

RNN教程之-2 LSTM实战



Nicholas_Jela (/u/a24acc5449c7) [+ 关注](#)

2016.10.25 10:15* 字数 1942 阅读 4941 评论 42 喜欢 25 赞赏 1

(/u/a24acc5449c7)

本文由清华大学硕士大神金天撰写，欢迎大家转载，不过请保留这段版权信息，对本文内容有疑问欢迎联系作者微信：jintianiloveu 探讨，多谢合作~

UPDATE 2017-4-11

这篇是之前写的文章，关于时间序列的更新版本在这里 (<http://www.jianshu.com/p/fbd6d3c1dc21>), 稍后会开源所有代码。

前言

说出来你们不敢相信，刚才码了半天的字，一个侧滑妈的全没了，都怪这Mac的触摸板太敏感沃日。好吧，不浪费时间了，前言一般都是废话，这个教程要解决的是一个LSTM的实战问题，很多人问我RNN是啥，有什么卵用，你可以看看我之前写的博客可以入门，但是如果你想实际操作代码，那么慢慢看这篇文章。本文章所有代码和数据集在我的Github Repository (https://github.com/jinfagang/LSTM_learn)下载。

问题

给你一个数据集，只有一列数据，这是一个关于时间序列的数据，从这个时间序列中预测未来一年某航空公司的客运流量。

首先我们数据预览一下，用pandas读取数据，这里我们只需要使用后一列真实数据，如果你下载了数据，数据大概长这样：

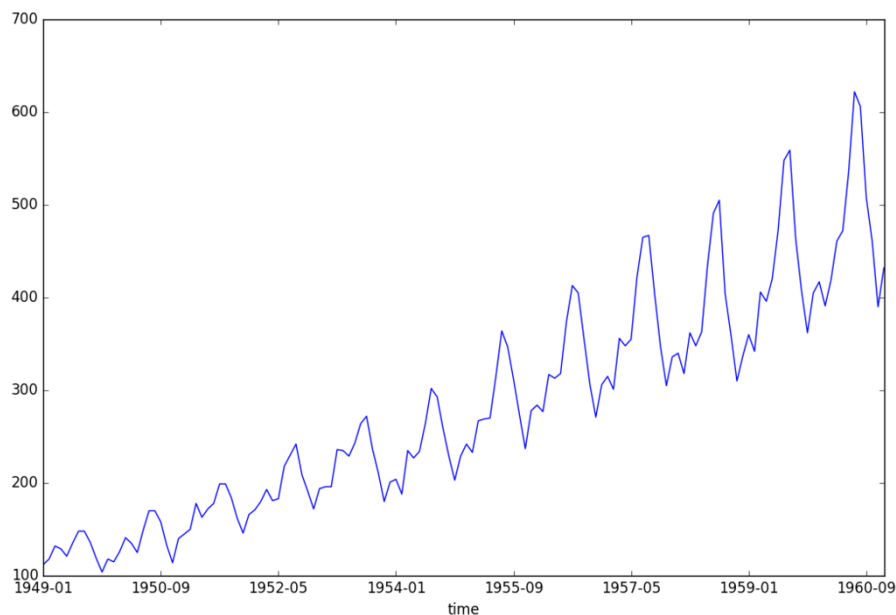
	time	passengers
0	1949-01	112
1	1949-02	118
2	1949-03	132
3	1949-04	129
4	1949-05	121
5	1949-06	135
6	1949-07	148
7	1949-08	148
8	1949-09	136
9	1949-10	119
...

第一列是时间，第二列是客流量，为了看出这个我们要预测的客流量随时间的变化趋势，本大神教大家如何把趋势图画出来，接下来就非常牛逼了。用下面的代码来画图：

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv('international-airline-passengers.csv', sep=',')
df = df.set_index('time')
df['passengers'].plot()
plt.show()
```

这时候我们可以看到如下的趋势图：



figure_1.png

可以看出，我们的数据存在一定的周期性，这个周期性并不是一个重复出现某个值，而是趋势的增长过程有一定的规律性，这个我们肉眼就能看得出来，但是实际上计算机要识别这种规律就有一定的难度了，这时候就需要使用我们的LSTM大法。
好的，数据已经预览完了，接下来我们得思考一下怎么预测，怎么把数据处理为LSTM网络需要的格式。

LSTM数据预处理

这个过程非常重要，这也是很多水平不高的博客或者文章中没有具体阐述而导致普通读者不知道毛意思的过程，其实我可以这样简单的叙述，LSTM你不要以为各种时间序列搞的晕头转向，其实本质它还是神经网络，与普通的神经网络没有任何区别。我们接下来就用几行小代码把数据处理为我们需要的类似于神经网络输入的二维数据。

首先我们确确实实需要的只是一列数据：

```
df = pd.read_csv(file_name, sep=',', usecols=[1])
data_all = np.array(df).astype(float)
print(data_all)
```

输出是：

```
[[ 112.]
 [ 118.]
 [ 132.]
 [ 129.]
 [ 121.]
 [ 135.]
 [ 148.]
 [ 148.]
 [ 136.]
 [ 119.]
 [ 104.]
 [ 118.]
 [ 115.]
 ....
 ]
```

非常好，现在我们已经把我们需要的数据抠出来了，继续上面处理：

```
data = []
for i in range(len(data_all) - sequence_length - 1):
    data.append(data_all[i: i + sequence_length + 1])
reshaped_data = np.array(data).astype('float64')
print(reshaped_data)
```

这时候你会发现好像结果看不懂，不知道是什么数据，如果你data_all处理时加ravel() (



用来把数据最里面的中括号去掉)，即：

```
df = pd.read_csv(file_name, sep=',', usecols=[1])
data_all = np.array(df).ravel().astype(float)
print(data_all)
```

那么数据输出一目了然：

```
[ 112.  118.  132. ...,  136.  119.  104.]
[ 118.  132.  129. ...,  119.  104.  118.]
[ 132.  129.  121. ...,  104.  118.  115.]
...,
[ 362.  405.  417. ...,  622.  606.  508.]
[ 405.  417.  391. ...,  606.  508.  461.]
[ 417.  391.  419. ...,  508.  461.  390.]]
```

是的，没有错！一列数据经过我们这样不处理就可以作为LSTM网络的输入数据了，而且和神经网络没有什么两样！！牛逼吧？牛逼快去哥的Github Repo给个star，喊你们寝室的菜市场的大爷大妈都来赞！越多越好，快，哥的大牛之路就靠你们了！然而这还是只是开始。。接下来要做的就是将数据切分为训练集和测试集：

```
split = 0.8
np.random.shuffle(reshaped_data)
x = reshaped_data[:, :-1]
y = reshaped_data[:, -1]
split_boundary = int(reshaped_data.shape[0] * split)
train_x = x[:split_boundary]
test_x = x[split_boundary:]

train_y = y[:split_boundary]
test_y = y[split_boundary:]
```

这些步骤相信聪明的你一点看得懂，我就不多废话了，我要说明的几点是，你运行时直接运行Github上的脚本代码，如果报错请私信我微信 jintianiloveu，我在代码中把过程包装成了函数所以文章中的代码可能不太一样。在实际代码中数据是需要归一化的，这个你应该知道，如何归一化代码中也有。

搭建LSTM模型

好，接下来是最牛逼的部分，也是本文章的核心内容（但实际内容并不多），数据有了，我们就得研究研究LSTM这个东东，不管理论上吹得多么牛逼，我只看它能不能解决问题，不管黑猫白猫，能抓到老鼠的就是好猫，像我们这样不搞伪学术注重经济效益的商人来说，这点尤为重要。搭建LSTM模型，我比较推荐使用keras，快速简单高效，分分钟，但是牺牲的是灵活性，不过话又说回来，真正的灵活性也是可以发挥的，只是要修改底层的东西那就有点麻烦了，我们反正是用它来解决问题的，更基础的部分我们就不



研究了，以后有时间再慢慢深入。

在keras 的官方文档中，说了LSTM是整个Recurrent层实现的一个具体类，它需要的输入数据维度是：

形如 (samples, timesteps, input_dim) 的3D张量

发现没有，我们上面处理完了数据的格式就是 (samples, timesteps) 这个time_step是我们采用的时间窗口，把一个时间序列当成一条长链，我们固定一个一定长度的窗口对这个长链进行采用，最终就得到上面的那个二维数据，那么我们缺少的是input_dim这个维度，实际上这个input_dim就是我们的那一列数据的数据，我们现在处理的是一列也有可能是很多列，一系列与时间有关的数据需要我们去预测，或者文本处理中会遇到。我们先不管那么多，先把数据处理为LSTM需要的格式：

```
train_x = np.reshape(train_x, (train_x.shape[0], train_x.shape[1], 1))
test_x = np.reshape(test_x, (test_x.shape[0], test_x.shape[1], 1))
```

好的，这时候数据就是我们需要的啦。接下来搭建模型：

```
# input_dim是输入的train_x的最后一个维度，train_x的维度为(n_samples, time_steps, input_dim)
model = Sequential()
model.add(LSTM(input_dim=1, output_dim=50, return_sequences=True))
model.add(LSTM(100, return_sequences=False))
model.add(Dense(output_dim=1))
model.add(Activation('linear'))
model.compile(loss='mse', optimizer='rmsprop')
```

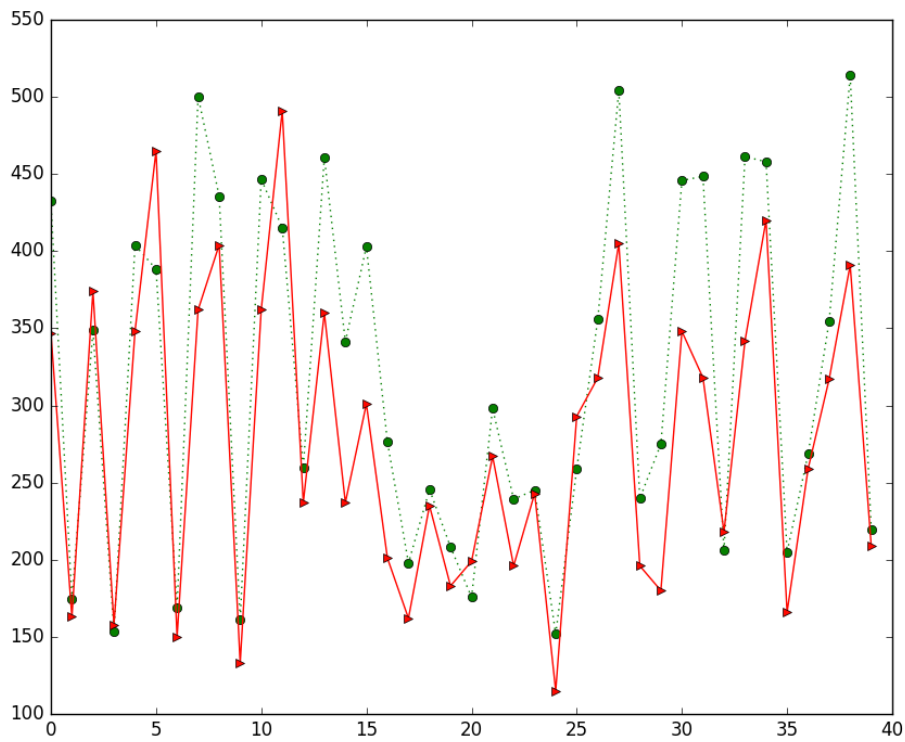
看到没，这个LSTM非常简单！！甚至跟输入的数据格式没有任何关系，只要输入数据的维度是1，就不需要修改模型的任何参数就可以把数据输入进去进行训练！

我们这里使用了两个LSTM进行叠加，第二个LSTM第一个参数指的是输入的维度，这和第一个LSTM的输出维度并不一样，这也是LSTM比较“随意”的地方。最后一层采用了线性层。

结果

预测的结果如下图所示：






result.png

这个结果还是非常牛逼啊，要知道我们的数据是打乱过得噢，也就是说泛化能力非常不错，厉害了word LSTM！
筒子们，本系列教程到此结束，欢迎再次登录老司机的飞船。。。如果有不懂的私信我，想引起我的注意快去Github上给我star！！

系列文章结尾安利：Python深度学习基地群 216912253 一个即谈理想又谈技术的技术人聚集地，欢迎加入。

RNN系列 (nb/4972171) [举报文章](#) [© 著作权归作者所有](#)



Nicholas_Jela (u/a24acc5449c7)

写了 30143 字，被 242 人关注，获得了 82 个喜欢 (u/a24acc5449c7)

+ 关注


人工智能爱好者，清华大学控制科学与工程硕士，深度学习领域内的一位大神级人物，魔术师，个人主页...

^

🔗

♥ 喜欢 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-like-button) | 2







更多分享

被以下专题收入，发现更多相似内容

- 深度学习·神经... (/c/71b2d8a98305?utm_source=desktop&utm_medium=notes-included-collection) /images /6531238 /weibo
- NLP (/c/423c379fdbd8?utm_source=desktop&utm_medium=notes-included-collection) /image_de2de6d83e88.jpg
- LSTM (/c/ef46838f7c61?utm_source=desktop&utm_medium=notes-included-collection)
- 训练调参 (/c/7b98c1b69fb3?utm_source=desktop&utm_medium=notes-included-collection)
- 机器学习 (/c/7e8dddb61c49?utm_source=desktop&utm_medium=notes-included-collection)

^

