Module 3

Unsupervised Learning

Overview Of Topics

- Unsupervised Learning clustering and K-means
- Hierarchical Clustering
- Principal Component Analysis
- Density based spacial clustering of application with noise DBSCAN
- Gaussian Mixture Models : Expectation Maximization algorithm for gaussian mixture models

Unsupervised Learning: K-Means and Clustering

K-means Clustering:

what you think *K-means clustering* tries to do?

- Pick **K** initial cluster centers (centroids).
- Assign each data point to its **nearest centroid**.
- Recompute each centroid as the **mean** of its assigned points.
- Repeat until things stop changing (convergence).

What is the meaning of the means part of K-means?
mean stands for average
The “means” refers to taking the **mean (average)** position of the data points in each cluster — that’s how the new **centroid** is found after every iteration.

The K part stands for the number of cluster centers, Centroids

Now how do we pick K?

```
we use the elbow method
```

Hierarchical Clustering

- Cluster analysis Method
- We are trying to build a tree like structure. This structure is called a dendrogram
- It shows how different datapoints group together at different levels of similarity.
- Types:
  - **Agglomerative**- bottom up.
    - start with 2 points
    - merge together till u get 1 big cluster
  - **Divisive**- top down
    - start with 1 big cluster
    - split the cluster till u get each point standing alone

How do we know which clusters are together?
For this we use distance metrics.
Such as:
Euclidean
Manhattan
Cosine Distance

- If we cut the dendrogram at different levels we can get different number of clusters as needed

Example:

Point	Value
A	1
B	2
C	6
D	7

Compute Pairwise distance

Pair	Distance
A-B	1
A-C	5
A-D	6
B-C	4
B-D	5
C-D	1

At first:
`{A}, {B}, {C}, {D}`

The **closest pair** is **A-B (1)** and **C-D (1)**.

Let's merge both pairs.

Now clusters are:
`{A,B}` and `{C,D}`

Compute distance between the two clusters

Average distance between `{A,B}` and `{C,D}` =

$$\frac{(A–C)+(A–D)+(B–C)+(B–D)}{4} = \frac{5+6+4+5}{4} = 5$$

So distance = **5**.

Since only two clusters remain, we merge them at distance 5.
Final cluster = `{A,B,C,D}`

Complete linkage → compact clusters.
Single linkage → long, chain-like clusters.

Principal Component Analysis

- It is a dimensionality reduction technique
- We combine correlated data into newer axis
- We call the newly made axis the principal component

PCA creates **new axes** (directions) in the data — called **principal components** — such that:

- The **first component (PC1)** captures the *most variation* in the data.
- The **second component (PC2)** captures the *next most*, and so on.
- All components are **independent** (uncorrelated).

Think of it like rotating your view of the data so that you're looking **along the direction where it varies the most**.

Step	Meaning
Center data	Remove bias (shift to origin)
Covariance	Find relationships between features
Eigenvectors	Find new directions (principal components)
Eigenvalues	Measure how important each direction is
Projection	Express data in those directions
Dim. Reduction	Keep only top components

Covariance Formula:

```
$Cov(x,y)=\frac{Σ(xi-x¯)(yi-y¯)}{n-1}$
```

🧩 Step 1: Represent your data

Suppose we have this dataset with **2 features** and **3 observations**:

Observation	x_1	x_2
1	2.5	2.4
2	0.5	0.7
3	2.2	2.9

We'll represent it as a matrix:

$$X = \begin{bmatrix} 2.5 & 2.4 \\ 0.5 & 0.7 \\ 2.2 & 2.9 \end{bmatrix}$$

⚖️ Step 2: Standardize (center) the data

PCA depends on variance, so we first **subtract the mean** of each column.

Mean of $x_1 = (2.5 + 0.5 + 2.2)/3 = 1.73$
Mean of $x_2 = (2.4 + 0.7 + 2.9)/3 = 2.0$

Centered data Xc:

$$\begin{bmatrix} 2.5-1.73 & 2.4-2.0 \\ 0.5-1.73 & 0.7-2.0 \\ 2.2-1.73 & 2.9-2.0 \end{bmatrix} = \begin{bmatrix} 0.77 & 0.40 \\ -1.23 & -1.30 \\ 0.47 & 0.90 \end{bmatrix}$$

Step 3: Compute the Covariance Matrix

Covariance measures how the variables vary together:

$Cov(X) = \frac{1}{n-1} X_c^T X_c$

$$Cov(X) = \begin{bmatrix} 1.16 & 1.09 \\ 1.09 & 1.00 \end{bmatrix}$$

Step 4: Find Eigenvalues and Eigenvectors

These tell us the **directions** and **strengths** of variance.

For the above matrix:

- Eigenvalues ≈ 2.03 and 0.14
- Corresponding eigenvectors (normalized):
  - For 2.03 → | 0.736, 0.677 |
  - For 0.14 → | -0.677, 0.736 |

Step 5: Choose the principal components

- The eigenvector with the **largest eigenvalue (2.03)** is **PC1** → the direction of maximum variance.
- The second eigenvector is **PC2**.

Step 6: Project the data

To express data in the new coordinate system:

$$Y = Xc \times W$$

Density based spacial clustering of application with noise DBSCAN

K-Means and **Hierarchical Clustering** struggle when:

- clusters are *not circular* (e.g., crescent or spiral shapes), or
- there's *noise/outliers* in the data

DBSCAN fixes this by defining clusters **based on data density**, not shape

Instead of assuming how many clusters there are, DBSCAN looks for:

- **Dense regions of points** (clusters)
- **Sparse regions** (noise or boundaries)

It uses two parameters:

1. **ε (epsilon)** — neighborhood radius around each point
2. **minPts** — minimum number of points needed to form a dense region

Step 3: How it works

Let's go step by step:

1. **Pick a random point.**
2. Look at all points within distance **ε**.
   - If there are **≥ minPts**, this point is a **core point** — it starts a cluster.
3. All points within that neighborhood are added to the same cluster.
4. Then, expand — for each new point in the cluster, repeat the neighborhood check.
5. Stop when no more points can be added.
6. Points that don't belong to any cluster are marked as **noise**.

Step 4: Types of points

- **Core point** → has ≥ minPts in its ε-neighborhood
- **Border point** → has < minPts, but is near a core point
- **Noise point** → not near any core point

Gaussian Mixture Models

A **Gaussian Mixture Model** assumes that the data is generated from a **mixture of several Gaussian (normal) distributions**, each representing a different cluster

Each Gaussian component has:

- a **mean (μ_k)** — center of the cluster
- a **covariance (Σ_k)** — shape/spread
- a **mixing coefficient (π_k)** — proportion of that cluster in the data

Overall Probability density:

```
$p(x)=\sum_{k=1}^{K}π_kN(x|μ_k,\sum_k)$
```

the weird n type thing means:
the value of the *Gaussian (normal) probability density function* for data point x,
given mean μ_k and covariance Σ_k

Expectation Maximization Algorithm

Goal: find parameters (π_k, μ_k, Σ_k) that maximize log-likelihood

wtf is a gaussian: its the bell-curve thing u learnt in math class (normal distribution)

why is this showing up here: The sum (or average) of many independent random variables tends to be Gaussian — even if the original variables aren't

Goal: What are we trying to do?
We have some data (like numbers or points), but we don't know where they came from.

Laymans terms

Imagine you walk into a classroom and see test scores:

```
Scores: (65, 68, 70, 72, 75, 88, 90, 91, 93, 95)
```

You notice two groups:

One around 70
One around 90
But no one told you who studied and who didn't.

You want to figure out:

How many groups are there?
→ Let's say 2
What's the average score in each group?
How spread out are the scores in each group?
Which student probably belongs to which group?

That's what EM with Gaussians does — quietly, step by step.

Step 1: Make a Guess (Like Solving a Mystery)

You don't know the truth yet. So you start with a guess.

Let's pretend:

Group A (slackers): average = 70, spread = 5
Group B (nerds): average = 90, spread = 5
Half are in each group
These "average" and "spread" define a bell curve — that's a Gaussian.

So now you have two bell curves, one centered at 70, one at 90.

Step 2: E-step – "Who Belongs Where?"

Now look at each student's score and ask:

"If I had to bet, which group is this person more likely from?"

Take the student with score = 72:

It's close to 70 → pretty likely for slackers
Far from 90 → not very likely for nerds
You use math (the Gaussian formula) to compute how likely this score is under each curve.

This gives you:

"Responsibility" of Group A for this student: 80%
Responsibility of Group B: 20%
You do this for every student.

This is the E-step:

Use current guesses about the groups (Gaussians) to estimate soft assignments ("this kid is 80% slacker, 20% nerd").

No hard labels — just probabilities.

Step 3: M-step – "Update Your Belief"

Now reverse the question:

"If these were the true memberships, what should the groups really look like?"

For example:

To find the new average of Group A (slackers), take all students' scores, weighted by how much they belong to Group A.
Same for spread (variance).
So if most low-scorers are 80–100% in Group A, the new average stays near 70 or shifts slightly.

But wait — maybe your first guess was off.

Say someone scored 80 — right in the middle. Originally, you thought it was 50/50. But after seeing others, maybe it leans more toward nerds?

So next time:

You update the bell curves based on these soft votes.
Then recompute who belongs where.
Then update again.

And you keep going…

Now the annoying math

E step

Relative Likelyhood: e raised to (x-mew)/variance

our prior is pi

so expectation maximisation we multiply prior with likelyhood

will give us responsibility

M-Step — “Update the Groups”

Now we update the **mean**, **variance**, and **mixing weight** using the responsibilities.

then repeat the E step