

forecast exchange rate

2019 年 1 月 15 日

1 Forecast Exchange Rate

杨宇昊

github 仓库 <https://github.com/QSCTech-Sange/Forecast-exchange-rate> 所有数据和过程都在上面，可以 replicate。原始提交作业的时候选取时间范围不慎，包含了 2015 年的数据，但是其中有汇改，导致数据波动，应该视为 outlier。这次 presentation 中使用的日期为 2016-2018 年。

1.1 预测原则

汇率的预测是有多个方面多个因子来影响的。但是这次小 Project 中，想测试单纯使用数学上的时间序列模型来进行预测。也即，只使用时间和历史汇率进行预测。显然这样的方法不够完善，但是也足够探讨目前最前沿的时间序列模型对于现实的拟合程度，并为大 Project 中的完整预测进行相应的铺垫。

1.2 数据来源

数据来自于 Csmar 国泰安的数据库，下载为 .txt 的格式，这次筛选了从 2015-11-30 到 2018-11-30 的完整数据。（所以也是最后一天才交的原因）将编码改为 UTF-8 以便使用 Pandas 导入。

1.3 模型依据

本次汇率预测采用的是业界最为先进和智能的来自 Facebook 的 Prophet 框架。它是一个时间序列的分析框架。时间序列预测对大部分公司而言都存在必要的需求，比如电商预测 GMV，外卖 O2O 预测成交量波动以便于运力分配，酒店预测间夜量来调整定价与销售，等等。但通常而言，时间序列预测对不少公司来说是一个难题。主要原因除了时间序列预测本身就是玄学之外，还要求分析师同时具备深厚的专业领域知识 (domain knowledge) 和时间序列建模的统计学知识。此外，时间序列模型的调参也是一个比较复杂而繁琐的工作。### 模型结构 Prophet 的本质是一个加法模型，基本形式如下：

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

其中 $g(t)$ 是趋势项， $s(t)$ 是周期项， $h(t)$ 是节假日项， ϵ_t 是误差项并且服从正态分布。### 趋势模型 Prophet 里使用了两种趋势模型：饱和增长模型 (saturating growth model) 和分段线性模型 (piecewise

linear model)。两种模型都包含了不同程度的假设和一些调节光滑度的参数，并通过选择变化点 (changepoints) 来预测趋势变化。最终形式：#### 饱和增长模型

$$g(t) = \frac{C(t)}{1 + \exp(-(k + a(t)^T \delta)(t - (m + a(t)^T \gamma)))}$$

分段线性模型

$$g(t) = (k + a(t)^T \delta)t + (m + a(t)^T \gamma)$$

周期模型 prophet 用傅里叶级数 (Fourier series) 来建立周期模型：

$$s(t) = \sum_{n=1}^N (a_n \cos(\frac{2\pi nt}{P}) + b_n \sin(\frac{2\pi nt}{P}))$$

对 N 的调节起到了低通滤波 (low-pass filter) 的作用。作者说对于年周期与星期周期，N 分别选取为 10 和 3 的效果比较好。

1.3.1 节假日与突发事件模型

节假日需要用户事先指定，每一个节假日都包含其前后的若干天。模型形式如下（就是一个虚拟变量）：

$$Z(t) = [1(t \in D_1), \dots, 1(t \in D_L)]$$

$$h(t) = Z(t)\kappa$$

$$\kappa \sim \text{Normal}(0, v^2)$$

1.4 模型实战

1.4.1 平台搭建

操作系统为 Arch Linux x64 (Windows 下 Prophet 无法正常工作)。环境为 Anaconda。语言使用 Python 3。通过添加 code-forge 的清华源并使用 `conda install fbprophet` 来安装 Prophet 框架。利用到了 `numpy` 和 `pandas` 来处理数据和矩阵。

1.4.2 导入框架

导入 Prophet 框架和 Pandas 框架。`add_changepoints_to_plot` 是为了调整变点先验尺度 (Changepoint Prior Scale)

```
In [1]: import pandas as pd
import numpy as np
import math
from fbprophet import Prophet
from fbprophet.plot import add_changepoints_to_plot
```

1.4.3 导入数据

使用 Pandas 读取汇率历史文件，保存到 df 对象当中（表示 dataframe 的意思）将变量名调整为 ds 和 y(时间变量和预测变量)，以便建模。将数据取对数处理，以使得预测尽可能贴近现实中的变动规律。最后一行展示导入的数据的前几行，检验是否导入成功。

```
In [2]: df = pd.read_table('TRD_Exchange.txt')
        df.columns=['ds','y']
        df['y']=np.log(df['y'])
        df.head()
```

```
Out[2]:
```

	ds	y
0	2018-11-29	1.936624
1	2018-11-28	1.938742
2	2018-11-27	1.938209
3	2018-11-26	1.938065
4	2018-11-23	1.935946

1.4.4 建模调参

接下来开始建模以及调参。m 是一个 Prophet 类型，来拟合刚刚的数据 df。经过大量测试，发现变点先验尺度为 0.5 最为合适。同时，设定天和星期的周期变化为 True。变点先验尺度默认是 0.05。变点先验尺度的直观含义是，对于每个拐点给予的权重。（可以通过二阶导判断拐点）。变点先验尺度过低会造成欠拟合的现象，过高会造成过拟合的现象。因为过低会不把拐点当拐点，会使得序列过于平稳，而过高则会把普通点当做拐点，序列波动非常剧烈。对于汇率，股票这样本身就变动剧烈的时间序列数据而言，倾向于增大变点先验尺度。至于为什么是 0.7 而不是 0.1,0.8,0.6，这通常是一个工程问题而不是经济问题或者力量问题，因为这就是通过大量的尝试来判断拟合的是否合适。设定天和星期的周期变化为 True 也是如此来判断的。

```
In [4]: m = Prophet(changepoint_prior_scale=0.7,weekly_seasonality=True,daily_seasonality=False)
        m.fit(df)
```

```
Out[4]: <fbprophet.forecaster.Prophet at 0x7fc74fb1ab00>
```

1.4.5 预测准备

future 是一个从 m 衍生的 Prophet 类型，设定 m 之后的 32 天的数据（也就是到 1 月 1 日）。cap 是一个设定的心理预期。因为在中美汇率当中，7 是一个很明显的心理上限。值得注意的是，并不是设定了 7.0 的心理上限，数据就一定不会突破 7.0。只是在接近 7.0 的时候，更倾向于下跌而不是上升。事实上，当我换一个时间区间或者或一个变点先验尺度，即便开启了 7.0 的上限，也是会轻松突破它的。

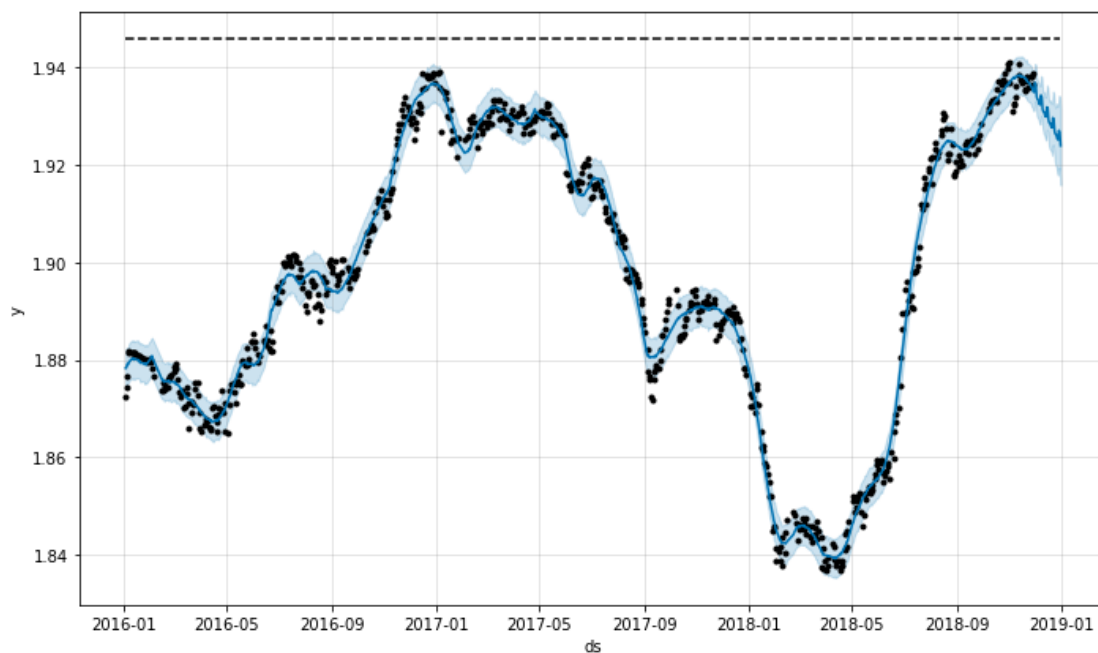
```
In [9]: future = m.make_future_dataframe(periods=32)
        future['cap'] = math.log(7.0)
```

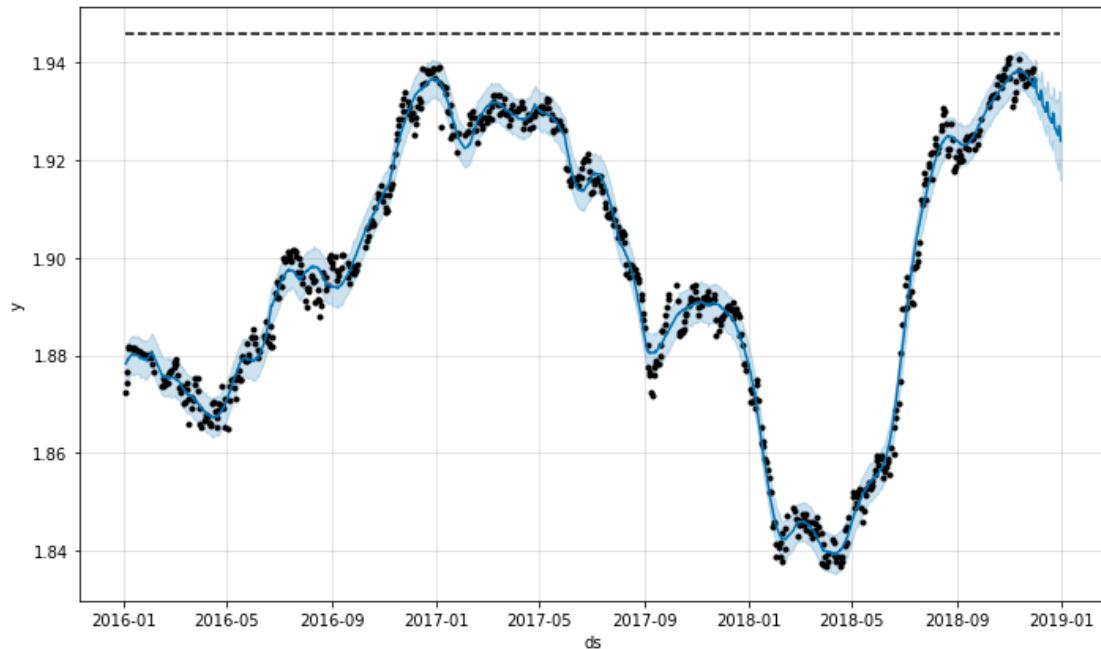
1.4.6 开始预测

开始激动人心的预测！设定 `future` 变量为对 `m` 的预测，参数为之前的 `future`。我们只需要时间，预测值和置信空间，所以第二行这样设置。`ds` 是时间，`yhat` 是预测值，`yhat_lower` 是置信区间下限，`yhat_upper` 是置信区间上限。可以看到图形的预测结果如下。黑色是现实对应中的实际点，蓝线是拟合波动线，浅蓝空间是置信区间。变点先验尺度可以通过此图说明，当变点先验尺度越大，蓝线就会越倾向于模拟每一个点的波动而不是总体趋势，变点先验尺度越小，蓝线就倾向于模拟大的趋势而更容易忽略拐点。我们通过图表来查看预测结果。虽然是取过对数的，但是增长趋势是相同的。

```
In [10]: forecast = m.predict(future)
         m.plot(forecast)
```

Out [10]:





1.4.7 完整预测数据

接下来我们讲数据完整地打印出来，这时候将取过对数的转回来。我们只需要时间，预测值和置信空间。ds 是时间，yhat 是预测值，yhat_lower 是置信区间下限，yhat_upper 是置信区间上限。

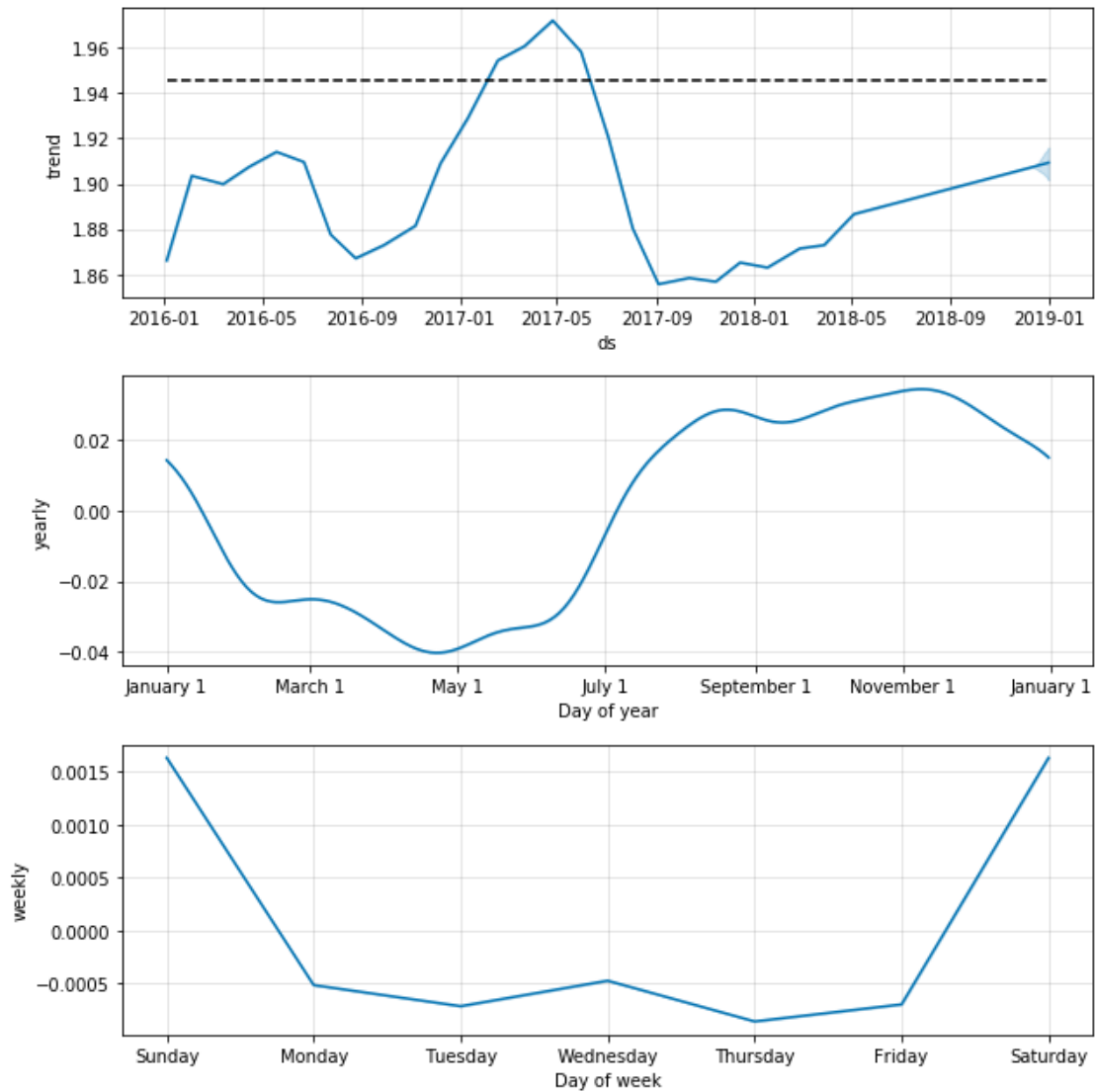
```
In [11]: forecast['yhat']=np.exp(forecast['yhat'])
forecast['yhat_lower']=np.exp(forecast['yhat_lower'])
forecast['yhat_upper']=np.exp(forecast['yhat_upper'])
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail(n=32)
```

```
Out[11]:
```

	ds	yhat	yhat_lower	yhat_upper
710	2018-11-30	6.923306	6.897250	6.952094
711	2018-12-01	6.937021	6.910925	6.963924
712	2018-12-02	6.934532	6.908858	6.962324
713	2018-12-03	6.917133	6.890636	6.945786
714	2018-12-04	6.913205	6.887228	6.939849
715	2018-12-05	6.912305	6.885772	6.937576
716	2018-12-06	6.907078	6.880015	6.934887
717	2018-12-07	6.905628	6.877847	6.932487
718	2018-12-08	6.919186	6.889471	6.946765
719	2018-12-09	6.916661	6.886368	6.945215
720	2018-12-10	6.899336	6.871766	6.925747

721	2018-12-11	6.895513	6.864838	6.922436
722	2018-12-12	6.894769	6.863371	6.924390
723	2018-12-13	6.889757	6.857475	6.916181
724	2018-12-14	6.888552	6.855564	6.919555
725	2018-12-15	6.902348	6.870686	6.933618
726	2018-12-16	6.900115	6.868511	6.930375
727	2018-12-17	6.883120	6.849357	6.914041
728	2018-12-18	6.879582	6.843626	6.910291
729	2018-12-19	6.879090	6.842441	6.911719
730	2018-12-20	6.874299	6.839548	6.907391
731	2018-12-21	6.873250	6.834704	6.907422
732	2018-12-22	6.887096	6.849313	6.924736
733	2018-12-23	6.884862	6.844079	6.921768
734	2018-12-24	6.867797	6.826756	6.904978
735	2018-12-25	6.864046	6.820927	6.901738
736	2018-12-26	6.863206	6.817726	6.907424
737	2018-12-27	6.857940	6.809188	6.903909
738	2018-12-28	6.856260	6.806389	6.903861
739	2018-12-29	6.869282	6.817202	6.918583
740	2018-12-30	6.866105	6.811605	6.917484
741	2018-12-31	6.847978	6.792850	6.899706

```
In [12]: fig2 = m.plot_components(forecast)
```



1.5 总结

这次汇率预测的结果预测在 2019-01-01 的汇率为 6.847978.

In []: