Artificial Intelligence

Date of submission 06-11-2019.

*Session 2019/Section B/Semester Fall 2019*

Binary Classifier Using Perceptron Model

# Haseeb Virk (2017-CS-105)

Corresponding author: Haseeb Virk (haseebvirk94@gmail.com ).

**ABSTRACT**

My project is a perceptron. The perceptron is an algorithm for supervised learning of binary classifiers. I make perceptron from scratch in which we preprocess the iris.csv file and make ANDGate.csv file.

**INDEX TERMS**

**INTRODUCTION**

Our project is a console base. When my project learn from given data then it save the weights of each iteration in file. When we test data they predict every data according to weights of last iteration. We also calculate the accuracy and presition of testing data and show result on console.

**DATA SET DETAILS**

This is flower dataset in which 4 features of flowers are present. Sepal\_length which is in integer, Sepal\_width which is in integer, Petel\_length which is in integer, Petal\_width which is in integer. According to size of petal and sepal it tell us about the species of flower.

**DATA PREPROCESSING DETAILS**

Mostly data in file is integer type but problem in species column which is in integer. We need to convert into 0 and 1.

If the species is Iris-setosa then place 1 on this column otherwise place 0 on the column. For this column I write a program in preprocessing.py file which convert string data into 0 or 1.

**Code of PreProcessing**

import pandas

data\_csv=pandas.read\_csv('iris.csv')

data=data\_csv.values.tolist()

colunms=['sepal\_length','sepal\_width','petal\_length','petal\_width','species']

mdf=pandas.DataFrame(columns=colunms)

for i in range(0,len(data)):

values=[]

values.append(data[i][0])

values.append(data[i][1])

values.append(data[i][2])

values.append(data[i][3])

if data[i][4]=='Iris-setosa':

values.append(1)

elif data[i][4]=='Iris-virginica':

values.append(0)

mdf.loc[len(mdf)]=values

print(values)

mdf.to\_csv("ANDGate.csv", index=False)

**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

import random

import sys

learning\_rate=0.2

data\_csv=pandas.DataFrame()

data=[]

weights=[]

prediction=[]

activation=[]

arguments=sys.argv

col\_names=[]

# for set the value of Threshould

def fixThreshold(error):

newBiase=weights[0]+learning\_rate\*(error)

if newBiase>learning\_rate and newBiase<1 :

weights[0]=newBiase

# for predict the value according to weights

def predict(row):

activation=0

for i in range(1,len(weights)):

activation+=weights[i]\*row[i-1]

activation-=weights[0]

return 1 if activation>=0 else 0

# update weights according to error

def errorSolving(error,row):

for i in range (1,len(weights)):

weights[i]=weights[i]+learning\_rate\*(error)\*row[i-1]

#inialize the weights and threshould randomly

def initialize():

weights.append(random.uniform(learning\_rate,1))

for j in range(0,len(data\_csv.columns)-1):

ran=random.uniform(0,1)

weights.append(ran)

# learn weights from given data file

def learn():

flag=True

while flag==True:

flag=False

for i in range(0,len(data)):

value=predict(data[i])

actualOutput=data[i][len(data\_csv.columns)-1]

error=actualOutput-value

errorSolving(error,data[i])

fixThreshold(error)

save()

if error!=0.0:

flag=True

print("Done")

#calculate the Accuracy

def Accuracy():

data\_csv=pandas.read\_csv('testpredict.csv')

data=data\_csv.values.tolist()

actualIndex=len(data\_csv.columns)-2

predictedIndex=len(data\_csv.columns)-1

correct=0

Truepositive=0

for i in range(0,len(data)):

if data[i][actualIndex]==data[i][predictedIndex]:

correct=correct+1

Accuracy=correct/len(data)\*100

print("Accuracy is: "+str(Accuracy)+"%")

# Precision

def Precision():

data\_csv=pandas.read\_csv('testpredict.csv')

data=data\_csv.values.tolist()

actualIndex=len(data\_csv.columns)-2

predictedIndex=len(data\_csv.columns)-1

total=0

correct=0

for i in range(0,len(data)):

if data[i][actualIndex]==1.0:

total=total+1

if data[i][actualIndex]==1.0 and data[i][actualIndex]==data[i][predictedIndex]:

correct=correct+1

precision=correct/total\*100

print("Precision is: "+str(precision)+"%")

# save weights of every epoche in file

def save():

values=[]

for i in range(0,len(weights)):

values.append(weights[i])

mdf.loc[len(mdf)]=values

print(values)

if len(arguments)<4:

mdf.to\_csv("data.csv", index=False)

else:

mdf.to\_csv(arguments[4]+".csv", index=False)

# when program learn from given data

if arguments[1]=="--learn":

data\_csv=pandas.read\_csv(arguments[2])

data=data\_csv.values.tolist()

initialize()

for i in range(0,len(weights)):

col\_names.append('w'+str(i))

mdf=pandas.DataFrame(columns=col\_names)

values=[]

for i in range(0,len(weights)):

values.append(weights[i])

mdf.loc[len(mdf)]=values

learn()

# when program use for testing

if arguments[1]=="--test":

data\_csv=pandas.read\_csv(arguments[2])

data=data\_csv.values.tolist()

if len(arguments)<4:

learningFile\_csv=pandas.read\_csv("data.csv")

learningFile=learningFile\_csv.values.tolist()

else:

learningFile\_csv=pandas.read\_csv(arguments[4])

learningFile=learningFile\_csv.values.tolist()

prdictedValue=0

for i in range(1,len(data\_csv.columns)):

col\_names.append('x'+str(i))

col\_names.append('actualAnswer')

col\_names.append('Predicted')

mdf=pandas.DataFrame(columns=col\_names)

for i in range(0,len(data)):

values=[]

activation=0

for j in range(0,len(data\_csv.columns)-1):

values.append(data[i][j])

activation+=learningFile[len(learningFile)-1][j+1]\*data[i][j]

activation=activation-learningFile[len(learningFile)-1][0]

if activation>=0:

predictedValue=1

else:

predictedValue=0

k=len(data\_csv.columns)-1

values.append(data[i][k])

values.append(predictedValue)

mdf.loc[len(mdf)]=values

mdf.to\_csv("testpredict.csv", index=False)

print("Done")

Accuracy()

Precision()

# EXPERIMENT AND RESULTS

# Accuracy and precision of my project on dataset which I used in this project is 100%

**INSTRUCTION TO EXECUTE THE PROGRAM**

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

[https:// <https://github.com/haseebvirk94/AI-Perceptron->](https://github.com/aliahmadcse/Books-Recommendor)

1. Once you downloaded the project, you need to open the perceptron.py file in VSCode.
2. Then write python perceptron.py –learn ANDGate.csv in terminal for learning on data.
3. When learning is complete then done will be print on console.
4. After learning you need to write python perceptron.py

–test.csv for testing.

**REFERENCES**

I take dataset from the site.

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