Lecture 2:

Confirmation bias: believing something that you think is correct (limiting your sight)

- Positive example: avoiding confirmation bias
- Negative example: falling into confirmation bias
- Both positive and negative can be wrong
- Tendency to choose positive over negative

Unsupervised: trying to find the pattern from the unlabeled dataset

- Data exploration and visualization: finding patterns that identify trends or clusters
- Extracting features: finding the important part of the data that is significant
- Filling in gaps in data: fill the NA using the pattern that we discover
- Get ML model faster: get rid of the noise data that affects the prediction of the model

Supervised learning: data with the label, and we make a prediction

In K-means, **k refers to the optimal number of clusters**.

What are the four types of data in statistics?

- Nominal data - can be labelled or classified into mutually exclusive categories within a variable. These categories cannot be ordered in a meaningful way.
    - Sad, happy, neutral
- Ordinal data - categorical, statistical data type where the variables have natural, ordered categories and the distances between the categories are not known
    - EXAMPLE: On a scale of 1-5, how much did you like this movie?
- Discrete data - Data that can only take certain values is called discrete data or discrete values. This is data that can be counted and has a limited number of values. It usually comes in the form of whole numbers or integers. These values must fit into certain categories and can't be broken into smaller parts.
    - Example: # of players on a team
- Continuous data - data that can take any value within a given range

What is the function minimized by the K-Means algorithm?

- Within-Cluster Sum of Squares

K-means is good at dividing up sphere-esque clusters, bad at dividing bends/arcs.

Lecture 3:

Data: n points(i) and m features or attributes(j)

Feature space: representing data **graphically** in a way that helps us understand patterns and relationships between features.

Dissimilarity function: get two data points and plug into the function, it will return a large value if the data is dissimilar

Distance function:

- Property of distance function:

  d is a distance function if and only if:

  - d(i, j) = 0 if and only if i = j
  - d(i, j) = d(j, i)
  - d(i, j) ≤ d(i, k) + d(k, j)

- Theorem: A and B are small, B and C are small, A and C are not necessarily small, but if we use the triangle inequality, which means that d(a,c)<=d(a,b) + d(b,c)

Minkowski distance:

Minkowski Distance

For **x**, **y** points in **d**-dimensional real space

I.e. **x = [x_1 , … , x_d]** and **y = [y_1 , … , y_d]**

p ≥ 1

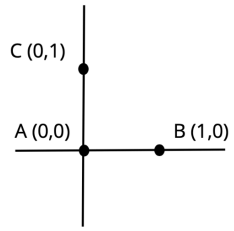
$$L_p(x, y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

When **p** = 2 -> Euclidean Distance

When **p** = 1 -> Manhattan Distance

- Measure a different d-dimensional space.
- P is the parameter that determines how the distance is calculated
- It will not work if 0<p<1, which it will violate the triangle inequality

Is L_p a distance function when $0 < p < 1$?



D(B,A) + D(A, C) = 2

$D(B, C) = 2^{1/p}$

So **D(B, C) > D(B, A) + D(A, C)** which violates the triangle inequality

-
- BC will have 2^1/p, which will end up being bigger than AC+AB, which violates the theorem.

Jaccard similarity

- Jaccard distance is 1- Jaccard similarity
- Jaccard similarity in [0,1], close to 1 means similar
- Jaccard similarity: (A and B)/(A or B) = and / or
- Example: d=100, only the last two words are different. Solution: similarity: 98/100, distance=1-0.98=0.02
- Manhattan distance, sum of difference (abs will output 0 or 1), sum of 1s means how many different pairs.

Cosine similarity or dissimilarity

- Theta is the distance between the x and y two vectors
- In the range 0 to 1, which 0 is completely different and 1 is identical (the same)
- -1 would be 180 degrees, which means opposite direction
- d(x,y)=1/s(x,y), we can use 1-s(x,y)
- Formula: s(x,y)= dot(x,y)/(norm(x)*norm(y))
- Minkowski←→Lp norm, not all distances can create norms

Lecture 4

Cluster: group similar objects to be in one group

Application of cluster: outlier detection, feature extraction, and filling gap for data

Type of cluster: partitional, hierarchical, density-based, and soft clustering

Partitional cluster: each object belongs to only one group

Goal of partitional cluster: Partition the dataset into k groups

Cost function: The sum of distance between data point and its assign centroid.

$$\sum_{i}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$$

- K=1 and K=n are easy
- D is Euclidean distance
- Xi lives in more than 2d, then it's an NP-Hard problem

Goal: minimize the cost function until convergence

Steps:

1. Initialize centroids
2. Assign data points to the nearest centroid (see the closest centroid)
3. Update centroids (compute the mean of all points in each cluster, move that centroid to this new mean position) Formula: mean(x)=new x, mean(y)= new y
4. Repeat 2 and 3 until convergence

Lecture 5:

Lloyd's algorithm will always converge to a local optimal solution, but it will not be guaranteed for the optimal solution. For example:

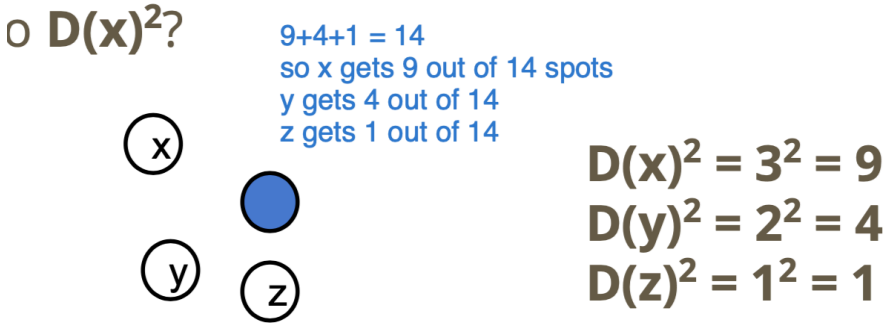
And that's why we need k-means++

Why K-Means++ Is Less Sensitive to Outliers Than FFT

1. **K-Means++ uses probability, not strict farthest-first selection**
   - While an outlier **has a higher probability of being chosen** (since it has a large squared distance), it is **not guaranteed** to be picked.
   - The selection process still allows for **other distant but representative points** to be chosen.
2. **FFT always picks the most extreme point**
   - **FFT deterministically selects the absolute farthest point** at each step.
   - This means if there's a single **outlier very far away**, FFT **will** pick it, leading to bad clustering.

Farest First Transversal: it would choose the farest point to be the next centroid

K-means++: start with a random center, and use D(x) to find the next center and choose the one that is proportional to D(x)^2



What happens if the black box can only generate numbers between 0 and 1?
We have it generate a number in 0 to 1 and see which range it falls on, the far point will have a greater range.

Elbow method:

Iterating the difference number of k to see the cost and graph it

How to choose the right k:

1. Iterating the different values of k
2. If prior knowledge is given, use it
3. Metrics for evaluating clustering quality

Sometimes, low cost != best choice of k, depending on circumstances.

To evaluate the quality of the cluster:

We can use **the Silhouette score:**

ai= the mean distance of point i to the points that in the same cluster

bi= the smallest mean distance of point i to every point in the other cluster

Formula:

$$s_i = (b_i - a_i) / \max(a_i, b_i)$$

What is the score mean?

- 0 means on the boundary between clusters
- 1 well-clustered
- -1 incorrectly clustered (anything that negative means misgroup)

Other topic:

K-Median: use L1norm instead of L2

K-Medoids: use any distance function +cluster centers must be actual data points

Weighted K-means: each point has a different weight when computing the mean of a cluster (some points are more important than others)

Lecture 6:

Hierarchical cluster:

Two types of hierarchical clustering: **Agglomerative and divisive**

Agglomerative: start with each point considered as a cluster group, at each step, merge the two closest clusters, stop when the whole dataset is in one cluster

Divisive: Start with everything in one cluster, stop until every point in its cluster

Agglomerative step:

1. Let each point become a cluster
2. Find the distance between all pairs of clusters
3. Each cluster merges its closest pair
4. Repeat 2 and 3 until it's all in one

Single Linkage: min of the min

For x1 from c1 and x2 from c2, they are the pair between these two cluster that have the smallest distance between them.

$$D_{SL}(C_1, C_2) = \min \{d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\}$$

AKA distance between the closest two points of each cluster

a=0,0 b=1,1 c=3,0 d=0,-2

A close to d and b close to c

d(a,d)=sqrt((0-0)^2+(0+2)^2)=sqrt(0+4)=2

d(b,c)=sqrt((1-3)^2+(1-0)^2)=sqrt(4+1)=sqrt(5)

We will group D

- Sensitive to noise
- Can handle different size
-

Complete Linkage: min of the max

$$D_{CL}(C_1, C_2) = \max \{d(p_1, p_2) \mid p_1 \in C_1, p_2 \in C_2\}$$

min of the max

a=0,0 b=1,1 c=3,0 d=0,-2

A is far from c and b is far from d

d(a,c)=sqrt((0-3)^2+0)=sqrt(9)=3

d(b,d)=sqrt((1-0)^2+(1+2)^2)=sqrt(1+9)=sqrt(10)

- Less sensitive to noise
- Create balance between group
- Tend to split up the large group of cluster
- All clusters have the same delimiter

Average Linkage:

$$D_{AL}(C_1, C_2) = \frac{1}{|C_1| \cdot |C_2|} \sum_{p_1 \in C_1, p_2 \in C_2} d(p_1, p_2)$$

I C_1 I is the cardinality of the set
AKA number of points in that cluster

Or we can just calculate pairs and divide by the number of pairs

Problem with this method: may not be that sensitive to noise and outliers, tends to be biased toward globular clusters

- Less susceptible to noise and outlier
- works best when clusters are **roughly spherical or compact**,

Centroid Linkage:

$$D_C(C_1, C_2) = d(\mu_1, \mu_2)$$

Instead of doing point comparison, we now do centroid comparison

Ward distance:

$$D_{WD}(C_1, C_2) = \sum_{p \in C_{12}} d(p, \mu_{12}) - \sum_{p_1 \in C_1} d(p_1, \mu_1) - \sum_{p_2 \in C_2} d(p_2, \mu_2)$$

Ward distance tells you how much of the total variance increases after merging the clusters c1=a,b c2=d, after merge c12=a,b,d

-

Lecture 6:

Density-based clustering:

Core points: if its e neighbor includes at least min point

Border point: if it's in the range of e of a core point

Noise point: not a core or border point

DBScan algorithm:

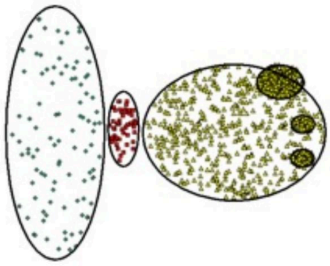
1. Find the neighbors in e radius (range)
2. Label the point if it satisfies the minimum number of points
3. Continue the discovery point that in the core point neighborhood
4. Mark not core point but in the range of a core to be broader
5. Mark is not border and not core to be noise
6. Assign border point to nearby cluster

DBScan benefits:

1. Can identify clusters of different sizes or shape
2. Resistant to noise “not sensitive to noise”

DBScan limitation:

1. Can’t identify cluster for the dataset with vary density



2. Tends to create cluster of the same density, it will cluster high density, but it will classify low density points as noise points
3. Don’t do well at higher dimensions

Lecture 7:

Soft Cluster:

Goal: assign data points to clusters probabilistically rather than using strict(hard) cluster like K-Means.

Example: A dataset is sampled from different animal species, instead of belonging to A, it may belong to A and B, which has a probability.

There is a prior probability of belonging to a species: Some species are more common than others, meaning that P(S) varies

Different species have different weight distribution: the distribution varies by species, meaning that we need to use probability density function (PDF) to model them.

$$P(S_j|X_i) = \frac{P(X_i|S_j)P(S_j)}{P(X_i)}$$

$P(S_j)$ is the prior probability of seeing species S_j (that probability would be higher for the Stegosauruses than the Raptors for example)

$P(X_i \mid S_j)$ is the **PDF** of species S_j weights evaluated at weight X_i (seeing a Sauropod that weighs 100 tons is way more likely than seeing a Raptor that weighs 100 tons)

X comes from a mixture model with k mixture components if the probability distribution of X is:

$$P(X) = \sum_j P(S_j)P(X|S_j)$$

Mixture proportion
Represents the probability of belonging to S_j

Probability of seeing x when sampling from S_j

P(xi): the total probability of observing the data point Xi, considering all possible clusters

P(sj): the prior probability of cluster Sj, how likely is that randomly chosen data point belongs to Sj

P(xi|sj): the likelihood of observing Xi given that it belongs to cluster Sj

This is a Gaussian Mixture Model formula:

A Gaussian Mixture Model (GMM) is a mixture model where

$$P(X|S_j) \sim N(\mu, \sigma)$$

- X given that it belongs to the species Sj is normally distributed

Maximum likelihood estimator: finding the best parameter for Gaussian Mixture Model

$$P(H)P(T)P(T)P(H)P(T) = p^2(1-p)^3$$

Goal: find the p that maximizes the probability

GMM clustering: find the GMM that maximizes the probability of seeing the data we have
Estimating 3k: P(sj), mean, covariance
GMM assign all data and give them probability to each k

1. Initialize parameters: select k Gaussian distributions (like k in k-means)

Note: at first, mean and data give sigma, p(si) or we called the weight of the cluster will be considered as a uniform distribution. For example: k=2, p(s1) and p(s2) will both be 0.5.

2. Compute the probability that each point belongs to each cluster
3. Update parameters mean, cov, and p(sj) based on these probabilities
4. Repeat until convergence

To get

$$\hat{\mu}_j = \frac{\sum_i P(S_j|X_i)X_i}{\sum_i P(S_j|X_i)}$$

$$\hat{\Sigma}_j = \frac{\sum_i P(S_j|X_i)(X_i - \hat{\mu}_j)^T(X_i - \hat{\mu}_j)}{\sum_i P(S_j|X_i)}$$

$$\hat{P}(S_j) = \frac{1}{N}\sum_i P(S_j|X_i)$$

Lecture 8:

Clustering: A group of clusters output by a cluster algorithm

Cluster: A group of points

Cluster Aggregation: Comparing clustering, and combining the information of multiple clustering into a new clustering

Goal of clustering aggregation:

1. Compare clustering: given two different cluster outputs, how similar or different are they?
2. Combine information from multiple clusterings: take multiple results and combine them into one, make it optimal

Given 2 clusterings P and C

$$D(P, C) = \sum_{x,y} \mathbb{I}_{P,C}(x, y)$$

where

$$\mathbb{I}_{P,C}(x, y) = \begin{cases} 1 & \text{if P \& C disagree on which clusters x \& y belong to} \\ 0 \end{cases}$$

	P	C
x_1	1	1
x_2	1	2
x_3	2	1
x_4	3	3
x_5	3	4

What is the disagreement distance between P and C?

Answer: 3

In P, x1 and x2 are the same, but in C are not.

In P x4 and x5 are the same, but in C are not.

In C x1 and x3 are the same, but in P are not.

How many total difference it can have, g choose 2, in this case is 5 choose 2=10

Property:

- Identity property: if D(c,p)=0, which means that c=p, which means that there's no disagreement
- D(c,p)=D(p,c), which means that the order of who compares first does not matter
- Hold for triangle inequality: D(a,c)<=D(a,b)+D(b,c)

Aggregate clustering:

Goal: From a set of clustering c1…cm, generate a clustering c* that minimizes the formula:

Aggregate Clustering

Goal: From a set of clusterings $C_1, ..., C_m$, generate a clustering C^* that minimizes:

$$\sum_{i=1}^{m} D(C^*, C_i)$$

The problem is equivalent to clustering categorical data

Benefits:

1. it can decide which k should be used in hard-clustering. Example: some group says 3 and some say 4, clustering agreement among methods and selects the optimal clustering structure.
2. It can handle and detect outliers. If a point is getting into a different cluster in different clustering, then it would be an outlier.
3. Improve robustness of clustering algorithm. It will end up with a more reliable final clustering, rather than depending on a single method that might be biased.

Problems:

1. It's np hard

Lecture 9:

Singular Value Decomposition (**SVD**):

Goal:

- Approximate matrix A into a smaller size matrix B that is easy to store but also contains the similar information as A
- Dimension decrease, feature extraction
- Anomity detection and reduction

Linearly Independent: it can only be represented by itself

a1v1+a2v2+...+agvg=0

Where a1=0, a2=0

Example:

We check:

$$a_1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} + a_2 \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This leads to:

$$a_1(1) + a_2(2) = 0$$

$$a_1(2) + a_2(4) = 0$$

Counterexample:

We check:

$$a_1 \begin{bmatrix} 1 \\ 2 \end{bmatrix} + a_2 \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This leads to:

$$a_1(1) + a_2(2) = 0$$

$$a_1(2) + a_2(4) = 0$$

Solving, we see $a_1 = -2a_2$, so we can pick $a_2 = 1$ and get $a_1 = -2$. Since a **non-trivial** solution exists, these vectors are **linearly dependent**.

Determinant:

Linearly independent have a determinant that is **non-zero**

Goal:

Approximate **A** with **A$^{(k)}$** (low-rank matrix) such that

1. **d(A, A$^{(k)}$)** is small
2. **k** is small compared to **m** & **n**

Linear Algebra rules:

- Det(A) not equal to 0 means that it's full rank
- Det(A) not equal to 0 means that it's linear independent

Matrix factorization:

- A=UV where U is n*k and V is k*m, which A results in n*m
- To store a matrix A with n*m requires m*n space
- If rank for A is k, we can have A=UV, and which requires k(n+m) space

Approximtion:

Goal: we reduce the rank of A to become Ak, which Ak contains similar information but rank has been reduce, such that:

- d(A,Ak) is small (not losing much important information)
- K is small compared to m&n (less storing and computation)

Frobenius distance:

- **Measure the distance between A and Ak**
- **Knowing how well is Ak represent original matrix A**
- **Showing how much error introduced when we use Ak instead of A**

Frobenius Distance

$$d_F(A, B) = \|A - B\|_F = \sqrt{\sum_{i,j} (a_{ij} - b_{ij})^2}$$

Singular Value Decomposition (SVD):

- SVD decomposes a matrix into three parts:
  - $\mathbf{A = U\Sigma V^T}$, where:
    - **U**: Orthogonal matrix with unit-length column vectors.
    - **Σ**: Diagonal matrix with **singular values** (square roots of eigenvalues of A^TA).
    - **V**: Orthogonal matrix with unit-length row vectors.

We want to find the Ak with the k that best minimized the Frobenius distance

Frobenius norm error:

A is the matrix

A^k is the dimension reduction matrix

Sigma is the singular value

The formula see the error between the A and its low rank matrix

Use: k=number of rank k

We only calculate the sigma that is not in our rank, for this example: if k=2, we can only calculate sigma3 if this is a 3x3 matrix.

Property:

$$d_F(A, A^{(k)})^2 = \sum_{i=k+1}^{r} \sigma_i^2$$

Note: the larger **k** is, the smaller the distance.

How to find the right k?

We can use elbow method like finding k for kmeans, which we plug different k and look at residual error to choose k.

In this case, residual error is frobenius error

Anomaly Detection

Define **O = A - $A^{(k)}$**

The largest rows of **O** could be considered anomalies