Batch: A2 **Roll No.: 16010121045**

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:

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

The Delta Rule uses the difference between target activation (i.e., target output values) and obtained activation to drive 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:

For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference. The change in weight from u_i to u_j is: $dw_{ij} = r^* ai * ej$.

where r is the learning rate, ai represents the activation of ui and ej is the difference between the expected output and the actual output of uj. If the set of input patterns form an independent set then learn arbitrary associations using the delta rule.

It has seen that for networks with linear activation functions and with no hidden units. The error squared vs. the weight graph is a paraboloid in n-space. Since the proportionality constant is negative, the graph of such a function is concave upward and has the least value. The vertex of this paraboloid represents the point where it reduces the error. The weight vector corresponding to this point is then the ideal weight vector.

We can use the delta learning rule with both single output unit and several output units.

While applying the delta rule assume that the error can be directly measured.

The aim of applying the delta rule is to reduce the difference between the actual and expected output that is the error.

Implementation Details:

1. **Implement Delta learning rule for the example given in Zurada book and verify the accuracy of the result.**

```
import numpy as np
import math


def bipolarBinarySigmoid(x):
    # return ((2*(1+math.exp(-x))**-1)-1)
    return (((2)/(1+math.exp(-x)))-1)
```

```
def deltaLearning(x):
    print("Input : ", x)
    w1 = np.array([1, -1, 0, 0.5])
    net = np.dot(w1, x)
    print("Net Input : ", net)
    o = float(bipolarBinarySigmoid(net))
    print("Output : ", o)
    f_net = 0.5 * (1 - o ** 2)
    print("f'(net) :- ", f_net)
    c = 0.1
    desired_output = 1
    print("Desired output :- ", desired_output)
    output_w = w1 + c * (desired_output - o) * f_net * x
    print(output_w, '\n')


x1 = [1, -2, 0, -1]
x2 = [0, 1.5, -0.5, -1]
x3 = [-1, 1, 0.5, -1]

x1 = np.array(x1)
x2 = np.array(x2)
x3 = np.array(x3)


deltaLearning(x1)
deltaLearning(x2)
```

2. **Compute the error for different learning rate**

Learning rate/constant (c) = 0.1

Output:

```
python3 -u "/Users/pargatsinghdhanjal/Desktop/Soft Computing/exp4.py"
❯ python3 -u "/Users/pargatsinghdhanjal/Desktop/Soft Computing/exp4.py"
Input :  [ 1 -2  0 -1]
Net Input :  2.5
Output :  0.8482836399575131
f'(net) :-  0.14020743309021616
Desired output :-  1
[ 1.00212718 -1.00425435  0.          0.49787282]

Input :  [ 0.   1.5 -0.5 -1. ]
Net Input :  -2.0
Output :  -0.7615941559557649
f'(net) :-  0.20998717080701307
Desired output :-  1
[ 1.         -0.94451317 -0.01849561  0.46300878]

Input :  [-1.   1.   0.5 -1. ]
Net Input :  -2.5
Output :  -0.8482836399575129
f'(net) :-  0.14020743309021633
Desired output :-  1
[ 0.97408569 -0.97408569  0.01295716  0.47408569]
```

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

Post Lab Descriptive Questions :

1. Comparison of Neural network and algorithmic computation

   Ans:

   Neural Networks

   - A neural network consists of a series of algorithms that endeavor to determine and identify patterns. It works similarly to how a human brain's neural network works. In

algorithms, a neural network refers to a network of neurons, where a neuron is a mathematical function used to collect as well as to classify data from a given model.

- A neural network is a mathematical model that is capable of solving and modeling complex data patterns and prediction problems. Neural network algorithms are developed by replicating and using the processing of the brain as a basic unit. As it has the capability of mimicking the functionality and operation of the human brain, it can do anything that a human can do.

Neural networks have many applications:

- Pattern recognition such as spam detection in email and cancer detection in the human body
- Prediction such as weather forecasting and stock market prediction

Genetic Algorithm

- The genetic algorithm is search heuristic which is inspired by Darwin's theory of natural evolution. It reflects the process of the selection of the fittest element naturally.
- Genetic algorithms are generally used for search-based optimization problems, which are difficult and time-intensive to solve by other general algorithms. Optimization problems refer to either maximization or minimization of the objective function. The genetic algorithm aims to find the optimal or near-optimal solution to the optimization problem.

- First of all, genetic algorithms are search-based optimization algorithms used to find optimal or near-optimal solutions for search problems and optimization problems. Neural networks, on the other hand, are mathematical models that map between complex inputs and outputs. They can classify elements that are not previously known.
- Genetic algorithms usually perform well on discrete data, whereas neural networks usually perform efficiently on continuous data.
- Genetic algorithms can fetch new patterns, while neural networks use training data to classify a network.
- A genetic algorithm doesn't always require derivative information to solve a problem.

- Genetic algorithms calculate the fitness function repeatedly to get a good solution. That's why it takes a good amount of time to compute a reasonable solution. Neural networks, in general, take much less time for the classification of new input.

2. Explain different types of supervised techniques.

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. In logistic regression, the output has discrete values based on a set of independent variables. This method can flounder when dealing with non-linear and multiple decision boundaries. Also, it is not flexible enough to capture complex relationships in datasets.

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 mean categorizing data into more than two classes.

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. As the model for supervised learning in ML helps construct the classifiers in a simple and straightforward way, it works great with very small data sets. This model draws on common data assumptions, such as each attribute is independent. Yet having such simplification, this algorithm can easily be implemented on complex problems.

Decision Trees

A decision tree is a flowchart-like model that contains conditional control statements, comprising decisions and their probable consequences. The output relates to the labelling of unforeseen data.

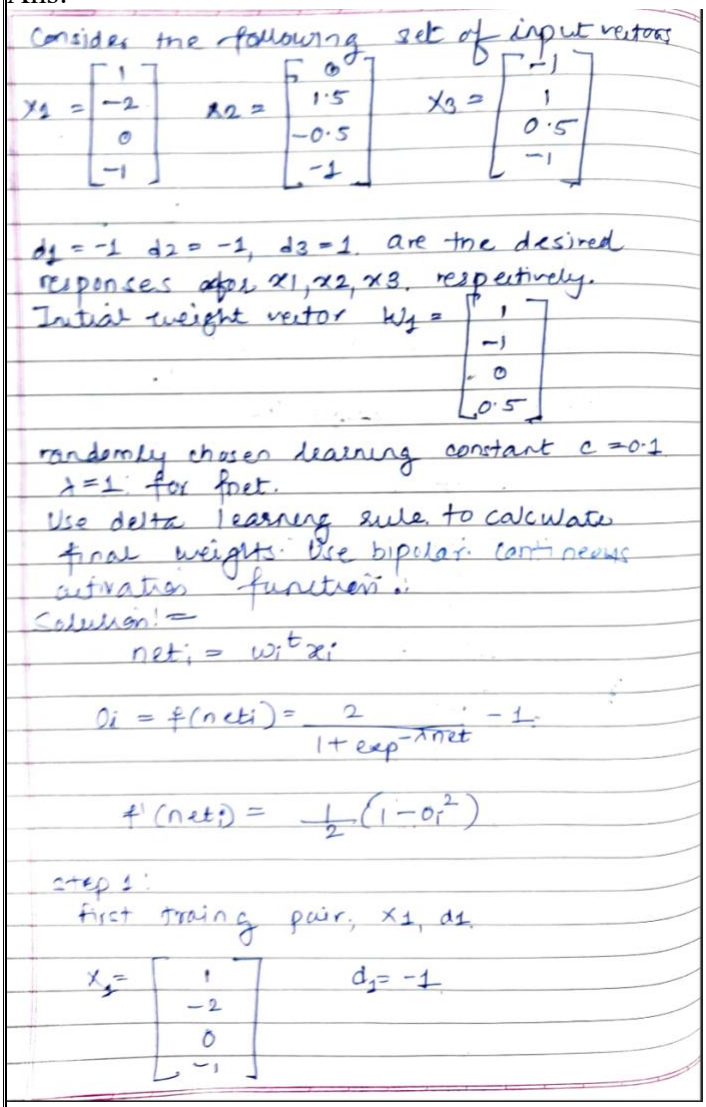
In the tree representation, the leaf nodes correspond to class labels, and the internal nodes represent the attributes. A decision tree can be used to solve problems with discrete attributes as well as boolean functions. Some of the notable decision tree algorithms are ID3 and CART.

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. Suppose you want to predict which undergraduate students will perform well in GMAT – a test taken for admission into graduate management programs. A random forest model would accomplish the task, given the demographic and educational factors of a set of students who have previously taken the test.

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

Ans:



$$net_1 = w_1^t \cdot x_1$$

$$= \begin{bmatrix} 1 & -1 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} = 2.5$$

$$o^1 = f(net^1) = \frac{2}{1 + exp^{-1 \times 2.5}} - 1 = 0.848$$

$$f'(net^1) = \frac{1}{2}[1 - (o^1)^2] =$$

$$= \frac{1}{2}[1 - 0.848^2]$$

$$= 0.140$$

$$w^2 = c \cdot (d_1 - o^1) f'(net^1) x_1 + w^1$$

$$= (0.1) \times (-1 - 0.848) \times 0.140 \times \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$$

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

Step 2:

Input is vector x_2, weight vector is w^2

$$net^2 = w_2^t \times 2$$

$$= \begin{bmatrix} 0.974 & -0.948 & 0 & 0.526 \end{bmatrix} \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}$$

$$= -1.948$$

$$o^2 = f(net^2) = \frac{2}{1+\exp^{-\lambda net_2}} - 1 = -0.75$$

$$f'(net^2) = \frac{1}{2}[1-(o^2)^2]$$

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

$$= 0.218$$

$$w^2 = c\cdot(d_2 - o^2)\cdot f'(net^2)\, x_2 + w^2$$

$$(0.1)\times(-1+0.75)\times 0.218 \times \begin{bmatrix}0\\1.5\\-0.5\\-1\end{bmatrix} + \begin{bmatrix}0.974\\-0.948\\0\\0.526\end{bmatrix}$$

$$= \begin{bmatrix}0.974\\-0.956\\0.002\\0.531\end{bmatrix}$$

Step 3 :

Input is x_3, weight vector is w^3

$$net^3 = w^{3t} \cdot x_3$$

$$= [0.974, -0.956, 0.002, 0.531] \begin{bmatrix}-1\\1\\0.5\\-1\end{bmatrix}$$

$$= -2.46$$

$$o^3 = f(net^3) = \frac{2}{1+\exp^{-\lambda net^3}} - 1 = -0.842$$

$$f'(net^3) = \frac{1}{2}[1 - (o^3)^2]$$

$$= \frac{1}{2}[1 - (-0.842)^2]$$

$$= 0.145$$

~~$f'(net^3)$~~ =

$$w^4 = c(d_3 - o^3) \cdot f'(net^3) \cdot x_3 + w^3$$

$$= (0.1) \times (1 - (-0.842) \times 0.145 \times \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix} + \begin{bmatrix} 0.974 \\ -0.956 \\ 0.002 \\ 0.531 \end{bmatrix}$$

$$= \begin{bmatrix} 0.947 \\ -0.929 \\ 0.016 \\ 0.505 \end{bmatrix}$$

Date: **Signature of faculty in-charge**