深度学习训练营

案例 1：Softmax 实现手写数字识别

1 简介

本次案例中，你需要用 python 实现 Softmax 回归方法，用于 MNIST 手写数字数据集分类任务。你需要完成前向传播和反向传播算法。

前向传播算法中，你需要实现 Softmax 函数和交叉熵损失函数的计算。

$$y = softmax(W^T x + b)$$
$$L = CrossEntropy(y, label)$$

反向传播算法中，你需要实现参数梯度、局部敏感度的计算，并按照随机梯度下降法来更新参数。

$$\frac{\partial L}{\partial W}, \frac{\partial L}{\partial b}, \delta$$

具体计算方法可自行推导，或参照第三章课件。

2 MNIST 数据集

MNIST 手写数字数据集是机器学习领域中广泛使用的图像分类数据集。它包含 60,000 个训练样本和 10,000 个测试样本。这些数字已进行尺寸规格化，并在固定尺寸的图像中居中。每个样本都是一个 784×1 的矩阵，是从原始的 28×28 灰度图像转换而来的。MNIST 中的数字范围是 0 到 9。下面显示了一些示例。注意：在训练期间，切勿以任何形式使用有关测试样本的信息。

3 任务要求

1. 代码清单

a) data/ 文件夹：存放 MNIST 数据集。你需要下载数据，解压后存放于该文件夹下。下载链接见文末，解压后的数据为 *ubyte 形式；

b) solver.py 这个文件中实现了训练和测试的流程。建议从这个文件开始阅读代码；

c) network.py 实现了网络模块，可在定义网络体系结构和进行模型训练时

使用；

d) dataloader.py 实现了数据加载器，可用于准备数据以进行训练和测试；
e) visualize.py 实现了 plot_loss_and_acc 函数，该函数可用于绘制损失和准确率曲线；
f) optimizer.py 你需要实现带 momentum 的 SGD 优化器，可用于执行前向和反向传播；
g) layers/fc_layer.py 你需要实现全连接层的前向和反向传播，该层将输入向量 x 映射到输出向量 u，其中 u = Wx + b，其中 W 是该层权重，b 是偏置；
h) loss.py 你需要实现 softmax_cross_entropy_loss；
i) runner.ipynb 完成所有代码后的执行文件，执行训练和测试过程。

2. 要求

我们提供了完整的代码框架，你只需要完成 **optimizer.py**，**layers/fc_layer.py**，**loss.py** 中的 **#TODO** 部分。你需要提交这三个代码文件和带有结果的 **runner.ipynb** 并且附一个报告内容包括：

a) 记录训练和测试的准确率。画出训练损失和准确率曲线；
b) 比较使用和不使用 momentum 结果的不同，可以从训练时间，收敛性和准确率等方面讨论差异；
c) 调整其他超参数，如学习率，Batchsize 等，观察这些超参数如何影响分类性能。写下观察结果并将这些新结果记录在报告中。

4 其他

1. 注意代码的执行效率，尽量不要使用 for 循环；
2. 不要在报告中粘贴很多代码(只能包含关键代码)，对添加的代码作出解释;
3. 不要使用任何深度学习框架，如 TensorFlow，Pytorch 等；
4. 禁止抄袭。

5 参考

1. 数据集下载：http://yann.lecun.com/exdb/mnist/