K. J. Somaiya College of Engineering, Mumbai-77

Batch: A1 Roll No.: 16010120015

Experiment No. 04

Grade: A / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Study and implementation of Delta learning rule

Objective: To write a program to implement Delta learning rule for a given training data.

Expected Outcome of Experiment:
CO3 : To Understand perceptron's and counter propagation networks

Books/ Journals/ Websites referred:

- J.S.R.Jang, C.T.Sun and E.Mizutani, "Neuro-Fuzzy and Soft Computing", PHI, 2004, Pearson Education 2004.
- Davis E.Goldberg, "Genetic Algorithms: Search, Optimization and Machine Learning", Addison Wesley, N.Y., 1989.
- S. Rajasekaran and G.A.V.Pai, "Neural Networks, Fuzzy Logic and Genetic Algorithms", PHI, 2003.
- http://library.thinkquest.org/C007395/tqweb/history.html

Pre Lab/ Prior Concepts:

Neural networks, sometimes referred to as connectionist models, are parallel-distributed models that have several distinguishing features-

1) A set of processing units;
2) An activation state for each unit, which is equivalent to the output of the unit;
3) Connections between the units. Generally, each connection is defined by a weight w_{jk} that determines the effect that the signal of unit *j* has on unit *k;*
4) A propagation rule, which determines the effective input of the unit from its external inputs;
5) An activation function, which determines the new level of activation based on the effective input and the current activation;
6) An external input (bias, offset) for each unit;
7) A method for information gathering (learning rule);
8) An environment within which the system can operate, provide input signals and, if necessary, error signals.

Delta learning rule and its application

Developed by *Widrow* and *Hoff*, the delta rule, is one of the most common learning rules. It depends on supervised learning.
This rule states that the modification in sympatric weight of a node is equal to the multiplication of error and the input.
In Mathematical form the delta rule is as follows:

$$\Delta w = \eta (t - y) x_i$$

Application:

The delta rule in an artificial neural network is a specific kind of backpropagation that assists in refining the machine learning/artificial intelligence network, making associations among input and outputs with different layers of artificial neurons.

Implementation Details:

```
import numpy as np
import TableIt

learning_rate = 0.7
bias = float(input("Enter Bias value: "))
weights=[]
for i in range(2):
    w = float(input(f"Enter Weight {i+1}: "))
    weights.append(w)
x = [[0,0],[0,1],[1,0],[1,1]]
desired_output = [0,0,0,1]
epoch = int(input("Enter number of epochs: "))

print(f"Weights: {weights}")
print(f"Bias: {bias}")
print(f"Learning Rate: {learning_rate}")
print(f"Desired Output: {desired_output}")
print(f"No. of epochs: {epoch}")


for e in range(epoch):
    print(f"\n>> Starting with epoch {e+1}")
    output = []
    for i in range(4):
        v = np.dot(weights,x[i])
        f_net = v+bias
        if f_net >= 0.5:
            output.append(1)
        else:
            output.append(0)

        if output[i] == desired_output[i]:
            pass
        else:
            error = desired_output[i] - output[i]
            # Update bias
            bias = bias + learning_rate * error
            # Update weights
            for k in range(2):
                weights[k] += learning_rate * error * x[i][k]
```

```
    if np.array_equal(output,desired_output):
        print("\tGot the Desired Output!")
        print(f"\t>Output: {output}")
        print(f"\t>Weights: {weights}")
        print(f"\t>Bias: {bias}")
        table = [
            ["x1","x2","y1"],
            ["0","0",output[0]],
            ["0","1",output[1]],
            ["1","0",output[2]],
            ["1","1",output[3]],
        ]
        TableIt.printTable(table, useFieldNames=True)
    else:
        print("\tOutput differs from Desired Output")
        print(f"\t>Output: {output}")
        print(f"\t>Weights: {weights}")
        print(f"\t>Bias: {bias}")
        continue
```

Output:

```
Weights: [0.2, 0.5]
Bias: 0.4
Learning Rate: 0.7
Desired Output: [0, 0, 0, 1]
No. of epochs: 10

>> Starting with epoch 1
        Output differs from Desired Output
        >Output: [0, 1, 0, 0]
        >Weights: [0.8999999999999999, 0.5]
        >Bias: 0.4

>> Starting with epoch 2
```

```
        Output differs from Desired Output
        >Output: [0, 1, 1, 0]
        >Weights: [0.8999999999999999, 0.5]
        >Bias: -0.29999999999999993

>> Starting with epoch 3
        Output differs from Desired Output
        >Output: [0, 0, 1, 0]
        >Weights: [0.8999999999999999, 1.2]
        >Bias: -0.29999999999999993

>> Starting with epoch 4
        Output differs from Desired Output
        >Output: [0, 1, 0, 0]
        >Weights: [1.5999999999999999, 1.2]
        >Bias: -0.29999999999999993

>> Starting with epoch 5
        Output differs from Desired Output
        >Output: [0, 1, 1, 0]
        >Weights: [1.5999999999999999, 1.2]
        >Bias: -0.9999999999999998

>> Starting with epoch 6
        Output differs from Desired Output
        >Output: [0, 0, 1, 0]
        >Weights: [1.5999999999999999, 1.9]
        >Bias: -0.9999999999999998

>> Starting with epoch 7
        Output differs from Desired Output
        >Output: [0, 1, 0, 1]
        >Weights: [1.5999999999999999, 1.2]
        >Bias: -1.6999999999999997

>> Starting with epoch 8
        Got the Desired Output!
        >Output: [0, 0, 0, 1]
        >Weights: [1.5999999999999999, 1.2]
        >Bias: -1.6999999999999997
+-------------+
| x1 | x2 | y1 |
```

```
+----+----+----+
| 0  | 0  | 0  |
| 0  | 1  | 0  |
| 1  | 0  | 0  |
| 1  | 1  | 1  |
+-------------+

>> Starting with epoch 9
        Got the Desired Output!
        >Output: [0, 0, 0, 1]
        >Weights: [1.5999999999999999, 1.2]
        >Bias: -1.6999999999999997
+-------------+
| x1 | x2 | y1 |
+----+----+----+
| 0  | 0  | 0  |
| 0  | 1  | 0  |
| 1  | 0  | 0  |
| 1  | 1  | 1  |
+-------------+

>> Starting with epoch 10
        Got the Desired Output!
        >Output: [0, 0, 0, 1]
        >Weights: [1.5999999999999999, 1.2]
        >Bias: -1.6999999999999997
+-------------+
| x1 | x2 | y1 |
+----+----+----+
| 0  | 0  | 0  |
| 0  | 1  | 0  |
| 1  | 0  | 0  |
| 1  | 1  | 1  |
+-------------+
```

Conclusion: Thus, we have successfully implemented Delta learning algorithm of Neural Network.

Post Lab Descriptive Questions :

1. Comparison of Neural network and algorithmic computation

Neural network	algorithmic computation
Programming is broken into small, unambiguous steps	Neural networks are trained
Algorithms must be already known and understood in order to obtain a result	Can obtain results when there is no "solution"
Errors in results are always software or hardware based	Errors are based on faulty training data sets

2. Explain different types of supervised techniques.

Supervised machine learning is immensely helpful in solving real-world computational problems. The algorithm predicts outcomes for unforeseen data by learning from labeled training data. Therefore, it takes highly-skilled data scientists to build and deploy such models.

Types of Supervised Learning

1. Regression

In regression, a single output value is produced using training data. This value is a probabilistic interpretation, which is ascertained after considering the strength of correlation among the input variables. For example, regression can help predict the price of a house based on its locality, size, etc.

2.Classification

It involves grouping the data into classes. If you are thinking of extending credit to a person, you can use classification to determine whether or not a person would be a loan defaulter. When the supervised learning algorithm labels input data into two distinct classes, it is called binary classification. Multiple classifications means categorizing data into more than two classes.

3. Naive Bayesian Model

The Bayesian model of classification is used for large finite datasets. It is a method of assigning class labels using a direct acyclic graph. The graph comprises one parent node and multiple children nodes. And each child node is assumed to be independent and separate from the parent.

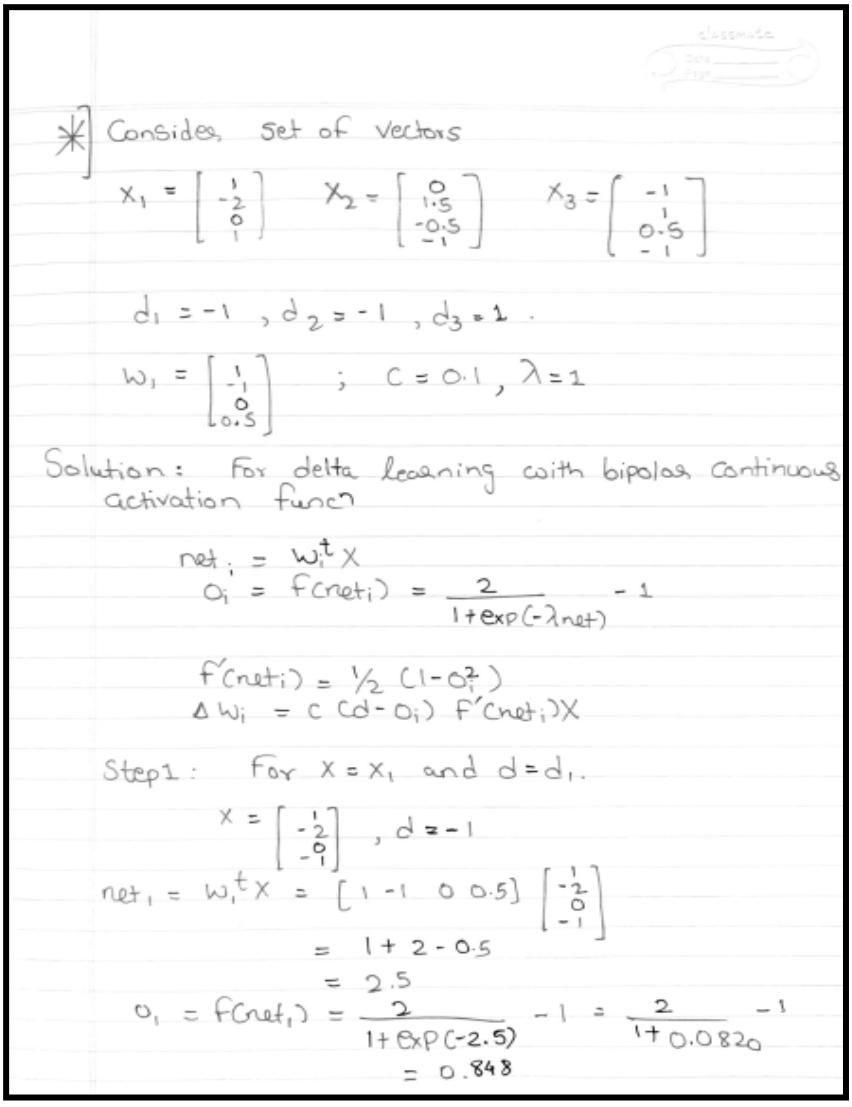
4. Random Forest Model

The random forest model is an ensemble method. It operates by constructing a multitude of decision trees and outputs a classification of the individual trees.

3. Take any example and show the different steps of Delta Algorithm.

Ans:

.



* Consider set of vectors

$X_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$ $X_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}$ $X_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$

$d_1 = -1$, $d_2 = -1$, $d_3 = 1$.

$W_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$; $C = 0.1$, $\lambda = 1$

Solution: For delta learning with bipolar continuous activation func^n

$net_i = W_i^t X$

$O_i = f(net_i) = \frac{2}{1+\exp(-\lambda net)} - 1$

$f'(net_i) = \frac{1}{2}(1 - O_i^2)$

$\Delta W_i = C(d - O_i) f'(net_i) X$

Step1 : For $X = X_1$ and $d = d_1$.

$X = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$, $d = -1$

$net_1 = W_1^t X = \begin{bmatrix} 1 & -1 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}$

$= 1 + 2 - 0.5$

$= 2.5$

$O_1 = f(net_1) = \frac{2}{1+\exp(-2.5)} - 1 = \frac{2}{1+0.0820} - 1$

$= 0.848$

$f'(net_1) = \frac{1}{2}(1-O_1^2) = \frac{1}{2}(1-0.848^2)$

$= 0.140$

$\Delta w_1 = c(d-O_1) f'(net_1) X$

$= 0.1(-1-0.948)(0.14)\begin{bmatrix}1\\-2\\0\\-1\end{bmatrix}$

$= -0.026\begin{bmatrix}1\\-2\\0\\-1\end{bmatrix}$

$= \begin{bmatrix}-0.026\\0.052\\0\\0.026\end{bmatrix}$

$w_2 = w_1 + \Delta w_1$

$= \begin{bmatrix}1\\-1\\0\\0.5\end{bmatrix} + \begin{bmatrix}-0.026\\0.052\\0\\0.026\end{bmatrix}$

$= \begin{bmatrix}0.974\\-0.948\\0\\0.526\end{bmatrix}$

Step 2 : $X = X_2$ and $d = d_2$

$X = \begin{bmatrix}0\\1.5\\-0.5\\-1\end{bmatrix}$, $d = -1$

$net_2 = w_2^t X = [0.974 \; -0.948 \; 0 \; 0.526]\begin{bmatrix}0\\1.5\\-0.5\\-1\end{bmatrix}$

$= -1.422 - 0.526$

$= -1.948$

$O_2 = f(net_2) = \frac{2}{1+\exp(1.948)} - 1$

$= \frac{2}{8.0146} - 1$

$= -0.75$

$f'(net_2) = \frac{1}{2}(1-O_2^2)$

$= \frac{1}{2}(1-(-0.75)^2)$

$= 0.2187$.

$\Delta w_2 = c(d-O_2) f'(net_2) X$

$= (0.1)(-1-(-0.75)) \times (0.2187)\begin{bmatrix}0\\1.5\\-0.5\\-1\end{bmatrix}$

$= (-0.00546)\begin{bmatrix}0\\1.5\\-0.5\\-1\end{bmatrix}$

$= \begin{bmatrix}0\\-0.00819\\0.00273\\0.00546\end{bmatrix}$

$w_3 = w_2 + \Delta w_2 = \begin{bmatrix}0.974\\-0.948\\0\\0.526\end{bmatrix} + \begin{bmatrix}0\\-0.00819\\0.00273\\0.00546\end{bmatrix}$

$= \begin{bmatrix}0.974\\-0.956\\0.0027\\0.5315\end{bmatrix}$

Step 3: $X = X_3$, $d = d_3$

$$X = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}, \quad d = 1$$

$$net_3 = W_3^t X = \begin{bmatrix} 0.974 & -0.956 & 0.0027 & 0.531 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

$$= -0.974 - 0.956 + 0.00135 - 0.5315$$

$$= -2.462$$

$$O_3 = f(net_3) = \frac{2}{1+\exp(2.462)} - 1$$

$$= \frac{2}{12.728} - 1$$

$$= -0.843$$

$$f'(net_3) = \tfrac{1}{2}(1 - O_3^2)$$

$$= \tfrac{1}{2}(1 - (-0.843)^2)$$

$$= 0.145.$$

$$\Delta W_3 = (0.1)(d - O_3) f'(net_3) X$$

$$= 0.1\,(1 - (-0.843))(0.145) \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

$$= 0.0267 \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0267 \\ 0.0267 \\ 0.0133 \\ -0.0267 \end{bmatrix}$$

$$W_4 = W_3 + \Delta W_3 = \begin{bmatrix} 0.974 \\ -0.956 \\ 0.0027 \\ 0.5315 \end{bmatrix} + \begin{bmatrix} -0.0267 \\ 0.0267 \\ 0.0133 \\ -0.0267 \end{bmatrix} = \begin{bmatrix} 0.947 \\ -0.929 \\ 0.016 \\ 0.504 \end{bmatrix}$$

Date: 15/10/2022 **Signature of faculty in-charge .**