ML UNIT 4 PART A Material

machine learning (Jawaharlal Nehru Technological University, Kakinada)

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

MACHINE LEARNING UNIT 4 (PART A)

SYLLABUS:

Unsupervised Learning Techniques:

Clustering, K-Means, Limits of K-Means, Using Clustering for Image Segmentation, Using Clustering for Preprocessing, Using Clustering for Semi-Supervised Learning, DBSCAN, Gaussian Mixtures.

Clustering:

Clustering analysis or simply Clustering is basically an unsupervised learning method that divides the data points into a number of specific batches or groups, such that the data points in the same groups have similar properties and data points in different groups have different properties in some sense.

E.g. K-Means (distance between points), Affinity propagation (graph distance), Mean-shift (distance between points), DBSCAN (distance between nearest points), Gaussian mixtures (Mahalanobis distance to centers), Spectral clustering (graph distance) etc.
Fundamentally, all clustering methods use the same approach i.e. first we calculate similarities and then we use it to cluster the data points into groups or batches.

K-Means:

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

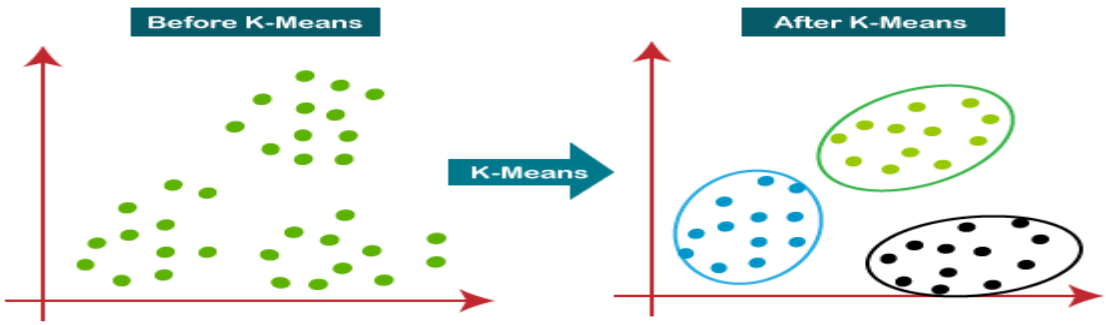
- Determines the best value for K center points or centroids by an iterative process.

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

- Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:



The working of the K-Means algorithm is explained in the below steps:

Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

Step-5: Repeat the third step, which means reassign each data point to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Choose the value of "K number of clusters" in K-means Clustering:

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. WCSS stands for Within Cluster

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$WCSS= \sum_{Pi \text{ in Cluster1}} \text{distance } P_i\ C_1)^2 + \sum_{Pi \text{ in Cluster2}} \text{distance}(P_i\ C_2)^2 + \sum_{Pi \text{ in CLuster3}} \text{distance}(P_i\ C_3)^2$$

In the above formula of WCSS,

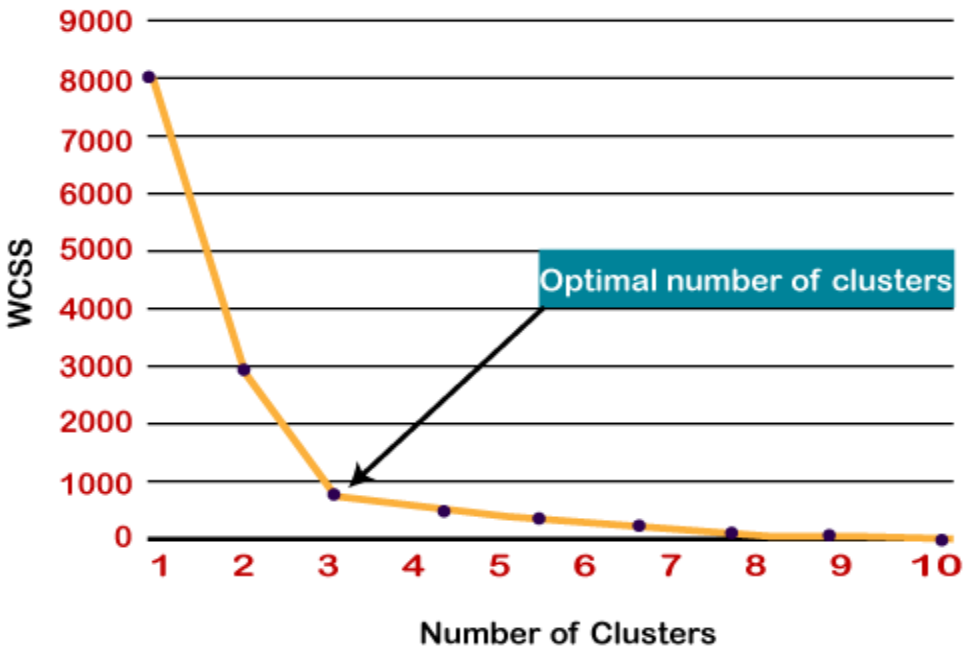
$\sum_{Pi \text{ in Cluster1}}$ distance $(P_i\ C_1)^2$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

- It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
- For each value of K, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters K.
- The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

Example:

Cluster the following eight points (with (x, y) representing locations) into three clusters: A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Initial cluster centers are: A1(2, 10), A4(5, 8) and A7(1, 2).
The distance function between two points a = (x1, y1) and b = (x2, y2) is defined as-

$$P(a, b) = |x2 – x1| + |y2 – y1|$$

Iteration-01:

- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.

The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-

Calculating Distance Between A1(2, 10) and C1(2, 10)-

P(A1, C1)
= |x2 – x1| + |y2 – y1|
= |2 – 2| + |10 – 10|
= 0

Calculating Distance Between A1(2, 10) and C2(5, 8)-

P(A1, C2)
= |x2 – x1| + |y2 – y1|
= |5 – 2| + |8 – 10|
= 3 + 2
= 5

Calculating Distance Between A1(2, 10) and C3(1, 2)-

P(A1, C3)
= |x2 – x1| + |y2 – y1|
= |1 – 2| + |2 – 10|
= 1 + 8
= 9

In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Next,

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

- We draw a table showing all the results.
- Using the table, we decide which point belongs to which cluster.
- The given point belongs to that cluster whose center is nearest to it.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (5, 8) of Cluster-02	Distance from center (1, 2) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	5	9	C1
A2(2, 5)	5	6	4	C3
A3(8, 4)	12	7	9	C2
A4(5, 8)	5	0	10	C2
A5(7, 5)	10	5	9	C2
A6(6, 4)	10	5	7	C2
A7(1, 2)	9	10	0	C3
A8(4, 9)	3	2	10	C2

Cluster-01:

First cluster contains points-

- A1(2, 10)

Cluster-02:

Second cluster contains points-

- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)
- A8(4, 9)

Cluster-03:

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

Third cluster contains points-

- A2(2, 5)
- A7(1, 2)

Now,

- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.

For Cluster-01:

- We have only one point A1(2, 10) in Cluster-01.
- So, cluster center remains the same.

For Cluster-02:

Center of Cluster-02
= ((8 + 5 + 7 + 6 + 4)/5, (4 + 8 + 5 + 4 + 9)/5)
= (6, 6)

For Cluster-03:

Center of Cluster-03
= ((2 + 1)/2, (5 + 2)/2)
= (1.5, 3.5)

This is completion of Iteration-01.

Iteration-02:

- We calculate the distance of each point from each of the center of the three clusters.
- The distance is calculated by using the given distance function.

The following illustration shows the calculation of distance between point A1(2, 10) and each of the center of the three clusters-

Calculating Distance Between A1(2, 10) and C1(2, 10)-

P(A1, C1)
= |x2 – x1| + |y2 – y1|
= |2 – 2| + |10 – 10|
= 0

Calculating Distance Between A1(2, 10) and C2(6, 6)-

P(A1, C2)
= |x2 – x1| + |y2 – y1|
= |6 – 2| + |6 – 10|

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

= 4 + 4
= 8

Calculating Distance Between A1(2, 10) and C3(1.5, 3.5)-

P(A1, C3)
= |x2 – x1| + |y2 – y1|
= |1.5 – 2| + |3.5 – 10|
= 0.5 + 6.5
= 7

In the similar manner, we calculate the distance of other points from each of the center of the three clusters.

Next,

- We draw a table showing all the results.
- Using the table, we decide which point belongs to which cluster.
- The given point belongs to that cluster whose center is nearest to it.

Given Points	Distance from center (2, 10) of Cluster-01	Distance from center (6, 6) of Cluster-02	Distance from center (1.5, 3.5) of Cluster-03	Point belongs to Cluster
A1(2, 10)	0	8	7	C1
A2(2, 5)	5	5	2	C3
A3(8, 4)	12	4	7	C2
A4(5, 8)	5	3	8	C2
A5(7, 5)	10	2	7	C2
A6(6, 4)	10	2	5	C2
A7(1, 2)	9	9	2	C3
A8(4, 9)	3	5	8	C1

From here, New clusters are-

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

Cluster-01:

First cluster contains points-

- A1(2, 10)
- A8(4, 9)

Cluster-02:

Second cluster contains points-

- A3(8, 4)
- A4(5, 8)
- A5(7, 5)
- A6(6, 4)

Cluster-03:

Third cluster contains points-

- A2(2, 5)
- A7(1, 2)

Now,

- We re-compute the new cluster clusters.
- The new cluster center is computed by taking mean of all the points contained in that cluster.

For Cluster-01:

Center of Cluster-01
= ((2 + 4)/2, (10 + 9)/2)
= (3, 9.5)

For Cluster-02:

Center of Cluster-02
= ((8 + 5 + 7 + 6)/4, (4 + 8 + 5 + 4)/4)
= (6.5, 5.25)

For Cluster-03:

Center of Cluster-03
= ((2 + 1)/2, (5 + 2)/2)
= (1.5, 3.5)

This is completion of Iteration-02.

After second iteration, the center of the three clusters are-

- C1(3, 9.5)
- C2(6.5, 5.25)

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

- C3(1.5, 3.5)

Limits of K-Means:

- It requires specifying the number of clusters (k) in advance.
- It cannot handle noisy data and outliers.
- It is not suitable to identify clusters with non-convex shapes.
- K-Means does not behave very well when the clusters have varying sizes, different densities, or non-spherical shapes.

Using Clustering for Image Segmentation:

Image segmentation is the task of partitioning an image into multiple segments. In semantic segmentation, all pixels that are part of the same object type get assigned to the same segment. For example, in a self-driving car's vision system, all pixels that are part of a pedestrian's image might be assigned to the "pedestrian" segment (there would just be one segment containing all the pedestrians). In instance segmentation, all pixels that are part of the same individual object are assigned to the same segment. In this case there would be a different segment for each pedestrian. The state of the art in semantic or instance segmentation today is achieved using complex architectures based on convolutional neural networks (see Chapter 14). Here, we are going to do something much simpler: color segmentation. We will simply assign pixels to the same segment if they have a similar color. In some applications, this may be sufficient, for example if you want to analyze satellite images to measure how much total forest area there is in a region, color segmentation may be just fine.
First, let's load the image using Matplotlib's imread() function:

```
>>> from matplotlib.image import imread # you could also use `imageio.imread()`
>>> image = imread(os.path.join("images","clustering","ladybug.png"))
>>> image.shape
(533, 800, 3)
```

The image is represented as a 3D array: the first dimension's size is the height, the second is the width, and the third is the number of color channels, in this case red, green and blue (RGB). In other words, for each pixel there is a 3D vector containing the intensities of red, green and blue, each between 0.0 and 1.0 (or between 0 and 255 if you use imageio.imread()). Some images may have less channels, such as grayscale images (one channel), or more channels, such as images with an additional alpha channel for transparency, or satellite images which often contain channels for many light frequencies (e.g., infrared).

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

Using Clustering for Preprocessing:

Clustering can be an efficient approach to dimensionality reduction, in particular as a preprocessing step before a supervised learning algorithm. For example, let's tackle the digits dataset which is a simple MNIST-like dataset containing 1,797 grayscale 8×8 images representing digits 0 to 9. First, let's load the dataset:

```
from sklearn.datasets import load_digits
X_digits, y_digits = load_digits(return_X_y=True)
```

Now, let's split it into a training set and a test set:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_digits, y_digits)
```

Next, let's fit a Logistic Regression model:

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

Let's evaluate its accuracy on the test set:

```
>>> log_reg.score(X_test, y_test)
0.9666666666666667
```

Okay, that's our baseline: 96.7% accuracy. Let's see if we can do better by using K-Means as a preprocessing step. We will create a pipeline that will first cluster the training set into 50 clusters and replace the images with their distances to these 50 clusters, then apply a logistic regression model.

Using Clustering for Semi-Supervised Learning:

Another use case for clustering is in semi-supervised learning, when we have plenty of unlabeled instances and very few labeled instances. Let's train a logistic regression model on a sample of 50 labeled instances from the digits dataset:

```
n_labeled = 50
log_reg = LogisticRegression()
log_reg.fit(X_train[:n_labeled], y_train[:n_labeled])
```

What is the performance of this model on the test set?

```
>>> log_reg.score(X_test, y_test)
0.8266666666666667
```

The accuracy is just 82.7%: it should come as no surprise that this is much lower than earlier, when we trained the model on the full training set. Let's see how we can do better. First, let's cluster the training set into 50 clusters, then for each cluster let's find the image closest to the centroid. We will call these images the representative images:

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

```
k = 50
kmeans = KMeans(n_clusters=k)
X_digits_dist = kmeans.fit_transform(X_train)
representative_digit_idx = np.argmin(X_digits_dist, axis=0)
X_representative_digits = X_train[representative_digit_idx]
```

Figure 1 shows these 50 representative images:

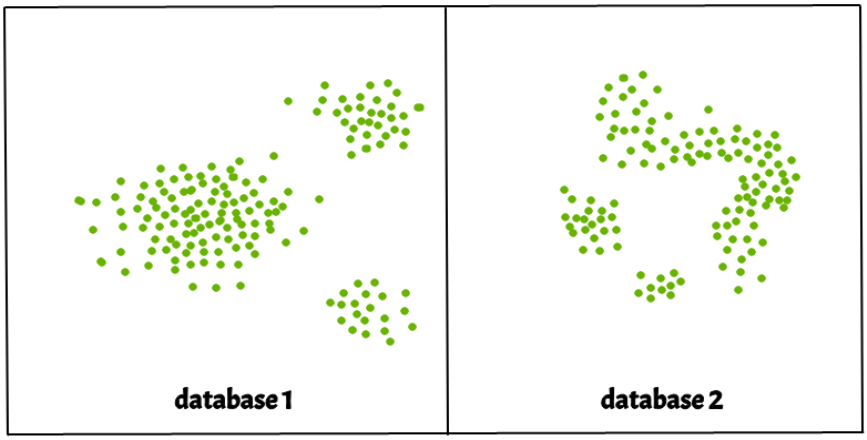
Figure 1. Fifty representative digit images (one per cluster)

DBSCAN:

Density-based spatial clustering of applications with noise (DBSCAN) clustering method.

Clusters are dense regions in the data space, separated by regions of the lower density of points. The **DBSCAN algorithm** is based on this intuitive notion of "clusters" and "noise". The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

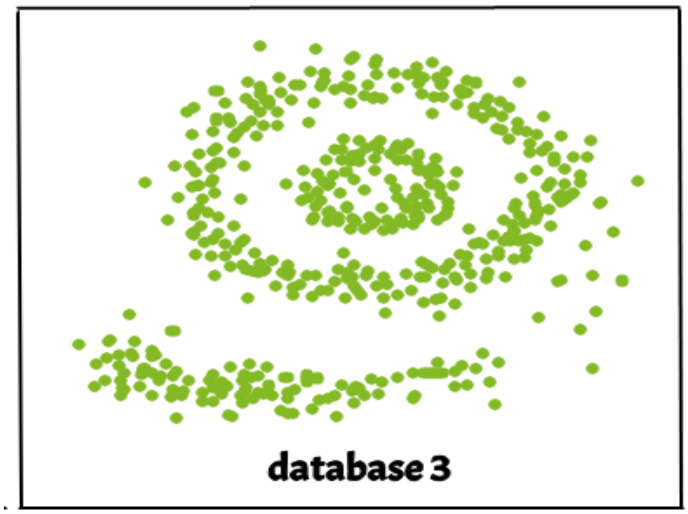
Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters. In other words, they are suitable only for compact and well-separated clusters. Moreover, they are also severely affected by the presence of noise and outliers in the data.



Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

Real life data may contain irregularities, like:

1. Clusters can be of arbitrary shape such as those shown in the figure below.
2. Data may contain noise.



The figure below shows a data set containing nonconvex clusters and outliers/noises. Given such data, k-means algorithm has difficulties in identifying these clusters with arbitrary shapes.

DBSCAN algorithm requires two parameters:

1. **eps** : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered neighbors. If the eps value is chosen too small then large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and the majority of the data points will be in the same clusters. One way to find the eps value is based on the **k-distance graph**.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

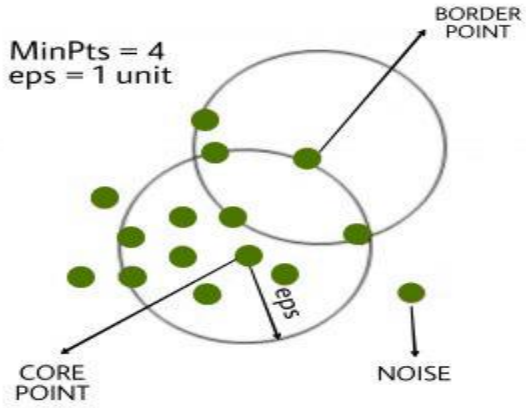
In this algorithm, we have 3 types of data points.

Core Point: A point is a core point if it has more than MinPts points within eps.

Border Point: A point which has fewer than MinPts within eps but it is in the neighborhood of a core point.

Noise or outlier: A point which is not a core point or border point.

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE



DBSCAN algorithm can be abstracted in the following steps:

1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density connected points and assign them to the same cluster as the core point.
   A point a and b are said to be density connected if there exist a point c which has a sufficient number of points in its neighbors and both the points a and b are within the eps distance. This is a chaining process. So, if b is neighbor of c, c is neighbor of d, d is neighbor of e, which in turn is neighbor of a implies that b is neighbor of a.
4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

Gaussian Mixtures:

A Gaussian mixture model (GMM) is a probabilistic model that assumes that the instances were generated from a mixture of several Gaussian distributions whose parameters are unknown. All the instances generated from a single Gaussian distribution form a cluster that typically looks like an ellipsoid. Each cluster can have a different ellipsoidal shape, size, density and orientation, just like in Figure 1. When you observe an instance, you know it was generated from one of the Gaussian distributions, but you are not told which one, and you do not know what the parameters of these distributions are. There are several GMM variants: in the simplest variant, implemented in the Gaussian Mixture class, you must know in advance the number k of Gaussian distributions.

The dataset **X** is assumed to have been generated through the following probabilistic process:

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

• For each instance, a cluster is picked randomly among k clusters. The probability of choosing the jth cluster is defined by the cluster's weight $\phi(j)$.6 The index of the cluster chosen for the ith instance is noted z(i).
• If z(i)=j, meaning the ith instance has been assigned to the jth cluster, the location $\mathbf{x}(i)$ of this instance is sampled randomly from the Gaussian distribution with mean $\boldsymbol{\mu}(j)$ and covariance matrix $\boldsymbol{\Sigma}(j)$.
This generative process can be represented as a graphical model (see Figure 1).
This is a graph which represents the structure of the conditional dependencies between random variables.

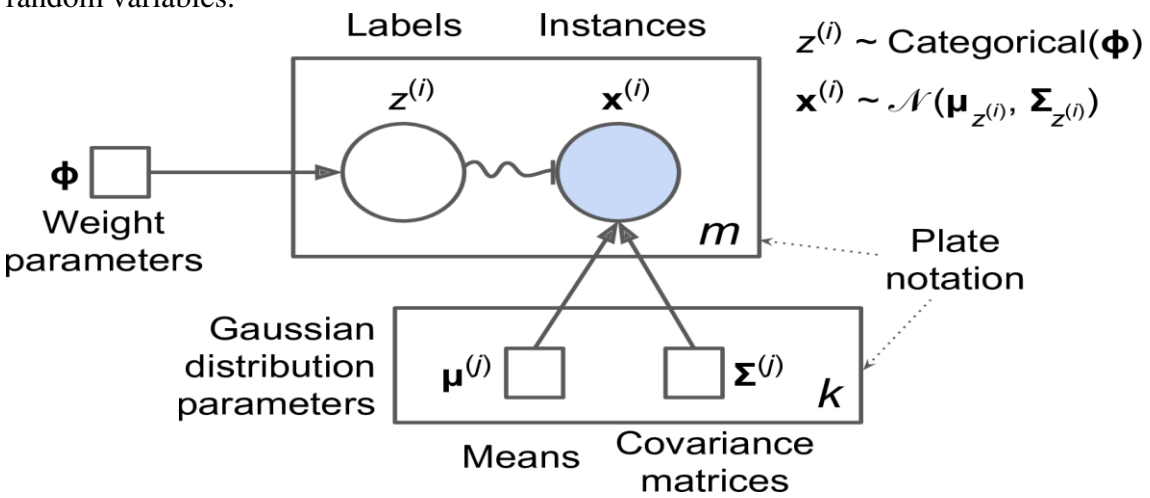


Figure 1. Gaussian mixture model

• The circles represent random variables.
• The squares represent fixed values (i.e., parameters of the model).
The large rectangles are called plates: they indicate that their content is repeated several times.
• The number indicated at the bottom right hand side of each plate indicates how many times its content is repeated, so there are m random variables z(i) (from z(1) to z(m)) and m random variables $\mathbf{x}(i)$, and k means $\boldsymbol{\mu}(j)$ and k covariance matrices $\boldsymbol{\Sigma}(j)$, but just one weight vector $\boldsymbol{\phi}$ (containing all the weights $\phi(1)$ to $\phi(k)$).
• Each variable z(i) is drawn from the categorical distribution with weights $\boldsymbol{\phi}$. Each variable $\mathbf{x}(i)$ is drawn from the normal distribution with the mean and covariance matrix defined by its cluster z(i).
• The solid arrows represent conditional dependencies. For example, the probability distribution for each random variable z(i) depends on the weight vector $\boldsymbol{\phi}$.
Note that when an arrow crosses a plate boundary, it means that it applies to all the repetitions of that plate, so for example the weight vector $\boldsymbol{\phi}$ conditions the probability distributions of all the random variables $\mathbf{x}(1)$ to $\mathbf{x}(m)$.

Dr K N MADHAVI LATHA
ASSOCIATE PROFESSOR
DEPT OF CSE
SIRCRRCOE

• The squiggly arrow from z(i) to **x**(i) represents a switch: depending on the value of z(i), the instance **x**(i) will be sampled from a different Gaussian distribution. For example, if z(i)=j, then $i \sim \mu j, \Sigma j$.
• Shaded nodes indicate that the value is known, so in this case only the random variables **x**(i) have known values: they are called observed variables. The unknown random variables z(i) are called latent variables.