
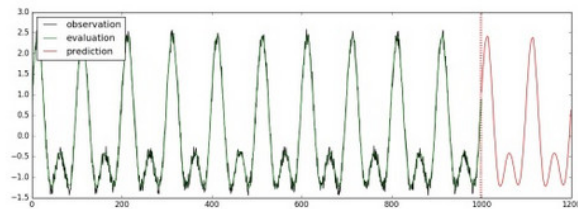


[首页](#) / [社区](#)[/ 如何优雅地用TensorFlow预测时间序列：TFTS库详细教程](#)

如何优雅地用TensorFlow预测时间序列：TFTS库详细教程

 lazyostrich 发布于1月前 阅读209次

❤️ 6 人点赞 💬 0 条评论



前言

如何用TensorFlow结合LSTM来做时间序列预测其实是一个很老的话题，然而却一直没有得到比较好的解决。如果在Github上搜索“tensorflow time series”，会发现star数最高的 [tgjeon/TensorFlow-Tutorials-for-Time-Series](#) 已经和TF 1.0版本不兼容了，并且其他的项目使用的方法也各有不同，比较混乱。

在刚刚发布的TensorFlow 1.3版本中，引入了一个TensorFlow Time Series模块（源码地址为：[tensorflow/tensorflow](#)，以下简称为TFTS）。TFTS专门设计了一套针对时间序列预测问题的API，目前提供AR、Anomaly Mixture AR、LSTM三种预测模型。

由于是刚刚发布的库，文档还是比较缺乏的，我通过研究源码，大体搞清楚了这个库的设计逻辑和使用方法，这篇文章是一篇教程帖，会详细的介绍TFTS库的



lazyostrich

❤️ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

[我要点评](#)

被 985人关注，获得了147个喜欢

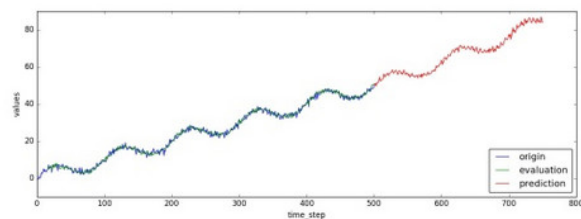
[+关注](#)[主题目录](#) [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结

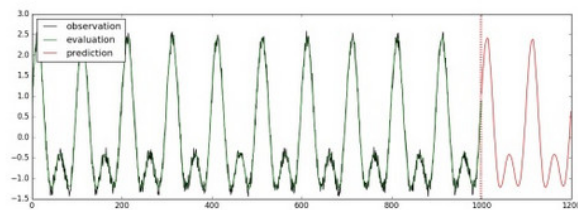
以下几个功能：

- 读入时间序列数据（分为从numpy数组和csv文件两种方式）
- 用AR模型对时间序列进行预测
- 用LSTM模型对时间序列进行预测（包含单变量和多变量）

先上效果图，使用AR模型预测的效果如下图所示，蓝色线是训练数据，绿色为模型拟合数据，红色线为预测值：



使用LSTM进行单变量时间序列预测：



使用LSTM进行多变量时间序列预测（每一条线代表一个变量）：



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

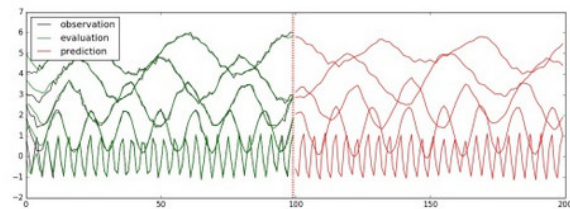
我要点评



📖 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结





文中涉及的所有代码已经保存在Github上了，地址是：[hzy46/TensorFlow-Time-Series-Examples](#)，以下提到的所有代码和文件都是相对于这个项目的根目录来说的。

时间序列问题的一般形式

一般地，时间序列数据可以看做由两部分组成：**观察的时间点**和**观察到的值**。以商品价格为例，某年一月的价格为120元，二月的价格为130元，三月的价格为135元，四月的价格为132元。那么观察的时间点可以看做是1,2,3,4，而在各时间点上观察到的数据的值为120,130,135,132。

从Numpy数组中读入时间序列数据

如何将这样的时间序列数据读入进来？TFTS库中提供了两个方便的读取器NumpyReader和CSVReader。前者用于从Numpy数组中读入数据，后者则可以从CSV文件中读取数据。

我们利用np.sin，生成一个实验用的时间序列数据，这个时间序列数据实际上就是在正弦曲线上加上了上升的趋势和一些随机的噪声：



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



☰ 主题目录 [只显示目录](#)

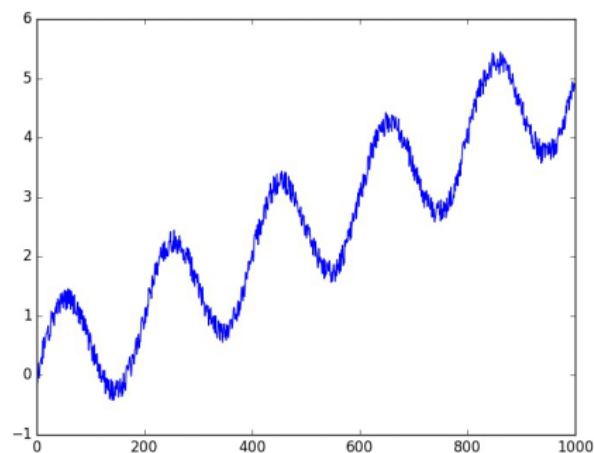
- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



```
# coding: utf-8
from __future__ import print_function
import numpy as np
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.contrib.timeseries.python.ti

x = np.array(range(1000))
noise = np.random.uniform(-0.2, 0.2, 1000)
y = np.sin(np.pi * x / 100) + x / 200. + noise
plt.plot(x, y)
plt.savefig('timeseries_y.jpg')
```

如图：



横坐标对应变量“x”，纵坐标对应变量“y”，它们就是我们之前提到过的“观察的时间点”以及“观察到的值”。TFTS读入x和y的方式非常简单，请看下面的代码：



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



📑 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



```
data = {
    tf.contrib.timeseries.TrainEvalFeatures.TIMES: x,
    tf.contrib.timeseries.TrainEvalFeatures.VALUES: y
}

reader = NumpyReader(data)
```

我们首先把x和y变成python中的词典（变量data）。

变量data中的键值

tf.contrib.timeseries.TrainEvalFeatures.TIMES实际就是一个字符串“times”，而

tf.contrib.timeseries.TrainEvalFeatures.VALUES就是字符串“values”。所以上面的定义直接写成“data = {'times':x, 'values':y}”也是可以的。写成比较复杂的形式是为了和源码中的写法保持一致。

得到的reader有一个read_full()方法，它的返回值就是时间序列对应的Tensor，我们可以用下面的代码试验一下：

```
with tf.Session() as sess:
    full_data = reader.read_full()
    # 调用read_full方法会生成读取队列
    # 要用tf.train.start_queue_runners启动队列
    coord = tf.train.Coordinator()
    threads = tf.train.start_queue_runners(sess)
    print(sess.run(full_data))
    coord.request_stop()
```



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



≡ 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



不能直接使用`sess.run(reader.read_full())`来从`reader`中取出所有数据。原因在于`read_full()`方法会产生读取队列，而队列的线程此时还没启动，我们需要使用`tf.train.start_queue_runners`启动队列，才能使用`sess.run()`来获取值。

我们在训练时，通常不会使用整个数据集进行训练，而是采用`batch`的形式。从`reader`出发，建立`batch`数据的方法也很简单：

```
train_input_fn = tf.contrib.timeseries.RandomWindowInputFn(reader, batch_size=2, window_size=10)
```

`tf.contrib.timeseries.RandomWindowInputFn`会在`reader`的所有数据中，随机选取窗口长度为`window_size`的序列，并包装成`batch_size`大小的`batch`数据。换句话说，一个`batch`内共有`batch_size`个序列，每个序列的长度为`window_size`。

以`batch_size=2, window_size=10`为例，我们可以打出一个`batch`内的数据：

```
with tf.Session() as sess:
    batch_data = train_input_fn.create_batch(
        coord = tf.train.Coordinator()
        threads = tf.train.start_queue_runners(sess)
        one_batch = sess.run(batch_data[0])
        coord.request_stop()

    print('one_batch_data:', one_batch)
```



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



☰ 主题目录 只显示目录

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- 使用LSTM预测多变量时间序列
- 总结



这部分读入代码的地址在 https://github.com/hzy46/Tensor-Flow-Time-Series-Examples/blob/master/test_input_array.py 。

从CSV文件中读入时间序列数据

有的时候，时间序列数据是存在CSV文件中的。我们当然可以将其先读入为Numpy数组，再使用之前的方法处理。更方便的做法是使用 `tf.contrib.timeseries.CSVReader` 读入。项目中提供了一个 `test_input_csv.py` 代码，示例如何将文件 `./data/period_trend.csv` 中的时间序列读入进来。

假设CSV文件的时间序列数据形式为：

```
1, -0.6656603714
2, -0.1164380359
3, 0.7398626488
4, 0.7368633029
5, 0.2289480898
6, 2.257073255
7, 3.023457405
8, 2.481161007
9, 3.773638612
10, 5.059257738
11, 3.553186083
```

CSV文件的第一列为时间点，第二列为该时间点上观察到的值。将其读入的方法为：



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



≡ 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



```
# coding: utf-8
from __future__ import print_function
import tensorflow as tf

csv_file_name = './data/period_trend.csv'
reader = tf.contrib.timeseries.CSVReader(csv
```

从reader建立batch数据形成train_input_fn的方法和之前完全一样。下面我们就利用这个train_input_fn来训练模型。

使用AR模型预测时间序列

自回归模型（Autoregressive model，可以简称为AR模型）是统计学上处理时间序列模型的基本方法之一。在TFTS中，已经实现了一个自回归模型。使用AR模型训练、验证并进行时间序列预测的示例程序为train_array.py。

先建立一个train_input_fn：



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



≡ 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结




```

x = np.array(range(1000))
noise = np.random.uniform(-0.2, 0.2, 1000)
y = np.sin(np.pi * x / 100) + x / 200. + noise
plt.plot(x, y)
plt.savefig('timeseries_y.jpg')

data = {
    tf.contrib.timeseries.TrainEvalFeatures.TRAIN_FEATURES: x,
    tf.contrib.timeseries.TrainEvalFeatures.VALIDATION_FEATURES: y
}

reader = NumpyReader(data)

train_input_fn = tf.contrib.timeseries.RandomBatchReader(
    reader, batch_size=16, window_size=40)

```

针对这个序列，对应的AR模型的定义就是：

```

ar = tf.contrib.timeseries.ARRegressor(
    periodicities=200, input_window_size=30,
    num_features=1,
    loss=tf.contrib.timeseries.ARModel.NORMAL
)

```

这里的几个参数比较重要，分别给出解释。第一个参数periodicities表示序列的规律性周期。我们在定义数据时使用的语句是：“ $y = \sin(\pi * x / 100) + x / 200. + \text{noise}$ ”，因此周期为200。input_window_size表示模型每次输入的值，output_window_size表示模型每次输出的值。input_window_size和output_window_size加起来必须等于train_input_fn中总的window_size。在这里，我们总的window_size为40，input_window_size为30，



♡ 6人点赞

🔖 收藏

共有0 条评论，1人收藏

我要点评



主题目录 只显示目录

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



output_window_size为10，也就是说，一个batch内每个序列的长度为40，其中前30个数被当作模型的输入值，后面10个数为这些输入对应的目标输出值。最后一个参数loss指定采取哪一种损失，一共有两种损失可以选择，分别是NORMAL_LIKELIHOOD_LOSS和SQUARED_LOSS。

num_features参数表示在一个时间点上观察到的数的维度。我们这里每一步都是一个单独的值，所以num_features=1。

除了程序中出现的几个参数外，还有一个比较重要的参数是model_dir。它表示模型训练好后保存的地址，如果不指定的话，就会随机分配一个临时地址。

使用变量ar的train方法可以直接进行训练：

```
ar.train(input_fn=train_input_fn, steps=6000)
```

TFTS中验证(evaluation)的含义是：使用训练好的模型在原先的训练集上进行计算，由此我们可以观察到模型的拟合效果，对应的程序段是：

```
evaluation_input_fn = tf.contrib.timeseries.  
evaluation = ar.evaluate(input_fn=evaluation
```

如果要理解这里的逻辑，首先要理解之前定义的AR模型：它每次都接收一个长度为30的输入观测序列，并输出长度为10的预测序列。整个训练集是一个长度为1000的序列，前30个数首先被当作“初始观测序列”输入到模型中，由此就可以计算出下面10步的预测值。



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



≡ 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



接着又会取30个数进行预测，这30个数中有10个数就是前一步的预测值，新得到的预测值又会变成下一步的输入，以此类推。

最终我们得到970个预测值（970=1000-30，因为前30个数是没办法进行预测的）。这970个预测值就被记录在evaluation['mean']中。evaluation还有其他几个键值，如evaluation['loss']表示总的损失，evaluation['times']表示evaluation['mean']对应的时间点等等。

evaluation['start_tuple']会被用于之后的预测中，它相当于最后30步的输出值和对应的时间点。以此为起点，我们可以对1000步以后的值进行预测，对应的代码为：

```
(predictions,) = tuple(ar.predict(
    input_fn=tf.contrib.timeseries.predict_co
    evaluation, steps=250)))
```

这里的代码在1000步之后又预测了250个时间点。对应的值就保存在predictions['mean']中。我们可以把观测到的值、模型拟合的值、预测值用下面的代码画出来：



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



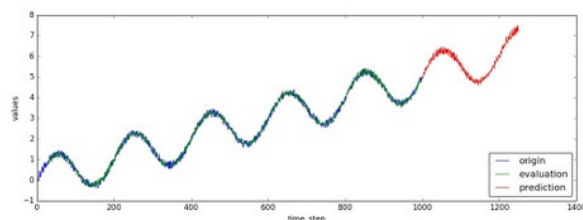
≡ 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



```
plt.figure(figsize=(15, 5))
plt.plot(data['times'].reshape(-1), data['values'], label='origin')
plt.plot(evaluation['times'].reshape(-1), evaluation['values'], label='evaluation')
plt.plot(predictions['times'].reshape(-1), predictions['values'], label='prediction')
plt.xlabel('time_step')
plt.ylabel('values')
plt.legend(loc=4)
plt.savefig('predict_result.jpg')
```

画好的图片会被保存为“predict_result.jpg”



使用LSTM预测单变量时间序列

注意：以下LSTM模型的例子必须使用TensorFlow最新的开发版的源码。具体来说，要保证“from tensorflow.contrib.timeseries.python.timeseries.estimators import TimeSeriesRegressor”可以成功执行。

给出两个用LSTM预测时间序列模型的例子，分别是 train_lstm.py 和 train_lstm_multivariate.py。前者是在LSTM中进行单变量的时间序列预测，后者是使用LSTM进行多变量时间序列预测。为了使用LSTM模型，我们需要先使用TFTS库对其进行定义，定义模型的代码来源于 TFTS的示例源码，在train_lstm.py和train_lstm_multivariate.py中分别拷贝了一份。



♡ 6人点赞

🔖 收藏

共有0 条评论，1人收藏

我要点评



📖 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结



我们同样用函数加噪声的方法生成一个模拟的时间序列数据：

```
x = np.array(range(1000))
noise = np.random.uniform(-0.2, 0.2, 1000)
y = np.sin(np.pi * x / 50) + np.cos(np.pi * x / 100) + noise

data = {
    tf.contrib.timeseries.TrainEvalFeatures.TRAIN_FEATURES: x,
    tf.contrib.timeseries.TrainEvalFeatures.VALIDATION_FEATURES: y
}

reader = NumpyReader(data)

train_input_fn = tf.contrib.timeseries.RandomWindowInputFn(
    reader, batch_size=4, window_size=100)
```

此处y对x的函数关系比之前复杂，因此更适合用LSTM这样的模型找出其中的规律。得到y和x后，使用NumpyReader读入为Tensor形式，接着用tf.contrib.timeseries.RandomWindowInputFn将其变为batch训练数据。一个batch中有4个随机选取的序列，每个序列的长度为100。

接下来我们定义一个LSTM模型：

```
estimator = ts_estimators.TimeSeriesRegressor(
    model=_LSTMModel(num_features=1, num_units=100),
    optimizer=tf.train.AdamOptimizer(0.001))
```



♡ 6人点赞

🔖 收藏

共有0 条评论，1人收藏

我要点评



📖 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结

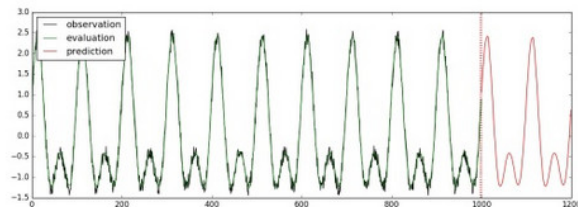


num_features = 1表示单变量时间序列，即每个时间点上观察到的量只是一个单独的数值。
num_units=128表示使用隐层为128大小的LSTM模型。

训练、验证和预测的方法都和之前类似。在训练时，我们在已有的1000步的观察量的基础上向后预测200步：

```
estimator.train(input_fn=train_input_fn,
                 evaluation_input_fn = tf.contrib.timeseries.
                 evaluation = estimator.evaluate(input_fn=eva
# Predict starting after the evaluation
(predictions,) = tuple(estimator.predict(
    input_fn=tf.contrib.timeseries.predict_co
    evaluation, steps=200)))
```

将验证、预测的结果取出并画成示意图，画出的图像会保存成“predict_result.jpg”文件：



使用LSTM预测多变量时间序列

所谓多变量时间序列，就是指在每个时间点上的观测量有多个值。在data/multivariate_periods.csv文件中，保存了一个多变量时间序列的数据：



♡ 6人点赞

🔖 收藏

共有0 条点评，1人收藏

我要点评



📖 主题目录 [只显示目录](#)

- 前言
- 时间序列问题的一般形式
- 从Numpy数组中读入时间序列数据
- 从CSV文件中读入时间序列数据
- 使用AR模型预测时间序列
- 使用LSTM预测单变量时间序列
- [使用LSTM预测多变量时间序列](#)
- 总结

