模式识别与机器学习实验一——Logistic回归

2021113117-王宇轩

1. 实验环境

- 操作系统: Windows
- 编程语言：Python
- IDE：PyCharm

2. 文件列表

文件名	内容
logistic.py	自编Logistic回归调用库
linear.py	线性问题求解程序
nonlinear.py	非线性问题求解程序
实验报告.pdf	实验报告

3. 实验过程

2.1 logistic.py

编写sigmoid函数:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

编写损失函数值计算函数，用于显示迭代次数epoch-损失函数变化图像:

```
def compute_cost(X, y, theta, L2, alpha):
    m = len(y)
    h = sigmoid(X @ theta)
    epsilon = 1e-5
    cost = (1 / m) * np.sum(-y * np.log(h + epsilon) - (1 - y) * np.log(1 - h +
epsilon))
    if L2:
        cost += (alpha / (2*m)) * np.sum(theta**2)
    return cost
```

随后，在logistic.py中定义了Model类，即logistic回归模型：

```
class Model:
    def __init__(self, data, lr, epoch, L2=False, alpha=0):
        self.lr = lr
        self.epoch = epoch
        self.data = data
        self.X = data.iloc[:, :-1].values  # 提取特征列（除了最后一列）
        self.y = data.iloc[:, -1].values  # 提取目标变量列（最后一列）
        self.X, self.mean, self.std = self.processData()
        self.theta = np.zeros((len(self.X[1]))).T
        self.L2 = L2
        self.alpha = alpha
```

其中，输入对象`data`为训练数据，可用pandas库的DataFrame结构；`lr`为学习率η，即更新θ的步长；`epoch`为迭代次数；`L2`为True则启用L2正则化。`alpha`为输入的L2正则化超参数λ。

在`processData()`方法中，进行了数据的特征规范化，并对X进行了增广，如下。

```
    # 处理数据：特征规范化、增广
    def processData(self):
        mean = np.mean(self.X, axis=0)  # 沿着第0维度（行）计算平均值
        std = np.std(self.X, axis=0)
        self.X = (self.X - mean) / std
        X = np.insert(self.X, 0, 1, axis=1)
        return X, mean, std
```

梯度下降方法如下。更新参数θ的规则：

$$\theta^{k+1} = \theta^k - \eta \nabla J(\theta^k)$$

其中：

$$\nabla J(\theta) = \frac{1}{m} X'(g(X\theta) - Y)$$

```
    # 梯度下降
    def gradient_descent(self):
        m = len(self.y)
        X = self.X
        y = self.y
        lr = self.lr
        alpha = self.alpha
        cost_history = []
        for i in range(self.epoch):
            h = sigmoid(X @ self.theta)
            if self.L2:
```

```
            self.theta -= (lr / m) * ((X.T @ (h - y)) - (alpha * self.theta))
        else:
            self.theta -= (lr / m) * (X.T @ (h - y))
        cost = compute_cost(X, y, self.theta, self.L2, self.alpha)
        cost_history.append(cost)
    # 输出最终损失函数值
    print(f'Final Loss:{cost_history[-1]}')
    # 显示损失函数图像
    plt.plot(range(self.epoch), cost_history)
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.show()
```

下面进行一下由损失函数

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}log(h_\theta(x^{(i)})+(1-y^{(i)})log(1-h_\theta(x^{(i)}))]$$

到梯度$\nabla J(\theta)$的推导。首先求偏导数$\frac{\partial J(\theta)}{\partial \theta_j}$，应用链式求导法则：

$$\frac{\partial J(\theta)}{\partial \theta_j}=\frac{\partial J(\theta)}{\partial h_\theta(x)}\cdot\frac{\partial h_\theta(x)}{\partial \theta_j} \tag{1}$$

其中

$$h_\theta(x)=g(\theta'x)=\frac{1}{1+exp(-\theta'x)}$$

偏导数

$$\frac{\partial h_\theta(x)}{\partial \theta_j}=h_\theta(x)(1-h_\theta(x))x_j \tag{2}$$

而

$$\frac{\partial J(\theta)}{\partial h_\theta(x)}=-\frac{1}{m}\sum_{i=1}^{m}\left[\frac{y^{(i)}}{h_\theta(x^{(i)})}-\frac{1-y^{(i)}}{1-h_\theta(x^{(i)})}\right] \tag{3}$$

将(2)和(3)代入(1)，化简可得

$$\frac{\partial J(\theta)}{\partial \theta_j}=(h_\theta(x)-y)x_j$$

即

$$\nabla J(\theta)=\frac{1}{m}X'(g(X\theta)-Y)。$$

接下来，由于数据进行了特征规范化，需要对θ进行对应的恢复。因此定义方法 `retheta()` 如下：

```
# 恢复因为特征规范化影响的theta
def retheta(self):
    self.theta[1:] = self.theta[1:] / self.std
    self.theta[0] -= np.dot(self.mean, self.theta[1:])
```

最主要地，用方法 `solva()` 进行该logistic回归模型的求解：

```
# 求解优化值
def solve(self):
    self.gradient_descent()
    self.retheta()
    return self.theta
```

最后，可调用 `show()` 显示原始数据和分界面图像：

```
# 显示分类界面图像
def show(self):
    data = self.data
    theta = self.theta
    data_0 = data[data['y'] == 0]
    data_1 = data[data['y'] == 1]
    # 数据图
    plt.scatter(data_0['x0'], data_0['x1'], c='#1f77b4')  # blue
    plt.scatter(data_1['x0'], data_1['x1'], c='#ff7f0e')  # orange
    plt.legend(['y = 0', 'y = 1'])
    # 获取图像显示范围
    x_min, x_max = plt.xlim()
    y_min, y_max = plt.ylim()
    # 画出模型方程值为0时等高线——分界线
    x = np.linspace(x_min, x_max, 100)
    y = np.linspace(y_min, y_max, 100)
    X, Y = np.meshgrid(x, y)
    Z = theta[0] * np.ones_like(X)
    # 由数据特征值名称恢复模型方程
    variables = {'x0': X, 'x1': Y}
    headers = data.columns.tolist()
    for i in range(len(headers)-1):
        Z += theta[i+1] * eval(headers[i], variables)
    plt.contour(X, Y, Z, levels=[0])
    plt.show()
```

在这里，分界面方程中的特征由是data中的列名称进行建立的，在**2.3 nonlinear.py**中给出了一个例子。对于分界面的绘制调用了等高线绘制方法`plt.contour()`，来绘制Z=0处的等高线，即为分界线方程$h_\theta(x)=0$。

2.2 linear.py

内容如下：

```
import logistic
import os
import pandas as pd
# 获取数据
data_file = os.path.join('.', 'logistic_data1.csv')
data = pd.read_csv(data_file, )
# 分析
re = logistic.Model(data=data, lr=0.001, epoch=10000)
theta = re.solve()
print(f'Theta:{theta}')
re.show()
```

导入库logistic，从文件logistic_data1.csv中读取数据后，即可调用`re = logistic.Model(data=data, lr=0.001, epoch=10000)`来建立一个logistic回归模型`re`，并指定输入数据和超参数；执行`re.solve()`即可求解最优指，在这里赋给变量`theta`并打印输出，最后可调用`re.show()`显示结果。

2.3 nonlinear.py

对于非线性logistic回归问题，需要构造高维特征，此处选择构造$x=(1,x_0,x_1,x_0^2,x_1^2,x_0x_1,x_0^3,x_1^4)$，代码如下：

```
# 获取数据
data_file = os.path.join('.', 'logistic_data2.csv')
data = pd.read_csv(data_file, )
# 增加非线性特征值
y = data['y']  # 取出y列
del data['y']

data['x0**2'] = data['x0'] ** 2
data['x1**2'] = data['x1'] ** 2
data['x0*x1'] = data['x0'] * data['x1']
data['x0**3'] = data['x0'] ** 3
data['x1**4'] = data['x1'] ** 4
data['y'] = y  # 恢复y列，使其保持为最后一列
```

需要注意，**这里data的列名是需要符合运算表达式的**，如`data['x0**2']`即代表此列特征为x_0^2，因为在`show()`方法中需要用这些列名来获取分界面方程，见**2.1 logistic.py**，此处绘制的分界面即$h_\theta(x)=\theta x=0$。

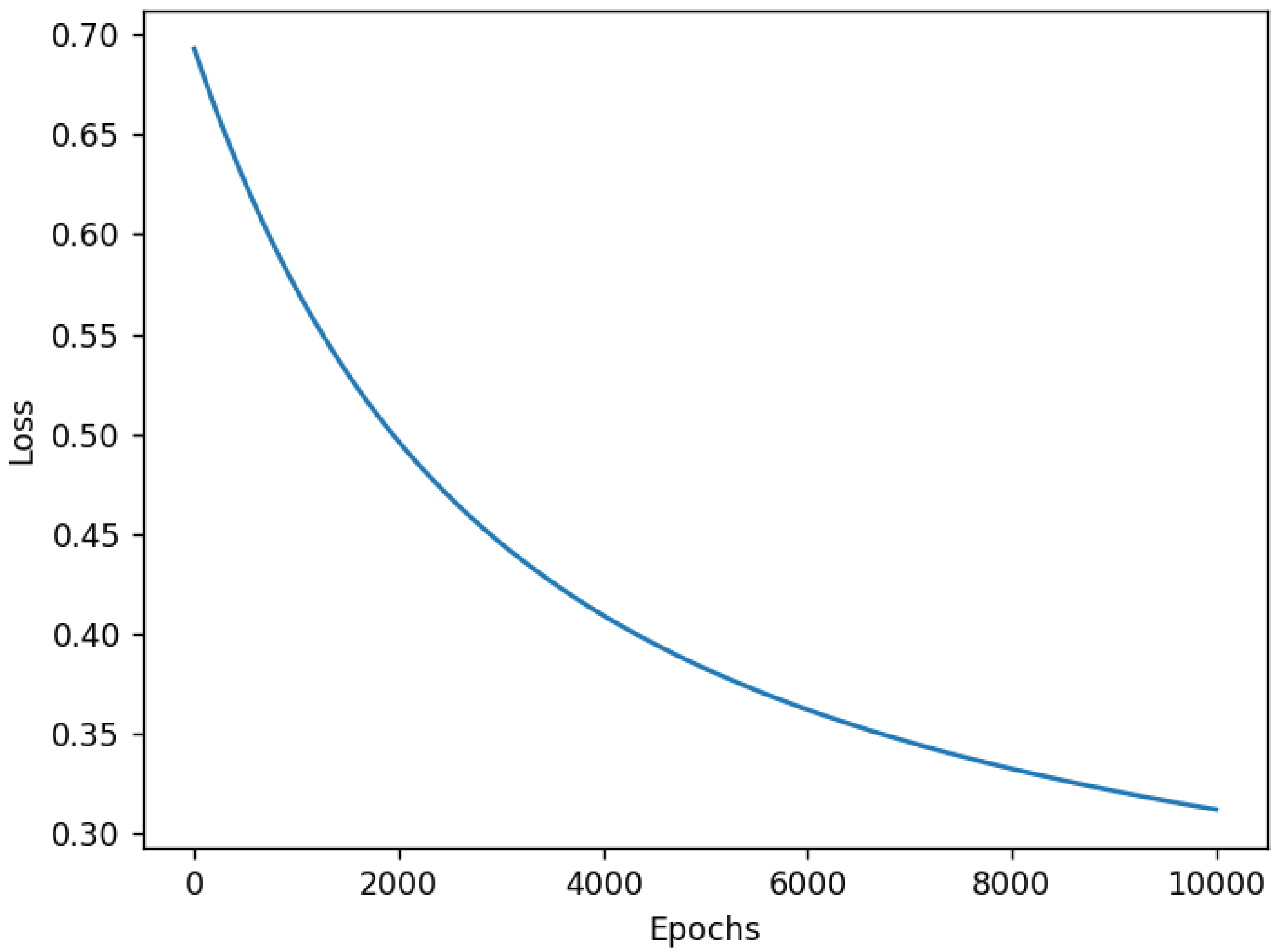
之后便可用与线性问题相似的方法调用解决，如下。

```
# 分析
re = logistic.Model(data=data, lr=0.01, epoch=10000, L2=True, alpha=0.1)
theta = re.solve()
print(f'Theta:{theta}')
re.show()
```

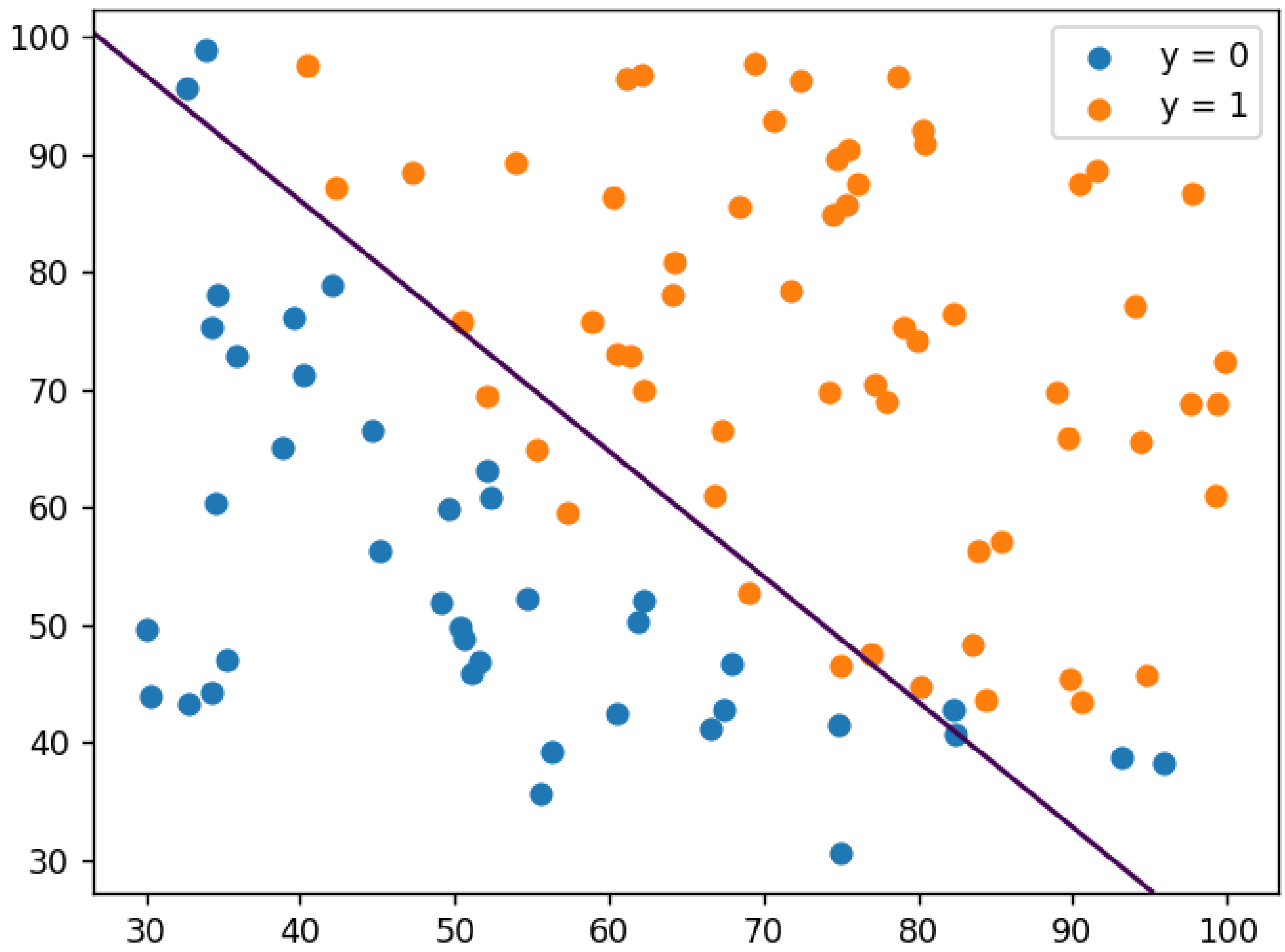
4. 实验结果

3.1 线性问题

损失函数图像：



结果：

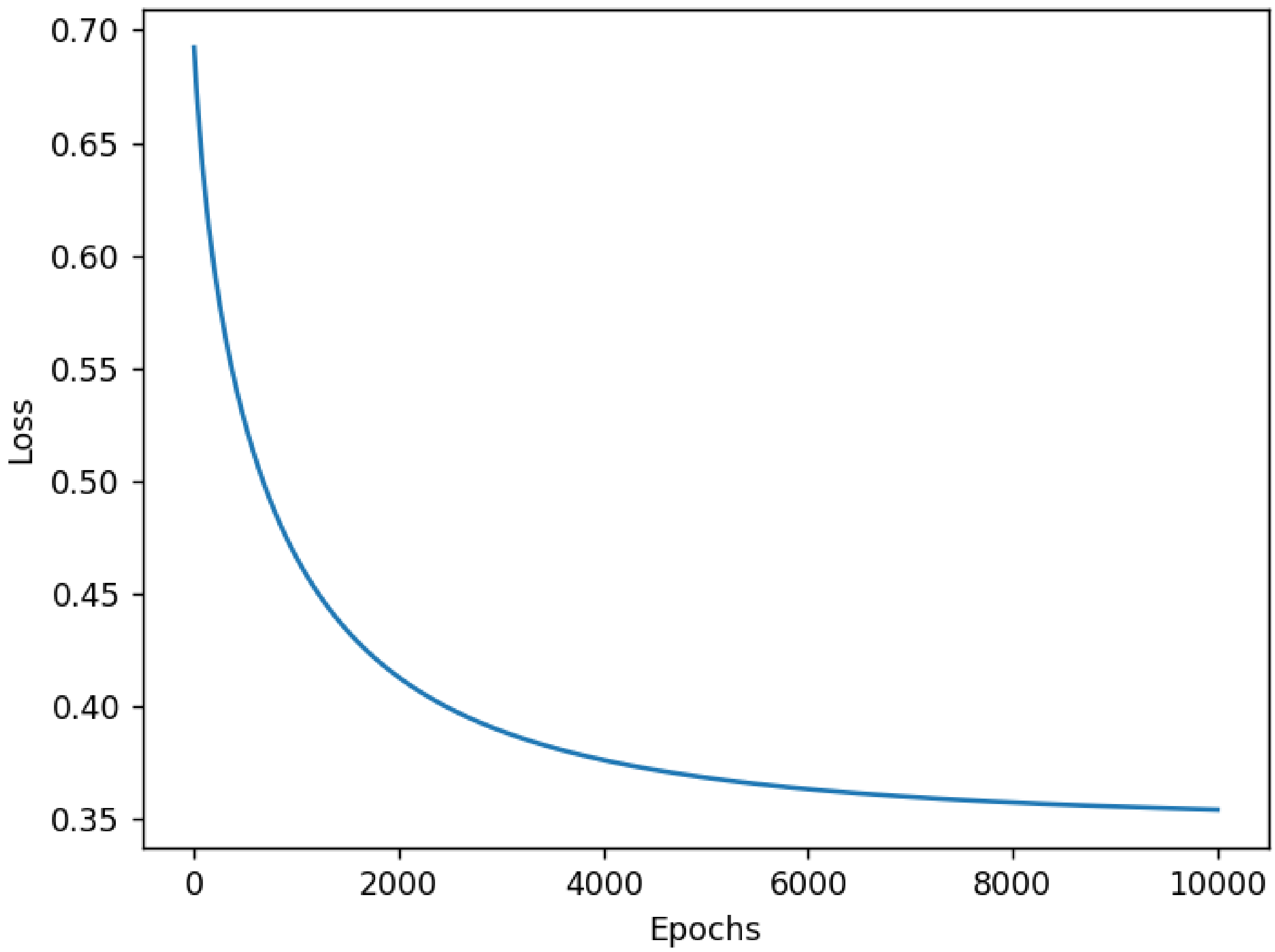


输出：

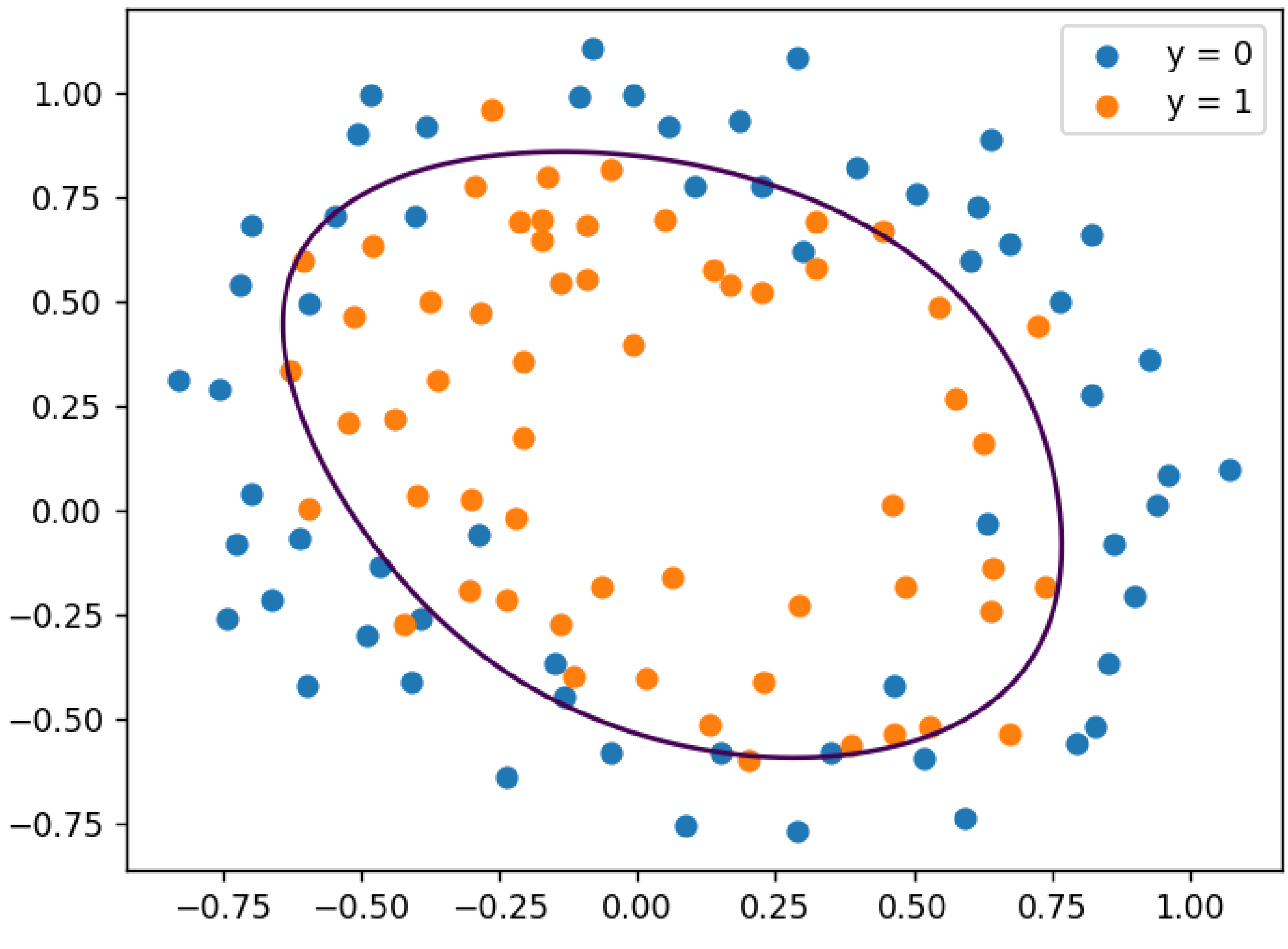
```
Final Loss:0.3118439379837738
Theta:[-8.02126764  0.06640911  0.06235306]
```

3.2 非线性问题

若不启用L2正则化结果如下：



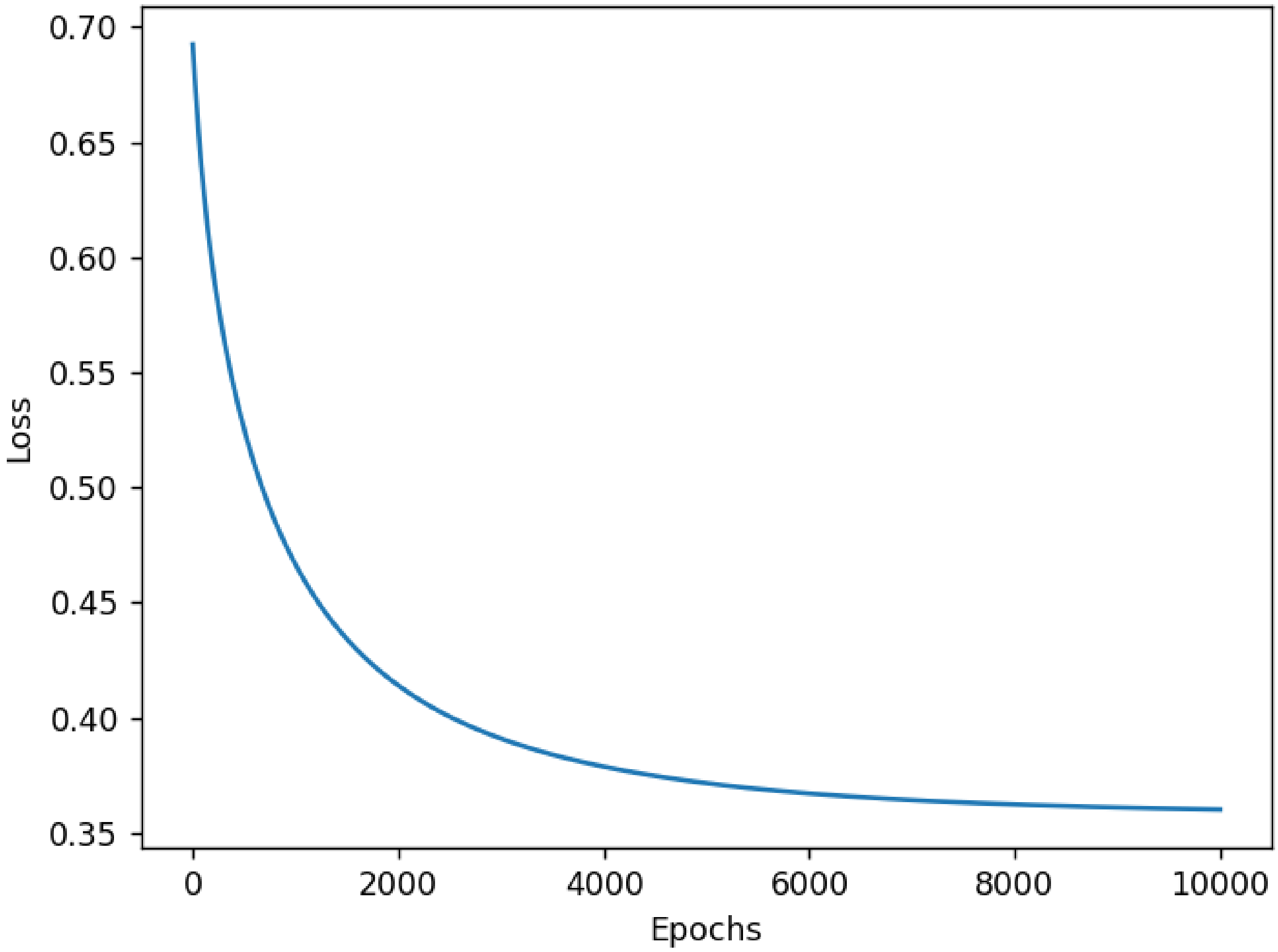
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
Loss
0
2000
4000
6000
8000
10000
Epochs

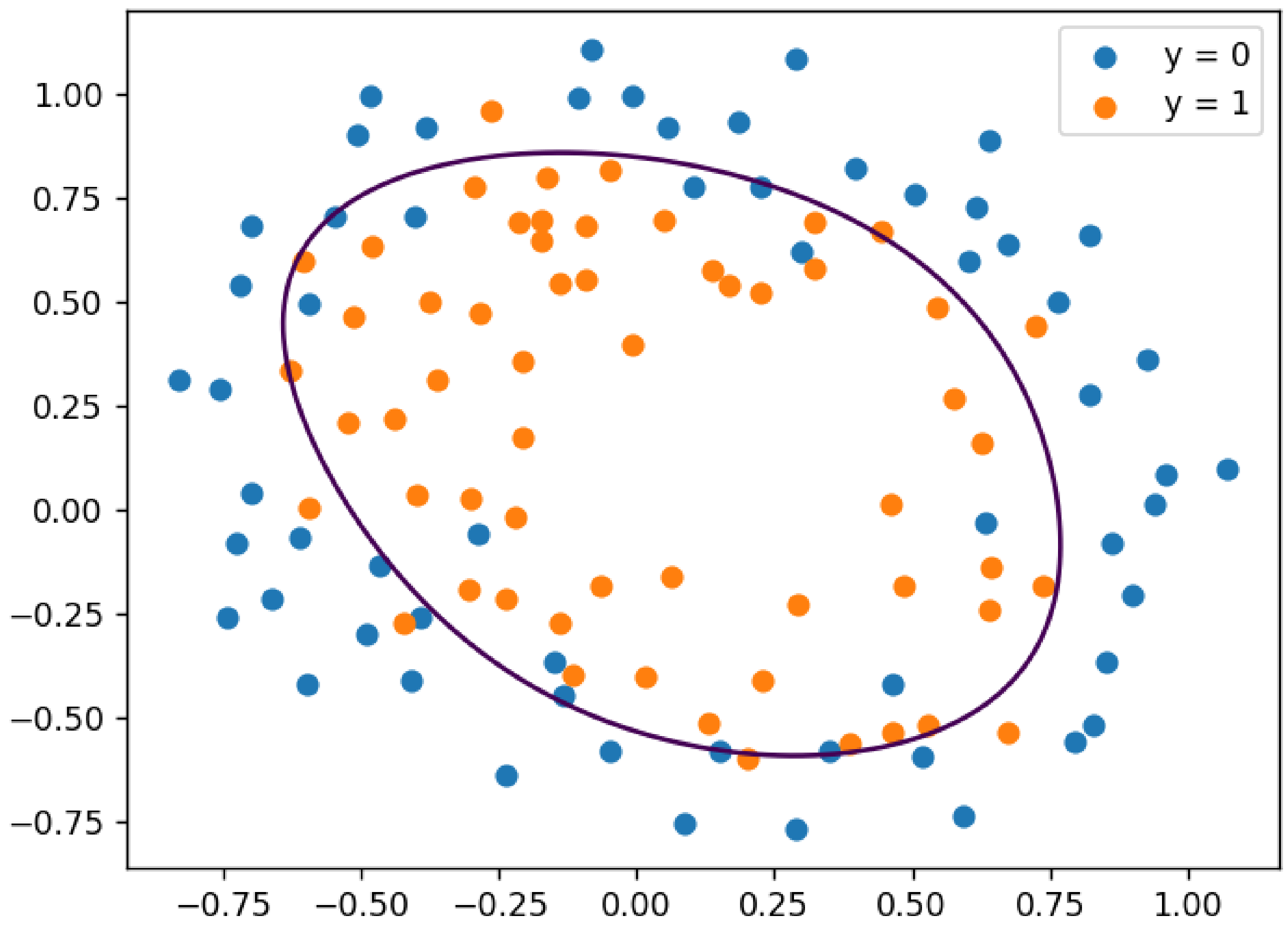


y = 0
y = 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

```
Final Loss:0.35434935193808414
Theta:[ 3.8313026   2.07943207  3.27292889 -9.78686064 -5.87137443 -5.56719969
  0.65759428 -4.54487757]
```

若启用L2正则化，结果如下。






```
Final Loss:0.3602510946605831
Theta:[  3.93834515   2.16025611   3.39352428 -10.07566417  -6.06472719
  -5.77829245   0.69544308  -4.6803373 ]
```

因为本问题中过拟合现象并不明显，L2正则化效果体现较弱。如果将参数alpha调大则会出现欠拟合现象，导致结果更差。