《机器学习基础》实验报告

<table>
<tr><td>年级、专业、班级</td><td colspan="3">2019 计算机科学与技术（卓越）02 班</td><td>姓名</td><td>钟祥新</td></tr>
<tr><td>实验题目</td><td colspan="5">对数几率回归算法实践</td></tr>
<tr><td>实验时间</td><td colspan="2">2021 年 10 月 24 日</td><td>实验地点</td><td colspan="2">DS1422</td></tr>
<tr><td>实验成绩</td><td colspan="2"></td><td>实验性质</td><td colspan="2">☐验证性 ☐设计性 ☐综合性</td></tr>
<tr><td colspan="6">教师评价：
☐算法/实验过程正确； ☐源程序/实验内容提交 ☐程序结构/实验步骤合理；
☐实验结果正确； ☐语法、语义正确； ☐报告规范；
其他：
评价教师签名：</td></tr>
</table>

一、实验目的

掌握线性模型、对率回归算法原理。

二、实验项目内容

1. 理解对率回归算法原理。
2. 编程实现对数几率回归算法。
3. 将算法应用于西瓜数据集、鸢尾花数据集分类问题。

三、实验过程或算法（源程序）

（一）Logistics 回归

1.1 模型原理

Logistics 回归是一种分类模型，由条件概率分布$P(Y|X)$表示。这里，随机变量X为实数，随机变量Y取值为 0 或 1。通过监督学习的方法估计模型的参数。

Logistics 回归模型是如下的条件概率分布：

$$P(Y=1|x)=\frac{\exp(w\cdot x+b)}{1+\exp(w\cdot x+b)}$$

$$P(Y=0|x)=\frac{1}{1+\exp(w\cdot x+b)}$$

对于给定的输入实例x，可以求得$P(Y=1|X)$和$P(Y=0|X)$，logistics 回归比较两个条件概率的大小，将实例x分到概率较大的那一类。

为了方便，将权值向量和输入向量加以扩充，仍记作w,x，得到 logistics 回归模

型如下：

$$P(Y=1|x)=\frac{\exp(w\cdot x)}{1+\exp(w\cdot x)}$$

$$P(Y=0|x)=\frac{1}{1+\exp(w\cdot x)}$$

现在考查 logistics 回归的特点。一个事件的几率是该事件发生的概率与不发生概率的比值。如果事件发生的概率是p，那么该事件的几率是$\frac{p}{1-p}$,则该事件的对数几率（log odds）是

$$logit(p)=log\frac{p}{1-p}$$

$$log\frac{P(Y=1|x)}{P(Y=0|x)}=w\cdot x$$

也就是说，输出$Y=1$的对数几率是由输入x的线性函数表示的模型，即 logistics 回归模型。

1.2 模型参数估计

现用极大似然估计法估计模型参数，从而得到 logistics 模型。

设：$P(Y=1|x)=f(x)$，$P(Y=0|x)=1-f(x)$，似然函数为：

$$\prod_{i=1}^{N}[f(x_i)]^{\{y_i\}}[1-f(x_i)]^{1-\{y_i\}}$$

对数似然函数为：

$$L(w)=\sum_{i=1}^{N}[y_i logf(x_i)+(1-y_i)\log(1-f(x_i))]$$

$$=\sum_{i=1}^{N}[y_i(w\cdot x_i)-\log(1+\exp(w\cdot x_i)]$$

对$L(w)$求极大值，得到w的估计值。下面推导梯度下降法公式。

由梯度下降法得:

$w:=\lambda\frac{\partial L(w)}{\partial w}$

$\frac{\partial L(w)}{\partial w}=\sum_{i=1}^{N}y_i x_i-\frac{e^{w\cdot x_i}}{1+e^{w\cdot x_i}}=\sum_{i=1}^{N}(\hat{y}_i-y_i)\cdot x_i$

$\therefore w:=\lambda\sum_{i=1}^{N}(\hat{y}_i-y_i)\cdot x_i$

在实现中，我们采用stochastics gradient descent 算法，减小计算量，每次随机取一个样本迭代.

在具体实现时，直接使用推导的公式，而不在代码中进行推导。

（二）实现逻辑

2.1 代码逻辑

本次实验代码结构见下图：

1. sigmoid 函数
2. 梯度下降估计参数函数
3. loss函数-记录loss随epoch变化
4. 评估函数-sklearn评估模型分类表现
5. 输出函数-打印分类结果
6. 可视化函数-分类结果可视化

Ps. 西瓜数据集只有 17 个样本，所以不划分训练集和测试集；鸢尾花数据集有 100 个样本（每个类别 50 个），我们将每个样本按照 4:1 划分为训练集和测试集。

```
feat = np.r_[iris.data[:40,:2],iris.data[50:90,:2]]#训练集 特征
label = np.r_[iris.target[:40],iris.target[50:90]] #训练集label
featTest = np.r_[iris.data[40:50,:2],iris.data[90:100,:2]] #测试集特征
labelTest = np.r_[iris.target[40:50],iris.target[90:100]] #测试集label
```

2.1.1 sigmoid 函数

实验中发现传入的x太小 Python 会提示溢出错误，因此当$x < -100$，令返回结果为 0。

```
'''
description: sigmoid函数，将函数值映射到(0,1)区间
param {*} X
return {*} 0 or 1.0/(1+exp(-X))
'''
def sigmoid(X):
    if (X < -100):
        return 0
    else:
        return 1.0/(1+exp(-X))
```

2.1.2 梯度下降估计参数函数

因为$L(w)$是凸函数，所以梯度下降一定能够求得极小值。这里我们令迭代次数为 1000 次，经检验最后收敛到极小值。

拓展 1：采用**改进的 SGD 算法，动态调节步长，**思想是“epoch 增加，减小学习率”，防止学习率过大越过极小值点的情况。

同时还设置一个 BGD 函数，用比较改进的 SGD 和基础梯度下降法的好坏。具体实现见代码。

```
'''
description: 梯度下降函数,估计模型参数,首先使用极大似然法得到对数似然函数,然后用梯度下降法估计模型参数.
             公式推导在实验报告中给出,此处直接用推导结果.
param {*} feat
param {*} label
return {*} weights
'''
def calW(feat, label):
    label = label
    featMat = np.mat(feat)
    m,n = featMat.shape #m是样本数量, n是特征数量
    featMat = np.c_[featMat,np.ones((m,1))]
    m,n = featMat.shape #m是样本数量, n是特征数量
    weights = np.ones((n,1)) #参数初始为1
    epoch = 1000 #迭代次数
    for i in range(epoch):
        idx = np.arange(m)
        for j in range(m):
            lam = 0.0001 + 4/(1+i+j) #epoch增加, 学习率降低
            randIdx = int(np.random.uniform(0,len(idx)))
            y_pre = sigmoid(np.sum(featMat[randIdx]*weights)) #预测值
            error = label[randIdx] - y_pre
            weights += lam*error*featMat[randIdx].transpose() #更新
            #idx = np.delete(idx,idx[randIdx])
        y_pre_arr = featMat.dot(weights)
        for i in range(len(y_pre_arr)):
            y_pre_arr[i] = sigmoid(y_pre_arr[i])
        y_pre_arr = y_pre_arr.round().flatten()
        loss_tmp = np.sum(abs(y_pre_arr - label))
        loss.append(loss_tmp)
    return weights
```

```
'''
description: Batch Gradient Descent, 对比SGD
param {*} feat
param {*} label
return {*} weights
'''
def BGD(feat, label):
    loss.clear()
    label = label
    m,n = feat.shape #m是样本数量, n是特征数量
    feat = np.c_[feat,np.ones((m,1))]
    m,n = feat.shape #m是样本数量, n是特征数量
    weights = np.ones((n,1)) #参数初始为1
    epoch = 1000 #迭代次数
    lam = 0.01 #BGD采用固定学习率
    error = np.zeros((n,1))
    for i in range(epoch):
        y_pre = feat.dot(weights)
        for j in range(len(y_pre)):
            y_pre[j] = sigmoid(y_pre[j])
        y_pre = y_pre.round()
        loss.append(np.sum(y_pre-label))
        for j in range(len(label)):
            tmp = (label[j]-y_pre[j])*feat[j]
            tmp = tmp.reshape((3,1))
            error += tmp
        weights += (1/m)*lam*error
    return weights
```

2.1.3 loss 函数

逻辑简单，可视化 loss 随 epoch 的变化

```python
'''
description: 展示迭代过程loss的变化情况
param {*} loss
return {*} none
'''
def showLoss(loss):
    loss = loss[::10]
    length = len(loss)
    x = np.arange(length)
    plt.plot(x,loss)
    plt.xlabel("Epoch")
    plt.ylabel("Loss")
    plt.show()
```

2.1.4 模型评估函数

拓展 2： 调用 sklearn 库的 metrics 子包，给出本 logistics 模型分类的 $accuracy, precision\ rate, recall\ rate$ 和 $f1\ score$，综合性评价模型表现。

```python
'''
description: 使用sklearn.metrics评价模型表现,并打印结果
param {*} feat
param {*} weights
param {*} label
param {*} num
return {*}
'''
def printResult(feat,weights,label,num):
    featMat = np.c_[feat,np.ones((num,1))]
    y_pre = featMat.dot(weights)
    for i in range(len(y_pre)):
        y_pre[i] = sigmoid(y_pre[i])
    y_pre = y_pre.round().flatten()
    acc = mtc.accuracy_score(label,y_pre)
    precision = mtc.precision_score(label, y_pre, labels=None, pos_label=1, average='binary') #查准率
    recall = mtc.recall_score(label, y_pre, labels=None, pos_label=1, average='binary', sample_weight=None) #查全率
    f1 = mtc.f1_score(label, y_pre, labels=None, pos_label=1, average='binary', sample_weight=None)
    Confusion = mtc.confusion_matrix(label,y_pre,labels=None, sample_weight=None, normalize=None)
    print("Accuracy: ", acc)
    print("Precision rate:", precision)
    print("Recall rate: ", recall)
    print("F1 score: ", f1)
    print("Confusion metrix: ", Confusion)
```

2.1.5 可视化函数

由于可视化需要，此处仅选择二维特征作为展示，画出分类结果，可视化评价 logistics 模型表现，结果见下节*实验结果及分析*。

```python
'''
description: 画出分类结果，为了可视化仅选择两个特征作分类，在拓展部分增加特征数量
param {*} feat
param {*} label
param {*} weights
return {*} none
'''
def plotResult(feat,label,weights):
    x1 = []
    y1 = []
    x2 = []
    y2 = []
    length = len(label)
    for i in range(length):
        if (label[i] == 1):
            x1.append(feat[i][0])
            y1.append(feat[i][1])
        else:
            x2.append(feat[i][0])
            y2.append(feat[i][1])
    plt.scatter(x1,y1,marker='o', c = 'b', s = 30, label = 'cat1')
    plt.scatter(x2,y2,marker='o', c = 'r', s = 50, label = 'cat2')
    plt.xlabel("Density")
    plt.ylabel("Sugar Rate")
    plt.title("Watermelon Result")
    MINN = min(np.min(x1),np.min(x2))
    MAXX = max(np.max(x1),np.max(x2))
    x = np.arange(MINN,MAXX,0.1)
    y = (-weights[2]-weights[0]*x)/weights[1]
    plt.plot(x,y)
    plt.show()
```

四、实验结果及分析

（一）分类结果

1.1 鸢尾花数据集

1.1.1 模型评估

下面是使用 SGD 方法估计权重的模型评估（训练集）:

```
Accuracy:  1.0
Precision rate: 1.0
Recall rate:  1.0
F1 score:  1.0
Confusion metrix:  [[40  0]
 [ 0 40]]
```

下面是使用 SGD 方法估计权重的模型评估（测试集）:

```
Accuracy:  0.95
Precision rate: 0.9090909090909091
Recall rate:  1.0
F1 score:  0.9523809523809523
Confusion metrix:  [[ 9  1]
 [ 0 10]]
```

可以看出，在训练集上的分类效果非常好，所有样本均能够正确分类。

下面是使用 BGD 方法估计权重的模型评估（训练集）：

```
Accuracy:  0.9875
Precision rate: 0.975609756097561
Recall rate:  1.0
F1 score:  0.9876543209876543
Confusion metrix:  [[39  1]
 [ 0 40]]
```

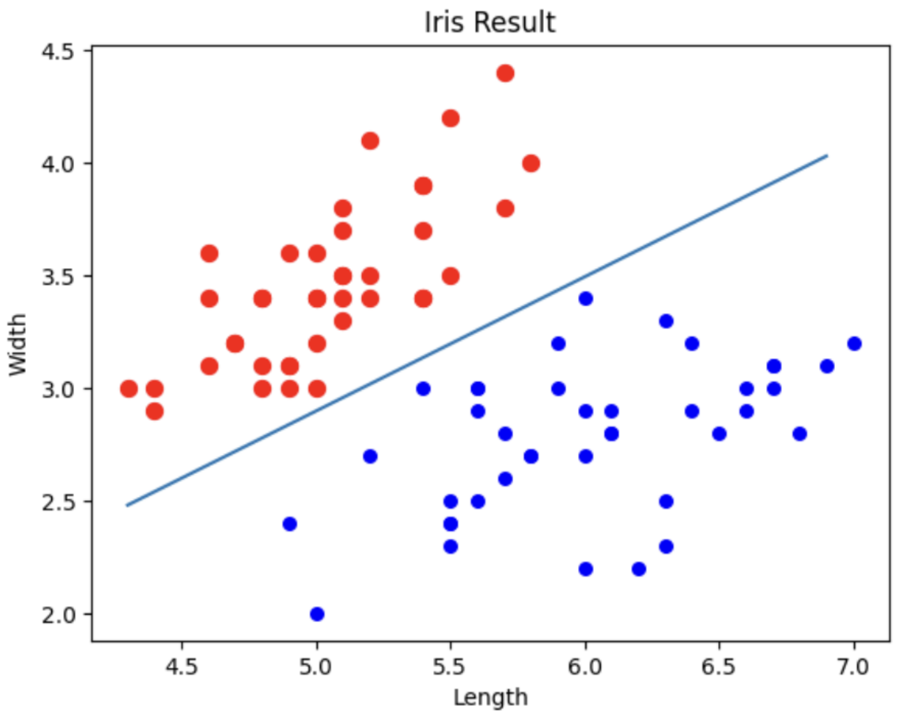
下面是使用 BGD 方法估计权重的模型评估（测试集）：

```
Accuracy:  0.95
Precision rate: 0.9090909090909091
Recall rate:  1.0
F1 score:  0.9523809523809523
Confusion metrix:  [[ 9  1]
 [ 0 10]]
```

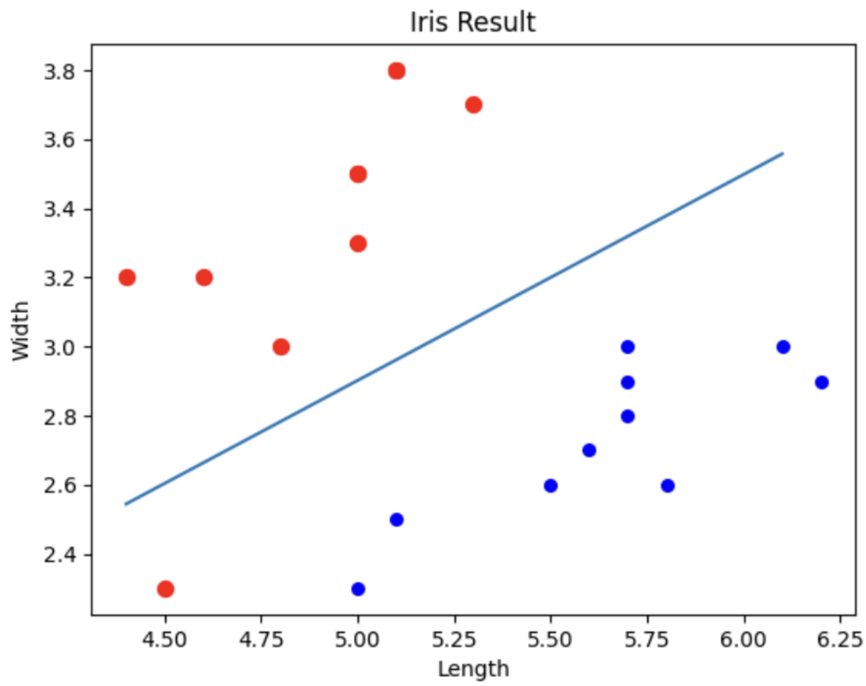
可以看出，由于鸢尾花数据集良好的性质（同类别聚集，不同类分散），两种方法估计的参数相差不大，都是极小值，模型表现也差不多。

1.1.2 分类结果可视化

下面是 SGD 分类结果（训练集）



下面是使用 SGD 的分类结果（测试集））

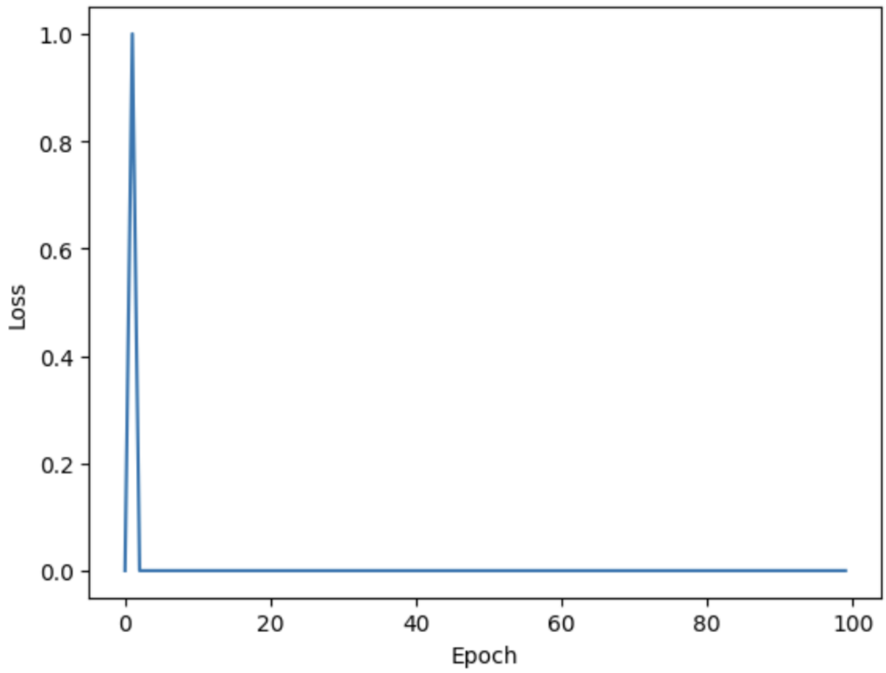


这里挑选长度和宽度作为特征，可以看出，训练的 logistics 模型能够较好地划分两类鸢尾花。除了少数异常数据，其余数据均能够正确分类。

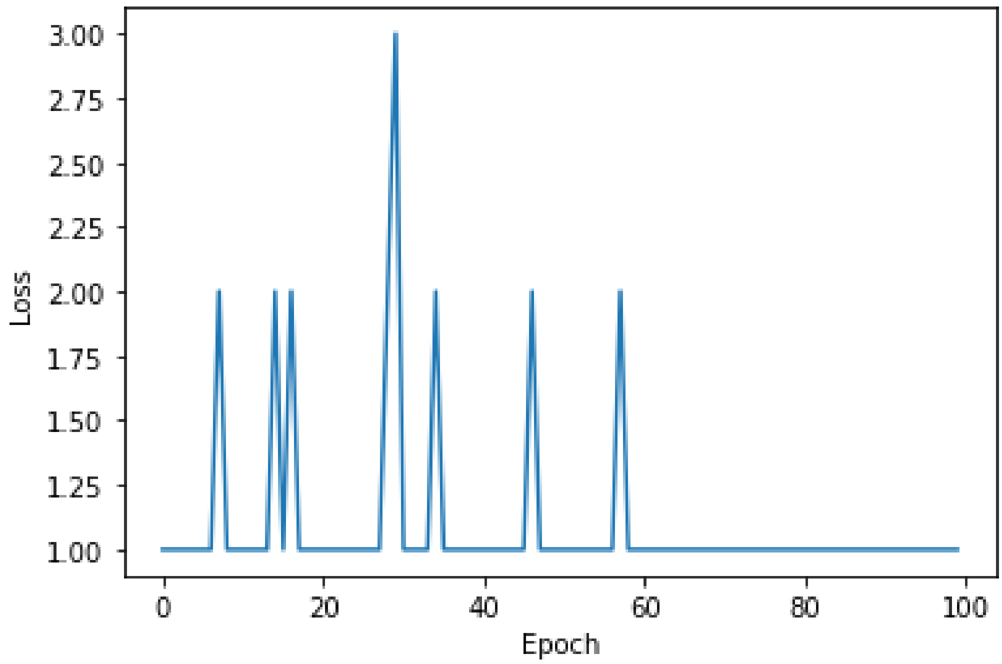
Ps. 补充蓝色分类线如何作出：在画图时，我们把两个特征定义为x, y并作图。我们把$p = 0.5$作为划分标准，当$p = 0.5$时，对数几率为 **0**（见实验原理部分），已知一个特征X，可以根据权重WEIGHTS求出另一个特征Y，由此作出蓝色分类线。

1.1.3 loss-epoch

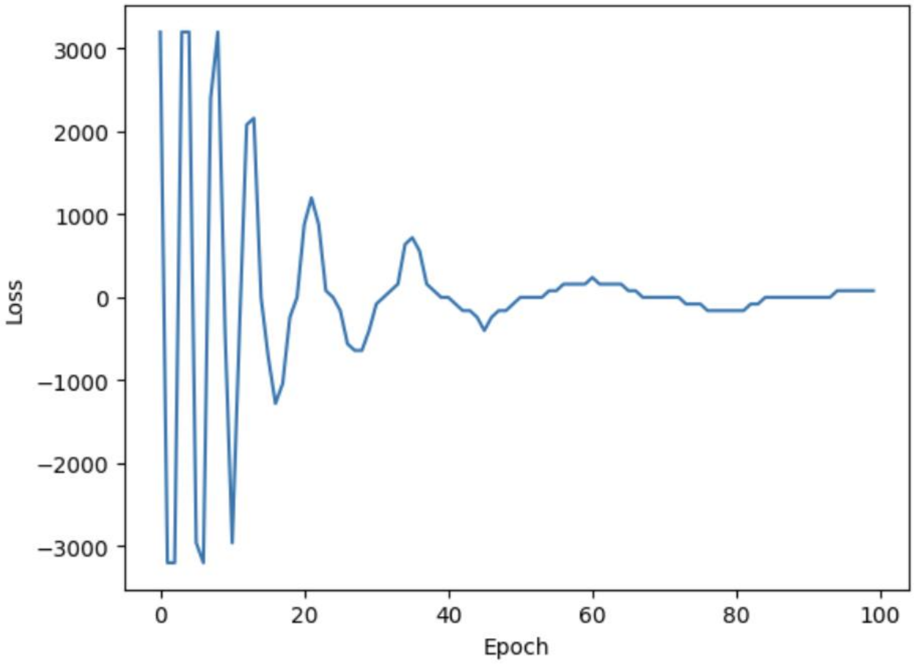
下面是 SGD 的 loss-epoch 图（训练集）：



下面是 SGD 的 loss-epoch 图（测试集）：



下面是 BGD 的 loss-epoch 图：



可以看出，无论是 SGD 还是 BGD，由于数据集本身线性可分，所以迭代一定 epoch 后，weights 收敛于极小值，loss 趋近于 0。对于本数据集使用 SGD 和 BGD 收敛速度不同，但是都能取得极小值。

1.2 西瓜 3.0 数据集

1.2.1 模型评估

下面是 SGD 的模型评估：

```
Accuracy:  0.7647058823529411
Precision rate: 0.8333333333333334
Recall rate:  0.625
F1 score:  0.7142857142857143
Confusion metrix:  [[8 1]
 [3 5]]
```

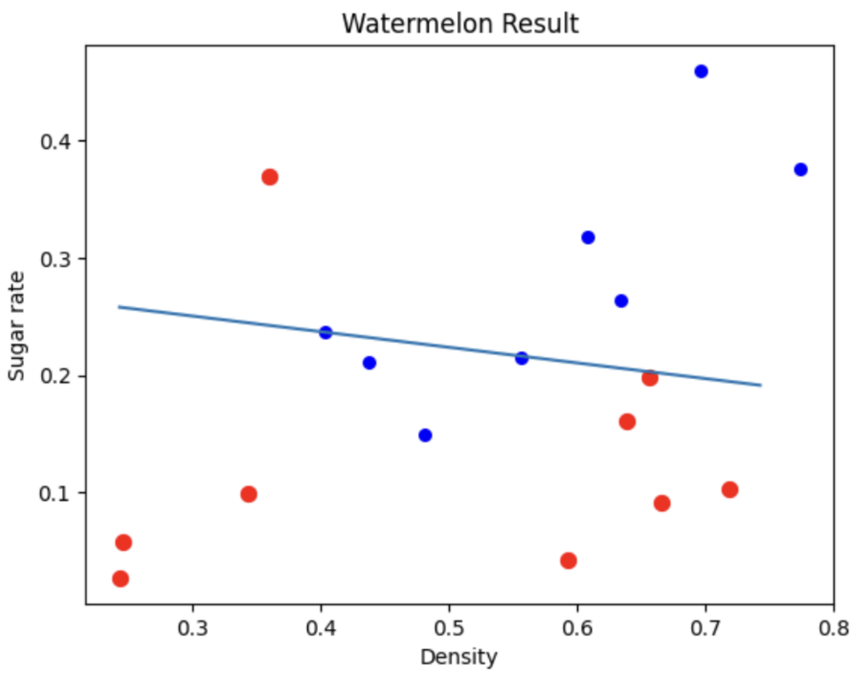
下面是 BGD 的模型评估：

```
Accuracy:  0.8235294117647058
Precision rate: 0.7272727272727273
Recall rate:  1.0
F1 score:  0.8421052631578948
Confusion metrix:  [[6 3]
 [0 8]]
```

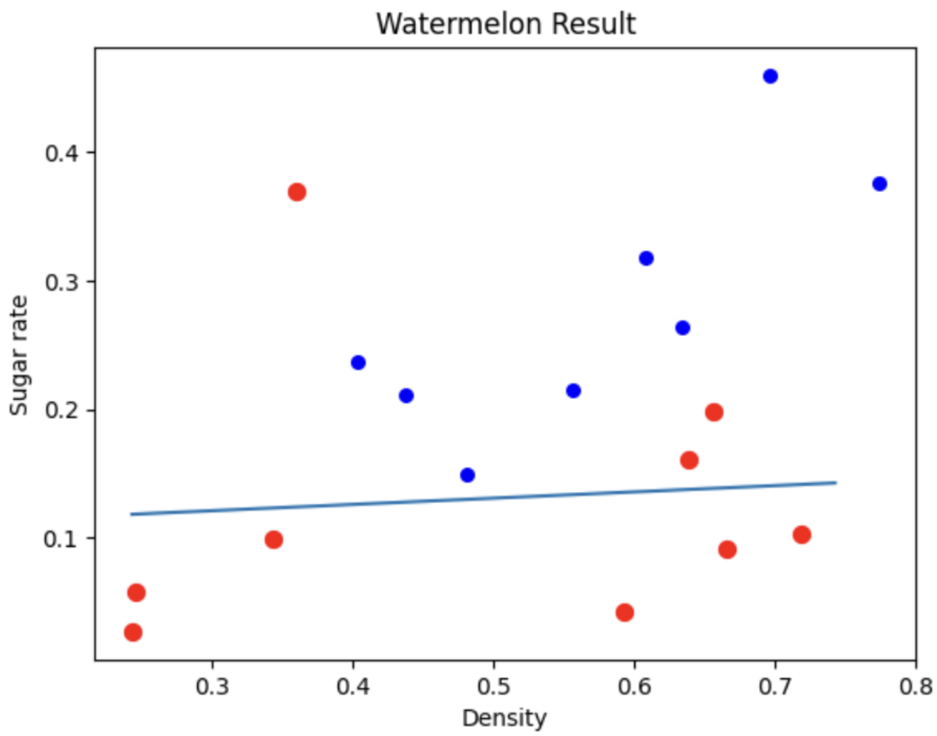
由于西瓜数据集样本很少（只有 17 个）并且可能存在线性不可分的问题（见下一节），使用 SGD 的随机性很高，估计参数时的震荡很强，并没有一直趋于极小值迭代，所以导致最终估计的效果没有使用 BGD 好。

1.2.2 分类结果可视化

下面是 SGD 的分类结果：



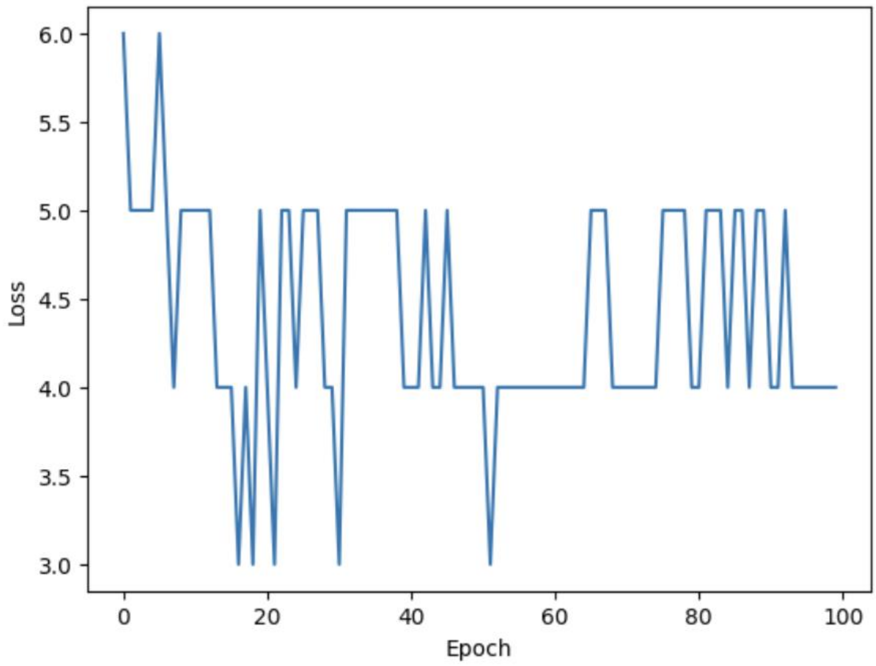
下面是 BGD 的分类结果：



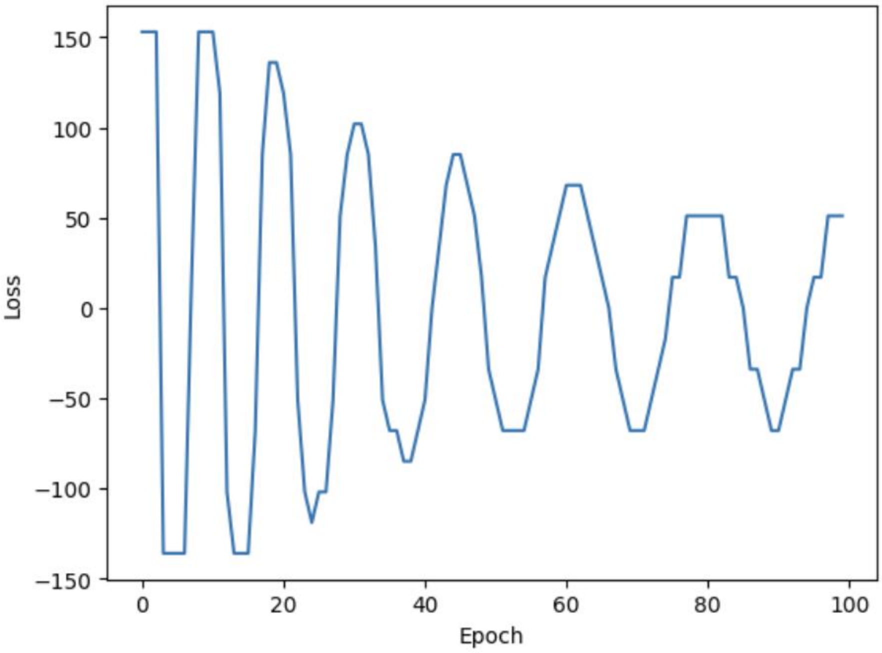
经分析，从图片观察来看，猜想西瓜数据集可能线性不可分，所以使用 logistics 分类模型无法正确分类。需要使用 SVM 软间隔法等非线性分类器求解。

1.2.3 loss-epoch

下面是 SGD 的 loss-epoch 图像：



下面是 BGD 的 loss-epoch 图像：



西瓜数据集的 loss 抖动很大，并且两次求解最后都没有趋近于 0，说明没有收敛于**最优解**。

（二）实验总结

本次实验设计并实现了对数几率回归模型，并用于西瓜数据集和鸢尾花数据集。我手动编程实现所有算法细节。并在完成基本要求的基础上进行探索，实现了动态调节学习率和 sklearn 评估模型表现，给出了评价指标。

通过本次实验，我加深了对 logistics 模型原理和应用的理解。