ML HW2 Report

B03902082 資工三 江懿友

Logistic regression function

```
#!/usr/bin/env python3

import numpy as np
import math
from sys import argv, stderr

def die(msg):
    print(msg, file = stderr)
    exit(1)

def randomInitWeight():
    w = np.random.random(57) * 2 - 1
    b = np.random.random() * 2 - 1
    w *= 0.001
    b *= 0.001
    return w, b

def sigmoid(z):
    if z > 0.0:
        return 1.0 / (1.0 + math.exp(-z))
    else:
        z = math.exp(z)
        return z / (1.0 + z)

def f(w, b, x):
    return sigmoid((w * x).sum() + b)

def loss(w, b, Data):
    res = 0.0
    for (x, y) in Data:
        fwb = f(w, b, x)
        PredictY = 1 if f(w, b, x) > 0.5 else 0
        res += abs(PredictY - y)
    return res

def gradient(w, b, x, y):
    y_ = f(w, b, x) - y
    return y_ * x, y_

def main():
    if len(argv) != 3:
        die('Usage {} [train data] [output model]'.format(argv[0]))

    Data = []
    for line in open(argv[1], 'r'):
        row = line.rstrip('\r\n').split(',')
        for i in range(1, len(row)):
            row[i] = float(row[i])
        Data.append((np.array(row[1:-1]), row[-1]))

    MaxIterations = 10000
    LR = 0.01
    w, b = randomInitWeight()
    AccuGw = np.zeros(w.shape)
    AccuGb = 0.0
    for i in range(MaxIterations):
        print (loss(w, b, Data))
        gw = np.zeros(w.shape)
        gb = 0.0
        for (x, y) in Data:
            g = gradient(w, b, x, y)
            gw += g[0]
            gb += g[1]

        AccuGw += gw ** 2
        AccuGb += gb ** 2
        w -= LR * gw / np.sqrt(AccuGw)
        b -= LR * gb / np.sqrt(AccuGb)

    ModelFD = open(argv[2], 'w')
    ModelFD.write(' '.join([str(e) for e in w]))
    ModelFD.write('\n{}\n'.format(b))
    ModelFD.close()

if __name__ == '__main__':
    main()
```

The other method

我使用 maximum likelihood method 生出 Gaussian model， 並且讓 covariance matrix 共用，用來算出每筆 data 在各個 category 的機率大小。
Train 出來的 model 在 public set 的表現是 0.87667， 低於 logistic regression 的 0.92667。
但是若是把兩個 Gaussian model 的權重調成 1:1 的話，正確率就上生到了 0.91000，雖然很難單單看數字就斷言調整權重後的 model 在任何 testdata 表現一定比較好，但是至少在 Public set 上的確比較好；我想這可能代表著 Gaussian model 因為考慮到不同 class 的出現次數，而這個機率分佈卻導致了 model overfit 到 training data 上了。