# 实验8《自编码器》

实验学时： 2 实验地点： 二综204 实验日期： 2019/12/05

1. **问题描述**

前见过的所以神经网络（比如全连接网络和卷积神经网络）都有一个主要特点，那就是它们都没有记忆。它们单独处理每个输入，在输入和输入之间没有保存任何状态。对于这样的网络，要想处理数据点的序列或者时间序列，你需要向网络同时展示整个序列，即将序列转换成单个数据点。例如，在实验九词嵌入中就是这么做的：将全部电影评论转换为一个大向量，然后一次性处理。

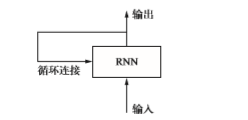
与此相反，当人在阅读这个句子时，是一个词一个词地阅读（或者说，眼睛一次扫视一次扫视地阅读），同时会记住之前的内容，这让你能够动态理解这个句子所传达的含义，以渐进的方式处理信息，同时保存一个关于所处理的内部模型，这就是循环神经网络，它广泛应用于自然语言处理中，本实验会介绍简单的循环神经网络（SimpleRNN）来处理文本数据，然后改进网络使用LSTM神经网络对比效果。

**二、 设计简要描述**

1．简单循环神经网络

#### 1.1 循环网络简介

RNN处理序列的方式是，遍历所以序列元素，并保存一个状态，其中包含与已查看内容相关的信息。实际上，RNN是一类具有内部环的的神经网络（如下图）。在处理两个不同的独立序列（比如两条不同的IMDB评论）之间，RNN状态会被重置，因此，你仍可以将一个序列看作单个数据点，即网络的单个输入。真正改变的是，数据点不再是在单个步骤中进行处理，相反，网络内部会对序列元素进行遍历。



#### 1.2 Keras中的循环层

在Keras中能够通过以下代码实现一个简单的RNN层

#### 1.3 数据处理

我们还是使用IMDB电影评论的数据，但是我们需要引入一个sequence模块把我们的数据格式化输入到神经网络中

#### 1.4 搭建自己RNN神经网络

1. 从kaeras导入全连接层模块
2. 定义一个序列模型
3. 添加一个Embedding层，参数是（max\_features,32）
4. 添加一个SimpleRNN层，输出维度32
5. 添加一个全连接层，输出维度1，激活函数‘sigmoid’
6. 编译模型，参数分别是‘rmsprop’,’binary\_crossentropy’,[‘acc’]
7. 拟合模型，epochs=10.batch\_size=128,validation\_split=0.2，结果返回给history

#### 1.5 绘制结果

能够通过以下代码绘制训练损失和准确率率结果

2. LSTM循环神经网络

#### 2.1 SimpleRNN的缺陷

SimpleRNN最大的问题是：在时刻t,理论上来说，它应该能够记住许多时间步之前见过的信息，但实际上它是不可能学到这种长期依赖的，其原因在于梯度消失问题，随着层数的增加，网络最终变得无法训练，LSTM层和GRU层都是为了解决这个问题而设计的。本实验主要介绍使用更为广泛的LSTM模型

LSTM层是SimpleRNN层的一种变体，它增加了一种携带信息跨越多个时间步的方法。假设设有一条传送带，其运行方向平行于你所处理的序列。序列中的信息可以在任意位置跳上传送带，然后被传送到更晚的时间步，并在需要是原封不动地跳回来。这实际上就是LSTM的原理：它保存信息以便后面使用，从而防止较早期的信号在处理过程中逐渐消失。

#### 2.2 搭建自己的LSTM神经网络

1）从kaeras导入LSTM模块

2）定义一个序列模型

3）添加一个Embedding层，参数是（max\_features,32）

4）添加一个LSTM层，输出维度32

5）添加一个全连接层，输出维度1，激活函数‘sigmoid’

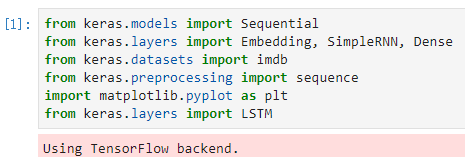
6） 编译模型，参数分别是‘rmsprop’,’binary\_crossentropy’,[‘acc’]

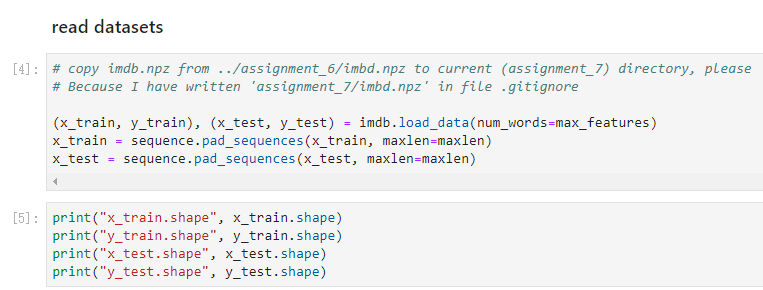
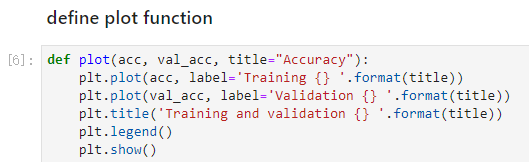
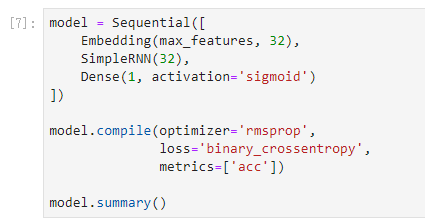
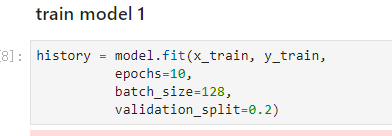
7）拟合模型，epochs=10.batch\_size=128,validation\_split=0.2，结果返回给history

#### 2.3 绘制结果

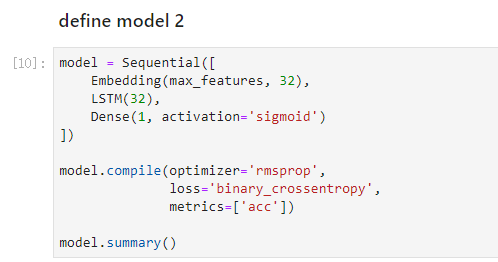
参考1.5节绘制损失值和准确率的结果图

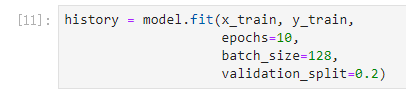
1. **程序清单**
2. **导入相应工具包**

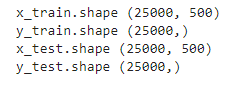
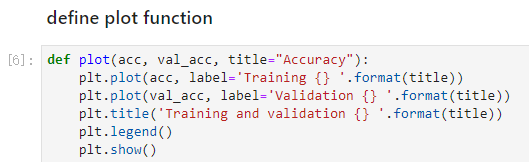
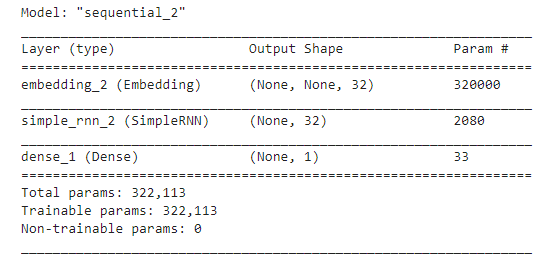
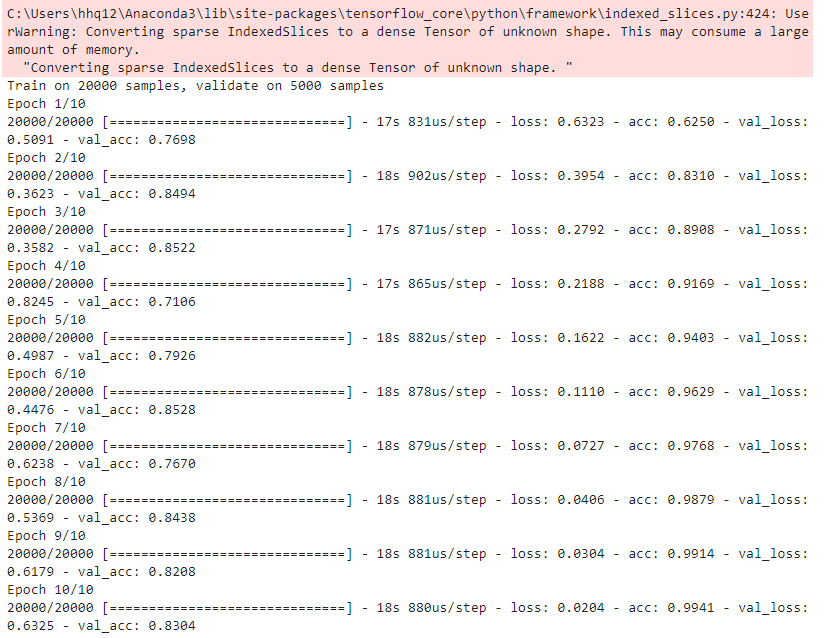
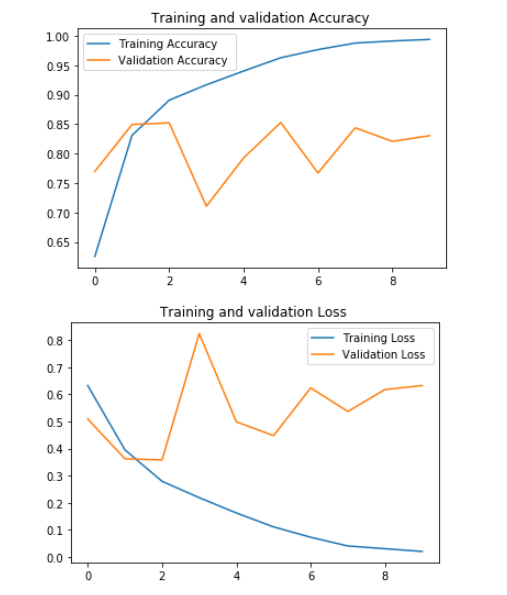
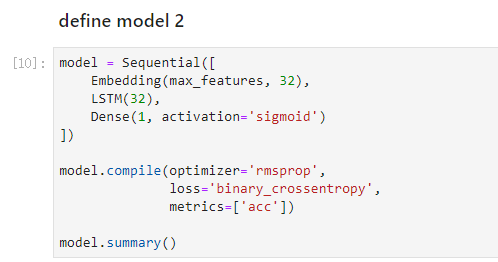
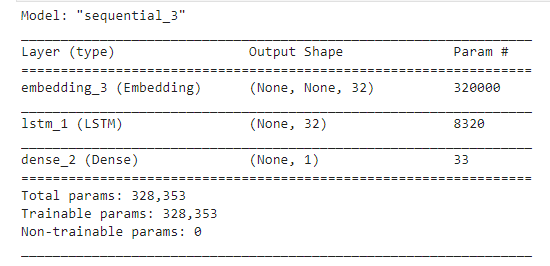
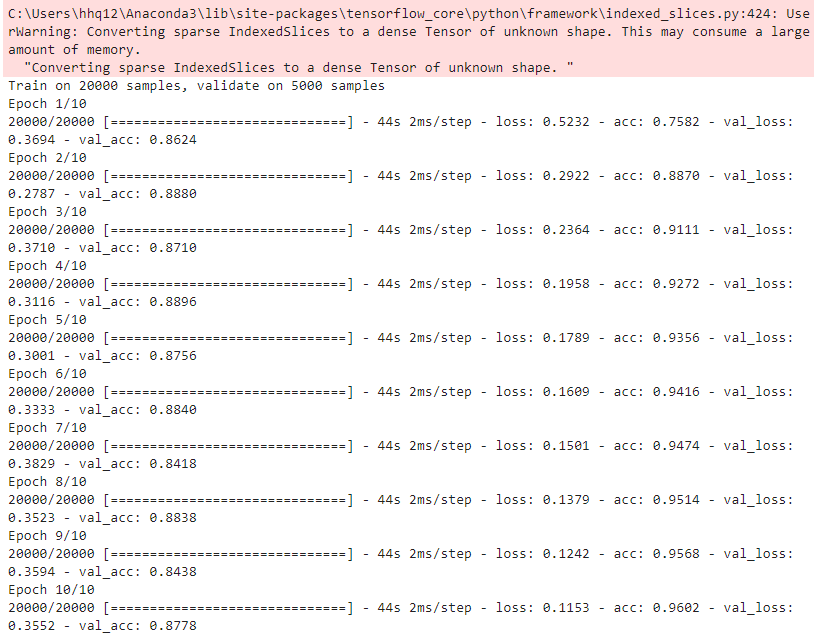


1. **读取数据**
2. **定义画图函数**
3. **定义简单RNN模型**
4. **训练模型**
5. **查看训练结果**



1. **定义LSTM模型**
2. **训练LSTM模型**



1. **查看模型训练结果**
2. **结果分析**
3. **导入相应工具包**
4. **读取数据**
5. **定义画图函数**
6. **定义简单RNN模型**
7. **训练模型**
8. **查看训练结果**
9. **定义LSTM模型**
10. **训练LSTM模型**
11. **查看模型训练结果**



1. **调试报告**
2. **所遇到的问题**

基本没遇到问题，一些小问题通过报错信息也很快就解决了。

1. **经验和体会**

使用keras定义神经网络还是挺方便实用的，一个复杂的RNN，使用keras几行代码就解决问题了。Keras简直越用越香。从训练的效果图来看，LSTM会比单纯的RNN好一点，看来LSTM确实比SimpleRNN实用吧。

源码链接：

<https://github.com/sunnyswag/ml_dl_assignments/blob/master/assignment_8/assignment_8.ipynb>