黄冈师范学院
HUANGGANG NORMAL UNIVERSITY

人工智能与机器学习

Artificial Intelligence and Machine Learning

章节：实验6-逻辑回归之信用卡逾期情况预测

教师：刘重

学院：计算机学院

厚德 博学 力行 致远

一、实验目的

- (1) 逻辑回归算法原理；
- (2) 精确率和召回率；
- (3) 实例应用—信用卡逾期情况预测。

一、问题描述

二、实验步骤

1.加载数据集

2. 绘制数据的散点图，查看数据分布情况

3.定义Sigmoid、损失函数，使用梯度下降确定模型参数

4、初始化模型，并对模型进行训练

5、根据得到的参数，绘制模型分类线

6、绘制损失函数变化曲线

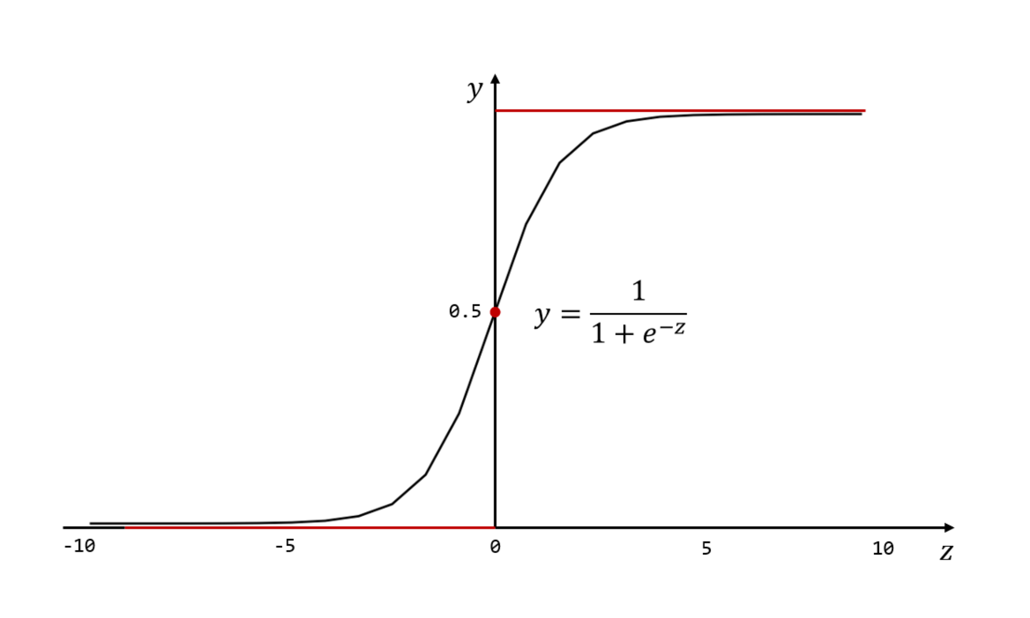
三、实验结果分析

逻辑回归模型原理与推导

- 线性模型如何执行分类任务呢？只需要找到一个单调可微函数将分类任务的真实标签y与线性回归模型的预测值进行映射。在线性回归中，模型的学习目标是直接逼近真实标签y，但在逻辑回归中，我们需要找到一个映射函数将线性回归模型的预测值转化为0/1值。

- sigmoid函数正好具备上述条件，单调可微、取值范围为(0, 1)，且具有较好的求导特性。

$$y = \frac{1}{1 + \mathrm{e}^{-z}}$$



三、实验原理

- 线性回归模型的公式为:

$$z=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$$

- 将上式代入到sigmoid函数中:

$$y=\frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}$$

- 两边取对数并转换为:

$$\ln\frac{y}{1-y}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$$

- 如果将y看作样本$\boldsymbol{x}$作为正例的可能性，那么$1-y$即为样本作为反例的概率。$\frac{y}{1-y}$也称“几率”（odds），对几率取对数则得到对数几率，所以上式为对数几率回归建模公式。

- 为了确定上式中的模型参数$\boldsymbol{w}$和b，我们需要推导逻辑回归的损失函数，然后对损失函数进行最小化，得到$\boldsymbol{w}$和b的估计值。给定训练数据集$\{(\boldsymbol{x}_i,y_i)\}_{i=1}^{m}$，将式中的$y$视作类后验概率估计$p(y=1|\boldsymbol{x})$，则逻辑回归模型的表达式可重写为:

$$\ln\frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b$$

三、实验原理

- 对上式展开：

$$p(y=1|\boldsymbol{x})=\frac{\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}x+b}}{1+\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}x+b}}=\hat{y}$$

$$p(y=0|\boldsymbol{x})=\frac{1}{1+\mathrm{e}^{\boldsymbol{w}^{\mathrm{T}}x+b}}=1-\hat{y}$$

- 综合两式：

$$p(y|\boldsymbol{x})=\hat{y}^{y}+(1-\hat{y})^{1-y}$$

- 两边取对数，改为求和式，并取负号： $-\ln p(y|\boldsymbol{x})=-\frac{1}{m}\sum_{i=1}^{m}[y\ln\hat{y}+(1-y)\ln(1-\hat{y})]$

  该式即为经典的交叉熵损失函数，其中$\hat{y}=\frac{1}{1+\mathrm{e}^{-(\boldsymbol{w}^{\mathrm{T}}x+b)}}$。

- 令$L=-\ln p(y|\boldsymbol{x})$，基于$L$分别对$\boldsymbol{w}$和$b$求偏导，有：

$$\frac{\partial L}{\partial \boldsymbol{w}}=\boldsymbol{x}(y-\hat{y})$$

$$\frac{\partial L}{\partial b}=y-\hat{y}$$

四、问题描述

- 某银行搜集了用户贷款、收入和信用卡是否逾期的信息。使用这些数据建立一个能预测信用卡逾期情况的逻辑回归模型。使用梯度下降法确定模型参数，并绘图显示损失函数的变化过程。
- 使用由credit-overdue.csv素材文件提供的数据集

1	debt	income	overdue
2	1.86	4.39	0
3	0.42	4.91	0
4	2.07	1.06	1
5	0.64	1.55	0
6	1.24	2.48	0
7	2.21	1.55	1
8	2.43	4.4	0
9	0.96	4.35	0
10	0.3	1.76	0
11	2.29	4.64	0
12	0.68	1.92	0
13	1.96	0.09	1
14	0.14	1.12	0
15	2.5	4.79	0
16	0.83	3.23	0
17	1.07	0.36	1
18	0.71	1.99	0
19	0.2	1.52	0
20	2.12	2.43	0
21	1.52	4.16	0
22	1.8	2.99	0
23	2.14	4.05	0
24	2.05	0.97	1
25	2.14	2.87	0
26	0.15	3.46	0
27	2.1	4.45	0
28	1.08	2.52	0
29	0.86	2.78	0
30	1.04	2.65	0

五、实验步骤

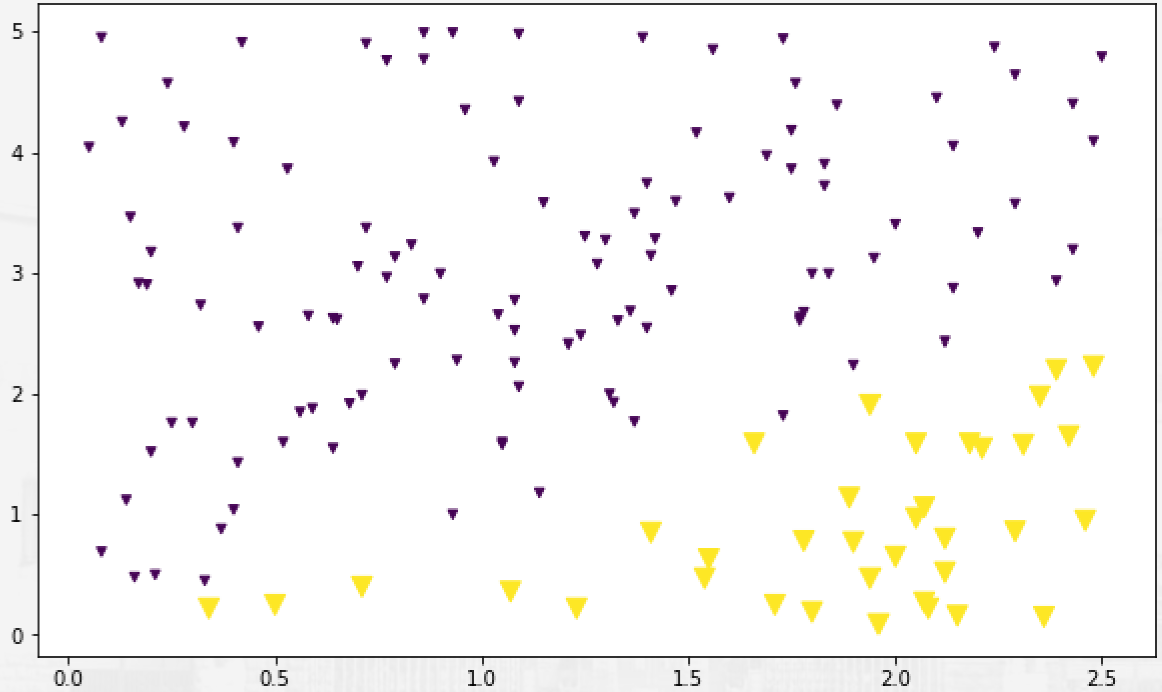
- 1、加载数据集

```python
import numpy as np
import pandas as pd
df = pd.read_csv("credit-overdue.csv", header=0) # 加载数据集
df.head()  #查看前5行数据
```

	debt	income	overdue
0	1.86	4.39	0
1	0.42	4.91	0
2	2.07	1.06	1
3	0.64	1.55	0
4	1.24	2.48	0

- 2、绘制数据的散点图，查看数据分布情况

```
from matplotlib import pyplot as plt
plt.figure(figsize=(10, 6))
map_size = {0: 20, 1: 100}
size = list(map(lambda x: map_size[x], df['overdue']))
plt.scatter(df['debt'],df['income'], s=size,c=df['overdue'],marker='v')
```



- 3、定义Sigmoid、损失函数，使用梯度下降确定模型参数

```python
#定义Sigmoid函数
def sigmoid(z):
    sigmoid = 1 / (1 + np.exp(-z))
    return sigmoid
#定义对数损失函数
def loss(h, y):
    loss = (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()
    return loss
#定义梯度下降函数
def gradient(X, h, y):
    gradient = np.dot(X.T, (h - y)) / y.shape[0]
    return gradient
```

- 3、定义Sigmoid、损失函数，使用梯度下降确定模型参数

```
# 逻辑回归过程
def Logistic_Regression(x, y, lr, num_iter):
    intercept = np.ones((x.shape[0], 1))  # 初始化截距为 1
    x = np.concatenate((intercept, x), axis=1)
    w = np.zeros(x.shape[1])  # 初始化参数为 0

    for i in range(num_iter):  # 梯度下降迭代
        z = np.dot(x, w)     # 线性函数
        h = sigmoid(z)      # sigmoid 函数
        g = gradient(x, h, y)  # 计算梯度
        w -= lr * g     # 通过学习率 lr 计算步长并执行梯度下降
        z = np.dot(x, w)     # 更新参数到原线性函数中
        h = sigmoid(z)      # 计算 sigmoid 函数值
        l = loss(h, y)        # 计算损失函数值
    return l, w         # 返回迭代后的梯度和参数
```

- 4、初始化模型，并对模型进行训练

```
x = df[['debt','income']].values
y = df['overdue'].values
lr = 0.001                    # 学习率
num_iter = 10000              # 迭代次数
# 模型训练
L = Logistic_Regression(x, y, lr, num_iter)
print("逻辑回归模型为: \n", L)
```

```
逻辑回归模型为:
 (0.19383368371859122, array([ 0.05603937,  0.9925221 , -1.3325938 ]))
```

- 5、根据得到的参数，绘制模型分类线

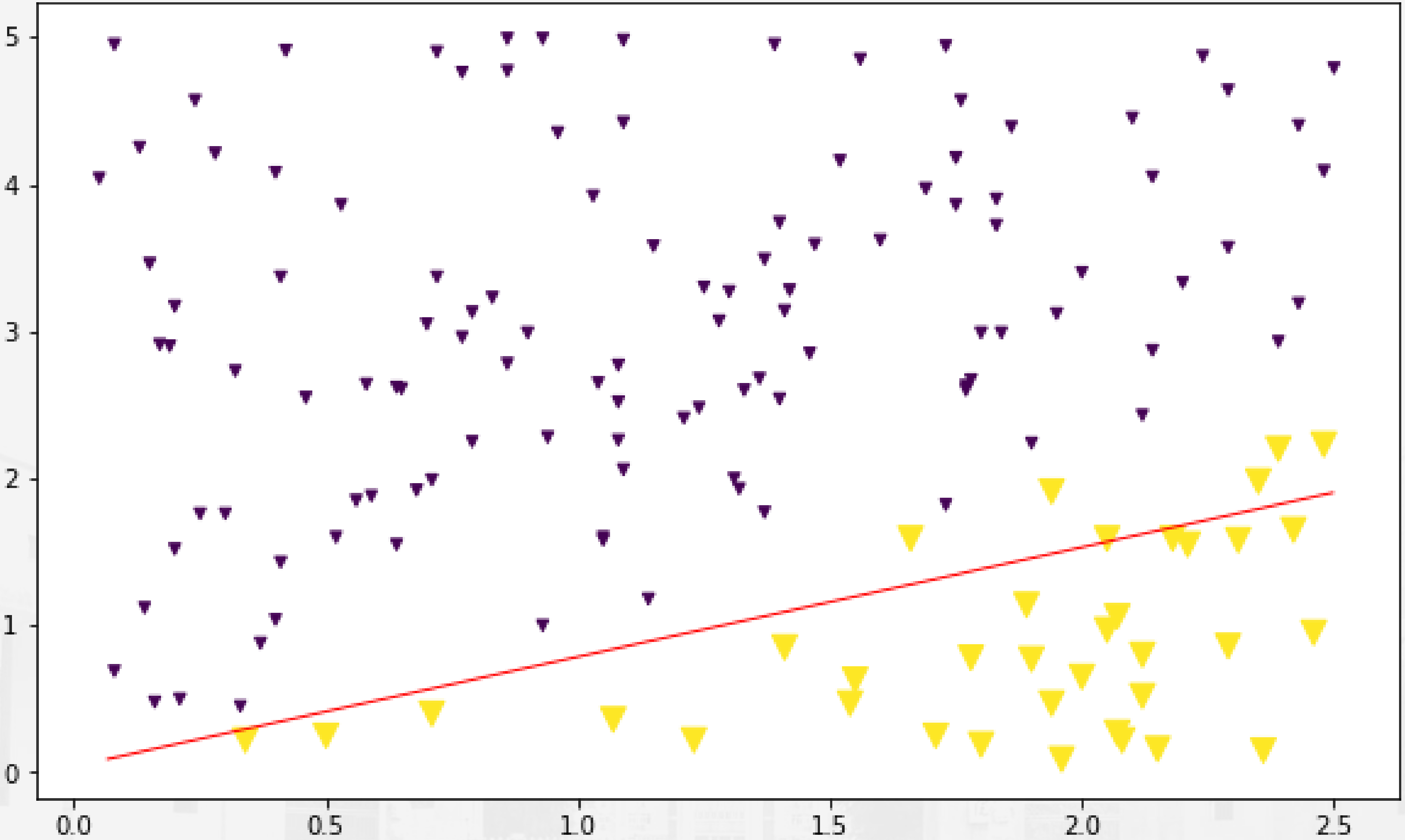
```
plt.figure(figsize=(10, 6))
map_size = {0: 20, 1: 100}
size = list(map(lambda x: map_size[x], df['overdue']))
plt.scatter(df['debt'],df['income'], s=size,c=df['overdue'],marker='v')

x1_min, x1_max = df['debt'].min(), df['debt'].max(),
x2_min, x2_max = df['income'].min(), df['income'].max(),

xx1, xx2 = np.meshgrid(np.linspace(x1_min, x1_max), np.linspace(x2_min, x2_max))
grid = np.c_[xx1.ravel(), xx2.ravel()]

probs = (np.dot(grid, np.array([L[1][1:3]]).T) + L[1][0]).reshape(xx1.shape)
plt.contour(xx1, xx2, probs, levels=[0], linewidths=1, colors='red');
```

五、实验步骤

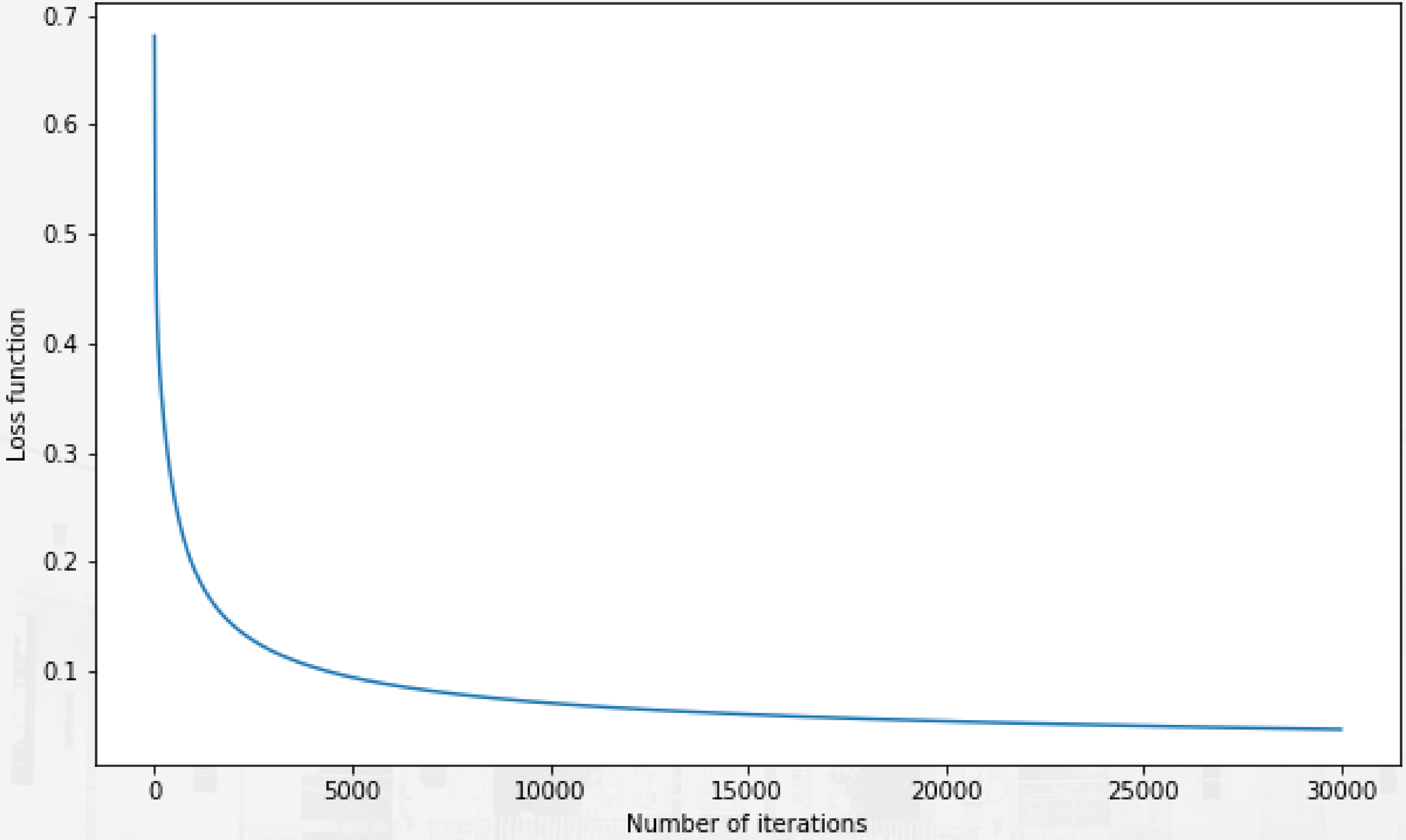
- 6、绘制损失函数变化曲线

```python
def Logistic_Regression(x, y, lr, num_iter):
    intercept = np.ones((x.shape[0], 1))  # 初始化截距为 1
    x = np.concatenate((intercept, x), axis=1)
    w = np.zeros(x.shape[1])          # 初始化参数为 1

    l_list = []                 # 保存损失函数值
    for i in range(num_iter):    # 梯度下降迭代
        z = np.dot(x, w)       # 线性函数
        h = sigmoid(z)       # sigmoid 函数

        g = gradient(x, h, y)    # 计算梯度
        w -= lr * g      # 通过学习率 lr 计算步长并执行梯度下降

        z = np.dot(x, w)  # 更新参数到原线性函数中
        h = sigmoid(z)  # 计算 sigmoid 函数值

        l = loss(h, y)    # 计算损失函数值
        l_list.append(l)
    return l_list

lr = 0.01         # 学习率
num_iter = 30000  # 迭代次数
l_y = Logistic_Regression(x, y, lr, num_iter)  # 训练

# 绘图
plt.figure(figsize=(10, 6))
plt.plot([i for i in range(len(l_y))], l_y)
plt.xlabel("Number of iterations")
plt.ylabel("Loss function")
```

五、实验步骤



五、实验步骤


0.7
0.6
0.5
0.4
0.3
0.2
0.1
Loss function
0
5000
10000
15000
20000
25000
30000
Number of iterations

六、实验报告要求

- 1、实验目的
- 2、实验内容
- 3、实验原理
- 4、实验代码
- 5、运行结果与分析
- 6、实验小结

- 说明：每个学生都要交电子版的实验报告，命名格式:
- 01/02-XXXX（学号）-XXX（姓名）

黄冈师范学院
HUANGGANG NORMAL UNIVERSITY

Q&A

>>>>>>>>>>>>>>>>>> <<<<<<<<<<<<<<<<<<

厚德 博学 力行 致远