Question 3: Derive the perceptron training rule and gradient descent training rule for a single unit with output o, where $o = w0 + w1x1 + w1x\ 2\ 1 + + wnxn + wnx\ 2\ n$. What are the advantages of using gradient descent training rule for training neural networks over the perceptron training rule?

Answers :

a). <u>DERIVATION OF PERCEPTRON TRAINING RULE:</u>

Output o = $w0 + w1x1 + w1x1^2 + w2x2 + w2x2^2 \ldots\ldots. + wnxn + wnxn^2$

According to perceptron training rule , weights are modified at each step , which revises the weight w_i associated with input x_i as:

$$wi = wi + \Delta wi$$

$$\Delta Wi = \eta (y - 0)\, x_i$$

In above equation, η is learning rate(a small positive constant) , y is target output and o is generated output .

We can have 3 scenarios :

1. If $y = o$: This means target output and generated output are same.
   - ⇒ $y - o = 0$
   - ⇒ $\Delta wi = 0$
   - ⇒ **No change in weights**

2. If $y > o$ **:**
   - ⇒ $y - o > 0$
   - ⇒ for positive values of xi , Δwi will be +ve

   This implies , we need to **increase the weight for +ve Xi and decrease the weight for -ve Xi , when** *target output* $>$ *generated output*

3. If $y < o$:
   - ⇒ $y - o < 0$
   - ⇒ for +ve values of xi , Δwi will be -ve

   This implies we need to **decrease the weight for +ve Xi and increase the weight for -ve Xi , when** *target output* $<$ *generated output*

Perceptron Training Rule is proven to converge in finite number of iterations to a weight vector , that correctly classifies all training examples , provided that data is <u>linearly separable.</u>

b). DERIVATION OF GRADIENT DESCENT TRAINING RULE :

Output o = $w0 + w1x1 + w1x1^2$ ……………..+ $wnxn \ + \ wnxn^2$

$$o = w_0 + \sum_{i=1}^{n} w_i x_i (1 + x_i)$$

:

We can say generated output o(x) is:

where: $f(xi) \ = \ 1 \, for \ i = 0$
$f(xi) \ = \ xi(1 + xi)$ otherwise

$$o(x) = \sum_{i=0}^{n} w_i f(x_i)$$

If y is target output , in gradient descent we can write Error $E(w)$ in terms of weight as:

$$E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - f(x))^2$$

Where E(w) is Error in weights

$(x,y) \in D$ is all data in dataset.

We can write it as activation a , when output is not thresholded:

$$activation \ a \ = \ \sum_i w_i x_i$$

Error E(w) can be rewritten as:

$$E(w) = \frac{1}{2} \sum_{(x,y) \in D} (y - a)^2$$

We can minimize the error by modifying the weights . So , partial derivative of the error with respect to weight wi as:

$$\frac{\partial E}{\partial wi} \ = \ \frac{\partial}{\partial wi} \frac{1}{2} \sum (y - a)^2$$

$$\Rightarrow \sum_{(x,y) \in D} (y - a) \frac{\partial}{\partial wi} \ - \ \sum xiwi$$

$$\Rightarrow \sum_{(x,y) \in D} (y - a)(-xi)$$

$$\boxed{\Delta wi = \sum_{(x,y) \in D} (y - a)(-xi)}$$

c).**Advantages of using gradient descent training rule for training neural networks over the perceptron training rule:**

Perceptron training rule guarantees the finite convergence only in case of Linearly Separable data.
For non linearly separable data , we have gradient descent training rule for training neural networks. Gradient Descent rule is good , because it is more robust to datasets that are not linearly separable. It only converges to the limit of local optima.

****
****

Question 2: Design a two-input perceptron that implements the boolean function $A \wedge \neg B$. Design a two-layer network of perceptrons that implements $A \oplus B$ ($\oplus$ is XOR).

Answer:
a). Two input perceptron that implements the boolean function $A \wedge \neg B$

Truth table:

A	B	$A \wedge \neg B$
0	0	0
0	1	0
1	0	1
1	1	0

As per Aw1+Bw2 >= ϴ

1. For A=0 ,B=0 output is 0

=> $Aw1+Bw2 >= \Theta$ is False
=> $\Theta > Aw1 + Bw2$
=> $\Theta > 0$ ------------------------------------Equation 1

2. For A=0 , B= 1 output is 1
=>$Aw1+Bw2 >= \Theta$ is True
=> $\Theta <= Aw1 + Bw2$
=> $\Theta <= w2$ ------------------------------------Equation 2

3. For A=1 , B= 0 output is 1
=>$Aw1+Bw2 >= \Theta$ is True
=> $\Theta <= Aw1 + Bw2$
=> $\Theta <= w1$ -----------------------------------Equation 3

4. For A=1 , B= 1 output is 0
=>$Aw1+Bw2 >= \Theta$ is False
=> $\Theta > Aw1 + Bw2$
=> $\Theta > w1+w2$ -------------------------------Equation 4

Θ =1 , w1 =1 and w2 =-1 values satisfy the above 4 equations.
Answer : perceptron unit has following values :
Θ =1 , w1 =1 and w2 = -1

b). Two-layer network of perceptrons that implements $A \oplus B$ ($\oplus$ is XOR).

Truth Table:

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

It is impossible to find the single unit perceptron, so we are going to do it with network of perceptrons:

Let us consider:

A	B	A∧B	A∨B	A⊕B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

If we look at A⊕B , it is exactly same as A∨B, except the last row,
We can write A⊕B = **(**A∨B) - (A∧B)
=> XOR = OR - AND

We can design the multilayer perceptron as:

As per Aw1+Bw2 >= ϴ

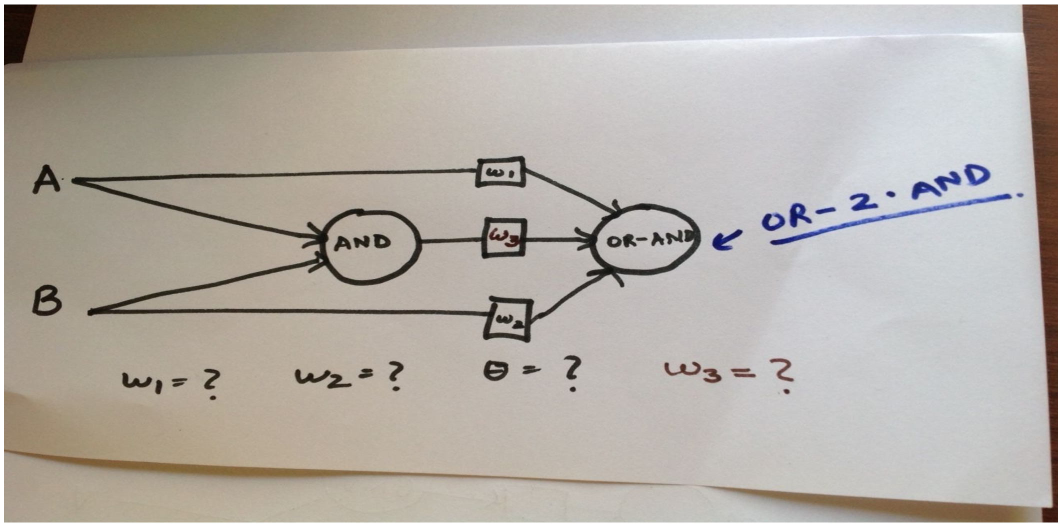
1. For A=0 ,B=0 output is 0
=> Aw1+Bw2 >= ϴ is False
=> ϴ > Aw1 +Bw2
=> ϴ > 0 ------------------------------------Equation 1

2. For A=0 , B= 1 output is 1
=>Aw1+Bw2 >= ϴ is False
=> ϴ > Aw1 +Bw2
=> ϴ > w2 ------------------------------------Equation 2

3. For A=1 , B= 0 output is 1
=>Aw1+Bw2 >= ϴ is True
=> ϴ <= Aw1 +Bw2
=> ϴ <= w1 -----------------------------------Equation 3

4. For A=1 , B= 1 output is 0
=>Aw1+Bw2 >= ϴ is False
=> ϴ > Aw1 +Bw2
=> ϴ > w1+w2 -------------------------------Equation 4

It is impossible to find the single unit perceptron, which satisfies above 4 equations.



Based on truth table ,we can design the 2 layer perceptron as above,

So we can write :

Aw1+bw2+ (A AND B)w3 >= ϴ

1. For A=0 ,B=0 output is 0
   => Aw1+bw2+ (A AND B)w3 >= ϴ is False
   => ϴ > 0

2. For A=0 ,B=1 output is 1
   => Aw1+bw2+ (A AND B)w3 >= ϴ is True
   => w2 >= ϴ

3. For A=1 ,B=0 output is 1
   => Aw1+bw2+ (A AND B)w3 >= ϴ is True
   => w1 >= ϴ

4. For A=1 ,B=1 output is 0
   => Aw1+bw2+ (A AND B)w3 >= ϴ is False
   => w1 +w2 +w3 < ϴ

So from above equations : $\Theta = 1$, $w1=1$, $w2 = 1$ satisfy the above 3 equations ,
Putting these values in equation 4 gives :
$2+w3 < 1$
$\Rightarrow w3 < -1$
$\Rightarrow w3 = -2$

So the end unit can be written as : **A XOR B = (A OR B) - 2. (A AND B)**

Answer : weights and threshold :
$w1=1$, $w2=2$ and $w3 =-2$, $\Theta = 1$

****
****

Question :
Suggest a lazy version of the eager decision tree learning algorithm ID3. What are the advantages and disadvantages of your lazy algorithm compared to the original eager algorithm?

Answer: Lazy version of ID3:

In eager decision tree algorithm ,complete decision tree is constructed and stored in training phase then to classify the incoming instances , we trace from the root to leaf by querying interior nodes through the route.

While in lazy version of ID3 algorithm , it does not construct the complete decision tree and stores. Here training phase, is replaced by one that also considers query instance in order to keep the produced DT concise. So it constructs only one branch for the given instance by making use of the heuristic characteristic, in which the class labeled on the leaf node is that of the given test instance.

Advantages of using lazy algorithm as compared to original ID3 Algorithm:

1. Much less storage space is required , since the complete decision tree is not constructed and stored in training phase , rather one branch of decision tree is created for each distinct incoming classification query.
2. Missing values or noisy data can be handled by simply avoiding branching on those values.
3. In theory, we would like to select the best decision tree for each test instance, i.e., pick the best tree from all possible trees. While in observation: only the path the test instance takes really matters. We don’t need to search or build all

possible trees, but at possible paths. So Lazy algorithm seems more robust practically.

References:
1. Machine Learning Book by Tom Mitchell
2.http://robotics.stanford.edu/~ronnyk/lazyDT-talk.pdf
3. Udacity lectures
4. https://www.computer.org/csdl/proceedings/icmlc/2003/7865/03/01259741.pdf