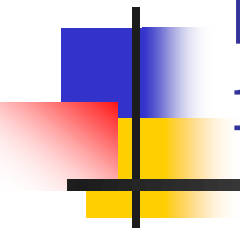
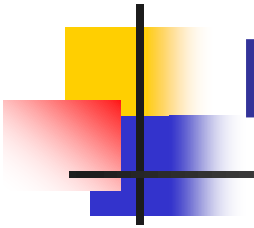


Machine Learning

12. week

- Multi Layer Perceptron (MLP) Network
- Radial Basis Function (RBF) Network



Multi Layer Perceptron Network

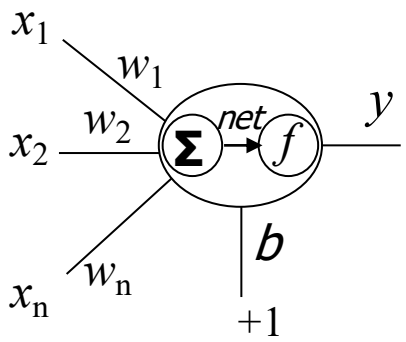
Multi layer perceptron network (MLP) is proposed after Adaline and Perceptron methods have failed producing non-linear solutions to the problem.

MLP is improved in terms of both architectural and training algorithm.

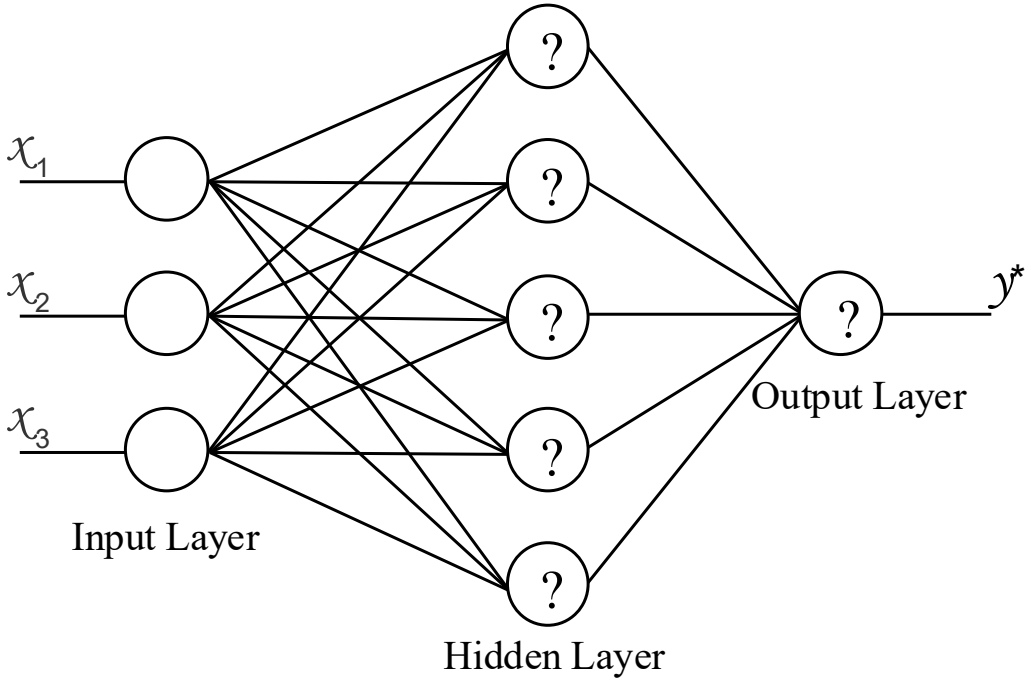
MLP is formed by many neurons each having non-linear activation function.

MLP uses back-propagation learning algorithm in addition to the advantages of Perceptron and Adaline methods.

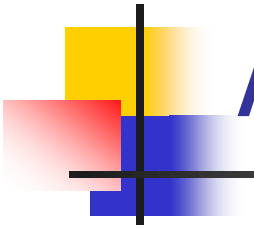
MLP Structure



General structure of a neuron used in MLP.



General structure of an MLP network.



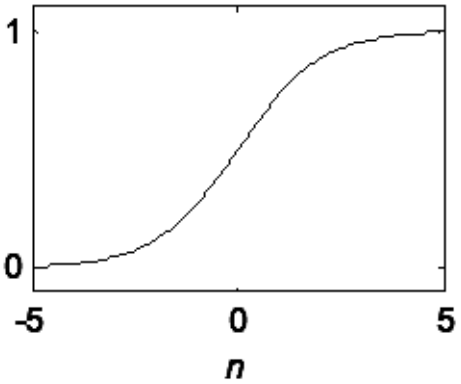
Activation Function

Activation function is basicly a linear or non-linear function that transforms a variable from one dimension into another dimension. Ease of its derivability increases the training speed. Sigmoid, hyperbolic tangent and step functions are the most frequently used activation functions.

Activation Function

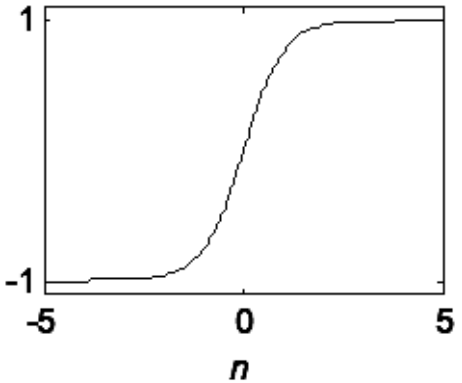
Sigmoid

$$y = \frac{1}{1 + e^{-net}}$$



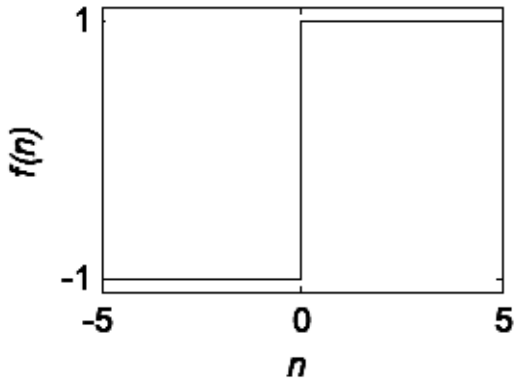
Hyperbolic tangent

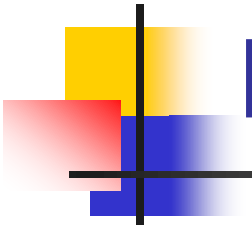
$$y = \frac{1 - e^{-2net}}{1 + e^{2net}}$$



Step function

$$y = \begin{cases} 1 & net >= 0 \\ -1 & net < 0 \end{cases}$$





Backpropagation

All of the weight values are calculated by a method called gradient descent which minimizes the error function. Calculated error is adjusted by the derivation of the activation function while going back from the output end of the neuron to the input end. The applied error value to the derivation of the function is distributed to all of the weights proportional to neurons' input values.

Backpropagation

$$E = \frac{1}{2}\sum_j e_j^{\,2} = \frac{1}{2}\sum_j \left(d_j - y_j\right)^2$$

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \left[\frac{\partial E}{\partial y_j}\right]\left[\frac{\partial y_j}{\partial net_i}\right]\left[\frac{\partial net_i}{\partial w_{ij}}\right]$$

$$= -\eta\left[-e_j\right]\left[\frac{\partial y_j}{\partial net_i}\right]\left[x_i\right]$$

If sigmoid is used as activation function;

$$\Delta w_{ij} = -\eta\left[-e_j\right]\left[\left(1-y_j\right)y_j\right]\left[x_i\right] = \eta\, e_j\left(1-y_j\right)y_j x_i$$

Backpropagation

Prove the statement below for $y = \frac{1 - e^{-2net}}{1 + e^{2net}}$

$$-\eta \frac{\partial E}{\partial w_{ij}} = \eta e_j \left(1 - y_j^2\right) x_i$$

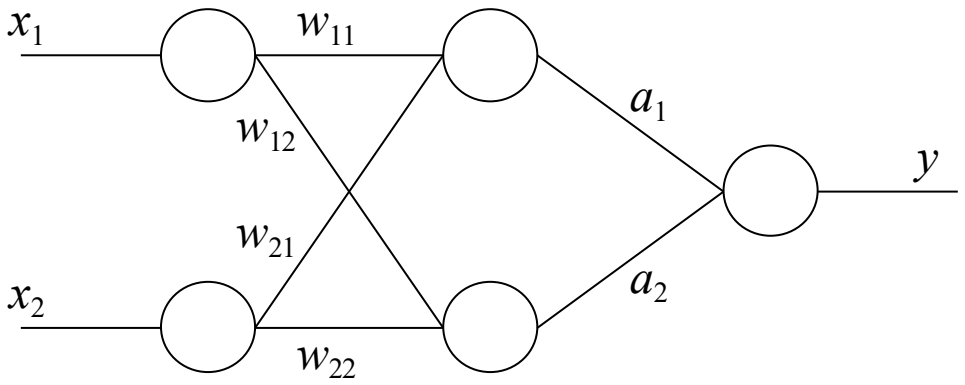
Momentum Effect

While system converges through the optimal result step by step using gradient descent, sometimes it gets stuck to one of the local minimum points and the global target cannot be reached. To prevent this, a portion (α) of value change occured in previous step is used as a momentum effect.

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \, \Delta w_{ij}(t)$$

Example

Lets solve XOR problem for an MLP network which has two inputs and one output and has two neurons on its single hidden layer.



$\eta = 0.5$

$\alpha = 0.1$

$w_{11} = 1$

$w_{12} = -2$

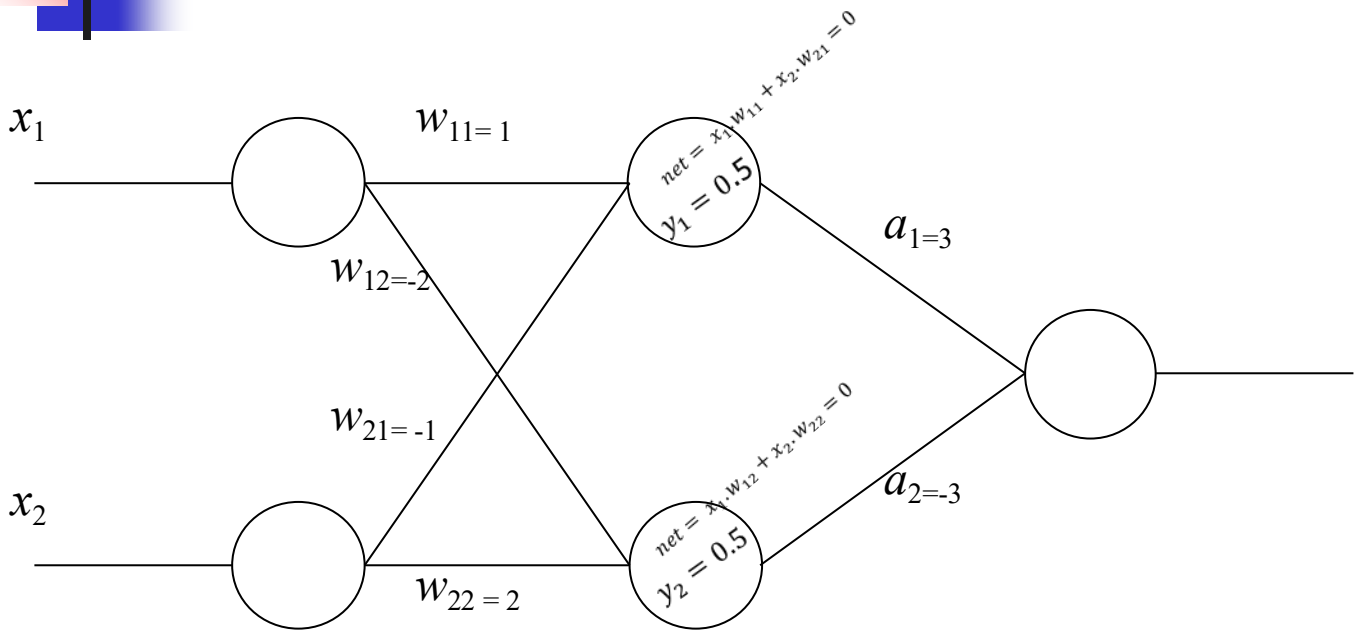
$w_{21} = -1$

$w_{22} = 2$

$a_1 = 3$

$a_2 = -3$

Example



$\eta = 0.5$

$\alpha = 0.1$

$w_{11}= 1$

$w_{12}= -2$

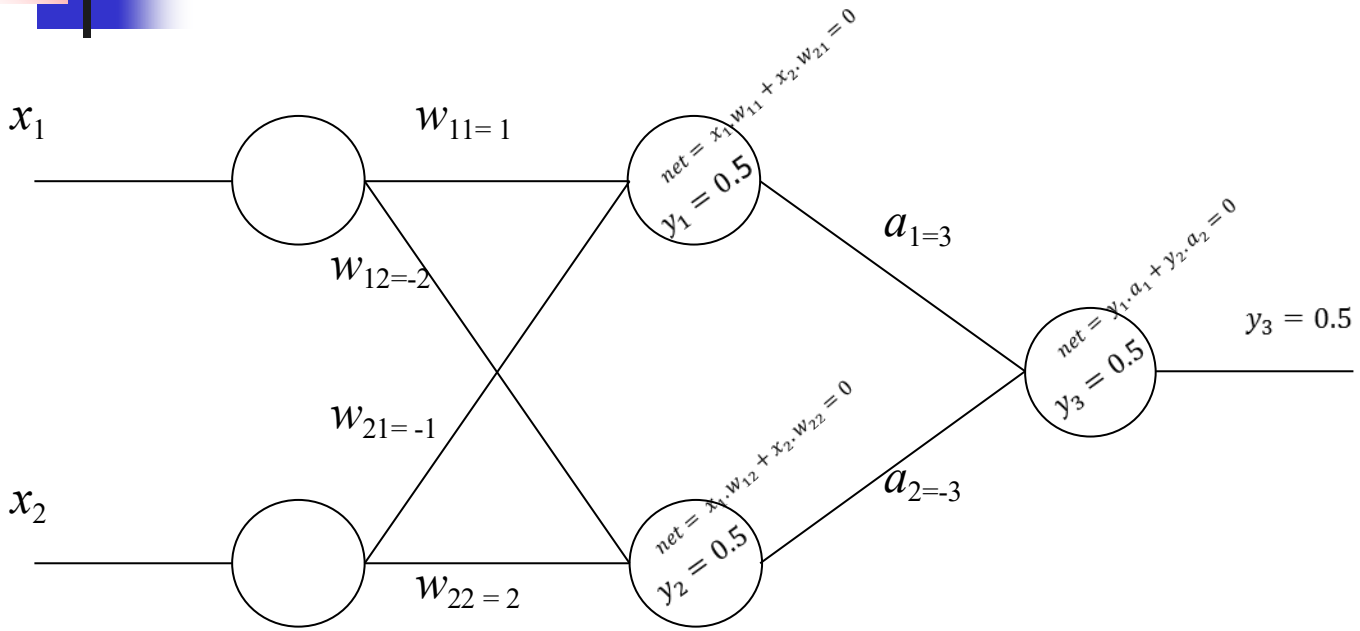
$w_{21}= -1$

$w_{22}= 2$

$a_1 = 3$

$a_2 = -3$

Example



$\eta = 0.5$

$\alpha = 0.1$

$w_{11}= 1$

$w_{12}= -2$

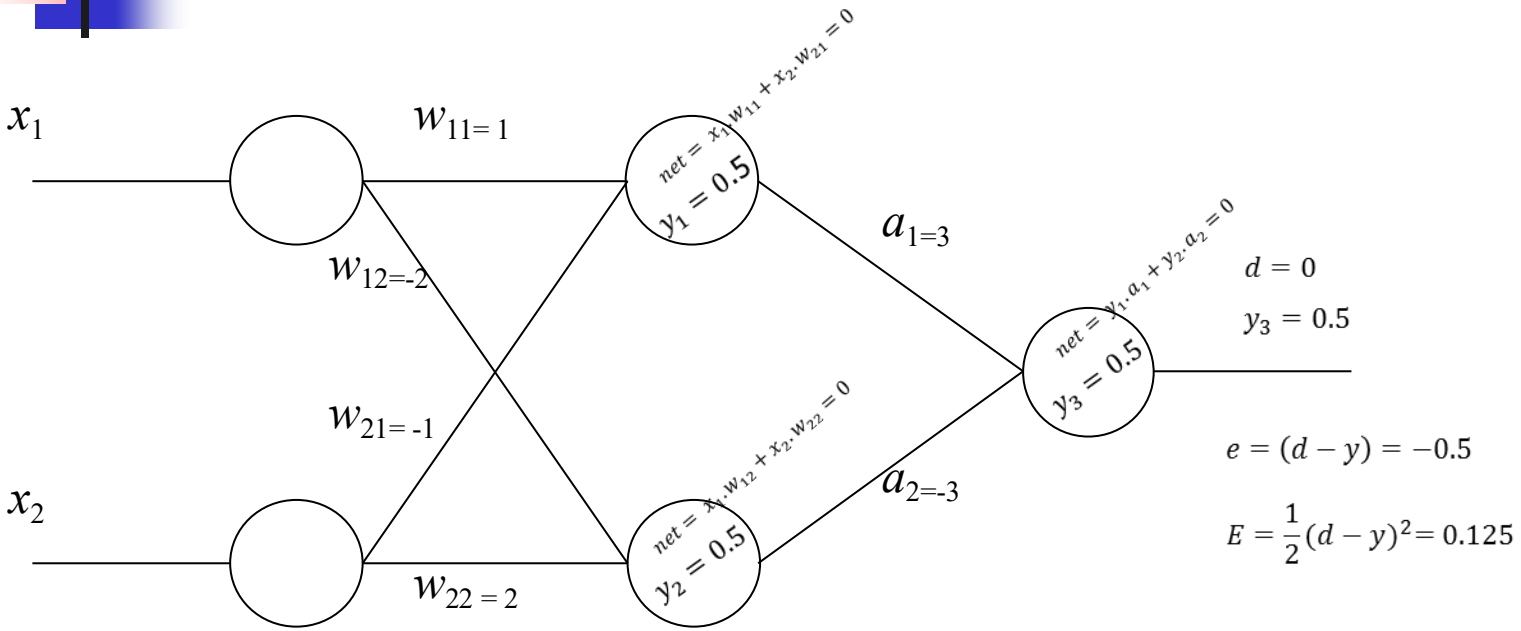
$w_{21}= -1$

$w_{22}= 2$

$a_1 = 3$

$a_2 = -3$

Example



$\eta = 0.5$

$\alpha = 0.1$

$w_{11}= 1$

$w_{12}= -2$

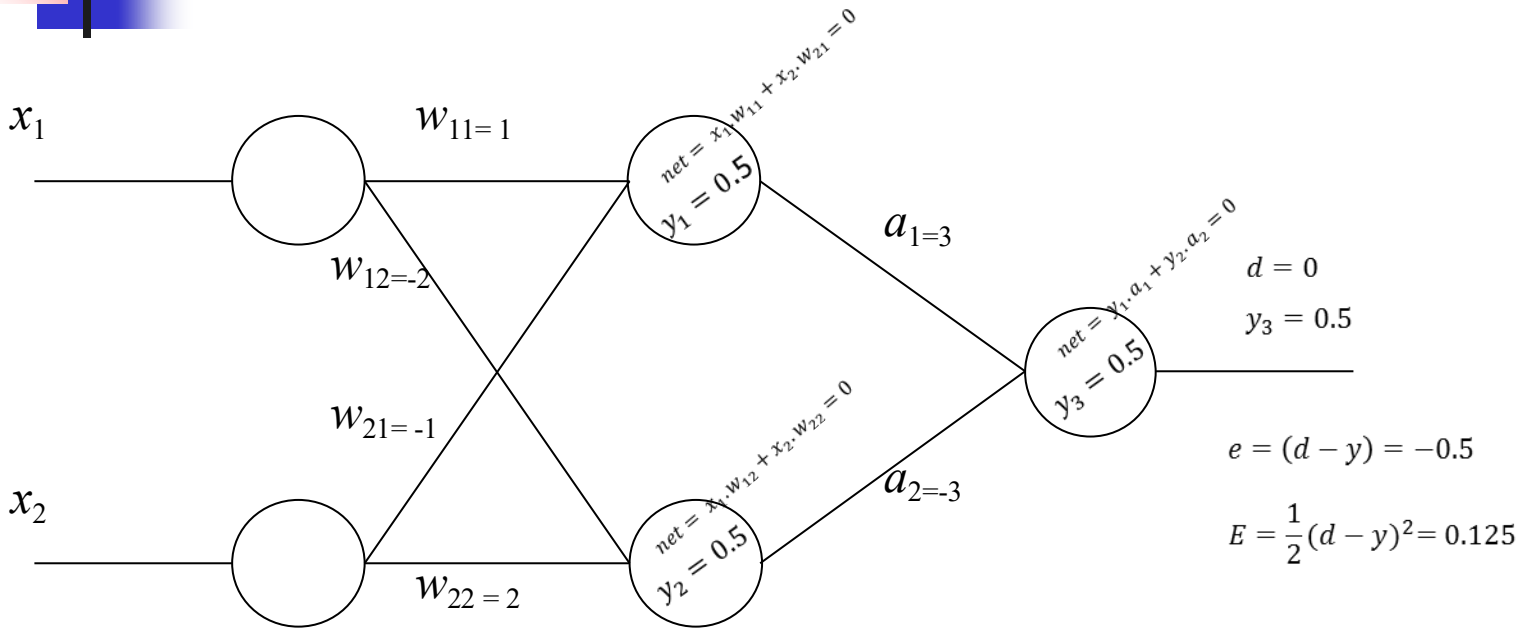
$w_{21}= -1$

$w_{22}= 2$

$a_1 = 3$

$a_2 = -3$

Example



$\eta = 0.5$

$\alpha = 0.1$

$w_{11}= 1$

$w_{12}= -2$

$w_{21}= -1$

$w_{22}= 2$

$a_1 = 3$

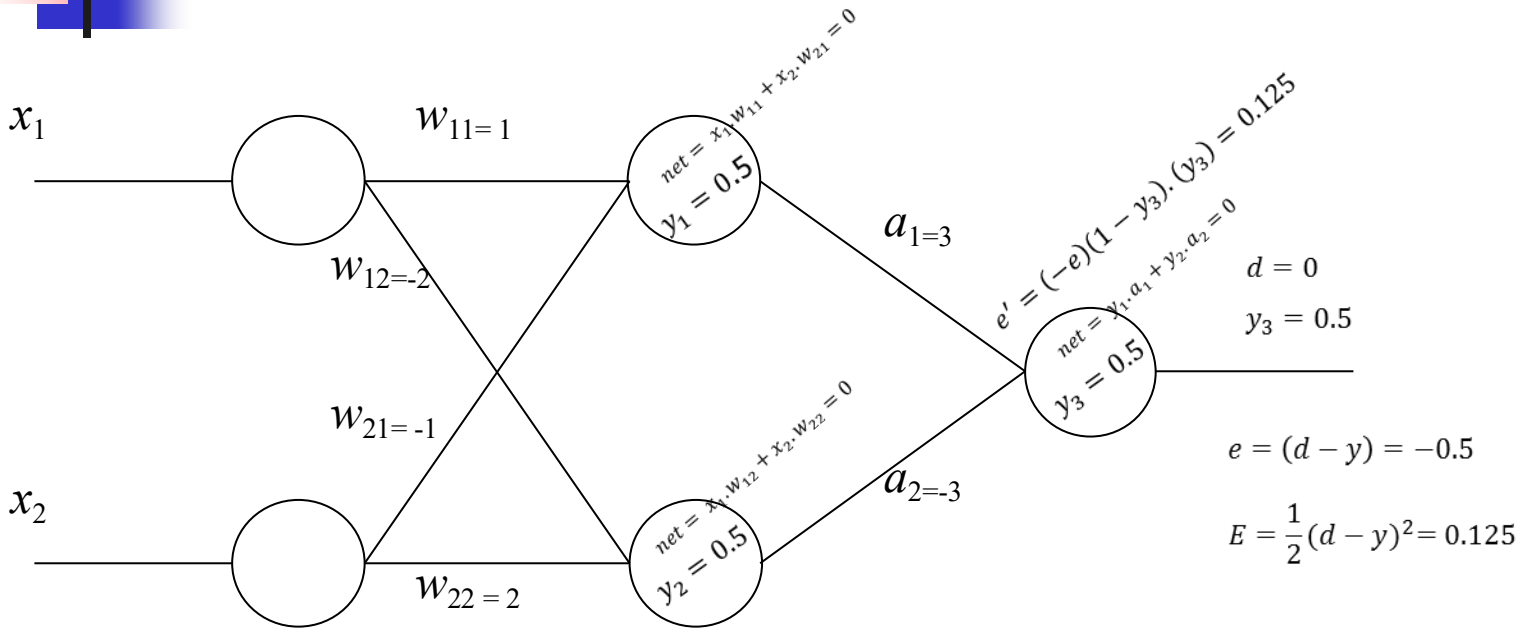
$a_2 = -3$

$$\Delta w_{ij} = -\eta[-e_j][(1-y_j)y_j][x_i] = \eta e_j(1-y_j)y_j x_i$$

$$\Delta a_1 = (-\eta).(-e).(1-y_3).(y_3).(y_1) = -0.03$$

$$a_{1_{new}} = 3 + (-0.03) + \alpha.\Delta a_{1_{old}} = 2.97$$

Example



$\eta = 0.5$

$\alpha = 0.1$

$w_{11}= 1$

$w_{12}= -2$

$w_{21}= -1$

$w_{22}= 2$

$a_1 = 3$

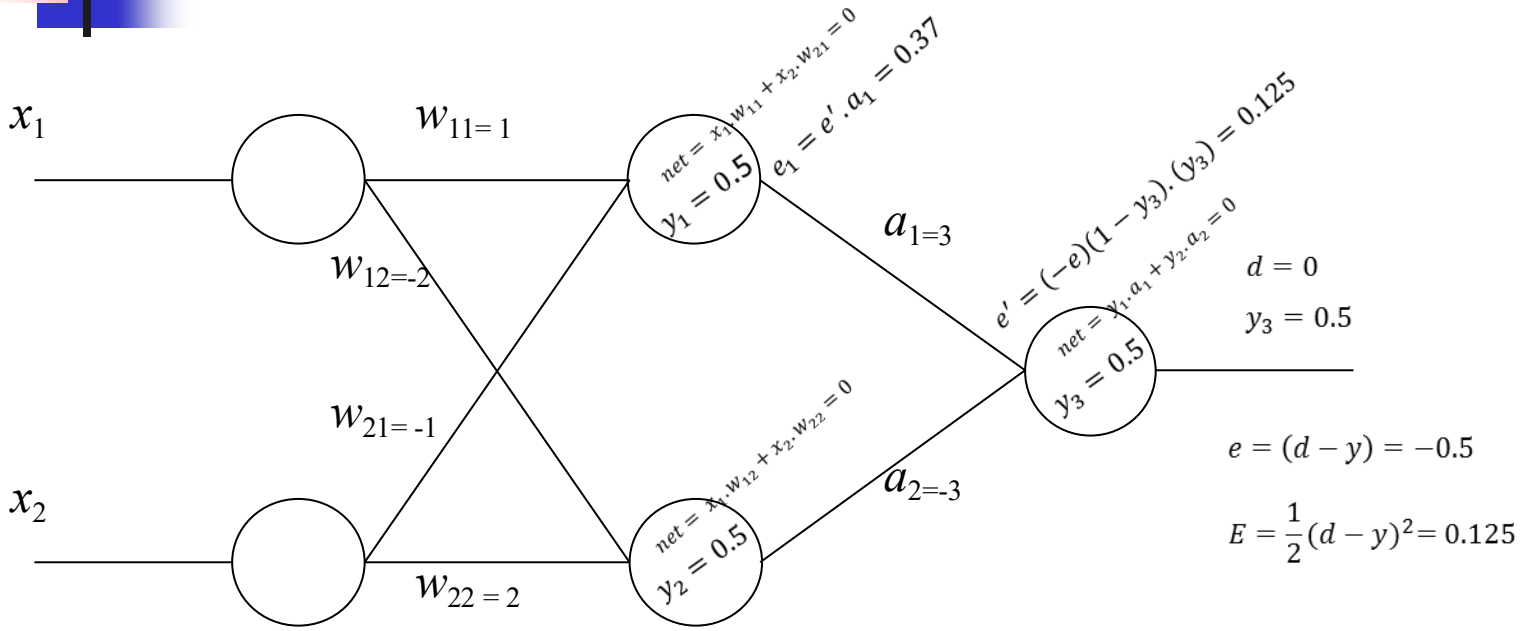
$a_2 = -3$

$$\Delta w_{ij} = -\eta[-e_j][(1-y_j)y_j][x_i] = \eta\, e_j(1-y_j)y_j x_i$$

$$\Delta a_1 = (-\eta).(-e).(1-y_3).(y_3).(y_1) = -0.03$$

$$a_{1_{new}} = 3 + (-0.03) + \alpha.\Delta a_{1_{old}} = 2.97$$

Example



$\eta = 0.5$

$\alpha = 0.1$

$w_{11}= 1$

$w_{12}= -2$

$w_{21}= -1$

$w_{22}= 2$

$a_1 = 3$

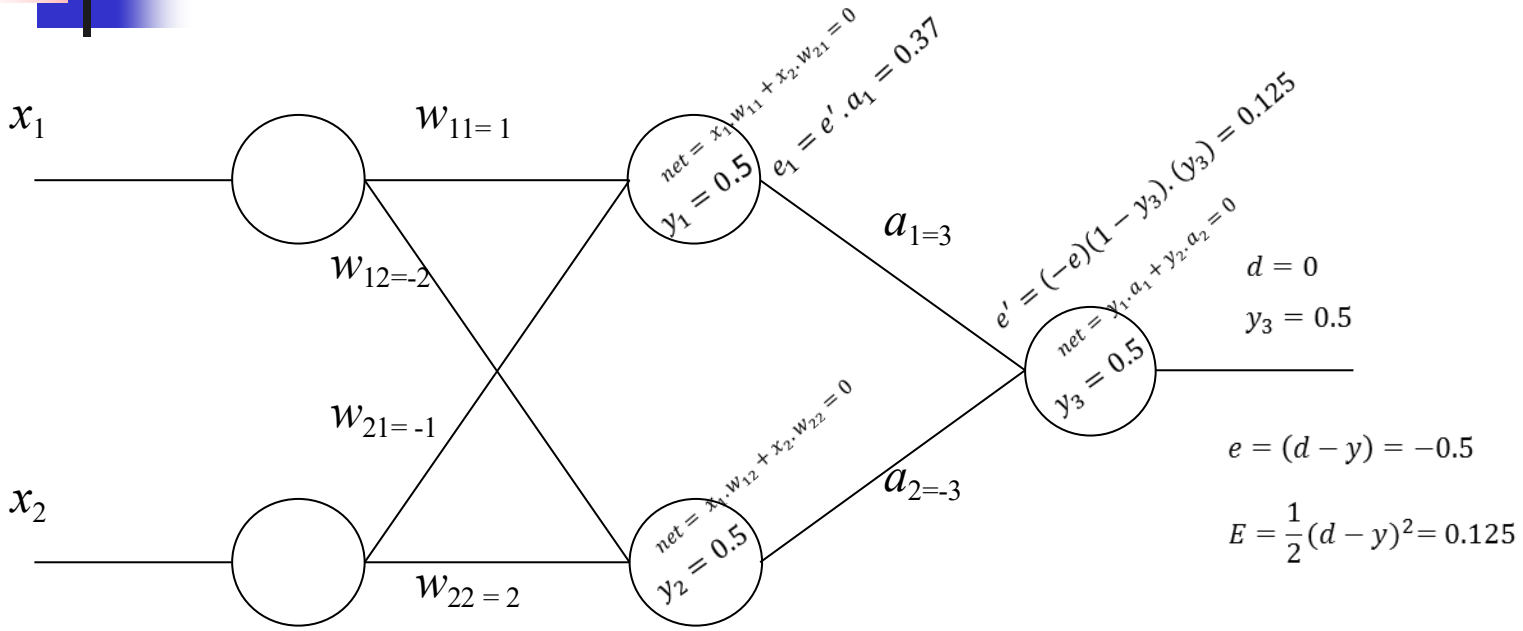
$a_2 = -3$

$$\Delta w_{ij} = -\eta[-e_j][(1-y_j)y_j][x_i] = \eta e_j(1-y_j)y_j x_i$$

$$\Delta a_1 = (-\eta).(-e).(1-y_3).(y_3).(y_1) = -0.03$$

$$a_{1_{new}} = 3 + (-0.03) + \alpha.\Delta a_{1_{old}} = 2.97$$

Example



$\eta = 0.5$

$\alpha = 0.1$

$w_{11}= 1$

$w_{12}= -2$

$w_{21}= -1$

$w_{22}= 2$

$a_1 = 3$

$a_2 = -3$

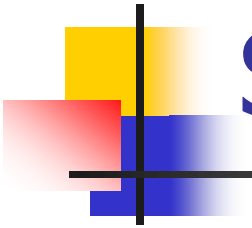
$$\Delta w_{ij} = -\eta\left[-e_j\right]\left[(1-y_j)y_j\right]\left[x_i\right] = \eta\, e_j(1-y_j)y_j x_i$$

$$\Delta a_1 = (-\eta).(-e).(1-y_3).(y_3).(y_1) = -0.03$$

$$a_{1_{new}} = 3 + (-0.03) + \alpha.\Delta a_{1_{old}} = 2.97$$

$$\Delta w_{11} = (-\eta).(-e_1).(1-y_1).(y_1).(x_1) = 0.04$$

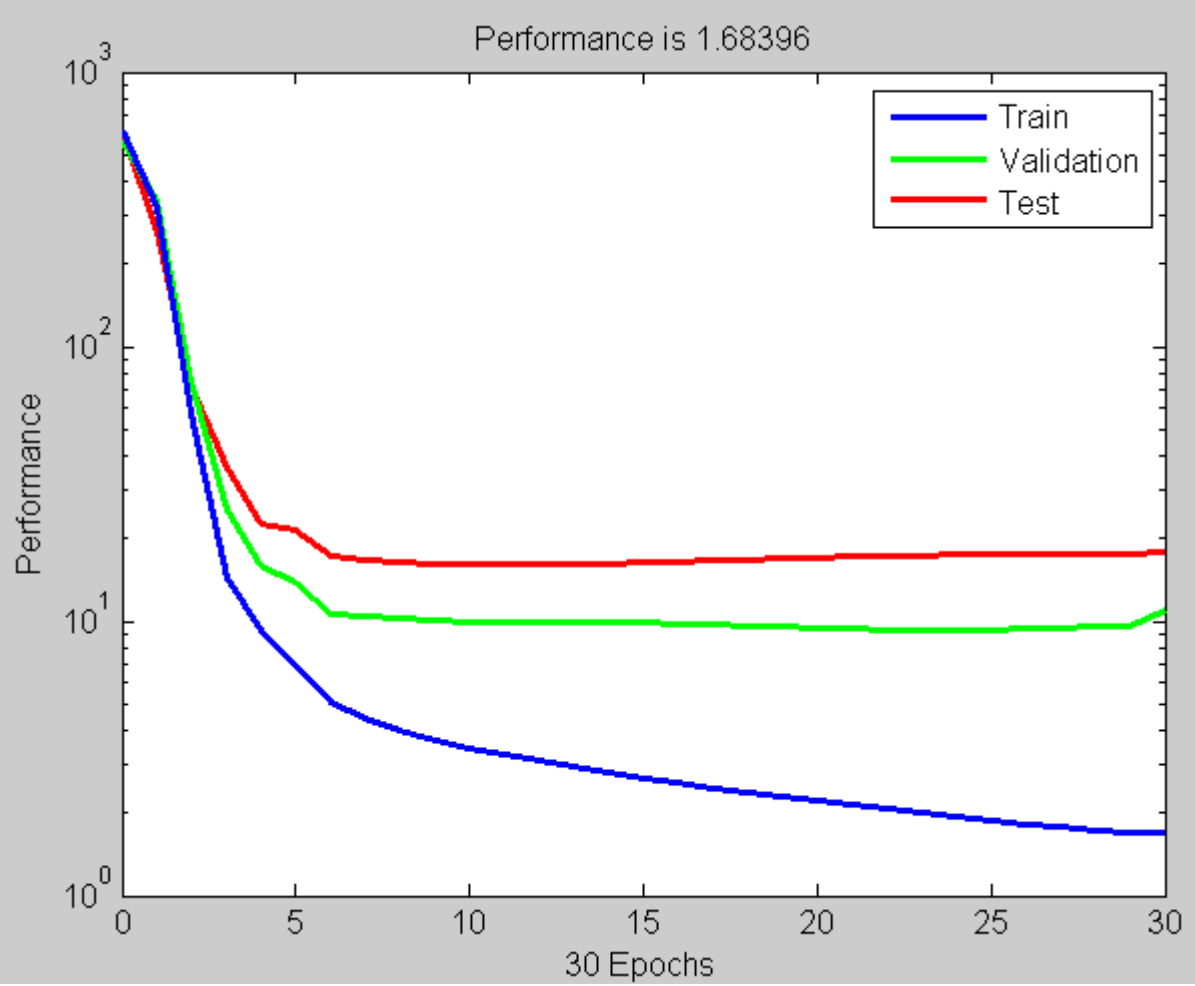
$$w_{11_{new}} = 1 + (0.04) + \alpha.\Delta w_{11_{old}} = 1.04$$

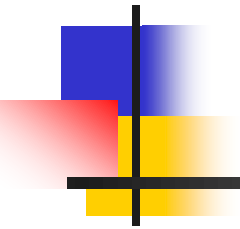


Stopping Criteria

Without a stopping criteria to determine when to stop the training, overfitting occurs. The training set, which is a subset of the entire set, is divided into two and while one part of it is used for updating the weight values, the other part is used for validation to calculate the success of the training. If success drops rapidly, training is stopped.

Stopping Criteria



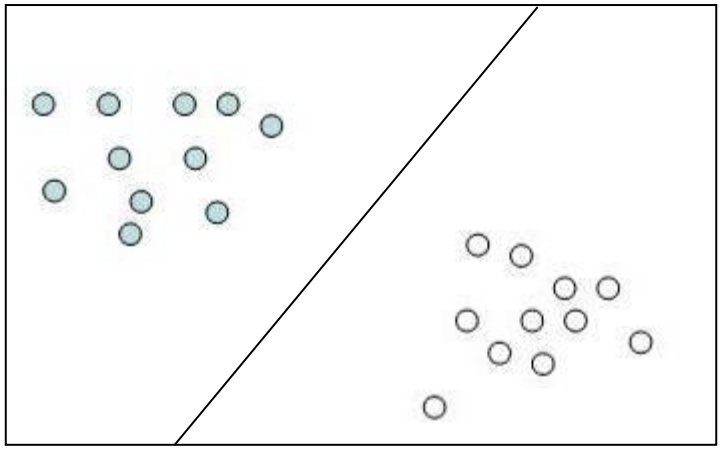


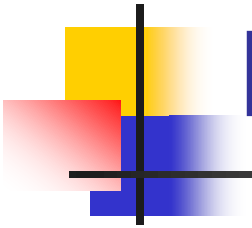
- Mapping Concept
- Radial Basis Functions (RBF)
- RBF Networks

Mapping

It is probably the best scenario for the classification of two dataset is to separate them linearly. As you see in the below figure, two different dataset are classified linearly.

But this scenario is not always possible in real life datasets.





Mapping

All of the proposed machine learning methods have been developed using linear classification.

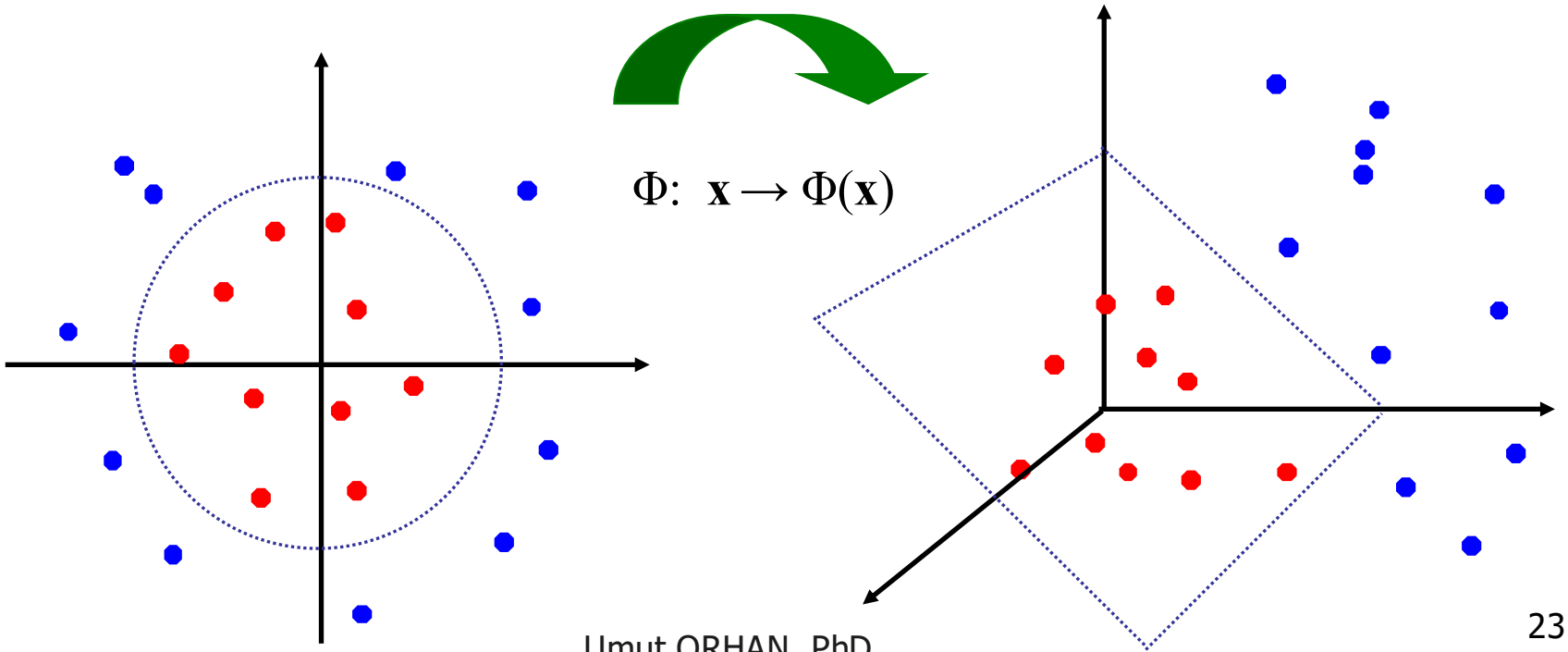
So they can not find solution for the non linear dataset.

Instead of solving datasets in their own space, data is moved to a new space where they can be separated linearly.

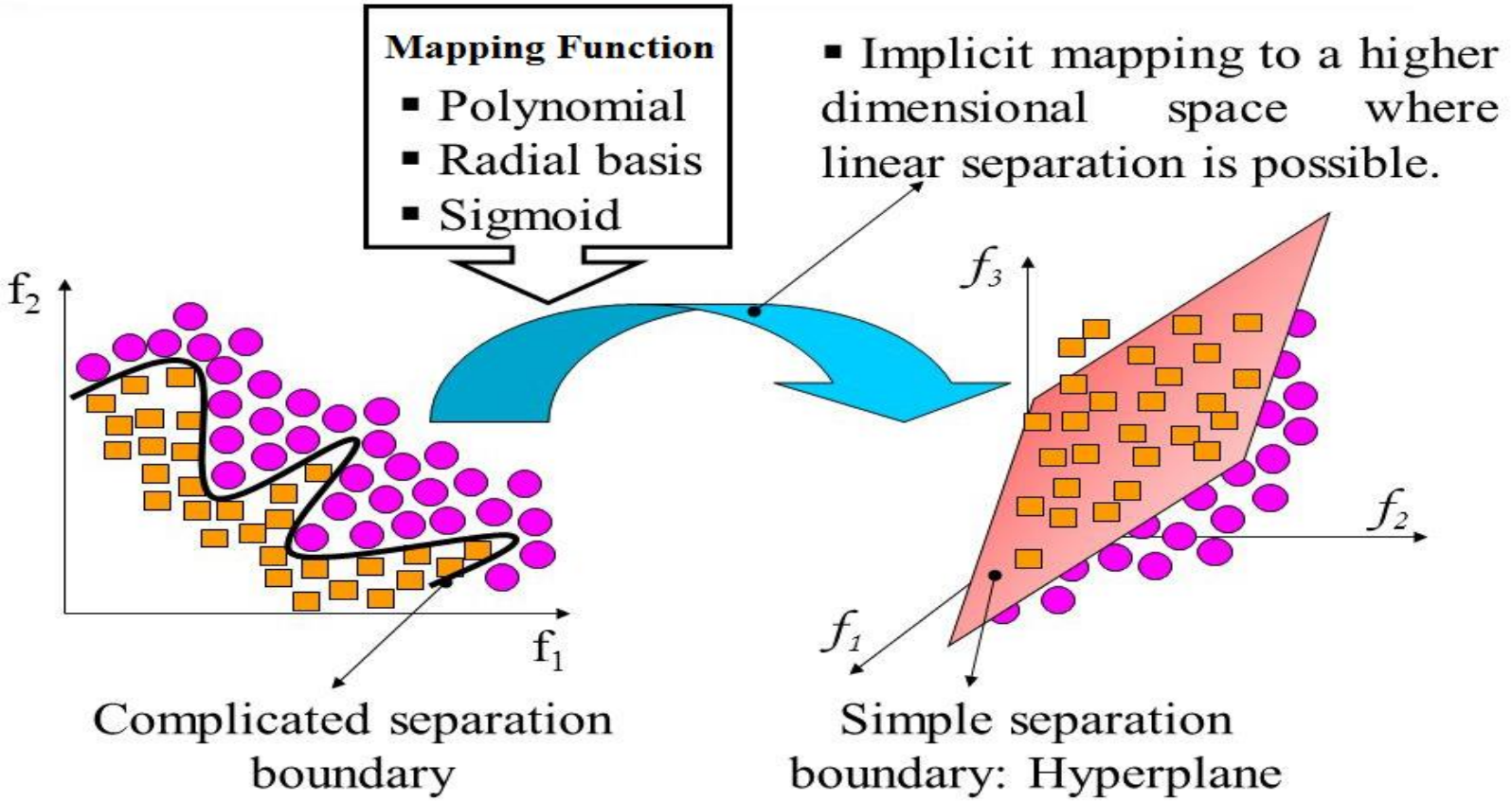
This process is called Mapping.

Mapping

Generally a Φ function is used to convert x into a linear seperatable status.



Mapping


Mapping Function
▪ Polynomial
▪ Radial basis
▪ Sigmoid
▪ Implicit mapping to a higher dimensional space where linear separation is possible.
f2
f1
f3
f2
f1
Complicated separation boundary
Simple separation boundary: Hyperplane

Mapping with RBF

Φ Function RBF (Radial Basis Function) is given equation below. Where (c_j) is the center points which represents data, (x_i) is the data points.

Φ Function is the exponantial affect of the distance between (c_j) and (x_i)

$$\Phi(x) = \exp\left(-\frac{\|c_j - x_i\|^2}{2r^2}\right)$$

RBF Networks

Neurons in hidden layer are calculated as in below equation, where x_i is the data point from input with no weight, c_j is the prototype hidden in the neuron. Simply it is a Euclidian distance.

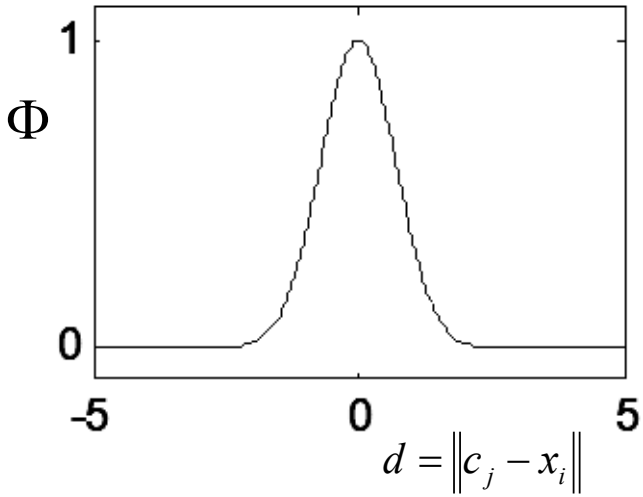
$$d = \left\| c_j - x_i \right\|$$

After that this distance is evaluated in Radial Basis Function.

RBF Networks

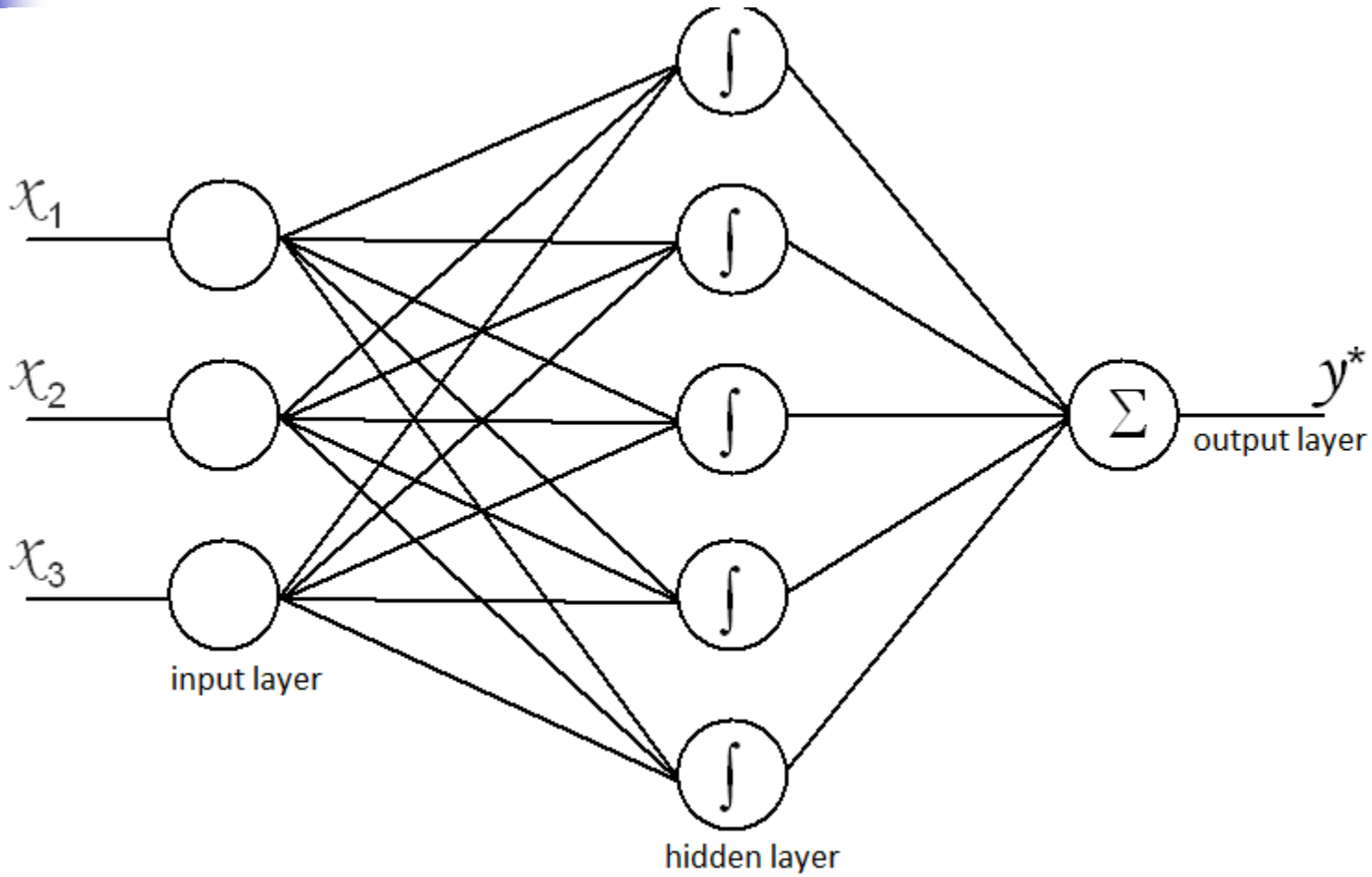
Widely used radial basis function equation and graphical relation of distance-function output is shown below.

$$\Phi(x) = \exp\left(-\frac{\|c_j - x_i\|^2}{2r^2}\right)$$



r value represents the diameter of prototype.

RBF Networks Architecture



Estimation in RBF Networks

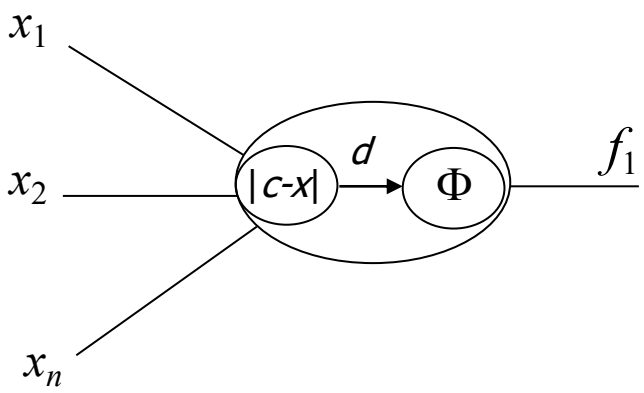
y represents target class, and it can be calculated with following equation where *x* represents input.

$$y_j = \sum_{1 \le i \le K} w_i \, \Phi(x_j)$$

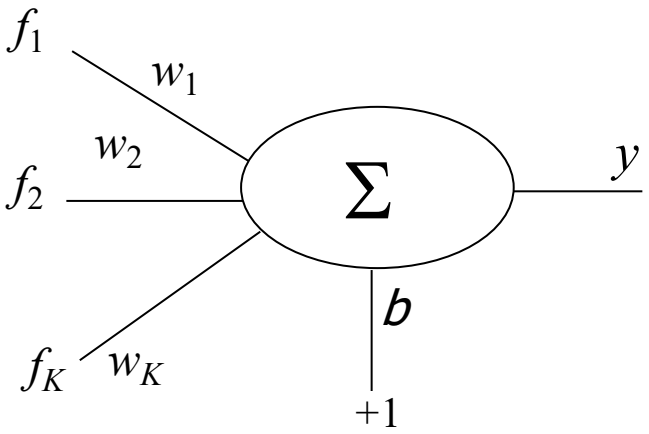
K value shows number of neuron in hidden layer.

Structure of RBF Networks

In below figures you can see neurons of hidden layer and output layer.



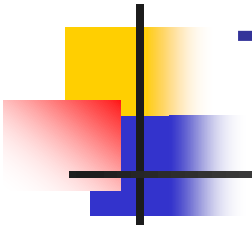
Hidden layer neuron



Output layer neuron

Features of RBF Networks

- RBF networks are feed forward networks with supervised learning.
- Generally a unique hidden layer is used in which each neuron contents a RBF function.
- Although training of RBF networks looks similar with back-propogation, they are trained faster than MLP.
- With advances of radial basis fuctions, they are less affected from unstable input problems.



Training in RBF Networks

There are four parameters in RBF networks which is unknown and need to be learned:

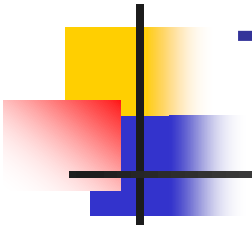
- Number of neurons in hidden layer
- Coordinate of each prototype neuron
- Prototype diameter of each neuron
- Output Weights

Training in RBF Networks

In most of the proposed methods, it is outlined that number of neurons in hidden layer variable needs to be researched by trying.

Since output weights are components of a linear equation, solution is easy and it depends on neuron outputs.

So, most important point in training depends on to know position and diameter values of prototypes.



Training in RBF Networks

Prototype position information in neurons can be found by unsupervised learning methods (like K-Means)

- Training can be faster by running supervised and unsupervised trainings paralelly.
- Accuracy of unsupervised learning classifiers are always lower than fully supervised learning methods.

Training in RBF Networks

Another proposal is to use the randomly selected samples as a prototype and not update position information. In this case, diameter (variants) data should be known and selected samples should represent every region in data space very well.

Training in RBF Networks

In order to find Prototype diameter (width of each RBF unit, also known as spread) K-nearest neighbour algorithm is used.

Root-mean squared distance between current cluster center and K nearest neighbours is calculated. And this is the value chosen for the unit width(r).

So if current cluster center is cj, the r value is;

$$rj = \sqrt{\sum_{i=1}^{k} \frac{(cj - ci)^2}{k}}$$

Training in RBF Networks

When we study structure of RBF networks, we can find some similarities with MLP network model. By means of linear structured similarity in output layer, it might be benefited from back propagation algorithm during training of RBF networks.

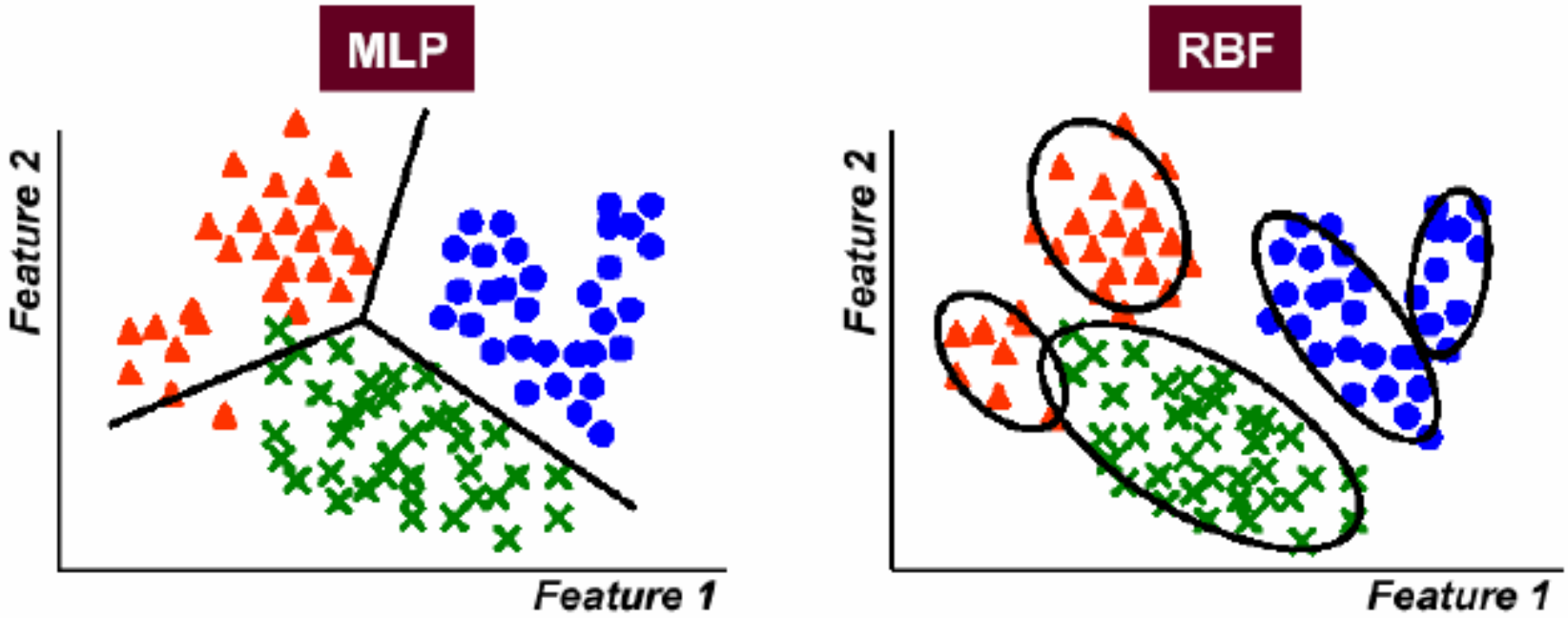
$$E = \frac{1}{2}\sum_{j} e_j^{\,2} \qquad \Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

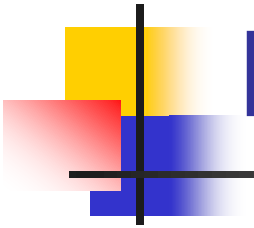
Training in RBF Networks

Output weights are updated with back propagation training and update quantities of function exits in hidden layer neurons can be predicted. This can be used indirectly in update of prototype diameter values and protoype coordinates.

$$\Delta c_i = -\eta \frac{\partial E}{\partial c_i} \qquad \Delta r_i = -\eta \frac{\partial E}{\partial r_i}$$

Classification with RBF Networks





Presentation Task

Please compare success of the results using linear regression for following mapping functions by using two dimensional non-linear artificial dataset.

- Sigmoid
- Hiperbolic Tangent
- RBF
- 3th degree polynomial