Similalry, $\boldsymbol{K} \rightarrow < 0,25 >, \boldsymbol{L} \rightarrow < 100,40 >$

2. **Calculate Distances**: Calculate Euclidean distance between vectors to measure similarity

$$\|\boldsymbol{J}-\boldsymbol{K}\|_{\mathbf{2}} = \sqrt{(0-0)^2+(20-25)^2} = \mathbf{5}$$

$$\|\boldsymbol{J}-\boldsymbol{L}\|_{\mathbf{2}} = \sqrt{(0-100)^2+(20-40)^2} = \mathbf{101.98}$$

Thus, John is much more “similar” to Kevin than to Lea…

https://stackoverflow.com/questions/29771355/how-can-we-measure-the-similarity-distance-between-categorical-data

Vector Distances

- Some issues with this method include
  - "Artificial" difference inflation using larger scale ranges
    - For example, having a scale range from 0-100 makes the scaled difference between male and female 100, whereas a scale range from 0-10 would make the scaled difference between male and female 10
  - All features are weighted equally
    - For example, a difference in age of 100 years is mathematically equivalent to a difference in gender of male vs. female. Whether or not these two "differences" are equivalent in significance depends on the nature of the system being modeled

- The Gower's distance metric addresses these limitations

Distances when Categorical Features are Used

- Gower's distance between two samples j, k

$$d_{j,k} = \frac{\sum_{i=1}^{m} w_i d_{i,j,k}}{\sum_{i=1}^{m} w_i}$$

where

m = number of features per sample

w_i = weights for each feature

$d_{i,j,k}$ = distance between the i^{th} feature of j and k

If feature i is

1. Continuous

$$d_{i,j,k} = 1 - \frac{|x_{i,j} - x_{i,k}|}{R_i}$$

R_i = range of i^{th} feature

2. Categorical

$$d_{i,j,k} = \begin{cases} 0 : x_{i,j} = x_{i,k} \\ 1 : x_{i,j} \neq x_{i,k} \end{cases}$$