Week 01

01] What is Clustering ?

*) Clustering is an unsupervised learning technique used to group similar data points together based on some measure of similarity (usually distance)

*] The goal is to discover patterns or structures in data without labels

*) Each group is called a cluster, and points within the same cluster are more similar to each other than to points in other clusters

ex : market segmentation, Grouping similar news

02] k - means Algorithm

*] most simplest and popular clustering algorithms.

*] It tries to partition the data into k clusters where each data point belongs to the cluster with the nearest centroid (mean points in that cluster)

ProMate

K - means algorithm Steps

1] Chose the number of clusters (k)
- Decide how many groups you want the algorithm to find.

2] Initialize centroids ($\mu_1, \mu_2, \ldots \mu_k$)
- Randomly select K points (or use a smart method like k-Means ++) as initial cluster Centers

3] Then assign points to the nearest Centroid;
for i = 1 to m (training examples)
$c^{(i)}$:= Index (from 1 to k) of cluster Centroid closest to $x^{(i)}$

$$\min \| x^{(i)} - \mu_k \|^2$$

($x^1, c^1 = ①$ | $x^{12}, c^{12} = ②$ | $x^5, c^5 = ③$)

4] Update the centroids (move centroids)
for k = 1 to k
μ_k := Average (mean) of points assigned to cluster k

5] Repeat 3, 4 until Convergence
- when cluster assignments no longer change or the centroid position stop moving significantly

ProMate

*) If we found a cluster which has no training samples, we simply remove that cluster (k-1)

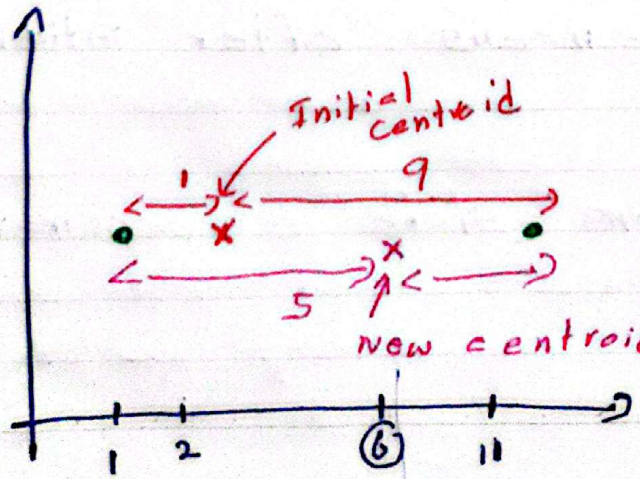
03] Optimization Objective (Cost Function)

*) k-means aims to minimize the sum of squared distance between each point and its cluster centroid

$$J(c^{(1)}, \dots c^{(m)}, \mu_1, \dots \mu_k) = \frac{1}{m} \sum_{i=1}^{m} \| x^{(i)} - \mu_{c^{(i)}} \|^2$$

m = number of data points
$c^{(i)}$ = index of the cluster assigned to point i
$\mu_{c^{(i)}}$ = Centroid of the cluster point i belongs to

*) The goal of k means is to minimize $J(c, \mu)$



$\frac{1}{2}(1+11) = 6$

$\frac{1}{2}(1^2 + 9^2) = \frac{1}{2}(1+81)$

$= 41$

mini mized

$\frac{1}{2}(5^2 + 5^2) = 25$

ProMate

05) Initializing K-Means

Random Initialization

*) Pick k random points from the dataset as initial centroids

*) May lead to different results each time because k-means can converge to local minima

K-Means ++ Initialization

*) A smarter method that spreads out the initial centroids to improve performance

1) Pick the first centroid randomly

2) For each remaining centroid, pick a point with probability proportional to its squared distance from the nearest existing centroid.

3) Run normal k-means after intialization

*) This gives more stable & accurate clustering.

ProMate

06] Chosing the number of Clusters (k)

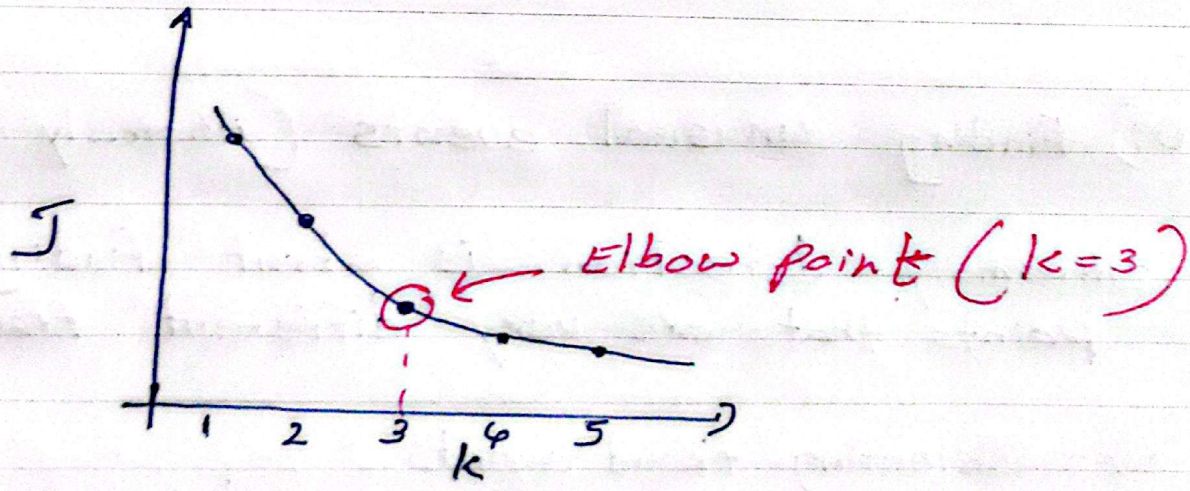
*] Selecting the right k is crucial and often depends on experimentation.

Elbow method

*] Plot the cost J(distortion) vs number of clusters k.

*] As k increases, cost decreases (clusters fit data better)

*] Chose k at the "elbow" point, where improvement slows down.



Silhoutte Score

*] Measures how similar a point is to its own cluster compared to others.

*) Ranges -1 to +1
=> +1 = well clustered
-1 = misclasified
0 = on the boundary

*) Chose k that gives highest silhoutte score.

Gap statistic

*) Compares the total within-cluster variation for different k values with reference random data

*) Chose k where the gap is largest.

07) Finding unusual events (Anomaly Detection)

*) Anomaly detection is about finding data points that are very different from the majority

ex: Detecting fraud events
Detecting abnormal server behaviour
Detecting defective products in manufacturing

*) We model what 'normal' looks like, Then if a new example has a very low probability of being normal, we flag it "anomaly"

ProMate

08] Gaussian Distribution (Normal Distribution)

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

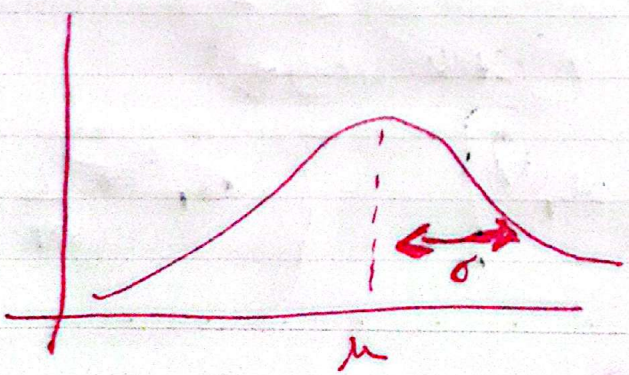
- μ = mean
- σ^2 = Variance
- x : data point
- $p(x)$: probability Density

*) most data points are near the mean (typical behaviour)

*) Values far from the mean are rare (potential anomalies)

*) If we have multiple features, we assume they are independent and compute

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$



ProMate

09) Anomaly Detection Steps

(Anomaly detection system for a server)

*) Each example (data point) describes the server's behaviour at a given time using features like

1) CPU usage
2) memory usage

1] Collect Training Data

*) we collect the many normal examples (times when the server was working fine)

*) we will learn what "normal" CPU and Memory values look like

2] Estimate Parameters

*) we find the mean and variance for each feature (CPU & Memory)

⇒ For feature 1 (CPU usage)

$$\mu_1 = \frac{1}{m} \sum_{i=1}^{m} x_1^{(i)}$$

$$\sigma_1^2 = \frac{1}{m} \sum_{i=1}^{m} (x_1^{(i)} - \mu_1)^2$$

ProMate

=) For feature 2 (Memory Usage)

$$\mu_2 = \frac{1}{m} \sum_{i=1}^{m} x_2^{(i)}$$

$$\sigma_2^2 = \frac{1}{m} \sum_{i=1}^{m} (x_2^{(i)} - \mu_2)^2$$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

i = example number (row in the dataset)
j = feature number (Column in the dataset)

*) ex : $x_2^{(5)}$ = Value of feature 2 (memory) for example #5

*) We compute μ and σ^2 for each column of data

3) Compute probability of Each example

*) we assume each feature allows a Gaussian (normal) Distribution

For one feature : $$p(x_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}}$$

ProMate

If we have multiple features:

$$p(x^{(i)}) = p(x_1^{(i)}) \times p(x_2^{(i)})$$

*) This gives probability score for each example.

High $p(x)$ ⇒ looks normal
Low $p(x)$ ⇒ looks strange (maybe an anomaly)

4) Decision phase

*) Now let's test a new data point:

CPU - 90t., Memory - 95t.

we Compute : $p(x_{test}) = p(90; \mu_1, \sigma_1^2) \times p(95, \mu_2, \sigma_2^2)$

*) If this probability is very small,

$$p(x_{test}) < \varepsilon$$

we classify its as a anomaly

5] Decision Rule

$P(x) \geq \varepsilon \Rightarrow$ Normal $(y=0)$

$P(x) < \varepsilon \Rightarrow$ Anomaly $(y=1)$

10] Developing and Evaluating an Anomaly Detection System

1] Chose a Dataset

we split our data into 3 parts

1] Training set → Mostly normal examples
- Learn what normal behaviour looks like

2] Cross - Validation → Normal + Some anomalies set
- Tune (Chose) threshold (ε)

3] Test set → Mix of normal + anomalies
- Final evaluation of system performance

ex :- 10,000 Normal / 50 Anomaly
Training - 8000 normal
CV set - 1000 nomal + 25 anomalies
Test - 1000 normal + 25 anomalies

ProMate

Date No

2] Fit the model

*) Now we use the training set (normal data) to model normal behaviour

For each feature j,
Compute mean (μ_j)
Compute variance (σ_j^2)

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} \left(x_j^{(i)} - \mu_j \right)^2$$

*) Then we calculate, for any new data point x, the probability that it came from this distribution

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2)$$

*) This gives each point "normality score"

*) Low $p(x)$ → unusual → maybe an anomaly

3) Chose Threshold ε

*) We use CV set for this

*) We try different possible thresholds ε (0.1, 0.001, 0.01, 0.0001, etc.)

*) For each threshold,
   => Predict anomaly if $(p(x)) < \varepsilon$
   => Predict normal if $(p(x)) \geq \varepsilon$

*) Then we check how many predictions are correct
(Semi - Supervised learning)

4) Evaluate using metrics

*) Because anomalies are rare, normal "accuracy" isn't enough (99% normal data, still get 99%.)

*) Instead we use precision, recall, F1 score

*) Usually we calculate the F1 score and select the ε with the highest F_1

5) Test the system

*) Finally, we test the model on the test set using the chosen ε

*) Then gets a fair estimate (by, f1, recall, precision)

11) Anomaly Detection vs Supervised learning

A.D	S.L
*) Very small no. of positive examples ($y=1$). (0-20) Large number of negative examples ($y=0$)	*) Large no. of positive and negative examples
*) Many different 'types' of anomalies. Hard for any algorithm to learn from positive exampes what the anomalies look like; future anomalies may look nothing like any of the anomalous examples we've seen so far. Fraud Detection	*) Enough positive examples for algorithm to get a sense of what positive examples are like, future positive examples likely to be similar to ones in training set. *) Email spam Detection

ProMate

12] Chosing what features to use

*) The success of anomaly detection heavily depends on feature selection

*) Normalize features to similar scales (z-score normalization

*) Use domain knowladge — features that correlate with failures or anomalies

*) Sometimes apply tranformations (log, ratio)

*) Remove irrelevant / noisy features