I. Soft Clustering

Soft clustering (also called probabilistic clustering) assigns a probability of membership to each data point for every cluster, instead of forcing a single assignment.

So instead of “this point is in Cluster 1” we say “this point has a 70% chance of being in Cluster 1 and 30% in Cluster 2.” Each point’s probabilities across all clusters sum to 1.

Soft clustering is useful because it provides a more realistic and flexible way to analyze data with overlapping or ambiguous categories, unlike hard clustering where points are forced into single groups.

II. Applying Bayes Theorem to Soft Clustering

Problem statement: given a dataset of weights sampled from N different animals,
can we determine which weight belongs to which animal?

You have a bunch of datapoints with just one variable that is weight, e.g. [100, 120, 150, 500, 520, 550].

You know they come from a few different species (say, Raptors and Stegosauruses),
but you don’t know which number belongs to which species.This is a clustering problem -- but instead of arbitrary clusters, we want to group weights by their likely species.

Output: So instead of saying “this 120 belongs to Species 1” we say “this 120 has a 90% probability of being Species 1 and 10% of being Species 2.”

So for each data point X_i :

We compute $P(S_1 \mid X_i)$, $P(S_2 \mid X_i)$, … for all S (species/clusters).
Meaning ….. find the probability that X_i belongs to S_j for every possible S (cluster/species)

Example Output:
Say we have 3 species (clusters):

- S_1 = Raptor
- S_2 = Stegosaurus
- S_3 = Sauropod

And our dataset has 3 observed weights (like in pounds lb): X = [120, 520, 1000] … the goal is: can we figure out which datapoint (which weight) corresponds to which species (which cluster).

After computing $P(S_j \mid X_i)$ for each point, you might get something like:

Data point (Weight)	P(Raptor)	P(Stegosaurus)	P(Sauropod)
120	0.92	0.07	0.01
520	0.05	0.85	0.10
1000	0.00	0.05	0.95

But how do we get $P(S_j \mid X_i)$???? This is the goal in soft clustering, "given a particular weight X_i, what's the probability that it came from each species S_j". You would **use bayes theorem** to get the probability $P(S_j \mid X_i)$ ……. let's look at what we need:

Bayes Theorem: $$P(S_j \mid X_i) = \frac{P(X_i \mid S_j) \times P(S_j)}{P(X_i)}$$

*1. P(Sj): Some species (clusters) are simply **more common** than others in the dataset*

So before you even look at a specific weight (lb) X_i there's already a **baseline chance** that it came from a certain species.

For example, in Jurassic Park, there might be many Stegosauruses but only a few Raptors. So even without knowing the animal's weight, it's statistically more likely that a random animal is a Stegosaurus than a Raptor.

That's what $P(S_j)$ -- the **prior probability** -- represents.

2. P(Xi | Sj): weights vary within each species

Each species doesn't just have a single "typical weight" …. they have a distribution -- some variation around a mean.

Each species S_j has its own **mean** and **variance** -- (its animal weight distribution).

If you see a weight X_i=500:

- It might be *impossible* for a Raptor (too big).
- Quite *probable* for a Stegosaurus.
- *Possible but less typical* for a Sauropod.

That's exactly what $P(X_i \mid S_j)$ measures -- **how likely** it is to see that weight given the species.

3. P(Xi): total probability of observing an arbitrary datapoint

$P(X_i)$ represents the overall probability of seeing a specific weight (datapoint), no matter which species it came from. It's computed by summing across all possible species:

$$P(Xi) = \sum_j P(Xi \mid Sj) \times P(Sj)$$

Basically: what's the chance this weight (datapoint) would appear if it came from species j, weighted by how common that species is.

4. How Bayes Theorem Fits In

$P(Sj_j)$: prior → how common the species is overall
$P(X_i \mid S_j)$: likelihood → how likely that weight is for that species
$P(X_i)$: the total probability of seeing that weight, no matter which species it came from.

Finally, Bayes' Theorem: combines the prior knowledge (how common each species is) with the likelihood (how well each species explains the observed weight) and normalizes it by the total probability of seeing that weight.

This gives us the **posterior probability P(Sj | Xi)**: the probability that a particular data point Xi belongs to species (cluster) Sj.

In soft clustering, these posterior probabilities become the **soft assignments** — each datapoint has a probability of belonging to every cluster, rather than having a single hard classification label.

III. Mixture Model

Imagine you have a dataset with a few variables (features) — for example:

Person	Height (cm)	Weight (kg)	Age (years)
A	155	55	20
B	165	62	25
C	180	80	35
D	190	90	40
…	…	…	…

A typical dataset (call it ds1) is just a collection of observed data points — you have values for your features (e.g., height, weight, income, etc.), but **you don't assume** how those points were generated. You don't assume any particular underlying process produced the datapoints — they're just data.

A mixture model adds a generative assumption: it **looks at the typical dataset** (i.e. ds1) BUT assumes that your dataset was "generated" by several hidden groups (clusters), and that each group generates data according to its own probability distribution.

Formally: $P(X) = \sum_{j=1}^{k} P(S_j) \times P(X \mid S_j)$

- Each hidden group (cluster) Sj has its **own distribution** P(X | Sj) describing how its data points are spread out.
- P(Sj) tells you how common each cluster is overall.
- The entire dataset's distribution P(X) is the **weighted blend** of all those cluster distributions.

So instead of one "big blob" of data, you assume your dataset is the combined result of several smaller distributions, each describing one group's typical pattern.

Regular clustering (like K-means) simply groups points based on distance or similarity — no assumption about how data was generated. Mixture models go further: they assume there is a probabilistic process that "produced" the data.

What happens under the hood…… when we fit a mixture model to this data:

We **don't know** which data point belongs to which cluster. Model tries to **learn**:

1. Each cluster's distribution parameters, which depend on the type of distribution used in the mixture model. *(e.g., Gaussian mixture model uses mean and covariance) ……..... finding the distribution parameters for a cluster means finding* P(X | Sj) (the clusters probability distribution).

2. Each cluster's overall probability P(Sj) (how common that cluster is in the dataset)

AFTER we fit (train) the model ……..... **the result is a probabilistic model** of the data that describes: the **distribution parameters** for each cluster (how each cluster looks statistically), and the **mixing proportions** P(Sj) — how frequent each cluster is overall.

FINALLY……. **the resulting model** can find the probability that a specific data point belongs to each cluster (soft assignment) …… it now knows P(Sj | Xi) for every datapoint.

IV. Gaussian Mixture Model

A Gaussian Mixture Model (GMM) is a type of mixture model where each cluster's distribution P(X | Sj) is assumed to follow a Gaussian (Normal) distribution.

This means we're assuming the data was generated by several **Gaussian-shaped clusters**, each with its own mean and spread.

When we fit a GMM, we learn:

1. The Gaussian parameters of each cluster
    a. Each cluster's **mean** μj — where that Gaussian is centered.
    b. Each cluster's **covariance Σ**j — how wide or stretched the cluster is.
2. Each cluster's mixing proportion $P(Sj)$ — how common the cluster is in the dataset.

We still don't know which data points belong to which cluster — that's learned probabilistically using soft assignments (GMM clustering).

V. Maximum Likelihood Estimation (MLE)

MLE chooses the parameter values that maximize the probability (likelihood) of seeing the data we actually observed.

Intuitive explanation: "out of all possible versions of this model, which one would most likely have generated the data we actually observed?"

…….so if you assume your data follow some distribution (like a Gaussian), MLE finds the mean and variance of that Gaussian that make your dataset most likely. Ex: If your data points are all near 10, the MLE for μ will be close to 10.

How it applies to mixture models: in a mixture model (like a GMM), we use MLE to find the parameters that maximize the overall likelihood of the dataset.

VI. GMM Clustering

Goal: find the Gaussian Mixture Model (GMM) that maximizes the probability (likelihood) of seeing the data we actually observed ….. which is P(Xi)

Remember the total likelihood of observing a datapoint $P(Xi) = \sum_j P(Xi \mid Sj) \times P(Sj)$

What parameters define a GMM? **……** each cluster Sj in the model has:

- Mean μj — the center of the Gaussian
- Covariance Σj — how spread or stretched the cluster is
- Mixing weight P(Sj) — how common that cluster is overall

Collectively, these parameters are written as:

Θ = {μ1, …, μk, Σ1, … , Σk, P(S1), ..., P(Sk)}
for k components (clusters/gaussian distributions)

If we assume data points X1,X2,...,Xn are independent, the total likelihood of observing the dataset is:

—----------

$$P(X \mid \Theta) = \prod_{i=1}^{n} \sum_{j=1}^{k} P(Xi \mid Sj) \times P(Sj)$$ (for n datapoints)

—----------

Each term P(Xi | Sj) is given by the Gaussian distribution for cluster j (based on that cluster's parameters, which are a subset of Θ. P(Sj) for that cluster is also found in Θ.

This expression represents a mixture of Gaussians, where the probability of seeing each data point is the sum over all clusters, weighted by their mixing proportions.

NOW……. we want to find parameter values Θ that **maximize** P(X | Θ)
HOWEVER …….. this equation can't be solved directly — because the cluster memberships (which datapoint belongs to which Gaussian) are unknown.

To handle this, we use the log-likelihood ….. taking the log makes it easier to work with (and doesn't change the critical points).

$$log\ P(X \mid \Theta) = \sum_{i=1}^{n} log\Big(\sum_{j=1}^{k} P(Xi \mid Sj) \times P(Sj)\Big)$$

FINALLY ….Solving with the Expectation-Maximization (EM) Algorithm
The EM algorithm is used to fit (train) the Gaussian Mixture Model — meaning, to find the best parameters (μ, Σ, P(Sj)) that maximize the likelihood of the data.

Because we don't know which datapoint belongs to which cluster, EM alternates between guessing cluster memberships and updating cluster parameters — refining both until they stabilize.

Expectation Maximization Algorithm Steps (in brief):

1. Initialize: start with random guesses for cluster means (μ), covariances (Σ), and mixing weights P(Sj).

2. E-step (Expectation): compute the probability P(Sj | Xi) that each datapoint Xi belongs to each cluster Sj using the current parameters.

3. M-step (Maximization): update μ, Σ, and P(Sj) using those probabilities (weighted averages of data).

4. Repeat: alternate E and M steps until the parameters stop changing (model converges).