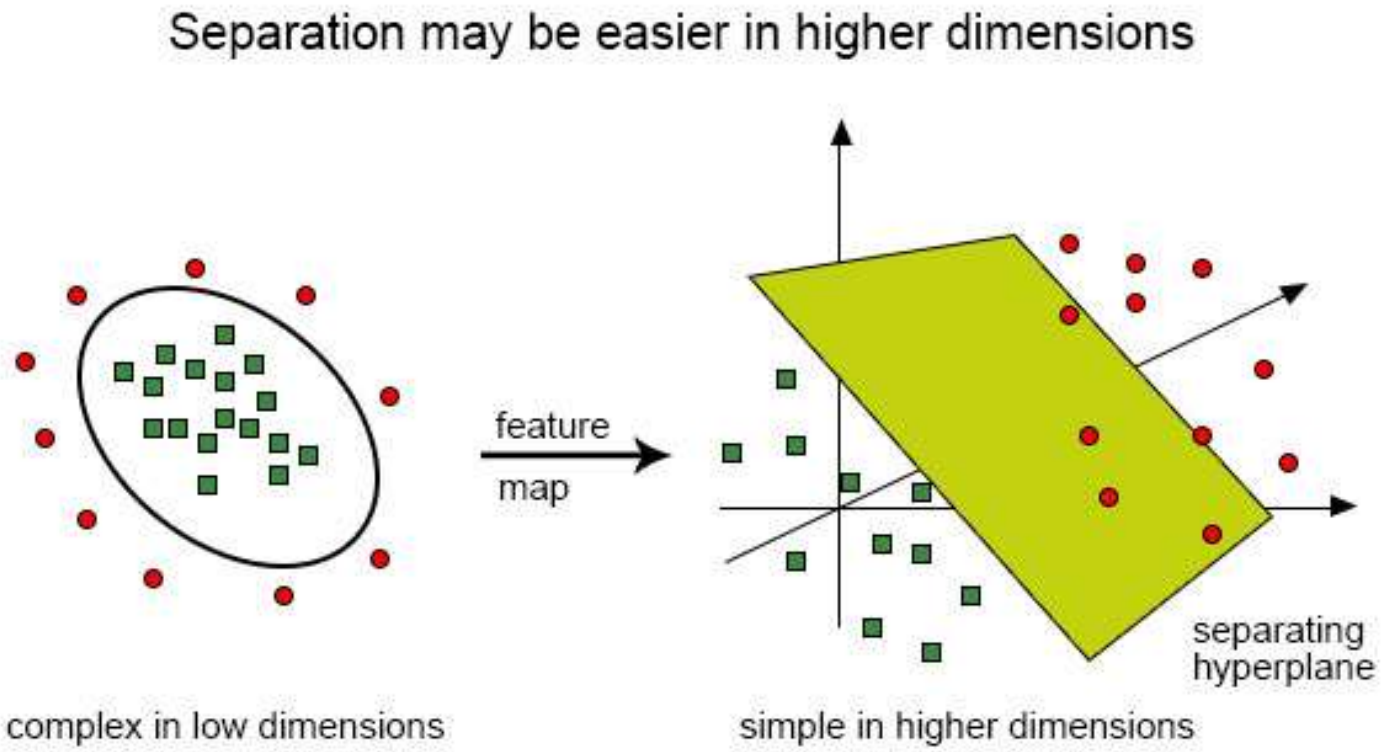
Unit-6
Kernel Methods and RBFNN

By: Arjun Saud, Asst. Prof. CDCSIT,TU

Kernel Methods

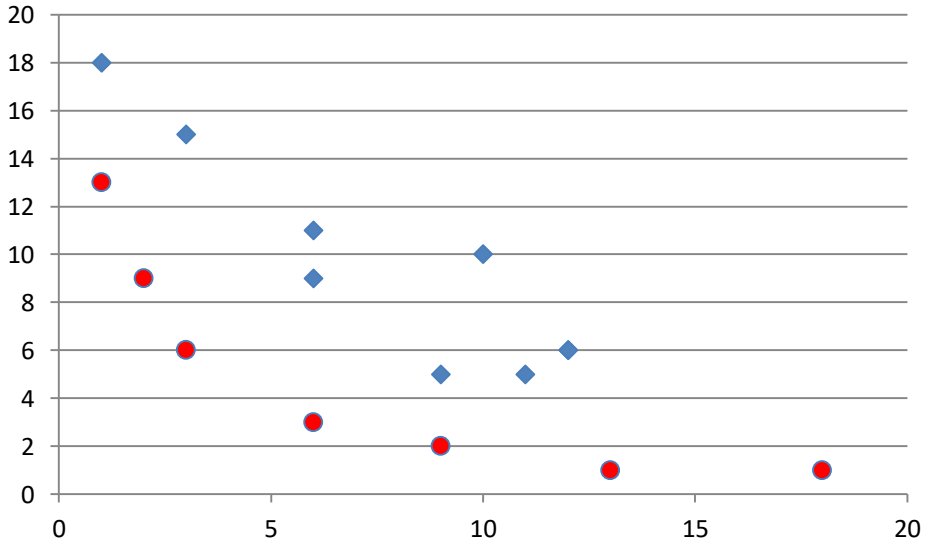
- Kernels or kernel methods (also called Kernel functions) are sets of different types of algorithms that are being used for pattern analysis. They are used to solve a non-linear problem by using a linear classifier.
- Sometimes it may be difficult to divide lower dimension dataset using a linear line or hyperplane. Here comes the use of kernel function which takes the points to higher dimensions, solves the problem over there and returns the output.
- Consider the situation shown in the figure given in next slide.

Kernel Methods



Kernel Methods

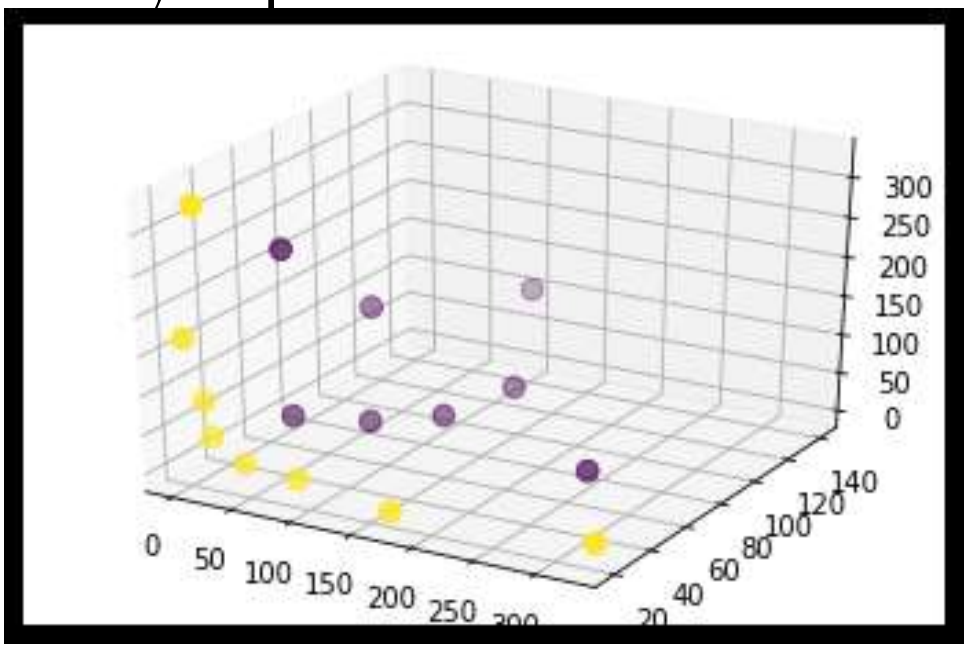
- Consider the following data points: {(1,18), (1,13), (2,9), (3,6), (3,15), (6,11), (6,9), (6,3), (9,5), (9,2), (10,10),(11,5),(12,6), (13,1),(16,3),(18,1)}.
- If plot the above dataset in graph, we can clearly see that the dataset is not linearly separable.



Kernel Methods

- Lets transform above data into 3D data using the following polynomial kernel given below and plot the graph of resulting points, we will get the graph like below. From the graph we can clearly see that data points are linearly separable now.

$$f(x) = (x^2, \sqrt{2}xy, y^2)$$



Kernel Methods

- There are different types of kernel like linear kernel, polynomial kernel, exponential kernel, Gaussian kernel, etc.
- Gaussian kernel and exponential kernel are examples of a radial basis function kernel. The equations given below are Gaussian and exponential kernel functions respectively .

$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right), \text{where } \sigma \text{ is free parameter}$$

$$k(x,y) = \exp\left(-\frac{\|x-y\|}{2\sigma^2}\right)$$

Cover's Theorem of Separability of Patterns

- When a radial-basis function (RBF) network is used to perform a *complex pattern classification* task, the problem is basically solved by first transforming it into a high dimensional space in a nonlinear manner and then separating the classes in the output layer. The underlying justification is found in *Cover's theorem*.
- The theorem states that *"A complex pattern-classification problem, cast in a high-dimensional space nonlinearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated"*.

XOR Problem with Radial Basis Function

- We know that XOR problem cannot be solved by single layer perceptron because it is not linearly separable.
- We have dealt with this problem by introducing a hidden layer with two neurons in MLP.
- Alternatively, we can deal with XOR problem using Radial Basis Function (RBF).
- To achieve the XOR function we would like to construct a pattern classifier that produces the output 0 for the input patterns (0,0) and (1,1) and output 1 for the input patterns (0,1) and (1,0).

XOR Problem with Radial Basis Function

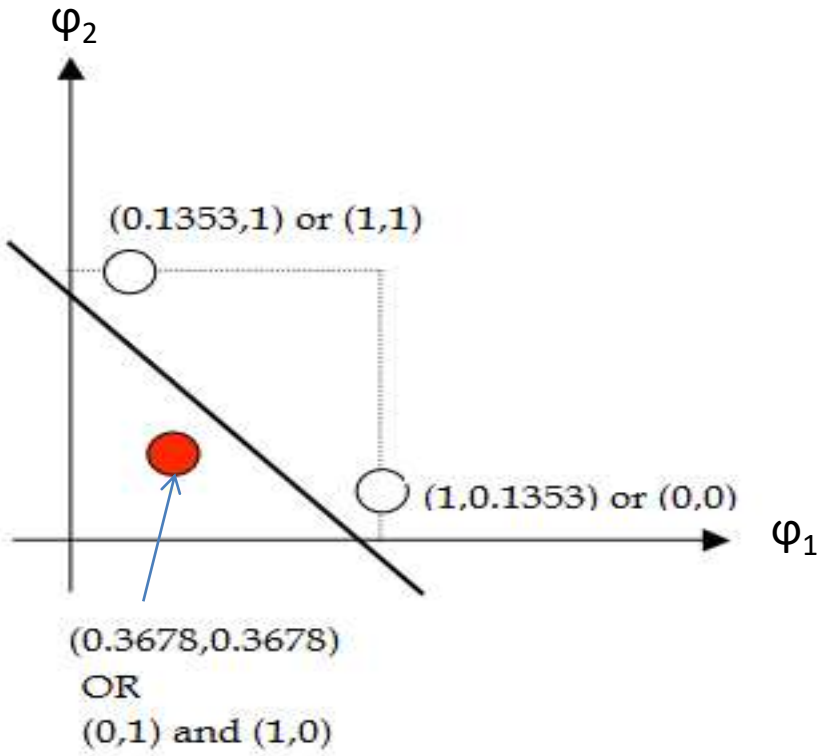
- Lets define a pair of Gaussian hidden functions as below:

$$\varphi_1(x) = \exp\left(-\|x - c_1\|^2\right) \qquad \varphi_2(x) = \exp\left(-\|x - c_2\|^2\right)$$

- RBF centers need to be chosen, assume that c_1=(0,0) and c_2=(1,1).
- After applying radial basis function on the XOR input patterns, we will get the transformed input patterns as shown in next slide.
- If we draw the transformed data points we will get the graph as below, which is clearly linearly separable.

XOR Problem with Radial Basis Function

x_1	x_2	φ_1	φ_2
0	0	1.0	0.1353
0	1	0.3678	0.3678
1	0	0.3678	0.3678
1	1	0.1353	1.0



XOR Problem with Radial Basis Function

Exercise

- Solve the XOR problem using RBF by taking (1,0) and (0,1) as centers.
- Solve the XOR problem using RBF such that transform 2D data points into 4D data points by taking each of the as data point center. (Hint use the RBF $\varphi(x) = \exp\left(-\|x - c\|^2\right)/2$

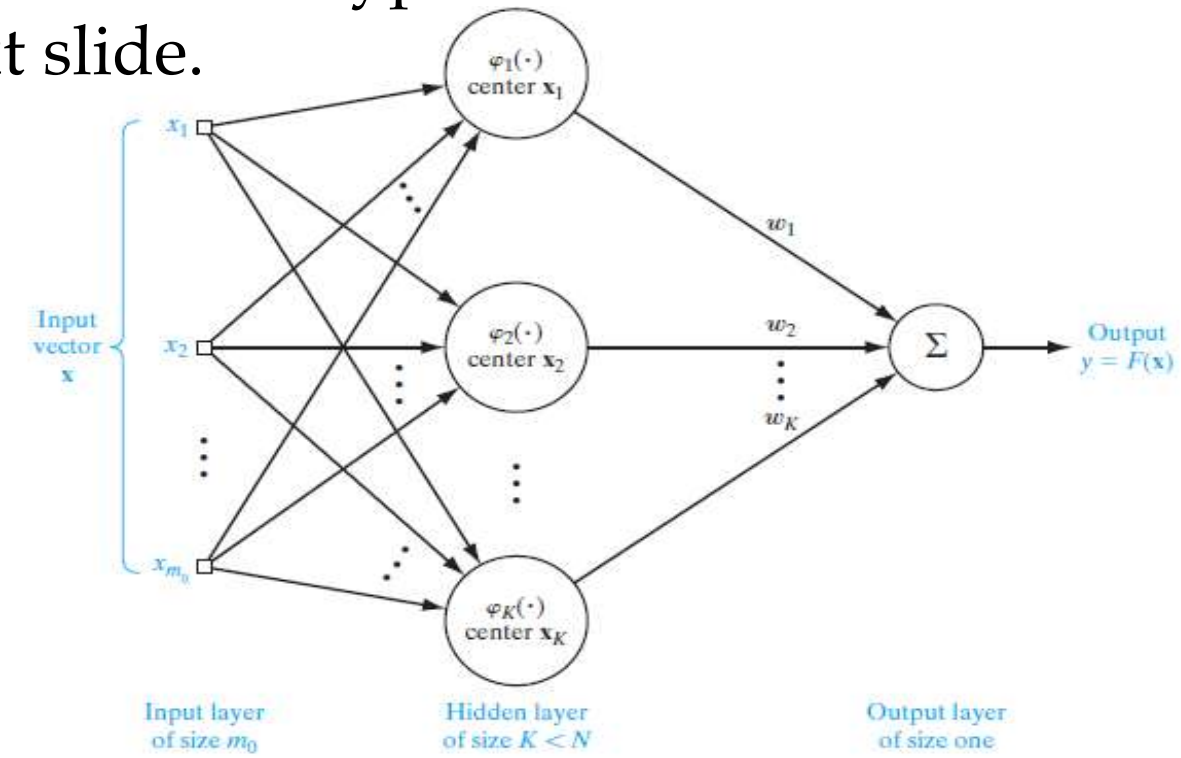
Radial Basis Function Networks

- Radial basis function (RBF) networks are artificial neural networks that typically have three layers: an *input layer*, a *hidden layer* with a non-linear RBF activation function and a linear *output layer*.
- **Input Layer:** This layer consists of m_0 source nodes, where m_0 is the dimensionality of the input vector **x.**
- **Hidden Layer:** This layer consists of the K computation units (K≤N), where N is number training samples. Each hidden unit is equipped with a radial basis activation function. The RBF is given below.

$$\varphi(x) = \varphi\left(\left\| x - x_j \right\|\right), \quad j = 1,2,\ldots.K$$

Radial Basis Function Networks

- **Output Layer**: This layer of neural network contains one or more nodes depending upon the need.
- Structure of a typical RBF neural network is given in next slide.



Radial Basis Function Networks

- RBF networks are typically trained from pairs of input and target values by a two-step algorithm.
- In the first step, the center vectors of the RBF functions in the hidden layer are chosen. This step can be performed in several ways; centers can be randomly sampled from some set of examples, or they can be determined using K-means clustering.
- The second step simply fits a model with coefficients to the hidden layer's outputs with respect to some objective function.

K-means Clustering

- The process of grouping a set of data objects into classes of *similar* objects is called clustering.
- A cluster is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters.
- Clustering is unsupervised machine learning technique.
- During clustering of data objects, we need to find similarity between data objects. The most popular similarity measure used in clustering is *Euclidean distance*.

K-means Clustering

- k-means is one of the simplest unsupervised learning algorithms used to solve clustering problem.
- The main idea is to define k centers, one for each cluster. The better choice is to place them as much as possible far away from each other.

K-means Clustering

K-means Algorithm

1. Randomly select *K* cluster centers.
2. Calculate the distance between each data point and cluster centers.
3. Assign the data point to the cluster center that is closest to the data point.
4. Recalculate the new cluster center using: $c_i = \frac{1}{|c_i|} \sum_{i=1}^{|c_i|} x_i$
5. If no data point change cluster then stop, otherwise repeat go to step 2.

K-means Clustering

Example

Divide the data points {(1,1), ((2,1), (4,3), (5,4)} into two clusters.

Solution

Let p1=(1,1) p2=(2,1) p3=(4,3) p4=(5,4)

Initial step

Let c1=(1,1) and c2=(2,1) are two initial cluster centers.

K-means Clustering

Iteration 1

Calculate distance between clusters centers and each data points

$$d(c1, p1) = 0$$

$$d(c2, p1) = \sqrt{(2-1)^2 + (1-1)^2} = 1$$

$$d(c1, p2) = \sqrt{(2-1)^2 + (1-1)^2} = 1$$

$$d(c2, p2) = \sqrt{(2-2)^2 + (1-1)^2} = 0$$

$$d(c1, p3) = \sqrt{(4-1)^2 + (3-1)^2} = 3.6$$

$$d(c2, p3) = \sqrt{(4-2)^2 + (3-1)^2} = 2.83$$

$$d(c1, p4) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(c2, p4) = \sqrt{(5-2)^2 + (4-1)^2} = 4.24$$

Thus after first iteration

Cluster 1= {p1} *Cluster 2={p2,p3,p4}*

K-means Clustering

Iteration 2

New cluster centers are: c1=(1,1) and c2=(11/3,8/3)

Calculate distance between new cluster centers and each data points

$$d(c1, p1) = 0$$

$$d(c1, p2) = \sqrt{(2-1)^2 + (1-1)^2} = 1$$

$$d(c1, p3) = \sqrt{(4-1)^2 + (3-1)^2} = 3.6$$

$$d(c1, p4) = \sqrt{(5-1)^2 + (4-1)^2} = 5$$

$$d(c2, p1) = \sqrt{(11/3-1)^2 + (8/3-1)^2} = 3.14$$

$$d(c2, p2) = \sqrt{(11/3-2)^2 + (8/3-1)^2} = 2.35$$

$$d(c2, p3) = \sqrt{(11/3-4)^2 + (8/3-3)^2} = 0.46$$

$$d(c2, p4) = \sqrt{(11/3-5)^2 + (8/3-4)^2} = 1.88$$

- Thus after second iteration:

  Cluster 1= {p1,p2} Cluster 2={p3,p4}

- Repeat this process till no re-assignment of data points to clusters.

K-means Clustering

Iteration 3

New cluster centers are: c1=(3/2,1) and c2=(9/2,7/2)

Calculate distance between new cluster centers and each data points

$d(c1, p1) = ?$

$d(c1, p2) = ?$

$d(c1, p3) = ?$

$d(c1, p4) = ?$

$d(c2, p1) = ?$

$d(c2, p2) = ?$

$d(c2, p3) = ?$

$d(c2, p4) = ?$

- Thus after third iteration:

  Cluster 1= { } Cluster 2={ }

- Repeat this process till no re-assignment of data points to clusters.

Training RBF Hidden Layer Using K-means

1. Initialization: Choose the cluster centers randomly.
2. Sampling: Draw a sample vector (x) from training set.
3. Similarity Matching: Find Euclidean distance of the sample vector from each cluster center and determine closest (winning) cluster center. Mathematically, winning cluster center is given by:

   $k(x) = \arg\min_k \left(\|x - \mu_k\| \right)$, *where* μ_k *is cluster center of* k^{th} *cluster*
4. Update Cluster Center: Update winning cluster center as below

   $\mu_k = \mu_k + \alpha(x - \mu_k)$

Training RBF Hidden Layer Using K-means

- After deciding RBF centers, next task is to decide value of parameter sigma used in RBF.
- We can use following equation to determine value of sigma.

$$\sigma = \frac{d_{\max}}{\sqrt{2K}}, \text{ where } K \text{ is the number of centers and}$$

$$d_{\max} \text{ is the maximum distance between them}$$

- Alternatively we can use following equation to determine the value of sigma.

$$\sigma = \sqrt{\frac{1}{P}\sum_{k=1}^{P}(x - x_k)^2}, \text{ where } P \text{ is the number of nearest cluster centers}$$

Training RBF Hidden Layer Using K-means

Example

- Consider the XOR problem. Define two RBF centers and calculate the value of sigma for the RBF.

Solution

XOR function is given below

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Let us consider (0,0) and (1,1) are two RBF centers and $\varphi_1(x)$ and $\varphi_2(x)$ are two RBFs.

Training RBF Hidden Layer Using K-means

We know that

$$\sigma = \frac{d_{\max}}{\sqrt{2K}}$$

$$\Rightarrow \sigma_1 = \frac{d_{\max}}{\sqrt{2K}} = \frac{\sqrt{2}}{\sqrt{4}} = \frac{1}{\sqrt{2}} \quad \text{and} \quad \sigma_2 = \frac{d_{\max}}{\sqrt{2K}} = \frac{\sqrt{2}}{\sqrt{4}} = \frac{1}{\sqrt{2}}$$

Thus,

$$\varphi_1(x) = \exp\left(-\frac{\|x - c_1\|^2}{2\sigma^2}\right) = \exp\left(-\|x - c_1\|^2\right)$$

$$\varphi_2(x) = \exp\left(-\frac{\|x - c_2\|^2}{2\sigma^2}\right) = \exp\left(-\|x - c_2\|^2\right)$$

Training RBF Hidden Layer Using K-means

Alternatively, We know that $\sigma = \sqrt{\frac{1}{P}\sum_{k=1}^{P}(x - x_k)^2}$

Let us choose P=2, thus we get

$$\sigma_1 = \sqrt{\frac{1}{2}\{(0-1)^2 + (0-0)^2\} + \{(0-0)^2 + (0-1)^2\}} = 1$$

$$\sigma_2 = \sqrt{\frac{1}{2}\{(1-1)^2 + (1-0)^2\} + \{(1-0)^2 + (1-1)^2\}} = 1$$

Thus,

$$\varphi_1(x) = \exp\left(-\frac{\|x - c_1\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\|x - c_1\|^2}{2}\right)$$

$$\varphi_2(x) = \exp\left(-\frac{\|x - c_2\|^2}{2\sigma^2}\right) = \exp\left(-\frac{\|x - c_2\|^2}{2}\right)$$

LMS Estimation of Weight Vector

- Let us assume that N is the number of input samples in the training set, M is the dimension of hidden layer and C is the number of classes (i.e. number of output nodes)
- For notational simplicity assume that $\varphi_j(x_i) = \varphi_{ji}$
- For the properly adjusted weights, following relation must hold

$$\begin{bmatrix} \varphi_{11} & \varphi_{21} \cdots \varphi_{M1} \\ \varphi_{12} & \varphi_{22} \cdots \varphi_{M2} \\ \vdots & \\ \varphi_{1N} & \varphi_{2N} \cdots \varphi_{MN} \end{bmatrix} \begin{bmatrix} w_{k1} \\ w_{k2} \\ \vdots \\ w_{kM} \end{bmatrix} = \begin{bmatrix} d_{k1} \\ d_{k2} \\ \vdots \\ d_{kN} \end{bmatrix}$$

where d_{kj} is the desired value of summing function of k^{th} output neuron for j^{th} input

LMS Estimation of Weight Vector

- We know that

$$d_{kj} = 1,\ if\ x_j \in H_k\ and$$

$$d_{kj} = -1,\ if\ x_j \notin H_k$$

- Above equation can be written in simplified form as below:

$$\varphi_{ji} w_{kj} = d_{ki},\ i = 1.....N, j = 1.....M, k = 1.....C$$

- However, if the network is not properly trained, above equation is not satisfied. In this case the error term is given as below

$$e = \varphi w_k - d_k$$

LMS Estimation of Weight Vector

- Now, the energy function can be written as

$$E(w_k) = \frac{1}{2}\sum e^2 = \frac{1}{2}\sum (\varphi w_k - d_k)^2$$

- Now, Using gradient descent algorithm

$$\nabla E = \varphi^T (\varphi w_k - d_k)$$

- Equating above equation with zero and rearranging the terms, we get

$$w_k = (\varphi^T \varphi)^{-1} \varphi^T d_k$$

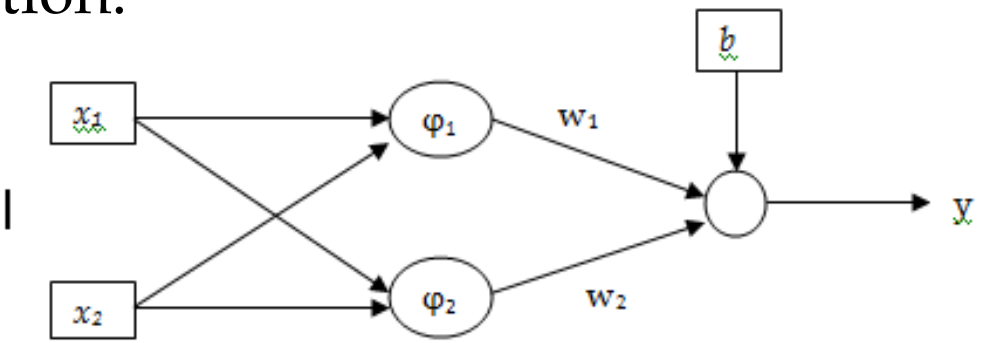
LMS Estimation of Weight Vector

Example

- Devise RBF neural network for XOR function using two cluster centers.

Solution

Since we are assuming two cluster centers, structure of RBF network for XOR function will be like below. Assume that Activation function of output layer is step function.



LMS Estimation of Weight Vector

- Assume that (0,0) and (1,1) are two RBF centers. Thus, we have following values of sigma for two RBFs.

$$\sigma = \frac{d_{\max}}{\sqrt{2K}}$$

$$\Rightarrow \sigma_1 = \frac{d_{\max}}{\sqrt{2K}} = \frac{\sqrt{2}}{\sqrt{4}} = \frac{1}{\sqrt{2}} \quad \text{and} \quad \sigma_2 = \frac{d_{\max}}{\sqrt{2K}} = \frac{\sqrt{2}}{\sqrt{4}} = \frac{1}{\sqrt{2}}$$

- Thus we have,

$$\varphi_1(x) = \exp\left(-\frac{\|x - c_1\|^2}{2\sigma^2}\right) = \exp\left(-\|x - c_1\|^2\right)$$

$$\varphi_2(x) = \exp\left(-\frac{\|x - c_2\|^2}{2\sigma^2}\right) = \exp\left(-\|x - c_2\|^2\right)$$

LMS Estimation of Weight Vector

- Now, we get following output from hidden layer of RBF network.

x_1	x_2	φ_1	φ_2
0	0	1.0	0.1353
0	1	0.3678	0.3678
1	0	0.3678	0.3678
1	1	0.1353	1.0

- From LMS estimation method we have

$$w_k = (\varphi^T \varphi)^{-1} \varphi^T d_k$$

LMS Estimation of Weight Vector

$$w = \left(\begin{bmatrix} 1.0 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1.0 & 1 \end{bmatrix}^T \begin{bmatrix} 1.0 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1.0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1.0 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1.0 & 1 \end{bmatrix}^T \begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} -5.02 \\ -5.02 \\ 4.68 \end{bmatrix}$$

Recursive Least Square (RLS) Estimation of Weight Vector

- LMS algorithm for estimation of weight vector of RBF network requires computation of matrix inverse operation. This is computationally expensive operation.
- Hence, Recursive Least Square (RLS) is alternative to LMS algorithm for computation of RBF weight vector.

Recursive Least Square (RLS) Estimation of Weight Vector

Algorithm

1. Repeat Steps 2 to 5 for N times, where N is number of training samples.

2. $$p(n) = p(n-1) - \frac{p(n-1)\varphi(n)\varphi^T(n)p(n-1)}{1+\varphi^T(n)p(n-1)\varphi(n)}$$

3. $$g(n) = p(n)\varphi(n)$$

4. $$\alpha(n) = d(n) - w^T(n-1)\varphi(n)$$

5. $$w^T(n) = w^T(n-1) + g(n)\alpha(n)$$

Recursive Least Square (RLS) Estimation of Weight Vector

- RLS algorithm is recursive algorithm and hence we require initial conditions.
- Initializations for RLS algorithm is given below:

$$w^T(0) = \vec{0}$$

$$p(0) = \lambda^{-1} I, \textit{ where } I \textit{ is identity matrix}$$

Hybrid Learning Procedure for RBF Networks

- Learning in RBF network use K-means and LMS of RLS algorithm. This hybrid learning procedure employed in RBF networks is summarized below.
- Size of input layer is determined by dimensionality of training sample, let m_0 is the dimensionality of training sample.
- Size of input layer is determined by number of clusters, let K=m_1 is the number of clusters.
- Use K-means algorithm to determine cluster centers.

Hybrid Learning Procedure for RBF Networks

- Determine value of parameter σ (RBF width) using following formula

$$\sigma = \frac{d_{\max}}{\sqrt{2K}}, \text{ where } K \text{ is the number of centers and}$$

$$d_{\max} \text{ is the maximum distance between them}$$

- Determine the values of parameters (weights and bias) using LMS or RLS algorithm.

Kernel Functions

- Kernel function is a similarity function that takes two inputs and spits out how similar they are. Among the large variety of kernel functions Gaussian kernel functions are widely used. Gaussian kernel function is a function of the form

$$k(x) = a.\exp\left(-\frac{(x-b)^2}{2c^2}\right)$$

- In the above equation *a, b* and *c are constants*. The parameter *a* is the height of the curve's peak, *b* is the position of the center of the peak and *c* (the standard deviation) controls the width of the curve.

Kernel Regression

- The regression of data points that is based on kernel function is called kernel regression.
- Unlike linear, polynomial, or logistic regression in which the optimal parameter needs to be learnt, kernel regression is non-parametric, meaning that it calculates the target y by performing computations directly on the input x.
- Given the data points (x_i, y_i) Kernel Regression first constructs a kernel k for each data point x_i.

Kernel Regression

- Then for a given new input x, it computes a similarity score with each x_i (given by x_i-x) using the kernel.
- Then calculate weight w_i for each input as below. Weight represents the importance of that kernel (and corresponding label y_i) in predicting the target y.

$$w_i = k_i / \sum_{i=1}^{n} k_i$$

- The prediction is then obtained by multiplying the weight vector w= [w_1, w_2, …, w_n] with the label vector y= [y_1, y_2, …, y_n].

Kernel Regression

Example

Consider the following data points. Use kernel regression to estimate y(50).

x	11	22	33	44	56	67	78	89	100
y	2337	2750	2301	2500	2100	1100	1000	1642	1932

Assume following Kernel function: $k(x) = \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)$

and assume width of kernel function is 10

Kernel Regression

Example

Solution

Given Kernel function $k(x) = \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)$

Width of the Kernel $\sigma = 10$ and x=50

Now, compute kernel function value, and weight for each given data point

x_i	11	22	33	44	56	67	78	89	100
y_i	2337	2750	2301	2500	2100	1100	1000	1642	1932
k_i	0.000498	0.0198	0.236	0.835	0.835	0.336	0.0198	0.000498	0.0000037
w_i	0.00023	0.0091	0.108	0.383	0.383	0.108	0.0091	0.00023	0.0000017

Kernel Regression

Finally, Compute value of y(50) as below:

$$y(50) = y_1 w_1 + y_2 w_2 + y_3 w_3 + y_4 w_4 + y_5 w_5 + y_6 w_6 + y_7 w_7 + y_8 w_8 + y_9 w_9$$
$$= 2165.325$$

Relation of Kernel regression with RBF Networks

- Kernel Regression function is given by

$$F(x) = \sum_{i=1}^{N} y_i w_i, \ \ where\ w_i = \frac{K_i}{\sum_{i=1}^{N} K_i}$$

- In the above equation K_i is kernel function value which is given by

$$K_i = \exp\left(-\frac{(x - x_i)^2}{2\sigma^2} \right)$$

- Combining above two equations, we get

$$F(x) = \frac{\sum_{i=1}^{N} y_i \exp\left(-\frac{(x - x_i)^2}{2\sigma^2} \right)}{\sum_{i=1}^{N} \exp\left(-\frac{(x - x_i)^2}{2\sigma^2} \right)} \qquad (1)$$

Relation of Kernel regression with RBF Networks

- Normalized Radial Basis function for the given N data points is

$$\varphi(x) = \frac{\exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)}$$

- Thus, input/output mapping of Normalized RBF network is given by

$$f(x) = w_i \varphi_i(x) = \frac{\sum_{i=1}^{N} w_i \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)} \qquad (2)$$

Relation of Kernel regression with RBF Networks

- Equations 1 and 2 are similar. Thus, we can say that kernel regression is equivalent to Normalized RBF networks that contains one hidden layer neuron for each of the N given data points.
- However, usually RBF network contains K neurons in hidden layer such that K<=N.