# Data Mining and Data Warehousing 9

## SVM & Back Propagation

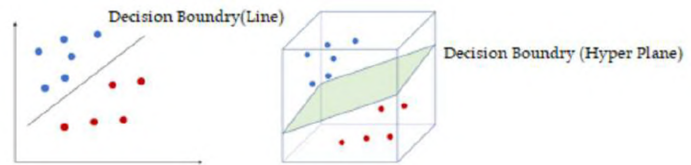
Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

# SVM (Support Vector Machine)



## SVM (Support Vector Machine)

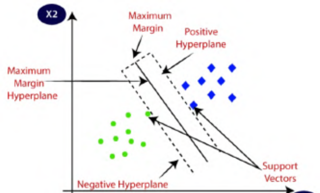
- SVM is one of the most popular supervised learning algorithm used for classification and regression problems
- The goal of the SVM algorithm is to create the best line or decision boundary (or hyperplane)that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in future
- SVM takes input data points and outputs the hyperplane (or a line in 2D) that best separates the data points into two classes. This line or hyperplane is the decision boundary: any data points that falls to one side of it is classified in one class and the data points that falls to the other of it is classified in another class



# SVM (Support Vector Machine)...

## SVM (Support Vector Machine)...

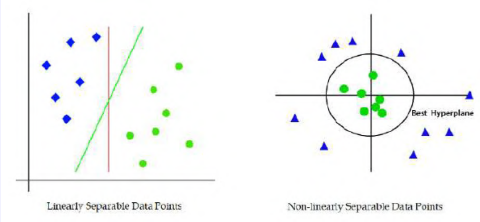
- Support vectors are data points that are closest to the hyperplane and influence the position and orientation of the hyperplane
- To separate the two classes of data points, SVM algorithm selects the optimal hyperplane by choosing hyperplane with largest margin. Such hyperplane is called maximum marginal hyperplane (MMH)
- Let H1 and H2 are planes that passes through support vectors and parallel to the hyperplane of decision boundary. Distance between H1 and the hyperplane should be equal to distance between H2 and the hyperplane. Margin is the distance between H1 and H2



# SVM (Support Vector Machine)...

## SVM Types

- **Linear SVM**: used for linearly separable data. If a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is called as linear SVM classifier
- **Non-Linear SVM**: used for non-linearly separated data. If a dataset can't be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as non-linear SVM classifier
- For linear data, use two dimensions; for non- linear data, add extra dimension through SVM kernel, which converts non-separable problem to separable problem

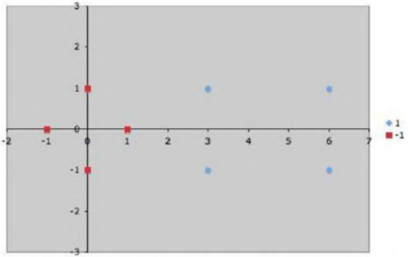


## SVM (Support Vector Machine)...

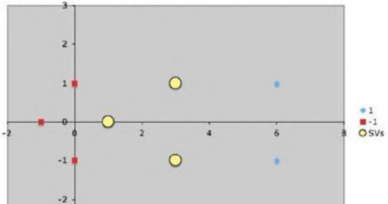
Q. Determine the equation of hyperplane that divides the data points into two classes

Positively labelled data points: (3,1), (3,-1), (6,1), (6,-1)

Negatively labelled data points: (1,0), (0,1), (0,-1), (-1,0)




Point (1,0) is nearest to negatively labelled data points and (3,1) & (3,-1) are nearest to positively labelled data points. So, support vectors are: S1=(1,0), S2=(3,1) and S3=(3,-1)



# SVM (Support Vector Machine)...



Here, we'll use vectors augmented with a 1 as a bias input, and for clarity we'll differentiate these with an over-tilde.

$$\tilde{s_1} = \begin{pmatrix}1\\0\\1\end{pmatrix}, \ \tilde{s_2} = \begin{pmatrix}3\\1\\1\end{pmatrix} \text{ and } \tilde{s_3} = \begin{pmatrix}3\\-1\\1\end{pmatrix}$$

Now, find 3 parameters α1, α2 and α3 from these 3 linear equations:

$$\alpha_1\tilde{s_1}.\tilde{s_1} + \alpha_2\tilde{s_2}.\tilde{s_1} + \alpha_3\tilde{s_3}.\tilde{s_1} = -1 \ \ (-ve\ class)$$
$$\alpha_1\tilde{s_1}.\tilde{s_2} + \alpha_2\tilde{s_2}.\tilde{s_2} + \alpha_3\tilde{s_3}.\tilde{s_2} = +1 \ \ (+ve\ class)$$
$$\alpha_1\tilde{s_1}.\tilde{s_3} + \alpha_2\tilde{s_2}.\tilde{s_3} + \alpha_3\tilde{s_3}.\tilde{s_3} = +1 \ \ (+ve\ class)$$

Let's substitute the values for $\tilde{s_1}$, $\tilde{s_2}$ and $\tilde{s_3}$ in the above equations.

$$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} + \alpha_2\begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} + \alpha_3\begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}1\\0\\1\end{pmatrix} = -1$$
$$\alpha_1(1+0+1) + \alpha_2(3+0+1) + \alpha_3(3+0+1) = -1$$
$$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1 \text{ -------------------- (i)}$$

$$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} + \alpha_2\begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} + \alpha_3\begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}3\\1\\1\end{pmatrix} = 1$$
$$\alpha_1(3+0+1) + \alpha_2(9+1+1) + \alpha_3(9-1+1) = 1$$
$$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = 1 \text{ -------------------- (ii)}$$

$$\alpha_1\begin{pmatrix}1\\0\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} + \alpha_2\begin{pmatrix}3\\1\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} + \alpha_3\begin{pmatrix}3\\-1\\1\end{pmatrix}\begin{pmatrix}3\\-1\\1\end{pmatrix} = +1$$
$$\alpha_1(3+0+1) + \alpha_2(9-1+1) + \alpha_3(9+1+1) = 1$$
$$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = 1 \text{ -------------------- (ii)}$$

Simplifying the above 3 equations (i), (ii) and (iii) we get:

$$\alpha_1 = -3.5, \alpha_2 = 0.75 \text{ and } \alpha_3 = 0.75$$

# SVM (Support Vector Machine)...



The hyperplane that discriminates the positive class from the negative class is

$$Weight\ Vector\ (\tilde{w}) = \sum_i \alpha_i \tilde{S}_i$$

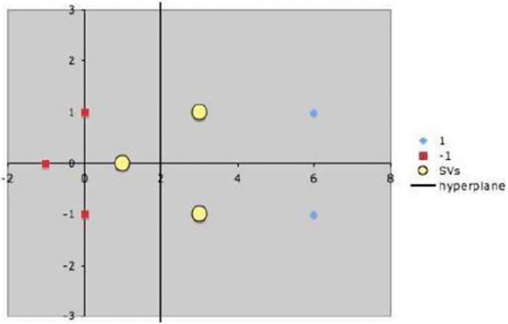
$$\tilde{w} = \alpha_1 \tilde{s_1} + \alpha_2 \tilde{s_2} + \alpha_3 \tilde{s_3}$$

$$\tilde{w} = -3.5 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

Finally, remembering that our vectors are augmented with a bias. Hence we can equate the last entry in $\tilde{w}$ as the hyperplane offset $b$. Therefore the separating hyperplane equation

$$\boldsymbol{y} = \boldsymbol{wx} + \boldsymbol{b} \text{ with } \boldsymbol{w} = \begin{pmatrix} \boldsymbol{1} \\ \boldsymbol{0} \end{pmatrix} \text{ and offset } \boldsymbol{b} = -\boldsymbol{2}.$$

Hence, equation of hyper plane that divides data points is $x - 2 = 0$.



# SVM (Support Vector Machine)...

## Advantages

- More effective in high dimensional spaces
- It is relatively memory efficient
- It can be used for both regression and classification problems
- It can work well with image data as well

## Disadvantages

- Not suitable for large datasets
- It can't perform well when the dataset has more noise
- More complex than decision tree

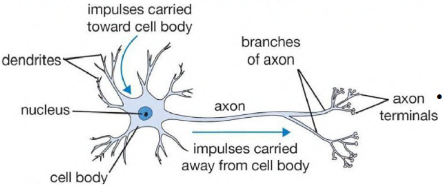
# Artificial Neural Networks (ANN)

## Artificial Neural Networks (ANN)

- ANN is a network model having a set of connected input-output units where each connection has a weight associated with it. The network/model tries to learn by its own through the adjustment of weights [Learning process]
- The ANN model used for the classification purpose uses a multilayer feed forward neural network where the model iteratively learns a set of weights for prediction of the class label of the tuples
- It consists of input layer, one or more hidden layer and an output layer

# Artificial Neural Networks (ANN)...

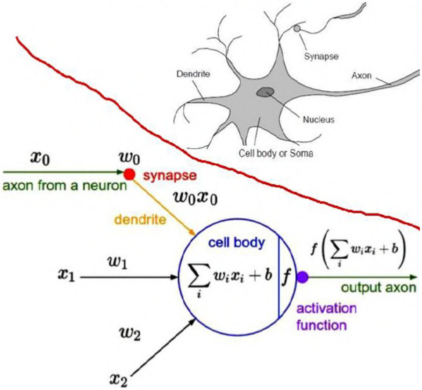
## The Structure of Biological Neurons




- The brain consists of a complex network of cells called neurons.
- Neurons communicate by transimitting electrochemical signals throughout the network.
- Each input signal to a neuron can inhibit or excite the neuron. When the neuron is excited enough (exceeds a certain amount of threshold), it fires its own electrochemical signal.
- A neuron has a cell body, a branching input structure (the dendrIte) and a branching output structure (the axOn)

# Artificial Neural Networks (ANN)...

## Properties of Artificial Neural Nets (ANNs)




- ANN posess a large number of processing elements called nodes/neurons which operate in parallel
- Neurons are connected with others by connection link. Each link is associated with weights and contains information about the input signal.
- Each neuron has an internal state of its own which is a function of the inputs that neuron receives- Activation function.
- The network Learn by tuning the connection weights

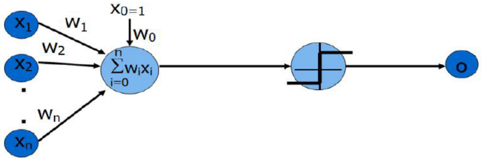
# Artificial Neural Networks (ANN)...





## Important Terminologies of ANN

1. **Perceptron:** A Perceptron is a type of artificial neuron which takes several binary inputs $<x_1, x_2, \ldots, x_n>$, weights for respective inputs, bias value per node and produces a single binary output.




2. **Weight:** Each neuron is connected to every other neuron by means of directed links. Links are associated with weights. The weight value lies between 0 to 1.

3. **Bias** is another weight included by adding a component $x_0 = 1$ to the input vector X.

$$X=(1, X_1, X_2 \ldots X_i, \ldots X_n)$$

Bias is of two types: - Positive bias: increase the net input

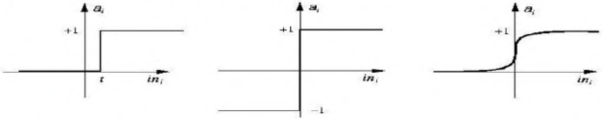
- Negative bias: decrease the net input

# Artificial Neural Networks (ANN)...

## Important Terminologies of ANN...

**4. Activation Function:** It is used to calculate the output response of a neuron. Sum of the weighted input signal is applied with an activation to obtain the response.

- Activation functions can be linear or non linear
- Types of Activation function
  - Identity function
  - Single/binary step function
  - Discrete/continuous **sigmoidal function**.




(a) Step function (b) Sign function (c) Sigmoid function

**5. Target Value (t)**: This is the known class value to which the perceptron output should match. When the network is under training phase, it is required to give the target value in addition to input, weight and bias value to compare the network's accuracy.

**6. Error**: The error value is the amount by which the value output by the network differs from the target value. For example, if we required the network to output 0 and its output as 1, then error =0-1= -1

# Artificial Neural Networks (ANN)...

## Important Terminologies of ANN...

**7. Learning Rate (η):** Used to control the amount of weight adjustment at each step of training. Learning rate ranges from 0 to 1. It determines the rate of learning in each time step/ iteration.

It is a process by which a neural network adapts itself to a stimulus/model by making proper parameter adjustments, resulting in the production of desired response

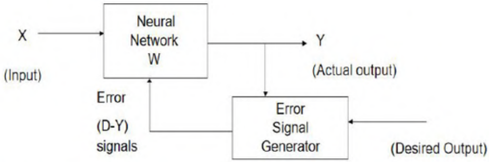
Two kinds of learning

- **Parameter learning**:- connection weights are updated
- **Structure Learning**:- change in network structure

**8. Training:** The process of modifying the weights of the connectors between network layers with the objective of achieving the expected output is called training a network.

Traing a network can be achieved through

- Supervised learning
- Unsupervised learning
- Reinforcement learning

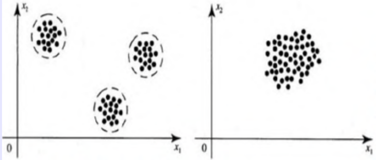



– **Supervised learning**

In supervised training, the network is trained by presenting it a sequence of training inputs (patterns), each with an associated target output value. Weights in the network are adjusted according to a learning algorithm.

# Artificial Neural Networks (ANN)...

## Important Terminologies of ANN...

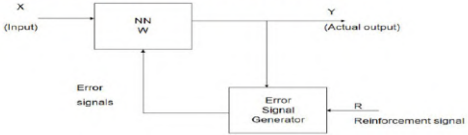


– **Unsupervised learning**

In unsupervised training, a sequence of training inputs is provided, but no target output values are specified. The weights are adjusted according to a learning algorithm. All similar input patterns are grouped together as clusters.

-- **Reinforcement learning**

If less information is available about the target output values (critic information). Learning based on this critic information is called reinforcement learning and the feedback sent is called reinforcement signal. Feedback in this case is only evaluative and not instructive



# Artificial Neural Networks (ANN)...

## Important Terminologies of ANN...

**9. Perceptron Learning Rule:**

The perceptron compute the output **(Y)** on the basis weight(s), bias value and activation function and compare the computed output with the target output **(t)**

- If the computed output is correct ($t == Y$) the weights $w_i$ are not changed
- If the output is incorrect ($t \neq y$) the weights $w_i$ are changed by value $\Delta w_i$ such that the output of the perceptron for the new weights will be closer to target t.

$$\boxed{\begin{array}{l} w_{ij} = w_{ij} + \Delta w_{ij} \\ \quad \text{where } \Delta w_{ij} = \eta Err_j \, Y_i \end{array}}$$

- Err is the obtained error
- Yi is the output of lower ith unit
- $\eta$ is Learning Rate (small constant <1)

The algorithm converges to the correct classification

- if the training data is linearly separable
- and $\eta$ is sufficiently small

# Artificial Neural Networks (ANN)...

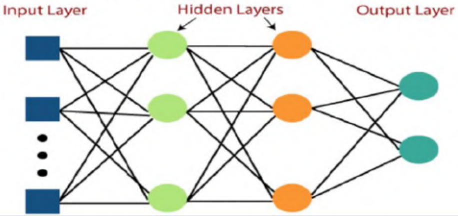


## Important Terminologies of ANN...

### 10. Multilayer Perceptron (MLP):

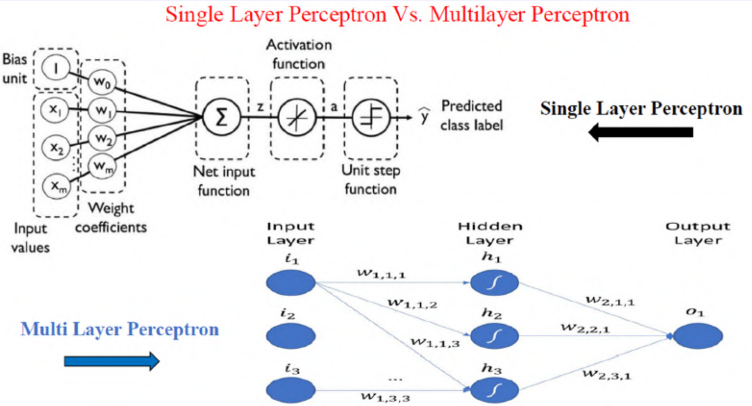
Multi-layer perception is a fully connected dense multiple layers neural network which transform any input dimension to the desired dimension.

A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes. A schematic diagram of a Multi-Layer Perceptron (MLP) is depicted below.



# Artificial Neural Networks (ANN)...

## Important Terminologies of ANN...



# Artificial Neural Networks (ANN)...



## Important Terminologies of ANN...

**11. Learniing Algorithm (Backpropagation Algorithm):**

The algorithm has mainly 2 phases.

- **Forward Pass Phase:** This phase computes the "functional signal" and does the feed forward propagation of input signals through network till the output neuron/node.
- **Backward Pass Phase:** This phase compute the "error signal" and propagate the error backwords through network starting from output units towards input node by updating the weight value such that the output of the perceptron for the updated weights will be closer to the target.

**12. Epoch:**

In the training phase one by one tuple is given as input to the MLP. The presentation of the entire training set to the neural network is called as one epoch.

For instance, for AND function an epoch consists of four sets of inputs being presented to the network (i.e. [0,0], [0,1], [1,0], [1,1])

# Multi-Layer Neural Networks

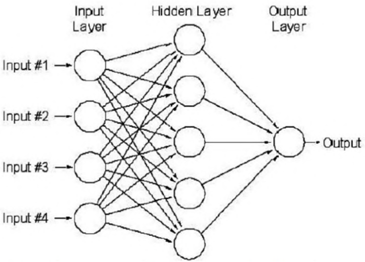


## Multi-Layer Neural Networks

- It is made up from an input, output and one or more hidden layers
- Each node from input is connected to a node from hidden layer and every node from hidden layer is connected to a node in output layer
- There is usually some weight associated with every connection. Input layer represents the raw information that is fed into the network
- Every single input to the network is duplicated and send down to the nodes in hidden layer. Hidden layer accepts data from input layer
- Output layer process information received from the hidden layer and produces an output. This output is then processed by activation function
- This network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer

# Multi-Layer Neural Networks...

## Multi-Layer Neural Networks...




**Number of Input Nodes:**
Number of input nodes depends on training set in hand or number of nodes is equal to the number of features (columns) in training dataset. Some NN configurations add one additional node for a bias term

**Number of Output Nodes:**
If the NN is a regressor, then the output layer has a single node. If the NN is a classifier, then it output nodes corresponds to per class label of the model

**Number of Hidden Layers:**
The hidden layer configuration can be set by using just two rules:
(i) number of hidden layers equals one or two
(ii) number of neurons in that layer is the mean of the neurons in the input and output layers

# Backpropagation Algorithm





## Backpropagation Algorithm

The training algorithm of backpropagation involves 4 steps

1. **Intialization of Weight:** Some small random weights (-1 to +1 or -0.5 to +0.5) are assigned to each connectors and some bias values are assigned to each computational unit (neurons).
2. **Feed Forward [Propagate the input forward]:**
   - The training tuple is fed to the network input unit layer ($x_i$) which propagate the received input ($x_i$) to the hidden layer as aj and then to the subsequent layers as follows.

     $\boxed{\text{Input from } i^{th} \text{ unit to } j^{th} \text{ unit} = I_j = \sum_i w_i x_i + \theta_i}$ Where $w_{ij}$ = Weight of connector i to j

     $x_i$ = Input from $i^{th}$ unit

     $\theta_j$ = Bias value of jth unit
   - Each $j^{th}$ hidden unit [or output unit] compute the otput ($Y_j$) throuh the activation function (sigmoid) and propagate the output to the next unit [or outputs the response: class]

     $$\boxed{\text{Output of } j^{th} \text{ unit} = Y_j = f(I_j) = \frac{1}{1+e^{-Ij}}}$$

# Backpropagation Algorithm...

## Backpropagation Algorithm...

3. **Backpropagation of Errors:** Each unit (output/hidden) compares the computed output and targeted output to determine the **associated error** of that unit. The error is propagated backward by **updating the weights and biases** to rectify the error of the network prediction.

$$\text{Associated Error of } j^{th} \text{ unit}(Err_j) = \begin{cases} \text{if j is the output unit: } Err_j = Y_j(1-Y_j)(T_j - Y_j) \\ \text{if j is the hidden unit: } Err_j = Y_j(1-Y_j)\sum_k Err_k * w_{jk} \end{cases}$$

Where, $T_j$: Target output, $Y_j$ : ComputedOutput, $Y_j(1-Y_j)$: Derivative of the sigmoid function
$Err_k$: Higher layer error, k: Higher layer index

$$\text{Updating weight } (W_{ij})\text{: } W_{ij} = W_{ij} + \Delta W_{ij} \quad \text{Where } \Delta W_{ij} = \eta * Err_j * Y_i$$

$$\text{Updating Bias } (\theta_j)\text{: } \theta_j = \theta_j + \Delta\theta_j \quad \text{Where } \Delta\theta_j = \eta * Err_j$$

Where, $\eta$: Given learning rate of network
$Err_j$: Error at $j^{th}$ unit of higher layer
$Y_i$: Output from $i^{th}$ unit of lower laye

4. **Iteratively updation of weights and bias value for one epoch.**

# Backpropagation Algorithm...

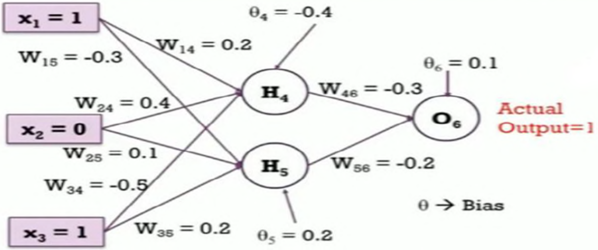
## Backpropagation Algorithm...

- **Advantages**
  - Relatively simple implemenion
  - Mathematical formula used in the algorithm can be applied to any network
  - Computing time can be reduced if the weight choosen are small at the beginning
  - Well suited for continuous valued input and output
  - High tolerant for noisy data
  - Ability to classify patterns for which they haven't been trained
- **Disadvantages**
  - Slow and inefficient. Can stuck in local minima resulting in sub-optimal solution
  - In case of large input, difficult to relate output w.r.t. inputs
  - Require long training time
  - Poor interpretability: difficult to interpret parameter values
- **Application**
  - Successful in real world data: Handwritten character Recognition, pathology and laboratory medicine

# Backpropagation Algorithm...

Q: Assume that the neurons have a sigmoid activation function, perform a forward pass and backward pass on the network to update the weights. Assume target output is 1 and learning rate is 0.9.

| X1 | X2 | X3 | W14 | W15 | W24 | W25 | W34 | W35 | W46 | W56 | $\theta 4$ | $\theta 5$ | $\theta 6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0.2 | -0.3 | 0.4 | 0.1 | -0.5 | 0.2 | -0.3 | -0.2 | -0.4 | 0.2 | 0.1 |



# Backpropagation Algorithm...



**Step 1 (Feed Forward) :-** Compute Hidden Layer's & Output Layer's Input (Ij) and output (Yj)

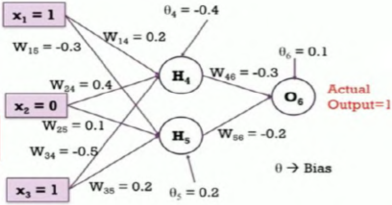
**Input from $i^{th}$ unit to $j^{th}$ unit** $= I_j = \sum_i w_i x_i + \theta_j$

$I_4 = (0.2*1 + 0.4*0 + -0.5*1) + -0.4 = -0.7$

$I_5 = (-0.3*1 + 0.1*0 + 0.2*1) + 0.2 = 0.1$

$I_6 = (W_{46}*Y_4 + W_{56}*Y_5) + \theta_6$

$= -0.3 *0.332 + -0.2*.525 = -0.105$




**Output of $j^{th}$ unit** $= Y_j = f(I_j) = \frac{1}{1+e^{-Ij}}$

$Y_4 = 1/1+e^{-0.7} = 0.332$

$Y_5 = 1/1+e^{-0.1} = 0.525$

$Y_6 = 1/1+e^{+0.105} = 0.474$

# Backpropagation Algorithm...

**Step 2 (Backpropagate)** :- Update weight of each connector by evaluating the **corresponding error**

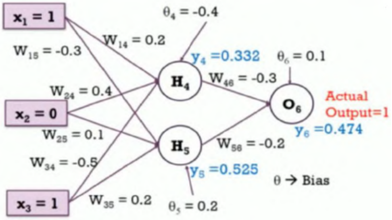
Updated Weight $W_{ij} = W_{ij} + \Delta W_{ij}$

Where $\Delta W_{ij} = \eta * Err_j * Y_i$

Update Weight $W_{46}$ & $W_{56}$ using $Err_6$

Update Weight $W_{14}$ $W_{24}$ & $W_{34}$ using $Err_4$

Update Weight $W_{15}$ $W_{25}$ & $W_{35}$ using $Err_5$



# Backpropagation Algorithm...

**Step 2 (Backpropagate)** :-
(Compute Error)

Associated Error of $j^{th}$ unit($Err_j$) $=$

$$\begin{cases} \text{if j is the output unit: } Err_j = Y_j(1-Y_j)(T_j - Y_j) \\ \text{if j is the hidden unit: } Err_j = Y_j(1-Y_j)\sum_k Err_k * w_{jk} \end{cases}$$

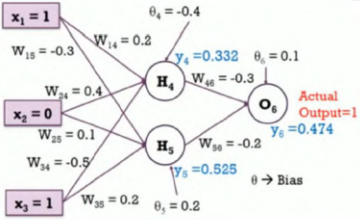
**Output Layer Error:**

$$Err_6 = Y_6 * (1-Y_6) * (T_6 - Y_6)$$
$$= 0.474 * (1-0.474) * (1-0.474) = 0.1311$$

**Hidden Layer Error:**

$$Err_4 = Y_4 * (1-Y_4) * W_{46} * Err_6$$
$$= 0.332 * (1-0.332) * -0.3 * 0.1311$$
$$= -0.0087$$

$$Err_5 = Y_5 * (1-Y_5) * W_{56} * Err_6$$
$$= 0.525 * (1-0.525) * -.2 * 0.1311$$
$$= -0.0065$$



# Backpropagation Algorithm...



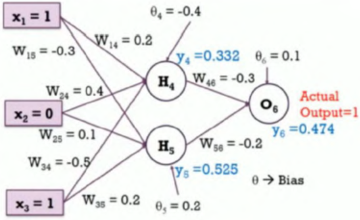
**Step 2 (Backpropagate)** :-
(Update Weight and Biase)

Updated Weight $W_{ij} = W_{ij} + \Delta W_{ij}$
Where $\Delta W_{ij} = \eta * Err_j * Y_i$

**Unit - 6**

$$\text{Update } W_{46} = W_{46} + \eta * Err_6 * Y_4 = -0.3 + 0.9 * 0.1311 * 0.332 = -0.261$$

$$\text{Update } W_{56} = W_{56} + \eta * Err_6 * Y_5 = -0.2 + 0.9 * 0.1311 * 0.525 = -0.138$$



# Backpropagation Algorithm...

**Step 2 (Backpropagate)** :-
(Update Weight and Biase)

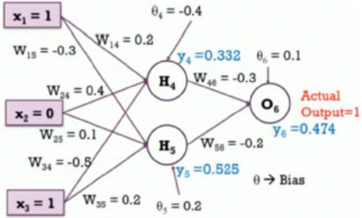
Updated Weight $W_{ij} = W_{ij} + \Delta W_{ij}$
Where $\Delta W_{ij} = \eta * Err_j * Y_i$

**Unit - 4**

Update $W_{14} = W_{14} + \eta * Err_4 * Y_1$
$= -0.2 + 0.9 * -0.0087 * 1$
$= 0.192$

Update $W_{24} = W_{24} + \eta * Err_4 * Y_2$
$= 0.4 + 0.9 * -0.0087 * 0$
$= 0.4$

Update $W_{34} = W_{34} + \eta * Err_4 * Y_3$
$= -0.5 + 0.9 * -0.0087 * 1$
$= -0.508$



# Backpropagation Algorithm...



**Step 2 (Backpropagate) :-**
(Update Weight and Biase)

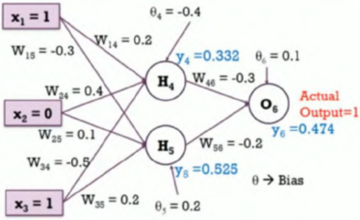
Updated Weight $W_{ij} = W_{ij} + \Delta W_{ij}$
Where $\Delta W_{ij} = \eta * Err_j * Y_i$

<u>**Unit - 5**</u>

$$\text{Update } W_{15} = W_{15} + \eta * Err_5 * Y_1 = -0.3 + 0.9 * -0.0065 * 1 = -0.306$$

$$\text{Update } W_{25} = W_{25} + \eta * Err_5 * Y_2 = 0.1 + 0.9 * -0.0065 * 0 = 0.1$$

$$\text{Update } W_{35} = W_{35} + \eta * Err_5 * Y_3 = 0.2 + 0.9 * -0.0065 * 1 = 0.194$$



# Backpropagation Algorithm...





**Step 2 (Backpropagate) :-**
(Update Weight and Biase)

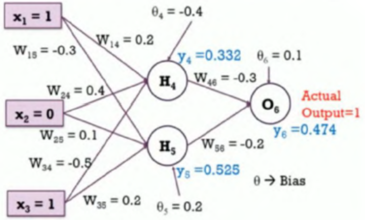
Updated Bias $(\theta_j)$: $\theta_j = \theta_j + \Delta\theta_j$
Where $\Delta\theta_j = \eta * Err_j$

**New $\theta$ of Unit: 4, 5 & 6**

Update $\theta_6 = \theta_6 + \eta * Err_6$
$= 0.1 + 0.9 * 0.1311$
$= 0.218$

Update $\theta_5 = \theta_5 + \eta * Err_5$
$= 0.2 + 0.9 * -0.0065$
$= 0.194$

Update $\theta_4 = \theta_4 + \eta * Err_4$
$= -0.4 + 0.9 * -0.0087$
$= -0.408$



# Backpropagation Algorithm...





| Parameter (Weight) | OLD | NEW |
|---|---|---|
| W14 | 0.2 | 0.192 |
| W15 | -0.3 | -0.306 |
| W24 | 0.4 | 0.4 |
| W25 | 0.1 | 0.1 |
| W34 | -0.5 | -0.508 |
| W35 | 0.2 | 0.194 |
| W46 | -0.3 | -0.261 |
| W56 | -0.2 | -0.138 |

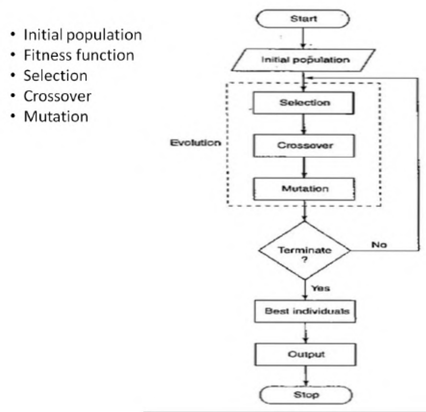
| Parameter (Bias Value) | OLD | NEW |
|---|---|---|
| $\theta 4$ | -0.4 | -0.408 |
| $\theta 5$ | 0.2 | 0.194 |
| $\theta 6$ | 0.1 | 0.218 |

Iterate the same process of finding error and updating parameters till the error value is not acceptable

# Genetic Algorithm (GA)

## Genetic Algorithm (GA)

- It is based on an analogy to biological evolution
- This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation
- Five phases are considered in a genetic algorithm



# Genetic Algorithm (GA)...



## Genetic Algorithm (GA)...

- **Initial Population**: is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - Ex:if A1 and $\neg A2$ then C2 can be encoded as 100
  - If an attribute has k > 2 values, k bits can be used
- **Fitness Function**: Based on the notion of survival of the fittest, a new population is formed to consist of the fittest rules and their offspring. The fitness of a rule is represented by its classification accuracy on a set of training examples
- **Selection, Mutation & Crossover**: Offspring are generated by crossover and mutation and then is selected/tested on the basis of fitness function
- **Termination Condition**: The process continues until a population P evolves when each rule in P satisfies a prespecified threshold
- It is slow but easily parallelizable

# Performance Measure of Classifier

## Confusion Matrix

- A confusion matrix is a table that is used to describe the performance of classifier based on test dataset for which, actual class label are known
- For m-classes, a confusion matrix is a table of size m x m
- The matrix compares the actual target values with those predicted by the classification model. This gives us a holistic view of how well our prediction model is performing and what kinds of errors it is making
- A classifier is called as ideal classifier if in its confusion matrix the diagonal entries contain most/all tuples (number) and rest entries contain less/zero tuples

| | Actual Class 1 | Actual Class 2 |
|---|---|---|
| Predicted Class 1 | TP | FP |
| Predicted Class 2 | FN | TN |

# Performance Measure of Classifier...



## Confusion Matrix...

- **True Positive (TP)**: It represents correctly classified positive classes. Both actual and predicted classes are positive here
- **False Positive (FP)**: It represents incorrectly classified positive classes. These are the positive classes predicted by the model that were actually negative. This is called Type I error
- **True Negative (TN)**: It represents correctly classified negative classes. Both actual and predicted classes are negative here
- **False Negative (FN)**: It represents incorrectly classified negative classes. These are the negative classes predicted by the model that were actually positive. This is called Type II error

# Performance Measure of Classifier...

## Confusion Matrix...

| RID | Age | Income | Student | Credit_Rating | Actual Class: Buys Computer | Predicted Class: Buys Computer |
|---|---|---|---|---|---|---|
| 1 | Youth | High | No | Fair | No | Yes |
| 2 | Youth | High | No | Excellent | No | Yes |
| 3 | middle_aged | High | No | Fair | Yes | No |
| 4 | senior | Medium | No | Fair | Yes | Yes |
| 5 | senior | Low | Yes | Fair | Yes | No |
| 6 | senior | Low | Yes | Excellent | No | Yes |
| 7 | middle_aged | Low | Yes | Excellent | Yes | Yes |
| 8 | Youth | Medium | No | Fair | No | Yes |
| 9 | Youth | Low | Yes | Fair | Yes | No |
| 10 | senior | Medium | Yes | Fair | Yes | No |
| 11 | Youth | Medium | Yes | Excellent | Yes | Yes |
| 12 | middle_aged | Medium | No | Excellent | Yes | No |
| 13 | middle_aged | High | Yes | Fair | Yes | No |
| 14 | senior | Medium | No | Excellent | No | Yes |

Confusion matrix of the given dataset is:

| Actual class\Predicted class | buy_computer = yes | buy_computer = no | Total |
|---|---|---|---|
| buy_computer = yes | 3 | 6 | 9 |
| buy_computer = no | 5 | 0 | 5 |
| Total | 8 | 6 | 14 |

## Performance Measure of Classifier...

### Accuracy

- It is the percentage of correct predictions made by the model
- $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- Misclassification rate/Error rate = 1-Accuracy(M)

### Precision

- It is the percentage of predicted positives that are actually positive
- $Precision = \frac{TP}{TP+FP}$

### Recall (Sensitivity (True Positive Rate))

- It is the percentage of actual positives that are correctly classified by the model
- $Recall = \frac{TP}{TP+FN}$

# Performance Measure of Classifier...



## F1-Score

- It is the harmonic mean of recall and precision. It becomes high only when both precision and recall are high
- $F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}} = \frac{2 * Precision * Recall}{Precision + Recall}$

## Specificity (True Negative Rate)

- It is the proportion of negative tuples that are correctly identified. $Specificity = \frac{TN}{FP + TN}$

| Actual class\Predicted class | buy_computer = yes | buy_computer = no | Total |
|---|---|---|---|
| buy_computer = yes | 3 | 6 | 9 |
| buy_computer = no | 5 | 0 | 5 |
| Total | 8 | 6 | 14 |

Accuracy = (3+0)/(3+0+6+5) = 3/14 = 21%

Error Rate = 1 – Accuracy = 79%

Precision = 3/(3+6) = 3/9 = 33%

Recall = 3/(3+5)= 3/8 = 38%

F1-Score = 35%

Specificity = 0

# Performance Measure of Classifier...

## Performance Measure of Classifier...

Q: Suppose we have to classify 100 people (which includes 40 pregnant women and the remaining 60 are not pregnant women and men with a fat belly) as pregnant or not pregnant. Out of 40 pregnant womem, 30 pregnant women are classified correctly and the remaining 10 pregnant women are classified as not pregnant by the classifier. On the other hand, out of 60 people in the not pregnant category, 55 are classified as not pregnant and the remaining 5 are classified as pregnant. Compute accuracy, precision, recall, f1-score and specificity

| | Actual Positive | Actual Negative |
|---|---|---|
| **Predicted Positive** | TP= 30 | FP= 5 |
| **Predicted Negative** | FN= 10 | TN= 55 |

Accuracy = (30+55)/(30+55+5+10) = 85/100 = 85%

Precision = 30/(30+5) = 30/35 = 86%

Recall = 30/(30+10)= 30/40 = 75%

F1-Score = 80%

Specificity = 55/(5+55) = 55/60 = 92%

Ref: J. Han, M. Kamber and J. Pei, "Data Mining: Concepts and Techniques", Morgan Kaufmann, 3rd edition