**循环神经网络RNN实验**

#### 实验目的

学习掌握循环神经网络（RNN）的基本原理及LSTM的基本结构；

掌握利用LSTM神经元构造循环神经网络进行训练和预测时间序列。

#### 实验内容

通过PC上位机连接服务器，登陆SimpleAI平台，利用python语言搭建基于LSTM的RNN模型。利用RNN模型对正线曲线或余弦曲线的数值变化进行预测。所用到的数据集由学生自己利用python编写代码采样生成。目标是利用正弦曲线或者余弦曲线的前10个点的数值预测下一个点的数值。

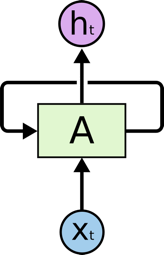
#### 实验环境

硬件：x86\_64 Centos 3.10.0服务器/GPU服务器、GPU、PC上位机

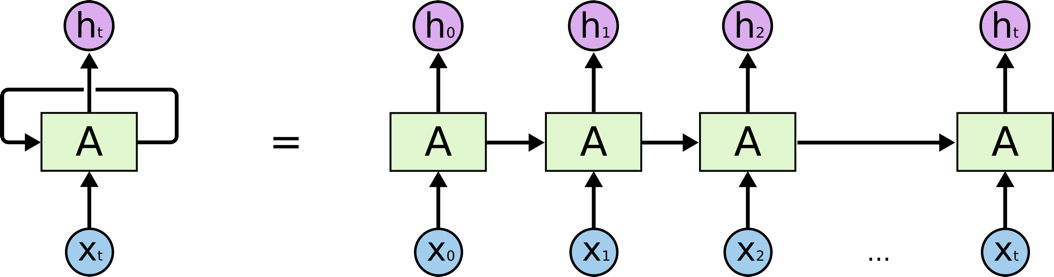
软件：SimpleAI实验平台、Docker下Ubuntu16.04镜像、python3.5、tensorflow1.7, numpy1.12.1

#### 实验原理

RNN是一种用于处理时序数据的神经网络模型。在传统神经网络中，模型不会关注上一时刻的处理会有什么信息可以用于下一时刻，每一次都只会关注当前时刻的处理。举个例子来说，我们想对一部影片中每一刻出现的事件进行分类，如果我们知道电影前面的事件信息，那么对当前时刻事件的分类就会非常容易。实际上，传统神经网络没有记忆功能，所以它对每一刻出现的事件进行分类时不会用到影片已经出现的信息。基于对这种时间序列关注的需求，循环神经网络应运而生。递归神经网络的结果与传统神经网络有一些不同，它带有一个指向自身的环，用来表示它可以传递当前时刻处理的信息给下一时刻使用，结构如下：

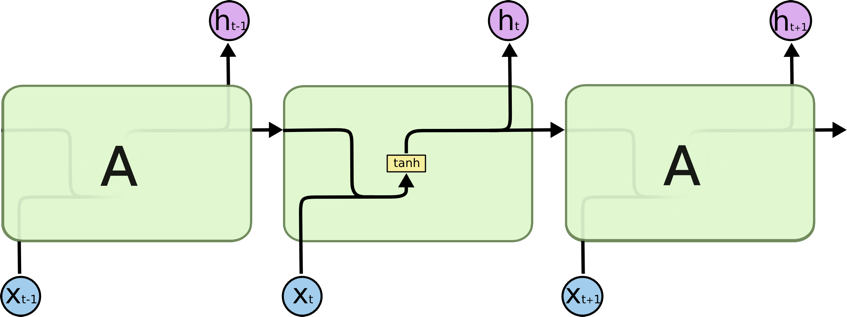


其中，为时间节点的输入，为模型处理部分，为时间节点的输出。为了更好地说明RNN的工作原理，可以将上图的循环网络结构展开，得到：

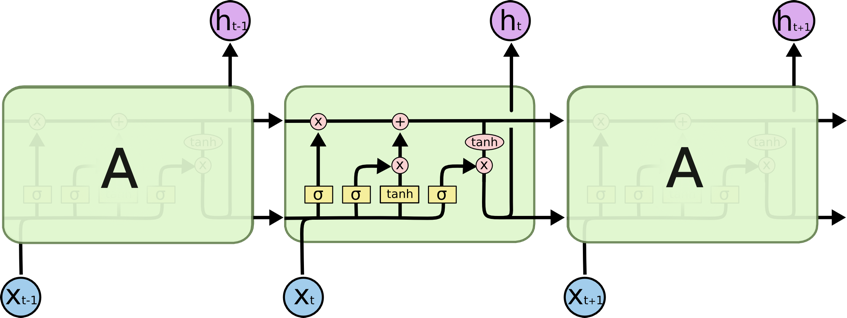


这样的一条链状神经网络代表了一个递归神经网络，可以认为它是对相同神经网络的多重复制，每一时刻的神经网络会传递信息给下一时刻。在本次试验中可以认为为当前时的正线曲线值，为预测的时刻的正弦曲线值。

不过RNN也有它自己的问题，即在当预测点与依赖的相关信息距离比较远的时候，就难以学到该相关信息。为了解决这类问题，人们发明了Long Short Term Memory（LSTM）。其结构与普通的循环神经网络的神经元结构对比如下：

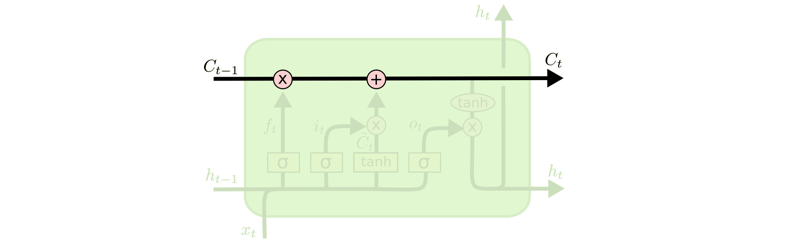


普通RNN

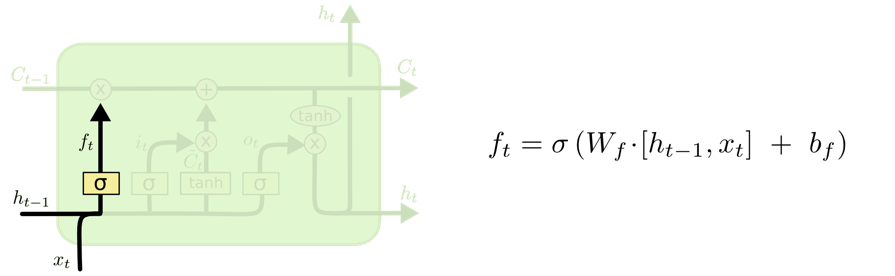


基于LSTM的RNN

理解LSTMs的关键就是下面的矩形方框，被称为memory block（记忆块），主要包含了三个门（forget gate、input gate、output gate）与一个记忆单元（cell）。方框内上方的那条水平线，被称为cell state（单元状态），它就像一个传送带，可以控制信息传递给下一时刻。

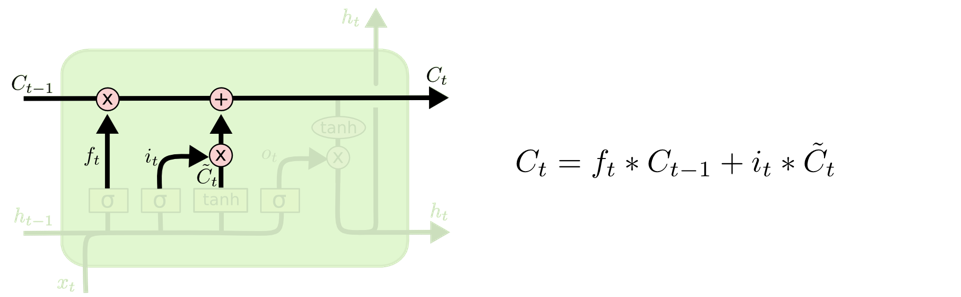


下面来逐步了解LSTM的工作原理。LSTM第一步是用来决定什么信息可以通过cell state。这个决定由“forget gate”层通过sigmoid来控制，它会根据上一时刻的输出和当前输入来产生一个0到1 的值，来决定是否让上一时刻学到的信息通过或部分通过。如下：

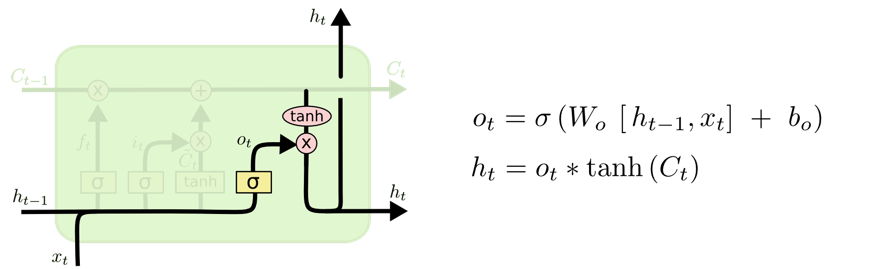


举个例子来说就是，我们在之前的句子中学到了很多东西，一些东西对当前来讲是没用的，可以对它进行选择性地过滤。

第二步是产生我们需要更新的新信息。这一步包含两部分，第一个是一个“input gate”层通过sigmoid来决定哪些值用来更新，第二个是一个tanh层用来生成新的候选值，它作为当前层产生的候选值可能会添加到cell state中。我们会把这两部分产生的值结合来进行更新。

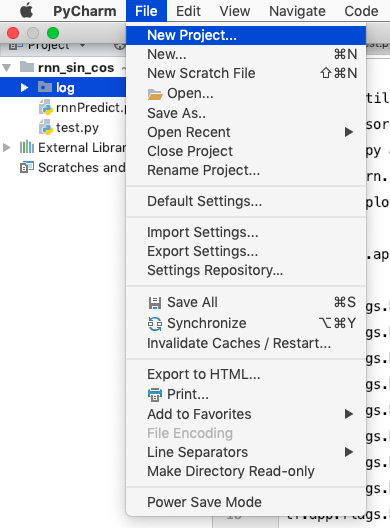


最后一步是决定模型的输出，首先是通过sigmoid层来得到一个初始输出，然后使用tanh将值缩放到-1到1间，再与sigmoid得到的输出逐对相乘，从而得到模型的输出。

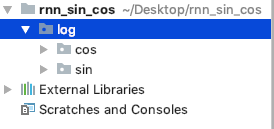


#### 实验步骤

1、打开pycharm编辑器，在上方工具栏中找到New Project选项新建项目，File->New Project。项目名称为rnn\_sin\_cos。

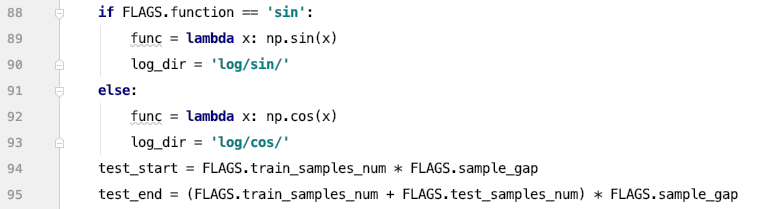


2、在rnn\_sin\_cos目录下新建子目录log/sin文件夹和log/cos文件夹。这两个文件夹将用于存储训练好的RNN模型目录结构如下图所示：

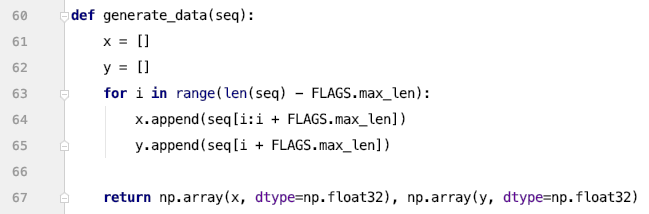


3、新建python文件，rnnPredict.py进行模型编写。

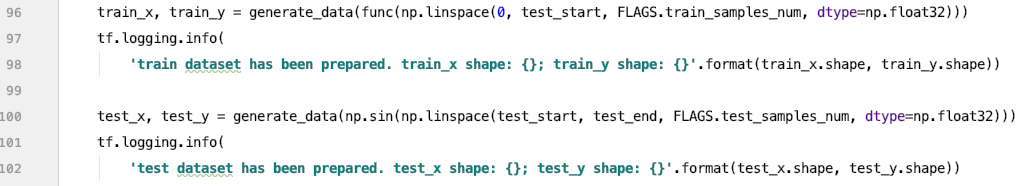
4、定义主函数，首先进行数据集的创建，首先利用numpy工具包进行正弦值或者余弦值序列的生成，并划分训练集和测试集的范围。



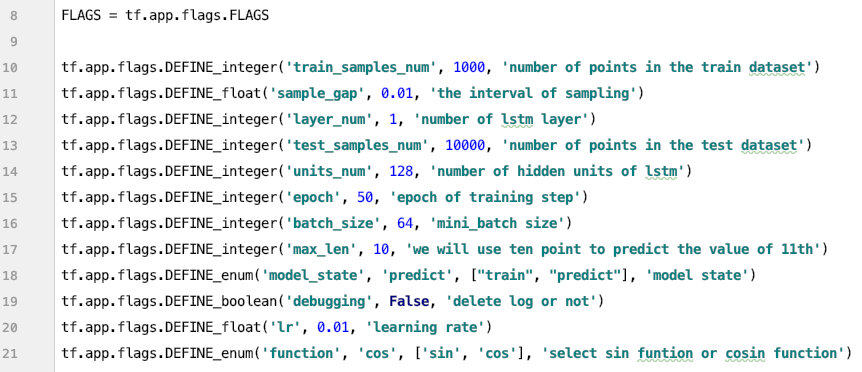
5、编写采样函数对正弦曲线值或者余弦曲线值进行采样。每11个采样点为一组数据，其中前10个为输入数据（x），最后一个为预测数据（label）。



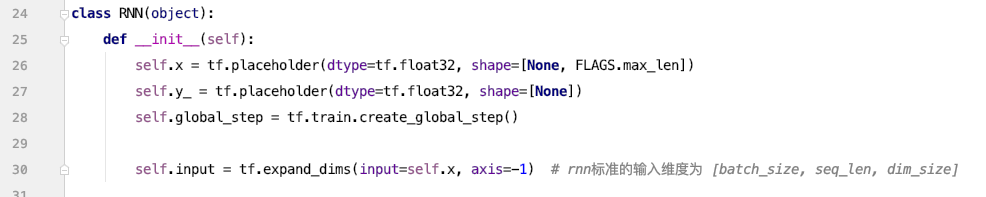
随后在主函数中调用采样函数生成训练集（x,y）对和测试集（x,y）对。



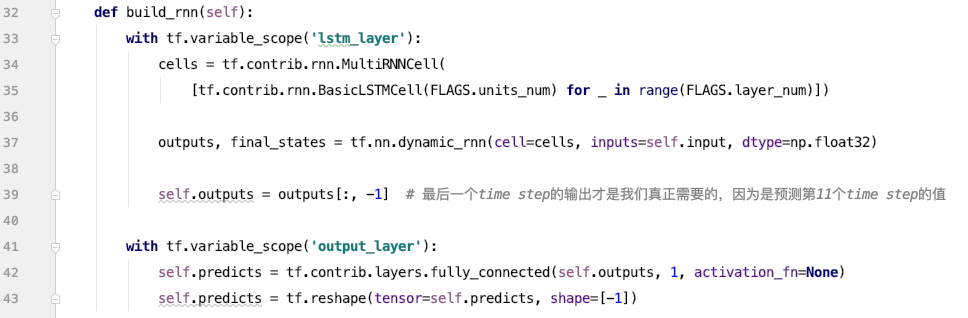
6、编写RNN模型，首先对超参数进行定于。在三个参数分别代表超参数变量名称、默认值、参数含义描述。需要特别注意的两个参数是model\_state和debugging。model\_state控制模型是训练状态还是预测状态，训练状态将进行反向传播进行优化，预测状态只会前向传播进行预测。debugging参数用于控制是否删除当前保存的模型，debugging为True即为重新训练，为False则读取原有模型继续训练。



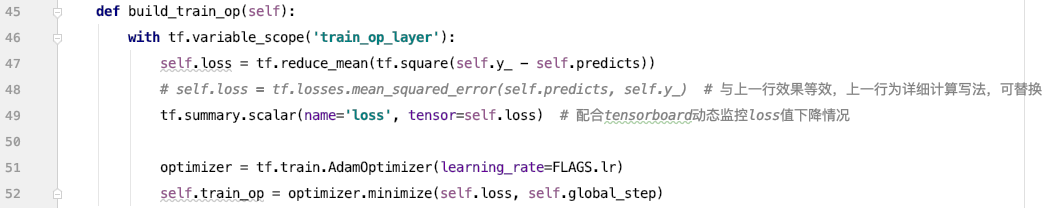
7、创建模型类，名为RNN，编写初始化函数。定义输入x和真实标签y。



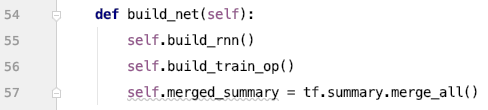
8、编写RNN模型定义函数。



9、编写优化操作，用于模型训练。



10、编写模型创建函数，该函数为class对外界的接口，用于创建模型图对象。



11、至此RNN模型定义完毕，回到主函数定义模型训练过程。首先创建RNN对象实体并调用build\_net函数构建模型图。



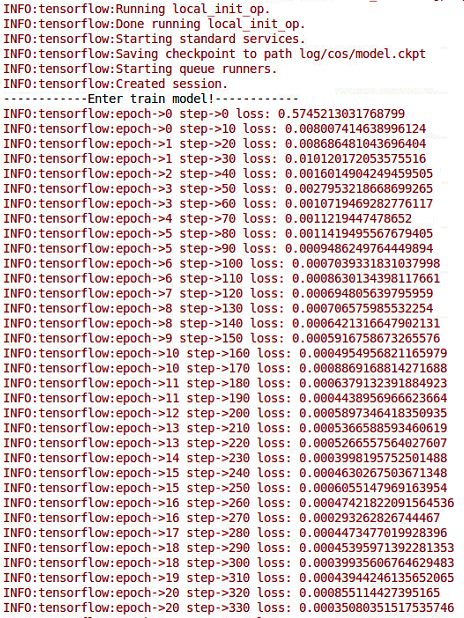
12、创建session对象。



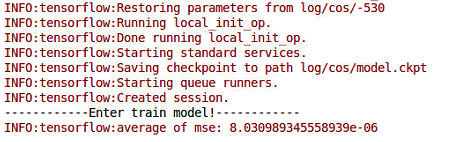
13、定义训练过程和预测过程。主要区别是在sess.run函数中，预测阶段不需要调用rnn\_model.train\_op这个operation，即无需优化损失函数。



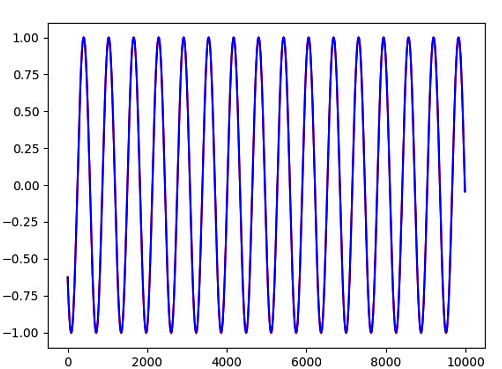
训练过程输出如下图所示，每10次优化输出一次当前损失函数值，当损失函数值不再下降时则应停止训练。



预测阶段输出如下图所示将显示预测结果的平均MSE值。



最后也可以将预测结果可视化。查看预测效果。示范预测结果如下图所示，蓝色线为真实值，红色线为预测值。



为了真切的看到差别，将曲线图放大，如下所示，可以看到虽然预测的很接近，但还是存在一定的偏差。

