161910126 赵安

习题1

（1）由闭式解的定义并带入相关数据得 w* = $\frac{1+1}{1\times1}=\frac{1}{2}$ b* = $\frac{1}{3}$

（2）

$$(w_E,b_E)=argmin\sum_{i=1}^{m}\frac{|y_i-wx_i-b|^2}{(1+w^2)}$$

$$\frac{\partial E(w_E,b_E)}{\partial w}=\sum_{i=1}^{m}\frac{-2x_i(y_i-wx_i-b)(1+w^2)-2w(y_i-wx_i-b)^2}{(1+w^2)^2}$$

$$\frac{\partial E(w_E,b_E)}{\partial b}=\frac{2mb-2\sum_{i=1}^{m}(y_i-wx_i)}{1+w^2}$$

令上面两式等于0 ,并带入相关数据得

$$b_E=\frac{1}{3}$$

$$w_E=\sqrt{\frac{13}{9}}-\frac{2}{3}=0.53518375$$

(3)

欧氏距离：

$$(w_E,b_E)=argmin\sum_{i=1}^{m}\frac{|y_i-wx_i-b|}{\sqrt{1+w^2}}$$

当w* = $\frac{1+1}{1\times1}=\frac{1}{2}$ b* = $\frac{1}{3}$ 时

$$\sum_{i=1}^{m}\frac{|y_i-wx_i-b|}{\sqrt{1+w^2}}=0.596284794$$

存在 w = 0.5 , b = 0.5 使得

$$\sum_{i=1}^{m}\frac{|y_i-wx_i-b|}{\sqrt{1+w^2}}=0.4472135955<0.596284794$$

所以w* = $\frac{1+1}{1\times1}=\frac{1}{2}$ b* = $\frac{1}{3}$ 不是该问题的解

习题2

采用softmax函数，令

$$p(y=i|x)=\frac{e^{z_i}}{\sum_{m=1}^{K}e^{z_m}}$$

习题3

```python
import pandas as pd
import numpy as np
import math


# 构造对数几率函数
def sigmoid(z):
    return 1 / (1 + math.exp(-z))


# 读取文件
train_feature = pd.read_csv("train_feature.csv", delimiter=",")
train_target = pd.read_csv("train_target.csv", delimiter=",")
val_feature = pd.read_csv("val_feature.csv", delimiter=",")
val_target = pd.read_csv("val_target.csv", delimiter=",")

# 构造训练集相关数组
x_train = np.array(train_feature.loc[:, :])
y_train = np.array(train_target.loc[:, :])
X_hat = np.append(x_train, np.ones([600, 1]), axis=1)
# 构造测试集相关数组
x_val = np.array(val_feature.loc[:, :])
y_val = np.array(val_target.loc[:, :])
X_val = np.append(x_val, np.ones([200, 1]), axis=1)

# 正例数量
T_num = np.sum(y_val == 1)
# 反例数量
F_num = np.sum(y_val == 0)

# 闭式解获得beta
Beta = np.dot(np.linalg.inv(np.dot(X_hat.T, X_hat)), np.dot(X_hat.T, y_train))

TP = 0
FP = 0
TN = 0
# 计算过程
for i in range(200):
    z = sigmoid(np.dot(Beta.T, X_val[i]))
    if z >= 0.5 and y_val[i] == 1:
        TP += 1
    if z >= 0.5 and y_val[i] == 0:
        FP += 1
    if z < 0.5 and y_val[i] == 0:
        TN += 1

P = TP / (TP + FP)
R = TP / T_num
Accuracy = (TP + TN) / (T_num + F_num)
print("闭式解")
print("Accuracy:", Accuracy)
print("Precision:", P)
print("Recall:", R)

```

```python
# 数值方法解得beta
Beta_N = np.ones([11,1])


grad_1 = np.zeros([1, 11])
grad_2 = np.zeros([11, 11])


for k in range(20):
    for i in range(600):
        temp11 = np.dot(X_hat, Beta_N)
        p1 = math.exp(temp11[i][0]) / (1 + math.exp(temp11[i][0]))
        grad_1 += X_hat[i:i+1, 0:12] * (float(y_train[i]) - p1)
        grad_2 += np.dot(X_hat[i:i+1, 0:12].T, X_hat[i:i+1, 0:12]) * p1 * (1 - p1)
    temp22 = np.dot(np.linalg.inv(grad_2), -grad_1.T)
    Beta_N = Beta_N - temp22

TP = 0
FP = 0
TN = 0
for i in range(200):
    z = sigmoid(np.dot(Beta_N.T, X_val[i]))
    if z >= 0.5 and y_val[i] == 1:
        TP += 1
    if z >= 0.5 and y_val[i] == 0:
        FP += 1
    if z < 0.5 and y_val[i] == 0:
        TN += 1

P = TP / (TP + FP)
R = TP / T_num
Accuracy = (TP + TN) / (T_num + F_num)
print("牛顿法: ")
print("Accuracy:", Accuracy)
print("Precision:", P)
print("Recall:", R)
```

结果如下:

```
闭式解
Accuracy: 0.74
Precision: 0.6666666666666666
Recall: 1.0
牛顿法:
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
```