Content

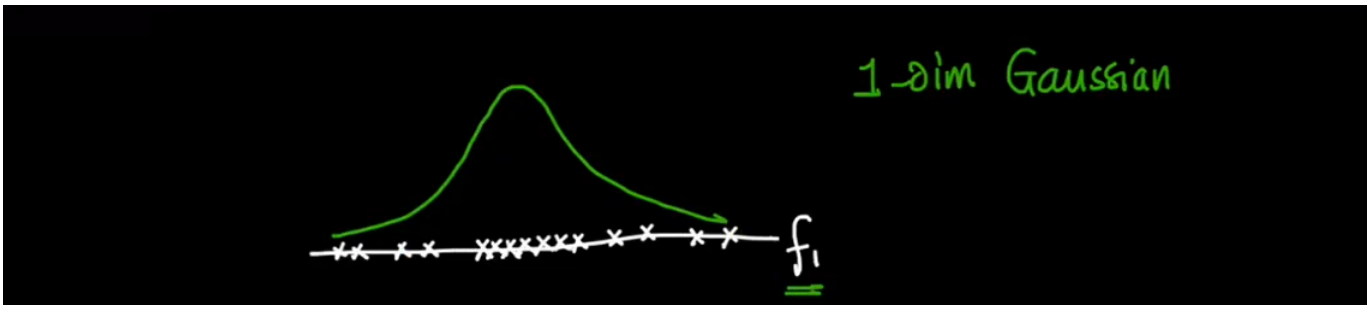
- Introduction to Gaussian Mixture Models(GMM)
- Extending GMMs to multi-dimension
- GMM Algorithm & Implementation

Introduction to Gaussian Mixture Models(GMM)

Let's start with very basic concepts of Gaussian.

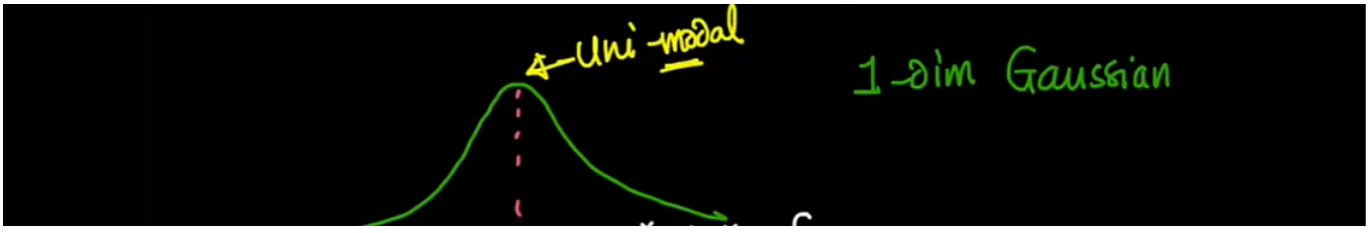
Imagine that we have a feature $f_{1}$$f_{1}$, and we're having dense datapoints in the center, and sparse data at the start and at the end. What type of distribution feature $f_{1}$$f_{1}$ might have?

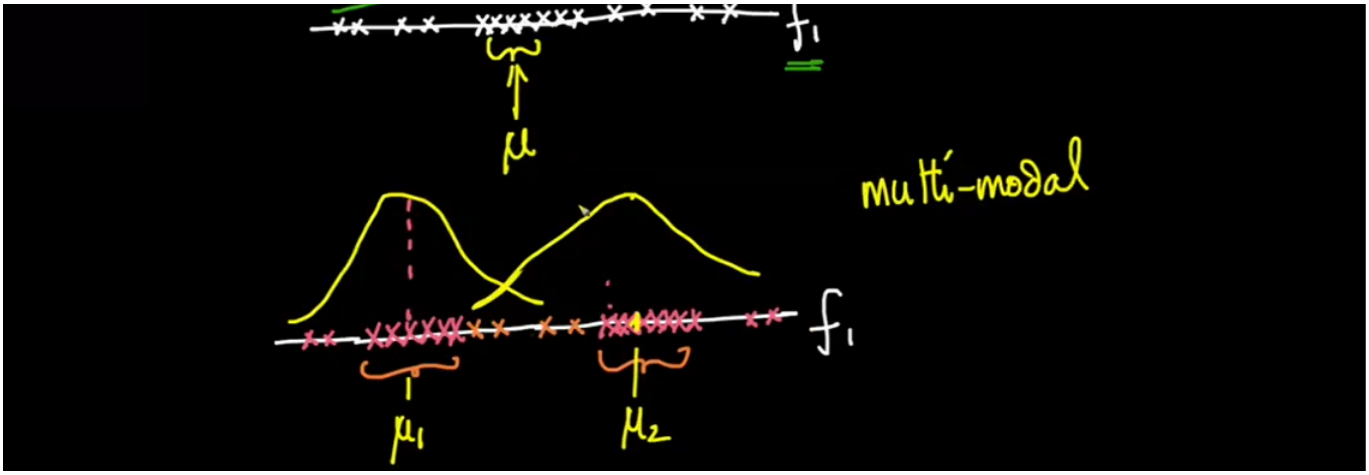
Normal Distribution!! This is known as 1-Dimensional Gaussian Distribution



> **Now, what if the pattern of the spread repeats two times?**

- In that case, there will be two **Gaussian Distributions** with mean $\mu_{1}$$\mu_{1}$ and $\mu_{2}$$\mu_{2}$.
- If there is only a single **mount(peak)** in the distribution of a data, then the data is known as **uni-modal data**
- If there are more than one **mount(peak)** in the distribution of a data, then the data is known as **multi-modal data**

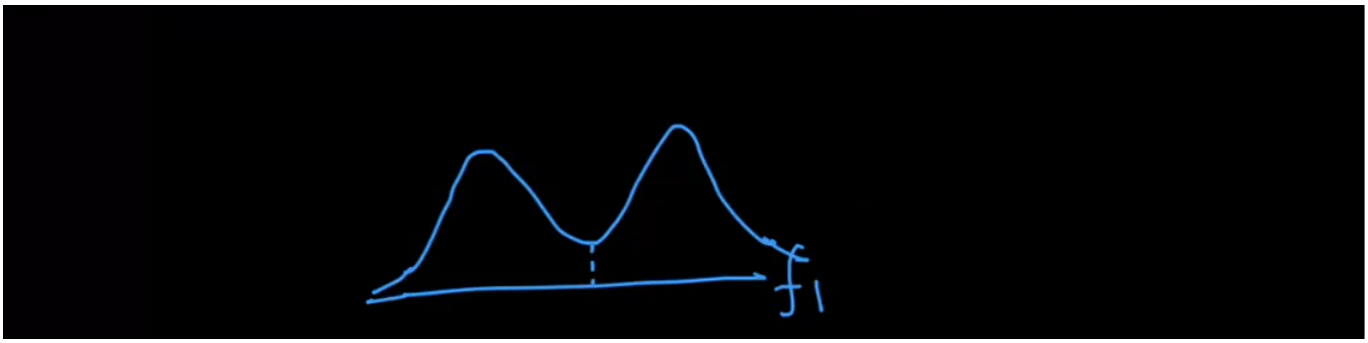




If we combine both the gaussians, the point at which both distributions intersects will have more height than the start and the end of the distribution.

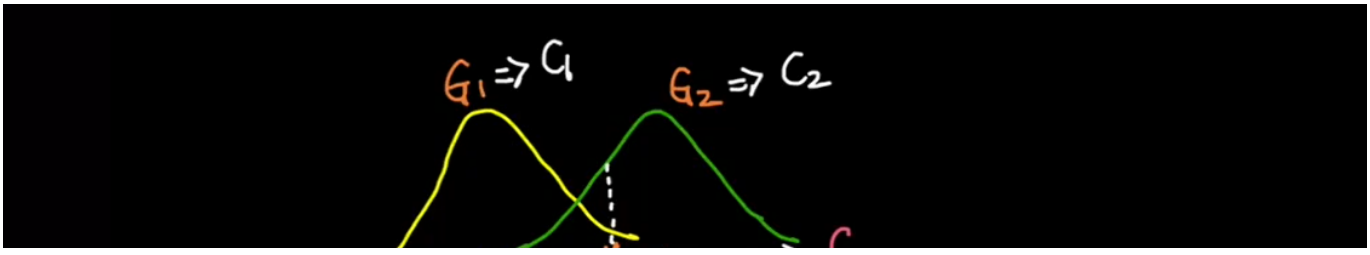
This is known as **Mixture of Gaussians**

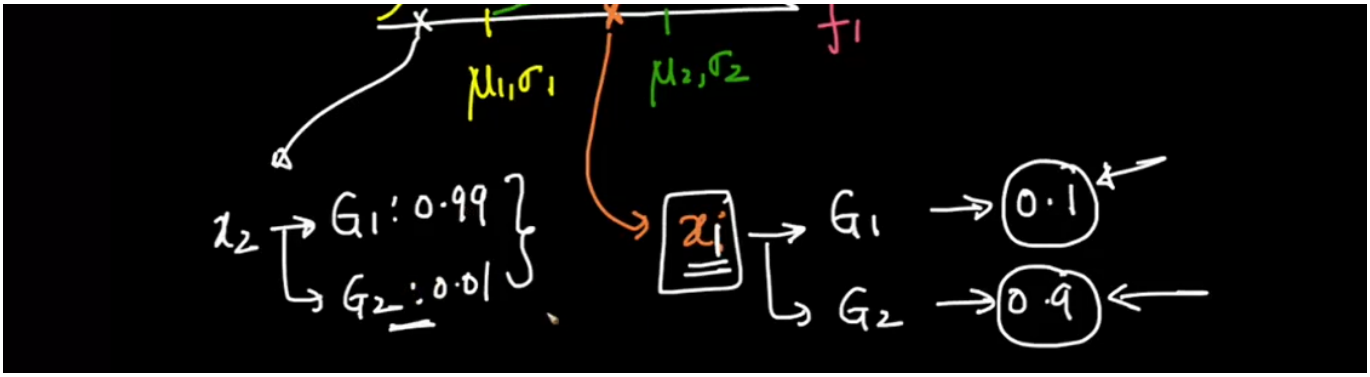
> Q. What does the intersection represents?
>
> - It says that the point at intersection can either belong to distribution G_{1} G_{1} or distribution $G_{2}$$G_{2}$



Now, for a feature $f_{1}$$f_{1}$, suppose there are two gaussian distribution $G_{1}$$G_{1}$ and $G_{2}$$G_{2}$, with mean values as $\mu_{1}$$\mu_{1}$ and $\mu_{2}$$\mu_{2}$, and standard deviations as $\sigma_{1}$$\sigma_{1}$ and $\sigma_{2}$$\sigma_{2}$ respectively.

If we consider a point $x_{i}$$x_{i}$, it could belong to either $G_{1}$$G_{1}$ and G_{2} G_{2}, but the probability of that point will be higher for either one of them, based on its coordinate





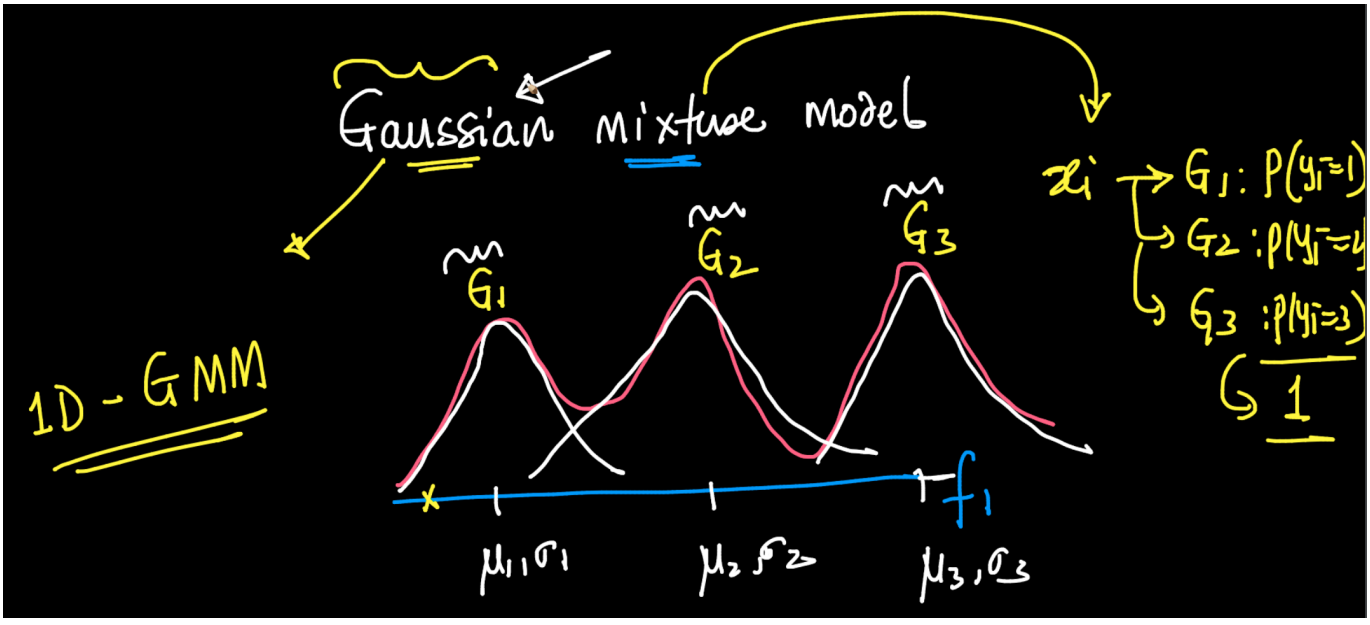
So, in **Gaussian Mixture Models**, we are trying to represent any data as a combination of Gaussian distributions and it is called a mixture model, because each point could belong to any of the gaussian with some probabilities, such that sum of probabilities sum up to 1

Intuition behind Gaussian Mixture Model

- The basic intuition behind Gaussian Mixture Model is that **we can model any data, that we have, as a mixture of Gaussians**.

Let's take 1-Dimensional data again, i.e., data points having only 1 feature

- This time let's say that data points are distributed among **'k' Gaussians**, where each Gaussian has a different Mean ($\mu_k$$\mu_k$) and Standard Deviation ($\sigma_k$$ \sigma_k$), as shown below:



- Every data point $x_i$$x_i$ will have a probabilty of belonging to each of the Gaussians, i.e.,

  $P(y_i = 1), P(y_i = 2), P(y_i = 3), ...$$P(y_i = 1), P(y_i = 2), P(y_i = 3), ...$

where, 1, 2, 3 ... are the indices of Clusters (Gaussians)

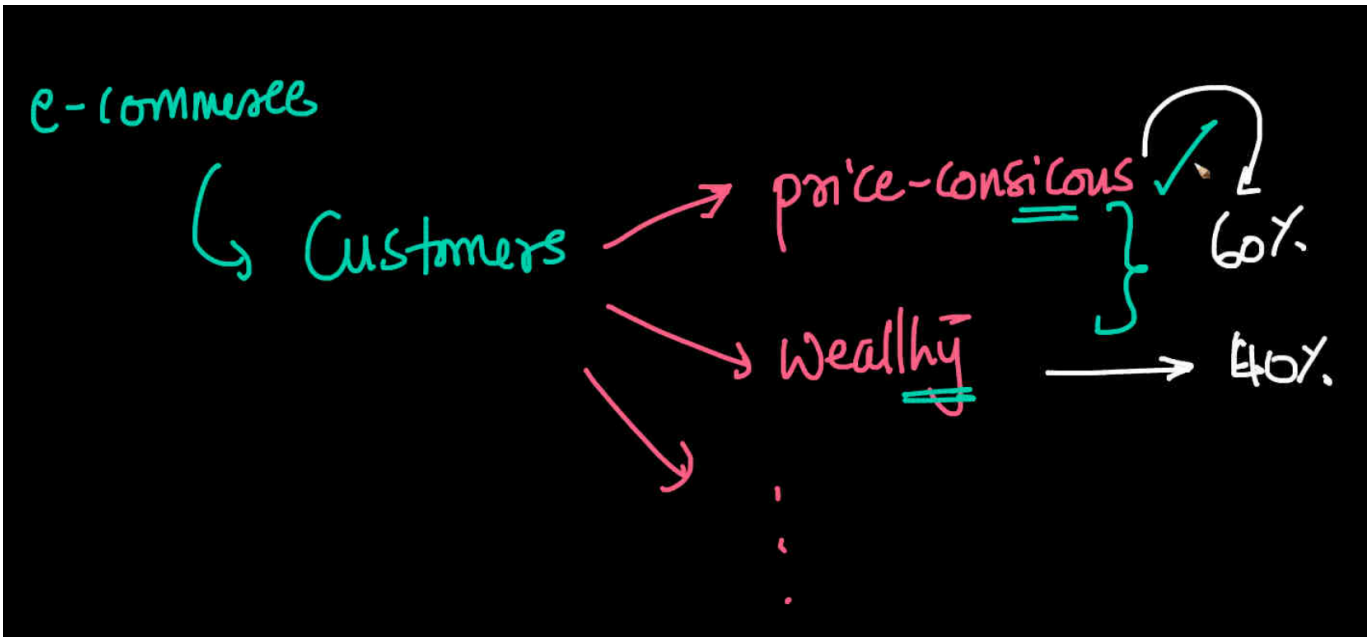
- The sum of these probabilities will be 1, i.e,

  $P(y_i = 1) + P(y_i = 2) + P(y_i = 3) + ... = 1$$P(y_i = 1) + P(y_i = 2) + P(y_i = 3) + ... = 1$

- So, we are representing data as a combination of Gaussian distributions. Each data point belongs to each of the Gaussians with a non-zero probability, no matter how small that probability is.
- This is a 1-Dimensional Gaussian Mixture Model (GMM).

Let's take the example of an E-Commerce Website

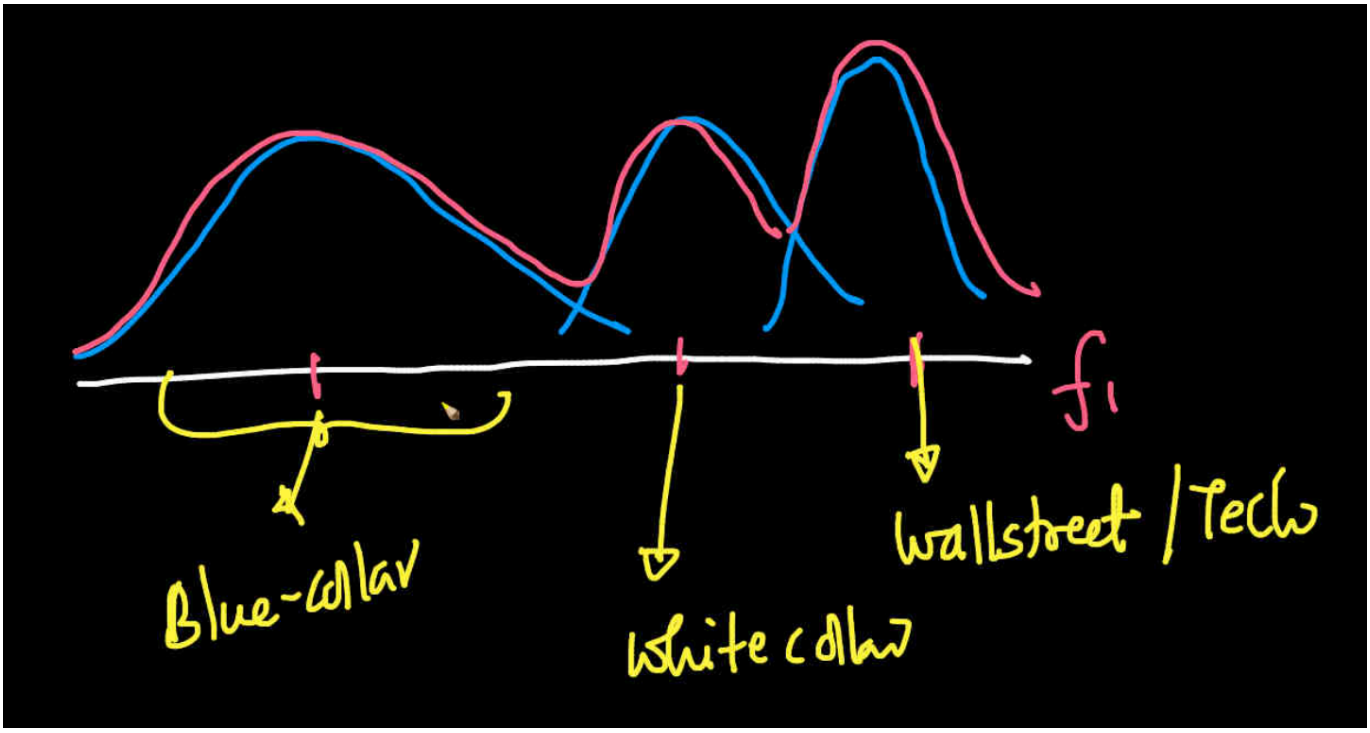
- As we saw earlier, E-Commerce Websites do **Customer Segmentation** to increase their sales/revenue.
- Customers are clustered into different groups based on similar traits.
- It is possible that not all customers will solely belong to a single cluster.



- For example, a customer can be both wealthy and still be price-conscious. Let's say, he/she can fall in the "wealthy" cluster with a probability of 40%, but in the "price-concsious" cluster with a probability of 60%.
- So, we can create a Gaussian Mixture Model with such customers belonging to different clusters with different probabilities.

An important thing to note is that each Gaussian Distribution in the mixture can have different spreads

- That is, each Gaussian Distribution in the GMM can have different Mean and Standard Deviation.
- It is not necessary that all distributions in the Mixture Model will have to look same.

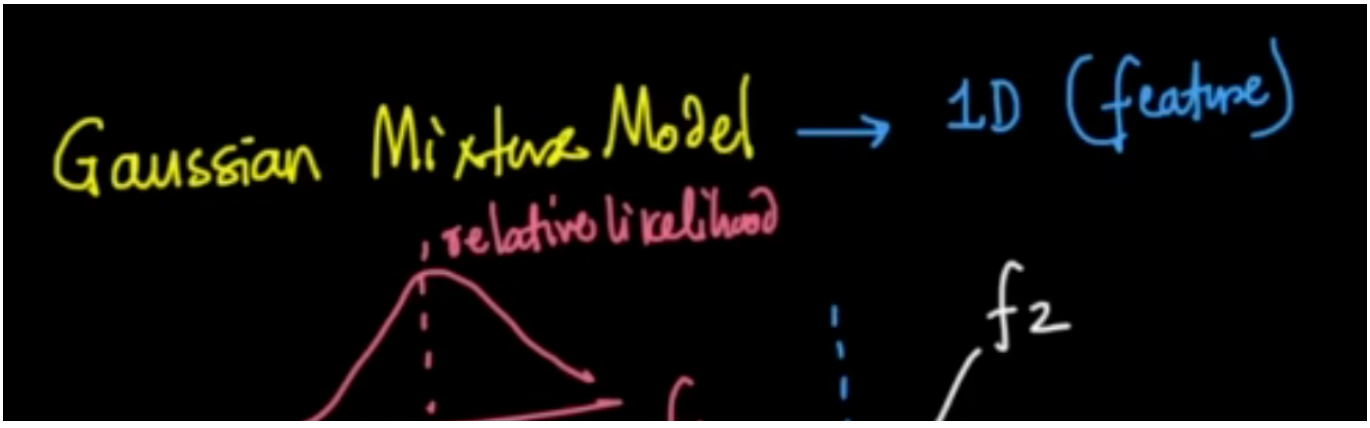


Extending GMMs to multi-dimension

- So far we have seen GMMs with only 1 feature - **1-D Gaussian Mixture Models**

Q. **What if our data has more than 1 features?**

- 2 features having continuous values can be thought of being on a plane with each data point as a point on the plane.
- The probability of a point belonging to each combination of those 2 continuous variables will come on the 3rd axis (z-axis).

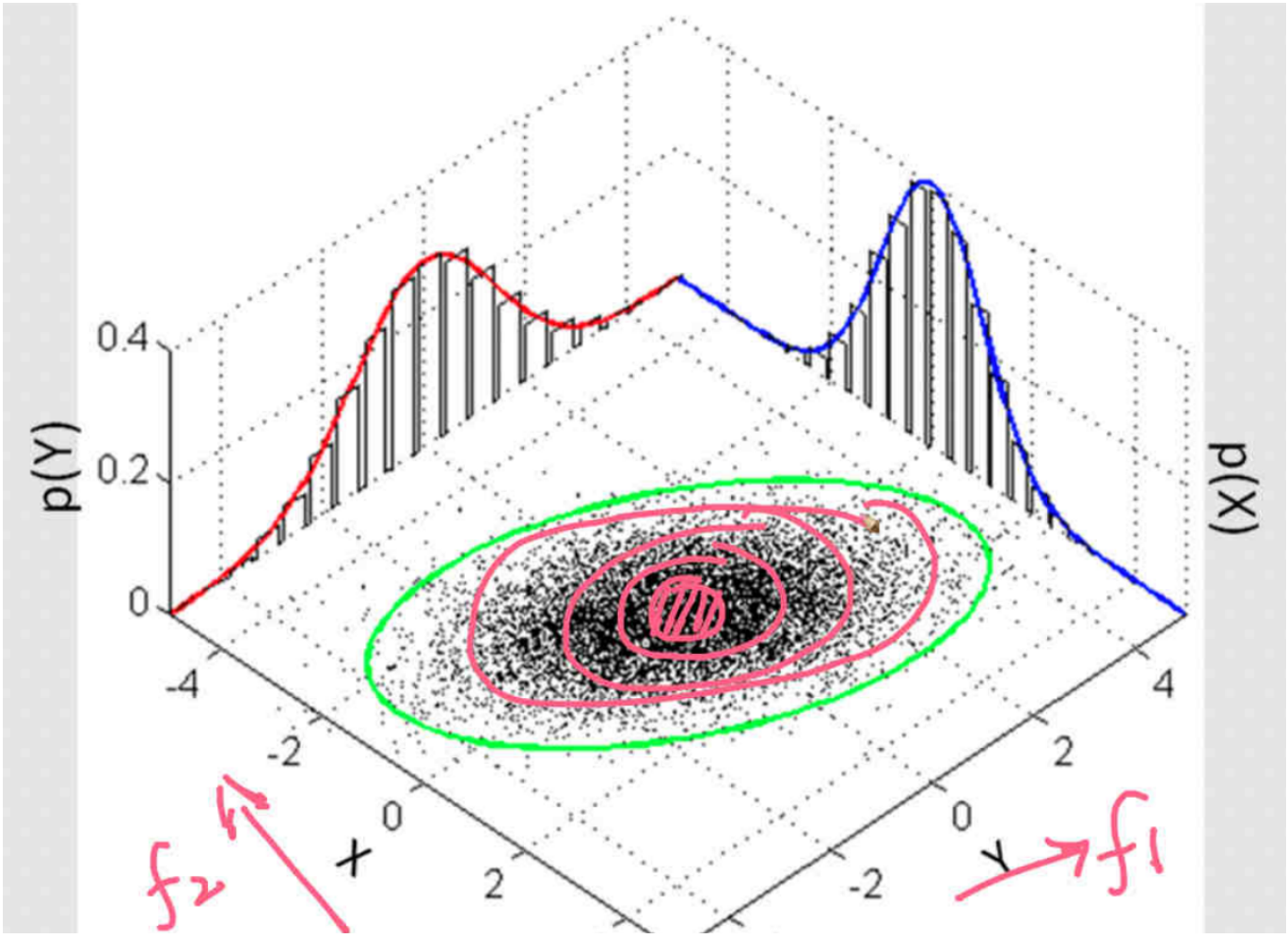


- The Gaussian Distribution would look like a hill.

> **Instructor Notes**
>
> - Use this visualizer tool for making them familiar with normal distribution with 2 features
> - http://socr.ucla.edu/htmls/HTML5/BivariateNormal/
> - https://demonstrations.wolfram.com/TheBivariateNormalDistribution/

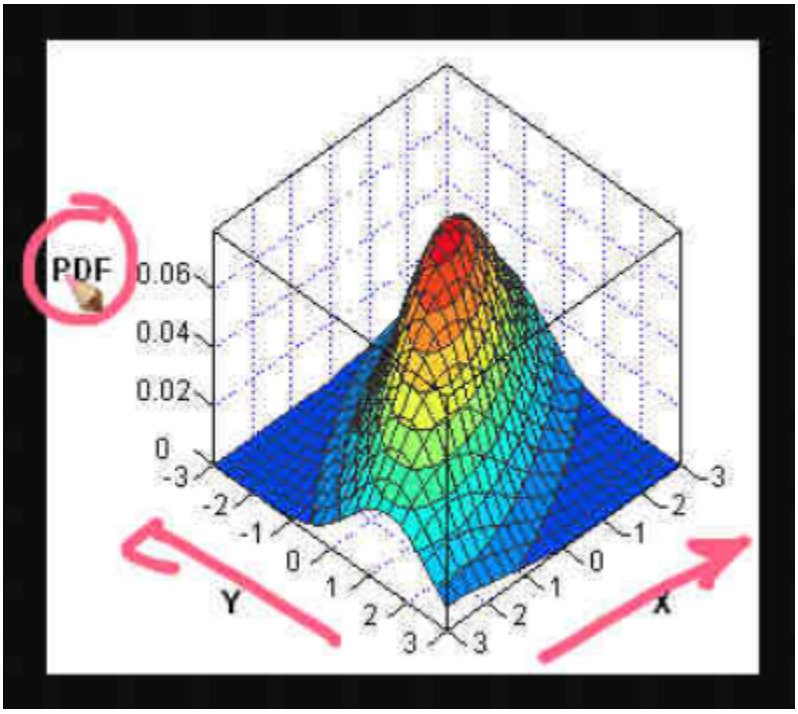
Example Gaussian Distribution with 2 Features

- Let's take the below Gaussian Distribution with 2 Features as an example:



4 -4

- As we can see, the data points are more dense in the centre and as we move away from the centre, the density of points decreases. This is to simulate a Gaussian (Normal) Distribution.
- If we consider only 1 feature at a time, we can see its Gaussian Distribution:
  - Red curve represents the Gaussian Distribution for Feature 1 alone.
  - Blue curve represents the Gaussian Distribution for Feature 2 alone.
- The Gaussian Distribution of the 2 features combined would look something like shown below:



- This is a 2-D Gaussian where the peak is in 3-D.

For a D-dimensional or Multivariate data, the Gaussian Distribution will be in D+1 dimensions.

1D GMM vs multi-dimensional GMM

- A **1-D (having only 1 feature) Gaussian Distribution** can be represented as:

  $N(\mu, \sigma)$$N(\mu, \sigma)$

  where, $\mu$$\mu$ is the mean and $\sigma$$\sigma$ is the Standard Deviation of the Gaussian Distribution

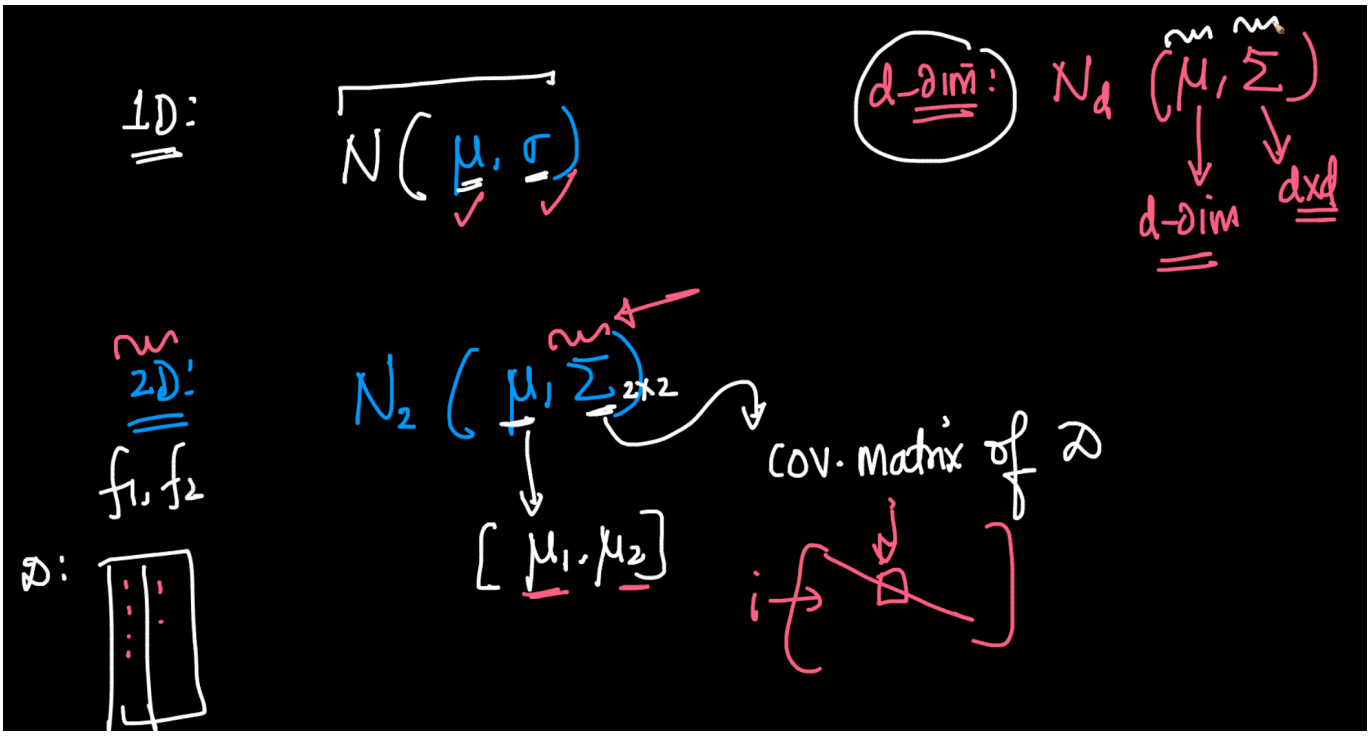
- A **2-D (having 2 features) Gaussian Distribution** can be represented as:

  $N_2(\mu^{(2)}, \sum^{(2 \times 2)})$$N_2(\mu^{(2)}, \sum^{(2 \times 2)})$

Now here,

$\mu^{(2)}$$\mu^{(2)}$ is not just a mean, it is a **vector of means $[\mu_1, \mu_2]$ $[\mu_1, \mu_2]$**, where $\mu_1$$\mu_1$ and $\mu_2$$\mu_2$ are the means of features $f_1$$f_1$ and $f_2$$f_2$

And, $\sum^{(2 \times 2)}$$\sum^{(2 \times 2)}$ is the **$2 \times 2$$2 \times 2$ Covariance Matrix** of the whole data.



- Mean ($\mu$$\mu$) and Standard Deviation ($\sigma$$\sigma$) of 1-D Gaussian Distribution are scaler values, whereas for 2-D Gaussian Distribution, we have a vector of Means (**$\mu^{(2)}$$\mu^{(2)}$**) and a $2 \times 2$$2 \times 2$ Covariance Matrix ($\sum^{(2 \times 2)}$$\sum^{(2 \times 2)}$).

Now, What about a d-Dimensional Gaussian Distribution?

- A **d-Dimensional Gaussian Distribution** can be represented as:

  $N_d(\mu^{(d)}, \sum^{(d \times d)})$$N_d(\mu^{(d)}, \sum^{(d \times d)})$

  Here, $\mu^{(d)}$$\mu^{(d)}$ is a d-Dimensional vector of means of all d-features, i.e., $[\mu_1, \mu_2, \mu_3, ... \mu_d]$$[\mu_1, \mu_2, \mu_3, ... \mu_d]$ And $\sum^{(d \times d)}$$\sum^{(d \times d)}$ here is a $d \times d$$d \times d$ Covariance Matrix of the data.

- So, a high dimensional Gaussian Distribution can be mathematically represented in terms of Vector of Means and Covariance Matrix of the data.

Q. How determine these values of $\mu$$\mu$ and $Σ$$Σ$ in higher dimensions?

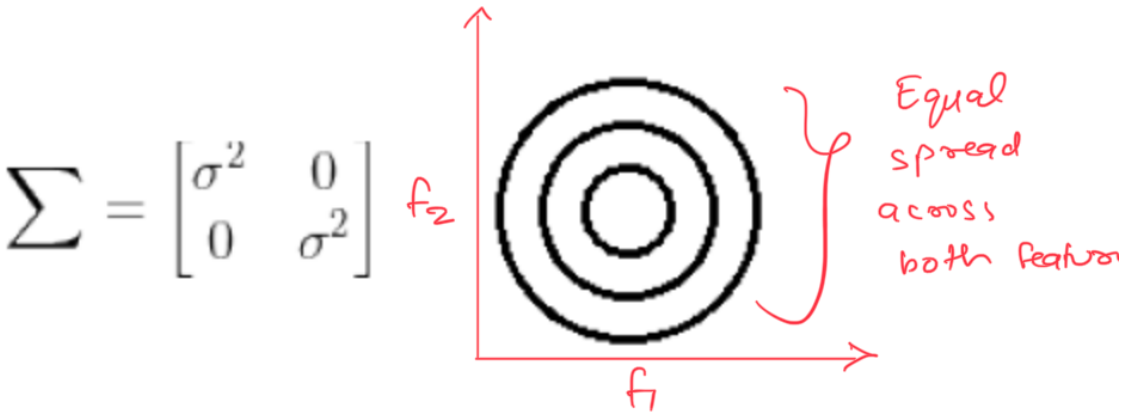
- For 1D data(with only one feature), it fairly simple. We calculate mean and standard deviation of the distribution.
- In 2D, for feature $f1$$f1$ and $f2$$f2$, we calculate per feature mean ($\mu_1$$ \mu_1$, $\mu_2$$\mu_2$), and we compute covariance matrix for the whole dataset which will be of dimension $(2 * 2)$$(2 * 2)$
- Same idea can be extended to the d-dimensions as well.

Q. How do we form clusters with some means and covariance matrix?

- Consider that we have a 2-D feature set, which will return a $(2*2)$$(2*2)$ covariance matrix. The covariance matrix will look something like:

\begin{bmatrix} σ^2 & 0\\ 0 & σ^2 \end{bmatrix}\begin{bmatrix} σ^2 & 0\\ 0 & σ^2 \end{bmatrix}

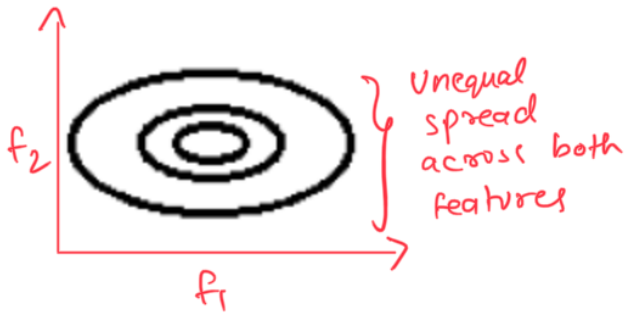
- The diagonal term represents the variance of Gaussian Distribution in each direction of the axis. If both the diagonal elements are equal, the clusters shape would result into a circle. This also means that the covariance between both features will be close to zero.(non-diagonal elements)
- So, the covariance matrix is capturing the per feature standard deviation and how each of the feature are correlated.



Covariance matrix with equal variance (Image by Author)

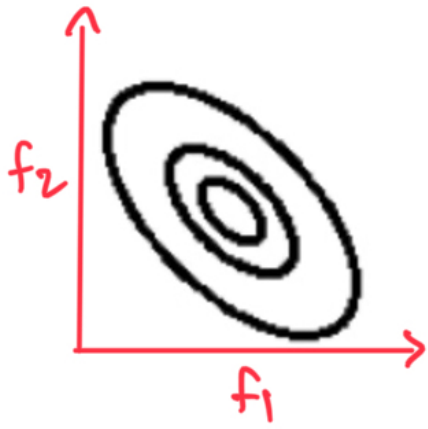
- If the dimension of the variance is changed, the circular shape will be converted into an ellipse.

$$\sum = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$$

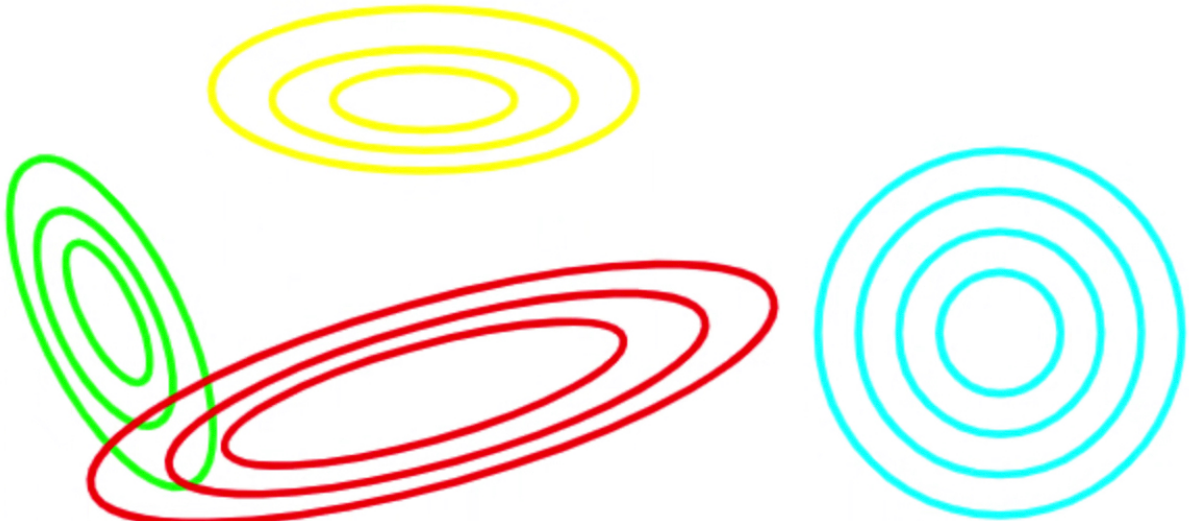


- The above shown clusters were aligned in a regular X-Y coordinate system. To change the orientation from regular axis alignment, we add the correlation terms in the off-diagonal positions.

$$\sum = \begin{bmatrix} \sigma_1^2 & \sigma_{1,2} \\ \sigma_{2,1} & \sigma_2^2 \end{bmatrix}$$

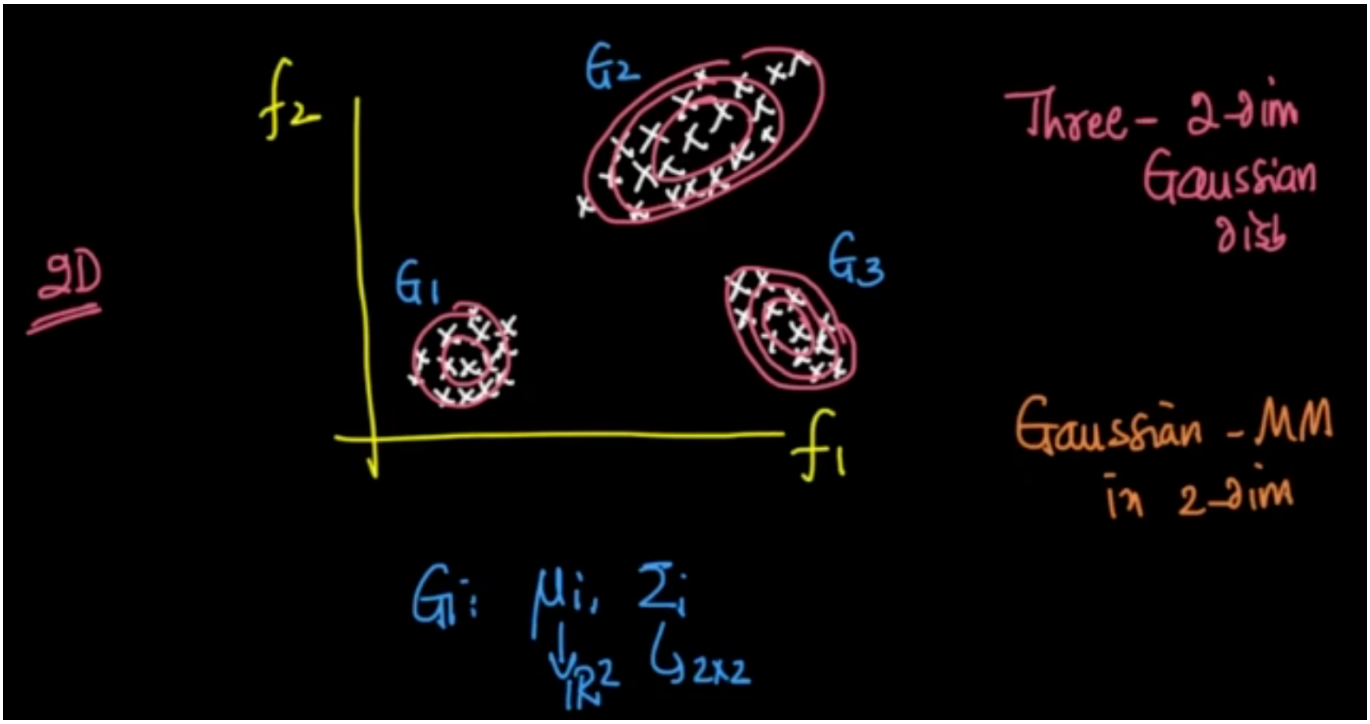


- Now finally, how is the positions of clusters varied across the multiple dimensions?
- For that we make use of mean vector ($μ$$μ$)



Multi-Modal Gaussian Distributions in Multiple Dimension (Mixture Models)

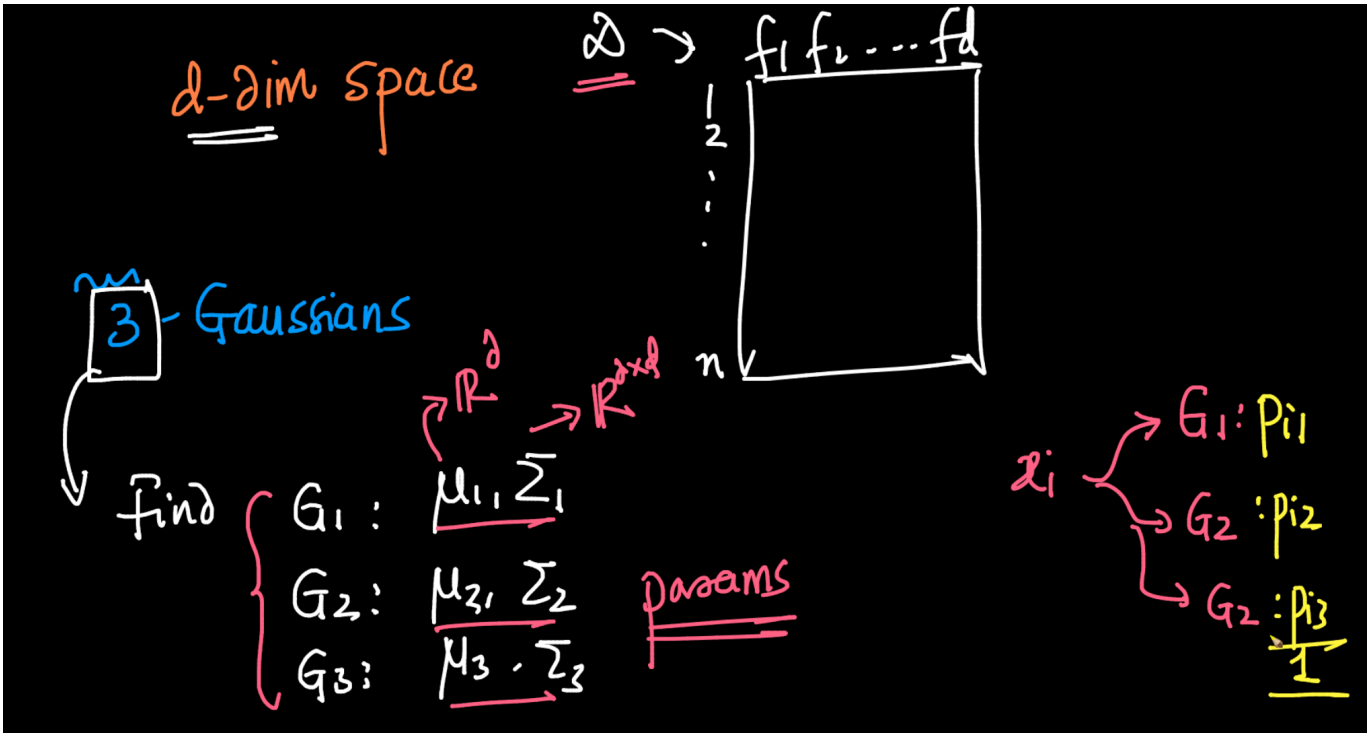
- So far, we studied only Multi-Dimensional Data with a single Gaussian Distribution, i.e., there was only 1 peak (hill) in the data.
- Now, Let's extend this concept to a **Mixture of Gaussian Distributions in Multi-Dimensional Data**.
- Here's an example of a **Three 2-Dimensional Gaussian Distributions**:



- If we look at it, the data points are concentrated at 3 different locations in the 2-D plane. Therefore, the data will have 3 peaks with different distributions → 3 Gaussian Distributions.

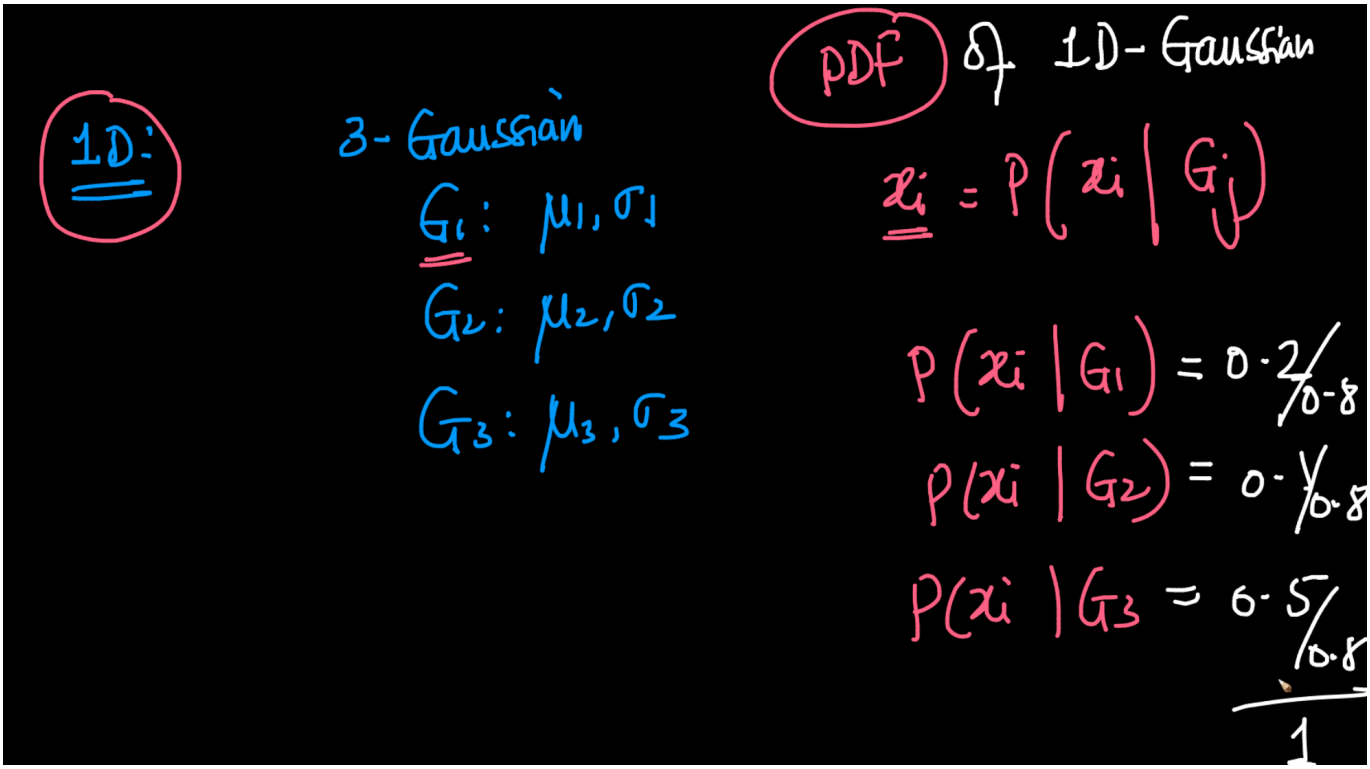
What if we want to model a d-Dimensional 3-Gaussian Mixture Model?

- In order to represent a d-Dimensional data, i.e., data having d-features, as a 3-Gaussian Mixture Model, we'll need to find all 3 Gaussian Distributions.
- And to find all 3 Gaussian Distributions, for each Gaussian Distribution, we'll need to find:
  - The d-Dimensional vector of means $\mu^{(d)}$$\mu^{(d)}$
  - The $d \times d$$d \times d$ Covariance Matrix $\sum^{(d \times d)}$$\sum^{(d \times d)}$
- Each of the 3 Gaussians will have these as its parameters.



For three 1-D Gaussian Mixture Model, we'll need:

- Mean ($\mu$$\mu$) of all 3 Gaussian Distributions.
- Standard Deviation ($\sigma$$\sigma$) of all 3 Gaussian Distributions.



- Given this information, we can find the probability of any data point $x_i$$x_i$ belonging to all the 3 Gaussians
- That is, $P(x_i | G_1), P(x_i | G_2), P(x_i | G_3)$$$P(x_i | G_1), P(x_i | G_2), P(x_i | G_3)$

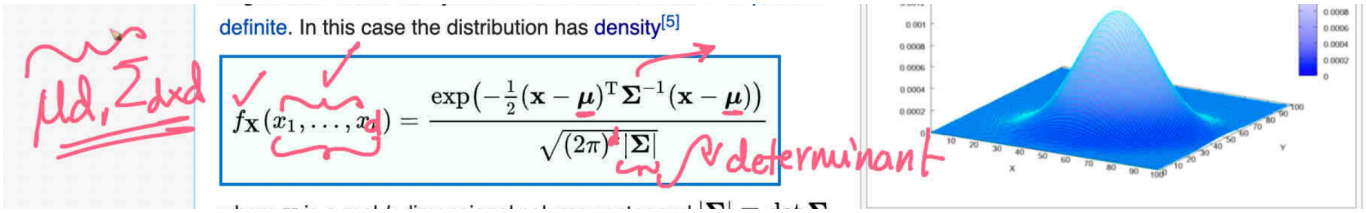
- That is, $P(x_i | G_1), P(x_i | G_2), P(x_i | G_3)$$P(x_i | G_1), P(x_i | G_2), P(x_i | G_3)$

- We can just plug the data point $x_i$$x_i$ in **Probability Density Function (PDF)** and compute the probability of the data point belonging to each Gaussian.
- The Probability Density Function gives the distribution of probabilities of data points belonging to each of the Gaussians.

PDF	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

Now similarly, **For a d-Dimensional Gaussian Mixture Model**, we'll need:

- The d-dimensional vector of means $\mu^{(d)}$$\mu^{(d)}$ and the $d \times d$$d \times d$ Covariance Matrix $\sum^{(d \times d)}$$\sum^{(d \times d)}$, i.e., the Normal Distribution $N_d$$N_d$.
- If we have this Normal Distribution $N_d$$N_d$, we can compute the probability of a data point belonging to a Gaussian (Cluster) using a slightly more complex density function:



- This is the **Probability Density of the d-Dimensional vector**.

(The mathematical derivation of this formula is beyond the scope of this lecture)

- If you see, calculating this expression is very compuatationally expensive because of the terms $Σ^{-1}$$Σ^{-1}$ and $|Σ|$$|Σ|$.
- Also, this will have a lots of local minima, which is why Gradient Descent Fails here.
- So, to solve this, the **Expectation-Maximization** comes into play.

Expectation-Maximization Algorithm

- Let θ denote the parameters:
  - Probability of point $P$$P$ belonging to $j^{th}$$j^{th}$ cluster
  - Mean Vector $\large(μ_{d*d}^j)$$\large(μ_{d*d}^j)$
  - Covariance Matrix $\large(Σ_{d*d}^j)$$\large(Σ_{d*d}^j)$

- So, $\largeθ$$\largeθ$ = $[P_j, (μ_{d*d}^j), (Σ_{d*d}^j); ∀_j = 1 → k]$$[P_j, (μ_{d*d}^j), (Σ_{d*d}^j); ∀_j = 1 → k]$
- So, there are two steps here:
    1. Expectation
    2. Maximization.
- The high-level idea here is that:
    - In Expectation Step, we'll fix the mean vector and covariance matrix, and update probabilities.
    - In Maximization Step, we fix the probabilities and update the mean vector and covariance matrix.

To make this more clear, we'll compare EM with a concept called **Coordinate Ascent** approach.

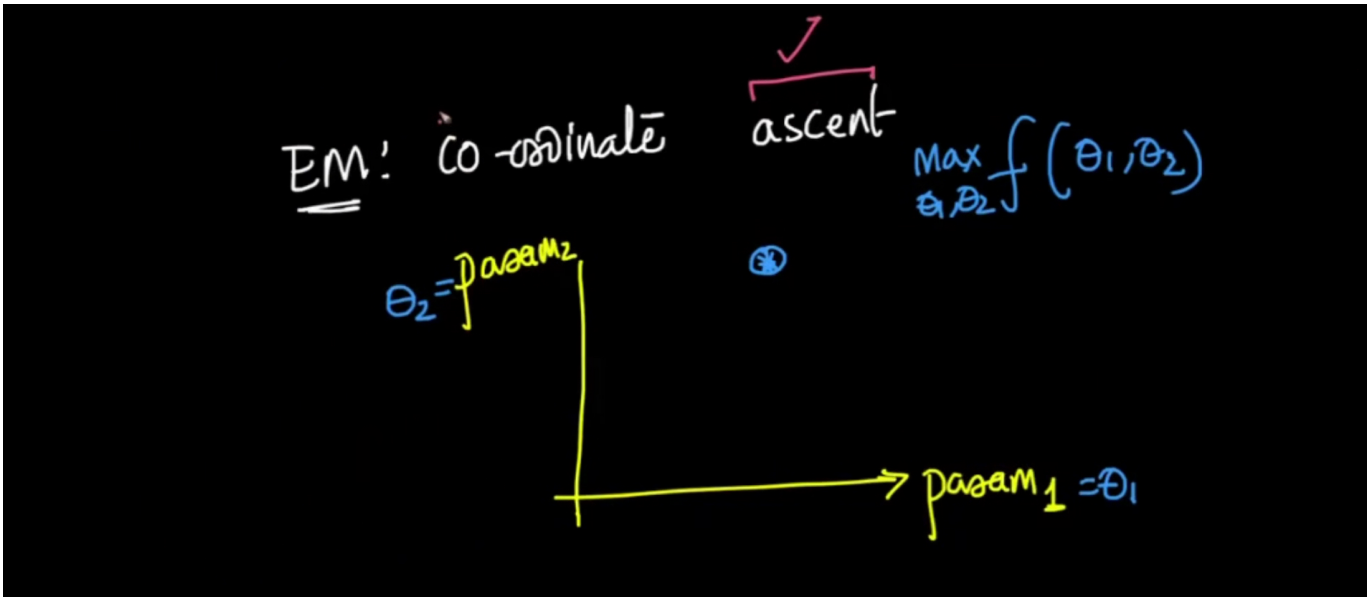
https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/016/181/original/ezgif-5-38f1772929_%281%29.mp4?1665485304

Coordinate Ascent

- EM algorithm is a special type of **coordinate ascent** based approach.

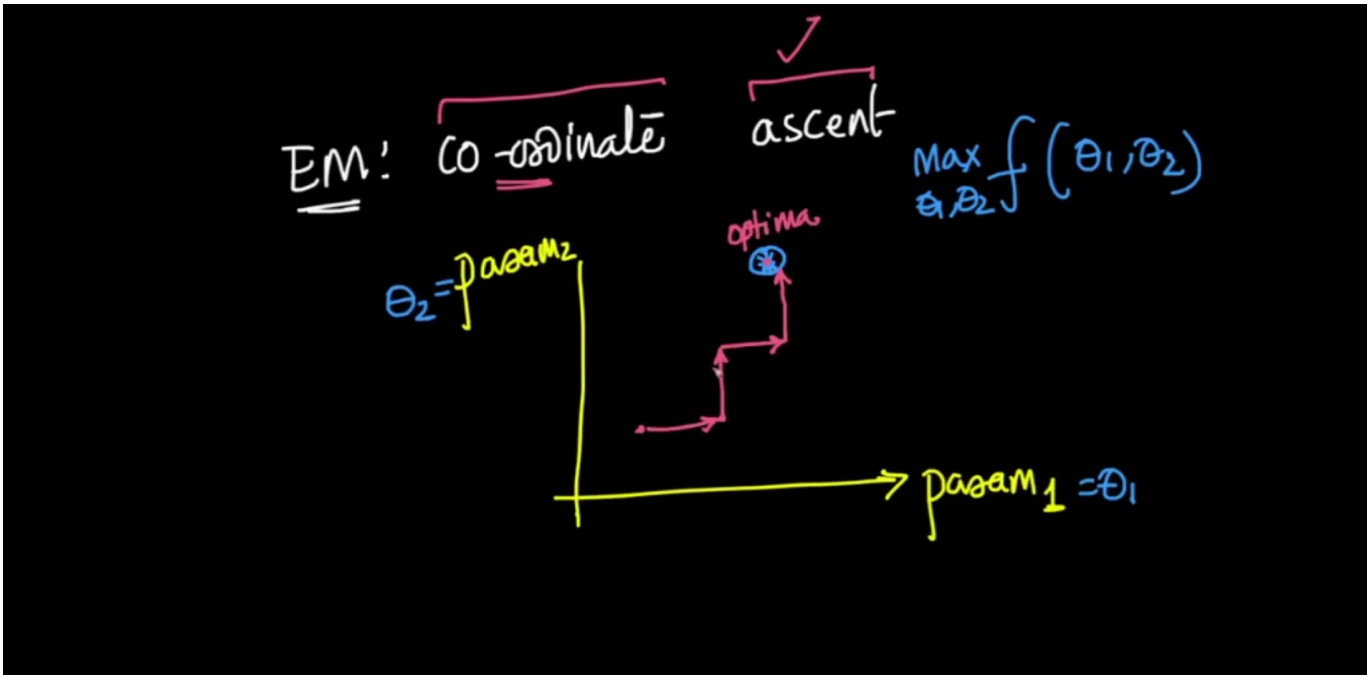
This is the core idea of **Coordinate Ascent** approach:

- Suppose there are two parameters $θ_1$$θ_1$ and $θ_2$$θ_2$ that we are trying to optimize and there is some optimal point.
- To reach that point, we maximize $θ_1$$θ_1$ and $θ_2$$θ_2$ for some function $f(θ_1, θ_2)$$f(θ_1, θ_2)$



Suppose you're starting optimizing at some random values of $θ_1$$θ_1$ and $θ_2$$θ_2$.

- What coordinate ascent will do is it will fix one of the parameters (say $θ_1$$θ_1$) and will try to move to the optimal point by updating another parameter ($θ_2$$θ_2$).
- In the next step it will fix $θ_2$$θ_2$ and will update $θ_1$$θ_1$ for moving closer to the optimal point.



- In EM algorithm also, for $K$$K$ Gaussians what we're doing is we're fixing all the $K$$K$ mean vectors and covariance matrix(of shape $$) in first step, and updating all the K probabilities.
- And, in the second step, we're fixing probabilities and updating mean and standard deviations.

Now, let's just see how EM algorithm updates these parameters

GMM Algorithm & Implementation

There are two steps involved in GMM Algorithm:

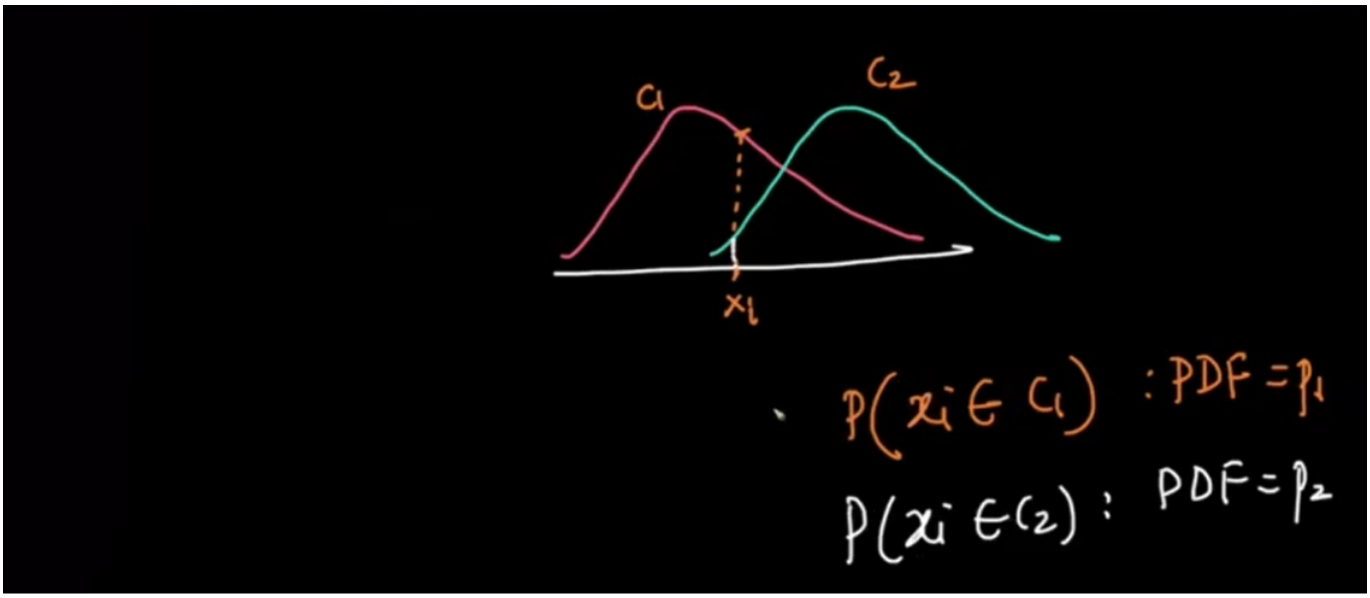
1. Expectation
2. Maximization

Lets see what both these steps do.

Given a datapoint $x_{i}$$x_{i}$ where $i = 1$$i = 1$ to $n$$n$, and $x_{i} \in D$$x_{i} \in D$, we want to find parameters $θ$$θ$ that best generated the data.

Step-1. Expectation:

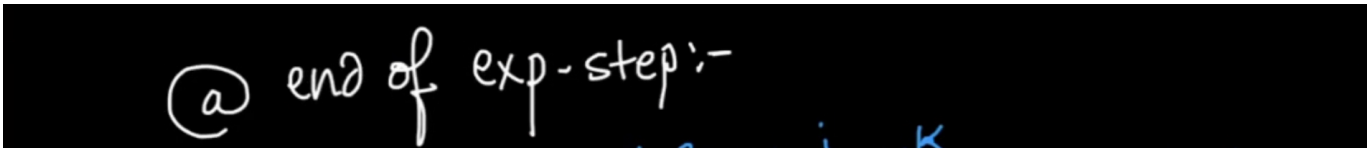
- This is basically an assignment step.
- In this step, for each point $x_{i}$$x_{i}$, we compute the probability of point $x_{i}$$x_{i}$ belonging to a $j^{th}$$j^{th}$ cluster
- So, initially we start by randomly assigning a probability such that it belings to j^{th} j^{th} gaussian ($j^{th}$$j^{th}$ cluster).
- We then compute the probabilities of a point belonging to k different clusters using Probability Density Function

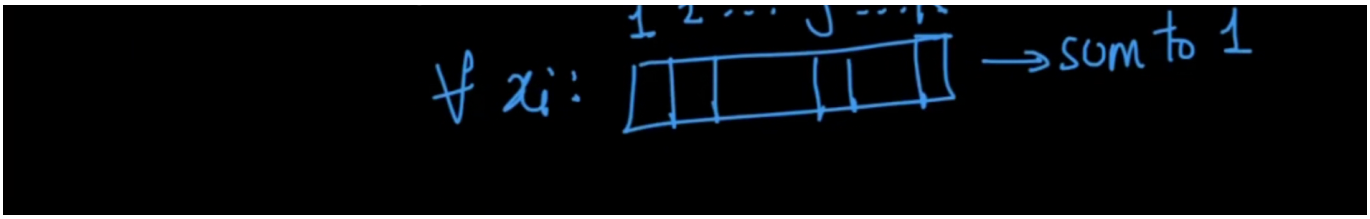


- There might be the case when probabilities of a point belonging to $k$$k$ clusters wont sum up to 1.

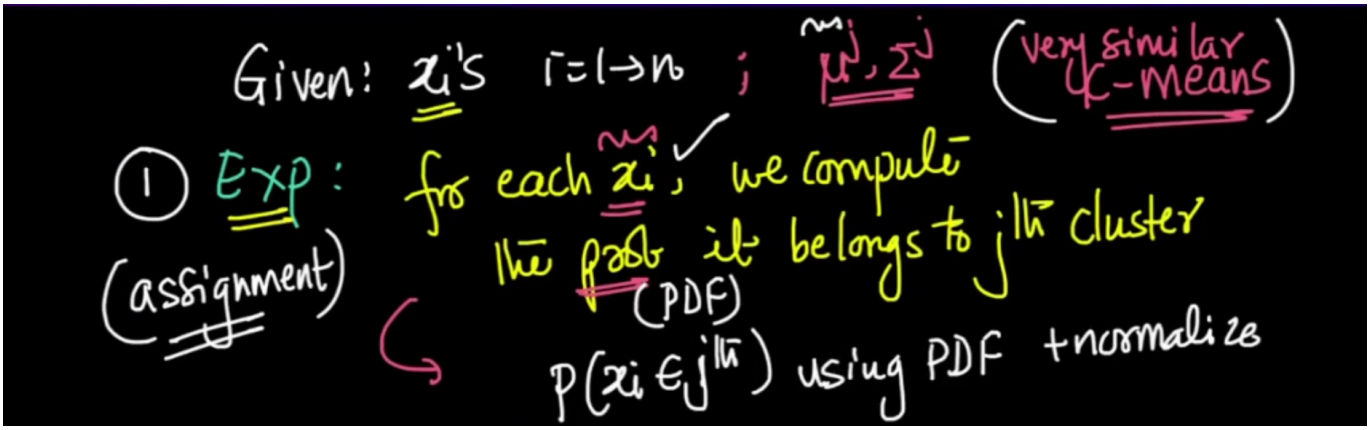
> **Q. How we will solve that?**
>
> - We normalize the probabilties. If there are two clusters, then; $P_{1} = \frac{P_{1}}{P_{1} + P_{2}}$$P_{1} = \frac{P_{1}}{P_{1} + P_{2}}$, and $P_{2} = \frac{P_{2}}{P_{1} + P_{2}}$$P_{2} = \frac{P_{2}}{P_{1} + P_{2}}$

- So, at the end of this step, for each point, we get a vector of normalized probabilities of that point belonging to each cluster, and they all sum up to 1





- So, if you notice, on a very high level, GMM works in a similar to K-Means.
  - The first step of the GMM (expectation) assigns a point to the cluster based on the probability.
  - This is what we do in the K-Means too. We initialize random cluster centroids at the start.
- Here in GMMs, we can relate these cluster centroids with the means of the mean vector(μ)



Step-2. Maximization:

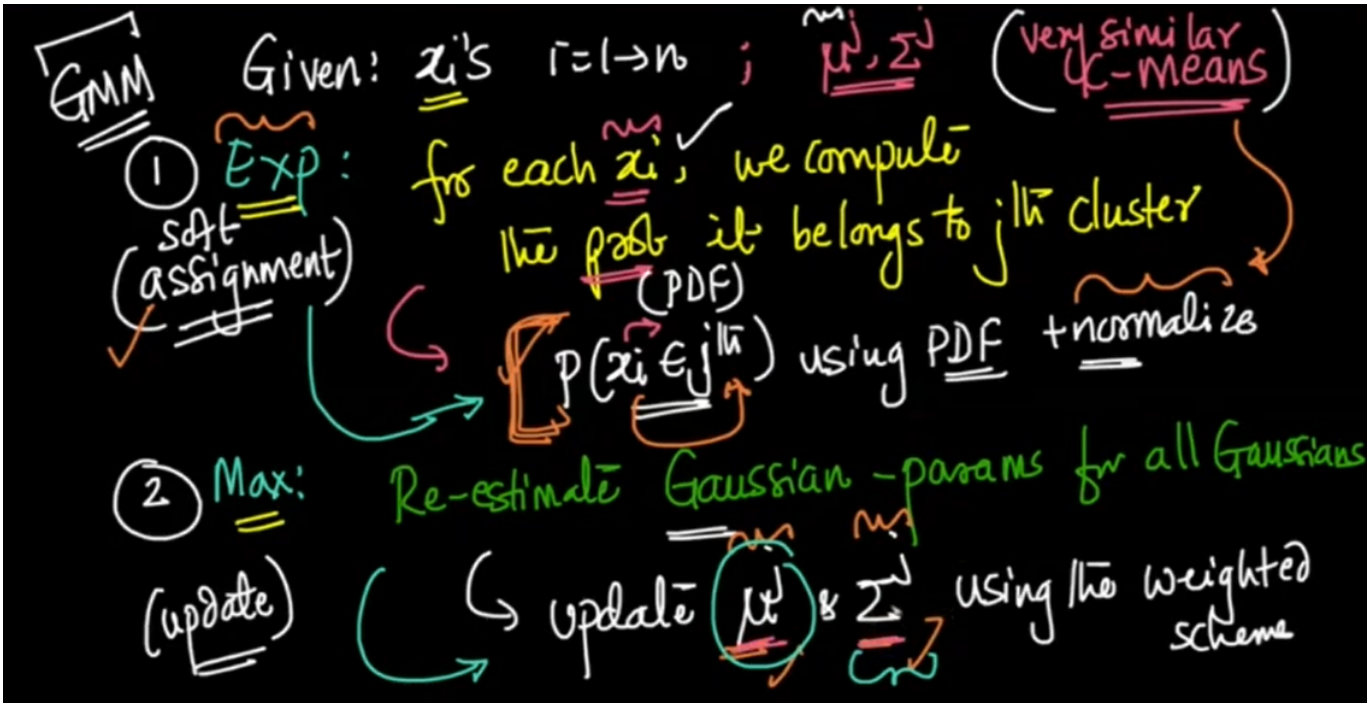
- In GMMs, we compute normalized probabilities of each and every point belonging to a cluster.
- This is kind of like a soft assignment, whereas in KMeans we did the Hard Assignment by randomly picking centroids.
- In the Maximization step, we re-estimate gaussian parameters, for all $K$$K$ gaussians.

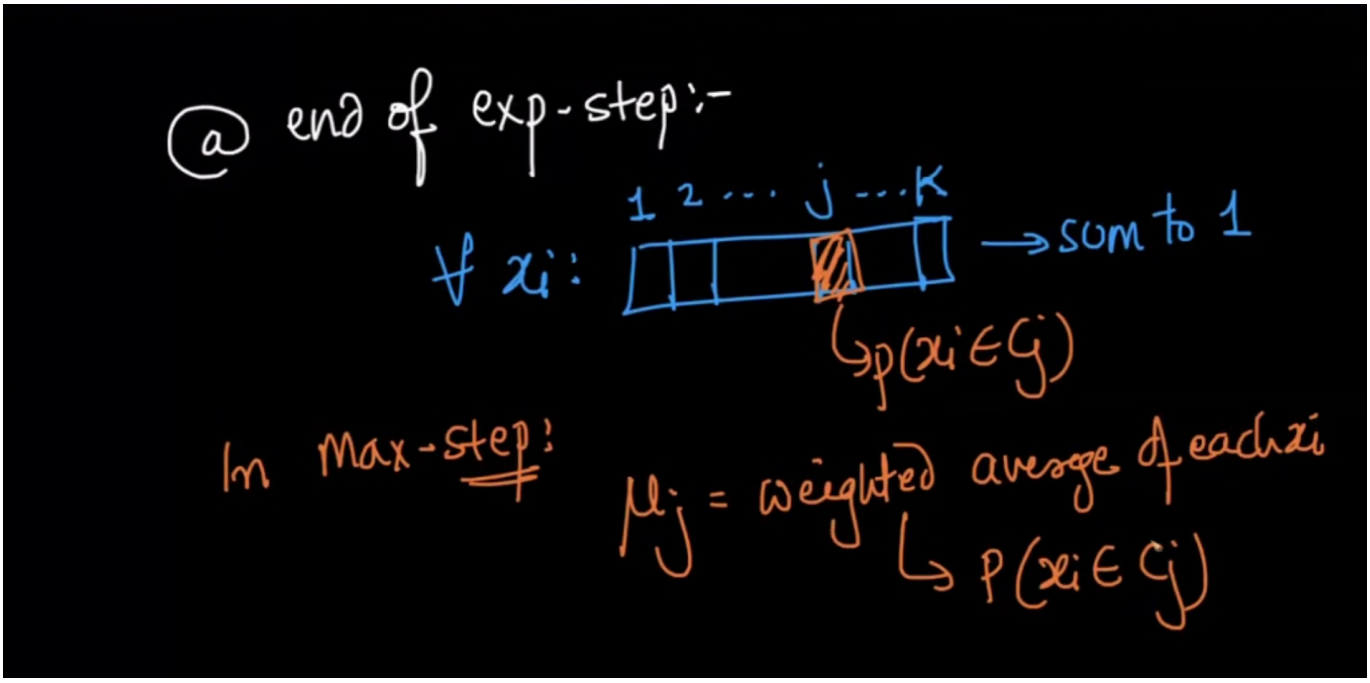
> **Q. How do we do that?**
>
> - We update mean vectore ($\mu^j$$\mu^j$) and covariance vectors ($ \sum^j$$\sum^j$) using weighted scheme

> Q. **What is weighted scheme?**
>
> - Suppose we've to update mean vector($\mu^j$$\mu^j$)
> - Instead of taking simple average as we do in K-Means, we compute weighted average of each $x_{i}$$x_{i}$, where the weightage is based on

> the probabilities that the point $x_{i} \in$$x_{i} \in$ cluster $j$$j$

- We also compute covariance vector in the same fashion
- In the seond step of K-means, we update the points based on the updated centroids of the cluster
- Whereas in GMM, we update a point based on the mean (which is same as centroid in K-Means) and covariance. The only difference is that we use weighting scheme in GMM Models





Putting it all Together:

- Expectation step computes the probabilities of a point belonging to certain cluster $j$$j$. These probabilities would be normalized.
- What it does is it assigns each and every point to a cluster with maximum probability but this is a soft assignment because the point still has a probability(low) of belonging to another cluster..
- In K-Means, we make clusters based on the distance we just assign the point to the closest centroid which is hard assignment.
- Now, in the second step, recall what we did in KMeans. We update the centroid(mean of cluster) and reassign the points to the closest cluster. This is again hard assignment.
- But, In maximization step, we update our parameters (mean vector and covariance matrix) based on the weighted average scheme.
- We do this because of the reason that each and every point do not belong to the cluster $j$$j$. It partially belongs to cluster because of the weighted probability.
- This is somewhat like Fuzzy Clustering.

Quiz:

```
K–Means require the number of clusters to be specified in the beginning, whereas
need this prior information. Is this statement True or False?

a. True
b. False
```

Answer

b. False

GMM Implementation

Use case: Customer Grouping with DBSCAN

In today's lecture, we'll try to segment customers based on 'Wholesale Customers Data'.

Q. What is meant by customer segmentation?

- Customer segmentation is the process by which you divide your customers up based on common characteristics – such as demographics or behaviours, so you can market to those customers more effectively.
- Almost all the companies in today's world, one way or other, make use of customer segmentation for marketing.

Q. But, how would we segment the customers based on some numbers present in the data?

- So far, we have seen K-Means which is a centroid based algorithm.
- We saw Agglomerative Clustering which is hierarchical system.
- Today we will see an overview (very high level) for 2 more clustering algorithms. The goal for this lecture is not to go into detail about these algorithms but to just get intuitive understanding of how they work, since they are not as popular in the community.
- We'll use these algorithms to do the segmentation of the customers on 'Wholesale Customers Data'.
- First, we will start with DBSCAN which is more of density based; i.e. the clustering should be done on the basis of how dense a neighborhood around a point.

Before jumping in, let's first understand and import the data, and try to visualize it on a plot.

Dataset - Wholesale Customers Data

The dataset used for this case study contains the following parameters:

1. **Fresh:** annual spending (some monetary unit) on fresh products (Continuous);
2. **Milk:** annual spending (some monetary unit) on milk products (Continuous);
3. **Grocery:** annual spending (some monetary unit) on grocery products (Continuous);
4. **Frozen:** annual spending (some m.u.) on frozen products (Continuous)
5. **Detergentss_Paper:** annual spending (some m.u.) on detergents and paper products (Continuous)
6. **Delicassen:** annual spending (some m.u.) on and delicatessen products (Continuous);
7. **Channel:** customers Channel - (Two possible values: Horeca (Hotel/Restaurant/Cafe) or Retail channel) (Nominal)
8. **Region:** customers Region (Three possible values: Lisnon, Oporto or Other) (Nominal)

```
1 !gdown 10ZdMFvfhGeXAa8xR0J00kHIHp9y9m9uL
```

```
Downloading...
From: https://drive.google.com/uc?id=10ZdMFvfhGeXAa8xR0J00kHIHp9y9m9uL
To: /content/wholesaledata.csv
```

```
  100% 15.0k/15.0k [00:00<00:00, 30.8MB/s]
```

```
1 from sklearn.cluster import DBSCAN
2 from sklearn.preprocessing import StandardScaler
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
6 plt.rcParams["figure.figsize"] = (16,12)
```

```
1 df = pd.read_csv('./wholesaledata.csv')
2 df.info()
```

```
  <class 'pandas.core.frame.DataFrame'>
  RangeIndex: 440 entries, 0 to 439
  Data columns (total 8 columns):
   #   Column            Non-Null Count  Dtype
  ---  ------            --------------  -----
   0   Channel           440 non-null    int64
   1   Region            440 non-null    int64
   2   Fresh             440 non-null    int64
   3   Milk              440 non-null    int64
   4   Grocery           440 non-null    int64
   5   Frozen            440 non-null    int64
   6   Detergents_Paper  440 non-null    int64
   7   Delicassen        440 non-null    int64
  dtypes: int64(8)
  memory usage: 27.6 KB
```

```
1 df.head()
```

- The dataset consists of 440 customers and has 8 attributes for each of these customers.
- No missing values
- Only two variables are **non-continuous (categorical)** in nature: Channel and Region.
- So for easing our computations, we will drop these two categorical features.

```
# Dropping categorical variables for simplicity
df.drop(["Channel", "Region"], axis = 1, inplace = True)
```

Visualizing the data in 2D

Using 2 features:

1. **Grocery:** The customer's annual spending (in some monetary unit) on grocery products.
2. **Milk:** The customer's annual spending (in some monetary unit) on milk products.

Let's plot two features data now:

```
x = df['Grocery']
y = df['Milk']

plt.scatter(x,y)
plt.xlabel("Groceries")
plt.ylabel("Milk")
plt.show()
```

Let us now implement GMM with help of sklearn's GaussianMixture() class.

We'll also compare GMM with K-Means, and you'll get to see how GMM performs relatively similar to the K-Means

```
!gdown 1TyC3sl4OM-NhmUTcCt4NN8nDZOWHOBrh
```

```
Downloading...
From: https://drive.google.com/uc?id=1TyC3sl4OM-NhmUTcCt4NN8nDZOWHOBrh
To: /content/std_df_gmm.pkl
100% 7.19k/7.19k [00:00<00:00, 25.3MB/s]
```

```
Start coding or generate with AI.
```

```
!gdown 1Z6nMciyzLvw9fdt8ajdNDebkRTbj5FW-
```

```
Downloading...
From: https://drive.google.com/uc?id=1Z6nMciyzLvw9fdt8ajdNDebkRTbj5FW-
To: /content/df_gmm.pkl
100% 7.71k/7.71k [00:00<00:00, 16.6MB/s]
```

```
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (16,12)
```

```
import pickle
```

```
with open('/content/std_df_gmm.pkl','rb') as f:
    std_df = pickle.load(f)


with open('/content/df_gmm.pkl','rb') as f:
    df = pickle.load(f)
```

```
std_df
```

```
array([[-4.11148934e-02,  5.23567773e-01],
       [ 1.70318354e-01,  5.44457667e-01],
```

```
[-2.81571010e-02,  4.08537706e-01],
[-3.92976899e-01, -6.24019925e-01],
[-7.93561833e-02, -5.23964546e-02],
[-2.97637045e-01,  3.34066589e-01],
[-1.02848766e-01, -3.52315651e-01],
[ 1.55358951e-01, -1.13980948e-01],
[-1.85336177e-01, -2.91409401e-01],
[ 1.15142340e+00,  7.18494904e-01],
[ 5.29133322e-01, -5.33459952e-02],
[-3.61161832e-01, -6.33786629e-01],
[ 4.00924920e-01,  8.84800166e-01],
[ 7.40671917e-01,  5.58511794e-02],
[ 4.36111121e-01,  4.97658878e-01],
[-4.35116062e-01, -6.35143116e-01],
[ 4.39271558e-01,  4.09622895e-01],
[-5.28665002e-01,  4.89330975e-02],
[ 2.26258092e-01,  7.19933704e-02],
[ 1.59362172e-01, -4.47812310e-01],
[-3.52839347e-01, -1.73259414e-01],
[-6.25901120e-01, -6.68105741e-01],
[-3.66850619e-01, -5.26217238e-01],
[ 1.48200513e+00,  4.15447557e+00],
[ 6.15307909e-01,  5.39845613e-01],
[-3.75330646e-02, -2.12461878e-01],
[-5.36250052e-01, -6.55897361e-01],
[-5.16866037e-01, -6.77329850e-01],
[ 1.89686519e+00,  1.99237151e+00],
[-5.62797724e-01, -5.01393533e-01],
[ 3.32448781e-01, -2.96564050e-01],
[-5.07595421e-01, -1.97676174e-01],
[-5.33616354e-01, -6.07470788e-01],
[-6.58716513e-02, -1.37041221e-01],
[-5.99353447e-01, -5.17807021e-01],
[ 3.30763215e-01, -4.14089128e-02],
[-2.65821977e-01, -1.94556255e-01],
[ 4.76775412e-01,  6.45651571e-01],
[ 9.22607751e-01,  1.34736211e+00],
[-7.42626599e-01, -7.10970719e-01],
[-3.36510422e-01, -1.98625714e-01],
[-2.10198283e-01, -3.70492572e-01],
[ 7.38459611e-01,  2.38569930e-01],
[ 1.69048864e+00,  7.18766201e-01],
[ 2.65447513e-01,  1.66947435e-01],
[ 1.43059535e+00,  2.20398342e+00],
[ 1.47526287e+00,  1.12218533e+00],
[ 5.01663807e+00,  6.57390514e+00],
[ 3.07270632e-01,  4.82548542e-02],
[ 2.20911638e+00,  2.11825347e+00],
[-6.29061557e-01, -6.37720440e-01],
[-1.00636460e-01, -2.36742989e-01],
[-2.18626115e-01, -2.55055559e-01],
[ 3.77221641e-01,  6.34392732e-01],
[-6.32538038e-01, -5.89700813e-01],
[-3.10384141e-01, -2.86661697e-01],
[ 1.99262643e+00,  3.26855414e+00],
```

```
       [ 2.67133079e-01,  5.61142453e-01],
```

```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=2).fit(std_df)
labels = gmm.predict(std_df)
plt.scatter(std_df[:, 0], std_df[:, 1], c=labels, s=40, cmap='viridis');
```

Comparing with K-Means

```
1 from sklearn.cluster import KMeans
2
3 k = 2 ## arbitrary value
4 kmeans = KMeans(n_clusters=k)
5 y_pred = kmeans.fit_predict(std_df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: Future
  warnings.warn(
```

```
1 clusters = df.copy()
2 clusters['label'] = kmeans.labels_
```

```
1 plt.scatter(clusters['Grocery'], clusters['Milk'], c=clusters['label'])
```

Takeaways from GMMs:

- One thing to note is that, while GMM may sound tempting, it performs quite similar to K-Means, and people end up using K-Means instead.
- It is less used because Hard GMM is same as K-Means.
- They also make a strong assumption that the data is generated with Mixture of Gaussian Distributions, which may not be True always.