LSTM TensorFlow 教程

杨雨辰

2018 年 6 月 1 日

目录

简介

本教程旨在介绍如何使用TensorFlow以及LSTM神经网络进行深度学习（了解更多机器学习请参见教程[1]）。本教程的演示方法为：以实数序列的预测模型为例进行论述，并对股票价格或股指指数进行实测。

1 序列数据

本教程将使用神经网络作为序列数据的预测模型。接下来我们介绍什么是序列数据。

1.1 数学上的实数序列

$\mathbf{v}$为一有限实数序列，即$\mathbf{v}=(v_1,...,v_M)$，其中$\forall i=1,...,M$，$v_i\in\mathbf{R}$为实数，M为一非负整数，亦即$\mathbf{v}$是一包含M个元素的实向量（若$M=0$,则$\mathbf{v}$为一空向量）。

1.2 Python中的序列数据

在Python语言中，我们通常使用numpy库中的array数据结构表示实数数组。对于一个实数序列，可以使用一个一维数组表达。通常原始序列数据在Python中的数据结构并非array，而是诸如tuple、list、pandas.Series等array_like数据结构（详见API文档[2]参考array_like）。在这种条件下，我们可以调用numpy.array构造函数进行数据转换（详见API文档[2]参考numpy.array），例如x=numpy.array(y)，其中输入参数y是一个array_like对象，返回值x则为一个ndarray对象，如果y是序列数据，那么x就是一个一维数组。

2 股价历史数据的处理

显然，某一股票的历史价格是一个实数序列，我们将每日闭市价作为研究对象，形如

$$\mathbf{P}=(P_{-T},...,P_0) \tag{1}$$

其中P_0是即期数据（当日闭市价），P_{-t}是的第t天前的闭市价。但在使用机器学习对股市进行分析的时候，并不建议直接使用股价数据（因为这涉及到神经网络的输入数据的数据结构与输出数据的数值范围）。我们会对股价进行一些处理，这包含两步，首先将股价转换为收益率，然后对收益率进行分组。前者是对数值的改变，后者是对数据结构的改变。

2.1 转义

比起股票的实际价格，我们更加关心股市的收益率，所以我们可以求得股市收益率$\mathbf{r}$,

$$\mathbf{r} = (r_{-T}, ..., r_{-1}) \tag{2}$$

其中，对任意的$i = -T, ..., -1$，有

$$r_i = (P_{i+1} - P_i)/P_i \tag{3}$$

将股价转化为股市收益率并不丢失信息（除了初始价格，但初始价格并不重要，因为我们通常只关心回报率）；而其便利之处在于得到的r_i是带限数据：以中国股市为例，盈亏不能超过10%，所以有$-0.1 \leq r_i \leq 0.1$。这一特点十分有利于后续进行机器学习，因为在神经网络深度学习当中，经常会使用到sigmoid函数和tanh函数作为激励函数，这些激励函数会将输入数据转换为带限数据输出，例如，LSTM神经网络会输出-1到1之间的数据。所以，神经网络无法模拟超出范围的数据，这使得没有固定区间限制的股价数据无法被直接使用。

2.2 划分

我们首先对收益率进行分组，将近期收益率关联起来作为一个整体；然后将各组收益率逐个纳入机器学习，分析长期收益率之间的关系。譬如，我们定义LSTM神经网络有$p > 0$组输入数据，每组数据的长度为$L > 0$，而神经网络最终输出q个数据，注意需满足条件

$$M := pL + q \leq T \tag{4}$$

那么，给定起始位置$-T \leq j \leq -M$，神经网络的输入为

$$\mathbf{x}^{(j)} = (\mathbf{a}_1^{(j)}, ..., \mathbf{a}_p^{(j)}) \tag{5}$$

其中，对任意的$i = 1, ..., p$，有

$$\mathbf{a}_i^{(j)} = (r_{(i-1)L+j}, ..., r_{iL+j-1}) \tag{6}$$

另一方面，神经网络的输出可以对比以下标签

$$\mathbf{y}^{(j)} = (r_{pL+j}, ..., r_{M+j-1}) \tag{7}$$

注意，$\mathbf{y}^{(j)}$是紧接着$\mathbf{x}^{(j)}$之后的股市收益率数据，所以该模型$\mathbf{y}^{(j)} = f\left(\mathbf{x}^{(j)}\right)$旨在预测未来股价。

2.3 训练集与测试集

我们定义机器学习的训练集样本容量为$N_0 > 0$，测试集样本容量为$N_1 > 0$，即使用N_0个样本用于训练，使用N_1个样本进行测试。注意，需满足条件

$$N_0 + N_1 - 2 \leq T - 2M \tag{8}$$

通常而言，为保证测试质量，训练数据和测试数据应该相互独立，所以定义训练集与测试集为

$$S_{train} = \left\{\left\{\mathbf{x}^{(j)}\right\}, \left\{\mathbf{y}^{(j)}\right\}\right\}, \qquad -2M - N_1 - N_0 + 2 \le j \le -2M - N_1 + 1 \tag{9}$$

$$S_{test} = \left\{\left\{\mathbf{x}^{(j)}\right\}, \left\{\mathbf{y}^{(j)}\right\}\right\}, \qquad -M - N_1 + 1 \le j \le -M \tag{10}$$

2.4 TensorFlow中的输入输出数据结构

在使用TensorFlow时，我们需要把神经网络的输入数据$\left\{\mathbf{x}^{(j)}\right\}$表达为形状为$(N, p, L)$的数组，而其最终输出数据$\left\{\mathbf{y}^{(j)}\right\}$则为一个形状为$(N, q)$的数组，其中$N$为的样本容量。我们用关键字batch_size、max_time、input_size、output_size表达RNN神经元输入张量与输出张量的形状：

RNN神经网络输入 形如(batch_size,max_time,input_size)的Tensor对象

RNN神经网络输出 形如(batch_size,output_size)的Tensor对象

3 LSTM神经网络

我们使用TensorFlow建立LSTM神经网络，会调用到以下函数。我们对此作出简介。建立神经网络的详细代码请参见文件LSTMPredict.py[10]。

3.1 BasicLSTMCell and LSTMCell

何为RNN以及LSTM神经网络请参见简明文章[3]。单层LSTM神经网络在TensorFlow当中的被封装为BasicLSTMCell类或者LSTMCell类，调用其构造函数以生成单层神经网络（详细介绍请分别参见API文档[4]和[5]）。

3.2 DropoutWrapper

在机器学习中常会遇见overfitting现象（何为overfitting，过适，过拟合，请参见Wikipedia[6]）。一个常用的解决overfitting的方法就是dropout，亦即在神经网络中随机抛出一些输入数据、神经元或者输出数据，仅仅使用剩余的部分进行模型数据拟合。这样可以有效地排除掉overfitting。在TensorFlow当中，类DropoutWrapper用于在已建成的RNN神经网络上进行一次dropout（详细介绍请参见API文档[7]）。

3.3 MultiRNNCell and dynamic_rnn

为了实现深度学习，建立多层RNN神经网络，在TensorFlow当中，类MultiRNNCell用于合并多层神经网络（详细介绍请参见API文档[8]）。对于多层RNN网络，在TensorFlow当中可以调用dynamic_rnn函

数，来进行由第一层网络的输入到最后一层网络的输出的运算，这一功能使用方便，代码简洁（详细介绍请参见API文档[9]）。

4 LSTMPredict类

本教程创建LSTMPredict类以便对实数序列数据进行预测。该类实现四个方法：__init__、_train、_test、_pred。__init__构造函数用来生成类对象；_train函数接受训练集数据，用以对神经网络的参数进行训练；_test函数接受测试集数据，用以对模型性能进行评估；_pred接受某股价或某股指收益率数据，对下一阶段收益率进行预测。本人编写了实现该类的Python代码存放于文件LSTMPredict.py，见于[10]。以下对该类的四个方法作出说明。

4.1 __init__

函数：__init__(graph, max_time, input_size, layers_units, state_keep_probs, learning_rate)
用途：初始化LSTMPredict类的对象。
参数:

- graph: tf.Graph object. 神经网络所在的图。
- max_time: int. 神经网络输入数据的组数。
- input_size: int. 每组输入数据的个数。
- layers_units: a sequence of ints. 各层神经层的神经元个数，最后一层神经元个数等于输出数据个数。
- state_keep_probs: a sequence of floats. 各层神经层的神经元保留率（介于0和1之间），亦即在训练时会随机保留部分神经元而去除掉剩余的神经元。
- learning_rate: float. 训练算法的学习效率（介于0和1之间）。

4.2 _train

函数：loss, outputs, labels = _train(data)
用途：训练模型。
参数：

- data: a sequence of floats. 用作训练的历史股票回报率（介于-1到1之间）。

返回值：

- loss: float. 当次训练前的模型误差。
- outputs: (batch_size,output_size) array. 当次训练前模型的输出值。
- labels: (batch_size,output_size) array. 训练标签。

4.3 _test

函数：loss, outputs, labels = _test(data)
用途：测试模型。
参数：

- data: a sequence of floats. 用作测试的历史股票回报率（介于-1到1之间）。

返回值：

- loss: float. 测试模型误差。
- outputs: (batch_size,output_size) array. 测试模型的输出值。
- labels: (batch_size,output_size) array. 真实观测值。

4.4 _pred

函数：outputs = _pred(data)
用途：给出历史股票回报率，预测下一阶段回报率。
参数：

- data: a sequence of floats. 历史股票回报率（介于-1到1之间）。

返回值：

- outputs: (output_size) array. 模型预测回报率。

5 数据实测

我们以道琼斯工业平均指数为例进行测试。数据下载已下载到文件DJI.csv[10]。测试代码请见文件test.py[10]。结果如图1-4。我们发现常规的LSTM并不能准确预测股市回报率，其结果随机性很大。预测的总体趋势时而于真实值上方，时而于真实值下方。预测的波动率也不能反应真实波动。这主要由于股票数据噪音很大，同时也不具有显著的历史相关性。综上，常规的LSTM网络并不能给出足够支撑投资决策的结果。

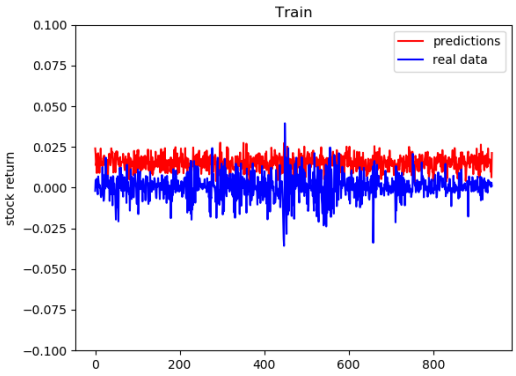

图 1: 第一次实验训练阶段

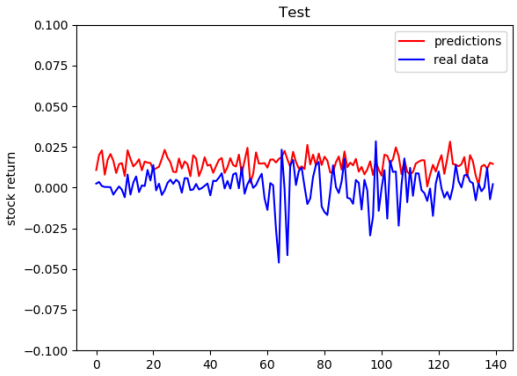

图 2: 第一次实验测试阶段

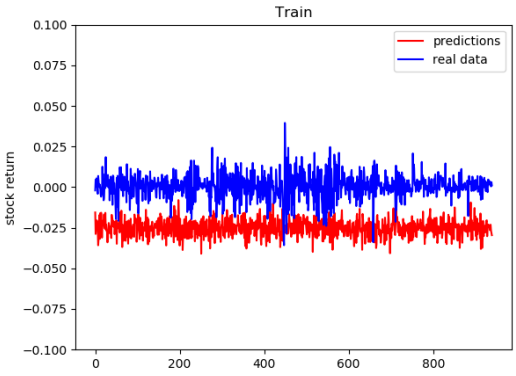

图 3: 第二次实验训练阶段

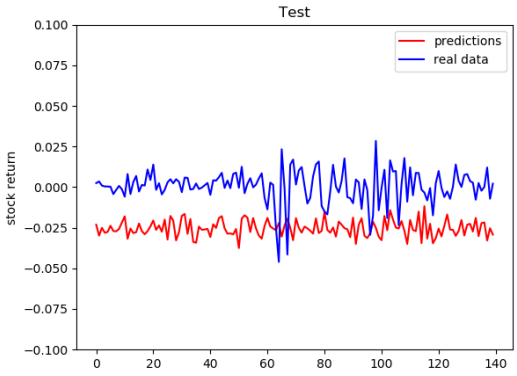

图 4: 第二次实验测试阶段

参考文献

[1] https://developers.google.com/machine-learning/crash-course/

[2] https://docs.scipy.org/doc/numpy/reference/generated/numpy.array.html

[3] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[4] https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/BasicLSTMCell

[5] https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/LSTMCell

[6] https://en.wikipedia.org/wiki/Overfitting

[7] https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/DropoutWrapper

[8] https://www.tensorflow.org/api_docs/python/tf/contrib/rnn/MultiRNNCell

[9] https://www.tensorflow.org/api_docs/python/tf/nn/dynamic_rnn

[10] https://github.com/Perry2018/LSTM