Machine Learning HW2 part1

E94071209 工科系 111 林政旭

Problem1:

Code: HW2_part1_problem1

參數: Number_observation=5000 =>常態分布的粒子個數

Learning rate=5e-5

Num_steps=3000 =>迭代次數

Random.seed(1)

1.設定 x1,x2 為常態分布，透過 logistic_regression 方程式，迭代計算 weights，並得到 weights=[-0.0107074 2.01105661]，可得 Sigmoid logistic function:

$$R(x) = \frac{1}{1 + \exp(-2.01105661x)} = \sigma(2.01105661\mathrm{x})$$

or 透過 Bayes Rule 可得到理論值 $R(x) = \sigma(2\mathrm{x})$

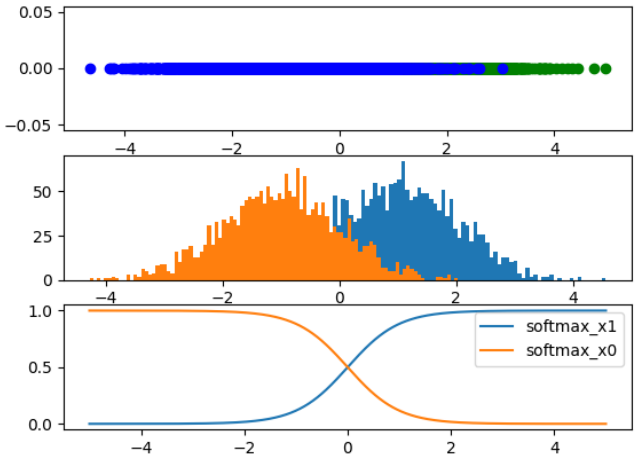
2.透過權重 weights 也可得 Softmax function:

透過 Softmax 定義:

$$Softmax(a.b) = [\frac{\exp(a)}{\exp(a) + \exp(b)}, \frac{\exp(b)}{\exp(a) + \exp(b)}]$$

$$[1 - R(x), R(x)] = [\frac{\exp(-2x)}{\exp(0) + \exp(-2x)}, \frac{\exp(0)}{\exp(0) + \exp(-2x)}]$$

$$= Softmax(-2x, 0)$$



```python

import numpy as np
import matplotlib.pyplot as plt

np.random.seed(1)
number_observation=5000

x1=np.random.multivariate_normal([1],[[1]],number_observation)
x0=np.random.multivariate_normal([-1],[[1]],number_observation)

#將其兩個從原本的shape都為(number_observation,1) 轉變成(2*number_observation,1) 疊加的概念
features=np.vstack((x0,x1)).astype(np.float32)
labels=np.hstack((np.zeros(number_observation),np.ones(number_observation)))
y=np.zeros(number_observation)

plt.subplot(3,1,1)
plt.scatter(x1,y,c='g')
plt.scatter(x0,y,c='b')

def sigmoid(x):
    return 1/(1+np.exp(-x))

def logistic_regression(features,target,num_steps,learning_rate,add_intercept=False):

    if add_intercept:
        intercept=np.ones((features.shape[0],1))
        features=np.hstack((intercept,features))
    weights=np.zeros(features.shape[1])

    #開始進行迴圈跑最出最佳參數
    for step in range(num_steps):
        scores=np.dot(features,weights)
        #function套入sigmoid function得1機率值
        predictions=sigmoid(scores)
        #觀看誤差
        output_error_singal=target-predictions
        #gradient
        gradient=np.dot(features.T,output_error_singal)
        weights+=learning_rate*gradient
    return weights

weights=logistic_regression(features,labels,num_steps=3000,learning_rate=5e-5,add_intercept=True)
print('權重:',weights)

def softmaxx0(x,weight):
    return np.exp(weight[0]*x)/(np.exp(weight[0]*x)+np.exp(weight[1]*x))
def softmaxx1(x,weight):
    return np.exp(weight[1]*x)/(np.exp(weight[0]*x)+np.exp(weight[1]*x))
```

```python

def sigmoidx1(x,weight):
    return 1/(1+np.exp(-(weight[1])*x))
def sigmoidx0(x,weight):
    return 1/(1+np.exp(-(weight[0])*x))

sigma = 1
sampleNo = 2000

s = np.random.normal(1, sigma, sampleNo)
b = np.random.normal(-1, sigma, sampleNo)
x = np.linspace(-5,5,2000)

plt.subplot(3,1,2)
plt.hist(s, bins=100)
plt.hist(b, bins=100)

plt.subplot(3,1,3)
plt.plot(x,softmaxx1(x,weights),label='softmax_x1')
plt.plot(x,softmaxx0(x,weights),label='softmax_x0')
plt.legend()
print('sigmoid c1:',sigmoidx1(1,weights))
print('sigmoid c0:',sigmoidx0(1,weights))
print('softmax c1:',softmaxx1(1,weights))
print('softmax c0:',softmaxx0(1,weights))
plt.show()

```

Problem2:

參數:在範圍(-10,10)內，每 0.0001 產生一個數字

設定 a=random 的數字

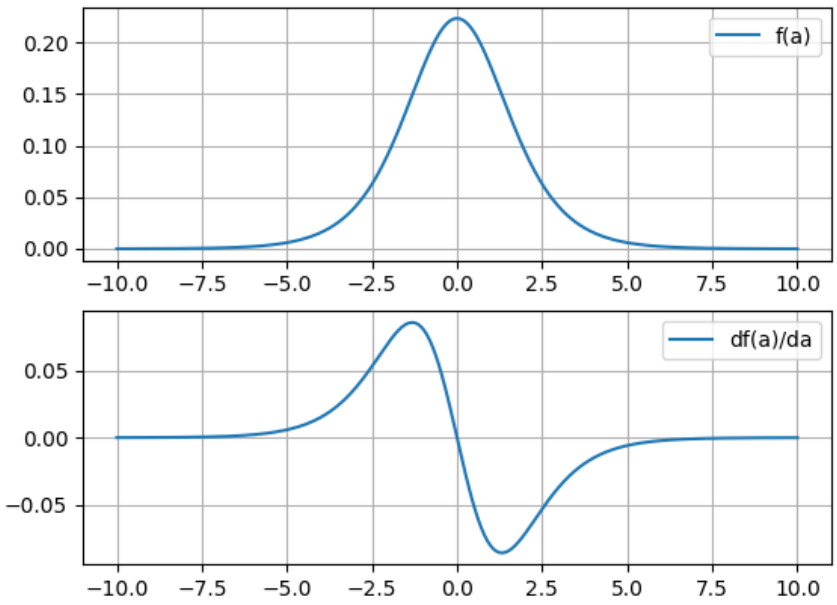
令 f(a)=∀a $argmax\ \sigma(a+b)\,\sigma(a-b)=0$

對其微分=0 可得極端值

$\frac{d}{da}f(a)=\frac{exp(-b-a)-\exp(b-a)}{(\exp(b-a)+1)^2*(\exp(-b-a)+1)^2}$=0

透過程式可得在 b=-2.3305801732931286e-11 時，有最大值(約等於 0)

其 probability 為 0.6224593312004854，其 log_odds_ratio=0.49999999999417366



```python
import matplotlib.pyplot as plt
import numpy as np
import random
a=random.uniform(-3,3)#random產生一個fixed constant number
x=np.arange(-10,10,0.0001)

def f(x):
    return 1/(1+np.exp(-x))

y=f(a+x)*f(a-x)
dy=(np.exp(-x-a)-np.exp(x-a))/(pow((np.exp(x-a)+1),2)*pow((np.exp(-x-a)+1),2))
ans=np.argmax(y)
probability=np.exp(f(x[ans]))/(1+np.exp(f(x[ans])))
logoddsratio=np.log(probability/(1-probability))

plt.subplot(2, 1, 1)
plt.plot(x,y,label='f(a)')
plt.grid()
plt.legend()
plt.subplot(2, 1, 2)
plt.plot(x,dy,label='df(a)/da')
plt.grid()
plt.legend()
print(ans)#在第ans個 有dy=0的極端值
print(x[ans])#最大值在x[ans]的地方 約=0 ⇒ 要set b to zero
print(probability)#0.6224593312004854
print(logoddsratio)#0.49999999999417366
plt.show()

```