Ecperiment1-2: Logistic Regression

Time: 2022/3/5
Location: Science_Building_119
Name: 易弘睿
Number: 20186103

Part I: Introduction

对初始代码的修改主要如下：
1. 对代码按照不同部分功能进行命名；
2. 对代码进行每行注释；
3. 增加乳腺癌数据库的逻辑回归。

Part II: Annotation

1. 数据库的导入

In [1]:

```
import numpy as np                # 导入numpy库
from sklearn import datasets      # 从sklearn导入iris数据集
import matplotlib.pyplot as plt   # 导入matplot执行绘图任务
```

2. 数据集的制作

In [2]:

```
# 加载数据集
iris = datasets.load_iris()
x = iris.data                 # iris.data为numpy.ndarray类型，其不存储target的值，但却可以通过target的
y = iris.target               # target指代鸢尾花种类，共0 1 2三类
# 选取标签为0和1的数据
X = x[y < 2, :2]              # 只取前两类（0和1）鸢尾花的前两列数据：萼片长度及萼片宽度
y = y[y < 2]                  # 只取前两类（0和1）鸢尾花的类别标签
```

In [3]:

```
# 对数据和标签进行分割以获得两类数据的横纵坐标
class1x = X[y==0,0].reshape(-1,1) # 将第一类鸢尾花的萼片长度数据赋予class1x
class1y = X[y==0,1].reshape(-1,1) # 将第一类鸢尾花的萼片宽度数据赋予class1y
print(class1x)
print(class1y)
class2x = X[y==1,0].reshape(-1,1) # 将第二类鸢尾花的萼片长度数据赋予class2x
class2y = X[y==1,1].reshape(-1,1) # 将第二类鸢尾花的萼片宽度数据赋予class2y
print(class2x)
print(class2y)
# 数据维度的变换
X = X.T                                 #转置前两类（0和1）鸢尾花的前两列数据：萼片长度及萼片宽度
y = y.reshape(1,-1)                     #与转置等效，个人觉得此句可更改为直接转置，因为reshape需要输入行数
```

```
 [5. ]
 [5.5]
 [4.9]
 [4.4]
 [5.1]
 [5. ]
 [4.5]
 [4.4]
 [5. ]
 [5.1]
 [4.8]
 [5.1]
 [4.6]
 [5.3]
 [5. ]]
[[3.5]
 [3. ]
 [3.2]
 [3.1]
 [3.6]
```

3. 权重系数、偏置系数、学习率的初始化

In [4]:

```
w = np.zeros((2, 1)) #初始化参数w的值为一个2行1列的0值向量
b = 0.0              #初始化参数b的值为0
lr = 0.1             # 设置学习率为0.1
m = 100              #设置样本数为100
```

4. 梯度下降的执行

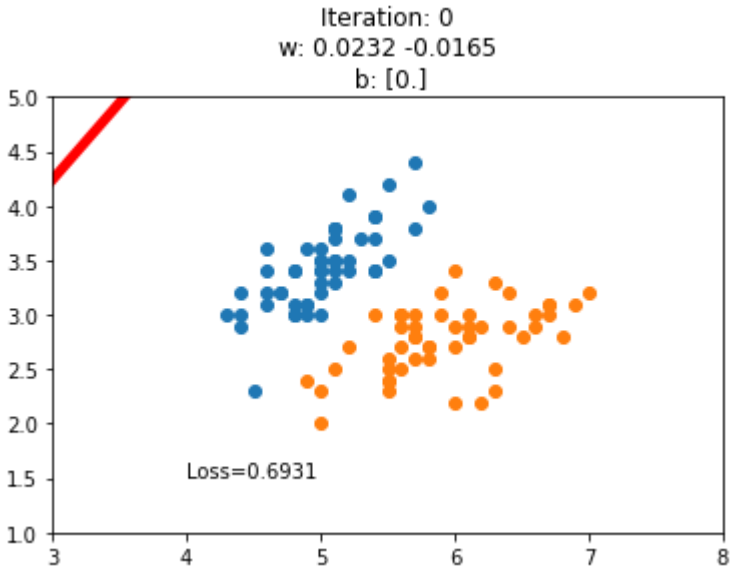
In [5]:

```
for iteration in range(10000):
    z = np.dot(w.T, X) + b    # z为根据拟合曲线计算的指数，利用w向量的转置点乘X向量再加上b得到
    a = 1 / (1 + np.exp(-z)) # a为Sigmoid Function值
    J = -(np.dot(y, np.log(a).T) + np.dot(1 - y, np.log(1 - a).T)) # 计算假设函数Hypothesis Functio
    dz = a - y                    # 计算逻辑判断值与数据集中对应标签值的差值，后续的偏导计算式可直接
    dw = np.dot(X, dz.T) / m      # 计算平均每个样本，J在w方向上的偏导
    db = np.sum(dz, axis=1) / m  # 计算平均每个样本，J在b方向上的偏导
    J /= m                        # 通过除以样本数计算代价函数的值

    # 利用偏导以及最速下降（梯度下降）法则反向更新迭代w、b的值
    w = w - lr * dw
    b = b - lr * db

    if iteration % 100 == 0:           # 每逢100次的整数倍迭代画一次图
        plt.scatter(class1x,class1y)  # 画出第一类鸢尾花萼片长度-萼片宽度数据散点图
        plt.scatter(class2x,class2y)  # 画出第二类鸢尾花萼片长度-萼片宽度数据散点图
        tempx = [i for i in range(8)] # 在[0,7]区间内均匀产生8个数
        tempy = [(-w[0][0]*i-b)/w[1][0] for i in tempx] # 根据拟合曲线的指数项计算上一行8个数对应的

        plt.plot(tempx, tempy, 'r-', lw=5) # 画出拟合分类曲线
        plt.text(4, 1.5, 'Loss=%.4f' % J)  # 打印损失值
        plt.xlim(3, 8)                     # 设置横轴坐标轴边界
        plt.ylim(1, 5)                     # 设置纵轴坐标轴边界
        plt.title("Iteration: {}\nw: {:.4f} {:.4f}\n b: {}".format(iteration, w[0][0],w[1][0], b)) #
        plt.pause(0.5)                     # 设置停顿时间
```



Part III: Diabetes Dataset

In [6]:

```
# 载入乳腺癌的数据
cancer = datasets.load_breast_cancer()
x = cancer.data
y = cancer.target
x = x[:, :3]
```

In [7]:

```
# 对数据和标签进行分割以获得两类数据的横纵坐标
class1x = x[y==0,0].reshape(-1,1)
class1y = x[y==0,1].reshape(-1,1)
class1z = x[y==0,2].reshape(-1,1)
class2x = x[y==1,0].reshape(-1,1)
class2y = x[y==1,1].reshape(-1,1)
class2z = x[y==1,2].reshape(-1,1)
# 数据维度的变换
X = x.T
y = y.reshape(1,-1)
```

In [8]:

```
# 权重系数、偏置系数、学习率初始化
w = np.random.rand(3, 1)
b = 0.0
lr = 0.005
m = 100
```

In [11]:

```
for iteration in range(10000):
    z = np.dot(w.T, X) + b
    a = 1 / (1 + np.exp(-z))
    J = -(np.dot(y, np.log(a).T) + np.dot(1 - y, np.log(1 - a).T))
    dz = a - y
    dw = np.dot(X, dz.T) / m
    db = np.sum(dz, axis=1) / m
    J /= m

    w = w - lr * dw
    b = b - lr * db
    if iteration % 1000 == 0:
        # 绘制良性肿块与恶性肿块的原始数据
        fig = plt.figure()
        ax = fig.gca(projection='3d')
        ax.scatter(class1x, class1y, class1z)
        ax.scatter(class2x, class2y, class2z)
        # 更换视角
        ax.view_init(elev=10, azim=-20)
        plt.xlabel("x")
        plt.ylabel("y")
        # 计算预测的分割曲线
        tempx = []
        tempy = []
        tempz = []
        for cur_x in range(30):
            for cur_y in range(40):
                tempx.append(float(cur_x))
                tempy.append(float(cur_y))
                tempz.append((- w[0][0] * cur_x - w[1][0] * cur_y - b) / w[2][0])
        # 打印损失值
        print("Iteration: {} Loss: {}".format(iteration, J))
        # 绘制分割面
        ax.scatter(tempx, tempy, tempz)
        # 打印图例
        plt.legend(labels=["Class: Benign","Class: Malignant", "Division"])
        plt.title("Iteration: {}\nw: {:.4f} {:.4f} {:.4f}\n b: {}".format(iteration, w[0][0],w[1][0]
        # 设置横轴坐标轴边界
        plt.xlim(5, 25)
        plt.ylim(5, 40)
        plt.pause(0.5)
```

```
<ipython-input-11-461108587b09>:4: RuntimeWarning: divide by zero encountered in
log
  J = -(np.dot(y, np.log(a).T) + np.dot(1 - y, np.log(1 - a).T))

Iteration: 0 Loss: [[nan]]
```

Iteration: 0
w: 0.0538 0.7267 -1.0321
b: [-0.0106]