HW1-批改标准

本次作业基础分15分，扣分制。有加分项，加分最多2分。

代码

- `run_mlp.py`（或自己编写的主程序）能正常运行，否则扣10分。
- 相关代码填补正确，一处错误扣1分。参考代码如下：

`layers.py`

```
class Relu(Layer):
    def __init__(self, name):
        super(Relu, self).__init__(name)

    def forward(self, input):
        self._saved_for_backward(input)
        return np.maximum(0, input)

    def backward(self, grad_output):
        input = self._saved_tensor
        return grad_output * (input > 0)

class Sigmoid(Layer):
    def __init__(self, name):
        super(Sigmoid, self).__init__(name)

    def forward(self, input):
        output = 1 / (1 + np.exp(-input))
        self._saved_for_backward(output)
        return output

    def backward(self, grad_output):
        output = self._saved_tensor
        return grad_output * output * (1 - output)

class Linear(Layer):
    def __init__(self, name, in_num, out_num, init_std):
        super(Linear, self).__init__(name, trainable=True)
        self.in_num = in_num
        self.out_num = out_num
        self.W = np.random.randn(in_num, out_num) * init_std
        self.b = np.zeros(out_num)

        self.grad_W = np.zeros((in_num, out_num))
        self.grad_b = np.zeros(out_num)

```

```
        self.diff_W = np.zeros((in_num, out_num))
        self.diff_b = np.zeros(out_num)

    def forward(self, input):
        self._saved_for_backward(input)
        output = np.dot(input, self.W) + self.b
        return output

    def backward(self, grad_output):
        input = self._saved_tensor
        self.grad_W = np.dot(input.T, grad_output)
        self.grad_b = np.sum(grad_output, axis=0)
        return np.dot(grad_output, self.W.T)

    def update(self, config):
        mm = config['momentum']
        lr = config['learning_rate']
        wd = config['weight_decay']

        self.diff_W = mm * self.diff_W + (self.grad_W + wd * self.W)
        self.W = self.W - lr * self.diff_W

        self.diff_b = mm * self.diff_b + (self.grad_b + wd * self.b)
        self.b = self.b - lr * self.diff_b
```

`loss.py`

```
class EuclideanLoss(object):
    def __init__(self, name):
        self.name = name

    def forward(self, input, target):
        return 0.5 * np.mean(np.sum(np.square(input - target), axis=1))

    def backward(self, input, target):
        return (input - target) / len(input)
```

易错点：

- `Sigmoid.backward` 导数计算错误
- `EuclideanLoss.forward` 没有除以batch_size
- 计算梯度时没有除以batch_size

报告

- 至少包括四个实验（一个、两个隐藏层的MLP，激活函数分别为Relu、Sigmoid），需要报告测试准确率，训练时间，loss变化曲线。缺一个实验扣1分，loss没画图扣1分，未报告准确率具体数值扣1分。一般来说 Relu 测试准确度在 98% 以上，Sigmoid 在 95% 以上。准确率较低视情况扣分。

- 超参数设置参考：

```
config = {
    'learning_rate': 0.01,
    'weight_decay': 0.0005,
    'momentum': 0.9,
    'batch_size': 100,
    'max_epoch': 100,
    'disp_freq': 50,
    'test_epoch': 5
}
```

  1层MLP结构为784 -- 256 -- 10，2层MLP结构为784 -- 500 -- 256 -- 10。初始化std = 0.01。以上参数并不唯一，但如果某些参数设置导致最后得出结论错误，应当指出可能原因。

- 结论：ReLU网络好于Sigmoid网络，accuracy更高，收敛更快； 2层网络略好于1层网络，但是运行时间更长。如果得出结论有错误之处，有可能问题出在参数设置上，应指出并扣分。以上结论错一处扣1分。没有给出结论扣3分。
- 加分项
  - 对于超参数进行探索，并研究对于最终结果影响。比如对于learning rate， weight decay，momentum， hidden size的调整。
  - 结合实验结果对网络结构、激活函数等深入分析，如从过拟合、loss曲线的不同以及理论的角度分析激活函数的性质。
  - 一般加1分，最多加2分。

补交

每迟一天扣一分，对于补充说明原来作业的以及有特殊情况的，可酌情扣分。