Lecture3编程作业

广义逆与梯度下降法代码见：[Linear Regression.py]()

第五题代码见：[function.py]()

1.广义逆和梯度下降法来求最小误差平方和最佳解的核心算法

广义逆：

```
def generalized_inverse(train_data,train_label):    #广义逆求解析解
    X=train_data
    Y=train_label.T
    A=np.dot(X.T,X)
    Apinv=np.linalg.pinv(A)
    Xw=np.dot(Apinv,X.T)
    w=np.dot(Xw,Y)
    return w
```

梯度下降法求最优解

```
def gradient(data,label,w):                         #求梯度
    datadim=len(train_data[0])
    L=np.zeros(datadim)
    for i in range(len(data)):
        L=np.add(L,(np.dot(w,data[i].T)-label[i])*data[i])
    return L

def descent(train_data,train_label,alpha,epoch):    #梯度下降求最优解
    datadim=len(train_data[0])
    w=np.zeros(datadim)
    lost=np.zeros(epoch)
    ep=np.zeros(epoch)
    for t in range(epoch):                          #epoch为算法迭代次数
        L=gradient(train_data,train_label,w)        #更新w，学习率为alpha
        w =np.subtract(w,alpha*L)
        ep[t]=t
        lost[t]=Lost(w,train_data,train_label)      #求每次迭代的损失函数
        if (L==np.zeros(datadim)).all():
            break;
    dp.epoch_line(lost,ep)                          #画出迭代次数与损失函数曲线
    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0.0001，迭代次数epoch设为100次

(2) 训练集和测试集上，两种算法的分类正确率

广义逆： $Accuracy_{(in)} = 1.0, Accuracy_{(out)} = 1.0$

梯度下降求最优解： $Accuracy_{(in)} = 1.0, Accuracy_{(out)} = 1.0$

(3) 两种算法的运行时间

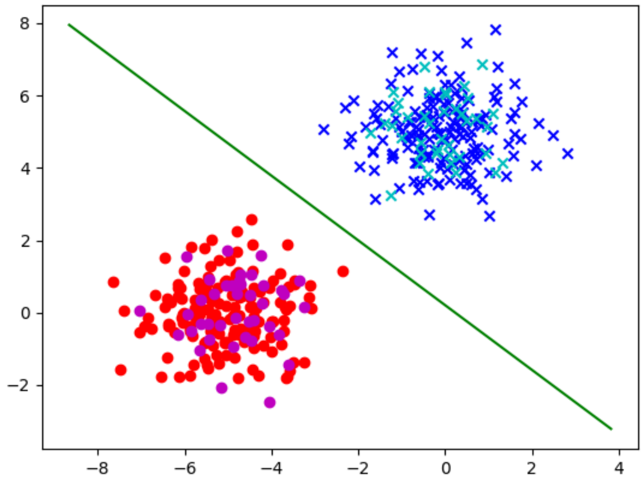
广义逆： Running time: 0.00098 Seconds

梯度下降求最优解： Running time: 0.307946 Seconds

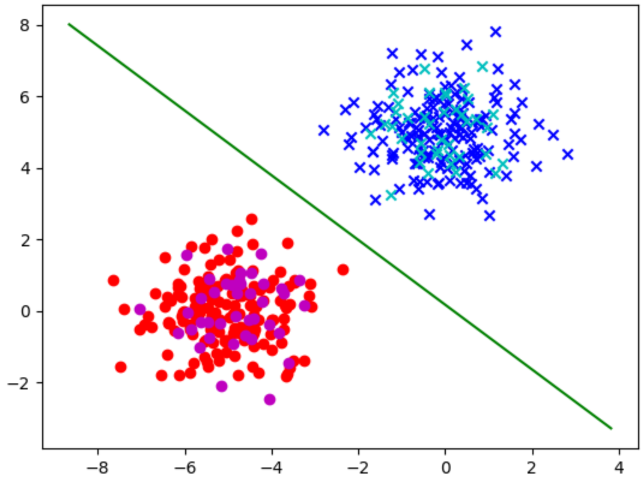
(4) 画出数据集和分类面

（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）

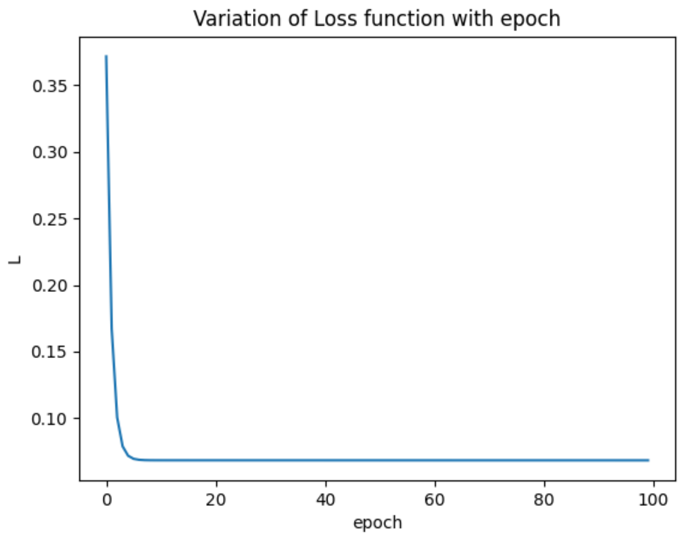
广义逆：



梯度下降求最优解：



(5) 损失函数随epoch增加的变化曲线



3.

(1) 数据集

重复第2题的内容，但数据集 X_1 和数据集 X_2 的均值向量分别改为 $m_1 = [1, 0]^T$和 $m_2 = [0, 1]^T$ ，其他不变。

本数据集中梯度下降求最优算法的学习率alpha设为0.0001，迭代次数epoch设为100次

(2) 训练集和测试集上，两种算法的分类正确率

广义逆： $Accuracy_{(in)} = 0.765625, Accuracy_{(out)} = 0.7625$

梯度下降求最优解： $Accuracy_{(in)} = 0.765625, Accuracy_{(out)} = 0.7625$

(3) 两种算法的运行时间

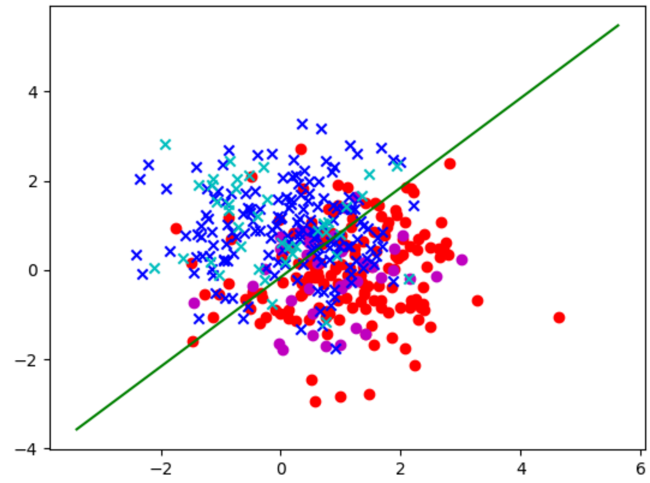
广义逆： Running time: 0.000872 Seconds

梯度下降求最优解： Running time: 0.32547 Seconds

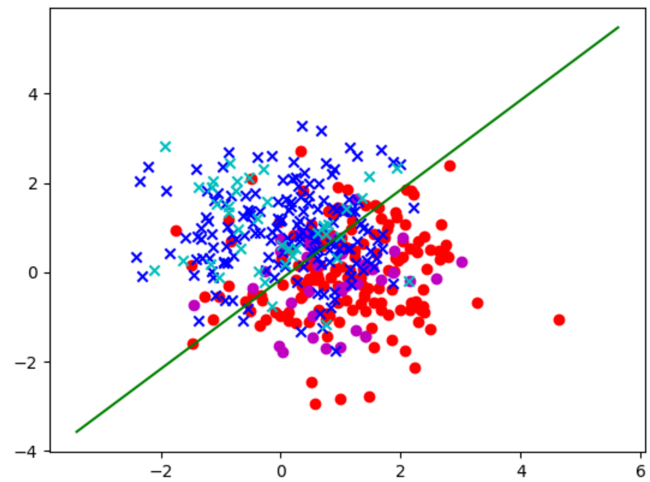
(4) 画出数据集和分类面

（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）

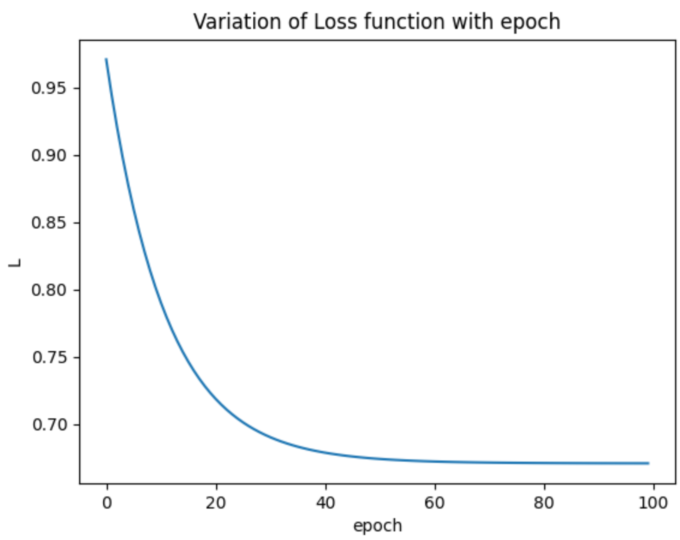
广义逆：



梯度下降求最优解：



(5) 损失函数随epoch增加的变化曲线



4.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。

广义逆:

广义逆算法不用设置超参数，根据公式可直接求出广义逆解。

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。线性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。

梯度下降求最优解:

梯度下降算法中的超参数为其学习率 lr，$batchsize$以及$epoch$ 。

上述题目算法中所用合适的学习率为$lr=0.0001$ 。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

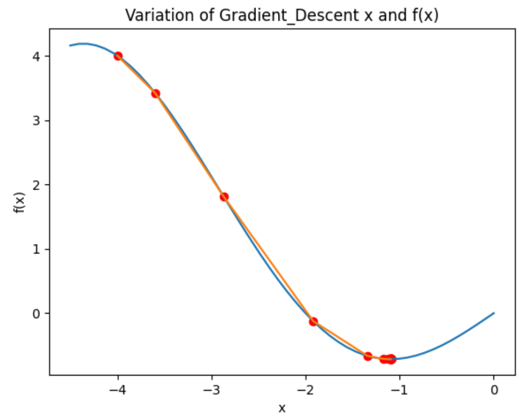
上述题目算法中batch size取所有数据。不同的batch size对训练过程速度产生影响，当较小的批量(batch)时做完一次epoch时需要花费更多的时间，但一次更新所花费时间减少，并且较小的批量容易使算法跳出局部最小值，便于找到真正的最优解。所以一个适当的batch size对算法的性能非常重要。

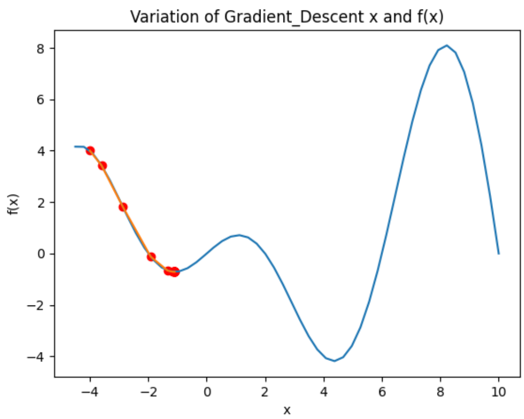
上述题目算法中epoch=1000，当epoch较小时，迭代次数少使得算法运行时间少，但可能算法还没收敛；当epoch较大时，迭代次数多使得算法运行时间久，算法收敛到最小值，但可能算法早早就已达到收敛结果，造成了不必的迭代次数。

5. f(x)=x*cos(0.25*πx)的各种下降算法

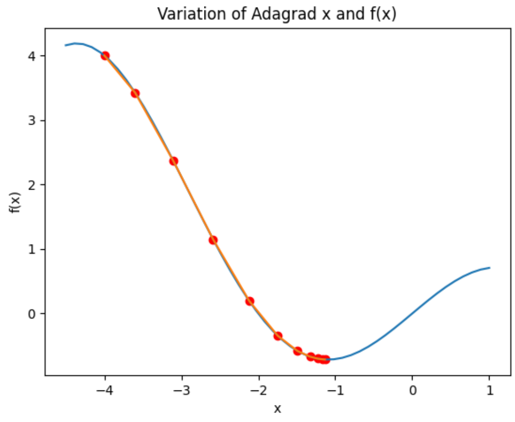
分别用梯度下降法、Adagrad、RMSProp、动量法（Momentum）和Adam共6种方法，**各迭代10次和50次。**

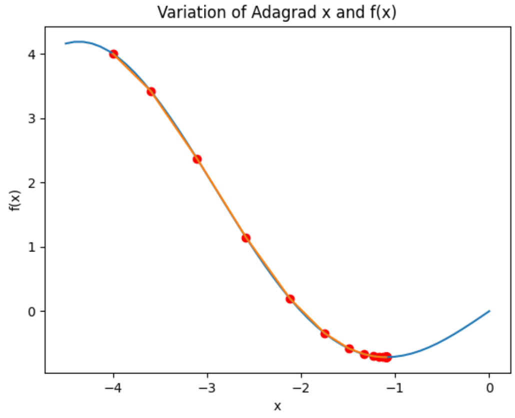
(1) Gradient-Descent:



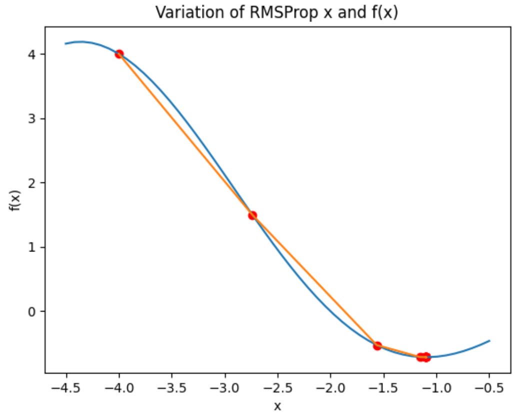


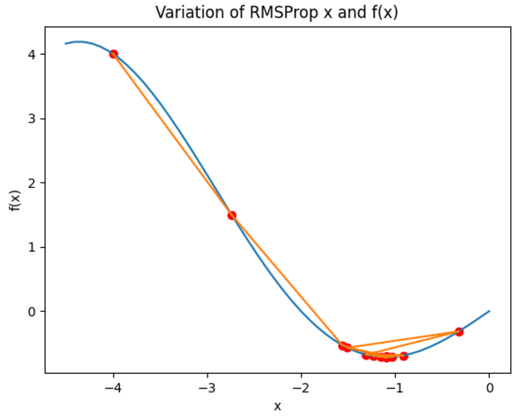
(2) Adagrad:



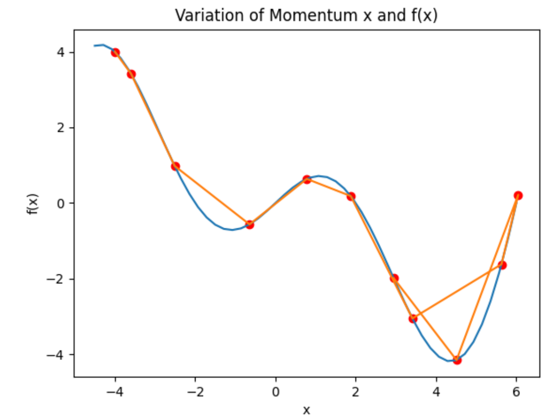


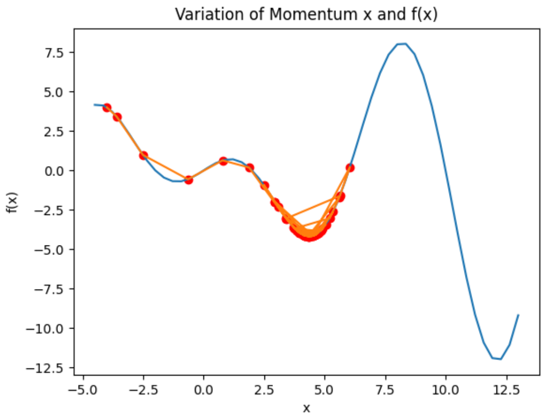
(3) RMSProp



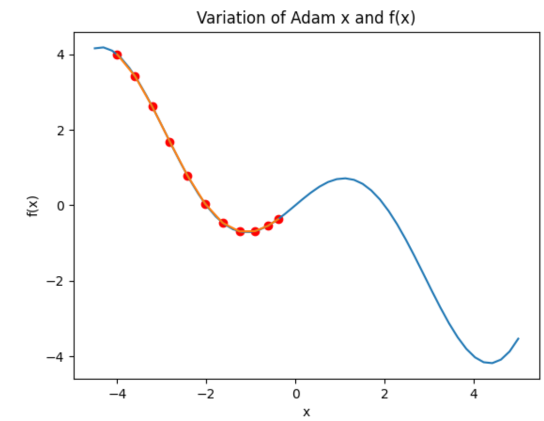


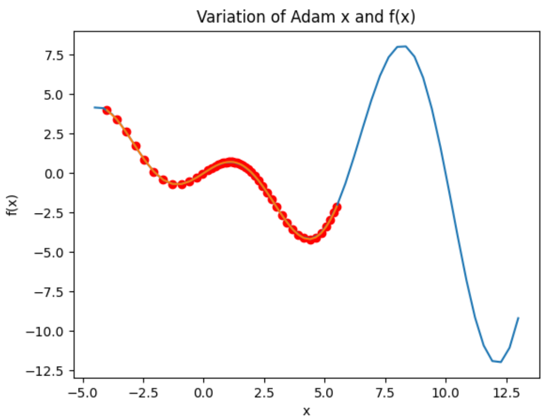
(4) Momentum


Variation of Momentum x and f(x)
f(x)
x
4
2
0
−2
−4
−4
−2
0
2
4
6


Variation of Momentum x and f(x)
f(x)
x
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

(5) Adam


Variation of Adam x and f(x)
f(x)
x
4
2
0
−2
−4
−4
−2
0
2
4


Variation of Adam x and f(x)
f(x)
x
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5

算法	优点	缺点
Gradient-Descent	目标函数为凸函数时，可以找到全局最优值	收敛速度慢，**需要用到全部数据，内存消耗大**
SGD	避免冗余数据的干扰，收敛速度加快，能够在线学习	更新值的方差较大，收敛过程会产生波动，可能落入极小值（卡在鞍点），选择合适的学习率比较困难（需要不断减小学习率）
Adagrad	实现**学习率的自动更改**	**依赖于人工设置一个全局学习率**，学习率设置过大，对梯度的调节太大。**中后期，梯度接近于0，使得训练提前结束**
RMSProp	可以缓解Adagrad学习率下降较快的问题，引入均方根，减少摆动，适合处理非平稳目标	依然依赖全局学习率
Momentum	引入动量概念，**更新的时候在一定程度上保留之前更新的方向**，能够在相关方向加速梯度下降，抑制振荡，从而加快收敛	需要人工设定学习率
Adam	速度快，对内存需求较小，为不同的参数计算不同的自适应学习率	在局部最小值附近震荡，可能不收敛