Lab1 简单神经网络

北京航空航天大学计算机学院

1　实验目标

搭建基本的多层神经网络，并在给定测试集上进行精度测试，要求：

- 不使用深度学习框架完成网络搭建
- 不限制编程语言，推荐使用 python 进行神经网络搭建，允许使用 numpy，cupy 等工具包
- 使用给定的训练集和测试集，可使用提供的模板项目（BP-Mnist-Numpy-Template）并在其基础上进行修改，也可以重新进行编写

2　数据集获取与使用

2.1　基本说明（必读）

本次实验采用 Mnist 官方数据集[1]，模板项目的 dataset 目录下亦有相关文件。

数据文件分为训练集和测试集，分别对模型进行训练以及训练后的模型的精度测试。每个数据集分别使用两个文件存储图片样本和标签，具体可参考表 1的说明。

表 1: 数据文件说明

文件名	样本数	文件说明
t10k-images.idx3-ubyte	10000	测试集数据文件
t10k-labels.idx1-ubyt	10000	测试集标签文件
train-images.idx3-ubyte	60000	训练集数据文件
train-labels.idx1-ubyt	60000	训练集标签文件

2.2　数据文件存储结构（了解）

以 train-images.idx3-ubyte 文件为例：

表 2: 数据集文件内容

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):			
[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
...			
xxxx	unsigned byte	??	pixel

[1] http://yann.lecun.com/exdb/mnist/

前面若干字节为文件参数信息，包括了图片数量，图片行列大小等。从第 16 个字节起是图片的像素信息。

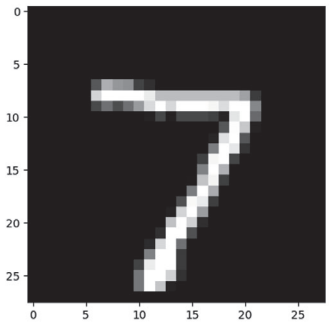

图 1: 示例图片

图片经过灰度处理后如图 1所示，即图像数据是一个单通道、大小为 28x28 的矩阵，矩阵中值的范围为 0 ∼ 255，表示每个像素点的数值大小。

标签文件 train-labels-idx1-ubyte 内容结构与图片文件类似，有兴趣的同学可以参阅[1] 自行理解。

3　主要代码介绍

代码包含三个文件，其中 main.py 是程序运行的入口，model.py 实现了网络的类，utils.py 中实现了数据集的读取。下面对重要的代码部分进行介绍。

3.1　数据集获取

本次实验，我们为大家提供了数据读取函数 load_mnist，函数存储路径为：BP-Mnist-Numpy-Template/utils.py。函数包含两个输入，分别是 path 和 kind。其中 path 表示存储上述 4 个数据集文件的目录路径，kind 表示想要提取的是训练集 ('train') 还是测试集 ('t10k')。

```
from utils import load_mnist

path = './dataset/'
train_images, train_labels = load_mnist(path, kind='train')
test_images, test_labels = load_mnist(path, kind='t10k')

print(train_images.shape, train_labels.shape)
print(test_images.shape, test_labels.shape)
```

上述代码分别获取了训练集和测试集，打印 shape 后输出如下所示：

```
(60000, 784) (60000,)
(10000, 784) (10000,)
```

其中，images 中的 60000 和 10000 分别表示拥有 60000 张训练集样本和 10000 张测试集样本；784 来源于图像数据 28x28 被展平后的结果。labels 中的 60000 和 10000 分别为 images 中的正确标签 (0 ∼ 9)，例如第一张训练集图片正确答案是 7，那么 train_labels[0] 为 7。

3.2　模型初始化

```
# number of input, hidden and output nodes
input_nodes = 784
hidden_nodes = 200
output_nodes = 10

# learning rate
learning_rate = 0.1

# create instance of neural network
model = neuralNetwork(input_nodes, hidden_nodes, output_nodes,
   learning_rate)
```

模型根据 model.py 中的模型类进行实例化，该类包含 3 个必须实现的函数：1) ___init___: 初始化网络需要的变量，例如各层大小、初始化权重等；2) forward: 实现神经网络前向传播的过程；3) backpropagation: 实现神经网络反向传播计算的过程，返回最后计算的 loss。

3.3　训练模型

```
def train(net, features, labels):
    for e in range(epochs):
        for i, feat in enumerate(features):
            ...
            net.forward(feat)
            loss += net.backpropagation(targets)
            ...
```

train() 函数实现了模型的训练过程，最外层循环表示使用训练集训练的轮数 (epochs)；下一层循环则是从数据集中获取数据，使用数据进行模型前向传播和反向传播（在反向传播中实现模型权重更新）。

3.4　测试精度

evaluate() 函数实现模型精度测试，该过程训练过程很相似，不同之处在于不需要进行反向传播，只需要根据模型前向传播结果计算模型精度即可。此处不再赘述。

3.5　保存权重

```
# Save weight parameters
np.save('./weights/who.npy', model.who)
np.save('./weights/whi.npy', model.wih)
```

将网络权重存储为.npy 文件，此处仅有 3 层全连接层，因此保存两份权重，如果有多层全连接层，用类似的方法存储即可。

4　作业要求

实现 model.py 中的网络类，并使用 main.py 能够正常运行（模版代码不是强制性的，可根据需求修改模板代码）。网络要求如下所示：

- 网络输入：784 个输入节点，每个节点对应图片的一个像素
- 网络输出：10 个输出节点，表示网络最后进行 10 分类
- 网络结构：4 层或 5 层全连接层
- 使用给定训练集（train-labels-idx1-ubyte，train-images-idx3-ubyte）进行权重训练，使用测试集（t10k-images-idx3-ubyte，t10k-labels-idx3-ubyte）测试并记录测试精度，不对精度做特别的要求，只需在合理范围内即可（>80%）

5　作业提交

- 将训练得到的权重数据用 numpy 保存为.npy 格式的文件，保存在 BP-Mnist-Numpy-Template/weights 目录下，让结果可以复现 (保存方法见 3.5小节)
- 撰写实验报告，包括但不限于网络原理说明、网络实现代码介绍、测试精度结果截图等。报告命名格式：学号 + 姓名.docx(.pdf)，如 XXXXXXXXX+ 张三 + 作业一报告.docx(.pdf)（更多细节参考“实验报告撰写格式”）
- 将完整的工程文件夹 BP-Mnist-Numpy-Template（**需要包含权重 weights 文件夹，不需要数据文件夹 dataset**）、实验报告打包成压缩文件，命名格式：学号 + 姓名.zip(.rar)，如 XXXXXXXXX+ 张三 + 作业一.zip(.rar)

6　其他

- 本次作业训练集大小为 60000，若使用 CPU 训练时间过长，可以考虑采用 GPU，或者分 batch 训练等方式（模板项目未进行 batch 划分，若有需求同学们可自己实现）。
- 由于本次作业并不涉及深度学习框架，只使用 numpy 等矩阵运算包，所以要想进行 GPU 加速则可以采用 CuPy，具体可以参考网上的相关资源[2] [3]。
- 网络实现代码可参考 network.py 文件中的实现

[2] https://cupy.dev/
[3] https://github.com/cupy/cupy