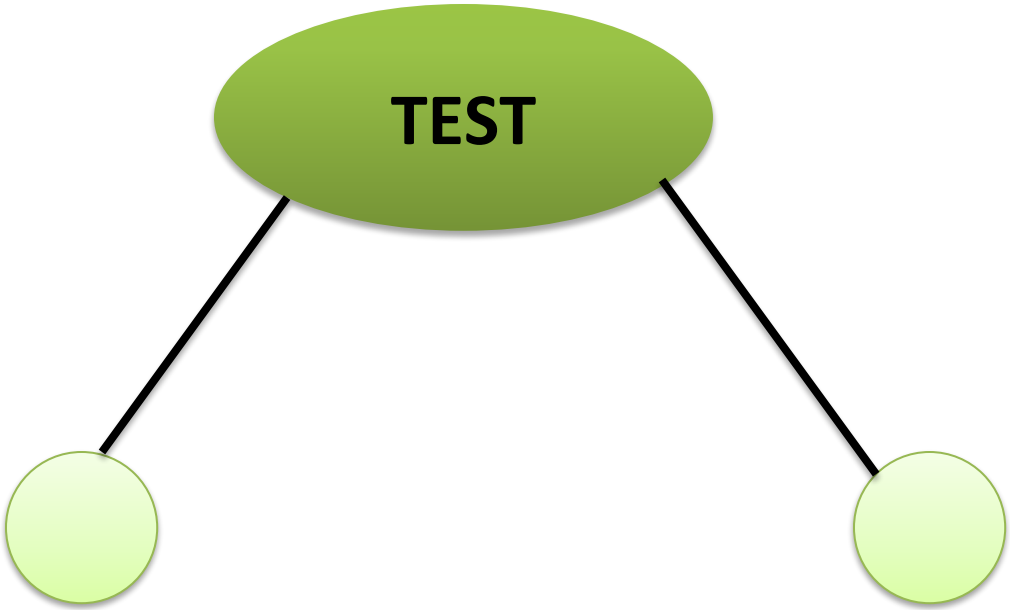
UNIT-2

DECISION TREE LEARNING

Decision Tree

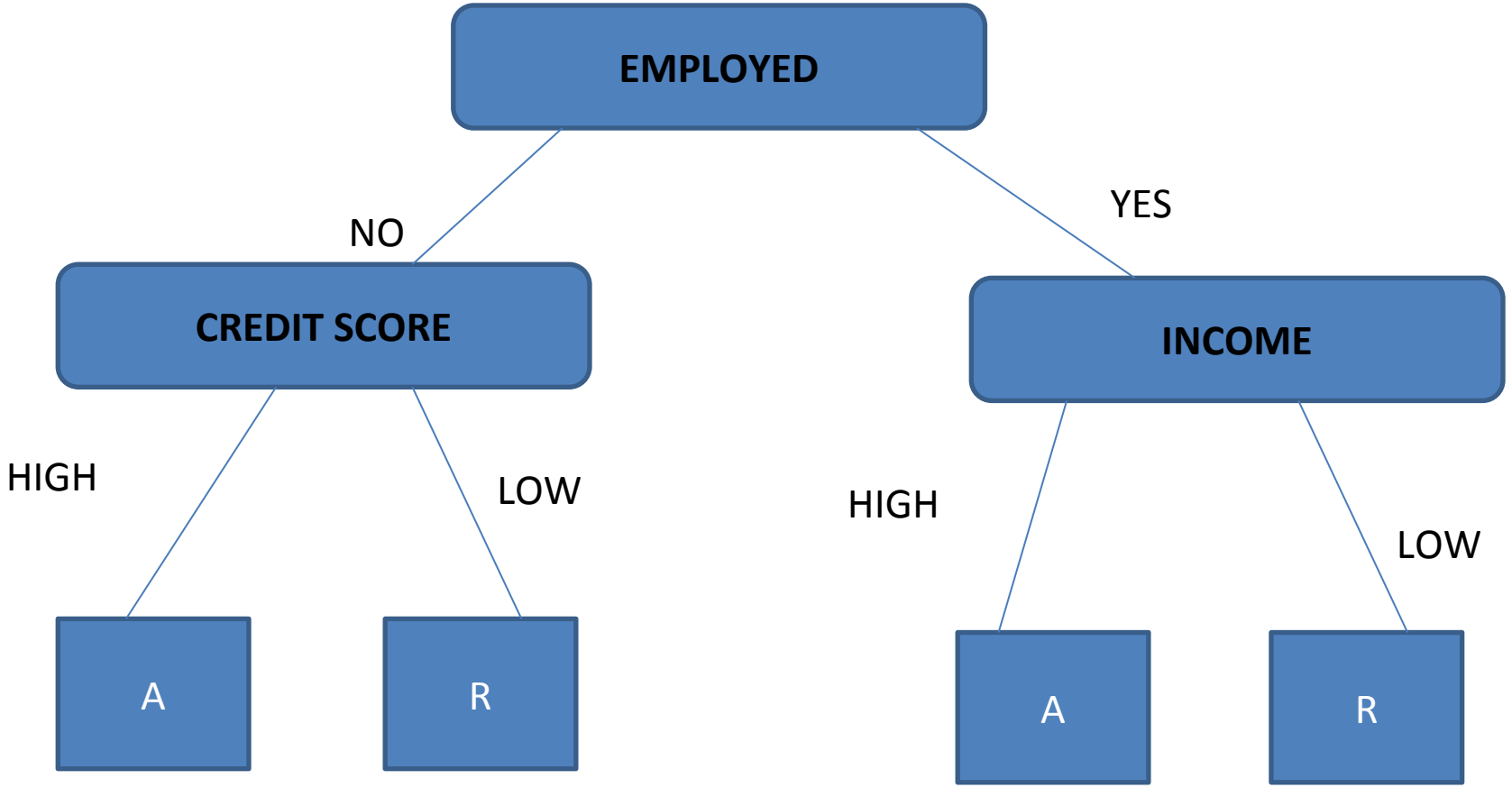
- Decision tree learning is one of the most widely method used for inductive learning.
- It is a method of approximating discrete valued functions that is robust to noisy data.
- Decision tree is a supervised learning where the data is continuously split on the basis of a certain parameter.
- Decision tree classifies instances by sorting them down the tree from root node to leaf node.
- Decision tree comprises of two types of nodes: Decision nodes and Leaf nodes.

- Decision nodes: specifies a choice or a “test” based on which we decide.



- Leaf node: They indicates the classification of the example or the value of the example.
- Decision tree can be used to treat both classification and regression problems.
- Logic of Decision tree working:

  “Start with the root node of the tree, based on the value of the test you move to the corresponding branch and this process continues until you reach the leaf node”.



DESIGNING A DECISION TREE FOR

" WHETHER LOAN SHOULD BE APPROVE OR NOT" !

APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

We can apply decision tree where

- Instances are represented by attribute value pairs.
- Target function has discrete output values.
- Problems in which training statements contain missing attributes.
- Training data may contain errors.

Strength and weakness of decision trees

- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.
- Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.

Decision Tree Learning Algorithm

- ID3 is one of the common Decision Tree Learning Algorithm .
- ID3 is acronym of Iterative Dichotomiser.
- It was introduced in 1986.
- Decision tree learning algorithm transform raw data to rule based decision making tree.
- ID3 build the decision trees in top down approach with the question ***which attribute to be tested at the root node?***
- The best attribute is selected and used as the test at the root node of the tree.
- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node.

- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.
- Now while designing a decision tree there are two decisions that needs to be made:

  (a) ***WHEN TO STOP***

  - no more input features

  - all examples classified are same

  (b) ***WHICH TEST TO SPLIT ON***

  - Split that gives the small error

  - Split that divides the values into ½.

- To decide on these two particular decisions we measure some statistical quantities.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Case 1: Which Attribute is the best classifier

- The central choice in ID3 algorithm is to select which attribute to test at each node.
- Hence in order to identify this, we consider a statistical property called INFORMATION GAIN.
- (a) ENTROPY: In order to define info gain we define a measure called "entropy". ***Entropy is the impurity of an arbitrary collection of examples.***
- Lets say there is a collection (S) containing positive and negative examples related to some target concept.

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Information Gain

Information gain, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute. More precisely, the information gain, **Gain(S, A) of an attribute A,** relative to a collection of examples **S, is defined as.**

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Gain (S, A) is the ***expected reduction in entropy*** caused by the value of attribute (A)

OR

Gain(S, A) is the ***information provided about the target function value***, given the value of some other attribute A

Example: The above figure is denoted as S which is a set of training examples with attributes.

Lets us consider the attribute " WIND" that can have values either " WEAK" or "STRONG".

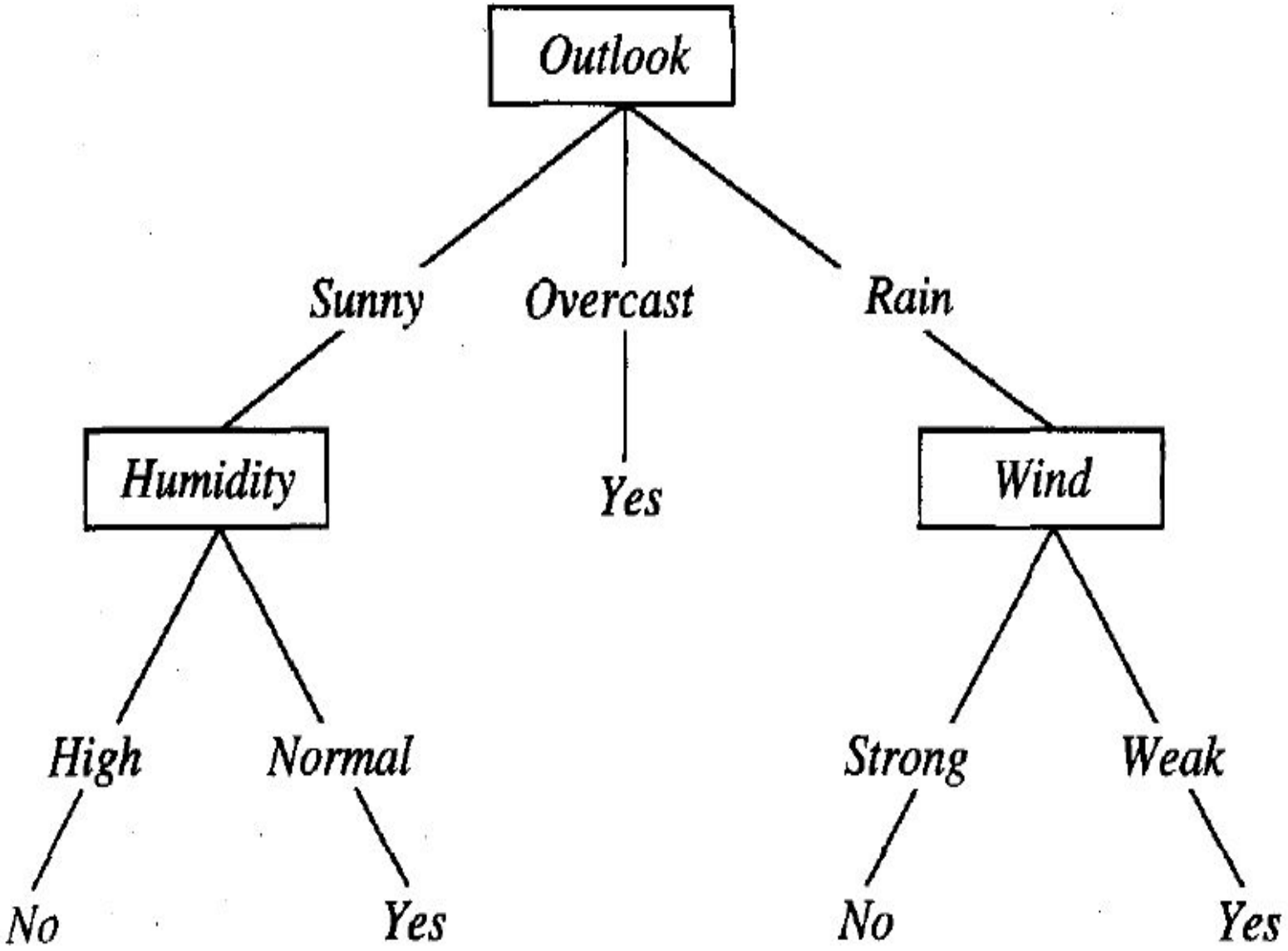
S is a collection containing 14 examples, [9+, 5-] Of these 14 examples, **Suppose 6 of the positive and 2 of the negative examples have *Wind = Weak, and* the remainder have *Wind = Strong.***

Values(Wind) = Weak, Strong

$$
\begin{aligned}
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

Similarly, the Gain for each attribute can be calculated;
Gain(S. Outlook)= 0.246
Gain(S, Humidity)=0.151
Gain(S, Wind)=0.048
Gain(S, Temperature)= 0.029
According to the ***information gain measure, the Outlook attribute provides the best prediction of the target attribute, Play Tennis***, over the training examples. Therefore, **Outlook is selected as the decision attribute for the root node**, and branches are created below the root for each of its possible values (i.e., Sunny, Overcast, and Rain).

Every example for which ***Outlook = Overcast is also a positive example*** of ***Play Tennis. Therefore, this node of the tree becomes a leaf node with*** the classification ***Play Tennis = Yes.***


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

INDUCTIVE BIAS IN DECISION TREE LEARNING

- What is the policy by which ID3 generalizes from observed training examples
- to classify unseen instances? In other words, what is its inductive bias?
- ID3 search strategy (a) selects in favor of shorter trees over longer ones, and
- (b) selects trees that place the attributes with highest information gain closest to the root.
- **Approximate inductive bias of ID3: Shorter trees are preferred over larger trees.**
- The inductive bias of ID3 is thus a preference for certain hypotheses over others (e.g., for shorter hypotheses), with no hard restriction on the hypotheses that can be eventually enumerated. This form of bias is typically called a preference bias

Issues in Decision Tree learning

- Overfitting the data: Given a hypothesis space *H*, a hypothesis is said to **overfit** the training data if there exists some alternative hypothesis , such that *h* has smaller error than *h*' over the training examples, but *h*' has smaller error than *h* over the entire distribution of instances.
- Guarding against bad attribute choices: The information gain measure has a bias that favors attributes with many values over those with only a few.

- Handling continuous valued attributes: Note that continuous valued attributes can be partitioned into a discrete number of disjoint intervals. Then we can test for membership to these intervals.
- If the *Temperature* attribute in the *PlayTennis* example took on continuous values in the range 40-90, note that we cannot just use the same approach as before.

Artificial Neural Network (ANN)

- ANN are the computational models that are inspired by the human brain.
- Information processing model that is inspired by the way biological nervous system (i.e) the brain, process information.
- ANN is composed of large number of highly interconnected processing elements(neurons) working in unison to solve problems.
- Processing units make up ANNs, which in turn consist of inputs and outputs. The inputs are what the ANN learns from examples to produce the desired output.

- The neural networks have the ability to learn by example which makes them very flexible and powerful.
- A neural network derives its computing power through its massively parallel distributed structure and its ability to learn and therefore generalize.
- Generalization refers to the neural network's production of reasonable outputs for inputs not encountered during training.
- Large number of processing elements called neurons exists here.
- Interconnections with weighted linkage hold informative knowledge.

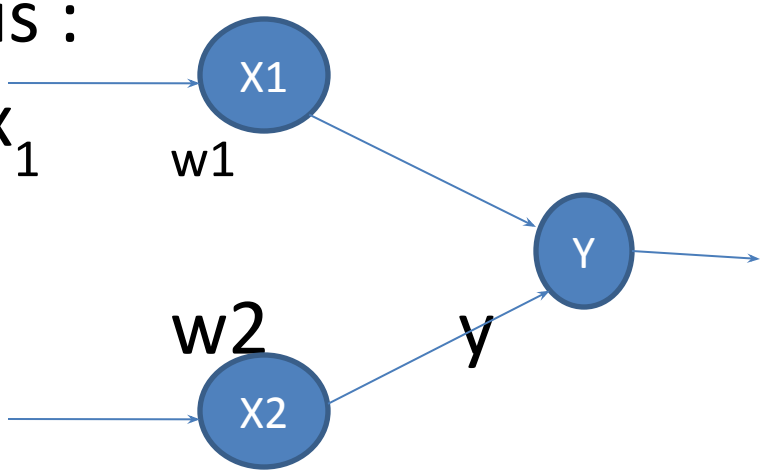
- Neurally implemented mathematical model
- Input signals arrive at processing elements through connections and connecting weights.
- Processing elements can learn, recall and generalize from the given data.
- Computational power is determined by the collective behavior of neurons.
- ANN is a connection models, parallel distributed processing models, self organizing systems.

- Nodes – interconnected processing elements (units or neurons).
- Neuron is connected to other by a **connection link.**
- Each connection link is associated with **weight which has** information about the input signal.
- ANN processing elements are called as **neurons or artificial neurons , since they have the capability to model networks of** original neurons as found in brain.
- Internal state of neuron is called **activation or activity level of** neuron, which is the function of the inputs the neurons receives.
- Neuron can send only one signal at a time.

Basic Operation of a Neural Network

- X1 and X2 – input neurons.
- Y- output neuron
- Weighted interconnection links- W1 and W2
- Net input calculation is :

$Y_{in} = w_1x_1 + w_2x_2$



x_1 w1 w2 y x_2

- Output is : $y = f(Y_{in})$
- The function to be applied over the net input is called **activation function.**

Important terminologies

- **Weight**
- The weight contain information about the input signal.
- It is used by the net to solve the problem.
- It is represented in terms of matrix & called as *connection matrix.*
- If weight matrix W contains all the elements of an ANN, then the
- set of all W matrices will determine the set of all possible
- information processing configuration.
- The ANN can be realized by finding an appropriate matrix W.

- Bias
  - Bias has an impact in calculating net input.
  - Bias is included by adding x0 to the input vector x.

- The net output is calculated by

$$y_{inj} = \sum_{i=0}^{n} x_i w_{ij}$$

$$y_{inj} = b_j + \sum_{i=0}^{n} x_i w_{ij}$$

Contd..

1. For the network shown in Figure 1, calculate the net input to the output neuron.

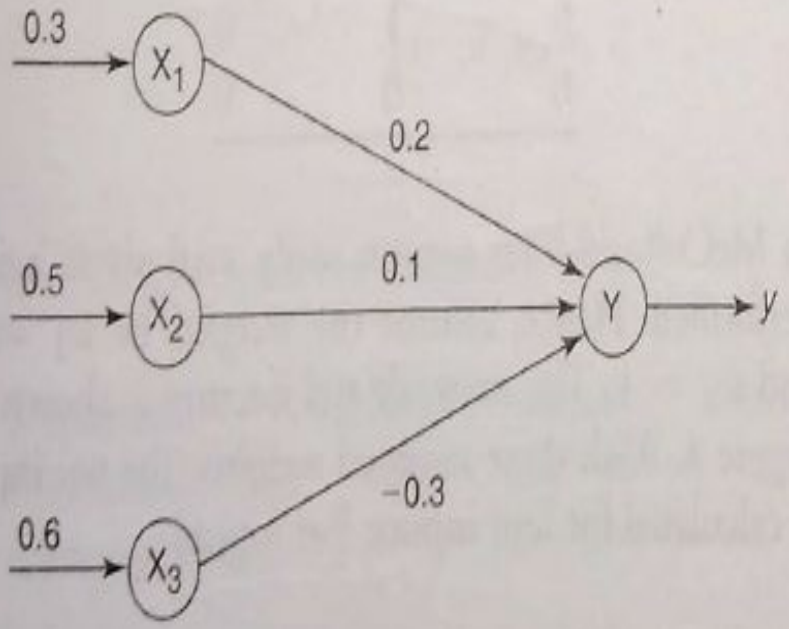


Figure 1 Neural net.

Solution: The given neural net consists of three input neurons and one output neuron. The inputs and weights are

$$[x_1, x_2, x_3] = [0.3, 0.5, 0.6]$$

$$[w_1, w_2, w_3] = [0.2, 0.1, -0.3]$$

The net input can be calculated as

$$\begin{aligned} y_{in} &= x_1 w_1 + x_2 w_2 + x_3 w_3 \\ &= 0.3 \times 0.2 + 0.5 \times 0.1 + 0.6 \times (-0.3) \\ &= 0.06 + 0.05 - 0.18 = -0.07 \end{aligned}$$

2. Calculate the net input for the network shown in Figure 2 with bias included in the network.

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

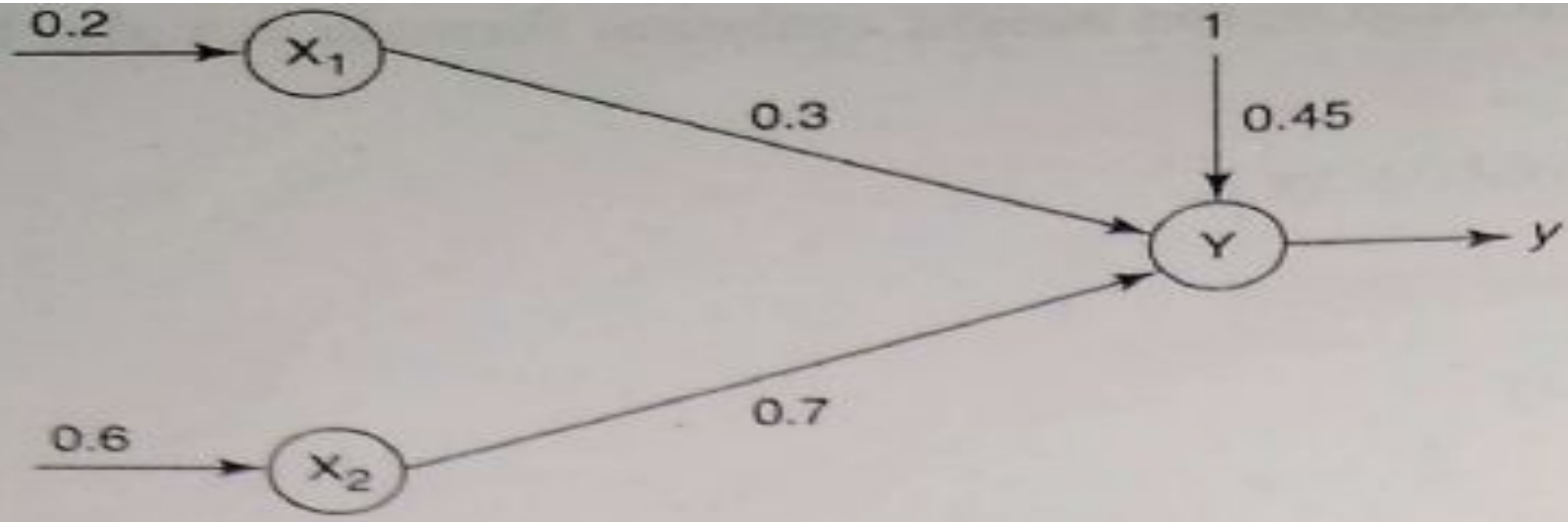


Figure 2 Simple neural net.

$[x_1, x_2] = [0.2, 0.6]$ 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, the net input is calculated as

$$\begin{aligned} y_{in} &= b + x_1 w_1 + x_2 w_2 \\ &= 0.45 + 0.2 \times 0.3 + 0.6 \times 0.7 \\ &= 0.45 + 0.06 + 0.42 = 0.93 \end{aligned}$$

Therefore $y_{in} = 0.93$ is the net input.

3. Obtain the output of the neuron Y for the network shown in Figure 3 using activation functions as: (i) binary sigmoidal and (ii) bipolar sigmoidal.

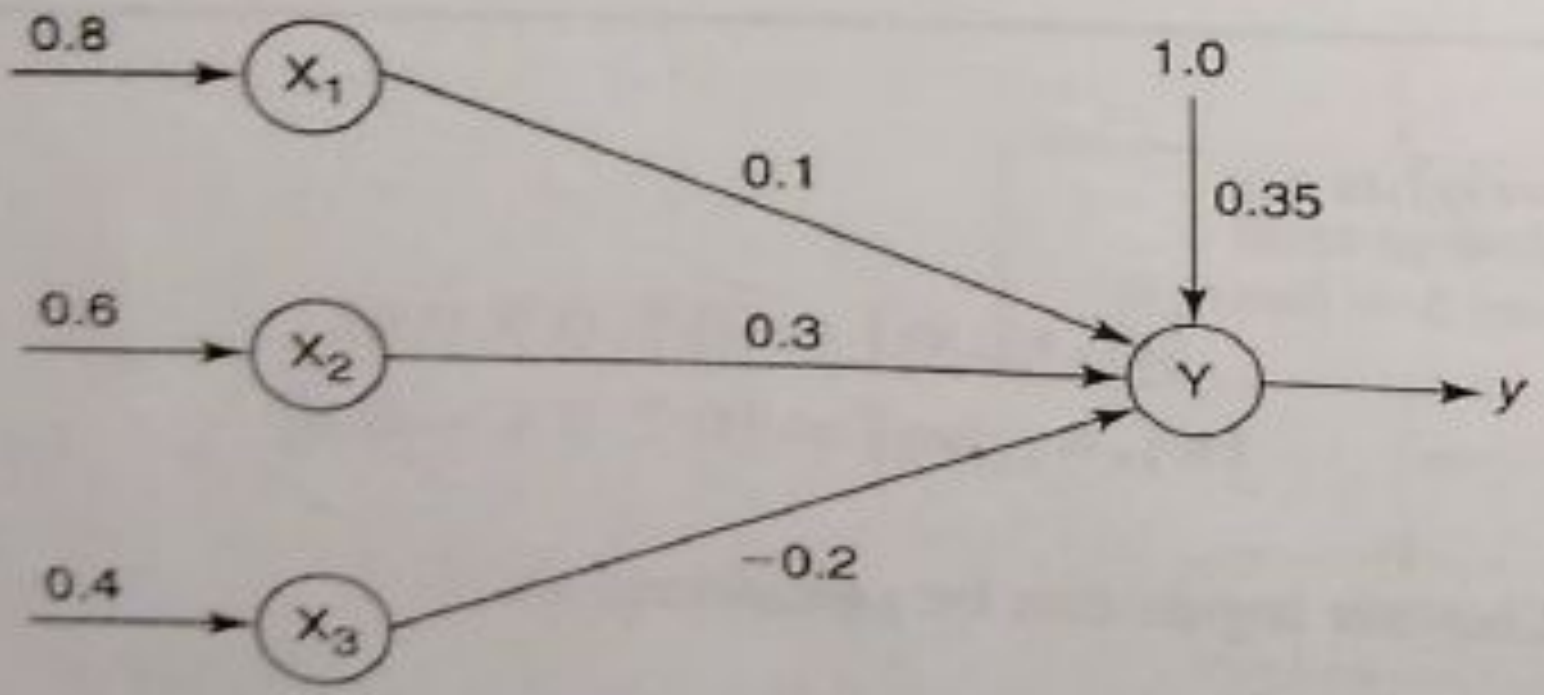


Figure 3 Neural net.

Solution: The given network has three input neurons with bias and one output neuron. These form a single-layer network. The inputs are given as $[x_1, x_2, x_3] = [0.8, 0.6, 0.4]$ and the weights are $[w_1, w_2, w_3] = [0.1, 0.3, -0.2]$ with bias $b = 0.35$ (its input is always 1).

The net input to the output neuron is

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

[$n = 3$, because only 3 input neurons are given]

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3$$
$$= 0.35 + 0.8 \times 0.1 + 0.6 \times 0.3 + 0.4 \times (-0.2)$$
$$= 0.35 + 0.08 + 0.18 - 0.08 = 0.53$$

(i) For binary sigmoidal activation function,

$$y = f(y_{in}) = \frac{1}{1 + e^{-y_{in}}} = \frac{1}{1 + e^{-0.53}} = 0.625$$

(ii) For bipolar sigmoidal activation function,

$$y = f(y_{in}) = \frac{2}{1 + e^{-y_{in}}} - 1 = \frac{2}{1 + e^{-0.53}} - 1$$

$$= 0.259$$

Biological Neural Network

- Has three main parts
  - Soma or cell body-where cell nucleus is located
  - Dendrites-where the nerve is connected to the cell body
  - Axon-which carries the impulses of the neuron
- Electric impulse is passed between synapse and dendrites.
- Synapse- Axon split into strands and strands terminates into small bulb like organs called as synapse.
- It is a chemical process which results in increase /decrease in the electric potential inside the body of the receiving cell.
- If the electric potential reaches a threshold value, receiving cell fires & pulse /action potential of fixed strength and duration is send through the axon to synaptic junction of the cell.
- After that, cell has to wait for a period called ***refractory period.***

Architecture of Biological Neuron

Terminology Relation Between Biological And Artificial Neuron

Biological Neuron	Artificial Neuron
Cell	Neuron
Dendrites	Weights or interconnections
Soma	Net input
Axon	Output

Characteristics of ANN

1. Speed: Faster processing , response time is in millisecond.
2. Processing: Parallel processing
3. Size and Complexity: Less size and complexity
4. Fault tolerance: Fault tolerance is the property that enables a system to continue operating properly rather than failing completely, when some part of the system fails.

When we choose ANN for ML Problem

- When instances has many attributes.
- Attribute can be discrete value, real value, vector value.
- When training time is more.
- When training data may contain noise or error.

Application:

- Driverless Car
- Speech Processing

Application of Neural Networks

- Air traffic control
- Animal behavior
- Appraisal and evaluation of property
- Betting on horse races, stock markets
- Criminal sentencing
- Complex physical and chemical process
- Data mining, cleaning and validation
- Direct mail advertisers
- Echo patterns
- Economic modeling
- Employee hiring

Contd..

- Expert consultants
- Fraud detection
- Hand writing and typewriting
- Lake water levels
- Machinery controls
- Medical diagnosis
- Music composition
- Photos and finger prints
- Recipes and chemical formulation
- Traffic flows
- Weather prediction

Brain Vs computer

Perceptrons

- One type of ANN system is based on a unit called a perceptron.
- A perceptron is a single processing unit.
- Perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise.

- The perceptron outputs 1 for instances lying on one side of the hyperplane and outputs a -1 for instances lying on the other side.

- Learning a perceptron involves choosing values for the weights ***wo, . . . , wn.***
- Therefore, the space H of candidate hypotheses considered in perceptron learning is the set of all possible real-valued weight vectors.

Representational Power of Perceptrons

- A single perceptron can be used to represent many boolean functions.
- Perceptrons can represent all of the primitive boolean functions AND, OR,NAND (1 AND), and NOR (1 OR).
- We can view the perceptron as representing a hyperplane decision surface in the n-dimensional space of instances (i.e., points).
- The perceptron outputs a 1 for instances lying on one side of the hyperplane and outputs a -1 for instances lying on the other side, as illustrated in Figure.

- The ability of perceptrons to represent AND, OR, NAND, and NOR is important because ***every boolean function can be represented by some network of*** interconnected units based on these primitives.

Perceptron Training Rule

- We are interested in learning networks of many interconnected units.
- Let us begin by understanding how to learn the weights for a single perceptron.
- The precise learning problem is to determine a weight vector that causes the perceptron to produce the correct output for each of the given training examples.
- One way to learn an acceptable weight vector is to begin with random weights.

- Then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.
- This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly.
- Weights are modified at each step according to the ***perceptron training rule, which revises the weight*** w_i associated with input x_i ***according to the rule***

- Where
- Here ***t is the target output for the current training example.***
- ***o is the output generated*** by the perceptron.
- *η is a positive constant called the* ***learning rate.***
- The role of the learning rate is to moderate the degree to which weights are changed at each step.
- It is usually set to some small value (e.g., 0.1).

Why should this update rule converge toward successful weight values?

- Suppose the training example is correctly classified already by the perceptron.
- In this case, ***(t - o) is zero, making Δw_i zero, so that no weights are updated.***

Gradient Descent and Delta Rule

- Although the perceptron rule finds a successful weight vector when the training examples are linearly separable, it can fail to converge if the examples are not linearly separable.
- If training examples are not linearly separable then we can not use perceptron training rule. In this case we can use delta rule.
- **A second training rule, called the *delta rule, is designed to*** overcome this difficulty. If the training examples are not linearly separable, the delta rule converges toward a best-fit approximation to the target concept.
- The key idea behind the delta rule is to use ***gradient descent to search the hypothesis*** space of possible weight vectors to find the weights that best fit the training examples.
- It is also important because gradient descent can serve as the basis for learning algorithms that must search through hypothesis spaces containing many different types of continuously parameterized hypotheses.

- The delta training rule is best understood by considering the task of training an unthresholded perceptron, that is, a linear unit for which the output o is given by o(x)= w.x
- In order to derive a weight learning rule for linear units specify a measure for the ***training error of a hypothesis.***

- Where D is the set of training examples, t_d ***is the target output for training example*** *d, and* o_d ***is the output of the linear unit for training example d.***

- Here E(w) is simply half the squared difference between the target output t_d and the linear unit output o_d, summed over all training examples.
- Here E is represented as a function of w, because the linear unit output o depends on this weight vector.
- Delta rule is also known as
- LMS (Least Mean Square) rule
- Adaline rule
- Widrow hoff rule

VISUALIZING THE HYPOTHESIS SPACE

- To understand the gradient descent algorithm, it is helpful to visualize the entire hypothesis space of possible weight vectors and their associated E values, as illustrated in Figure.
- Here the axes ***w_0 and w_1 represent possible values for*** the two weights of a simple linear unit.
- Here **w_0 , w_1 plane represents** the entire hypothesis space.
- The vertical axis indicates the error E relative to some fixed set of training examples.
- The error surface shown in the figure thus summarizes the desirability of every weight vector in the hypothesis space.

- The arrow shows the negated gradient at one particular point, indicating the direction in the w_0 , w_1 plane producing steepest descent along the error surface.
- Gradient descent search determines a weight vector that minimizes ***E by*** starting with an arbitrary initial weight vector.
- Then repeatedly modifying it in small steps.
- This process continues until the global minimum error is reached.