HW 5 LOGISTIC classifier

Due 03/30/2023 in class

The code from the lecture example is attached in the homework. You can modify the code.

1. Logistic classification for a different set of synthetic data

In the lecture, I gave an example of using synthetic data and feed it to the weight update algorithm and obtain a logistic probability distribution.

Now we wish to turn to anther probability distribution as a new synthetic dataset.

First, let's define a piecewise function y(x) as

$$y(x) = 0 \; when \; x < 1$$

$$y(x) = 0.5 * (x + 1) when 1 < x < 3$$

$$y(x) = 1 \; when \; x > 3$$

Now generate x^t from U(-4,4), i.e., a uniform distribution from the interval [-4,4]

For a given x^t, generate a random number zeta from U(0,1), i.e., a uniform distribution from interval [0,1] and the outcome r^t is determined by

r^t= 1 when 0<zeta< y(x^t)

r^t= 0 otherwise

In total, please generate 30 sample for x^t . t=1,2,……30 (Note we use Ethem Alpaydian's notation and t is sample index.)

Use this new data set to feed into the weight update algorithm for 100 iteration.

(a) Plot the new data for the interval [-4,4] (15%)

(b) Find the weight coefficient w1 and w0 after 100 iteration. Plot the logistic function using w1 and w0.(20%)

(c) Plot the cross entropy versus number of iteration. You can sample the data for every 10 iteration and use the data from 0 iteration to 90 iteration. (15%)

2. Calculation of decision boundary.

In the lecture, I show you the decision boundary for two bivariate Gaussian distribution is a circle assuming the prior for Gaussian 1 (class 1) and Gaussian 2(class 2) are equal P(C1)=P(C2)=0.5. In this problem, I will give you another set of parameters and ask you to use the formula to calculate the radius and location of decision boundary.

$$\|\mathbf{x} - \mathbf{x}_c\|^2 = r^2$$

$$\mathbf{x}_c = \frac{\sigma_2^2 \boldsymbol{\mu}_1 - \sigma_1^2 \boldsymbol{\mu}_2}{\sigma_2 - \sigma_1}$$

$$r^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_2^2 - \sigma_1^2} \left[\frac{\|\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2\|^2}{\sigma_2^2 - \sigma_1^2} + 4 \ln \left(\frac{\sigma_2}{\sigma_1} \right) \right]$$

For class 1, the mean and variance for Gaussian is given by

$$\boldsymbol{\mu_1} = (0{,}0)$$

$$\sigma_{\mathbf{1}} = 1$$

$$\boldsymbol{\mu_2} = (3{,}4)$$

$$\sigma_{\mathbf{2}} = 3$$

(a) Calculate the center and radius of the decision boundary (a circle)

(20%)

(b) Use the python code to calculate the misclassification (error) rate for class 1. The misclassification rate for class means that the sample is from Gaussian 1 but being classified as Gaussian 2. You can use 1000 sample point from Gaussian 1 and 1000 sample point from Gaussian 2.(10%)

(c) Use the python code to calculate the misclassification (error) rate for class 2. The misclassification rate for class means that the sample is from Gaussian 2 but being classified as Gaussian 1. This rate is low. For each run, you can use 1000 sample point from Gaussian 1 and 1000 sample point from Gaussian 2. Run the experiment 10 times and give the answer with mean $\pm$ standard deviation. (20%)

Appendix: code for logistic classification and decision boundary for two gaussian

```
import random
import math
w1_set=1.2
w0_set=0
x=0.0
outcome=0
w1=0.01
w0=0.01
eta=0.01
x_record=[]
r_record=[]

def sigmod(x,w1,w0):
    return 1/(1+ math.exp(-w1*x-w0))
for iteration in range(30):
    x=random.uniform(-2,2)
    zeta=random.uniform(0,1)
    y= sigmod(x,w1_set, w0_set)
    if zeta <= y:
        outcome=1
        x_record.append(float(x))
        r_record.append(int(outcome))
        print( 'iteration', iteration, 'x', round(x,2),' ', outcome)

    else:
        outcome=0
        x_record.append(float(x))
        r_record.append(int(outcome))
        print( 'iteration', iteration,'x', round(x,2),' ', outcome)


epoch=100
sum1=0.0
sum2=0.0

sum_en=0.0
print ('sum1', sum1, 'sum2', sum2)
for iteration2 in range(epoch):
    sum2=0.0
    sum1=0.0
    sum_en=0.0
    for i in range(30):
        y= sigmod(x_record[i],w1,w0)
        sum1= sum1+(r_record[i] -y)*x_record[i]
        sum2=sum2+(r_record[i] -y)
        #computing cross entropy
        sum_en= sum_en-(r_record[i]*math.log(y)+(1-r_record[i])*math.log(1-y))

    w1=w1+eta*sum1
    w0=w0+eta*sum2
    if iteration2 %10==0:
        print( 'iteration', iteration2, 'error', sum_en)
        #print( 'iteration', iteration2, 'w1', round(w1,2), 'w0', round(w0,2))

print('final result', w1,' ', w0)
```

```python
# class 1 data is from Gaussian with mean (0,0) and sigma 1
# class 2 data is from Gaussian with mean (2,0) and sigma 2
# 100 data points are generated
# error rate for class 1 falling outside the circle is calculated
# error rate for class 2 falling inside the circle is calculated
# in the textbook    error rate 1  0.1056 our answer 10%
# in the textbook    error rate 2  0.2642 our answer  26%


import random

x1_record=[]
x2_record=[]
x1=0.0
x2=0.0
y1=0.0
y2=0.0
count1_error=0
count2_error=0
#decision boundary

for i in range(100):
    x1=random.gauss(0,1) #create random number gauss (mean, sigma)
    x2=random.gauss(0,1)
    x1_record.append(float(x1))
    x2_record.append(float(x2))

    if (x1+0.667)**2+x2**2> 2.34**2 :
        count1_error += 1
    else: pass
print('error rate 1 in %', count1_error)
for i in range(100):
    y1=2.0+random.gauss(0,2)
    y2=0.0+random.gauss(0,2)

    if (y1+0.667)**2+y2**2< 2.34**2:
        count2_error += 1
    else: pass
print('error rate 2 in %', count2_error)
```