

Python 程序设计作业

题目:

用 numpy 实现神经网络全连接层，并使用所实现的神经网络完成以下某一任务（二选一）：

1. 手写数字(mnist)数据集(`sklearn.datasets.load_digits()`) 上完成多分类任务
2. 糖尿病数据集(`sklearn.datasets.load_diabetes()`) 完成回归任务

要求:

- 使用numpy库来实现模型，不可使用PyTorch, Keras, TensorFlow, PaddlePaddle, MXNet, Caffe, Theano等深度学习框架。
- 文档需要详细说明定义的每一个类(class)、方法(method)及函数(function)，以及整个代码实现逻辑。
- 模型训练的过程或结果（loss值）需要绘图展示，且在文档中说明。
- 原则上要求仅使用numpy (用于建立模型)，scikit-learn (用于下载数据集) 及Python标准库，若有其他需求需要和助教说明。

建议:

- 参考《深度学习入门：基于Python的理论与实现》，书籍电子版及其配套代码[下载](#)
- 数据集通过scikit-learn下载
- 模型实现框架可参考附录1

进阶:

- 编码规范采用Google Python Style，英文参考 [Google Python Style Guide](#)
中文参考: [Python 编码规范\(Google\)](#)
- 用numpy实现卷积神经网络，并使用卷积网络完成手写数字(mnist)数据集上的分类任务

附录1

```
1 class Linear(object):
2     '''全连接层
3
4     参数:
5     -----
6     n_out: int
7         输出维度
8     n_in: int
9         输入维度
10    '''
11    def __init__(self, n_in, n_out):
12        ...
13
14    def __call__(self, input, *args, **kwargs):
15        return self.forward(input, *args, **kwargs):
16
17    def forward(self, input, *args, **kwargs):
18        ...
19
20    def backward(self, preGrad, *args, **kwargs):
21        ...
22
```

```

23
24 class Relu(object):
25     '''Relu 激活函数
26     '''
27     def __init__(self):
28         ...
29
30     def forward(self, input, *args, **kwargs):
31         ...
32
33     def backward(self, preGrad, *args, **kwargs):
34         ...
35
36 class Sigmoid(object):
37     '''Sigmoid 激活函数
38     '''
39     def __init__(self):
40         ...
41
42     def forward(self, input, *args, **kwargs):
43         ...
44
45     def backward(self, preGrad, *args, **kwargs):
46         ...
47
48 class SoftmaxWithLoss(object):
49     '''附带交叉熵损失的SoftMax
50     '''
51     def __init__(self):
52         ...
53
54     def __call__(self, pred, label):
55         return self.forward(pred, label)
56
57     def forward(self, pred, label):
58         ...
59
60     def backward(self, preGrad=None):
61         ...
62
63 class MSE(object):
64     '''MSE 损失函数
65     '''
66     def __init__(self):
67         ...
68
69     def __call__(self, pred, label):
70         return self.forward(pred, label)
71
72     def forward(self, pred, label):
73         ...
74
75     def backward(self, grad=None):
76         ...
77
78 class SGD(object):
79     '''随机梯度下降
80     '''
81     def __init__(self, params, lr=0.01, momentum=0.9):

```

```
81         self.params=params
82         self.lr=lr
83         self.momentum=momentum
84
85     def zero_grad(self):
86         ...
87
88     def step(self):
89         ...
```