深度学习
Lab7-recurrent neural network

兰韵诗

本次Lab有作业，请在4月25号结束之前提交！！！

Lab4参考答案

```python
def compute_loss(self, log_prob, labels):
    '''
    log_prob is the predicted probabilities
    labels is the ground truth
    Please return the loss
    '''

    # ==========
    # todo '''请完成多分类问题的损失计算 损失为:  交叉熵损失 + L2正则项'''
    # ==========

    loss = np.sum(np.sum(-log_prob * labels, axis=1)) + self.lambda1*(np.sum(self.W1**2) + np.sum(self.W2**2))

    return loss
```

```python
class Relu:
    def __init__(self):
        self.mem = {}

    def forward(self, x):
        self.mem['x'] = x
        return np.where(x > 0, x, np.zeros_like(x))

    def backward(self, grad_y):
        '''
        grad_y: same shape as x
        '''

        # ==========
        # todo '''请完成激活函数的梯度后传'''
        # ==========

        x = self.mem['x']
        return (x > 0).astype(np.float32) * grad_y
```

```python
def forward(self, x):
    '''
    x is the input features
    Please return the predicted probabilities of x
    '''

    # ==========
    # todo '''请搭建一个MLP前馈神经网络 补全它的前向传播 MLP结构为FFN --> RELU --> FFN --> Softmax'''
    # ==========
    x = x.reshape(x.shape[0], -1)
    bias = np.ones(shape=[x.shape[0], 1])
    x = np.concatenate([x, bias], axis=1) # (batch_size, num_inputs+1)

    self.h1 = self.mul_h1.forward(self.W1, x.T)
    self.h1_relu = self.relu.forward(self.h1)
    self.h2 = self.mul_h2.forward(self.W2, self.h1_relu)
    self.h2_soft = self.softmax.forward(self.h2.T)
    h2_log = self.log.forward(self.h2_soft)

    return h2_log
```

Lab4参考答案

```python
def update(self):
    '''
    Please update self.W1 and self.W2
    '''

    # ==========
    # todo '''更新该前馈神经网络的参数'''
    # ==========

    self.W1 -= self.lr * (self.W1_grad + self.lambda1*self.W1)
    self.W2 -= self.lr * (self.W2_grad + self.lambda1*self.W2)
```

```python
def backward(self, label):
    '''
    label is the ground truth
    Please compute the gradients of self.W1 and self.W2
    '''

    # ==========
    # todo '''补全该前馈神经网络的后向传播算法'''
    # ==========

    self.h2_log_grad = self.log.backward(-label)
    self.h2_soft_grad = self.softmax.backward(self.h2_log_grad)
    #print(self.h2_soft_grad.T.shape)
    self.h2_grad, self.W2_grad = self.mul_h2.backward(self.h2_soft_grad.T)
    self.h1_relu_grad = self.relu.backward(self.h2_grad)
    self.h1_grad, self.W1_grad = self.mul_h1.backward(self.h1_relu_grad)
```

Lab4常见错误

forward最后没有经过log.forward层

Python 自动换行

```python
def forward(self, x):
    # 第一个全连接层
    # 检查输入特征 x 的形状

    # 将输入特征扁平化为一维向量
    x_flattened = np.reshape(x, (x.shape[0], -1))

    # 添加偏置项并检查形状
    x_with_bias = np.concatenate((x_flattened, np.ones((x_flattened.shape[0], 1))), axis=1)

    # 进行矩阵乘法操作
    h1 = self.mul_h1.forward(self.W1, np.transpose(x_with_bias))
    h1_relu = self.relu.forward(h1)

    # 第二个全连接层
    h2 = self.mul_h2.forward(self.W2, h1_relu)
    probs = self.softmax.forward(h2.T)

    # 将输出的形状转置为 (60000, 10)
    self.probs = probs
    return probs
```

Lab4常见错误

backward label不是-label

```
def backward(self, label):
    grad_y = self.log.backward(label)
    grad_y = self.softmax.backward(grad_y)
    grad_y, grad_W2 = self.mul_h2.backward(grad_y.T)

    grad_y = self.relu.backward(grad_y)
    _, grad_W1 = self.mul_h1.backward(grad_y)

    self.grad_W1 = grad_W1
    self.grad_W2 = grad_W2
```

Lab4常见错误

更新部分写到forward

```
def backward(self, label):
    grad_log = self.log.backward(label)
    grad_softmax = self.softmax.backward(grad_log)
    grad_layer2,grad_W2 = self.mul_h2.backward(grad_softmax.T)
    grad_relu = self.relu.backward(grad_layer2)
    grad_layer1,grad_W1 = self.mul_h1.backward(grad_relu)

    self.W1 += self.lr * grad_W1
    self.W2 += self.lr * grad_W2
    # ==========


def update(self):
    '''
    Please update self.W1 and self.W2
    '''

    # ==========
    # todo '''更新该前馈神经网络的参数'''

    # ==========
```

Lab4常见错误

更新里面有backward

```
def update(self):
    '''
    Please update self.W1 and self.W2
    '''

    # ==========
    # todo '''更新该前馈神经网络的参数'''
    # ==========
    grad_W1, grad_W2 = self.backward(label)  # Assuming label is defined elsewhere
    self.W1 -= self.lr * grad_W1
    self.W2 -= self.lr * grad_W2
```

Lab7

1.熟悉文本生成任务的流程

2.补全rnn_hard_version.py 文件中的基于GRU的歌词预测模型

Recurrent Neural Network

- 根据提示，补全**基于GRU的歌词预测模型**代码
  - 利用设定好的输入完成**GRU的前向传播**和**歌词预测模型主体**
  - **正确定义和初始化GRU中的参数**
  - 不能调用其他工具包，不能调用pytorch内置的GRU模块，只能在 “to do” 下面书写代码
  - 提交之后，测试集上的准确率应该降到一个正确的范围内可多次提交。即使对自己的代码没有自信也一定要提交，我们会酌情给过程分
- **TO DO**： 完成《Recurrent Neural Network》项目。补全rnn_hard_version.py文件使exercise_rnn.py文件中的train_with_RNN_hard()可以顺利执行。

Evaluation脚本

```python
def compute_acc(pred_file):
    with open('data/jaychou_test_y.txt') as f:
        gold = f.readlines()
    gold = [sent.strip() for sent in gold]

    with open(pred_file) as f:
        pred = f.readlines()
    pred = [sent.strip() for sent in pred]
    correct_case = [i for i, _ in enumerate(gold) if gold[i] == pred[i]]

    acc = len(correct_case)*1./len(gold)
    print('the predicted accuracy is %s' %acc)

if __name__ == '__main__':
    pred_file = 'data/predict.txt'
    compute_acc(pred_file)
```

Note：为了测试方便，这里我们使用准确率作为我们的生成评估标准。实际生成任务一般采用BLEU，Rouge等