Lecture5编程作业

逻辑斯蒂回归代码见：[logistic.py]

1.Logistic回归算法

```
def Logistic(data,label,alpha,epoch):                   #Logistic回归
    datadim=len(data[0])
    w=np.zeros(datadim)
    lost=np.zeros(epoch)
    ep=np.zeros(epoch)
    for t in range(epoch):
        L=gradient(data,label,w)                        #求梯度
        w-=alpha*L                                      #权重更新
        ep[t]=t
        lost[t]=Loss(train_data,train_label,w)          #求出损失函数
        if(L==np.zeros(datadim)).all():
            break
    dp.epoch_line(lost,ep)                              #画出损失函数同迭代次数的变化
    return w
```

2.

(1) 数据集

产生两个都具有200个二维向量的数据集 X_1 和 X_2 。数据集 X_1 的样本来自均值向量 $m_1=[-5,0]^T$，协方差矩阵 $s_1=I$ 的正态分布，属于“+1”类，数据集 X_2 的样本来自均值向量 $m_2=[0,5]^T$,协方差矩阵 $s_2=I$ 的正态分布，属于“-1”类，其中 I 是一个2*2的单位矩阵。其中的数据中80%用于训练，20%用于测试。

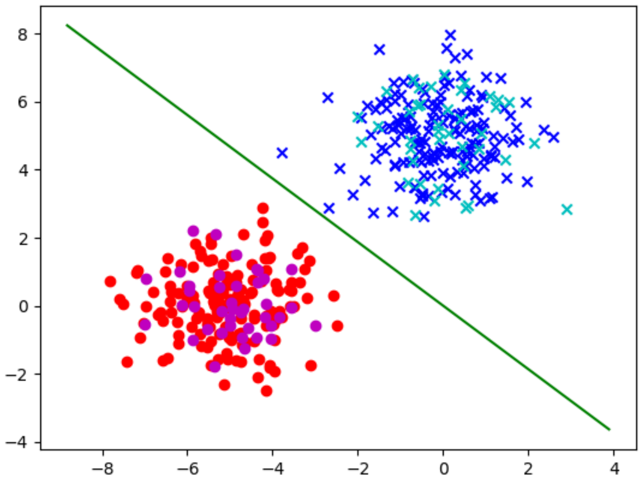
本数据集中梯度下降求最优算法的学习率alpha设为0.01，迭代次数epoch设为1000次

(2) 利用得到的分类面对测试集样本进行分类，并给出每个样本属于该类别的概率值。

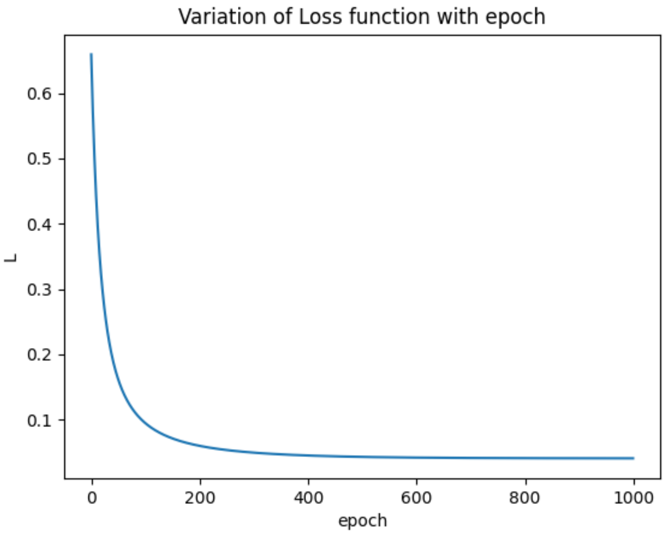
```
def predict(all_data,all_label,testdata,w):            #分类器在测试集样本属"1"类的概率值
    length=len(testdata)
    probability=np.zeros(length)
    y=np.ones(length)
    dp.draw(all_data,all_label,w)                      #画出分类面和数据集
    for i in range(length):
        probability[i] = sigmoid(-np.dot(w,test_data[i]))
        if probability[i]<0.5:                         #若概率小于0.5，则属“-1”类
            y[i]=-1
    return probability,y
```

(3) 画出数据集和分类面

（关于作图，训练集的数据标签为 “1” 和 “-1” 类分别对应红色和蓝色的o点；测试集的数据标签为“1”和“-1”分别对应紫红色和青色的x点）



(4) 损失函数随epoch增加的变化曲线



(5) 改变算法中的各类超参数、样本数量、样本分布等，对于梯度下降法还要改变不同的学习率以及不同的batch size和不同epoch次数，讨论实验结果。

样本数量对实验结果影响不大，样本分布对实验结果影响较大。逻辑斯蒂回归分类器在样本集分布较离散可分时，分类效果会更好。当样本重叠部分较多时，分类器分类效果一般，无法很好的对样本进行分类。

上述题目算法中所用合适的学习率为$lr=0.01$。不同的学习率对算法收敛的速度产生影响。

1）学习率设置太小，需要花费过多的时间来收敛
2）学习率设置较大，在最小值附近震荡却无法收敛到最小值

上述题目算法中batch size取所有数据。不同的batch size对训练过程速度产生影响，当较小的批量(batch)时做完一次epoch时需要花费更多的时间，但一次更新所花费时间减少，并且较小的批量容易使算法跳出局部最小值，便于找到真正的最优解。所以一个适当的batch size对算法的性能非常重要。

上述题目算法中epoch=1000，当epoch较小时，迭代次数少使得算法运行时间少，但可能算法还没收敛；当epoch较大时，迭代次数多使得算法运行时间久，算法收敛到最小值，但可能算法早早就已达到收敛结果，造成了不必的迭代次数。