4.1 HEBBIAN LEARNING RULE

Hebbian network is a single layer neural network which consists of one input layer with many input units and one output layer with one output unit.

This architecture is usually used for pattern classification. The bias which increases, has the net value 1. Hebbian rule works by updating the weights between neurons for each training sample.

4.1.1 Hebbian Learning Rule Algorithm

(1) Set all weights to zero, $w_i = 0$ for i = 1 to n, and bias to zero.

(2) For each input vector, S (input vector) t (target output pair), repeat steps 3-5.

(3) Set activations for input units with the input vector $X_i = S_i$ for i = 1 to n.

(4) Set the corresponding output value to the output neuron, i.e. y = t.

(5) Update weight and bias by applying Hebb rule for i = 1 to n :

$w_i(new) = w_i(old) + x_i y$

$b(new) = b(old) + y$

4.1.2 Solved Examples

Ex. 4.1.1 : We implement AND Gate : Truth-Table of AND Gate using bipolar sigmoidal function.

	Input			Target	
	x_1	x_2	b		y
x_1	−1	−1	1	y_1	−1
x_2	−1	1	1	y_2	−1
x_3	1	−1	1	y_3	−1
x_4	1	1	1	y_4	1

There are 4 training samples, so there will be 4 iterations. Since the activation function used is Bipolar sigmoidal function, so the range is [− 1, 1].

Soln. :

▶ **Step 1 :** Set weight and bias to zero, i.e. $w = [0\ 0\ 0]^T$ and b = 0.

▶ **Step 2 :** Set input vector $X_i = S_i$ for i = 1 to 4.

$\therefore X_1 = [-1\ -1\ 1]^T$

$X_2 = [-1\ 1\ 1]^T$

$X_3 = [1\ -1\ 1]^T$

$X_4 = [1\ 1\ 1]^T$

▶ **Step 3 :** Output value is y = t

▶ **Step 4 :** We modify weights using Hebbian rule first iteration

$$W(new) = w(old) + x_1 y_1$$
$$= [0\ 0\ 0]^T + [-1\ -1\ 1]^T [-1]$$
$$= [1\ 1\ -1]^T$$

For the second iteration, the final weight of the first one will be used and so on.

Second iteration

$$W(new) = [1\ 1\ -1]^T + [-1\ 1\ 1]^T \cdot [-1]$$
$$= [2\ 0\ -2]^T$$

Third iteration

$$W(new) = [2\ 0\ -2]^T + [1\ -1\ 1]^T \cdot [-1]$$
$$= [1\ 1\ -3]^T$$

Fourth iteration

$$W(new) = [1\ 1\ -3]^T + [1\ 1\ 1]^T \cdot [1]$$
$$= [2\ 2\ -2]^T$$

Hence the final weight matrix is

$$[2\ 2\ -2]^T = \begin{bmatrix} 2 \\ 2 \\ -2 \end{bmatrix}$$

Testing the Network

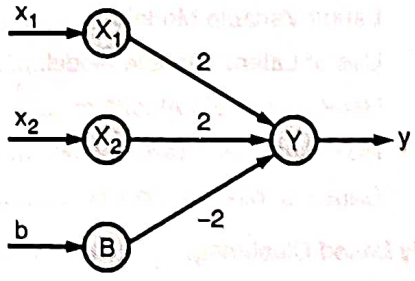

Fig. Ex. 4.1.1 : Network with the final weights

For $x_1 = -1$, $x_2 = -1$, $b = 1$,

$Y = (-1)(2) + (-1)(2) + (1)(-2) = -6$

For $x_1 = -1$, $x_2 = 1$, $b = 1$,

$Y = (-1)(2) + 1(2) + (1)(-2) = -2$

For $x_1 = 1$, $x_2 = 1$, $b = 1$,

$Y = (1)(2) + (-1)(2) + (1)(-2) = -2$

For $x_1 = 1$, $x_2 = 1$, $b = 1$,

$Y = (1)(2) + (1)(2) + (1)(-2) = +2$

Now, decision boundary is

$2x_1 + 2x_2 - 2b = y$

Replacing y with 0, we get

$2x_1 + 2x_2 - 2b = 0$,

Since, bias, $b = 1$,

$\therefore \quad 2x_1 + 2x_2 = 2$

i.e. $x_1 + x_2 = 1$ is the final equation

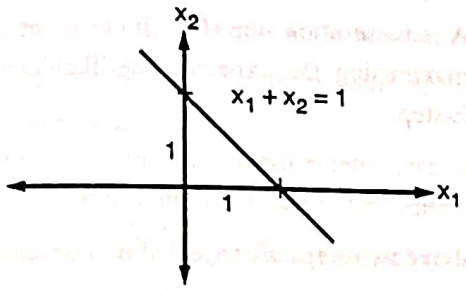

Fig. Ex. 4.1.1(a) : Decision boundary of AND

Ex. 4.1.2 : Calculate the net input for the network shown in Fig. Ex. 4.1.2 with bias included in the network.

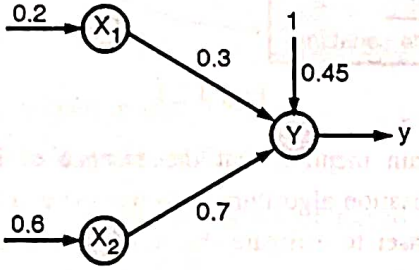

Fig. Ex. 4.1.2

Soln.

The given net consists of two input neurons, a bias and an output neuron.

The inputs are

$[x_1, x_2] = [0.2, 0.6]$ and the weights are

$[w_1, w_2] = [0.3, 0.7]$

Since the bias is included b = 0.45 and bias input x_0 is equal to 1, then the net input is given by

$y_{in} = b + x_1 w_1 + x_2 w_2$

$= 0.45 + (0.2)(0.3) + (0.6)(0.7)$

$= 0.45 + 0.06 + 0.42$

$= 0.93$

$\therefore \quad y_{in} = 0.93$ is the net input

4.1.3 Explanation of Hebb-rule

According to Hebb's rule, the weights are found to increase proportionately to the product of input and output.

It implies that in a Hebb network, it two neurons are interconnected then the weights associated with these neurons can be increased by changes in the synaptic gap.

4.1.4 Hebbian Learning is Unsupervised

Hebbian learning requires no other information than the activates, such as labels or error signals. It is an unsupervised learning method.

Hebbian learning is not a concrete learning rule, it is a postulate on the fundamental principle of biological learning.

4.1.5 Importance of Hebbian Learning

Hebbian learning can strengthen the neural response that is done by an input. This can be useful if the response made is appropriate to the situation, but it can also be counterproductive if a different response would be more appropriate.

4.1.6 Limitations of Hebbian Models

(i) Hebb's principle does not cover all forms of synaptic long-term plasticity.

(ii) Hebb did not give any rules for inhibitory synapses.

(iii) He also did not make any predictions for anti-causal spike sequences.

4.1.7 Hebbian Learning Rule in Machine Learning

The Hebbian Learning Rule is a learning rule that specifies how much the weight of the connection between two units should be increased or decreased in proportion to the product of their activation.

Principles

From the point of view of artificial neurons and artificial neural networks, Hebb's principle can be described as a method of determining how to alter the weights between model neurons.

The weight between two neurons increases if the two neurons activate simultaneously, and reduces if they activate separately.

Nodes that tend to be either both positive or both negative at the same time having strong positive weights, while those that tend to be opposite have strong negative weights.

4.2 MODEL BASED CLUSTERING

4.2.1 Expectation Maximization Clustering

- The expectation maximisation algorithm performs maximum likelihood estimation in the presence of latent variables.
- First it estimates the values of the latent variables, then it optimises the model.
- Then it repeats the two steps till convergence takes place.

Expectation maximization (EM) clustering is an unsupervised clustering algorithm. It also extends to NLP applications, like Hidden Markov Models and medical imaging. For this it uses Latent Dirichlet Allocation.

4.2.2 Use of EM

(i) The Expectation Maximisation (EM) algorithm can find maximum-likelihood estimates for model parameters when the data is incomplete.

(ii) It is an iterative way to approximate the maximum likelihood function.

(iii) This can be used to fill the missing data in a sample.

(iv) It can be used on the basis of unsupervised learning of clusters.

(v) It can be used for the purpose of estimating the parameters of Hidden Markov Model (HMM).

(vi) It can be use to find the values of latent variables

4.2.3 EM Algorithm

GQ. Write short note on : EM algorithm. **(5 Marks)**

- In statistics, an expectation-maximisation (EM) algorithm is an iterative method.
- It finds maximum a posterior (MAP) estimates of parameters in statistical models, where model depends on unobserved latent variables.
- The EM iteration alternates between :
  - (i) It performs an expectation (E) step : that creates a function for the expectation of the **log-likelihood** and it is evaluated using the current estimate for the parameters, and
  - (ii) A maximisation step (M) : it computes parameters maximising the expected log-likelihood fund on E-step.
- These parameter-estimates are used to determine the distribution of the latent variables in the next E-step.
- The above two steps are repeated till convergence takes place.

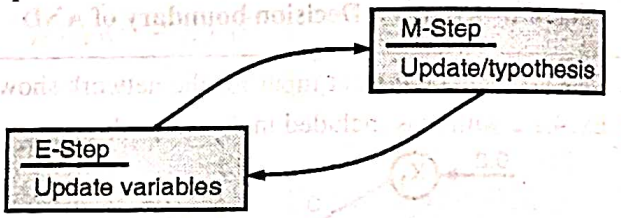

Fig. 4.2.1

- We again mention that the essence of Expectation-Maximisation algorithm is to use the available data of the dataset to estimate the missing data and then we use that data to update the values of the parameter.

Flow-chart for EM algorithm

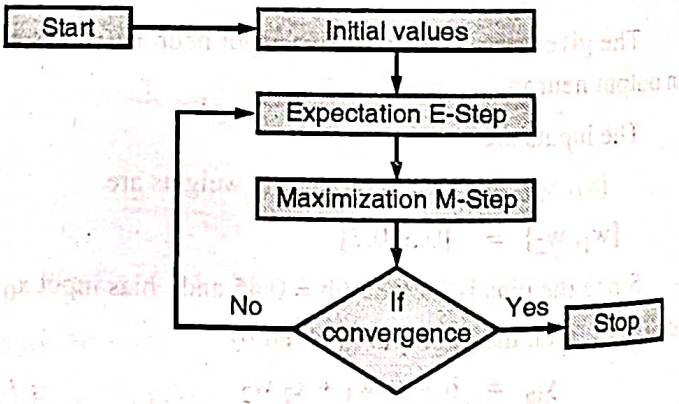

Fig. 4.2.2

4.2.4 Advantages and Disadvantages of EM Algorithm

Advantages

(i) It is guaranteed that likelihood will increase with each iteration.

(ii) The E-step and M-step are quite easy to implement for many problems.

(iii) Solutions to the M-steps exist in the closed form.

Disadvantages

(i) It has slow convergence

(ii) It makes convergence to the local optima only.

(iii) It requires both the probabilities, forward and backward (note that, numerical optimisation requires only forward probability).

4.2.5 Applications of EM-Algorithm

The latent variable model has several real-life applications in Machine-learning :

(i) Use to calculate the Gaussian density of a function.

(ii) It helps to fill in the missing data during a sample.

(iii) In domains like Natural Language Processing (NLP), computer Vision etc. It finds plenty of applications.

(iv) It is also helpful in image reconstruction in the field of Medicine, and structural Engineering.

(v) Used to estimate the prameters of mixed models like gaussian Mixture Models.

4.3 EXPECTATION MAXIMIZATION ALGORITHM

Purpose of Algorithm

- In many of the problem statements of machine learning, we obtain many relevant features to build the required model. But only a few of these features are observable. (The non-observable features are also called as latent features).
- As we do not have the values for non-observable (latent) variables, the Expectation maximization algorithm tries to use the existing data to determine the optimum values for these variables and then one can find model parameters.

4.3.1 Latent Variable Model

- A latent variable model consists of **observable** variables along with **unobservable** variables.
- Observed variables in the dataset can be measured whereas unobserved (latent / hidden) variables are inferred from the observed variables.

4.3.2 Use of Latent Variable Model

(i) It can be used to find the local maximum likelihood (MLE) parameters for latent variables in a statistical or mathematical model.

(ii) It is used to predict these missing values in the dataset if we can predict the general form of probability distribution associated with these latent variables.

(iii) The concept behind this algorithm is to use the observable samples of latent variables to predict the values of unobservable samples.

This process is repeated till the convergence of the value occurs.

4.3.3 Use-Case of EM Algorithm

- Normal distribution about which we are already familiar is also called Gaussian distribution.
- This distribution is used in the field of machine learning and statistics.
- We give below with a few Gaussian distribution with different values of σ^2.

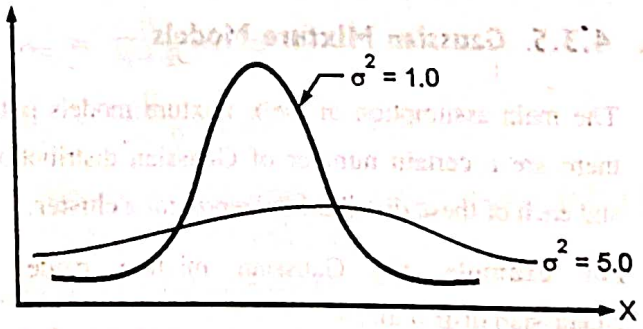

Fig. 4.3.1

- The higher the valve of σ (standard deviation) more would be the spread along X-axis.
- In 1-D space, the probability density function of Gaussian distribution is given by,

$$f(x \mid \mu, \sigma^2) = \frac{1}{(\sqrt{2\pi}) \cdot \sigma} e^{-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2}$$

where μ is mean and σ^2 is variance of the distribution.

- This is true only in 1-D only.
- In the case of two variables, we will have a 3D-bell curve instead of a 2-D bell shaped curve.
- The probability density function for 3 D-bell curve is given by,

$$f(x \mid \mu, \sum) = \frac{1}{\sqrt{2\pi \mid \sum \mid}} \exp.\left[-\frac{1}{2}(x-\mu) \sum^{-1} (x-\mu)\right]$$

where x is input vector, μ is the 2-D mean vector and $\sum$ is 2×2 covariance matrix.

- This can be generalized for d-dimension.

4.3.4 Multivariate Gaussian Model

- For the multivariate Gaussian model, we have x and μ as vector of length d, and $\sum$ would be a $d \times d$ covariance matrix.
- Hence, for a dataset having 'd' features, we would have a mixture of k Gaussian distributions (where k represents the number of clusters), each having a certain mean vector and variance matrix.
- For finding mean and variance for each Gaussian, we use a technique called Expectation Maximization (EM).

4.3.5 Gaussian Mixture Models

- The main assumption of these mixture models is that there are a certain number of Gaussian distributions, and each of these distributions represent a **cluster**.
- **For example**, the Gaussian mixture model of 2 Gaussian distributions : $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$
- Here we have to estimate a total of 5 parameters : $\theta = (p, \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$; where p is the probability that the data comes from the first Gaussian distribution and $(1 - p)$ is that it comes from second Gaussian distribution.

Then the probability density function of the mixture model is given by :

$$g(x \mid \theta) = p g_1 (x \mid \mu_1, \sigma_1^2) + (1-p)\, g_2 (x \mid \mu_2, \sigma_2^2).$$

We find $\theta = (p \mid \mu_1, \sigma_1^2, \mu_2, \sigma_2^2)$ through

EM iteration And that gives the best fit.

Ex. 4.3.1 : Suppose Coin A and B is used for tossing. Each coin is tossed 10 times. Following table shows the observation sequence of getting H and T when coin A and B is used. What is the probability of getting H if coin A and B is used?

	Number of Toss									
Coin Used for Toss	1	2	3	4	5	6	7	8	9	10
B	H	H	H	H	H	T	T	T	T	T
A	H	H	H	H	H	H	H	H	T	T
A	H	H	H	H	H	H	H	H	H	T
B	H	H	H	H	T	T	T	T	T	T
A	H	H	H	H	H	H	H	T	T	T

Solution :

Let's first calculate number of H and T for each coin as follows

	Number of Toss											
Coin Used for Toss	1	2	3	4	5	6	7	8	9	10	Coin A	Coin B
B	H	H	H	H	H	T	T	T	T	T		5H, 5T
A	H	H	H	H	H	H	H	H	T	T	8H, 2T	
A	H	H	H	H	H	H	H	H	H	T	9H, 1T	
B	H	H	H	H	T	T	T	T	T	T		4H, 6T
A	H	H	H	H	H	H	H	T	T	T	7H, 3T	
Total											24H, 6T	9H, 11T

Probability of getting Head when coin A is used,

$$P_A = \frac{24}{24+6} = 0.8$$

Probability of getting Head when coin B is used,

$$P_B = \frac{9}{9+11} = 0.45$$

In this example if the coin state is hidden i.e. whether coin A or B is used is not given then how we will calculate the probability? Answer to this is EM algorithm.

Ex. 4.3.2 : Suppose Coin A and B is used for tossing. Each coin is tossed 10 times. Following table shows the observation sequence of getting H and T for each round. But which coin is used for which round is not known. Then how to calculate the probability of getting H for coin A and B?

	Number of Toss									
Round number	1	2	3	4	5	6	7	8	9	10
0	H	H	H	H	H	T	T	T	T	T
1	H	H	H	H	H	H	H	H	T	T
2	H	H	H	H	H	H	H	H	H	T
3	H	H	H	H	T	T	T	T	T	T
4	H	H	H	H	H	H	H	T	T	T

☑ **Solution :**

When only observation sequence is known, but the state is not known (Coin A or B) then EM algorithm is used.

Now Let's solve this example using EM algorithm.

Assume $P_A = 0.6$ and $P_B = 0.5$

Round 0 : In round 0 there are 5 H, 5 T and total tosses are 10.

Now we will calculate probability of using A and B coin as,

$$A = (P_A)^H (1-P_A)^{N-H}$$
$$= (0.6)^5 (1-0.6)^{10-5} = 0.00079626$$
$$B = (P_B)^H (1-P_B)^{N-H}$$
$$= (0.5)^5 (1-0.5)^{10-5} = 0.0009765$$

Now we will apply Normalization,

$$A_N = \frac{A}{A+B} = 0.45$$

$$B_N = \frac{B}{A+B} = 0.55$$

Now we will calculate probability of getting H and T for Coin A and B for round 0 as,

$A_H = A_N$ * Number of H = 0.45* 5 = 2.25

$A_T = A_N$ * Number of T = 0.45* 5 = 2.25

$B_H = B_N$ * Number of H = 0.55* 5 = 2.75

$A_H = B_N$ * Number of T = 0.55* 5 = 2.75

Round 1 : In round 1 there are 8 H, 2 T and total tosses are 10.

Now we will calculate probability of using A and B coin as,

$$A = (P_A)^H (1-P_A)^{N-H}$$
$$= (0.6)^8 (1-0.6)^{10-8} = 0.002687$$
$$B = (P_B)^H (1-P_B)^{N-H}$$
$$= (0.5)^8 (1-0.5)^{10-8} = 0.0009755$$

Now we will apply Normalization,

$$A_N = \frac{A}{A+B} = 0.73$$

$$B_N = \frac{B}{A+B} = 0.27$$

Now we will calculate probability of getting H and T for Coin A and B for round 1 as,

$A_H = A_N$ * Number of H = 0.73* 8 = 5.84

$A_T = A_N$ * Number of T = 0.73* 2 = 1.46

$B_H = B_N$ * Number of H = 0.27* 8 = 2.16

$A_H = B_N$ * Number of T = 0.27* 2 = 0.54

Round 2 : In round 2 there are 9 H, 1 T and total tosses are 10.

Now we will calculate probability of using A and B coin as,

$$A = (P_A)^H (1-P_A)^{N-H}$$
$$= (0.6)^9 (1-0.6)^{10-9} = 0.004031$$
$$B = (P_B)^H (1-P_B)^{N-H}$$
$$= (0.5)^9 (1-0.5)^{10-9} = 0.0009765$$

Now we will apply Normalization,

$$A_N = \frac{A}{A+B} = 0.80$$

$$B_N = \frac{B}{A+B} = 0.20$$

Now we will calculate probability of getting H and T for Coin A and B for round 2 as,

$A_H = A_N$ * Number of H = 0.80 * 9 = 7.2

$A_T = A_N$ * Number of T = 0.80 * 1 = 0.8

$B_H = B_N$ * Number of H = 0.2 * 9 = 1.8

$A_H = B_N$ * Number of T = 0.2 * 1 = 0.2

Round 3 : In round 3 there are 4 H, 6 T and total tosses are 10.

Now we will calculate probability of using A and B coin as,

$$A = (P_A)^H (1 - P_A)^{N-H}$$
$$= (0.6)^4 (1 - 0.6)^{10-4} = 0.0005308$$
$$B = (P_B)^H (1 - P_B)^{N-H}$$
$$= (0.5)^4 (1 - 0.5)^{10-4} = 0.0009765$$

Now we will apply Normalization,

$$A_N = \frac{A}{A+B} = 0.35$$

$$B_N = \frac{B}{A+B} = 0.65$$

Now we will calculate probability of getting H and T for Coin A and B for round 3 as,

$A_H = A_N$ * Number of H = 0.35 * 4 = 1.4

$A_T = A_N$ * Number of T = 0.35 * 6 = 2.1

$B_H = B_N$ * Number of H = 0.65 * 4 = 2.6

$A_H = B_N$ * Number of T = 0.65 * 6 = 3.9

Round 4 : In round 4 there are 7 H, 3 T and total tosses are 10.

Now we will calculate probability of using A and B coin as,

$$A = (P_A)^H (1 - P_A)^{N-H}$$
$$= (0.6)^7 (1 - 0.6)^{10-7} = 0.001792$$
$$B = (P_B)^H (1 - P_B)^{N-H}$$
$$= (0.5)^7 (1 - 0.5)^{10-7} = 0.0009765$$

Now we will apply Normalization,

$$A_N = \frac{A}{A+B} = 0.65$$

$$B_N = \frac{B}{A+B} = 0.35$$

Now we will calculate probability of getting H and T for Coin A and B for round 4 as,

$A_H = A_N$ * Number of H = 0.65 * 7 = 4.55

$A_T = A_N$ * Number of T = 0.65 * 3 = 1.95

$B_H = B_N$ * Number of H = 0.35 * 7 = 2.45

$A_H = B_N$ * Number of T = 0.35 * 3 = 1.05

Now we will calculate P_A and P_B by summarizing the results of round 0 to round 4.

	Coin A		Coin B	
	A_H	A_T	B_H	B_T
0	2.25	2.25	2.75	2.75
1	5.84	1.46	2.16	0.54
2	7.2	0.8	1.8	0.2
3	1.4	2.1	2.6	3.9
4	4.55	1.95	2.45	1.05
Total	21.24	8.56	11.76	8.44

Probability of getting H when Coin A is used,

$$P_A = \frac{\Sigma H}{\Sigma H + \Sigma T} = 0.71$$

Probability of getting H when Coin B is used,

$$P_B = \frac{\Sigma H}{\Sigma H + \Sigma T} = 0.58$$

Now with these new values of P_A and P_B second iteration is applied. This process is repeated unless and until there will be no change in the values of P_A and P_B and then that will be the final probabilities.

4.4 DENSITY BASED CLUSTERING

- It is a clustering method and is Density-based spatial clustering of applications with noise (DBSCAN), clustering method.
- Clusters are dense regions in the data space. They are separated by regions of the lower density of points.
- The **DBSCAN algorithm** is based on this notion of "clusters" and "noise".
- The basic concept is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

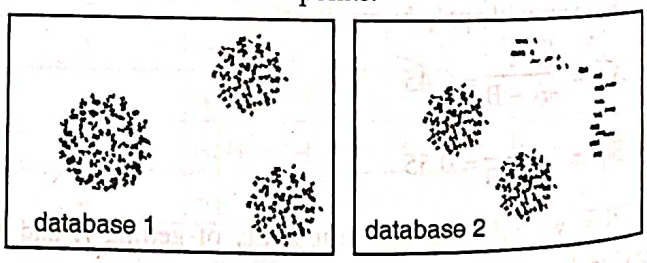

Fig. 4.4.1

Why DBSCAN ?

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

4.4.1 Irregularities in the Data

Data may contain irregularities like

1. Clusters can be of arbitrary shape as shown in the Fig. 4.4.2.
2. Data may contain noise.

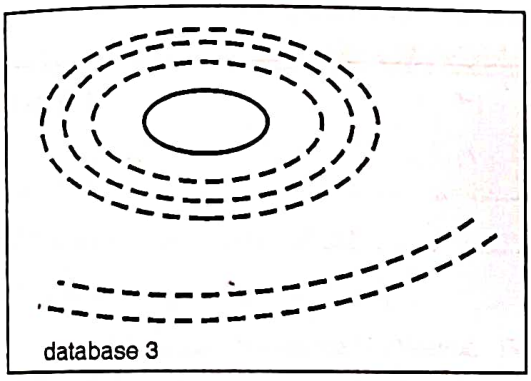

Fig. 4.4.2

For non-convex clusters and outliers / noises, K-means algorithm has difficulties for identifying these clusters with arbitrary shapes.

DBSCAN algorithm requires two parameters :

1. eps

- Around a data-point, it contains the neighbourhood. It means : If the distance between two points is lower or equal to 'eps' then they are considered as neighbours.
- If the eps value is very small, then large part of the data will be considered as outliers.
- If it is chosen very large, then clusters will merge and majority of the data-points will be in the same clusters.
- Hence the eps-value is based on the **k-distance graph.**

2. MinDts

- It is the minimum number of data points (neighbors) within eps radius. If the dataset is large, the larger value of MinDts must be chosen.
- In general, the minimum MinDts can be derived from the number of dimensions D in the dataset.
- Thus, MinDts ≥ D + 1
- The minimum value of MinDts should be at least 3.
- In this algorithm, we come across 3 types of Data points :

(i) **Core point :** A point is a core point if it has more than MinDts points within eps.

(ii) **Border point :** A point which has fewer than MinDts within eps but it is in the neighbourhood of a core point.

(iii) **Noise or outlier :** A point which is not a core point or border point.

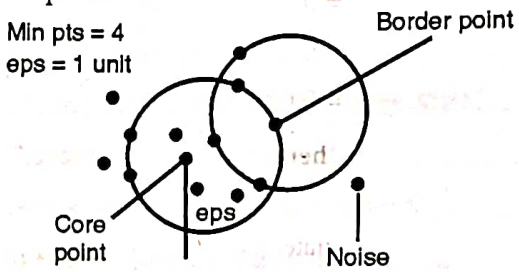

Fig. 4.4.3

- We carry out DBSCAN algorithm in the following steps :
  1. First find all the neighbour points in the eps and identify the core points or visited with more than MinDts neighbours.
  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.
- 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. Thus if b is neighbor of C, C is neighbor of d, d is neighbor of e, which in turn is a neighbor of a then it implies that b is neighbor of a.
- We iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

4.4.2 The basic Principle of DBSCAN Clustering

- The main principle of DBSCAN is to find the neighbourhoods of data points, exceeding certain density threshold.
- The density thereshold is defined by two parameters :
  1. The radius of neighbourhood (eps) and
  2. The minimum number of neighbours / data points (MinDts) within the radius of the neighbourhood.

4.4.3 DBSCAN and K-means

- DBSCAN clustering cannot efficiently handle high dimensional datasets.
- K-means clustering does not work well with outliers and noisy datasets.
- DBSCAN clustering efficiently handles outliers and noisy dataset.

Advantages and disadvantages of DBSCAN

1. DBSCAN does not require a priori specification of number of clusters.
2. It is able to identify noise data while clustering.
3. DBSCAN algorithm can find arbitrarily shaped and arbitrarily size clusters.

Chapter Ends...

□□□