神经网络

> 2023春机器学习 小作业2
>
> 201870139 赵耀
>
> 2023/3/21

一、作业要求

任务描述

- 用python语言，实现神经网络学习，完成"手写体识别"任务。建议：使用图像预处理技术（去噪，归一化，分割等），再使用CNN进行特征提取。

数据集

- MNIST数据集

二、算法原理

CNN：卷积神经网络（Convolutional Neural Networks）是一类包含卷积计算且具有深度结构的前馈神经网络（Feedforward Neural Networks），是深度学习（deep learning）的代表算法之一。

卷积神经网络的各层中的神经元是3维排列的：**宽度、高度和深度**。其中的宽度和高度是很好理解的，因为本身卷积就是一个二维模板，但是在卷积神经网络中的深度指的是
激活数据体的第三个维度，而不是整个网络的深度，整个网络的深度指的是网络的层数。举个例子来理解什么是宽度，高度和深度，假如使用CIFAR-10中的图像是作为卷积神经网络的输入，该
输入数据体的维度是32x32x3（宽度，高度和深度）。**
我们将看到，层中的神经元将只与前一层中的一小块区域连接，而不是采取全连接方式。**
对于用来分类CIFAR-10中的图像的卷积网络，其最后的输出层的维度是1x1x10，因为在卷积神经网络结构的最后部分将会把全尺寸的图像压缩为包含分类评分的一个向量，
向量是在深度方向排列的。

三、运行环境

- IDE: Pycharm for windows
- Python: Python 3.10.10
- Dependencies: h5qy, numpy, os, random, tensorflow

四、过程说明

获取训练集

```
mnist = tf.keras.datasets.mnist
(train_x, train_y), (test_x, test_y) = mnist.load_data()
train_x, test_x = train_x / 255, test_x / 255
```

数据预处理、归一化

```
X_train, X_test = tf.cast(train_x, tf.float32), tf.cast(test_x,
tf.float32)
Y_train, Y_test = tf.cast(train_y, tf.float32), tf.cast(test_y,
tf.float32)
```

参数设置

```
# 训练次数
epoch = 50
"""
学习方式
值为1,表示每lr_for_epochs轮固定按lr_change_rate比例更新学习率
值为2,表示记录5次学习率大小,当前轮次loss值大于前nub次（包括本次）loss平均值时,
学习率自动降为当前学习率0.1倍,当学习率降为last_lr时，训练终止，保存模型
"""
train_method = 2  # method 1 or 2

# train_method = 1 所需的参数
# 默认学习率
```

```
learn_rate = 0.01
# 初始学习率
init_learn_rate = 0.01
# 每10轮更新一次学习率
lr_for_epochs = 10
# 更新学习率的比例
lr_change_rate = 0.5

# train_method = 2的参数
# 截止学习率值
last_lr = 0.0001

# 输入层神经网络节点数=28*28
width_input = 784
# 第一层神经网络节点数
width_net1 = 100
# 第二层神经网络节点数
width_net2 = 100
# 输出层神经网络节点数
width_net3 = 10
```

训练过程

```
# 训练过程
for n in range(0, epoch + 1):
    # 如果是method1, 改变学习率
    if train_method == 1:
        learn_rate = init_learn_rate * lr_change_rate ** (int(n /
lr_for_epochs))  # 学习率随着学习轮数指数递减

    # 打乱样本
    r = np.random.permutation(60000)
    train_x = train_x[r, :, :]
    train_y = train_y[r]

    for i in range(0, 60000):
        x = np.array(train_x[i])
        x = x.reshape(width_input, )
        z1 = np.dot(x, w1) + b1
        a1 = feedforward(x, w1, b1)
        z2 = np.dot(a1, w2) + b2
        a2 = feedforward(a1, w2, b2)
        z3 = np.dot(a2, w3) + b3
        # y=softmax(z3)
        y = feedforward(a2, w3, b3)
        y_t = np.zeros((width_net3,))
        y_t[train_y[i]] = 1
```

```
        eta3 = (-y_t / y + (1 - y_t) / (1 - y)) * sigmoid(z3) * (1 -
sigmoid(z3))  # 此为反向传播过程中中间参数，下同
        # eta3=2*(y-y_t)*sigmoid(z3)*(1-sigmoid(z3))#此为反向传播过程中中间
参数，下同
        eta2 = np.dot(eta3, np.transpose(w3)) * sigmoid(z2) * (1 -
sigmoid(z2))
        eta1 = np.dot(eta2, np.transpose(w2)) * sigmoid(z1) * (1 -
sigmoid(z1))
        b3 = b3 - learn_rate * eta3
        b2 = b2 - learn_rate * eta2
        b1 = b1 - learn_rate * eta1
        w3 = w3 - learn_rate * np.dot(a2.reshape(width_net2, 1),
eta3.reshape(1, width_net3))
        w2 = w2 - learn_rate * np.dot(a1.reshape(width_net1, 1),
eta2.reshape(1, width_net2))
        w1 = w1 - learn_rate * np.dot(x.reshape(width_input, 1),
eta1.reshape(1, width_net1))

    loss1 = 0
    True_num = 0

    # 如果是method2，加载测试集，计算loss和precision
    for i in range(0, 10000):
        x = np.array(test_x[i])
        x = x.reshape(1, width_input)
        y_t = np.zeros((width_net3,))
        y_t[test_y[i]] = 1
        a1 = feedforward(x, w1, b1)
        a2 = feedforward(a1, w2, b2)

        y = feedforward(a2, w3, b3)
        if test_y[i] == np.argmax(y, axis=1):
            True_num = True_num + 1
        loss1 = loss1 + cross_entropy_loss(y, y_t)

    precision = True_num / 10000 * 100

    # 改变学习率，利用队列方式记录连续nub次loss值
    if train_method == 2:
        # 临时存储模型
        j = range(1, nub)
        k = range(0, nub - 1)
        w11[j] = w11[k]
        b11[j] = b11[k]
        w21[j] = w21[k]
        b21[j] = b21[k]
        w31[j] = w31[k]
```

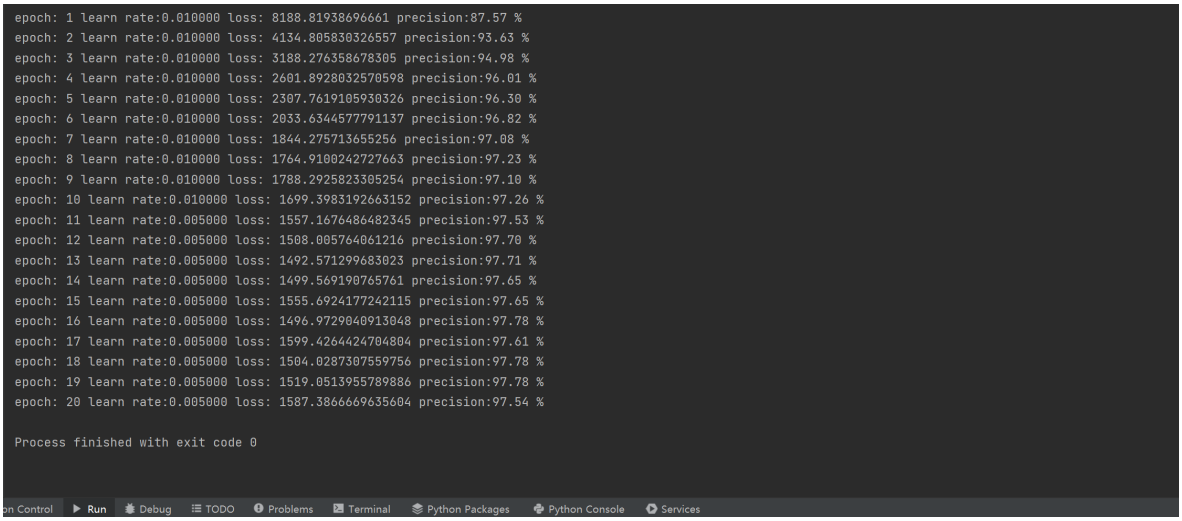
```
            w11[0] = w1
            b11[0] = b1
            w21[0] = w2
            b21[0] = b2
            w31[0] = w3
            b31[0] = b3
            loss2[j] = loss2[k]
            loss2[0] = loss1

            # 判断是否改变学习率
            if loss2[0] > np.mean(loss2) and loss2[nub - 1] > 0:
                learn_rate = learn_rate * 0.1
                if learn_rate < last_lr:
                    save_model(w11[np.argmin(loss2)], w21[np.argmin(loss2)],
w31[np.argmin(loss2)],
                               b11[np.argmin(loss2)], b21[np.argmin(loss2)],
b31[np.argmin(loss2)])
                    print("epoch:", n + 1, "lr:%.6f" % learn_rate, "loss:",
loss1, 'precision:%.2f' % precision, '%')
                    break

        if n % lr_for_epochs == 0:
            save_model(w1, w2, w3, b1, b2, b3)

        # 输出训练结果
        print("epoch:", n + 1, "learn rate:%.6f" % learn_rate, "loss:", loss1,
'precision:%.2f' % precision, '%')
```

五、样例截图

注意

对于训练过程中的输出，用shell命令`python neural-network.py > 1.txt && sort -u -k 7 -t ' ' 1.txt`可以查看最高准确率。

训练次数**epoch=20**，学习方式**train_method=1**，最高准确率为**97.78%**



H3 训练次数**epoch=50**，学习方式**train_method=1**，最高准确率为**98.06%**

```
epoch: 1 learn rate:0.010000 loss: 8692.531559560286 precision:86.61 %
epoch: 2 learn rate:0.010000 loss: 4229.678848592091 precision:93.28 %
epoch: 3 learn rate:0.010000 loss: 3295.520185228211 precision:95.04 %
epoch: 4 learn rate:0.010000 loss: 2554.4296087615835 precision:96.06 %
epoch: 5 learn rate:0.010000 loss: 2201.9336054823457 precision:96.60 %
epoch: 6 learn rate:0.010000 loss: 2064.635852994141 precision:96.76 %
epoch: 7 learn rate:0.010000 loss: 1968.0863814734048 precision:96.96 %
epoch: 8 learn rate:0.010000 loss: 1711.552931547748 precision:97.26 %
epoch: 9 learn rate:0.010000 loss: 1698.3136878473326 precision:97.40 %
epoch: 10 learn rate:0.010000 loss: 1536.068470723648 precision:97.63 %
epoch: 11 learn rate:0.005000 loss: 1451.232774742652 precision:97.67 %
epoch: 12 learn rate:0.005000 loss: 1511.9340014288105 precision:97.77 %
epoch: 13 learn rate:0.005000 loss: 1433.5606569838637 precision:97.80 %
epoch: 14 learn rate:0.005000 loss: 1427.8854699139413 precision:97.89 %
epoch: 15 learn rate:0.005000 loss: 1389.5408051316458 precision:97.78 %
epoch: 16 learn rate:0.005000 loss: 1444.3235162654196 precision:97.77 %
epoch: 17 learn rate:0.005000 loss: 1451.148572226583 precision:97.80 %
epoch: 18 learn rate:0.005000 loss: 1384.3403489354255 precision:97.93 %
epoch: 19 learn rate:0.005000 loss: 1417.299091250715 precision:98.00 %
epoch: 20 learn rate:0.005000 loss: 1415.1851282210205 precision:97.96 %
epoch: 21 learn rate:0.002500 loss: 1357.086133746151 precision:97.97 %
epoch: 22 learn rate:0.002500 loss: 1357.317741558411 precision:97.97 %
epoch: 23 learn rate:0.002500 loss: 1380.881231890703 precision:97.90 %
epoch: 24 learn rate:0.002500 loss: 1380.7802593014028 precision:97.88 %
epoch: 25 learn rate:0.002500 loss: 1359.0091715352044 precision:98.04 %
epoch: 26 learn rate:0.002500 loss: 1404.549285872783 precision:97.92 %
epoch: 27 learn rate:0.002500 loss: 1363.433762134613 precision:98.00 %
epoch: 28 learn rate:0.002500 loss: 1374.0441104002864 precision:97.96 %
epoch: 29 learn rate:0.002500 loss: 1375.8830764735137 precision:98.00 %
epoch: 30 learn rate:0.002500 loss: 1386.9668094842161 precision:97.89 %
```

```
epoch: 31 learn rate:0.001250 loss: 1367.7424992471745 precision:97.97 %
epoch: 32 learn rate:0.001250 loss: 1372.9246522168346 precision:97.98 %
epoch: 33 learn rate:0.001250 loss: 1381.3468404331534 precision:97.89 %
epoch: 34 learn rate:0.001250 loss: 1380.0783523514701 precision:98.00 %
epoch: 35 learn rate:0.001250 loss: 1372.6173038764584 precision:97.99 %
epoch: 36 learn rate:0.001250 loss: 1378.5270509137035 precision:97.95 %
epoch: 37 learn rate:0.001250 loss: 1372.1037904512789 precision:98.00 %
epoch: 38 learn rate:0.001250 loss: 1372.3684245908255 precision:97.99 %
epoch: 39 learn rate:0.001250 loss: 1372.4520821589663 precision:98.06 %
epoch: 40 learn rate:0.001250 loss: 1383.339718857074 precision:98.00 %
epoch: 41 learn rate:0.000625 loss: 1377.9204475133206 precision:98.04 %
epoch: 42 learn rate:0.000625 loss: 1383.2196415410144 precision:97.98 %
epoch: 43 learn rate:0.000625 loss: 1384.4226520409625 precision:97.91 %
epoch: 44 learn rate:0.000625 loss: 1377.7071068015978 precision:97.98 %
epoch: 45 learn rate:0.000625 loss: 1390.8720981576716 precision:98.02 %
epoch: 46 learn rate:0.000625 loss: 1370.8829100603684 precision:98.06 %
epoch: 47 learn rate:0.000625 loss: 1376.282920949894 precision:97.99 %
epoch: 48 learn rate:0.000625 loss: 1379.7642699099379 precision:98.04 %
epoch: 49 learn rate:0.000625 loss: 1377.6602682456435 precision:97.99 %
epoch: 50 learn rate:0.000625 loss: 1385.3762293628388 precision:98.02 %

Process finished with exit code 0
```

训练次数**epoch=20**，截止学习率值**last_lr = 0.0001**，学习方式**train_method=2**，最高准确率为**97.96%**

```
D:\DevelopTools\python\python.exe D:\code\SE\ML\神经网络\neural-network.py
epoch: 1 learn rate:0.010000 loss: 8621.14165159411 precision:86.30 %
epoch: 2 learn rate:0.010000 loss: 4292.567513382392 precision:93.38 %
epoch: 3 learn rate:0.010000 loss: 3108.2889942253305 precision:95.20 %
epoch: 4 learn rate:0.010000 loss: 2535.9486515576464 precision:96.04 %
epoch: 5 learn rate:0.010000 loss: 2324.833097657753 precision:96.30 %
epoch: 6 learn rate:0.010000 loss: 2070.298155368538 precision:96.69 %
epoch: 7 learn rate:0.010000 loss: 1880.4638509032636 precision:97.08 %
epoch: 8 learn rate:0.010000 loss: 1704.771873322336 precision:97.26 %
epoch: 9 learn rate:0.010000 loss: 1710.5934843672574 precision:97.29 %
epoch: 10 learn rate:0.010000 loss: 1618.3810096086252 precision:97.39 %
epoch: 11 learn rate:0.010000 loss: 1535.3116900946104 precision:97.57 %
epoch: 12 learn rate:0.010000 loss: 1572.3621224666113 precision:97.57 %
epoch: 13 learn rate:0.010000 loss: 1501.7408383285544 precision:97.73 %
epoch: 14 learn rate:0.010000 loss: 1506.0381469973436 precision:97.77 %
epoch: 15 learn rate:0.010000 loss: 1491.6491209434423 precision:97.71 %
epoch: 16 learn rate:0.001000 loss: 1514.3426726330435 precision:97.80 %
epoch: 17 learn rate:0.001000 loss: 1378.5497625527041 precision:97.90 %
epoch: 18 learn rate:0.001000 loss: 1376.4633264589775 precision:97.93 %
epoch: 19 learn rate:0.001000 loss: 1371.267190310222 precision:97.93 %
epoch: 20 learn rate:0.000100 loss: 1377.8025367644643 precision:97.96 %

Process finished with exit code 0
```

训练次数**epoch=50**，截止学习率值**last_lr = 0.0001**，学习方式**train_method=2**，最高准确率为**97.79%**

```
D:\DevelopTools\python\python.exe D:\code\SE\ML\神经网络\neural-network.py
epoch: 1 learn rate:0.010000 loss: 8962.540741639254 precision:86.57 %
epoch: 2 learn rate:0.010000 loss: 4548.650357890707 precision:92.73 %
epoch: 3 learn rate:0.010000 loss: 3313.985447249911 precision:94.64 %
epoch: 4 learn rate:0.010000 loss: 2587.6923397567907 precision:95.70 %
epoch: 5 learn rate:0.010000 loss: 2204.3425279572307 precision:96.51 %
epoch: 6 learn rate:0.010000 loss: 1939.8935287064849 precision:97.01 %
epoch: 7 learn rate:0.010000 loss: 1949.42317233049 precision:96.88 %
epoch: 8 learn rate:0.010000 loss: 1633.4583600126414 precision:97.31 %
epoch: 9 learn rate:0.010000 loss: 1570.5693206691085 precision:97.31 %
epoch: 10 learn rate:0.001000 loss: 1630.6033740798457 precision:97.46 %
epoch: 11 learn rate:0.001000 loss: 1429.986576806847 precision:97.59 %
epoch: 12 learn rate:0.001000 loss: 1429.6386553440307 precision:97.66 %
epoch: 13 learn rate:0.001000 loss: 1425.5758542927917 precision:97.63 %
epoch: 14 learn rate:0.001000 loss: 1419.5202570659458 precision:97.60 %
epoch: 15 learn rate:0.001000 loss: 1410.3271379927385 precision:97.67 %
epoch: 16 learn rate:0.001000 loss: 1405.898689505859 precision:97.69 %
epoch: 17 learn rate:0.001000 loss: 1396.2447302510575 precision:97.72 %
epoch: 18 learn rate:0.001000 loss: 1389.2798728086905 precision:97.76 %
epoch: 19 learn rate:0.001000 loss: 1390.5114162170132 precision:97.79 %
epoch: 20 learn rate:0.001000 loss: 1387.3824383725648 precision:97.78 %
epoch: 21 learn rate:0.001000 loss: 1381.8675179081022 precision:97.75 %
epoch: 22 learn rate:0.000100 loss: 1388.1628355253463 precision:97.78 %
epoch: 23 learn rate:0.000100 loss: 1375.9505632178389 precision:97.77 %
epoch: 24 learn rate:0.000100 loss: 1375.2619007455166 precision:97.79 %
epoch: 25 learn rate:0.000100 loss: 1375.4831775106468 precision:97.77 %
epoch: 26 lr:0.000010 loss: 1375.6314834269572 precision:97.79 %

Process finished with exit code 0
```

H2 六、数据集可视化

由于好奇MNIST的数据存储格式，额外实现了二进制码转图片的数据可视化。

代码如下：

```
root = "./data"
train_set = (
    mnist.read_image_file(os.path.join('./data', 'train-images-idx3-
ubyte')),
    mnist.read_label_file(os.path.join('./data', 'train-labels-idx1-
ubyte'))
)
test_set = (
    mnist.read_image_file(os.path.join('./data', 't10k-images-idx3-
ubyte')),
    mnist.read_label_file(os.path.join('./data', 't10k-labels-idx1-
ubyte'))
)
print("training set :", train_set[0].size())
print("test set :", test_set[0].size())


def convert_to_img(select):
    f = open(root + 'train.txt', 'w')
    if select == "train":
        data_path = root + '/train/'
    else:
```

```
        data_path = root + '/test/'

    if not os.path.exists(data_path):
        os.makedirs(data_path)
    for i, (img, label) in enumerate(zip(train_set[0], train_set[1])):
        img_path = data_path + str(i) + '.jpg'
        io.imsave(img_path, img.numpy())
        f.write(img_path + ' ' + str(label) + '\n')
    f.close()


convert_to_img("train")  # 转换训练集
convert_to_img("test")  # 转换测试集
```