Artificial Intelligence

Date of submission 06-11-2019.

*Session 2019/Section B/Semester Fall 2019*

Perceptron Learning Model

# Muhammad Talha (2017-CS-112)

Corresponding author: Haseeb Virk (muhammadtalha1726@gmail.com ).

**ABSTRACT**

Perceptron in binary classifier used to classify data into two 0 and 1 or -1 and 1. It is based on supervised learning can learn every linearly saperable dataset. I am using IRIS dataset for flowers classification.

**INDEX TERMS**

**INTRODUCTION**

Perceptron learn from given data. We initialize it with random weights and update it after each EPOCH. When we got 0 error in whole epoch we terminate learning process.

In testing phase we get last weights from learned data and process it for each itration.

**DATA SET DETAILS**

We have IRIS data set of flowers

1. Sepal\_length
2. Sepal\_width
3. Petal\_length
4. Petal\_width

In dataset only one column show output which is species.

Output of species may be 1 or 0.

**METHOD**

The Perceptron algorithm:

1. Load the data
2. Initialize weights of each feature randomly and initialize threshould value also randomly.
3. In each itearation we update the value of weights according to value of error which is calculated through the formula of (Yp –Ya).
4. We update the weights until the error on each given data is zero.

Code of Program:

import pandas as pd

import numpy as num

import sys

def **getAccureccy**(correct, total):  *# find Accureccy*

    return correct/total\*100

def **testData**(testFile, trainedFile):  *# test Data*

    pd\_learning = pd.read\_csv(testFile)

    rows, \_ = pd\_learning.shape

    global bised

    pd\_learned = pd.read\_csv(trainedFile)

    final = list(pd\_learned.iloc[-1, 7:11])

    correct = 0

    for i in **range**(rows):

        inputFeatures = list(pd\_learning.iloc[i, :])

        actual = inputFeatures.pop()

        predicted = predict(inputFeatures, final)

        print('Actual', actual, 'Predicted', predicted)

        if actual == predicted:

            correct = correct + 1

    accureccy = getAccureccy(correct, rows)

    print('----------')

    print('Accureccy: ', accureccy, '%')

def **checkError**(errorlist):  *# check error*

    if **all**(v == 0 for v in errorlist):

        return False

    else:

        return True

def **predict**(input, weight):  *# predict model*

    global bised

    z = 0

    for i in **range**(**len**(**input**)):

        z = z + **input**[i]\*weight[i]

    if z >= bised:

        return 1

    else:

        return 0

def **updateWeight**(yd, yp, input):  *# updatae weights*

    global weights

    a = 0.1

    for i in **range**(**len**(weights)):

        weights[i] = weights[i]+a\*(yd-yp)\***input**[i]

def **saveEpoch**(epoch, input, weights, predictedValue, actualValue, error):  *# save each epoch in data*

    global pd\_epoch

    dic = {'epoch': epoch, 'x1': **input**[0], 'x2': **input**[1], 'x3': **input**[2], 'x4': **input**[3], 'y': predictedValue, 'w0': bised,

           'w1': weights[0], 'w2': weights[1], 'w3': weights[2], 'w4': weights[3], 'd': actualValue, 'error': error}

    pd\_epoch = pd\_epoch.append(dic, ignore\_index=True)

def **learning**():  *# learning perceptron*

    global pd\_epoch

    global weights

    global bised

    global pd\_learning

    global columns

    global inputs

    isError = True

    epoch = 1

    while(isError):

        errorList = []

        print("Episode #", epoch)

        print('[', weights[0], weights[1], weights[2], weights[3], ']')

        for i in **range**(rows):

**input** = list()

            for j in **range**(columns):

**input**.append(pd\_learning.loc[i, inputs[j]])

            actualValue = **input**.pop()

            predictedValue = predict(**input**, weights)

            error = actualValue - predictedValue

            errorList.append(error)

            saveEpoch(epoch, **input**, weights,

                      predictedValue, actualValue, error)

            updateWeight(actualValue, predictedValue, **input**)

        epoch = epoch + 1

        isError = checkError(errorList)

    pd\_epoch.to\_csv('learnedData.csv', index=False)

    print("--------------")

    print('Total Episodes', epoch-1)

    print('Final Weights', weights, 'Treshold:', bised)

if \_\_name\_\_ == "\_\_main\_\_":

    command = sys.argv

    col = ['epoch', 'x1', 'x2', 'x3', 'x4', 'y', 'w0',

           'w1', 'w2', 'w3', 'w4', 'd', 'error']

    pd\_epoch = pd.DataFrame(columns=col)

    pd\_learning = pd.read\_csv(command[2])

    rows, columns = pd\_learning.shape

    inputs = list(pd\_learning.columns)

    weights = [2.1]\*(**len**(inputs)-1)

    bised = 2.3

    if command[1] == '--train':

        learning()

    elif command[1] == '--test':

        testData(command[2], command[4])

**Data Preprocess Code**

import pandas as pd

import numpy as num

import sys

species = set()

pd\_iris = pd.read\_csv('Iris.csv')

rows, columns = pd\_iris.shape

for i in **range**(rows):

    species.add(pd\_iris.loc[i, 'Species'])

species = list(species)

for i in **range**(rows):

    for s in species:

        if pd\_iris.loc[i, 'Species'] == s:

            pd\_iris.loc[i, 'Species'] = species.index(s)

pd\_iris.to\_csv('IRIS2.csv')

**INSTRUCTION TO EXECUTE THE PROGRAM**

1. You can clone or download the project by visiting the repository below:

<https://github.com/muhammadtalha112/PerceptronAI>

1. Once you downloaded the project, you need to open the perceptron.py file in VSCode.
2. Then write

***python index.py --train IRIS2.csv***

in terminal for learning on data.

1. When learning is complete then done will be print on console.
2. After learning you need to write

***python index.py --test IRIS2.csv -l learnedData.csv***

for testing.

**REFERENCES**

Dataset has been taken from

<https://www.kaggle.com/uciml/iris>