实验五：神经网络

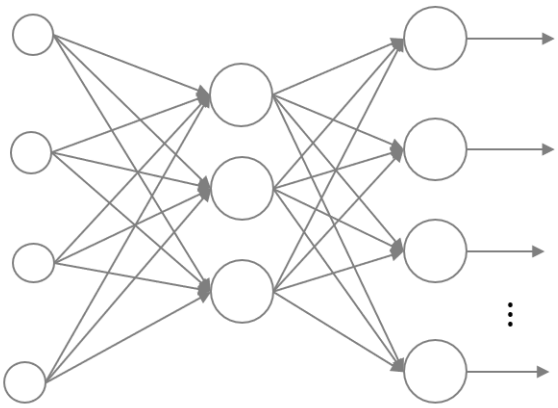
姓名：　　学号：

- 实验目的

理解神经网络原理与计算框架，包括前馈神经网络、激活函数、损失函数、后向传播过程等，学会使用梯度下降法对神经网络进行训练，学会分析不同学习率对梯度下降法收敛性影响。

- 实验要求

给定手写数字数据集，采用如下全连接神经网络进行分类。输入层 784 个节点，隐层 12 个节点，输出层 10 个节点，隐层和输出层均采用 sigmoid 激活函数，损失函数为平方损失函数。采用标准正态分布初始化权重和阈值参数，随机梯度下降最大轮次设置为 100，对比学习率为 0.001，0.005，0.01 时模型的损失函数迭代曲线和模型在测试集上的精度(accuracy)。



- 实验环境

Python、numpy、pandas、matplotlib

- 实验代码

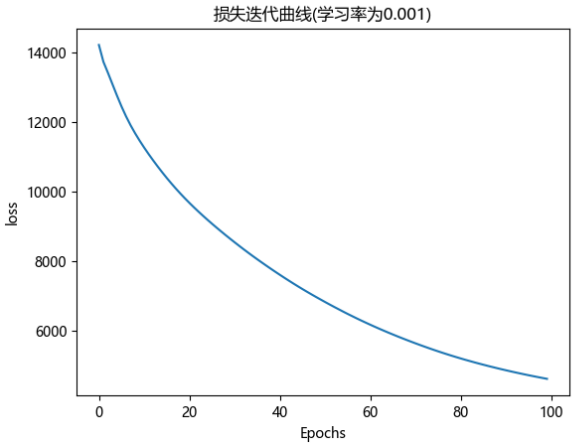
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# read data from files
train_data = pd.read_csv('experiment_05_training_set.csv')
test_data = pd.read_csv('experiment_05_testing_set.csv')
train_data = np.array(train_data)
test_data = np.array(test_data)
train_x = train_data[:, 1:] / 255
train_y = train_data[:, 0]
test_x = test_data[:, 1:] / 255
```

```
test_y = test_data[:, 0]
# one-hot encoding
n = train_x.shape[0]
one_hot_h = np.zeros((n, 10))
one_hot_h[np.arange(n), train_y.reshape((n,))] = 1
# initialization
theta = np.random.randn(10, 1)
V = np.random.randn(10, 12)
b = np.random.randn(12, 1)
W = np.random.randn(12, 784)

alpha = 0.001


# sigmoid 函数
def sigmoid(z_sigmoid):
    return 1 / (1 + np.exp(-z_sigmoid))


line = np.linspace(0, 29999, 30000)   #  创造索引数组
loss_array = []
for j in range(100):
    for i in range(train_x.shape[0]):
        np.random.shuffle(line)     #随机打乱索引数组
        x = train_x[i, :]
        x = x.reshape(-1, 1)
        h = one_hot_h[i, :]
        h = h.reshape(-1, 1)
        z = np.dot(W, x) + b
        a = sigmoid(z)
        t = np.dot(V, a) + theta
        y = sigmoid(t)
        L = 1 / 2 * np.square(y - h)
        L_theta = (y - h) * y * (1 - y)
        L_V = np.dot(L_theta, a.T)
        L_b = np.dot(V.T, L_theta) * a * (1 - a)
        L_W = np.dot(L_b, x.T)
        theta = theta - alpha * L_theta
        V = V - alpha * L_V
        b = b - alpha * L_b
        W = W - alpha * L_W
    Loss = 0
    for i in range(train_x.shape[0]):
        # for i in range(10):
        x = train_x[i, :]
        x = x.reshape(-1, 1)
        h = one_hot_h[i, :]
```

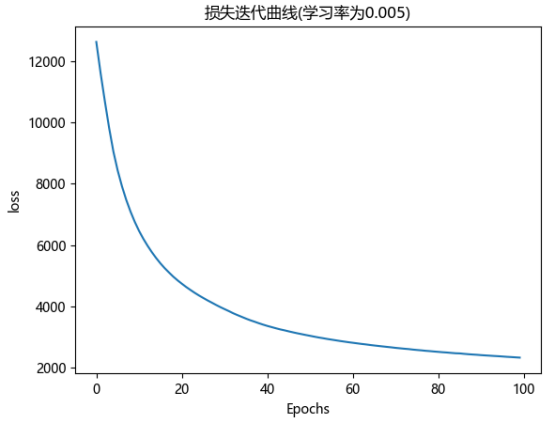
```
            h = h.reshape(-1, 1)
            z = np.dot(W, x) + b
            a = sigmoid(z)
            t = np.dot(V, a) + theta
            y = sigmoid(t)
            L = 1 / 2 * np.square(y - h)
            loss = np.sum(L)
            Loss += loss
        loss_array.append(Loss)
plt.plot(loss_array)
plt.xlabel("Epochs")
plt.ylabel("loss")
plt.title("损失迭代曲线(学习率为 0.001)")
plt.rcParams['font.sans-serif'] = ['Microsoft YaHei']
plt.show()

acc = 0

for i in range(test_data.shape[0]):
    x = test_x[i, :]
    x = x.reshape(-1, 1)
    h = test_y[i]
    z = np.dot(W, x) + b
    a = sigmoid(z)
    t = np.dot(V, a) + theta
    y = sigmoid(t)
    if np.argmax(y) == h:
        acc = acc + 1
print('Test Accuracy', acc / test_data.shape[0])
```

- 结果分析

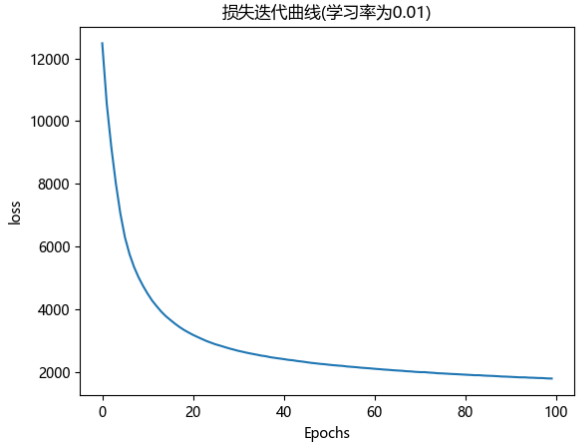
学习率为 0.001 时损失迭代曲线：



学习率为 0.005 时损失迭代曲线：



学习率为 0.01 时损失迭代曲线：



学习率	0.001	0.005	0.01
精度	0.8045	0.89375	0.90375