**2014555534 - UĞUR DURAN**

**PERCEPTRON**

**IMPLEMENT A FROM SCRATCH AND WITH OUT ANY FRAMEWORK**

**USING PERCEPTRON**

**1.INTRODUCTION**

**Perceptron :** In machine learning, the sensor is an algorithm for supervised learning of binary classifiers. A binary classifier is a function that can decide whether an entry represented by a numbers vector belongs to a particular class.

**A picture containing object, clock, drawing

Description automatically generated**

In this function, as shown above, W value is defined as the weight parameter, x value as input, b value as bias and y value as the output of the network. Here, our x input value gives the cat matrix of the cat picture, for example if we know the cat pictures, and y gives the score about how similar this picture is to the cat. Our parameters, W weight and b bias are used to improve this output score. In this sense, the main thing we do in multilayered neural networks or deep learning is to calculate the w and b parameter values ​​that will give the best score for our model.

Implementing a simple percepron and training it with gradient descent optimizer for linear regression task with out any frameworks such as Keras etc.

**2.MATERIAL AND METHOD**

**a - Dataset Description :** Data set, I use 3 different input data consisting of 200 point entry. Each entry has a point in the x and y coordinates. I pull this data from "data.csv" file and get results with python encoding. The data in my "Data.csv" file are real variables. Data from the file I took as input means "x", "w" and "b". I get the results as output in the variable "y". As a result of these data use, I have observed the distribution of the data in the linear plane.

**b - Theoretical details on the topic :**

**A close up of a map

Description automatically generated**

Single layer artificial neural networks consist only of input and output (y) layers. Output units are connected to all input units (X) and each connection has a weight (W).

In these networks, there is a threshold value (Φ) that prevents the values of the process elements and thus the output of the network from zero, and its value is always 1. The output of the network is found by adding the weighted input values with the threshold value. The output of the network is calculated by passing through an activation function with this input. In single layer sensors, the output function is a linear function and takes the values 1 or -1. If the output is 1, it is accepted to the second class if it is -1 to the first class. It then visualizes these accepted values on a linear plane.

**A picture containing clock

Description automatically generated**

**c – 1-Python Code : main2.py**

**--->** import libraries here.For mathematical calculation import “numpy”library and for graphical representation library using “matplotlib”.

****

**--->** using a self method defination a learning\_rate and bios\_learning\_rate value.

**A screenshot of a cell phone

Description automatically generated**

**--->** Calculation perceprton formula and Show result by “y”.

**A close up of a logo

Description automatically generated**

**--->** and then calculate result hold array b ,w1 and w2.

**A close up of a logo

Description automatically generated**

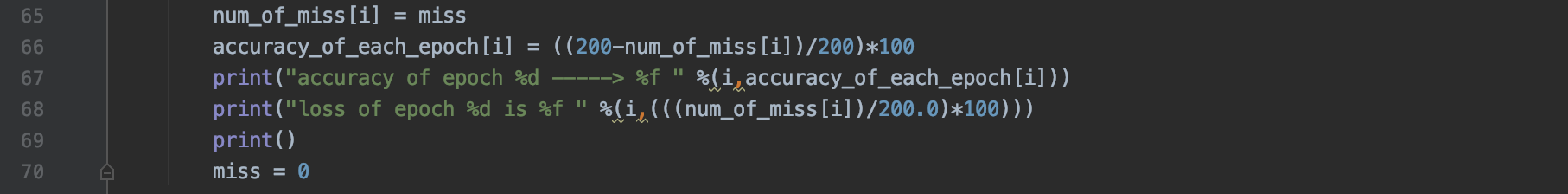
**--->** create a array groups for num\_of\_miss and accuracy\_of\_each\_epoch. X and Y values compare to lineer line before show up or down.

**A close up of a logo

Description automatically generated**

**--->**importing data set “data.csv”.And then returned prediction value starting by if loop. reads the x and y values of each value from the data. Miss 1 is increased if she returns from the prediction label. **A screenshot of a cell phone

Description automatically generated**

**--->** Whenever the miss value increases, it keeps the accuracy epoch values linearly in the array.After that, print Linear output of epoch and loss values. **  
--->** We assign the values returned from the point of gravity to the values of w1, w2 and b. Then I compare groupx and groupy data. I divide the data obtained linearly into two groups, blue and red. **A screenshot of a cell phone

Description automatically generated**

**--->** The data set was assigned on the graph in a visual way with the function of "scatter". I can draw point with different colors in the map.So I can do this by using for loop.Finally, I gave numbers in the ax.scatter().

**A screenshot of a cell phone

Description automatically generated**

**--->** Then, it calculates the accuracy and loss values of the epoch value as in the output of "main2" in the Linear plane.**A close up of a logo

Description automatically generated**

**c – 2-Python Code : main1.py**

**A screenshot of a cell phone

Description automatically generatedA screenshot of text

Description automatically generated**

**3.Results and Discussion**

**--->main1:** The first one is main1.py which is a simple implemtation that you can use it as for learning aspects. I mean there is a perceptron class which I can use it whenever I want with any training data required.

**--->main2:** In the second file named main2.py there is a more practical implemenation using the first one but on the real data. Here I had a real dataset of 200 points of x and y and what I did was training this dataset by using perceptron learning algorithm and see the result. Actually I had to have a big number of epochs becaue the data was really limited . At the end I have visualized my results by Matplotlib of python.

**A screenshot of a computer

Description automatically generated main2.py
**

**A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated**

**4.REFERANCES**

1. <https://www.linkedin.com/pulse/yapay-sinir-ağları-ve-tek-katmanlı-ağlarda-öğrenme-tanju-doğan/>
2. <https://devhunteryz.wordpress.com/2018/06/30/derin-ogrenme-perceptron-ogrenme-algoritmasi/>
3. <https://blog.dbrgn.ch/2013/3/26/perceptrons-in-python/>
4. Searches for Perceptron Algorithm;  
   <https://plato.stanford.edu/entries/perceptual-learning/>
5. Medium for Perceptron Algorithm;  
   <https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975>  
   <https://medium.com/@isikhanelif/perceptron-nedir-83dd39aadfa0>  
   <https://medium.com/@nikhilc3013/simple-perceptron-training-algorithm-explained-7bbfdff2c57d>
6. This perceptron Project shared my github repo;

<https://github.com/duranugur/perceptron>