Machine Learning

AIML CLZG565

Unsupervised Learning

Raja vadhana P

Assistant Professor,

BITS - CSIS

BITS Pilani

Pilani Campus

Disclaimer and Acknowledgement

- The content for these slides has been obtained from books and various other source on the Internet
- I here by acknowledge all the contributors for their material and inputs.
- I have provided source information wherever necessary
- I have added and modified the content to suit the requirements of the course

Source: Slides of Prof. Chetana, Prof.Vimal, Prof.Seetha, Prof.Sugata, Prof.Monali, Prof. Raja vadhana , Prof.Anita from BITS Pilani , CS109 and CS229 stanford lecture notes, Tom Mitchell, Andrew Ng and many others who made their course materials freely available online.

Course Plan

M1	Introduction & Mathematical Preliminaries
M2	Machine Learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

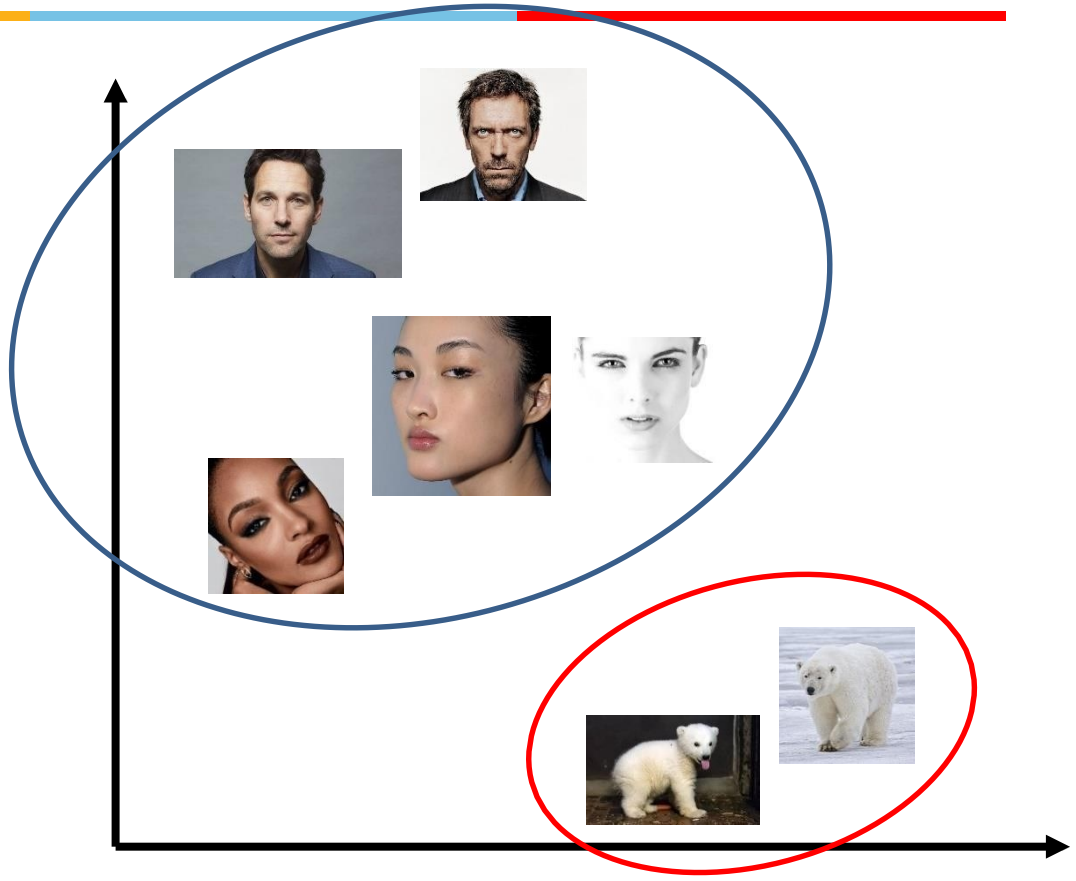
Module –Unsupervised Learning

- Mixture Models
- Expectation Maximization (EM) Algorithm
- K-means Clustering

Unsupervised Learning

- We only use the features X, not the labels Y
- This is useful because we may not have any labels but we can still detect patterns
- For example:
  - We can detect that news articles revolve around certain topics, and group them accordingly
  - Discover a distinct set of objects appear in a given environment, even if we don't know their names, then ask humans to label each group
  - Identify health factors that correlate with a disease
- Clustering: Grouping items that "belong together" (i.e. have similar features)

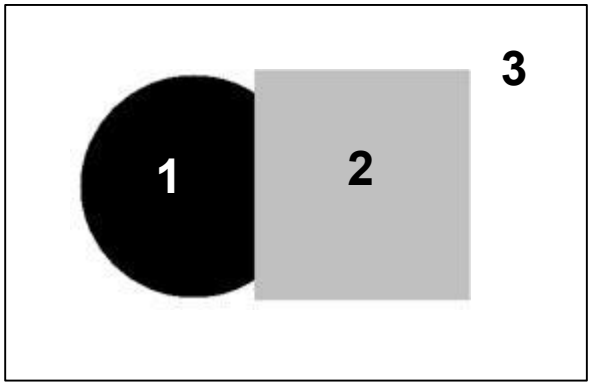
Unsupervised discovery



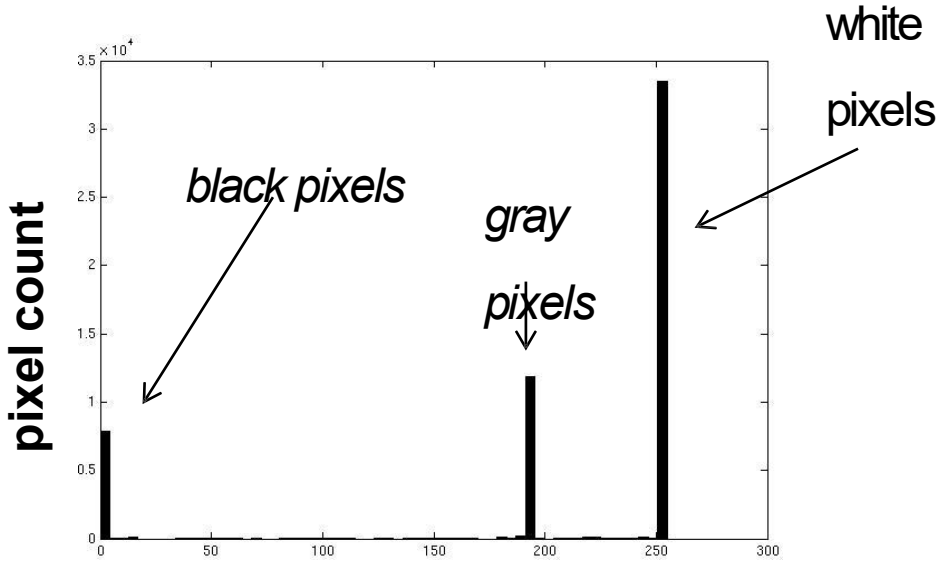
Clustering algorithms

- In depth
  - K-means (iterate between finding centers and assigning points)
  - Gaussian Mixture Models (GMMs)

Image segmentation: toy example



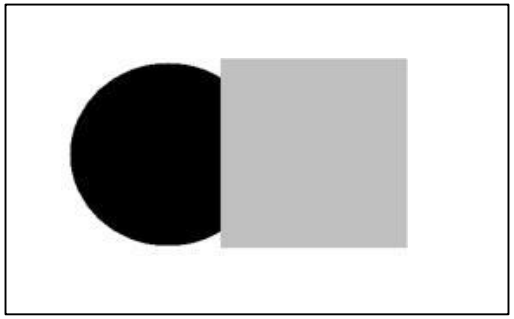
input image

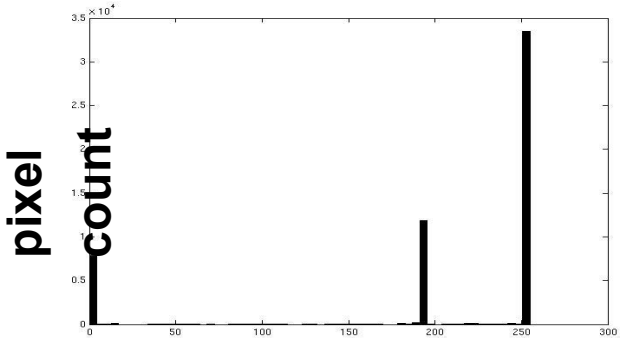


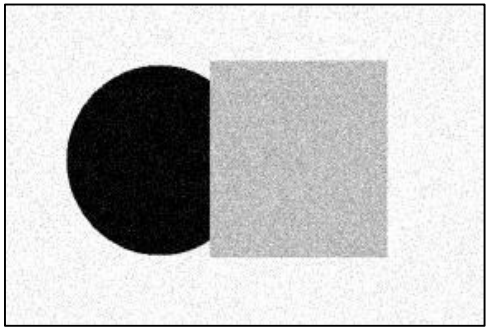
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn't quite so simple?

Source: K. Grauman

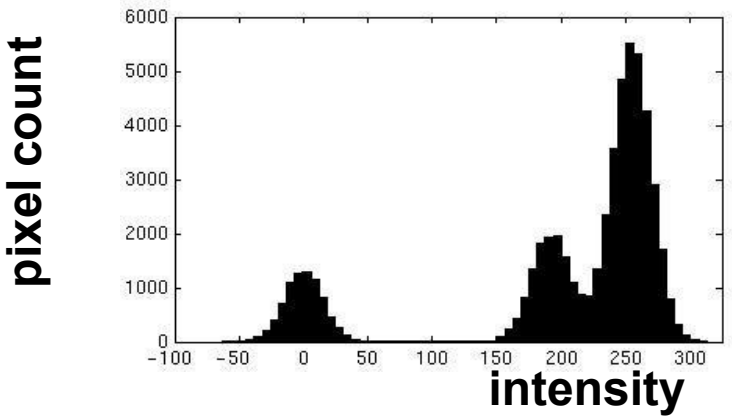
Image segmentation: toy example







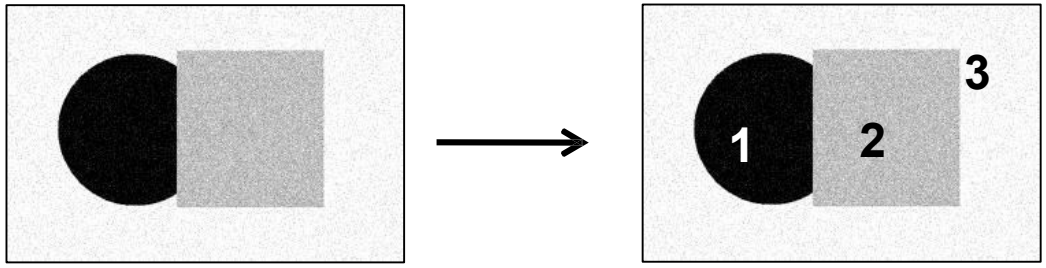
input image



- Now how to determine the three main intensities that define our groups?
- We need to **cluster.**

Source: K. Grauman

Image segmentation: toy example



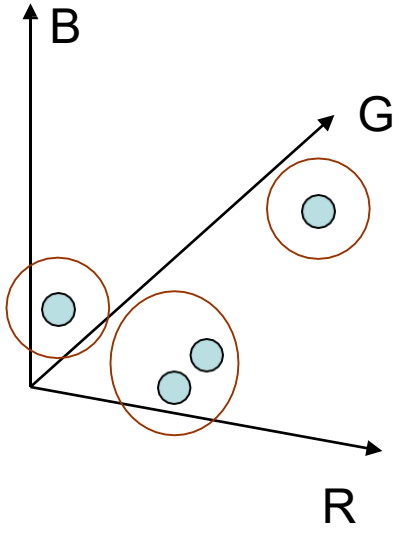
- **Goal: choose three "centers" as the representative intensities, and label every pixel according to which of these centers it is nearest to.**
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i: $$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

Source: K. Grauman

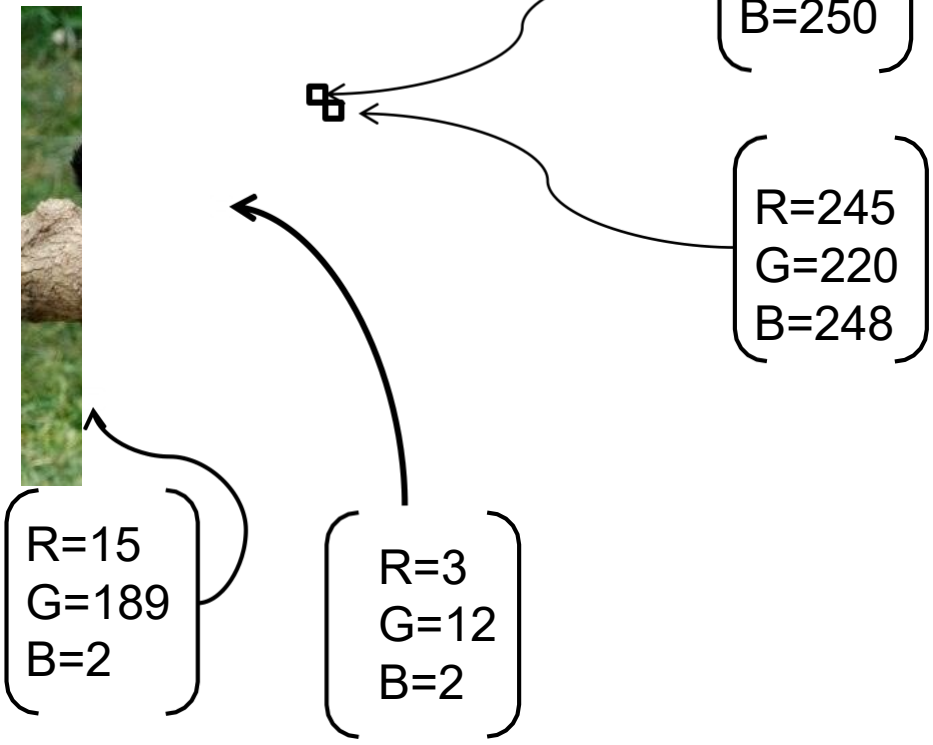
Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity



Feature space: color value (3-d)



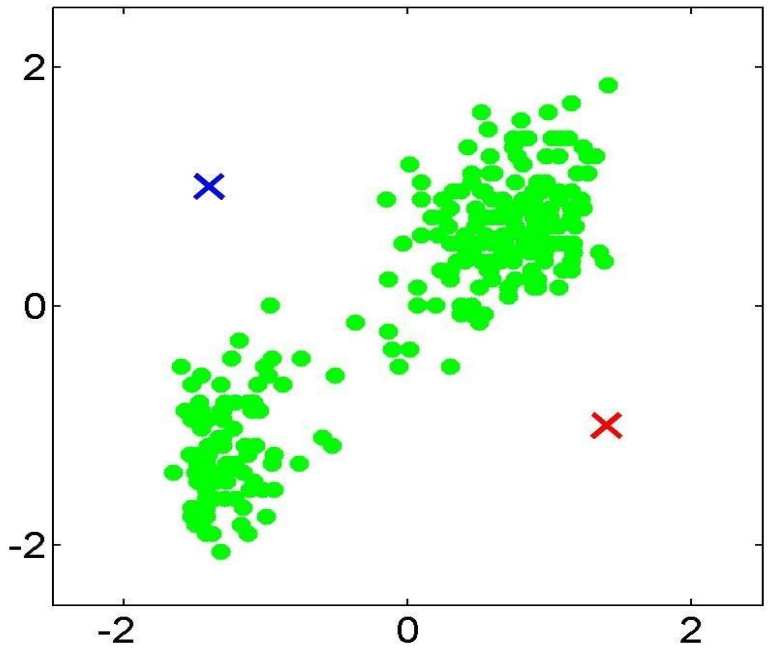
Source: K. Grauman

Image Compression using Segmentation

Smaller values of *K* give higher compression at the expense of poorer image quality.

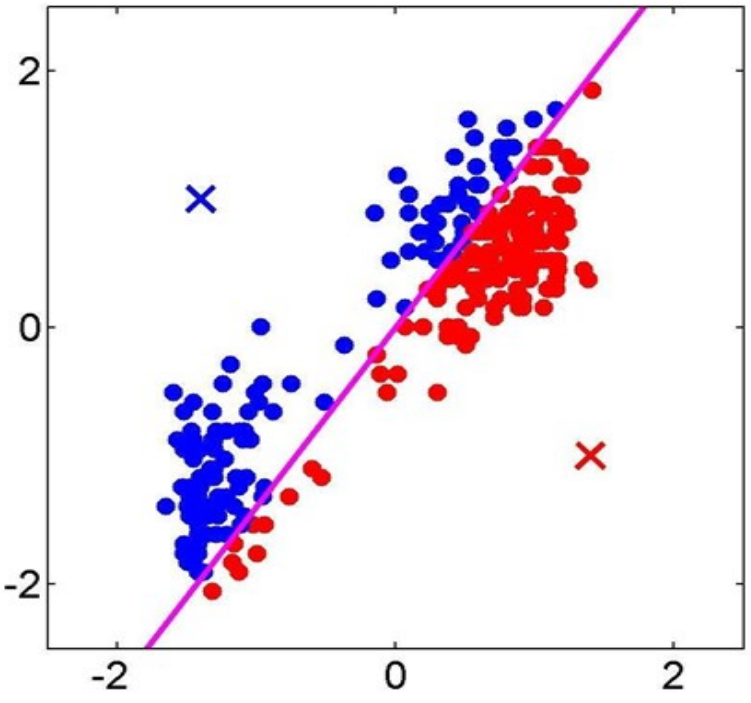
Expectation Maximization Algorithm

Example



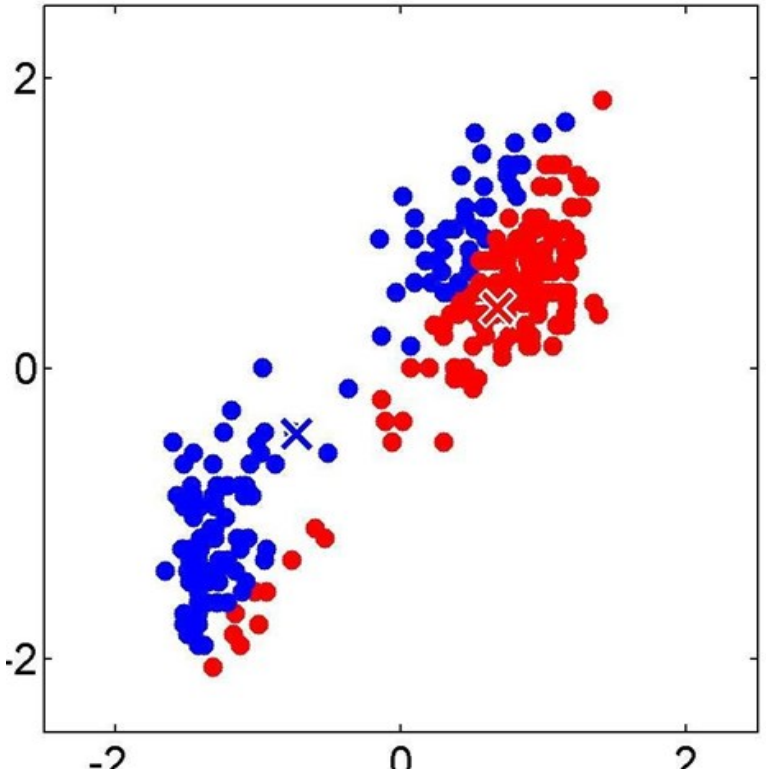
- Pick K random points as cluster centers (means)
- Shown here for K=2
- The initial choices for centres $\mu 1$ and $\mu 2$ are shown by the red and blue crosses, respectively

Iterative Step 1
Assign data points to closest cluster center



- In the initial E step, each data point is assigned either to the red cluster or to the blue cluster, according to which cluster center is nearer.
- This is equivalent to classifying the points according to which side of the perpendicular bisector of the two cluster centers, shown by the magenta line, they lie on.

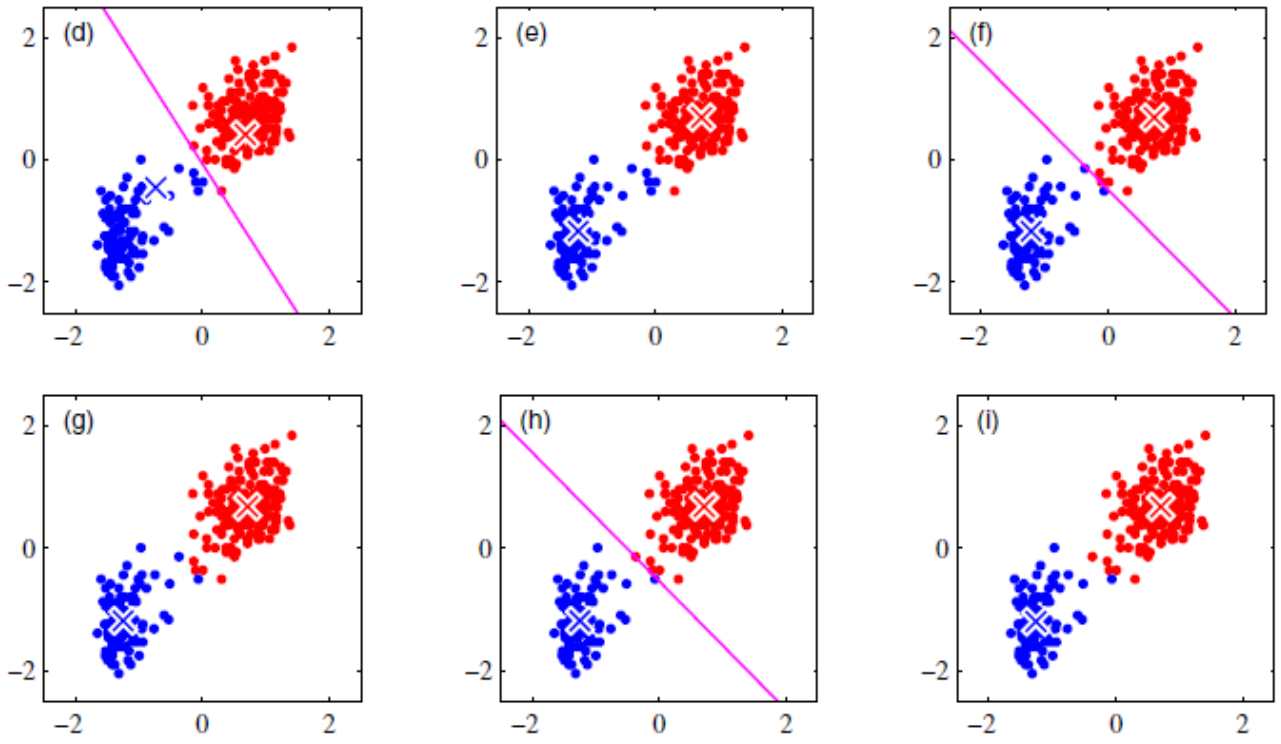
Iterative Step 2
Change the cluster center to the average of the assigned points



- In the M step, each cluster centre is re-computed to be the mean of the points assigned to the corresponding cluster.

Repeat until convergence

Repeat to alternate E and M steps through to final convergence of the algorithm.



k-Means Clustering

Exercise

innovate achieve lead

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \cdots (x_{ip} - x_{jp})^2}$$

Use the k-means algorithm and Euclidean distance to cluster the following 5 examples into 2 clusters:

A(0, 1) B(3, 0) C(2, 4) D(2, 1) E(3, 5)

Ex: Distance from A to C
=sqrt[(2-0)^2 + (4-1)^2]=3.61

Step 1 : Compute the Euclidean distance matrix

	A	B	C	D	E
A	0	3.16	3.61	2	5
B		0	4.12	1.414	5
C			0	3	1.414
D				0	4.12
E					0

Step 2 : Randomly choose two cluster centroids.
Let A and C be the cluster centroids.

Step 3 : Cluster 1: { A, B, D }
Cluster 2: { C. E}

k-Means Clustering Exercise

Step 4 : Compute the Centroids,

Cluster1 = mean of A,B,D

$= \frac{0+3+2}{3}, \frac{1+0+1}{3} = (1.66, 0.66)$

Cluster 2 = mean of C,E $= \frac{2+3}{2}, \frac{4+5}{2} = (2.5, 4.5)$

Step 5 : Repeat; Compute the Euclidean distance matrix to the cluster centers.

	A	B	C	D	E	C1	C2
A	0	3.16	3.61	2	5		
B		0	4.12	1.414	5		
C			0	3	1.414		
D				0	4.12		
E					0		
C1	1.69	1.49	3.35	0.48	4.54	0	
C2	4.30	4.52	0.70	3.53	0.70	3.93	0

k-Means Clustering Exercise

Step 6 : Cluster 1: { A, B, D }
Cluster 2: { C, E}

Step 7 : Recompute the Centroids,
Cluster1 = mean of A,B,D
$= \frac{0+3+2}{3}, \frac{1+0+1}{3} = (1.66, 0.66)$
Cluster 2 = mean of C,E $= \frac{2+3}{2}, \frac{4+5}{2} = (2.5, 4.5)$

Step 8 : Stop as converged or no change is seen.

1 of K coding mechanism

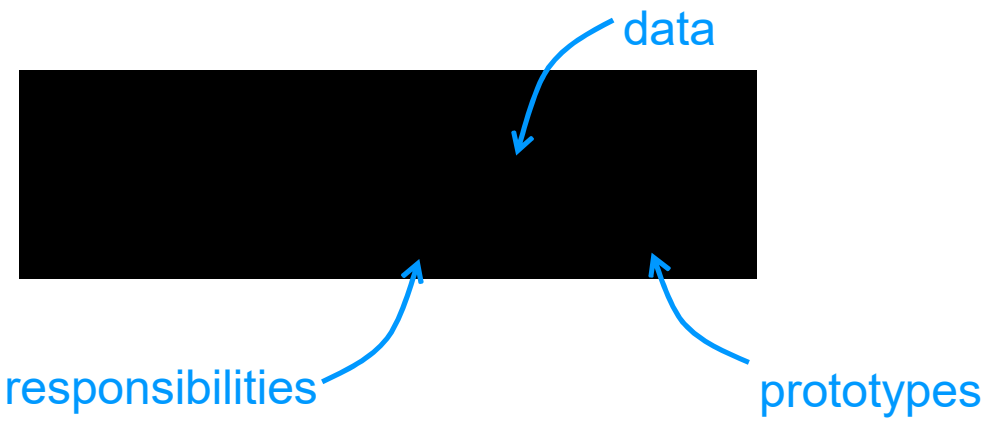
- For each data point x_n, we introduce a set of binary indicator variables $r_{nk} \in \{0,1)$ such that
- $$\sum_k r_{nk} = 1$$

where k =1,...,K describing which of the K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then r_{nk} =1, and r_{nj} =0 for j not equal to k.

- Example: 5 data points and 3 clusters

$$(r_{nk}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

K-means Cost Function



- **Goal is to find values for the $\{r_{nk}\}$ and the $\{\mu_k\}$ so as to minimize J** *(Distortion measure)*
- relatively slow, because in each E step it is necessary to compute the Euclidean distance between every prototype vector and every data point

Minimizing the Cost Function

- E-step: minimize J w.r.t r_{nk}

choose some initial values for the μ_k. minimize J with respect to the r_{nk}, keeping the μ_k fixed. Assign the nth data point to the closest cluster centre

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

- M-step: minimize J w.r.t μ_k

minimize J with respect to the μ_k, keeping r_{nk} fixed. The objective function J is a quadratic function of μ_k, and it can be minimized by setting its derivative with respect to μ_k to zero giving

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}}.$$

K-means Algorithm

- **Goal:** Represent a data set in terms of *K* clusters each of which is summarized by a prototype μ_k
- Initialize prototypes, then iterate between two phases:
  - **E-step:** assign each data point to nearest prototype
  - **M-step:** update prototypes to be the cluster means
- Simplest version is based on Euclidean distance

K Means Clustering

CGPA	Technical Interview	HR Discussion
5	10	9
6	10	9
6	8	8
7	8	8

I Standardize the data if required:

II Fix the no.of.cluster expected

III Initialize the prototypes:

IV Expectation-Step: Fix prototype & find the membership for which the distortion is minimum

V Maximization Step: Fix the membership(responsibility matrix) and re-estimate the prototype

VI Repeat E & M Step till convergence is achieved:

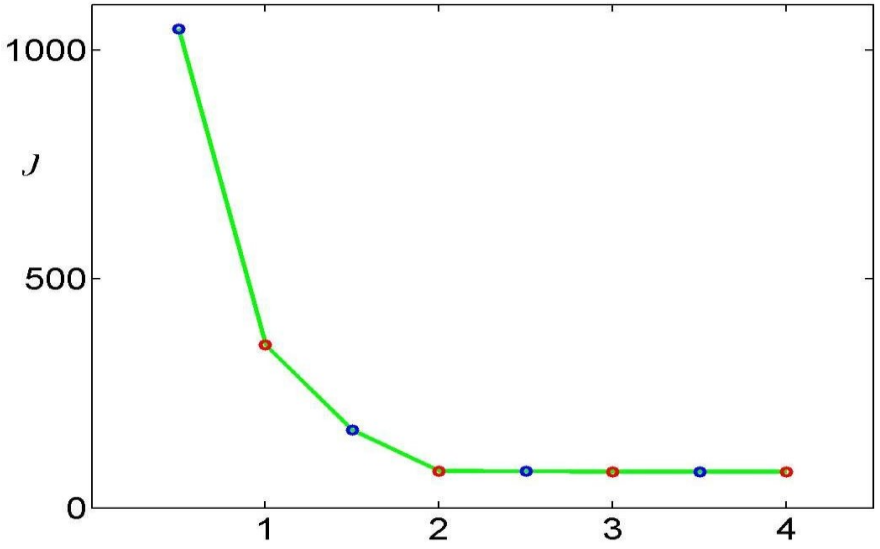
- •Centroids of newly formed clusters do not change
- •Points remain in the same cluster
- •Maximum number of iterations are reached

$$\sqrt{(5-6)^2+(10-7)^2+(9-7)^2}$$

K=2	C1=(5,9,9)	C2=(6,7,7)
D(p1,Ci)	1	3.87
D(p2,Ci)	1.41	3.61
D(p3,Ci)	1.73	1.41
D(p3,Ci)	2.45	1.73
Membership	Cluster 1 = {p1, p2}	Cluster 2 = {p3,p4}
New Centroid Take mean of cluster points. This is used in next iteration	C1= (5.5,10,9)	C2=(6.5,8,8)

Convergence

- Each E step (blue points) and M step (red points) . The algorithm has converged after the third M step, and the final EM cycle produces no changes in either the assignments or the prototype vectors.
- Because each phase reduces the value of the objective function *J*, convergence of the algorithm is assured. However, it may converge to a local rather than global minimum of *J*



Another way of writing objective function

K-means:

- The center of a cluster is not necessarily one of the input data points (it is the average between the points in the cluster).
- Uses squared Euclidean distance as the measure of dissimilarity between a data point and a prototype vector
- Not suitable where some or all of the variables represent categorical labels

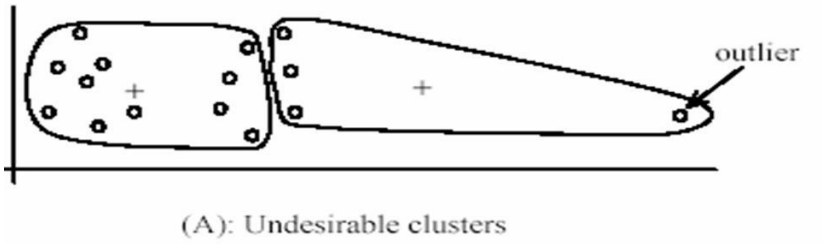
$$\widetilde{J} = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \mathcal{V}(\mathbf{x}_n, \boldsymbol{\mu}_k)$$

K-medoids (more general distances):

- It is common to restrict each cluster prototype to be equal to one of the data vectors assigned to that cluster and can be used with **arbitrary distances**
- k-medoids more robust to noise and outliers as compared to k-means because it minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances.

Limitations of K-means

- Setting k?

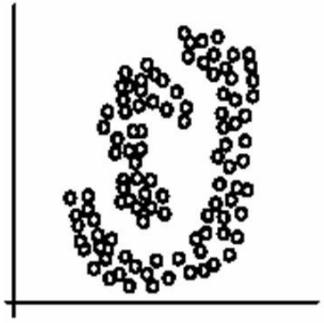


(A): Undesirable clusters

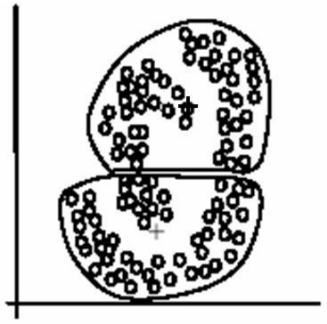
- Sensitive to outliers

(B): Ideal clusters

- Detects spherical clusters



(A): Two natural clusters

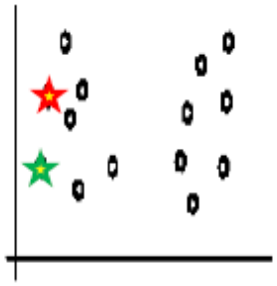


(B): *k*-means clusters

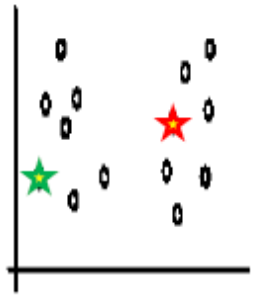
Source: K. Grauman

Limitations of K-means

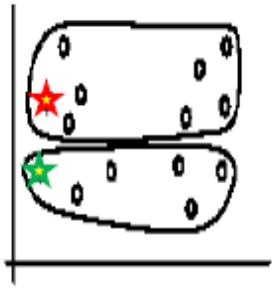
- Sensitive to initial centers
  - Use heuristics or output of another method



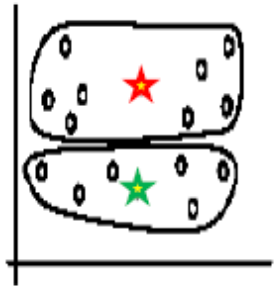
Random selection of seeds (centroids)



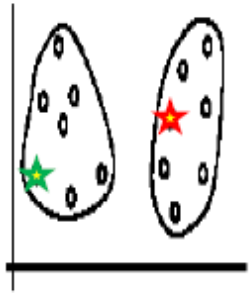
Random selection of seeds (centroids)



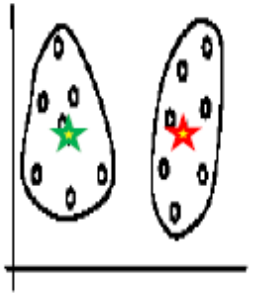
Iteration 1



Iteration 2



Iteration 1



Iteration 2

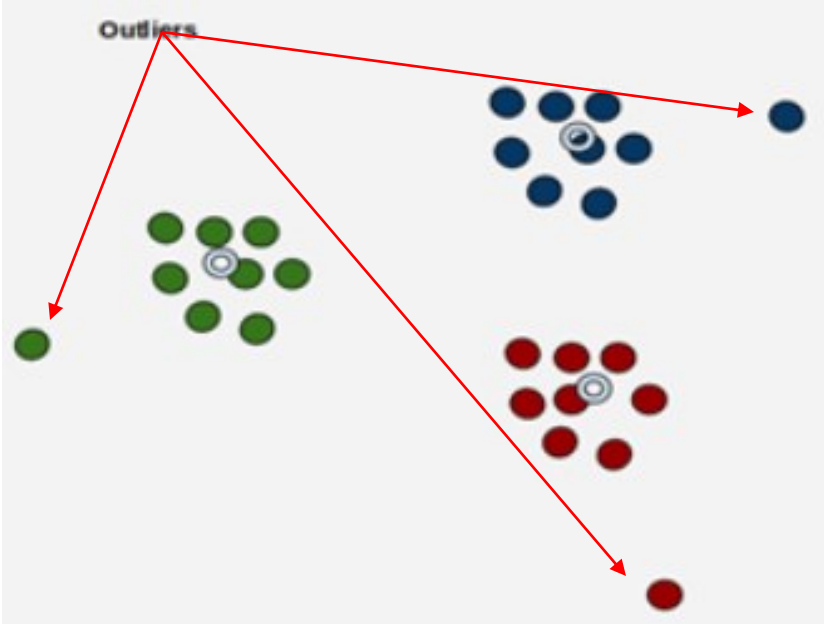
K Means – Notion of Outliers

Outliers are relatively away from the centroid

K-Means Clustered points:
dist(x, μk) be the distance of a point x, assigned to cluster k to its center μk.
Lμk be the average distance of all the points assigned to cluster k with its centre

The **ratio dist(x, μk) / Lμk** for each point is the outlier score for each point.

Higher the ratio for a point x, **more likely x is a outlier.**



K Means – Outlier Detection

Candidate	Round 1 Rating	Round 2 Rating
A	0	1
B	3	0
C	2	4
D	2	1
E	3	5

Outliers are relatively away from the centroid

K-Means Clustered points:
dist(x, μk) be the distance of a point x, assigned to cluster k to its center μk.
Lμk be the average distance of all the points assigned to cluster k with its centre

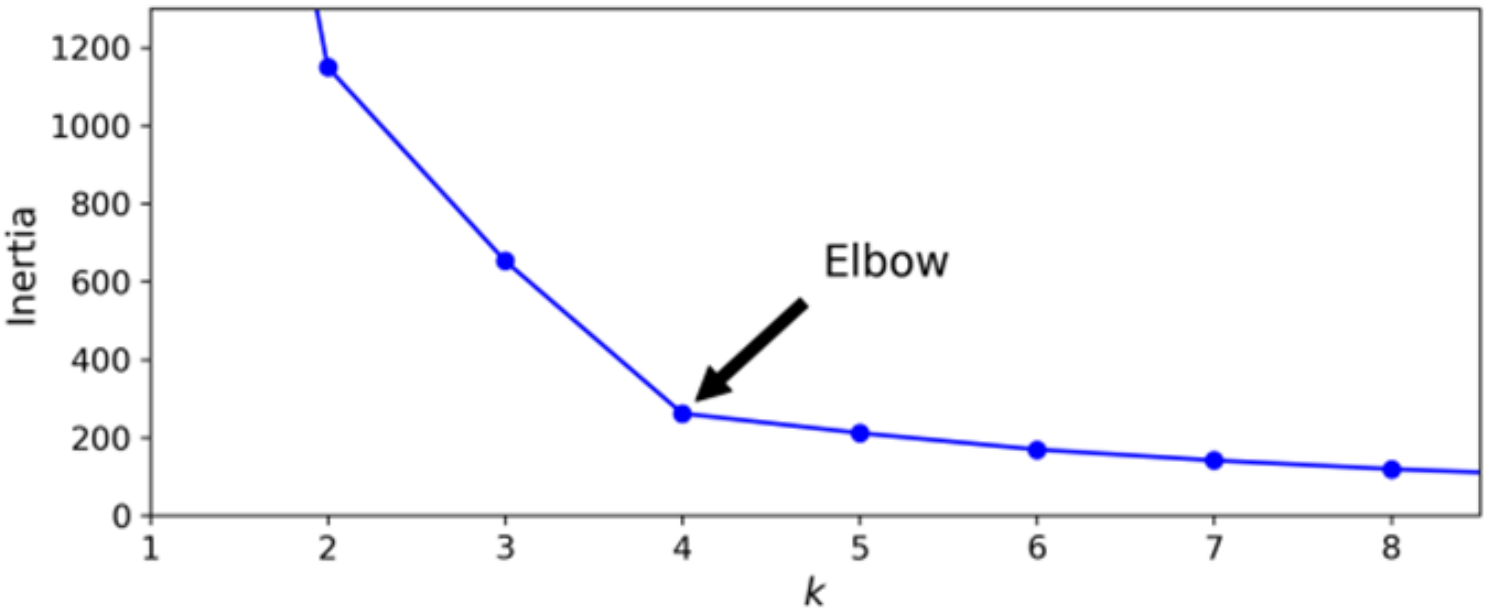
The **ratio dist(x, μk) / Lμk** for each point is the outlier score for each point.

Higher the ratio for a point x, **more likely x is a outlier.**

Random centroid assigned at the first iteration of K-Means

K=2	C1=(1.66, 0.66)	C2=(2.5,4.5)	dist(x, μk) / Lμk
D(A, Ci)	1.7		1.39
D(B, Ci)	1.49		1.22
D(C, Ci)	0.47		0.38
D(D, Ci)		0.71	1
D(E, Ci)		0.71	1
Membership calculated for 1st iteration are : →	Cluster 1 = {A,B,C,C1}	Cluster 2 = {D,E,C2}	
Lμk	1.22	0.77	

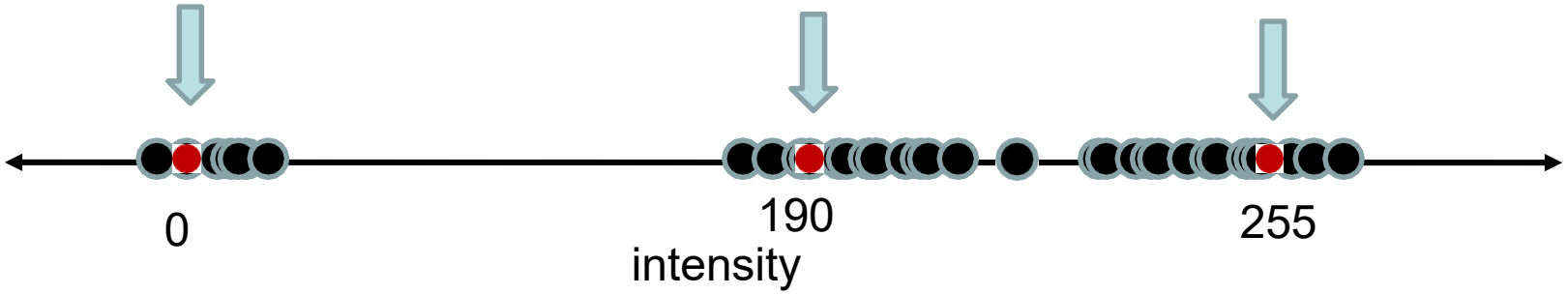
K Means –K Value



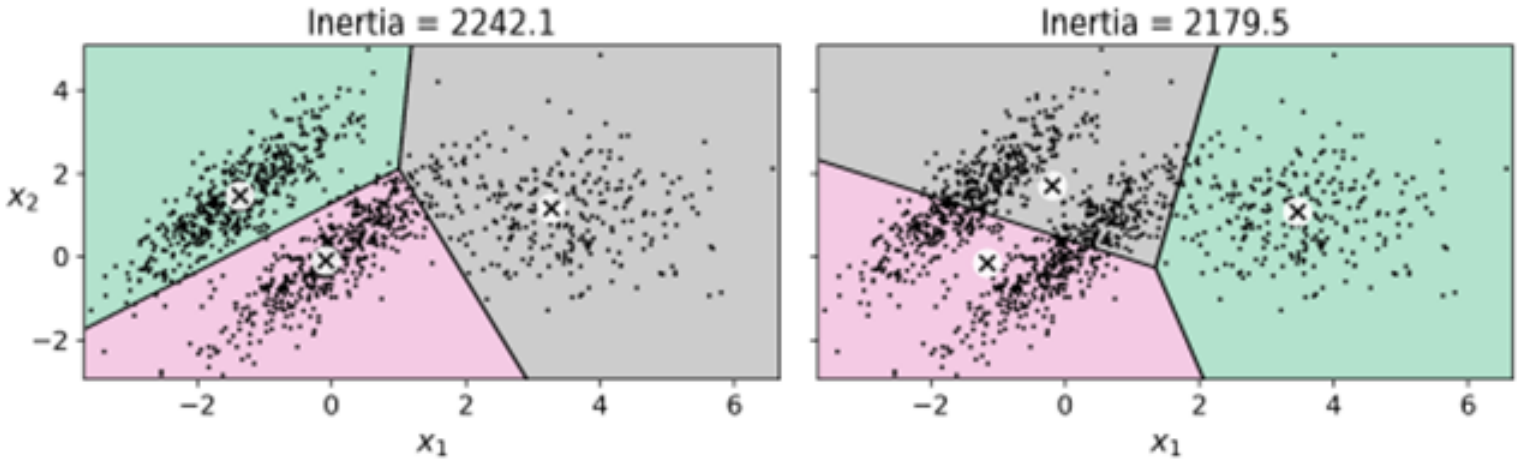
Notion of Mixture Models

K means – Hard clustering

- Assigned each example to exactly one cluster
- What if clusters are overlapping?
  - Hard to tell which cluster is right
  - Maybe we should try to remain uncertain
- What if cluster has a non-circular shape?
- Hard assignments of data points to clusters – small shift of a data point can flip it to a different cluster
- For example, if a point is near the 'border' between two clusters, it's often better to know that it has near equal membership probabilities for these clusters, rather than blindly assigning it to the nearest one.



K Means – Shortcomings in detecting non-spherical clusters



1 of K coding mechanism

- For each data point x_n, we introduce a set of binary indicator variables $r_{nk} \in [0,1]$ such that
- $$\sum_k r_{nk} = 1$$

where k =1,...,K describing which of the K clusters the data point x_n is assigned to, so that if data point x_n is assigned to cluster k then $r_{nk} > r_{nj}$ for j not equal to k.

- Example: 5 data points and 3 clusters

$$(r_{nk}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

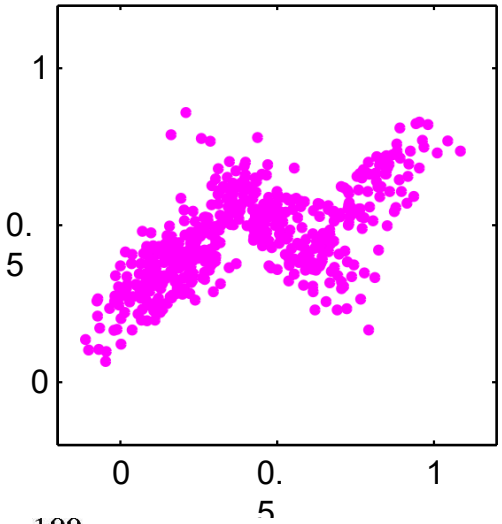
GMM - soft Clustering

- **Solution**: replace ‘hard’ clustering of K-means with ‘soft’ probabilistic assignments
- Probabilistic Clustering
  - Represents the probability distribution of the data as a **Gaussian mixture model (GMM)**
  - GMMs give a probabilistic assignment of points to clusters which quantify uncertainty.
  - Mixture models combines a set of distributions to create a convex space where we can search for the optimal parameters for such distributions using MLE.
- In GMM, Clusters modeled as Gaussians
- **EM algorithm:** assign data to cluster with some probability

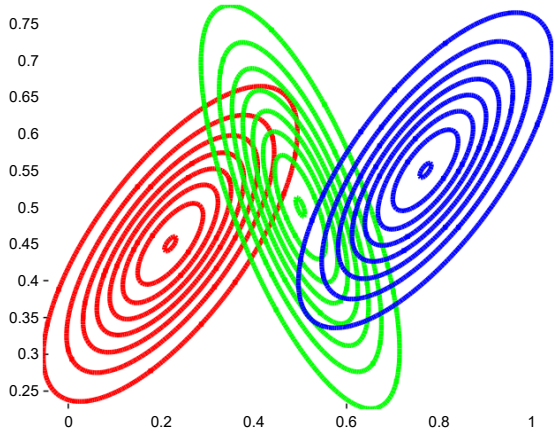
Source Credit: Christopher M. Bishop

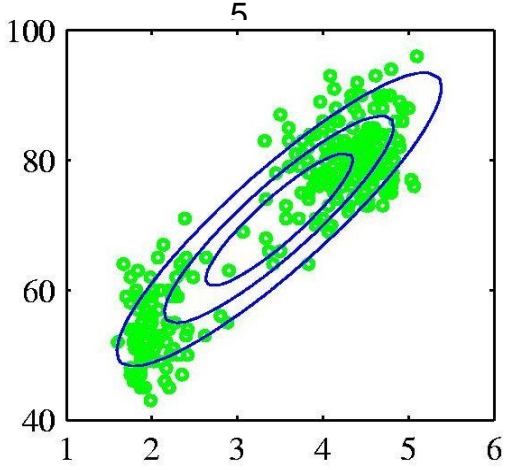
Learning a Mixture of Gaussians

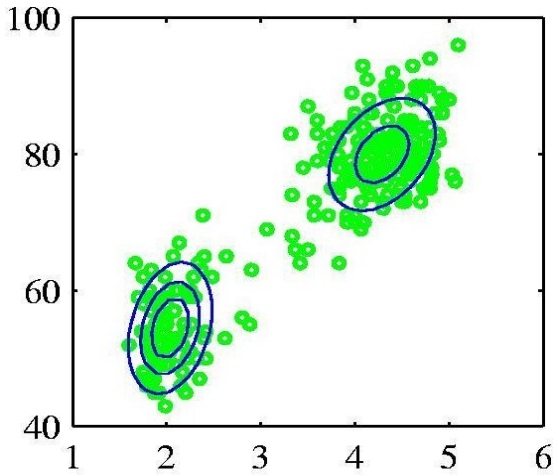
Our actual observations



Mixture of 3 Gaussians

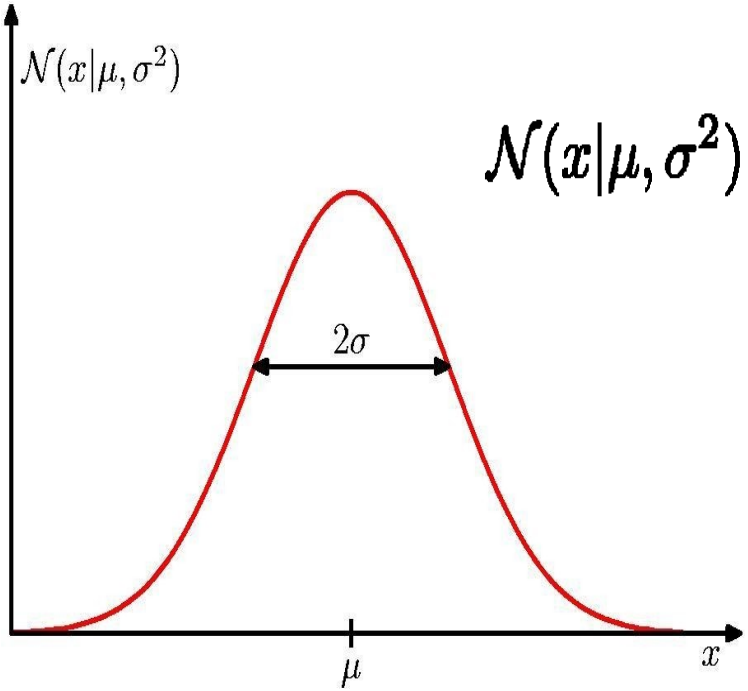






Source Credit :CS229: Machine Learning ©2021 Carlos Guestrin

Review: Gaussian Distribution

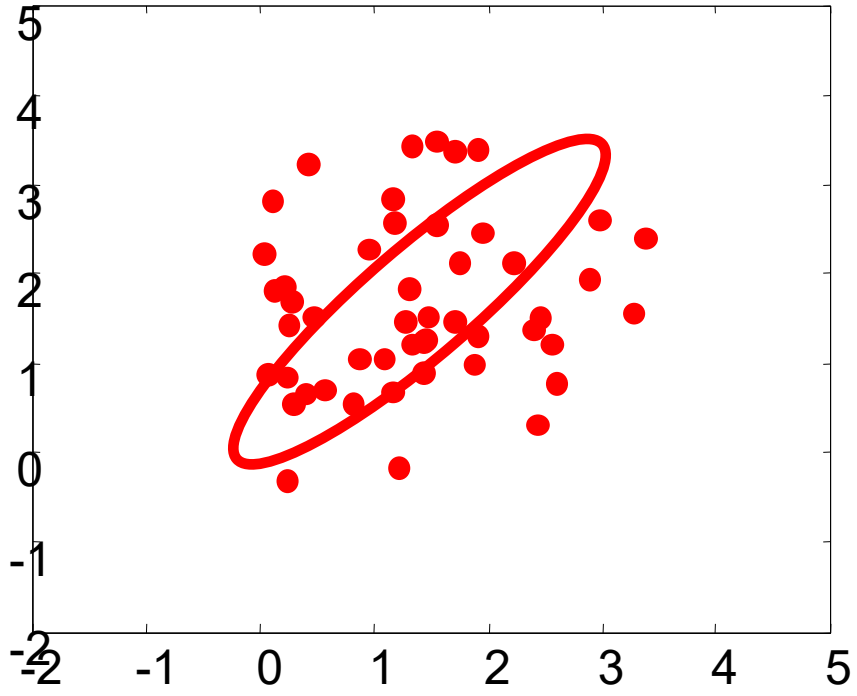


$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Source Credit : Chris Bishop

Multivariate Gaussian models

$$\mathcal{N}(\underline{x}\ ;\ \underline{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}}|\Sigma|^{-1/2}\exp\left\{-\frac{1}{2}(\underline{x}-\underline{\mu})^T\Sigma^{-1}(\underline{x}-\underline{\mu})\right\}$$



Maximum Likelihood estimates

$$\hat{\mu} = \frac{1}{N}\sum_i x^{(i)}$$

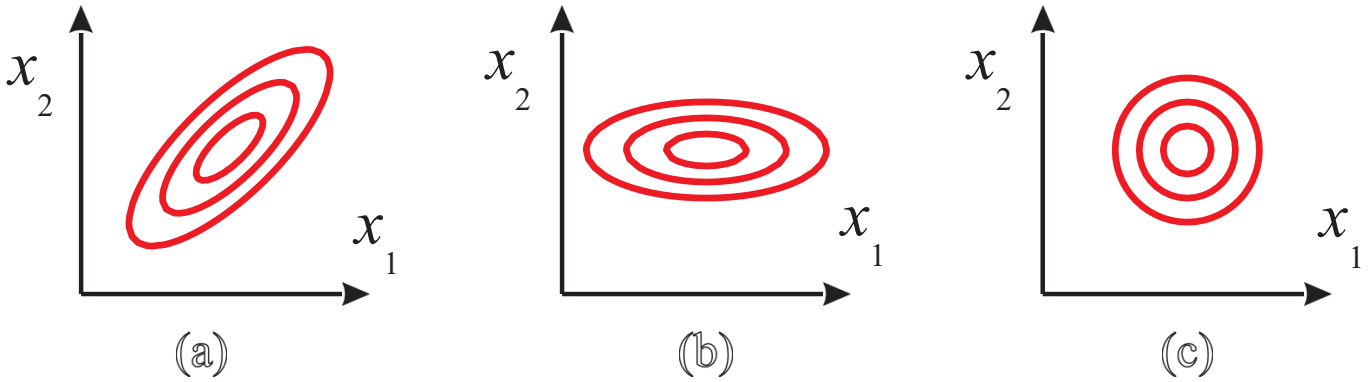
$$\hat{\Sigma} = \frac{1}{N}\sum_i (x^{(i)}-\hat{\mu})^T(x^{(i)}-\hat{\mu})$$

We' ll model each cluster using one of these Gaussian "bells"…

Slide Credit: Alexander Ihler

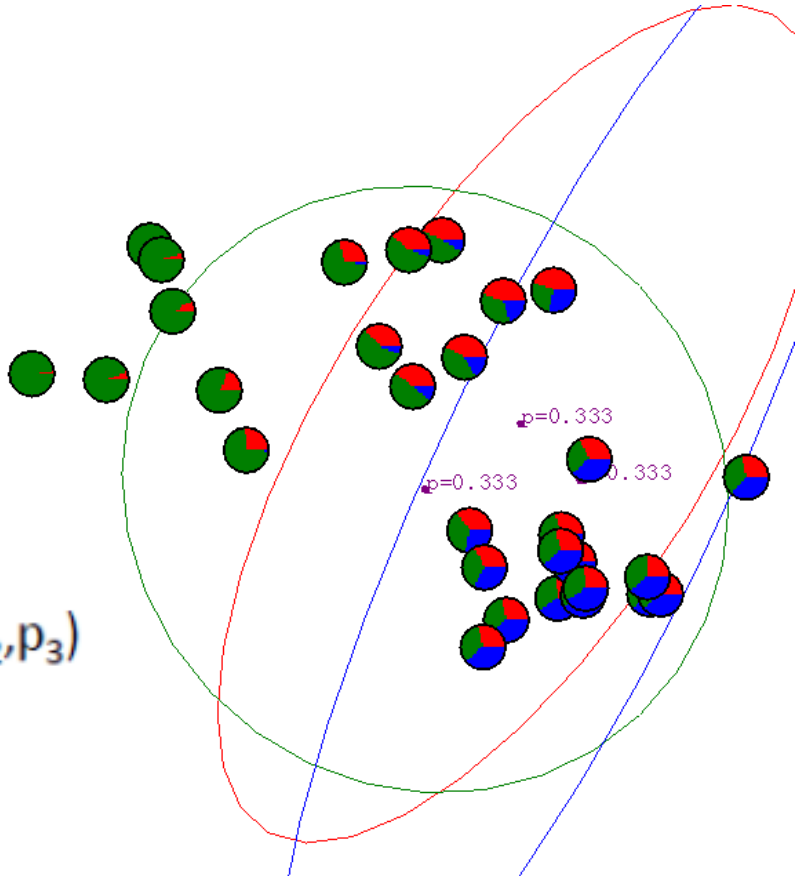
Example: Mixture of 3 Gaussians

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}$$

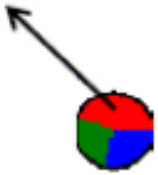


- The observed data come from a Gaussian mixture distribution which consists of K Gaussians with their own means and variances.
- K classes are latent

Gaussian Mixture Example: Start

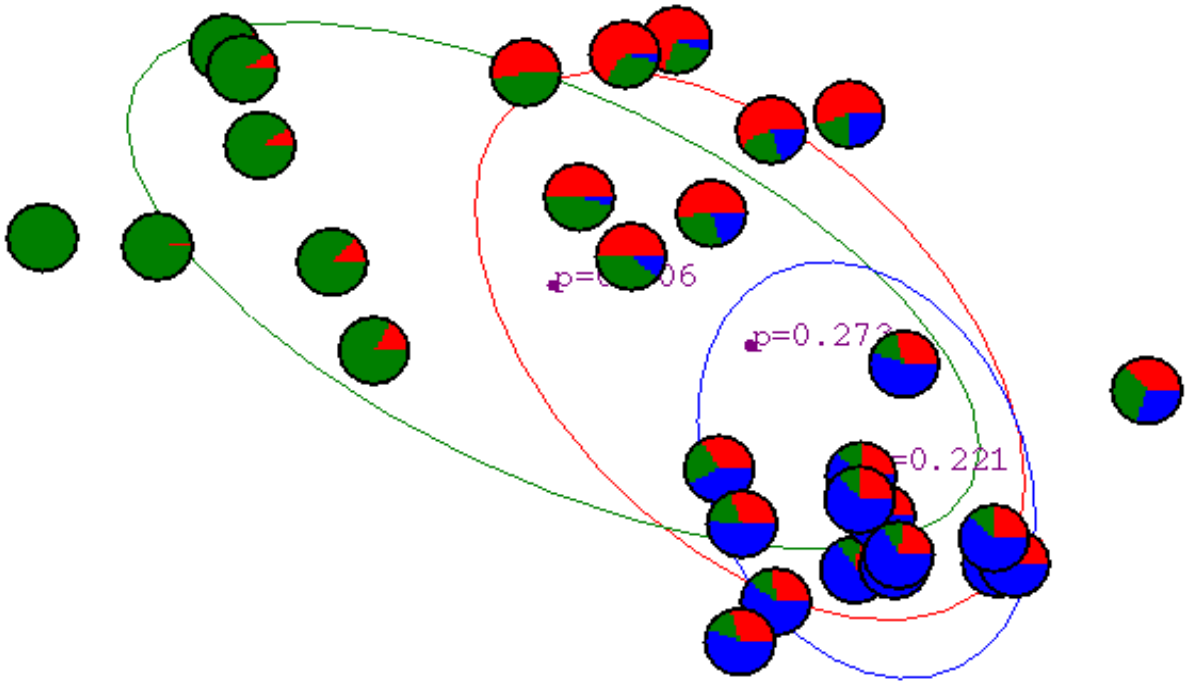


$$P(y = \bullet \mid x_j, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3, p_1, p_2, p_3)$$



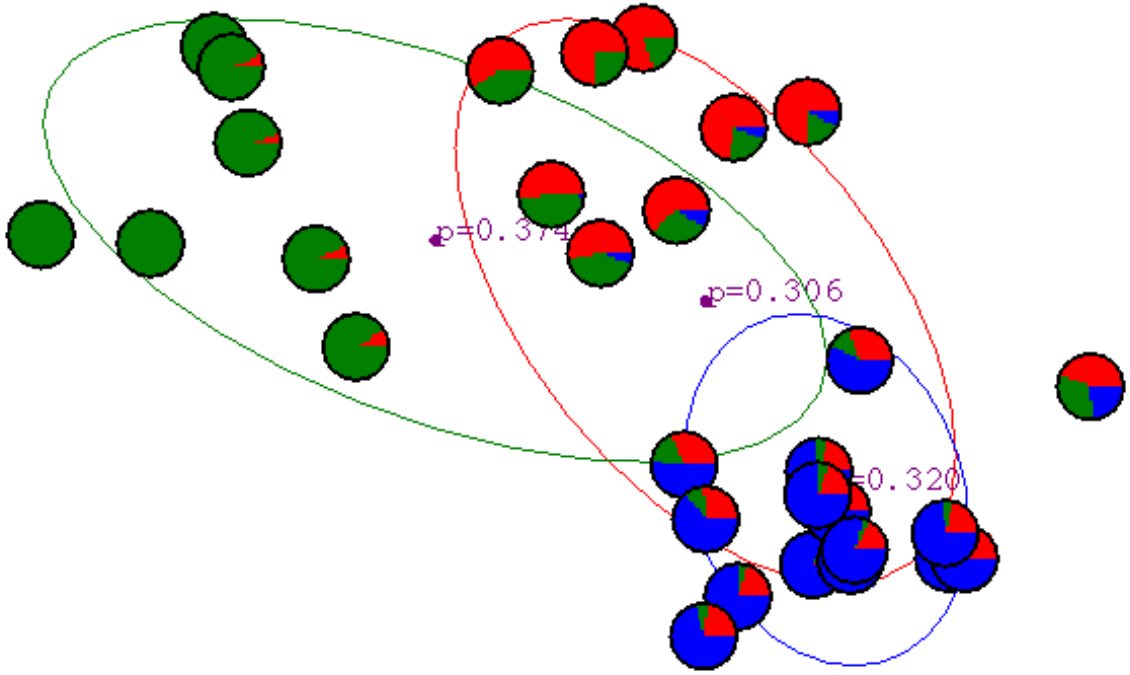
Slide Credit: Andrew W. Moore

After first iteration



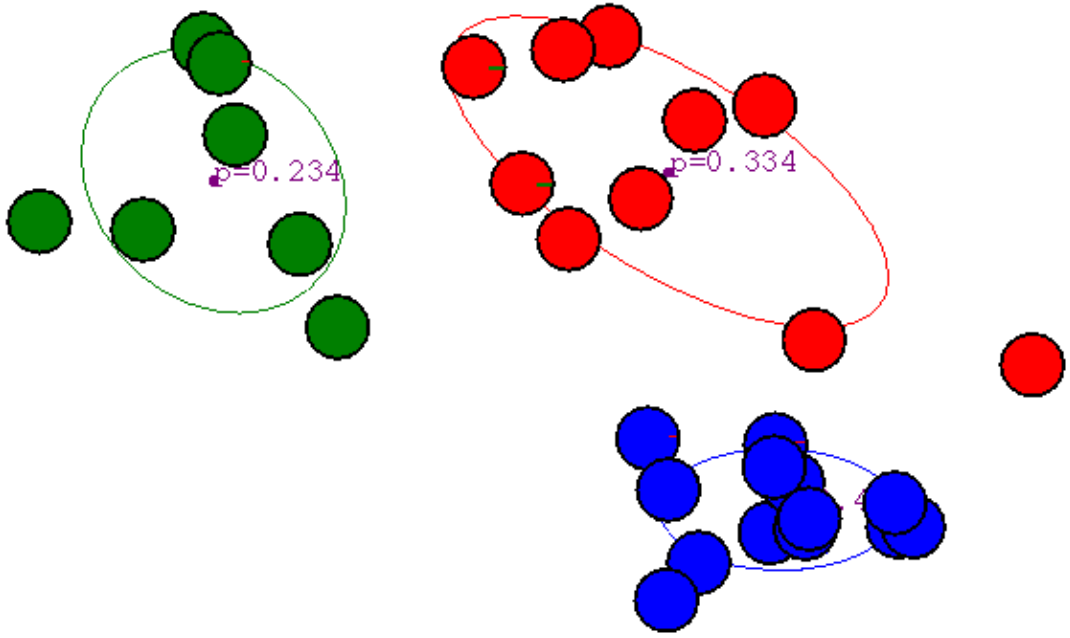
Slide Credit: Andrew W. Moore

After 2nd iteration



Slide Credit: Andrew W. Moore

After 20th iteration

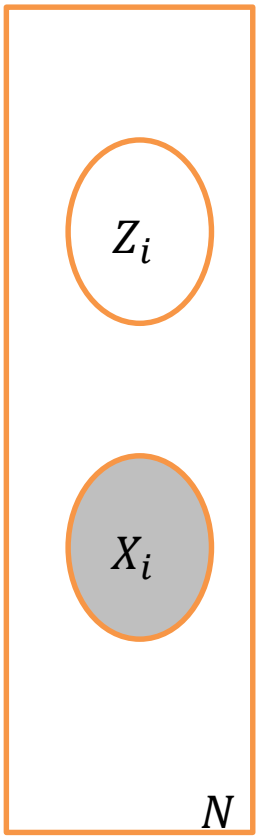


Slide Credit: Andrew W. Moore

GMM as Latent Variable model

- **Each data point is associated with a latent variable(new binary random variable Z_i for each X_i) that indicates which cluster it belongs to.**
- When fitting a GMM, we learn a distribution over these latent variables.
- This gives a probability that each data point is a member of each cluster. Given a data point x, what is the probability it came from Gaussian k
- Latent / hidden: not observed in the data

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$



GMM equations

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

Mixture models

K = number of mixture components (clusters),

π_j = mixture weights

$p_j(x)$ = family of distributions

(Gaussian, Poisson, Bernoulli, etc).

$$p(\mathbf{x}) = \sum_{j=1}^{K} \pi_j p_j(\mathbf{x})$$

$$0 \leq \pi_j \leq 1, \quad \sum_{j=1}^{K} \pi_j = 1$$

Gaussian Mixture models

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j})$$

$$0 \leq \pi_j \leq 1, \quad \sum_{j=1}^{K} \pi_j = 1$$

θ = collection of all the parameters of the model (mixture weights, means, and covariance matrices):

$$\boldsymbol{\theta} := \{\pi_1, \cdots, \pi_K, \boldsymbol{\mu_1}, \cdots, \boldsymbol{\mu_K}, \boldsymbol{\Sigma_1}, \cdots, \boldsymbol{\Sigma_K}\}$$

Parameters of GMM

X	Assuming Latent Variable z for 2 mixture components.	
	z(1)	z(2)
x(1)	0	1
x(2)	1	0
...		
x(N)	1	0

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

$\pi : \{\pi_1, \ldots, \pi_k\}$
$\mu : \{\mu_1, \ldots, \mu_K\}$
$\Sigma : \{\Sigma_1, \ldots \Sigma_K\}$

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$\boldsymbol{\theta} := \{\pi_1, \cdots, \pi_K, \boldsymbol{\mu_1}, \cdots, \boldsymbol{\mu_K}, \boldsymbol{\Sigma_1}, \cdots, \boldsymbol{\Sigma_K}\}$$

Mixture Density

GMM Model - π_k Mixing weights

- Marginal distribution over z is specified in terms of the mixing coefficients πk, such that p(zk = 1) =πk

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

- $0 \leq \pi_k \leq 1$ and $\Sigma\pi_k = 1$

- K-dimensional binary random variable z having a 1-of-K representation in which a particular element zk is equal to 1 and all other elements are equal to 0.

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

GMM Model- p(x|z) Component Density

- p(x|z) : Conditional distribution of x given a particular value for z is a Gaussian. Mixture densities

$p(\mathbf{x}|z_k = 1) = N(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$$

- z uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}$$

GMM as Clustering algorithm

- For every observed data point xi there is a corresponding latent variable z_i
- posterior probability of a given instance xi belonging to component k, referred to as the 'responsibility' of component k for producing xi , denoted as $\gamma(z_k) = p(z_k = 1 \mid x)$
- This gives us the probabilities of x_i belonging to the different components.
- That is precisely how a GMM can be used to cluster data.
- Use Bayes' theorem

GMM Model - the 'responsibility' of component k for producing x_i

Using Bayes' theorem

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

π_k : prior probability of z_k =1, i.e prior probability of picking the *k*th component

Maximum Likelihood

- Data set of observations $\{\mathbf{x}1, \ldots, \mathbf{x}N\}$, and we wish to model this data using a mixture of Gaussians.

- Log of likelihood function:

Summation inside the logarithm → Complicated expressions for ML solution
→ Cannot get closed form solutions to ML parameters.

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Maximizing the log likelihood function

- Set d/dθ {LL(θ)} = 0 and solve for θ : Non-linear, non-analytically **solvable**
- Use gradient Descent: Doable, but often slow
- **Use EM**

Expectation maximization (EM)

- Expectation maximization (EM) is an iterative optimization approach that is used in many ways to find maximum likelihood estimates of parameters in probabilistic models in the presence of missing data.
- A generalization of MLE based on maximization of a posterior that data was generated by a model.
- EM is “simpler” than gradient methods
  - No need to choose step size.
- EM alternates between performing
  - an expectation (E) step, which computes an expectation of the likelihood by including the latent variables as if they were observed
  - a maximization (M) step, which computes the MLEs of the parameters by maximizing the expected likelihood found on the E step.
  - The parameters found on the M step are then used to begin another E step, and the process is repeated.

General EM algorithm

Observed data: D={x1, ……., xn}

Unknown variables: y

In clustering: y=1….K clusters

Parameters: θ

In GMM : θ= π : {π1, . . . , πk}

μ : {μ1 , . . . , μK}

Σ : {Σ1, . . . ΣK }

Goal: $\hat{\theta}_n = \arg\max_{\theta} \log P(D|\theta)$

In GMM : $\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$

EM algorithm for GMM

- Start with parameters describing each cluster. Initialize the means μk, covariances Σk and mixing coefficients πk, and evaluate the initial value of the log likelihood.
- E Step :
  - "partial membership" of each data point in each constituent distribution is computed
  - Compute “expected” classes (conditional probabilities of the latent variables) of all datapoints for each class
  - In K-means “E-step” we do hard assignment. EM does soft assignment

EM algorithm for GMM

E step:

Evaluate the responsibilities (posterior probabilities) using the current parameter

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

- For each example x_n,
- Compute $\gamma(z_{nk})$ the probability that x_n is generated by component Z_k i.e it belongs to cluster k
- If x_n is very likely under the k_{th} Gaussian, it gets high weight

EM algorithm for GMM

M-Step :

- maximize the expectation of the complete-data log-likelihood, computed with respect to the conditional probabilities found in the Expectation step. The result of the maximization is a new parameter vector μnew , Σnew and πnew
- Keep γ (znk) fixed, and apply MLE for maximizing of ln p(X|π,μ,Σ) for μk , Σk and πk, to get μnew , Σnew and πnew

$$\ln p(\mathbf{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Repeat E & M until convergence

  - In practice, the algorithm is deemed to have converged when the change in the log likelihood function, or alternatively in the parameters, falls below some threshold

EM algorithm for GMM

To Estimate:

M-Step

Initialize **π**, μ,Σ *and also evaluate the log likelihood*

E-Step

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)}$$

$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Perform E-Step Given $\gamma(z_k)$

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\text{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

Perform M-Step Given π, μ,Σ

$$p(z_k = 1) = \pi_k$$

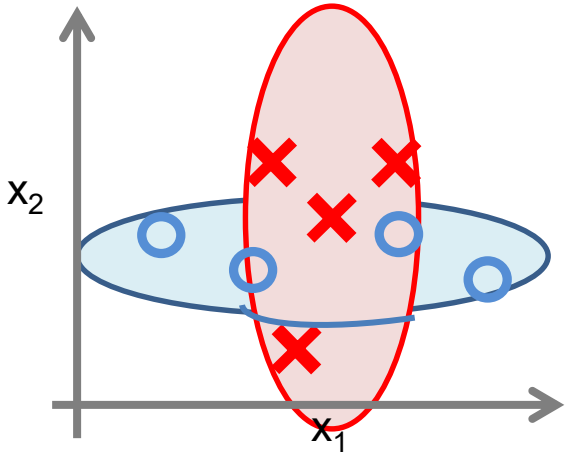
Repeat Until Convergence

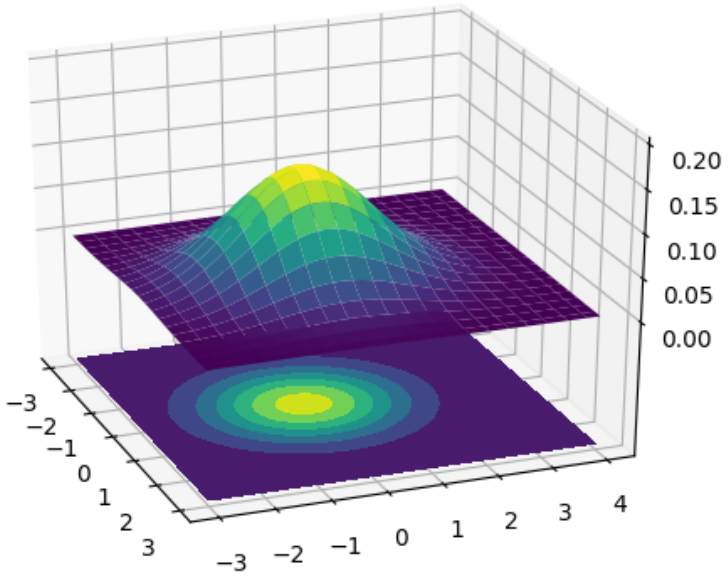
N_k = the effective number of points assigned to cluster k

Closing EM

- EM Algorithm breaks down the problem of estimating the ML parameters of a Mixture Model into E & M Steps and provides an iterative way of computing this
- EM Algorithm converges slowly
  - Use K-Means in the initialization
- EM Algorithm converges to local optimum
  - Sensitive to initialization
  - Each iteration increases the Log likelihood until convergence
  - Many restarts
- What is a good k?

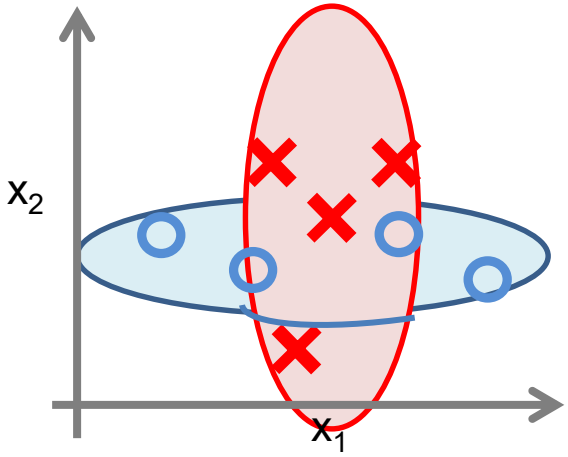
Gaussian Mixture Model

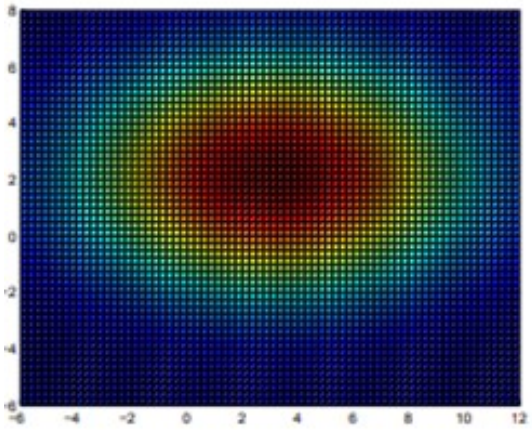


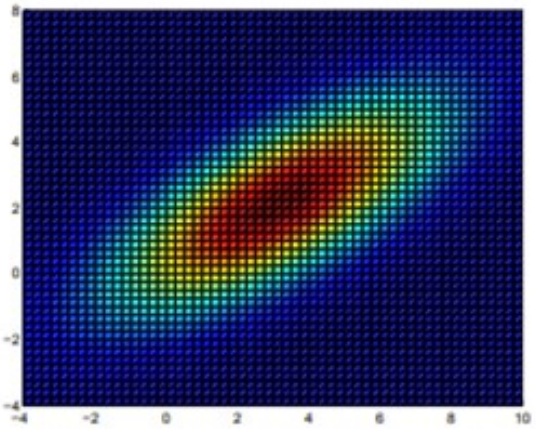


$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Gaussian Mixture Model







$$\frac{1}{\sqrt{2\pi}\sigma_1}\exp\left(-\frac{1}{2\sigma_1^2}(x_1-\mu_1)^2\right)\cdot\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left(-\frac{1}{2\sigma_2^2}(x_2-\mu_2)^2\right).$$

An n-dimensional Gaussian with mean μ ∈ Rn and diagonal covariance matrix Σ = diag(σ 2 1 , σ2 2 , . . . , σ2 n) is the same as a collection of n independent Gaussian random variables with mean μi and variance σ 2 i , respectively.

EM Algorithm

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

	Means	SDs	Weight
Cluster 1	(3,3)	(4,0.707)	0.5
Cluster 2	(4,4)	(0.5,0.707)	0.5

I Standardize the data if required:

II Fix the no.of.cluster expected

III Initialize the prototypes:
Mean, Covariance, Weights

IV Expectation-Step: Fix prototype & find the membership of each point weighted by the probability value

V. Calculate the log likelihood of the points

VI Maximization Step: Fix the membership(responsibility matrix) and re-estimate the prototypes

VII Calculate the new log likelihood of the points. Repeat E & M Step till convergence is achieved:

GMM

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

	Means	SDs	Weight
Cluster 1	(3,3)	(4,0.707)	0.5
Cluster 2	(4,4)	(0.5,0.707)	0.5

IV Expectation-Step: Fix prototype & find the membership of each point weighted by the probability value

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Sample calculation for this step is shown below:

Weight1*N1 =

$0.5 * P(Doc1, Cluter1)$

$= (Doc1, (3,3), (4,0.707))$

$$= 0.5 * \frac{e^{\frac{-(2-3)^2}{2*4^2}}}{4*\sqrt[2]{2*3.14}} * \frac{e^{\frac{-(2-3)^2}{2*0.707^2}}}{0.707*\sqrt[2]{2*3.14}}$$

$= 0.01004$

For Cluster 1 assumed parameters, here for Doc1 apply in the below formula and multiply with the weight = 0.5. For simplicity , assume off diagonal values in covariance matrix as 0 (This is usually done in text analysis use cases). similarly do it for all the Doc's

$$p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

GMM

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

	Means	SDs	Weight
Cluster 1	(3,3)	(4,0.707)	0.5
Cluster 2	(4,4)	(0.5,0.707)	0.5

IV Expectation-Step: Fix prototype & find the membership of each point weighted by the probability value

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Similarly:

For Cluster 2 assumed parameters, here for Doc1 apply in the below formula and multiply with the weight = 0.5. For simplicity , assume off diagonal values in covariance matrix as 0(This is usually done in text analysis use cases), similarly do it for all the Doc's

$$p(x;\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

GMM

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

	Means	SDs	Weight
Cluster 1	(3,3)	(4,0.707)	0.5
Cluster 2	(4,4)	(0.5,0.707)	0.5

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Per Docs sum the previous two found values.
This gives the denominator of this formula

For Doc2 :
Weight1*N1 = 0.0005
Weight2*N2 = 0.0828
Sum = 0.08329 = ~0.0833

$P(Z_{Doc1\text{-}Cluster1}) = 0.006$
$P(Z_{Doc1\text{-}Cluster1}) = 0.994$

Now find the membership of doc :
P(Zi1) = dividing Weight1*N1 by Sum
P(Zi2) = dividing Weight2*N1 by Sum

Similarly do it for all the Doc's

GMM

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

	Means	SDs	Weight
Cluster 1	(3,3)	(4,0.707)	0.5
Cluster 2	(4,4)	(0.5,0.707)	0.5

V. Calculate the log likelihood of the points

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Log likelihood = ln(sum of Doc1) + ln(sum of Doc2 + ln(Sum of Doc3)….

This value needs to be found for every iteration and at one iteration it will approach a very large value . That will be one of the stopping convergence point of the algorithm.

Log likehood = ln(0.010) + ln(0.083) + ln(0.006) =-12.16

GMM

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

	Means	SDs	Weight
Cluster 1	(3,3)	(4,0.707)	0.5
Cluster 2	(4,4)	(0.5,0.707)	0.5

VI Maximization Step: Fix the membership(responsibility matrix) and re-estimate the prototypes

	Doc1	Doc2	Doc3
P(Zi1)	0.9999	0.006	1
P(Zi2)	0.0001	0.994	0

	Means	SDs	Weight
Cluster 1			
Cluster 2			

Sample calculations shown below

$N_{k=1}$ = Cluster 1 =
= 0.9999+0.006+1
=2.0059
This is the value N1 represented here

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk})$$

N1 for cluster k=1 is got by summing these. Similarly do it for N2

For cluster k=1 is Means obtained by multiplying the data by the membership.

[0.9999(2) + 0.006(4) + 1(7)]/N1 , [0.9999(2) + 0.006(5) + 1(2)]/N1

$\text{MEAN}_{k=1}$ =
(4.49, 2.00) = ~(4.49,2)

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n$$

Similarly do it for cluster 2

$\text{Std.DeV}_{k=1}$ =

([0.9999(2−4.49)² + *0.006*(4−4.49)² + *1*(7−4.49)²]/*N1* , [0.9999(2−2)² + *0.006*(5−2)² + *1*(2−2)²]/*N1*)

= (6.23, 0.03)

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}) (\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}})^{\mathrm{T}}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N}$$

$\text{Weight}_{k=\text{cluster1 (new)}}$
= 2.0059/3 = 0.67

GMM - EM Algorithm

2nd Iteration continues

	Word1	Word2
Doc1	2	2
Doc2	4	5
Doc3	7	2

I Standardize the data if required:

II Fix the no.of.cluster expected

III Initialize the prototypes: Mean, Covariance, Weights

IV Expectation-Step: Fix prototype & find the membership of each point weighted by the probability value

V. Calculate the log likelihood of the points

VI Maximization Step: Fix the membership(responsibility matrix) and re-estimate the prototypes

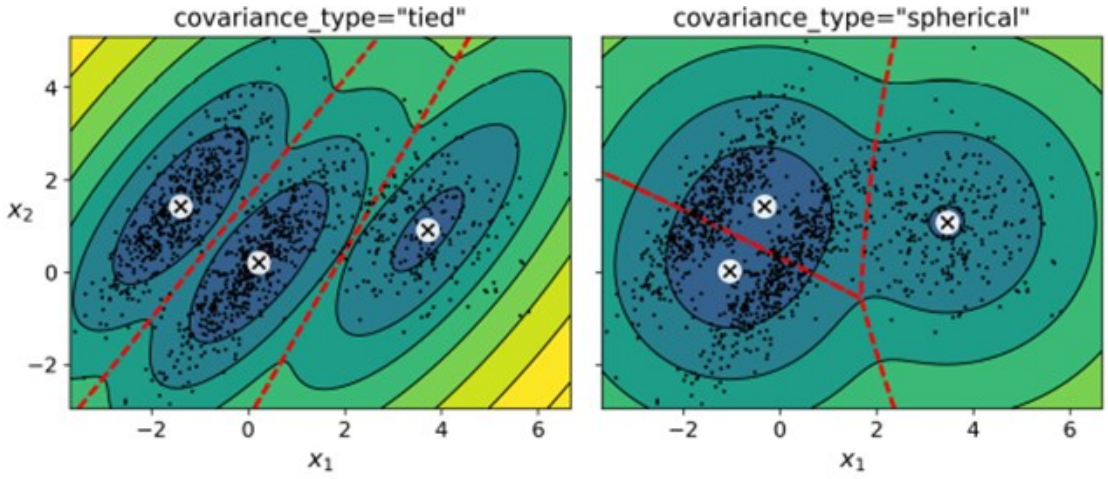
VII Calculate the new log likelihood of the points. Repeat E & M Step till convergence is achieved:

$$\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}.$$

	Means	SDs	Weight
Cluster 1	(4.49,2)	(6.32,0.03)	0.67
Cluster 2	(3.99,4.99)	(0.001,0.001)	0.33

	Doc1	Doc2	Doc3
Weight1 * N1			
Weight2 * N2			
Sum			
P(Zi1)			
P(Zi2)			

Gaussian Mixture Model



Note:
The GMM model can be relaxed by setting these parameters while calling the function

- "**spherical**": all clusters must be spherical, but they can have different diameters (i.e., different variances).
- "**diag**": clusters can take on any ellipsoidal shape of any size, but the ellipsoid's axes must be parallel to the coordinate axes (i.e., the covariance matrices must be diagonal).
- "**tied**": all clusters must have the same ellipsoidal shape, size and orientation (i.e., all clusters share the same covariance matrix).

Example – 2 for Practice – One Dimensional Data

A simple case:

- We have unlabeled data x1 , x2 , …, xn
- We know there are K classes
- We know P(z=1)=π1 , P(z=1)=π2 ,……., P(z=1)=πk
- We know common variance σ2
- We don't know μ1 , μ2 ,……, μk and we want to learn them

Example – 2 for Practice – One Dimensional Data

Let $(x_1, x_2, x_3) = (2, 4, 7)$ be our three datapoints, presumed to have each been generated from one of two Gaussians.

The stdev of both Gaussians are given: $\sigma_1 = \sigma_2 = 1/\sqrt{2}$.
The prior over the two Gaussians in also given: $\lambda_1 = \lambda_2 = 0.5$
Let j be an index over the Gaussians, i an index over the data points, and k an index over the E-M iterations.

We are trying to derive values for the two means $\boldsymbol{\mu} = (\mu_1, \mu_2)$ that maximize the likelihood of the given data, i.e.:

$$\boldsymbol{\mu}_{ML} = \arg\max_{\mu_1, \mu_2} \prod_i \sum_j \lambda_j e^{-\frac{(x_i - \mu_j)^2}{2\sigma^2}}$$

Let us initialize the Gaussian means to some reasonable values (inside the data range, and integer valued, to make calculation easy): $\mu_1^{[0]} = 3, \mu_2^{[0]} = 6$.

Let $z_{i,j} = 1$ if x_i was generated by Gaussian j, and 0 otherwise. The $z_{i,j}$'s are our latent variables.

The E-Step is:

$$E[z_{i,j}|\mu^{[k]}] = \frac{\lambda_j L(x_i|\mu = \mu_j^{[k]})}{\sum_{j'} \lambda_{j'} L(x_i|\mu = \mu_{j'}^{[k]})}$$

and the M-step is:

$$\mu_j^{[k+1]} = \frac{\sum_i E[z_{i,j}|\mu^{[k]}] \cdot x_i}{\sum_i E[z_{i,j}|\mu^{[k]}]}$$

Example – 2 for Practice – One Dimensional Data

Let $(x_1, x_2, x_3) = (2, 4, 7)$ be our three datapoints, presumed to have each been generated from one of two Gaussians.

The stdev of both Gaussians are given: $\sigma_1 = \sigma_2 = 1/\sqrt{2}$.
The prior over the two Gaussians in also given: $\lambda_1 = \lambda_2 = 0.5$
Let j be an index over the Gaussians, i an index over the data points, and k an index over the E-M iterations.

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

Let us initialize the Gaussian means to some reasonable values (inside the data range, and integer valued, to make calculation easy): $\mu_1^{[0]} = 3, \mu_2^{[0]} = 6$.

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

Example – 2 for Practice – One Dimensional Data

i	1	2	3
x_i	2.0	4.0	7.0
$\mathcal{N}$(xi|μ1)	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-1}$	$\frac{1}{\sqrt{\pi}}e^{-16}$
$\mathcal{N}$(xi|μ2)	$\frac{1}{\sqrt{\pi}}e^{-16}$	$\frac{1}{\sqrt{\pi}}e^{-4}$	$\frac{1}{\sqrt{\pi}}e^{-1}$
$\mathcal{N}$(xi|μ1)+$\mathcal{N}$(xi|μ2)	$\frac{1}{2\sqrt{\pi}}(e^{-1}+e^{-16})$	$\frac{1}{2\sqrt{\pi}}(e^{-1}+e^{-4})$	$\frac{1}{2\sqrt{\pi}}(e^{-16}+e^{-1})$
γ(zi,1)	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$	$\frac{e^{-1}}{e^{-1}+e^{-4}} \approx 0.953$	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$
γ(zi,2)	$\frac{e^{-16}}{e^{-1}+e^{-16}} \approx 0$	$\frac{e^{-4}}{e^{-1}+e^{-4}} \approx 0.047$	$\frac{e^{-1}}{e^{-1}+e^{-16}} \approx 1$

And therefore:

$$\mu_1^{[1]} \approx \frac{1*2.0+0.953*4.0+0*7.0}{1+0.953} \approx 2.976$$

and

$$\mu_2^{[1]} \approx \frac{0*2.0+0.047*4.0+1*7.0}{1+0.047} \approx 6.865$$

Gaussian Mixture Model

```
from sklearn.mixture import GaussianMixture

gm = GaussianMixture(n_components=3, n_init=10)
gm.fit(X)

gm.weights_
gm.means_
gm.covariances_
```

```
>>> gm.predict(X)
array([2, 2, 1, ..., 0, 0, 0])
>>> gm.predict_proba(X)
array([[2.32389467e-02, 6.77397850e-07, 9.76760376e-01],
       [1.64685609e-02, 6.75361303e-04, 9.82856078e-01],
```

References

- Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition

- https://www.youtube.com/watch?v=TG6Bh-NFhA0

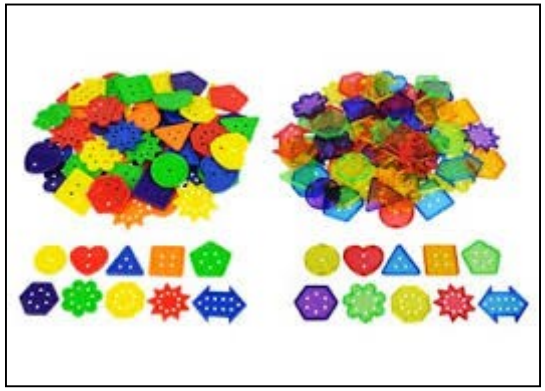
- https://www.youtube.com/watch?v=qMTuMa86NzU

Additional Reference

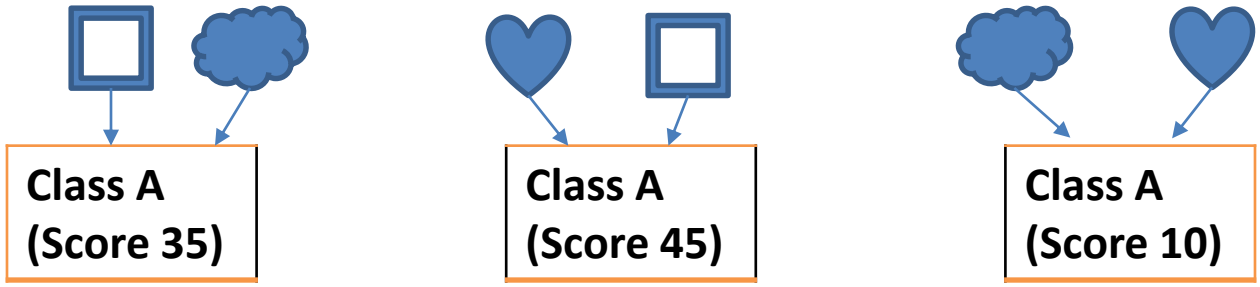
Notion of Similarity & Dissimilarity
Proximity Measures

Need for Measures of Similarities & Dissimilarities

Feedback = - 2 Suggestion = Place in Label Color XXX	No Feedback
Supervised	Unsupervised



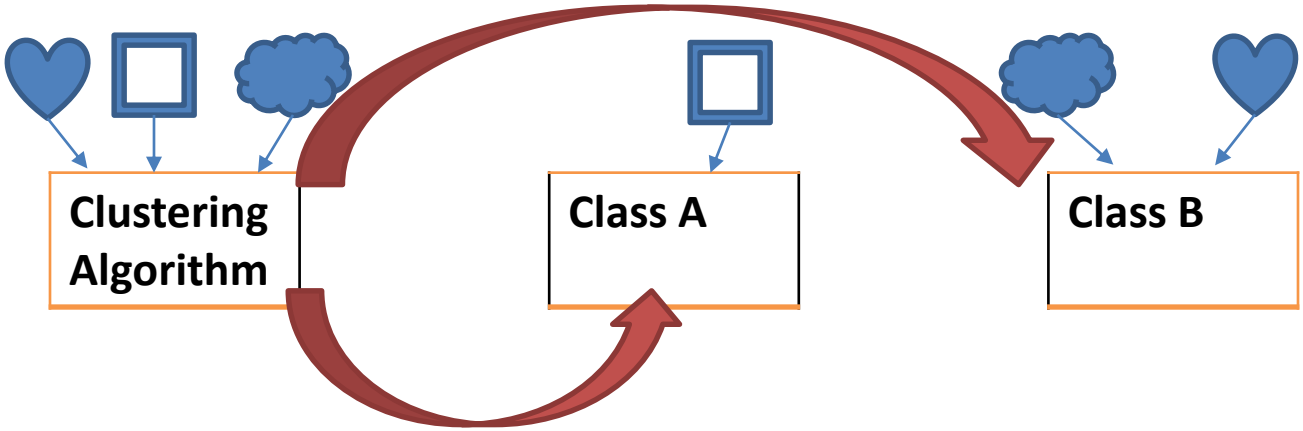
Summary of SUPERVISED LEARNING



1. Divide the data / record into training set and test set
2. Derive a model for the class attribute as a function of other important variables from the training set
3. Pass the test set and get the class value to Validate the accuracy

GOAL : Previously unseen records should be assigned a class as accurately as possible.

Idea : Unsupervised Learning



1. Get all or representative sample of the data sets
2. Filter the attributes of importance
3. Find one similarity measure
4. Similar data are grouped under same cluster

GOAL : Intra cluster distances are minimized and inter cluster distances are maximized

innovate achieve lead

BITS Pilani

Pilani Campus

Distance Measures for Unsupervised Learning

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	Male	5	Yes	Platinum	7.5	Yes	Upper	Bengaluru
Jill	Female	2	Yes	Gold	8.2	No	Middle	Mumbai
John	Male	9	No	Gold	7	Yes	Lower	Bengaluru
Mary	Male	6	No	Gold	6.0	No	Lower	Bengaluru

Measures of Similarity & Dissimilarity

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	Male	5	Yes	Platinum	7.5	Yes	Upper	Bengaluru
Jill	Female	2	Yes	Gold	8.2	No	Middle	Mumbai
John	Male	9	No	Gold	7	Yes	Lower	Bengaluru
Mary	Male	6	No	Gold	6.0	No	Lower	Bengaluru

1. Understand the distance type required
2. Normalize where necessary
3. Assign weightage based on attribute importance

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}}$$

Numerical Features

- Minkowski Distance

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

- h = 1: Manhattan (city block, L_1 norm) distance
  - E.g., the Hamming distance: the number of bits that are different between two binary vectors
- h = 2: (L_2 norm) Euclidean distance
- $h \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component (attribute) of the vectors

Measures of Similarity & Dissimilarity

Minkowski distance:

$$\sqrt[h]{(w_1|x_if_1 - xjf_1|^h + w_2|x_if_2 - xjf_2|^h + w_3|xif}$$

...

d(Jack,Mary) : Numeric Attribute: Lets use Euclidean distance by applying h=2 and equal weightage to all attributes thus all the w values are equal to 1.

	Jack	Jill	John	Mary
Jack	0			
Jill		0		
John			0	
Mary				0

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	Male	5	Yes	Platinum	7.5	Yes	Upper	Bengaluru
Jill	Female	2	Yes	Gold	8.2	No	Middle	Mumbai
John	Male	9	No	Gold	7	Yes	Lower	Bengaluru
Mary	Male	6	No	Gold	6.0	No	Lower	Bengaluru

Measures of Similarity & Dissimilarity

Minkowski distance:

$$\sqrt[h]{(w_1|x_if_1 - xjf_1|^h + w_2|x_if_2 - xjf_2|^h + w_3|xif}$$

...

d(Jack,Mary) : Numeric Attribute: Lets use Euclidean distance by applying h=2 and equal weightage to all attributes thus all the w values are equal to 1.

	Jack	Jill	John	Mary
Jack	0			**1.803**
Jill		0		
John			0	
Mary				0

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	Male	5	Yes	Platinum	7.5	Yes	Upper	Bengaluru
Jill	Female	2	Yes	Gold	8.2	No	Middle	Mumbai
John	Male	9	No	Gold	7	Yes	Lower	Bengaluru
Mary	Male	6	No	Gold	6.0	No	Lower	Bengaluru

Binary Features

- A contingency table for binary data

	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

- Distance measure for symmetric binary variables:

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

- Distance measure for asymmetric binary variables(Jaccard's Distance):

$$d(i,j) = \frac{r+s}{q+r+s}$$

- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

Measures of Similarity & Dissimilarity

d(Jack,Mary) : Symmetric Attribute:

	Jack	Jill	John	Mary
Jack	0			**1.803+0.5**
Jill		0		
John			0	
Mary				0

	1	0	Sum
1	q	r	q+r
0	s	t	s+t
sum	q+s	r+t	q+r+s+t

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	0	5	Yes	Platinum	7.5	1	Upper	Bengaluru
Jill	1	2	Yes	Gold	8.2	0	Middle	Mumbai
John	0	9	No	Gold	7	1	Lower	Bengaluru
Mary	0	6	No	Gold	6.0	0	Lower	Bengaluru

Binary Features

- A contingency table for binary data

	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

- Distance measure for symmetric binary variables:

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

- Distance measure for asymmetric binary variables(Jaccard's Distance):

$$d(i,j) = \frac{r+s}{q+r+s}$$

- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

Measures of Similarity & Dissimilarity

d(Jack,Mary) : Asymmetric Attribute:

	Jack	Jill	John	Mary
Jack	0			**1.803+0.5**
Jill		0		
John			0	
Mary				0

	1	0	Sum
1	q	r	q+r
0	s	t	s+t
sum	q+s	r+t	q+r+s+t

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	0	5	1	Platinum	7.5	1	Upper	Bengaluru
Jill	1	2	1	Gold	8.2	0	Middle	Mumbai
John	0	9	0	Gold	7	1	Lower	Bengaluru
Mary	0	6	0	Gold	6.0	0	Lower	Bengaluru

Ordinal Features

- For every Object put a rank per attribute

$$r_{if} \in \{1, \ldots, M_f\}$$

- Map the rank onto [0, 1] range by following transformation

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

- Compute distance between any two objects i, j using above values as interval type numerical value. The distance can be any of the numeric distance measure like Manhattan or Euclidean. Consider the set of all Z_{if} values as numeric

Measures of Similarity & Dissimilarity

- Define a rank
- Assign feature value map
- Normalize rank to rane [0,1] $Zif = \frac{r_{if}-1}{M_{if}-1}$
- Mij= Maximum rank of the attribute
- rij =rank of the value

$$Z_{(Jack,\ CardType)} = \frac{r_{if}-1}{M_{if}-1} = \frac{2-1}{3-1} = 0.5$$

d(Jack,Mary) :

	Jack	Jill	John	Mary
Jack	0			1.803+0.5 +1
Jill		0		
John			0	
Mary				0

Here Manhattan distance was used to show the calculations using L1 norm. Its recommended to use only L1 norm or only L2 norm consistently for a single data set wherever appropriate

CardType	Rank Values
Titanium	3
Platinum	2
Gold	1

Income Level	Rank Values
Upper	3
Middle	2
Lower	1

	isMultipleAccount Holder	Income Level	Region
	1	Upper	Bengaluru
	0	Middle	Mumbai
	1	Lower	Bengaluru
	0	Lower	Bengaluru

Mary	0	6	0	Gold	6.0

Measures of Similarity & Dissimilarity

d(Jack,Mary) :

	Jack	Jill	John	Mary
Jack	0			**1.803+0.5 +1+1.5**
Jill		0		
John			0	
Mary				0

CardType	Rank Values
Titanium	3
Platinum	2
Gold	1

Income Level	Rank Values
Upper	3
Middle	2
Lower	1

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	0	5	1	2 → 0.5	7.5	1	3→1	Bengaluru
Jill	1	2	1	1 →0	8.2	0	2→0.5	Mumbai
John	0	9	0	1→0	7	1	1→0	Bengaluru
Mary	0	6	0	1→0	6.0	0	1→0	Bengaluru

Nominal Features

Takes 2 or more states, e.g., red, yellow, blue, green (generalization of a binary attribute)

Method 1: Simple matching

- *m*: # of matches, *p*: total # of variables

$$d(i,j) = \frac{p - m}{p}$$

Method 2: Use a large number of binary attributes

- creating a new binary attribute for each of the *M* nominal states

Measures of Similarity & Dissimilarity

d(Jack,Mary) :

	Jack	Jill	John	Mary
Jack	0			**1.803+0.5 +1+1.5+0**
Jill		0		
John			0	
Mary				0

Aggregated distances is recommended to be scaled between 0-1 . Check the next slide for sample

	Gender	ServiceRating	IsPriority Customer	CardType	CreditScore	isMultipleAccount Holder	Income Level	Region
Jack	0	5	1	0.5	7.5	1	1	Bengaluru
Jill	1	2	1	0	8.2	0	0.5	Mumbai
John	0	9	0	0	7	1	0	Bengaluru
Mary	0	6	0	0	6.0	0	0	Bengaluru

Measures of Dis/similarity

Distances : Attributes of Mixed Type – Gower's Distance

- A database may contain all attribute types
  - Nominal, symmetric binary, asymmetric binary, numeric, ordinal

You can use Gower's distances if explicitly specified in the question. Else students are advised to use the distance measure (p-m)/p for categorical attributes' calculation and treat the encoded ordinal attribute as numerical attribute to apply the same Euclidean distance/ Manhattan distance.

$$d(i,j) = \frac{\sum_{c=1}^{n} \omega_c \delta_{ij}^{(c)} d_{ij}^{(c)}}{\sum_{c=1}^{n} \omega_c \delta_{ij}^{(c)}}$$

$d(i,j)$ = dissimilarity between row i and row j

c = the cth column

n = number of columns in the dataset

ω_c = weight of cth column $= \frac{1}{\text{nrows in dataset}}$

$$\delta_{ij}^{c} = \begin{cases} 0 & \text{if column c is missing in row i or j} \\ 0 & \text{if column c is asymmetric binary and both values in row i and j are 0} \\ 1 & \text{otherwise} \end{cases}$$

$$d_{ij}^{c}(\text{categorical}) = \begin{cases} 0 & \text{if i and j are equal in column c} \\ 1 & \text{otherwise} \end{cases}$$

$$d_{ij}^{c}(\text{continuous/ordinal}) = \frac{|\text{row i in column c - row j in column c}|}{\max(\text{column c}) - \min(\text{column c})}$$

Statistical Comparison – Document Data

Document	*team*	*coach*	*hockey*	*baseball*	*soccer*	*penalty*	*score*	*win*	*loss*	*season*
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

Measure of Similarity : Cosine
$\cos(d_1, d_2) = (d_1 \bullet d_2) / \|\|d_1\|\| \, \|\|d_2\|\|$,

where • indicates vector dot product, $\|\|d\|$: the length of vector d

Statistical Comparison – Document Data

Document	*team*	*coach*	*hockey*	*baseball*	*soccer*	*penalty*	*score*	*win*	*loss*	*season*
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

Measure of Similarity : Cosine → $\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|$,
where • indicates vector dot product, $\|d\|$: the length of vector d

$\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|$

$d_1 \bullet d_2 = (5*3)+(0*0)+(3*2)+(0*0)+(2*1)+(0*1)+(0*0)+(2*1)+(0*0)+(0*1)$

$\|d_1\| = 2\sqrt{(5^2 + 0^2 + 3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2)}$

$\|d_2\| = 2\sqrt{(3^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2)}$

$\cos(d_1, d_2) = \mathbf{0.94}$

The more the value tends →1, the more similar are the documents.

K Means

```
from sklearn.cluster import KMeans
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
```

y_pred

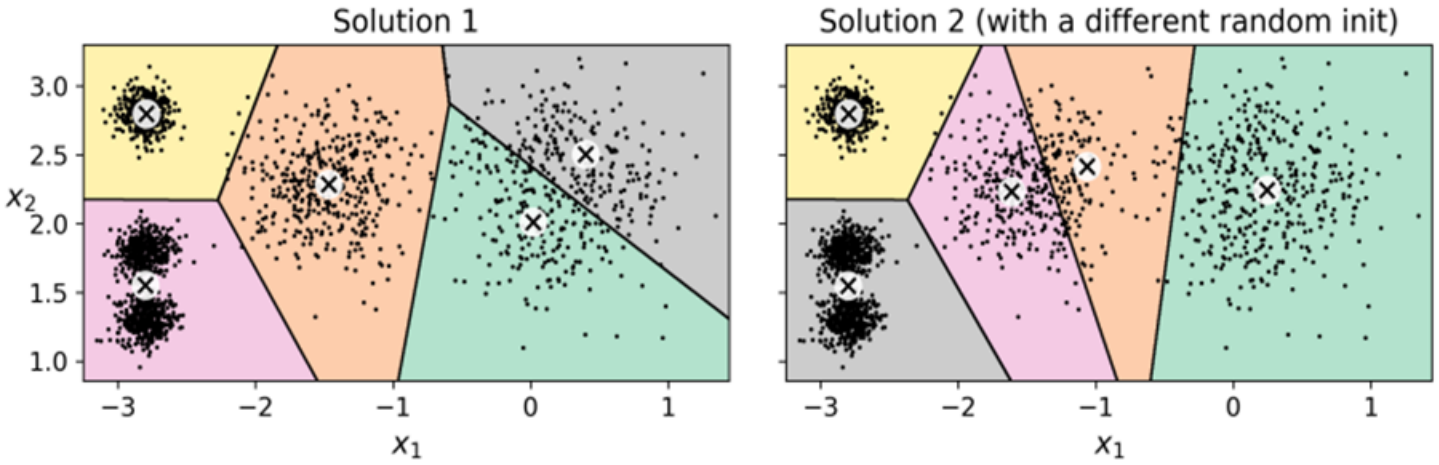
array([4, 0, 1, ..., 2, 1, 0], dtype=int32)

kmeans.cluster_centers_

array([[-2.80389616, 1.80117999], [0.20876306, 2.25551336], [-2.79290307, 2.79641063], [-1.46679593, 2.28585348], [-2.80037642, 1.30082566]])

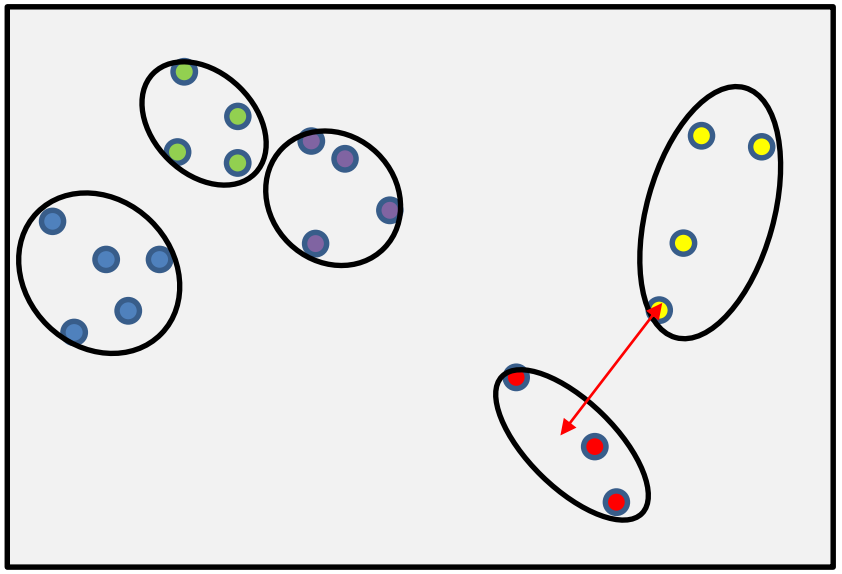
```
X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]]
kmeans.predict(X_new)

array([1, 1, 2, 2], dtype=int32)
```

K Means – Explicit Initialization of the centers




```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1)
```

K Means – Evaluation of Cluster Quality



$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i) \end{cases}$$

```
from sklearn.metrics import silhouette_score

silhouette_score(X, kmeans.labels_)

0.655517642572828
```

K Means – Evaluation of Cluster Quality

Metric: Silhouette Coefficient

$$S(p) = \frac{b(p)-a(p)}{\max(b(p),a(p))}$$

$$a(p) = \frac{\sum_{o` \in Ci,\, p \neq o`} dist(p,o`)}{||Ci||-1}$$

$$b(p) = \min_{Cj:0<j<=k,\, j \neq i} \frac{\sum_{o` \in Cj,\, p \neq o`} dist(p,o`)}{||Cj||}$$

a(p) reflects the compactness of the cluster to which an object belongs. LTB
It's the average distance between p and all other objects in the cluster to which p belongs
b(p) captures the degree to which an object is separated from other clusters. HTB
It's the minimum average distance from p and all clusters to which p does not belong

Interpretation:
The value of the silhouette coefficient is between −1 and 1
If S→1, Clusters are compact and well separated