MACHINE LEARNING HW2 REPORT

B02901065 電機四 李洺曦
討論者：B02901100 電機四 邱世鈞

1. neuron.py中的refresh_para主要負責更新係數，根據定義，將1/(1+$\sigma(z)$)求出gradient後乘上learning rate去更新係數，程式碼如下：

```
10    def cal_z(self, data):
11        return np.dot(data, self.omega) + self.bias
12
13    def cal_pro(self, data):
14        return 1 / (1 + np.exp((self.cal_z(data) * -1)))
15
16    def refresh_para(self, data, learning_rate, ta):
17        self.omega = self.omega + learning_rate * np.dot(np.transpose(data), (ta - self.new_a
    ns))
18        self.bias = self.bias + learning_rate * np.sum((ta - self.new_ans))
19        self.new_ans = self.cal_pro(data)
```

以上為training部份。
其中cal_pro所求的的解會是一個[0,1]之間的數，則將ans >= 0.5的當作class 1(1)，ans < 0.5的當作class0(0)，程式碼如下：

```
27    def get_ans(self, data):
28        temp1 = self.cal_pro(data)
29        temp = np.zeros(temp1.shape)
30        for i in range(temp1.shape[0]):
31            if temp1[i][0] >= 0.5:
32                temp[i][0] = 1
33            else:
34                temp[i][0] = 0
35        temp.astype(np.int64)
36        return temp
```

由此完成classification

2. 方法二我使用了Pocket PLA演算法，先假設一個線性方程式，隨機抽取一個點後利用此方程式進行classification，大於某個threshold時為1(1)，反之為0(-1)，如果假設錯誤，則利用正確答案進行線性方程式的修正，其修正法為：方程式法線向量 + 正確答案(1 or -1) * 該點的與原點的向量，透過此方法使方程式對於此點的分類正確，疊代做多次的修正，可以得到一個trained model，程式碼如下：

```
20    def refresh_para(self, data, ta, iteration):
21        least_false = self.count_false(self.cal_temp_z(data, self.omega, self.bias), ta)
22        total_counter = 0
23        global_omega = self.omega
24        global_bias = self.bias
25        while iteration > 0:
26            i = random.randint(0, data.shape[0] - 1)
27            temp = self.cal_temp_z(data[i], global_omega, global_bias)[0]
28            if (temp > 0 and ta[i][0] == 0) or (temp < 0 and ta[i][0] == 1):
29                false_count = 0
30                if temp > 0:
31                    global_omega = global_omega - np.transpose(data[i])
32                    global_bias += 1
33                else:
34                    global_omega = global_omega + np.transpose(data[i])
35                    global_bias -= 1
```

但是Pocket為了使model不會因為noise而做振盪，所以我們定義false_count為一個model對於training data的不準確資料數，一旦false_count比最小的該狀態最小的false_count還要小，則代表我們找到了一個更好的model，並且保存此最佳model，程式碼如下：

```
36                temp_ans = self.cal_temp_z(data, global_omega, global_bias)
37                false_count = self.count_false(temp_ans, ta)
38                if false_count <= least_false:
39                    least_false = false_count
40                    self.omega = global_omega
41                    self.bias = global_bias
```

以上為training部份。

model求出來後減去threshold值的數，以0為基準分為兩類，ans >= 0者為class1(1)，ans < 0 者為class0(-1)，程式碼如下：

```
59    def get_ans(self, data):
60        temp1 = self.cal_z(data)
61        temp = np.zeros(temp1.shape)
62        for i in range(temp1.shape[0]):
63            if temp1[i][0] >= 0:
64                temp[i][0] = 1
65            else:
66                temp[i][0] = 0
67        temp.astype(np.int64)
68        return temp
```

3. 將兩種model在kaggle上的分數以logistic regression較高，推測因PLA中取樣點是隨機的，有時會重複train到有時會被忽略，造成model較不準確