CSE 4088

Introduction to Machine Learning

--Homework 2--

Prepared By:

İsmail ÖKSÜZ
150119516

Before Begin

By running Answers.py, every answers of each questions that I solved will be displayed.
(The execution time for ThePerceptronLearningAlgorithm is a bit of slow)

Answers.py

```
1   import GeneralizationError
2   import ThePerceptronLearningAlgorithm
3
4   GeneralizationError
5   ThePerceptronLearningAlgorithm
```

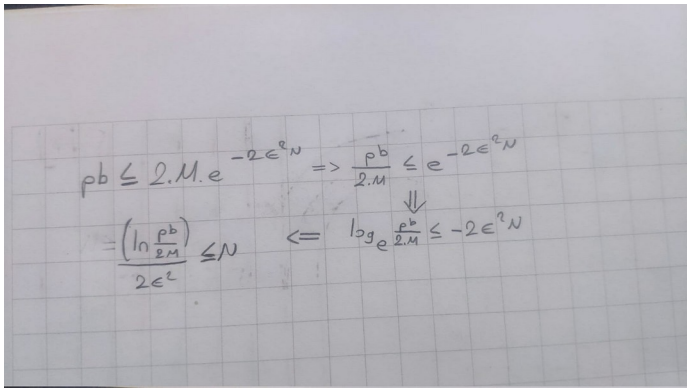
My answers:

```
exc@exc-NBLK-WAX9X:~/
nsions/ms-python.pyth
xc/Desktop/ödev/Answe
Question 1 => 1000
Question 2 => 1500
Question 3 => 2000
Question 4 => 15
Question 5 => 0.1
Question 6 => 100
Question 7 => 0.01
```

Generalization Error

That part contains questions 1-2-3

In that part, firstly I find the N's min value with pencil on paper.



Then I created GeneralizationError.py and calculated the results of the first 3 questions by that code.

```python
def findArgument(pb,M):
    return pb/(2*M)


'''findMin finds the min value of N
argument is the result of findargument part and e2 is the epsilon value'''
def findMin(e2,argument):
    ln=numpy.log
    minN=(ln(argument))/(-2*(e2**2))
    return minN


#findOption finds the correct option according to the minimum N value.
def findOption (minN):
    options=[500,1000,1500,2000]
    for j in range(0,len(options)-1):
        if(minN>options[j] and minN<=options[j+1]):
            return options[j+1]
    return "More exaples are needed"

#İsmail ÖKSÜZ
#150119516

#Created arrayM for M values
arrayM=[1,10,100]

#Created results to append answers
results=[]

#Code runs for every M values.
for i in range (0,len(arrayM)):

    #assign value to argument varible => ln(pb/(2*M))
    argument=findArgument(0.03,arrayM[i])

    #assign value to minN varible => argument/(-2*epsilon^2)
    minN=findMin(0.05,argument)

    #finds the correct option and add the result to results array
    results.append(findOption(minN))
```

Answer for Q1 = 1000 [b]
Answer for Q2 = 1500 [c]
Answer for Q3 = 2000 [d]

The Perceptron Learning Algorithm

That part contains questions 4-5-6-7

For that part, I use Dataset.py.

```
import random
import matplotlib.pyplot as plt
import numpy as np

def getRandomCoordinates():
    return (random.uniform(-1, 1),random.uniform(-1, 1))

class PerceptionLearningAlgorithm:
  def getCandidateFunction(self, point):
    return int(np.sign(self.weight[0]*1 + self.weight[1]*point[0] + self.weight[2]*point[1]))

  def __init__(self, dataset):
    self.weight = np.array([0, 0, 0])
    self.dataset = dataset

  def fit(self, iterationPloted=False):
    self.weight = np.array([0, 0, 0])
    iterationCount = 0
    while True:
      misclassifiedPointsArray = []
      for (x, y) in zip(self.dataset.xs, self.dataset.ys):
        if self.getCandidateFunction(x) != y:
          misclassifiedPointsArray.append((np.array([1, x[0], x[1]]), y))
      if len(misclassifiedPointsArray) > 0:
        iterationCount += 1
        mpa1, mpa2=random.choice(misclassifiedPointsArray)
        self.weight = self.weight + mpa1*mpa2
      else:
        return iterationCount

class Dataset:
  def __init__(self, num_points):
    axis1 = getRandomCoordinates()
    axis2 = getRandomCoordinates()
    self.augmented1 = (axis2[1] - axis1[1]) / (axis2[0] - axis1[0])
```

Calculator is a function that calculates the result for each question by taking several variable and returns the result.

```
def calculator(run,N,numpoints,qNo):
  iterationCountArray = []
  cntrTrueClassified = 0
  cntrErrorClassified = 0
  for i in range(run):
      dataset = Dataset.Dataset(num_points=numpoints)
      perceptionLearningAlgoritm = Dataset.PerceptionLearningAlgorithm(dataset)
      iterationCountArray.append(perceptionLearningAlgoritm.fit())
      for j in range(N):
          p = Dataset.getRandomCoordinates()
          if dataset.getTargetFunction(p) == perceptionLearningAlgoritm.getCandidateFunction(p):
              cntrTrueClassified += 1
          else:
              cntrErrorClassified += 1
  cntrClassified=cntrErrorClassified+cntrTrueClassified
  probError=answer57(float(cntrErrorClassified)/(cntrClassified))
  if qNo==4:
    noOfIteration=answer4(np.mean(iterationCountArray))
  elif qNo==6:
    noOfIteration=answer6(np.mean(iterationCountArray))
  if qNo%2==0:
    return noOfIteration
  else:
    return probError
```

For each question, I created different functions to find correct answer:

```
def answer4(noOfIteration):
  options=[1,15,300,5000,10000]
  dif=abs(noOfIteration-options[0])
  choice=0
  for i in range(0,len(options)):
    if abs(noOfIteration-options[i])<dif:
      dif=abs(noOfIteration-options[i])
      choice=i
  return options[choice]

#finds the correct option for question 5 and 7
def answer57(probError):

  #as the options are same for q5 and q7, i used single function
  options=[0.001,0.01,0.1,0.5,0.8]
  dif=abs(probError-options[0])
  choice=0
  for i in range(0,len(options)):
    if abs(probError-options[i])<dif:
      dif=abs(probError-options[i])
      choice=i
  return options[choice]

#finds the correct option for question 6
def answer6(noOfIteration):
  options=[50,100,500,1000,5000]
  dif=abs(noOfIteration-options[0])
  choice=0
  for i in range(0,len(options)):
    if abs(noOfIteration-options[i])<dif:
      dif=abs(noOfIteration-options[i])
      choice=i
  return options[choice]
```

Answer for Q4 = 15 [b]
Answer for Q5 = 0.1 [c]
Answer for Q6 = 100 [b]
Answer for Q7 = 0.01 [b]