《机器学习基础》实验报告

年级、专业、班级	2021 级计算机科学与技术 4 班	姓名	胡 鑫
实验题目	对数几率回归算法实践		
实验时间	2023 年 4 月 6 日	实验地点	DS3402
实验成绩		实验性质	☐验证性 ☐设计性 ☐综合性

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：

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）

1. 对率回归是一种分类算法，也是一种基于概率统计的线性分类模型，它的原理是通过寻找最佳分类边界，将给定的数据集分成两类，然后通过 sigmoid 函数将线性回归的输出映射到 0 到 1 的范围内，得到预测的概率值。在线性回归模型中，单位阶跃函数既不连续也不可微，而我们在做优化任务时，最好是让目标函数连续可微，于是就用到了对数几率函数 $y=\frac{1}{1+e^{-z}}$，它是一种“Sigmoid”函数，将对数几率函数变形为 $\ln\frac{y}{1-y}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$，其中 $\ln\frac{y}{1-y}$ 是样本作为正例的相对可能性的对数。将 y 视为类后验概率，于是可以将 $\ln\frac{y}{1-y}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$ 变为 $\ln\frac{p(y=1\mid\boldsymbol{x})}{p(y=0\mid\boldsymbol{x})}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$。在

二分类问题中，损失函数为$L=-[y\log\hat{y}+(1-y)\log(1-\hat{y})]$，当真实值 y 为 1 时，输出值 ŷ越接近 1，则 L 越小，当真实值 y 为 0 时，输出值 ŷ越接近于 0，L 越小。

由于希望每个样本属于真实标记的概率越大越好，即损失最小，所以用极大似然法求解最大化对数似然函数$\ell(\boldsymbol{w},b)=\sum_{i=1}^{m}\ln p(y_i\mid\boldsymbol{x}_i;\boldsymbol{w}_i,b)$，再转化为最小化负对数似然函数求解，令$\boldsymbol{\beta}=(\boldsymbol{w};b)\hat{\boldsymbol{x}}=(\boldsymbol{x};1)$，则将$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$简写为$\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}$，再令$p_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})=p(y=1\mid\hat{\boldsymbol{x}};\boldsymbol{\beta})$，

$p_0(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})=p(y=0\mid\hat{\boldsymbol{x}};\boldsymbol{\beta})=1-p_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})$，则似然项可以写为

$p(y_i\mid\boldsymbol{x}_i;\boldsymbol{w}_i,b)=y_ip_1(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})+(1-y_i)p_0(\hat{\boldsymbol{x}}_i;\boldsymbol{\beta})$，所以等价形式为最小化$\ell(\boldsymbol{\beta})=\sum_{i=1}^{m}\left(-y_i\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}_i+\ln\left(1+e^{\beta^{\mathrm{T}}\hat{\boldsymbol{x}}_i}\right)\right)$，求解$\boldsymbol{\beta}^*=\arg\min_{\boldsymbol{\beta}}\ell(\boldsymbol{\beta})$

2.

```
import numpy as np
class LogisticRegression:
    def __init__(self, learning_rate=0.01,
num_iterations=1000, print_cost=False):
        self.learning_rate = learning_rate      #学习率
        self.num_iterations = num_iterations    #迭代次数
        self.print_cost = print_cost
        self.theta = None

    #sigmoid 函数
    def sigmoid(self, z):
        return 1.0 / (1 + np.exp(-z))

    #代价函数和梯度下降算法
    def cost_function(self, X, y, theta):
        m = X.shape[0]
        h = self.sigmoid(X.dot(theta))
        cost = (-1/m) * np.sum(y * np.log(h) + (1 - y) * np.log(1
- h))#代价函数
        grad = (1/m) * X.T.dot(h - y)
        return cost, grad

    #训练模型，X 是训练集的特证矩阵，y 是训练集的标签向量
```

```
    def fit(self, X, y):
        m = X.shape[0]
        n = X.shape[1]
        self.theta = np.zeros((n, 1))
        for i in range(self.num_iterations):
            cost, grad = self.cost_function(X, y, self.theta)
            self.theta = self.theta - self.learning_rate *
grad#使用梯度下降算法更新 theta
            if self.print_cost and i % 100 == 0:
                print(f"Cost after iteration {i}: {cost}")

    #对新样本进行分类预测，X 是新样本的特征矩阵
    def predict(self, X):
        return np.round(self.sigmoid(X.dot(self.theta)))#返
回的结果为预测的标签向量
3.（1）
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 读取数据集
df = pd.read_csv('watermelon3.0a.csv')

# 将标签转换为 0 和 1
df['好瓜'] = df['好瓜'].map({'是': 1, '否': 0})

# 将数据集中的特征和标签进行分离
X = df.values[:,1:-1]
y = df.values[:, -1]

# 定义逻辑回归模型
class LogisticRegression:
    def __init__(self):
        self.w = None

    def sigmoid(self, Z):
        return 1 / (1 + np.exp(-Z))

    def train(self, X, y, learning_rate=0.1,
num_iterations=1000):
        m, n = X.shape
        self.w = np.zeros((n, 1))

        for i in range(num_iterations):
```

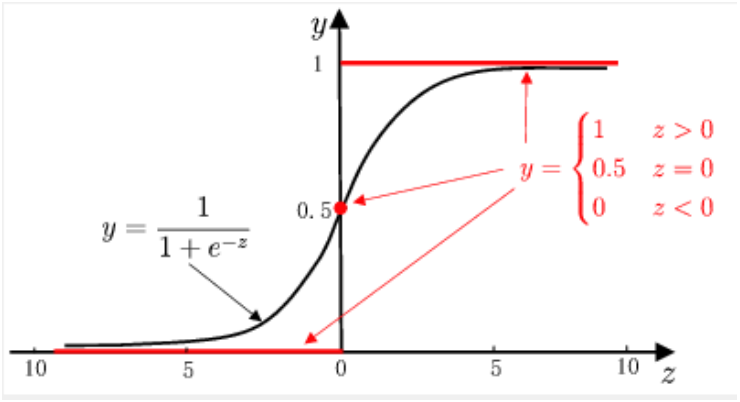
```
            Z = X.dot(self.w)
            A = self.sigmoid(Z)
            dz = A - y.reshape(-1, 1)
            dw = X.T.dot(dz) / m
            self.w -= learning_rate * dw

        return self.w

    def predict(self, X, threshold=0.5):
        Z = X.dot(self.w)
        y_pred = np.int32(self.sigmoid(Z) > threshold)
        return y_pred.flatten()

# 将数据集划分为训练集和测试集
X_train = np.vstack((X[0:6], X[9:]))
y_train = np.hstack((y[0:6], y[9:]))
X_test = X[6:9]
y_test = y[6:9]

# 训练和测试模型
model = LogisticRegression()
w = model.train(X_train, y_train)
y_pred = model.predict(X_test)

# 可视化分界线和样本点
plt.scatter(X[:, 0], X[:, 1], c=y)
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200),
np.linspace(y_min, y_max, 200))
X_grid = np.c_[xx.ravel(), yy.ravel()]
Z = model.predict(X_grid)
Z = Z.reshape(xx.shape)
plt.contour(xx, yy, Z, colors='k')
plt.xlabel('density')
plt.ylabel('sugar_content')
plt.title('Logistic_Regression')
plt.show()
accuracy = np.mean(np.int32(y_pred == y_test))
print("Test accuracy: ", accuracy)
（2）
import numpy as np
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
```

```
def load_data():
    iris = load_iris()
    X = iris.data[:,:2]# 只选取前两个特征维度以方便可视化
    y = iris.target
    return X, y

# 对数据集按照类别进行切分
X,y=load_data()#加载数据集
X_setosa = X[y == 0]
X_versicolor = X[y == 1]
X_virginica = X[y == 2]

#绘制数据
def plot_data(X_setosa, X_versicolor, X_virginica):
    plt.scatter(X_setosa[:, 0], X_setosa[:, 1], marker='o',
label='setosa')
    plt.scatter(X_versicolor[:, 0], X_versicolor[:, 1],
marker='x', label='versicolor')
    plt.scatter(X_virginica[:, 0], X_virginica[:, 1],
marker='*', label='virginica')
    plt.xlabel('Sepal length')
    plt.ylabel('Sepal width')
    plt.title('Logistic_Regression')
    plt.legend()
    plt.show()

plot_data(X_setosa[:, :2], X_versicolor[:, :2],
X_virginica[:, :2])

def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))

class LogisticRegression:
    def __init__(self, alpha=0.001, num_iter=100000):
        self.alpha = alpha    # 学习率
        self.num_iter = num_iter  # 迭代次数

    def fit(self, X, y):
        # 在特征向量中添加一列 1，以便进行截距的训练
        X = np.hstack([np.ones([X.shape[0], 1]), X])   # 在
鸢尾花数据集中为[1, X1, X2, X3, X4]
        self.coef_ = np.zeros(X.shape[1])   # 初始化权重
```

```
        # 梯度下降法
        for i in range(self.num_iter):
            z = np.dot(X, self.coef_)
            h = sigmoid(z)
            grad = np.dot(X.T, (h - y)) / y.size
            self.coef_ -= self.alpha * grad

    def predict(self, X):
        X = np.hstack([np.ones([X.shape[0], 1]), X])
        z = np.dot(X, self.coef_)
        h = sigmoid(z)
        return np.round(h)

classifier = LogisticRegression()
X_train = np.vstack([X_setosa[:25], X_versicolor[:25],
X_virginica[:25]])
y_train = np.array([0]*25 + [1]*25 + [2]*25)
classifier.fit(X_train[:, :2], (y_train == 2).astype(int))
# 输出模型在训练集上的准确率
y_pred = classifier.predict(X_train[:, :2])
accuracy = np.mean(y_train == 2 * y_pred)
print("Accuracy:", accuracy)


#鸢尾花代码二：sklearn 实现多分类的决策边界绘制
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

# 加载数据集
iris = load_iris()
X = iris.data[:, :2]  # 取前两个特征以便可视化
y = iris.target

# 模型训练（逻辑回归模型）
clf_lr = LogisticRegression(multi_class='multinomial',
solver='lbfgs')
clf_lr.fit(X, y)

# 绘制决策边界
def plot_decision_boundary(clf, X, y):
    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
```

```
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
np.arange(y_min, y_max, 0.01))
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, alpha=0.4)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')

# 绘制逻辑回归模型的决策边界
plot_decision_boundary(clf_lr, X, y)
plt.title('Logistic Regression Model')
plt.show()
```

四、实验结果及分析

1. 

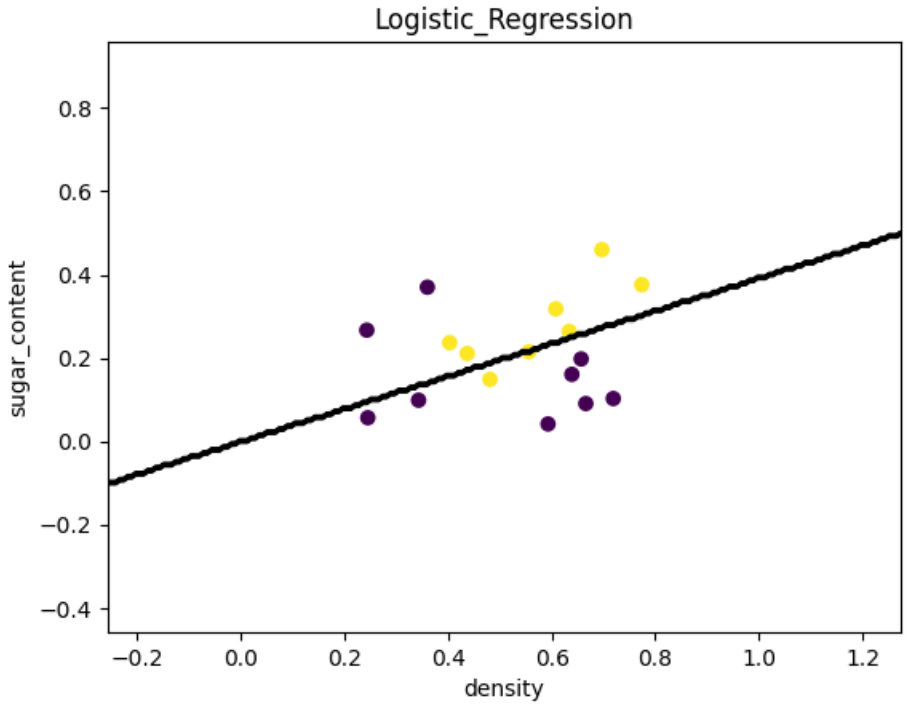
2. 在对率回归算法中，定义了一个 LogisticRegression 类，在其中包含初始化，sigmoid 函数，代价函数和梯度下降算法，还有一个训练模型 fit 函数，再定义了一个 predict 函数，可以对新样本进行分类预测。

3. （1）数据集如图所示

watermelon3.0a.csv

```
编号,密度,含糖率,好瓜
1,0.697,0.46,是
2,0.774,0.376,是
3,0.634,0.264,是
4,0.608,0.318,是
5,0.556,0.215,是
6,0.403,0.237,是
7,0.481,0.149,是
8,0.437,0.211,是
9,0.666,0.091,否
10,0.243,0.267,否
11,0.245,0.057,否
12,0.343,0.099,否
13,0.639,0.161,否
14,0.657,0.198,否
15,0.36,0.37,否
16,0.593,0.042,否
17,0.719,0.103,否
```

作出分类图形，黄色点和紫色点分别代表正类和负类，可以看出还是有一些点被错分。

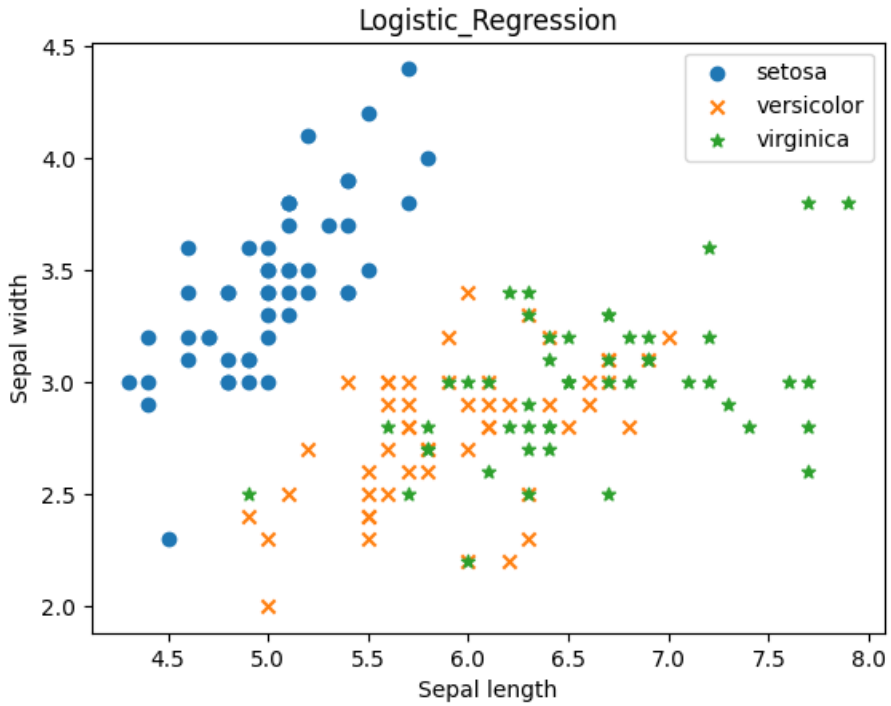


将对率回归算法应用到西瓜数据集 3.0a 上，排除编号对分类的影响后，

得到的正确率为 66.67%。

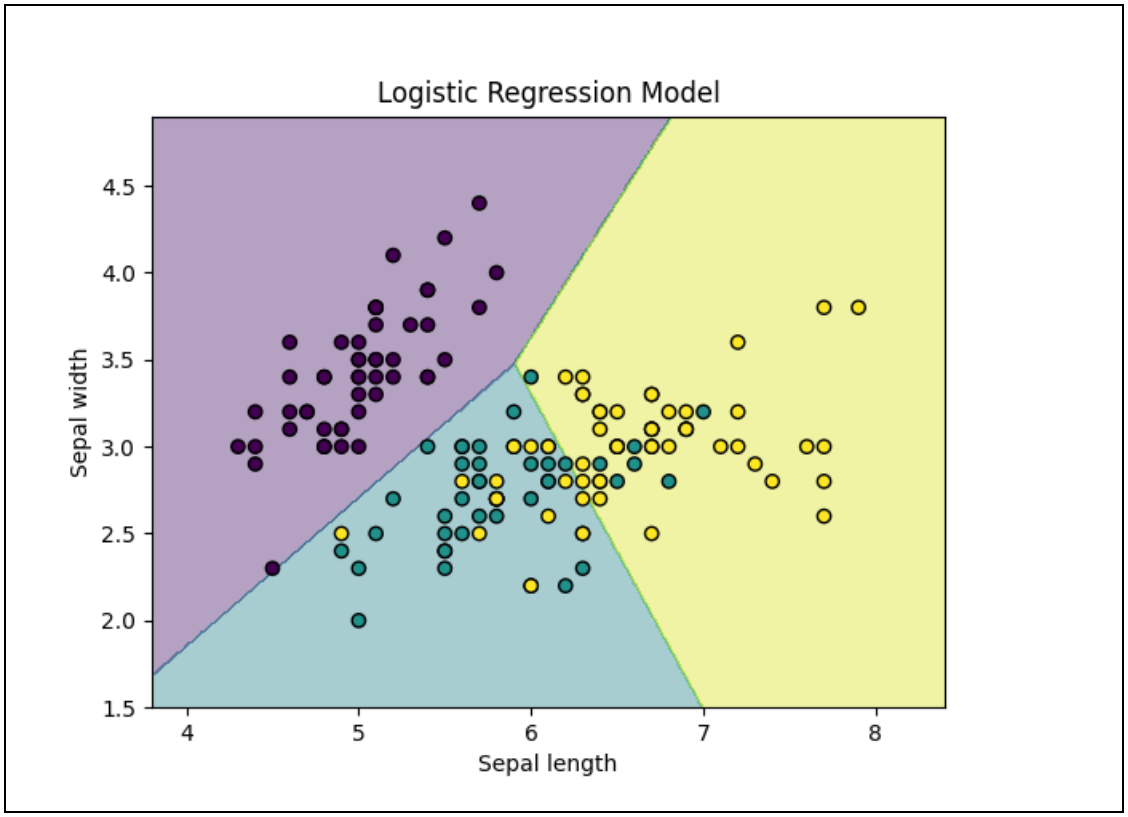
```
调试控制台  输出  问题  终端                                    Python
PS D:\ALL_Project\VSCode_Files> & D:/Python-3.11.2/python.exe d:/ALL_P
roject/VSCode_Files/PythonCode/Lab1/WaterMelon_LR.py
Test accuracy:  0.6666666666666666
PS D:\ALL_Project\VSCode_Files>
```

（2）下图作出了三种不同的鸢尾花的散点图，由图中的数据可以看出变色鸢尾和维吉尼亚鸢尾的花萼长度和宽度比较接近。用逻辑回归算法实现三种鸢尾花的的分类问题，其最后得到的准确率为 48%。




```
调试控制台  输出  问题  终端                                    Python
PS D:\ALL_Project\VSCode_Files> & D:/Python-3.11.2/python.exe d:/ALL_Project/VSCode_Files/Pyth
onCode/Lab1/Iris_LRnew.py
Accuracy: 0.48
PS D:\ALL_Project\VSCode_Files>
```

由于绘制多分类算法的决策边界可以通过直接调用 sklearn 中的函数实现，所以代码二展示了绘制鸢尾花实现三分类的决策边界，展示效果比前一幅图片的散点图更好。决策边界图如下图所示：


Logistic Regression Model
Sepal width
Sepal length
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4
5
6
7
8