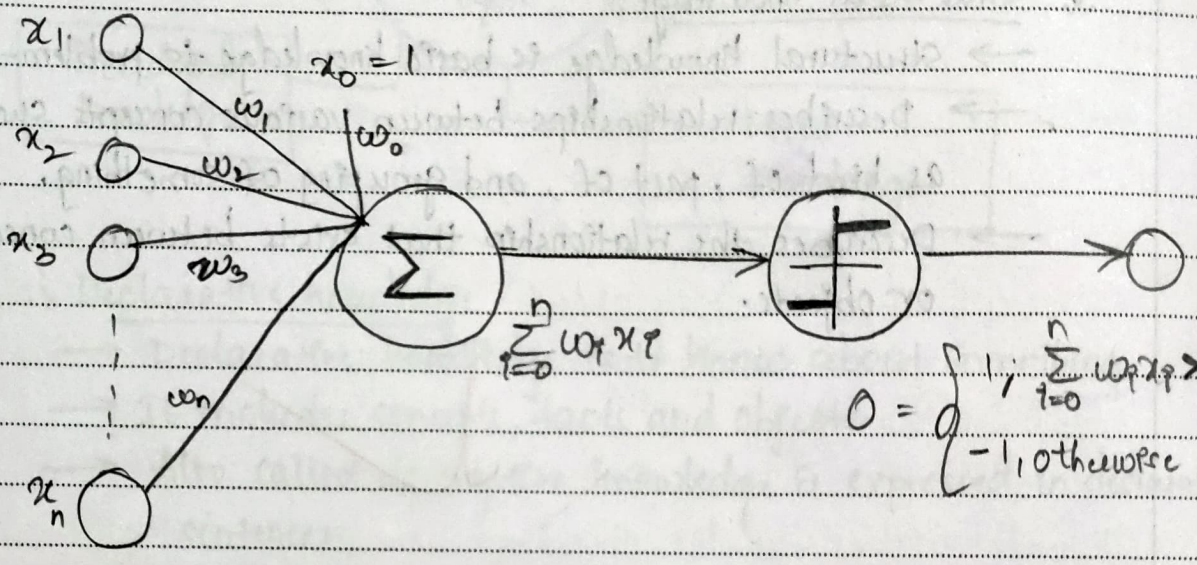
ASSIGNMENT-2

1. Explain the concept of a perceptron with a neat diagram, explain the single perceptron with its learning algorithm.

A. • One type of ANN system is based on a unit called perceptron. Perceptron is a single layer neural network.



• A 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 threshold and −1 otherwise.

• Sometimes, the perceptron function is written as,

$$o(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

where,

$$\text{sgn}(y) = \begin{cases} 1, & \text{if } y > 0 \\ -1, & \text{otherwise} \end{cases}$$

Learning algorithm

→ The learning problem is to determine a weight vector that causes the perceptron to produce the correct +1 (or) −1 output

for each of the given training examples.

To learn an acceptable weight vector

→ Begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it is misclassifying an example.

→ This process is repeated, iterating through the training examples as many times as needed till 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.

$$w_i \leftarrow w_i + \Delta w_i$$

where,

$$\Delta w_i = \eta (t - o) x_i$$

4. Derive the Gradient Descent Rule, write stochastic gradient descent algorithm for training a linear unit.

A Derivation of gradient descent rule

The gradient specifies the direction of steepest increase of E, the training rule for gradient descent is.

$$\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots \frac{\partial E}{\partial w_n} \right]$$

where,

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$$

$$\boxed{\Delta \vec{w} = -\eta \nabla E(\vec{w})}$$

This training rule can also be written in its component form.

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Calculate the gradient at each step. The vector of $\frac{\partial E}{\partial w_i}$ derivatives that forms the gradient can be obtained by differentiating E from eqt^n.

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2$$

$$= \frac{1}{2} \sum_d 2 (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)$$

$$= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \vec{w} \cdot \vec{x}_d)$$

$$\frac{\partial E}{\partial w_i} = \sum_d (t_d - o_d)(-x_{i,d})$$

$$\Delta w_i = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

Gradient Descent algorithm for training a linear unit

Each training example is a pair of the form $(\vec{x}, t)$, where $\vec{x}$ is a vector of input values and t is the target output value, η is the learning rate.

- Initialize each Δw_i to ~~zero~~ some random value.
- Until the termination condition is met; Do
  - Initialize each Δw_i to zero
  - For each linear unit weight w_i, Do

$$\Delta w_i \leftarrow \Delta w_i + \eta (t - o) x_i$$

  - For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$

6. Explain concept of entropy and information gain with formula and example. Describe ID3 algorithm with example.

A.
- Entropy is related to randomness in the information being processed in your machine learning project
- A high entropy means low information gain, and a low entropy means high information gain.
- Information gain can be thought as the purity in the system : the amount of clean knowledge available in a system.

$$\text{entropy} = -(p(0) * \log P(0)) + (P(1) * \log P(1))$$

Example: class 0 = 45/100
class 1 = 55/100.

```
entropy = -(class0 * log2(class0)) + (class1 * log2(class1))
print ("entropy - %0.3f %%", entropy)
```

O/P: entropy = 0.993

ID3 algorithm

Step①: Calculate entropy of the target.

$$\text{Entropy}(\text{PlayGolf}) = \text{Entropy}(5,9)$$
$$= \text{Entropy}(0.36, 0.64)$$
$$= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$
$$= 0.94.$$

Step②: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the information gain or decrease in entropy.

		PlayGolf	
		Yes	No.
Out look	Sunny	3	2
	Overcast	4	0
	Rainy	2	3.

Gain = 0.247

		PlayGolf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1

Gain = 0.029

		PlayGolf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

Gain = 0.152

		PlayGolf	
		Yes	No
Windy	False	6	2
	True	3	3

Gain = 0.048.

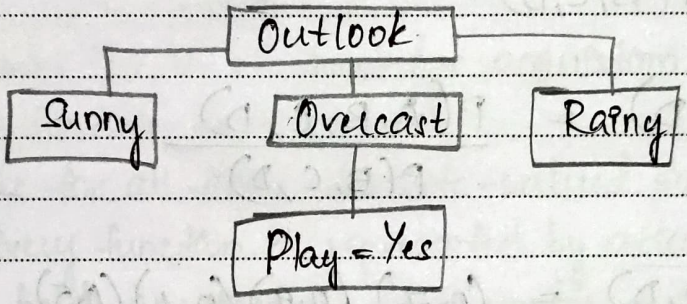
$$\text{Gain}(T,X) = \text{Entropy}(T) - \text{Entropy}(T,X).$$

Step (3): Choose attribute with the largest information as the decision node, divide the dataset by its branches and repeat the process on every branch.

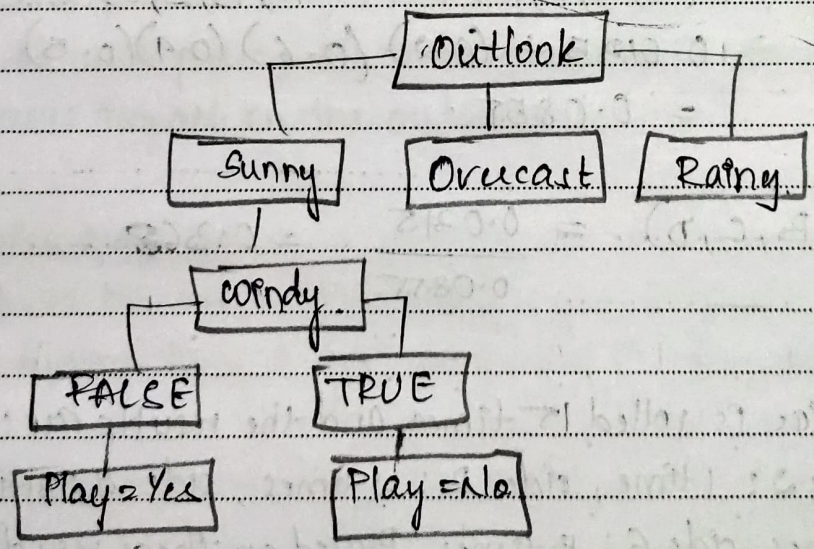
		Play Golf	
		Yes	No
Outlook.	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

Step (4): A branch with entropy 0 is a leaf node.

(a)

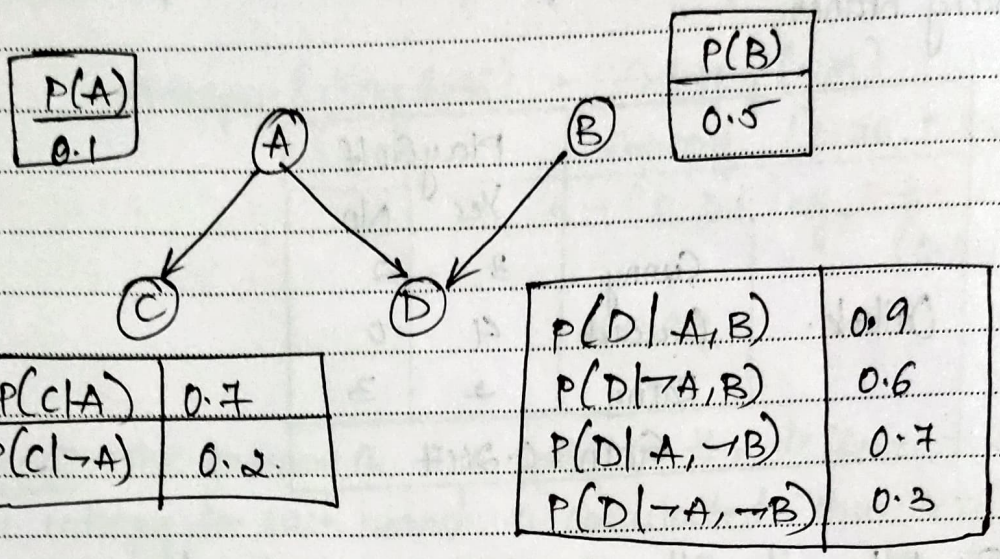


(b) A branch with entropy more than 0 needs further splitting



Step (5): The ID3 algorithm is run recursively on non-leaf branches, until all data is classified.

10. Consider the following Bayesian Network containing four Boolean random variables



Compute $P(\neg A | B, C, D)$

A. $$P(\neg A | B, C, D) = \frac{P(\neg A, B, C, D)}{P(B, C, D)}$$

$$P(\neg A, B, C, D) = (0.7)(0.9)(0.1)(0.5)$$
$$= 0.0315$$

$$P(B, C, D) = P(A, B, C, D) + P(\neg A, B, C, D)$$
$$= 0.0135 + (0.2)(0.6)(0.9)(0.5)$$
$$= 0.0855$$

Therefore

$$P(\neg A | B, C, D) = \frac{0.0315}{0.0855} = 0.368$$

9. A 6-sided dice is rolled 15 times and the results are: side 1: 0 times, side 2: 1 time, side 3: 2 times, side 4: 3 times, side 5: 4 times, side 6: 5 times. Based on these results, what is the probability of side 3 coming up when using Add-1 Smoothing?

7. $\frac{P(\text{side 3 on top})}{\text{Total probability}} = \frac{2+9}{1+2+3+4+5+6}$

Add-1 Smoothing performed.

$= \frac{2+1}{1+2+3+4+5+6} = \frac{3}{21} = \frac{1}{7}$

8. Discuss hypothesis space search and inductive bias in decision tree learning. What are issues of learning decision trees.

A. Hypothesis space search

- ID3 climbs the hill of knowledge acquisition by searching the space of feasible decision trees.
- It looks for all finite discrete-valued functions in the whole space. Every function is represented by atleast one tree.
- It only holds one theory. It is unable to inform us how many more feasible options exist.
- It's possible to get stranded on local optima.
- At each phase, all training examples are used. Errors have a lower impact on the outcome.

Inductive Bias

A set of assumptions made by an algorithm in order to perform induction into a general model of the domain.

(a) Approximate Inductive Bias: "Shorter trees are preferred over longer trees"

(b) A shorter tree is preferred over longer tree. Trees that place high information gain attributes close to the root are preferred over that do not.

Issues in decision tree learning

- Determine how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with different costs
- Improving computational efficiency.

5. Derive the Back propagation rule. Explain the following w.r.t back propagation algorithm:
(a) Convergence & Local minima
(b) Representational power of feed-forward networks
(c) Hypotheses Space search and Inductive bias
(d) Hidden layer representations
(e) Generalization, Overfitting, Stopping Criterion.

A. Back propagation rule

- Create a feed-forward network with n_{in} inputs, n_{hidden} units and n_{out} output units.
- Initialize all network weights to small random numbers.
- Until the termination condition is met, Do.
  - For each (x, t) in training examples, Do.
    - Propagate the input forward through the network:
      1. Input the instance x, to the network and compute the compute o_u of every unit in the network
    - Propagate the errors backward through the network
      2. For each network unit k, calculate its error term δ_k.

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

      3. for each network unit h, calculate its error term δ_h.

$$\delta_h \leftarrow o_h(1-o_h) \sum_{k \in o/p} w_{h,k}\, \delta_k.$$

4. Update each network weight w_{ji}.

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

where

$$\Delta w_{ji} = \eta \delta_j x_{ji}.$$

(a) Convergence & Local minima

The main idea of back propagation algorithm is to change the ~~derivation~~ derivative of the activation function so as to magnify the backward propagated error signal, thus the convergence rate can be accelerated and the local maxima can be escaped.

(b) Representational power of feed forward networks

It is the ability of a neural network to assign proper labels to a particular instance and create well defined accurate decision boundaries for that class.

(c) Hypotheses space search & Inductive bias

In back propagation, every possible assignment of networks weights represent a distinct hypotheses. This space is continuous, unlike the discrete representations such as decision trees. The hypotheses space is then an n-dimensional Euclidean space of the 'n' network weights.

The general inductive bias in back propagation can be described as "smooth interpolation between data points". In other words, given two positive data points with no negative examples between them, backpropagation tends to label the points in between as positive as well.

(d) Hidden Layer Representations

Because training examples only provide input and target output weights, the network is free to update the internal weights in order to minimize the error. This result in the network capturing properties that is not explicit in the input representation, which is a key feature to artificial neural network learning. This flexibility allows these networks to create new features not explicitly introduced by the designer, though these features must still be computable as sigmoid function units.

(e) Generalization, Overfitting and Stopping Criterion

One of the stopping conditions for backpropagation is to continue to run until the error falls below some threshold. However, this has the tendency to overfit the training examples. This occurs because the network tunes the weights to fit the training data explicitly and not the general distribution of the examples.

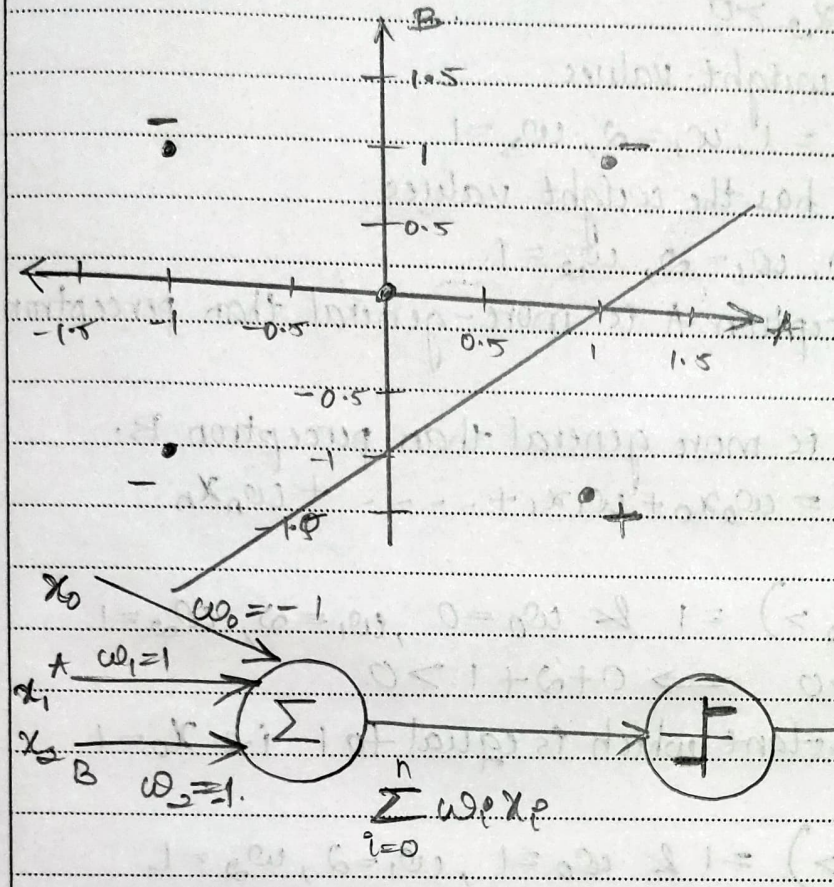
2. Design a two-input perceptron that implements the boolean function $A \wedge \neg B$. Design a 2-layer network of perceptron's that implements A XOR B.

A. The perceptron has two input A, B and constant 1.

A	B	$\neg B$	$A \wedge \neg B$
0 (-1)	0 (-1)	1	0 (-1)
0 (-1)	1	0 (-1)	0 (-1)
1	0 (-1)	1	1
1	1	0	0 (-1)

The values of A & B are 1 (true) (or) -1 (or) 0 for false.

Decision surfaces



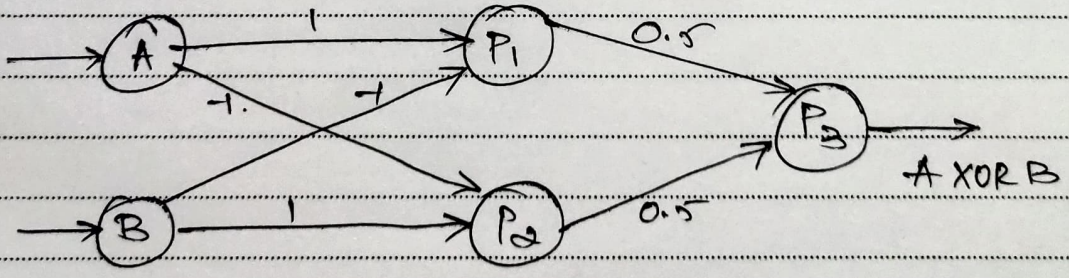
* The line crosses the A axis at 1 and B axis at -1

* The weights are
$w_0 = 1$
$w_1 = 1$, $w_2 = -1$

A XOR B cannot be calculated by a single perceptron, so build a two-layer network of perceptrons.

* Express A XOR B in terms of other logical connectives

$$A \text{ XOR } B = (A \wedge \neg B) \vee (\neg A \wedge B)$$



3. Consider two perceptrons defined by the threshold expression

$w_0 + w_1 x_1 + w_2 x_2 > 0$

Perceptron A has weight values

$w_0 = 1, w_1 = 2, w_2 = 1$

and perceptron B has the weight values

$w_0 = 0, w_1 = 2, w_2 = 1.$

True or False? Perceptron A is more-general than perceptron B.

A. True, perceptron A is more general than perceptron B.

$\Rightarrow O(x_1, \ldots, x_n) = w_0 x_0 + w_1 x_1 + \ldots + w_n x_n$

$\Rightarrow B(\langle x_1, x_2 \rangle) = 1$ & $w_0 = 0, w_1 = 2, w_2 = 1$

$0 + 2x_1 + x_2 > 0 \Rightarrow 0 + 2 + 1 > 0.$

where, x_0 is constant which is equal to 1 i.e $x_0 = 1$

$\Rightarrow A(\langle x_1, x_2 \rangle) = 1$ & $w_0 = 1, w_1 = 2, w_2 = 1.$

$1 + 2x_1 + x_2 > 0 \Rightarrow 1 + 2 + 1 > 0.$

Hence perceptron A is more general than perceptron B because every instance of x_1 & x_2 that satisfies perceptron B also satisfies perceptron A.