Lecture 11: Unsupervised Leanring

ADDO AI
2018

Instructor: Dr. Syed Waqar ul Qounain Jaffry
Assistant Professor PUCIT

Unsupervised Learning

Agenda

- Introduction
- Prerequisite Self Check
- Un Supervised Learning
- Principal Component Analysis (PCA)
- K-Mean Clustering
- Gaussian Mixture Models
- Expectation Maximization Algorithm

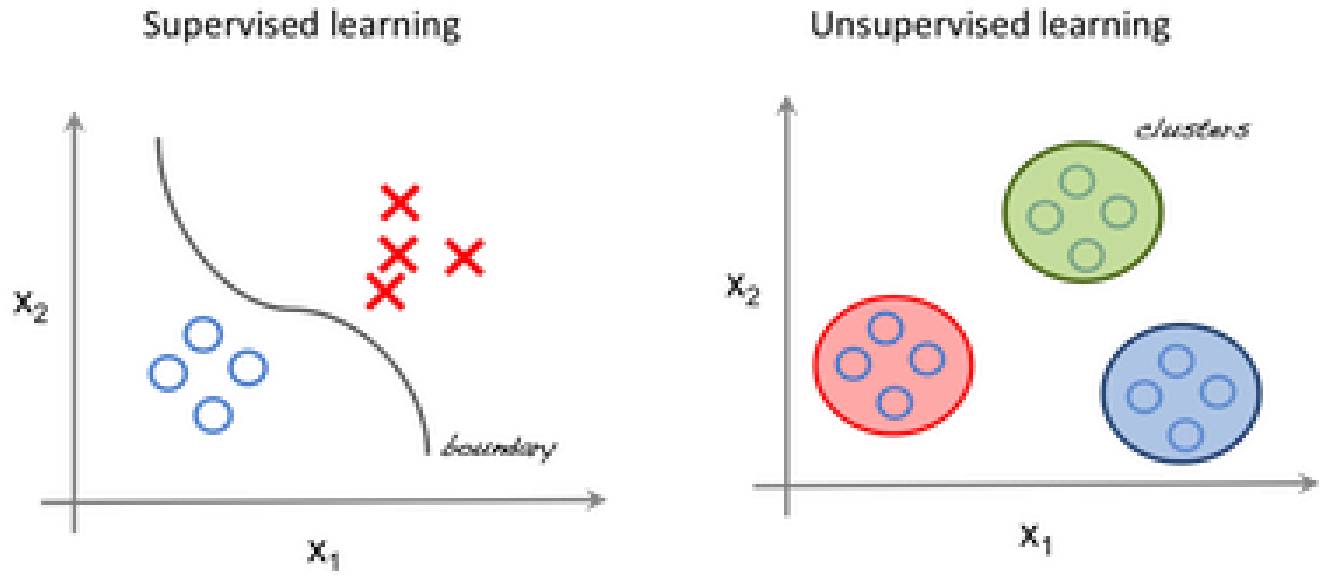
Introduction

- Supervised Learning
  - This type of learning requires ground truth
  - Using the ground truth, machine learning models train to differentiate
  - Already have prior knowledge of what the output values should be
  - E.g. Classification and Regression
- If ground truth is not available
  - Data can still carry patterns
  - Then how to process data?
- Solution
  - Unsupervised Learning

Pre-requisite Self Check

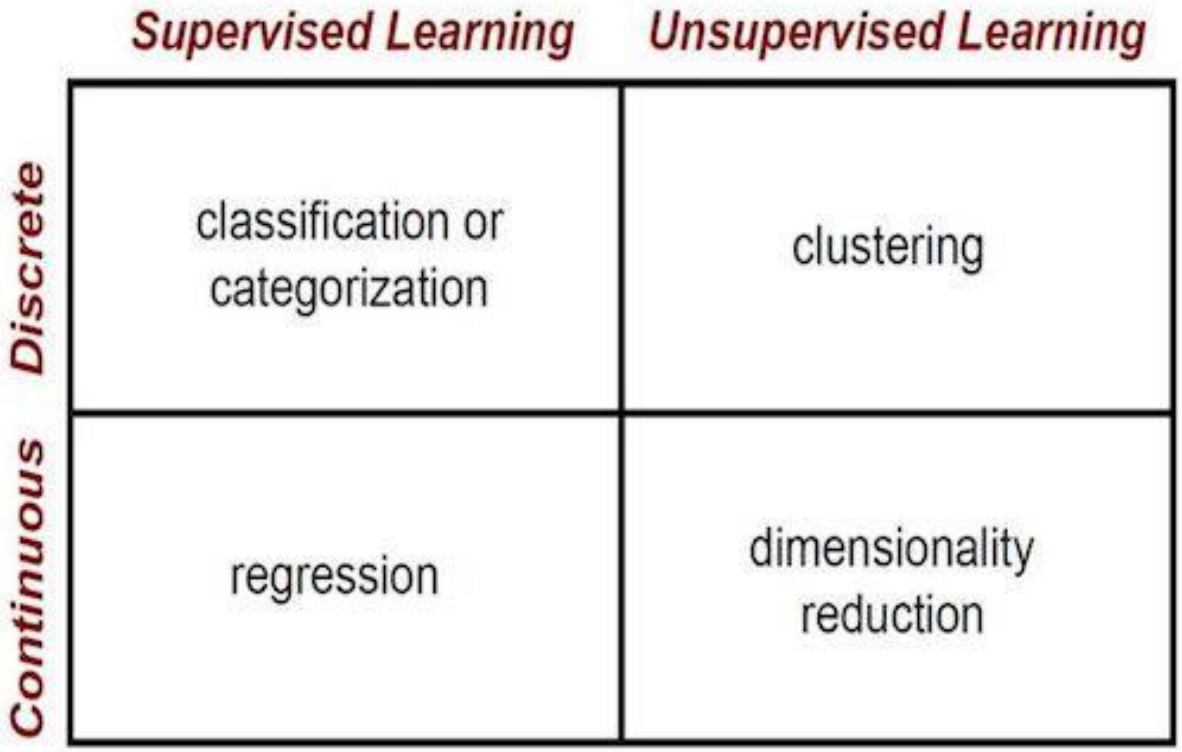
- Variance-Covariance-Matrix
- Langrange Multipliers
- Matrix Calculus $\quad d\, x^T A\, x / d\, x \quad =$
- Eigen Values of the Matrix
- Eigen Vectors of the Matrix
- Joint probability Multiplication rule: P(AB)

Supervised vs Unsupervised Learning



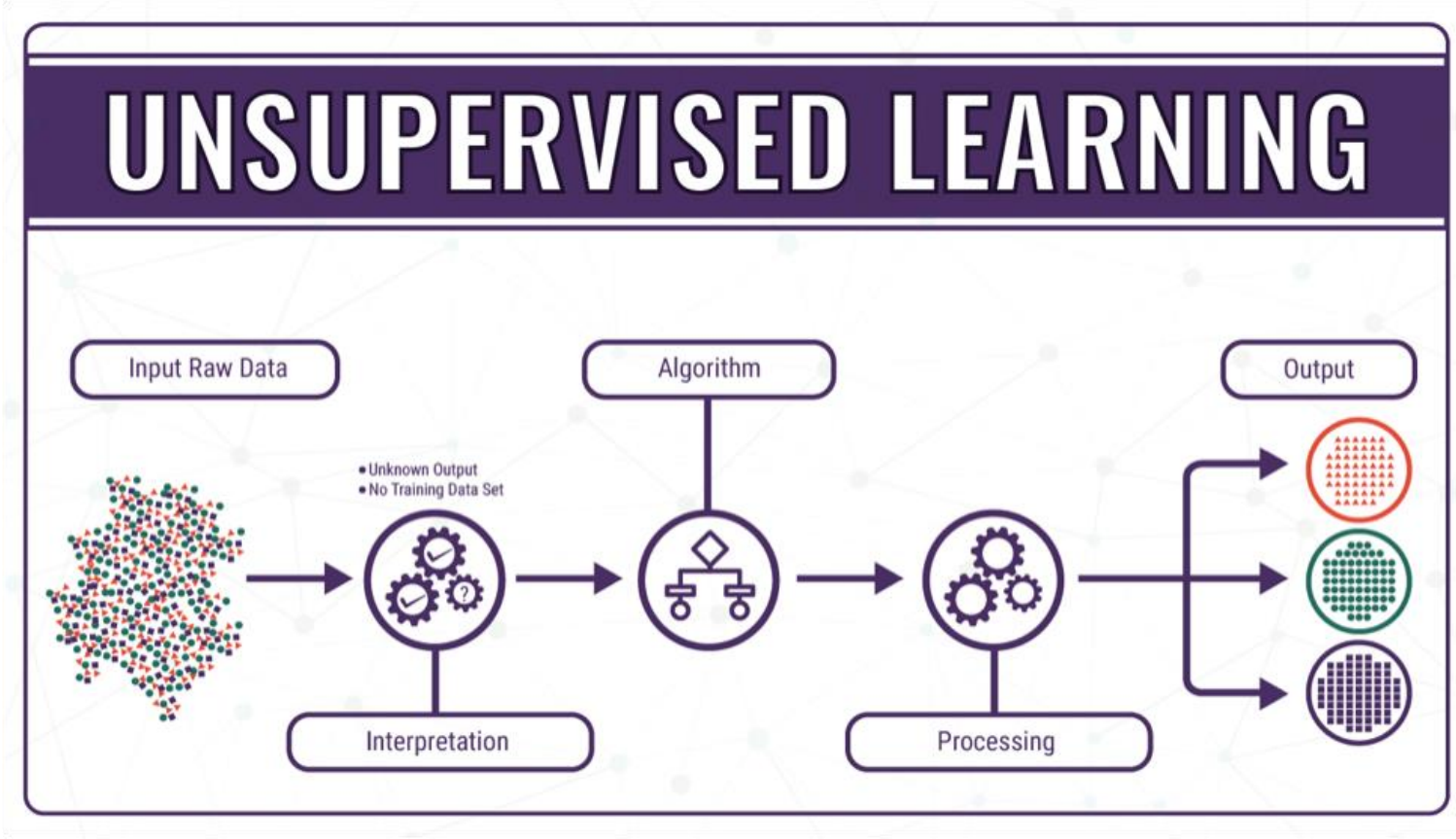
Unsupervised Learning

- It is focused on finding patterns in data
- Infer the natural structure present within data
- It is used for data exploration



Unsupervised Learning

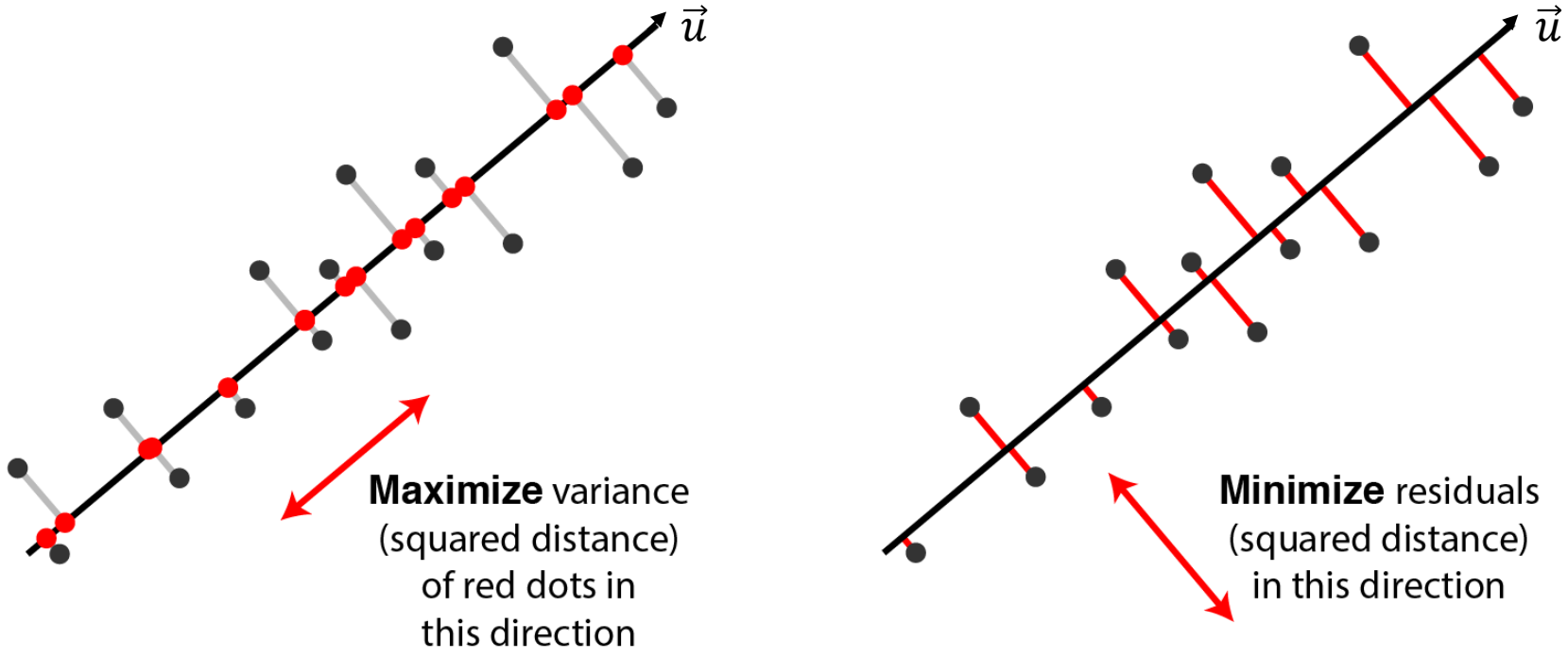
- Major applications
  - Dimensionality Reduction
    - PCA
  - Clustering
    - K-means
  - Density Estimation
    - Gaussian Mixture Models



Principal Component Analysis (PCA)

- Principal Component Analysis (PCA) is the primary example of linear unsupervised learning
- PCA is used for:
  - Dimensionality reduction
  - Feature extraction
  - Lossy data compression
- It is also called Karhunen-Loeve transform

Principal Component Analysis (PCA)



Maximum Variance Formulation

- Consider a set of data points
  - $X = [x_1, x_2, x_3, \dots, x_n]$ where each $x_i \in R_D$.
- Goal
  - To find a vector $\boldsymbol{u} \in R_D$ such that the variance of the projected data onto $\boldsymbol{u}$ is maximum.
  - Projections of a data points x_i onto $\boldsymbol{u}$ are obtained via dot-products $\boldsymbol{u^T x_i}$ for $i = 1,2, \dots, N$
  - Mean of projected data is computed as $\boldsymbol{u^T \overline{x}}$ where $\boldsymbol{\overline{x} = \sum_{i=1}^{n} x_i}$
  - Hence, variance of projected data along the direction $\boldsymbol{u}$ is computed as
    - $Var(u) = \frac{1}{N} \sum_{i=1}^{n} (u^T x_i - u^T \bar{x})^2$

Maximum Variance Formulation

- $Var(u) = \frac{1}{N}\sum_{i=1}^{n}(u^T x_i - u^T \bar{x})\,(u^T x_i - u^T \bar{x})^T$
- $Var(u) = \frac{1}{N}\sum_{i=1}^{n}(u^T x_i - u^T \bar{x})\left(x_i^T u - \bar{x}^T u\right)$
- $Var(u) = \frac{1}{N}\sum_{i=1}^{n} u^T(x_i - \bar{x})\left(x_i^T - \bar{x}^T\right)u$
- $Var(u) = u^T \frac{1}{N}\sum_{i=1}^{n}(x_i - \bar{x})\left(x_i^T - \bar{x}^T\right)u$
- Say $S = \frac{1}{N}\sum_{i=1}^{n}(x_i - \bar{x})\left(x_i^T - \bar{x}^T\right)$, that is a data-covariance matrix
- $Var(u) = u^T S u$

Maximum Variance Formulation

- Goal
  - To find a direction vector u that maximizes $u^T Su$
  - As we are interested in direction vector hence, constraint is applied
    - $u^T u = 1$ i.e. u is a unit vector
- To solve the optimization problem with constraint
  - Lagrang multipliers are used
  - Lagrang multiplier states that for a function f(x) with given constraint g(x)
  - $L(x, \lambda) = f(x) - \lambda g(x)$ where $\lambda \neq 0$ is called lagrang multiplier
  - In order to find maximum value, this lagrangian function is derivated and equated to zero

Principal Component Analysis

- Currently
  - $f(\boldsymbol{u}) = \boldsymbol{u}^T S\boldsymbol{u}$
- Constraint
  - $\boldsymbol{u^T u} = 1 \Rightarrow g(u) = \boldsymbol{u^T u} - 1$
- Hence, Lagragian function becomes
  - $L(\boldsymbol{u}, \lambda) = f(\boldsymbol{u}) - \lambda g(\boldsymbol{u}) \Rightarrow \boldsymbol{u^T} S\boldsymbol{u} - \lambda\left(\boldsymbol{u^T u} - 1\right)$
- To find optimal values, take gradient w.r.t **u**:
  - $\frac{\partial L(\boldsymbol{u},\lambda)}{\partial \boldsymbol{u}} = \frac{\partial}{\partial \boldsymbol{u}}\left(\boldsymbol{u^T} S\boldsymbol{u} - \lambda\left(\boldsymbol{u^T u} - 1\right)\right) = 2S\boldsymbol{u} - 2\lambda\boldsymbol{u}$

Principal Component Analysis

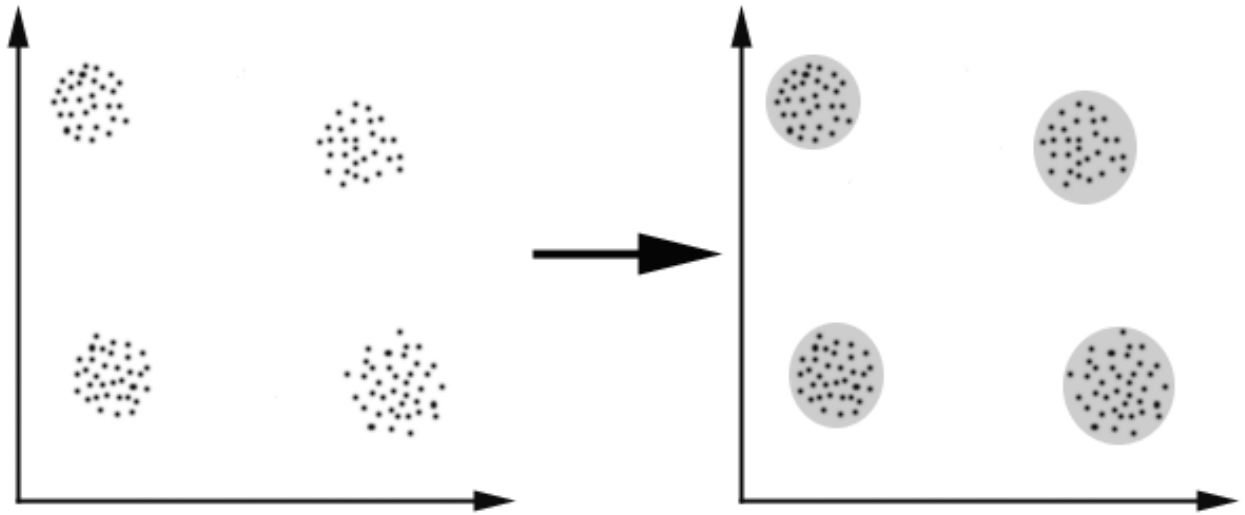
- Equation gradient to zero:

$$\Rightarrow\ 2S\boldsymbol{u}\ -\boldsymbol{2}\lambda\boldsymbol{u}=0\Rightarrow\ S\boldsymbol{u}=\lambda\boldsymbol{u}$$

- This should be recognizable as an eigenvector equation where $\boldsymbol{u}$ is an eigenvector of S and λ is the associated eigenvalue.
- The eigenvector of S corresponding to the largest eigenvalue is called the first principal component.
- Additional principal components can be defined incrementally by choosing each new projection direction as the one with maximum projected variance among all directions orthogonal to those already considered .
- First M principal components correspond to the eigenvectors $\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots, \boldsymbol{u_M}$ of S corresponding to the M largest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$.

Clustering

- Unsupervised learning algorithm to identify groups or clusters of similar data points
- Come up with clusters such that:
  - Within cluster, points have minimum distance
    - Small intra-cluster distances
  - Between clusters, distance is large
    - Large inter-cluster distances



K-means clustering

- Can be seen as an instance of the more powerful framework of Expectation Maximization (to be covered later)
- Given data points $x_1, x_2, x_3, \ldots, x_n$ where each $x_n \in R_D$, and an integer K > 1, the goal is to partition the data into K clusters .
- It performs mutually exclusive clustering
  - No data point can exist in multiple clusters
- Given K, randomly select K data points as clusters with each cluster mean denoted by μ_k
- At every step, calculate distance between remaining points with that of selected cluster.
  - For a data point x_i and any mean cluster point μ_k
    - the distance becomes $\|x_n - \mu_k\|^2$

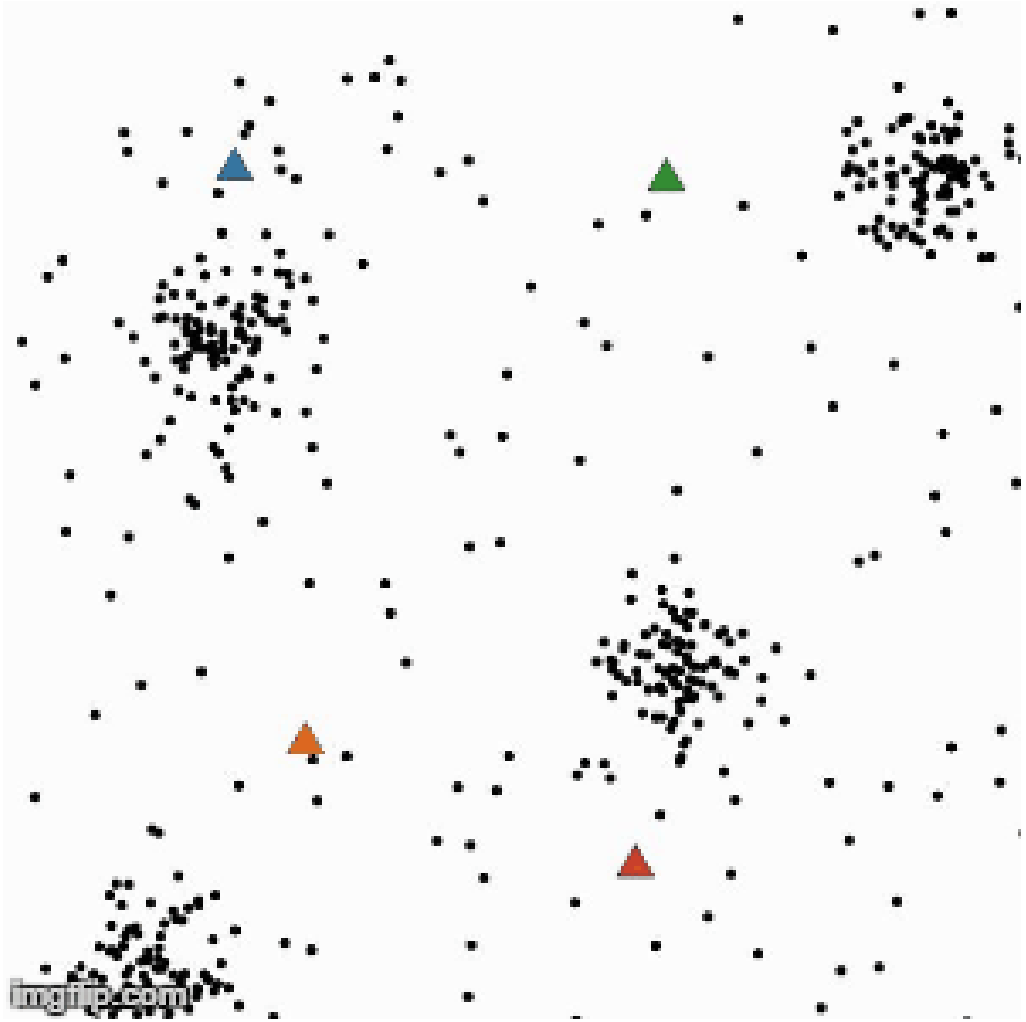
K-means clustering

- We also need a variable to denote assignment of any point x_n to the proper cluster.
- Let r_{nk} denote the assignment of data point x_n
- Then, as clusters to be formed are mutually exclusive:
  - Hence, a given data-point, can belong to only one cluster
  - This can be defined using hot-vector encoding or 1-of- K
  - Here x_{nk} = 1 if x_n belongs to cluster k and 0 otherwise
    - E.g. if a data point x_n belongs to cluster 2, whereas K=3:
      - respective vector would be [0 1 0]
      - Here $r_{n1} = r_{n3}$=0; as point x_n neither belongs to cluster 1 or 3, whereas r_{n2}= 1

K-means clustering

- Then for a particular set of clusters $\{\mu_1, \ldots, \mu_k\}$ and cluster assignments $\{\boldsymbol{r_1}, \ldots, \boldsymbol{r_N}\}$, where $\boldsymbol{r_k}$ is a 1-of-K vector
- We can compute the sum-of-squared distances between data points and their assigned clusters as
  - $Distance(\mu_k, r_{ik}) = \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} \|x_n - \mu_k\|^2$
- Goal:
  - $argminDistance(\mu_k, r_{nk})$
- Achieved via iterative, alternating optimization between assignments $\{r_{nk}\}$ and clusters $\{\mu_k\}$.

K-means clustering



K-means clustering

- Why alternating optimization?
- Finding cluster centers and cluster memberships simultaneously is a chicken-and-egg problem.
- However, individually these problems are much simpler.
  - Given memberships, computing cluster centers is trivial.
  - Given cluster centers, computing memberships is trivial.
- Alternating optimization gives us a powerful framework of solving complex problems by decomposing them into simpler ones .
- Notice that we appended the observed data x_i with some unobserved variables r_{ik} and then solved easy individual problems.
- These unobserved variables are called hidden or latent variables.

K-means clustering

- Data points $\{x_1, \ldots, x_N\}$, integer K > 1
- Result : Cluster representatives $\{\mu_k\}$, assignments $\{r_{nk}\}$

Choose some initial μ_k;

while not converged do

Fix clusters and update assignments ($\{r_{nk}\}$ = arg min $\{r_{nk}\}$ $\mathrm{Distance}(\mu_k, r_{nk})$);

Fix assignments and update clusters ($\{\mu_k\}$ = arg min $\{\mu_k\}$ $\mathrm{Distance}(\mu_k, r_{nk})$);

end

K-means clustering

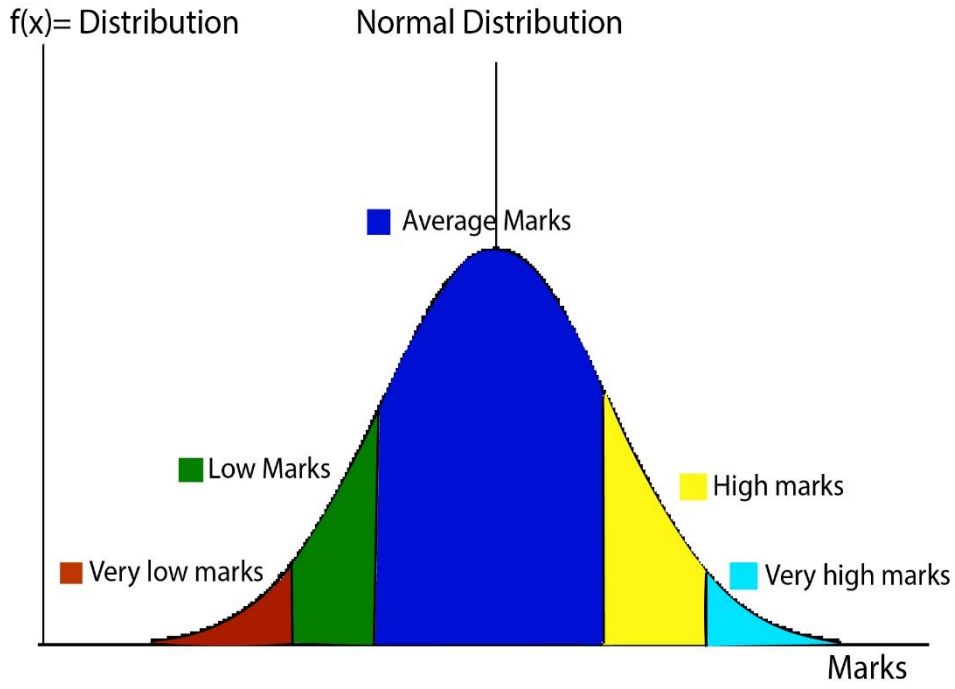
- $\{r_{nk}\}$ = arg min $\{r_{nk}\}$ Distance(μ_k, r_{nk})
  - To obtain this, we have to ensure that $\|x_n - \mu_k\|^2$ is minimum
  - In other words, point x_n is placed in cluster that is closest.
  - $r_{nk} = \begin{cases} 1 & if\ k = argmin_j \|x_n - \mu_j\|^2 \\ 0 & otherwise \end{cases}$
- $\{\mu_k\}$ = arg min $\{\mu_k\}$ Distance(μ_k, r_{nk})
  - As distance function is quadratic in terms of μ_k
  - $\frac{\partial}{\partial \mu_k} Distance(\mu_k, r_{nk}) = \frac{\partial}{\partial \mu_k} \sum_{k=1}^{K} \sum_{i=1}^{N} r_{nk} \|x_n - \mu_k\|^2 = 0$
  - $\Rightarrow -2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \Rightarrow \boldsymbol{\mu_k} = \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x_n}}{\sum_{n=1}^{N} r_{nk}}$

K-means clustering

- Requires value of K
- To determine value of K:
  - For various values of K , determine which value results in least error
- Hard decision
  - When points lie at cluster boundaries, hard decision is not also a good strategy
  - Use probability based soft assignments
    - Lead to mixture models

Gaussian Mixture Models

- Simple Gaussian Model
  - Two parameters
    - Mean
    - Variance
  - $N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-1}{2\sigma^2}(x-\mu)^2}$
- For a D-dimensional input vector **x**:
  - $N(\boldsymbol{x}|\boldsymbol{\mu},\sum) = \frac{1}{(2\pi)^{D/2}} \frac{1}{\sqrt{|\sum|}} e^{\frac{-1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \sum^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$
  - Where $\sum$ is called the covariance matrix that has D*D dimensions
  - $|\sum|$ is determinant and $\boldsymbol{\mu}$ is the D-dimensional mean vector

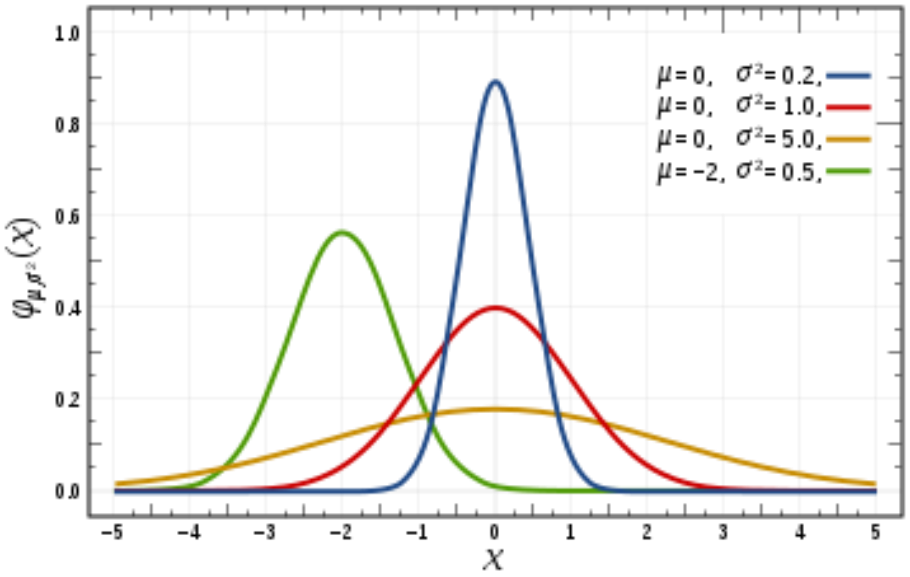


Gaussian Mixture Models (GMM)

- Multi model data
  - Can't be modeled using Simple Normal distribution/Uni-Modal Gaussian
  - Use mixture of models, that is linear superpositions of Uni-modal Gaussians

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}|\boldsymbol{\mu_k}, \sum_k)$$

  - Here, π_k refers to mixing coefficients and they must satisfy
    - $0 < \pi_k < 1$
    - $\sum_{k=1}^{K} \pi_k = 1$



Gaussian Mixture Models (GMM) Latent Variables

- Now, lets derive the p(x) using latent variables
- Like K-means, lets assume a latent variable **z** with 1-of-K encoding
- $p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}, z_k) = \sum_{k=1}^{K} p(z_k) p(\boldsymbol{x}|z_k)$
- As, now, soft-assignments are being made in form of probabilities:
  - $p(z_k = 1) = \pi_k$
- Due to 1-of-K encoding of latent variable i.e. vector **z:**
  - $p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$ # equivalent to $p(z_k = 1)$
- Similarly, the conditional distribution of x given particular value of **z** is:
  - $p(\boldsymbol{x}|z_k = 1) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \sum_k)$

Gaussian Mixture Models (GMM) Latent Variables

- Using whole vector **z** notation, it can be written as:
  - $p(\boldsymbol{x}|\boldsymbol{z}) = \prod_{k=1}^{K} N(\boldsymbol{x}|\boldsymbol{\mu_k}, \sum_k)^{z_k}$ # equivalent to $p(\boldsymbol{x}|z_k = 1)$
- Joint distribution then becomes:
- $p(\boldsymbol{x}) = \sum_{k=1}^{K} p(z_k)p(\boldsymbol{x}|z_k) = \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}|\boldsymbol{\mu_k}, \sum_k)$
- Here, using a latent variable, we have derived the similar expression for GMM

Gaussian Mixture Models (GMM) Responsibilities

- $p(\boldsymbol{x})$ is the marginal density that we are looking to model.
- $p(\boldsymbol{x}|z_k = 1)$ is the component conditional density . That is, probability density of x according to component k .
- $p(z_k = 1)$ is the prior probability of component k .
- $p(z_k = 1|\ \boldsymbol{x})$ is the posterior probability of component k.
  - It can be represented as responsibility r_k
  - Can be viewed as the responsibility that component k takes for explaining observation x .
  - Can be computed via Bayes' theorem
    - $r_k = p(z_k = 1|\ \boldsymbol{x}) = \frac{p(z_k=1)\ p(\boldsymbol{x}|\boldsymbol{z_k}=\mathbf{1})}{p(x)} = \frac{p(z_k=1)\ p(\boldsymbol{x}|\boldsymbol{z_k}=\mathbf{1})}{\sum_{j=1}^{K} p(z_J)p(\boldsymbol{x}|z_j)} = \frac{\pi_k N(\boldsymbol{x}|\boldsymbol{\mu_k}, \sum_k)}{\sum_{j=1}^{K} \pi_j N\left(\boldsymbol{x}\middle|\boldsymbol{\mu_j}, \sum_j\right)}$

Gaussian Mixture Models (GMM) Parameter Estimation

- Assuming independent and identically data distribution (IID)
  - It enables to express joint probability as product of marginals
  - p(x,y) = p(x) p(y) (with iid) p(x,y) = p(y|x) p(x) = p(x|y)p(y) (w/o iid)
- Likelihood of any model M with data is given by:
  - $L(D|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$

  $where\ \theta\ represents\ parameters\ of\ model\ and\ D\ represents\ input\ data\ [x_1, x_2, \dots x_n]$
- As likelihood and its log would present the same pattern, hence, for simplicity, log-likelihood is used
  - $Log\_L(D|\theta) = Log(\prod_{n=1}^{N} p(x_n|\theta)) = \sum_{n=1}^{N} \log(p(x_n|\theta))$

Gaussian Mixture Models (GMM) Parameter Estimation

- As $p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k N(\boldsymbol{x}|\boldsymbol{\mu_k}, \sum_k)$
- By taking likelihood of GMM:
  - $L(D|\theta) = \prod_{n=1}^{N}(\sum_{k=1}^{K} \pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k))$
- Then log-likelihood becomes:
  - $LogL(D|\theta) = \sum_{n=1}^{N} \log((\sum_{k=1}^{K} \pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k))$
- Derivate w.r.t mean $\boldsymbol{\mu_k}$
- $\frac{\partial LogL}{\partial \boldsymbol{\mu_k}} = \frac{\partial}{\partial \boldsymbol{\mu_k}}(\sum_{n=1}^{N} \log((\sum_{k=1}^{K} \pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k))) = \mathbf{0}$
- $\frac{\partial LogL}{\partial \boldsymbol{\mu_k}} = -2 \sum_{n=1}^{N} \left(\frac{\pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k)}{\sum_{j=1}^{K} \pi_j N\left(\boldsymbol{x_n}\middle|\boldsymbol{\mu_j}, \sum_j\right)} \right) \sum_k{}^{-1} (\boldsymbol{x_n} - \boldsymbol{\mu_k}) = \mathbf{0}$
- $\frac{\partial LogL}{\partial \boldsymbol{\mu_k}} = \sum_{n=1}^{N} r_{nk} \sum_k{}^{-1} (\boldsymbol{x_n} - \boldsymbol{\mu_k}) = \mathbf{0}$

Gaussian Mixture Models (GMM) Parameter Estimation

- $\Rightarrow \sum_{n=1}^{N} r_{nk} \sum_{k}^{-1} x_n = \sum_{n=1}^{N} r_{nk} \sum_{k}^{-1} \boldsymbol{\mu_k}$
- By multiplying both sides by$\sum_k$, equation yields :
  - $\boldsymbol{\mu_k} = \frac{\sum_{n=1}^{N} r_{nk} \boldsymbol{x_n}}{\sum_{n=1}^{N} r_{nk}} = \frac{\mathbf{1}}{\boldsymbol{N_k}} \sum_{n=1}^{N} r_{nk} \ \boldsymbol{x_n}$ $\quad where\ \boldsymbol{N_k} = \sum_{n=1}^{N} r_{nk}$
- This expression is the same as derived using K-means distance.
- Only difference is that in k-means, r_{nk} could either be 1 or zero, whereas here, it is probability based soft-assignments.

Gaussian Mixture Models (GMM) Parameter Estimation

- Similarly, log-likelihood function can be differentiated with respect to remaining parameters that include $\pi_k \; and \sum_k$
- $\frac{\partial LogL}{\partial \sum_k} = \frac{\partial}{\partial \sum_k} (\sum_{n=1}^{N} \log((\sum_{k=1}^{K} \pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k))) = 0$
- It yields:
  - $\sum_k = \frac{\mathbf{1}}{\boldsymbol{N_k}} \sum_{n=1}^{N} r_{nk} \; (\boldsymbol{x_n} - \boldsymbol{\mu_k})(\boldsymbol{x_n} - \boldsymbol{\mu_k})^{\boldsymbol{T}}$
- For π_k value, as it must follow a constraint
  - Lagrangian function will be used
  - f(x) = log-likelihood function g(x) : $\sum_{k=1}^{K} \pi_k = 1 \Rightarrow \sum_{k=1}^{K} \pi_k - 1 = 0$

Gaussian Mixture Models (GMM) Parameter Estimation

- Expression becomes:
  - $L(x,\lambda) = LogL(D|\boldsymbol{\pi}, \boldsymbol{\mu_k}, \sum_k) + \lambda(\sum_{k=1}^{K} \pi_k - 1) = 0$
  - $\frac{\partial}{\partial \pi_k} L(x,\lambda) = \frac{\partial}{\partial \pi_k} (\sum_{n=1}^{N} \log(\sum_{k=1}^{K} \pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k)) + \frac{\partial}{\partial \pi_k} [\lambda(\sum_{k=1}^{K} \pi_k - 1)] = 0$
  - $\Rightarrow \sum_{n=1}^{N} \left(\frac{N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x_n}|\boldsymbol{\mu_j}, \sum_j)} \right) + \frac{\partial}{\partial \pi_k} [\lambda(\pi_0 + \cdots . + \pi_k + \cdots + \pi_K)]$
  - $\Rightarrow \sum_{n=1}^{N} \left(\frac{r_{nk}}{\pi_k} \right) + \lambda = 0$ $\quad \#\mathrm{r_{nk}} = \frac{\pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x_n}|\boldsymbol{\mu_j}, \sum_j)} \Rightarrow \frac{N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x_n}|\boldsymbol{\mu_j}, \sum_j)} = \frac{r_{nk}}{\pi_k}$

Gaussian Mixture Models (GMM) Parameter Estimation

- Multiplyig both sides by π_{k}
  - $\sum_{n=1}^{N}(r_{nk}) + \lambda\pi_{\mathrm{k}}$=0…… (1)
  - $\Rightarrow \sum_{n=1}^{N}(r_{nk}) = -\lambda\pi_{\mathrm{k}}$
- Taking summation over k on both sides:
  - $\Rightarrow \sum_{k=1}^{K}\sum_{n=1}^{N}(r_{nk}) = -\lambda\sum_{k=1}^{K}\pi_{\mathrm{k}}$
  - $\Rightarrow \sum_{k=1}^{K}\sum_{n=1}^{N}(r_{nk}) = -\lambda$
  - $\Rightarrow$ **$\lambda = -N$**
- By putting value of λ in equation 1:
  - $\sum_{n=1}^{N}(r_{nk}) - N\pi_{\mathrm{k}}$=0…… (1)
  - $N_k = N\pi_{\mathrm{k}} \Rightarrow \pi_k = \frac{N_k}{N}$
- In words, mixing coefficient for component k is given by the average responsibility that it takes for explaining the training data points .

Gaussian Mixture Models (GMM) Parameter Estimation

- Notice that solutions for $\pi_k, \boldsymbol{\mu_k}, \sum_k$ are dependent on the responsibilities r_{nk}.
- However, the responsibilities depend on $\pi_k, \boldsymbol{\mu_k}, \sum_k$
- We can now present the alternating optimization algorithm for GMMs

Gaussian Mixture Models (GMM) Alternative optimization algorithms

- Data points $\{x_1, ..., x_N\}$, integer K > 1
- Result : Component Parameters$\{\mu_k, \sum_k\}$, mixing coefficients $\{\pi_k\}$

Choose some initial values for $\mu_k, \sum_k, \pi_k$;

1. Fix parameters and find $\mathrm{r_{nk}} = \frac{\pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k)}{\sum_{j=1}^{K} \pi_j N(\boldsymbol{x_n}|\boldsymbol{\mu_j}, \sum_j)}$
2. Fix responsibilities, and update parameters
   - $\boldsymbol{\mu_k}' = \frac{\mathbf{1}}{\boldsymbol{N_k}} \sum_{\boldsymbol{n=1}}^{\boldsymbol{N}} \boldsymbol{r_{nk}}\ \boldsymbol{x_n}$ $\qquad$ $\boldsymbol{where\ N_k = \sum_{n=1}^{N} r_{nk}}$
   - $\sum_k{}' = \frac{\mathbf{1}}{\boldsymbol{N_k}} \sum_{n=1}^{N} r_{nk}\ (\boldsymbol{x_n} - \boldsymbol{\mu_k}')(\boldsymbol{x_n} - \boldsymbol{\mu_k}')^{\boldsymbol{T}}$
   - $\pi_k = \frac{N_k}{N}$
3. Evaluate Log-likelihood:
   - $LogL(D|\boldsymbol{\pi}, \boldsymbol{\mu_k}, \sum_k) = \sum_{n=1}^{N} \log(\sum_{k=1}^{K} \pi_k N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \sum_k)$
4. Check for convergence, if not converged, go back to step 1 else stop.

Gaussian Mixture Models (GMM)

- Each iteration increases (or retains) the value of the log-likelihood.
  - Therefore, convergence to (local) maximum is guaranteed.
- This algorithm has a name – Expectation Maximization (EM)
- GMM converges slower than K-means and performs more computations per-iteration.
- Singularity issue:
  - If a component gets mapped to a training data point then:
  - $\frac{1}{(2\pi)^{D/2}} \frac{1}{\sqrt{|\sum|}} e^{\frac{-1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \sum^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} = \frac{1}{(2\pi)^{D/2}} \frac{1}{\sqrt{|\sum|}}$
  - Here if $|\sum|$ approaches to zero: this term can get infinite

Gaussian Mixture Models (GMM)

- Solution
  - Care must be taken to check if that has happened or is close to happening.
  - If so, the collapsing component should be reset to some other randomly chosen $\boldsymbol{\mu_k}$ and large $\sum_k$
  - Other part of the optimization algorithm should proceed as before
- GMM is used for performing density estimation as well.

Expectation Maximization Algorithm

- K-means and GMMs are examples of latent variable models.
- Specifically for GMMs, we have seen an incremental algorithm for learning the parameters using Machine Learning.
- That algorithm is actually an instance of a powerful framework called Expectation-Maximization (EM) .
- EM is used for solving latent variable problems using Machine Learning.
- We will now present a more general explanation of the EM algorithm.

Expectation Maximization Algorithm

- Maximum likelihood is equivalent to maximizing the log-likelihood

  $LogL(D|\boldsymbol{\theta})$ where $\boldsymbol{\theta}$ represents all parameters of a model.
- Using the sum-rule $LogL(D|\boldsymbol{\theta}) = \sum_{n=1}^{N} \log(\sum_{k=1}^{K} p(x_n|\theta_k)$
- Maximization is no longer straight-forward since log is ‘blocked’ by the summation.
- So we take another approach

Expectation Maximization Algorithm

- We will denote $\{x, z\}$ as the complete dataset.
  - Z represents latent variables
- We will denote $\{x\}$ as the incomplete dataset.
- The goal now is to maximize the complete-data log-likelihood function $p((x, z)|\, \theta)$.
- But for that we need to know the values of Z which are unobserved.
  - What can be computed about Z, however, is the posterior $p(z|\, x, \theta)$.
  - So instead of the incomputable log-likelihood, the next best computable number would be its expected-value under the posterior $p(z|\, x, \theta)$

Expectation Maximization Algorithm

- This yields the E-step of the EM algorithm.
  - $\mathrm{E}_{\mathrm{z|x},\theta_{old}} \log p(x,z|\theta) = Q(\theta,\theta_{old}) = \sum_{k=1}^{K} p(z|x,\theta_{old}) \log p(x,z|\theta)$
  - $= \sum_{k=1}^{K} p(z|x,\theta_{old}) \log p(\boldsymbol{z}|\theta)\, p(\boldsymbol{x}|z,\theta)$
- Since we are eventually interested in optimal parameters θ_{new} we treat this expectation as a function of θ and denote it by $Q(\theta,\theta_{old})$.
- The M-step corresponds to maximizing this expectation
  - $\theta_{new} = argmax_{\theta}\, Q(\theta,\theta_{old})$
- In short, EM replaces the log-likelihood by the expected log-likelihood and maximizes it.

Expectation Maximization Algorithm

- Goal is to maximize likelihood $p(z|x,\theta)$ with respect to θ by introducing joint distribution $p(x,z|\ \theta)$ involving latent variables Z.
- Steps:
  1. Choose initial θ_{old}
  2. E-step: Evaluate $p(z|x,\theta_{old})$ i.e. responsibilities by using θ_{old}
  3. M-step : Obtain new estimate θ_{new} by maximizing the expectation $Q(\theta,\theta_{old})$
     i. $\theta_{new} = argmax_{\theta}\ Q(\theta,\theta_{old})$
  4. Check for convergence of either log-likelihood or parameters. If not converged, then
     i. $\theta_{old} \leftarrow \theta_{new}$*(1)*
     ii. *return to step 2.*