Prophet 模型

FaceBook 于2017年开源了一款时间序列建模工具 Prophet。 Prophet 这个英文单词是先知的意思，顾名思义就是通过历史来预测未来。

下面是 Prophet 的三个官方链接：

Propeht官方文档
Propeht–Github
Propeht论文

在介绍 Propeht 之间，我们先宏观的来看一下时间序列的建模问题。

传统的时间序列预测有两种范式，分别是自回归和动态回归：

自回归:

直通过变量的实际观察值来进行回归，代表性算法:arima

动态回归:

考虑其他外部变量来预测实际观察值，代表性算法:arimax

关于传统时间序列建模和机器学习建模：

两者有很大的相似性

- 机器学习中的建模一般需要假设样本独立同部分，依托**大数定律** 拟合出参数。
- 时间序列中的建模一般需要假设样本平稳遍历性，依托**遍历定理**拟合出参数。

一个在**空间上**通过对样本的采样和计算得到随机变量的相关性质。
一个在**时间上**通过对样本的采样和计算得到随机变量的相关性质。

Prophet 的方法是将时间序列看成是关于 t 的一个函数，用拟合函数曲线的方来来进行预测，所以这也注定了 Prophet 只能做单预测变量的时间序列回归，所以和传统时间序列建模有本质上的区别。他更倾向于机器学习的建模方式，后面的章节会进一步的讨论。

Prophet 并不是适用于所有的时间序列问题，由于他的建模假设和过程， Prophet 方法具有一定的适用范围，他适用于如下的时间序列：

商业时间序列（business time series）

- 有很强的季节性，可是多季节的。
- 存在趋势变化。
- 偶尔有离群点。
- 有假期效应。

Prophet 模型有着直观的参数进行调整，不像其他模型(arima)需要了解很多模型的机理（没这么多数学名词）。

---

2. 模型建模

Prophet 的本质是一个广义加法模型，即将一个时间序列进行分解，核心公式如下：

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

其中 $g(t)$ 表示趋势项，$s(t)$ 表示季节项，$h(t)$ 表示节假日，$epsilon_t$ 表示误差项。

这种分解方式和传统的时序分解很相近（e.g STL算法）。唯一不同点在于引入了 $h(t)$ 。且所有的分解都是关于 t 的函数。注意，我们也可以通过乘法将这三项组织起来，两边取 log 后没有本质上的不同。

下面来详细拆解每项。

2.1 趋势项

趋势项分为两种：

- 饱和的趋势项，即有一个上界（比如人口的增长相对比较缓慢，且有一个上界）。
- 线性的增长项，即一直有一个向上的增长。

饱和的增长项：

$$g(t) = \frac{C}{1 + exp(-k(t - m))}$$

线性的增长项：

$$g(t) = kt + m$$

下面以饱和增长项为例（线性增长项同理）

C 是上限容量，K 是增长率，m 是偏移量。随着 t 的不断的增加，$exp(-k(t - m)) \to 0$，所以 $g(t) \to C$。

上面公式有一定的局限性：

1. C 一般不是一个常数，比如人口的总是是在增长的，所以我们期望 C(t) 是变化的。
2. k 也不是一个固定的增长率，比如新产品在区域内的增长率会快速改变。所以我们期望 $k(t)$ 是变化的。

我们首先来解决第二个问题，假设 k 是变化的，且这种变化是间断变化的。 用 δ_j 表示每一个时间点的变化值，s_j 为该点对应的时间。基础的变化量为 k，所以某个时刻 t 的增长率为前面所有变化量的累积值：

$$k + \sum_{i=1}^{s_i<t} \delta_i$$

注意：这里的时间点可以自己来定义，比如一年中的每周一课可以作为时间点。

令 $\delta = (\delta_1, \delta_2, \cdots, \delta_S)$，$\delta_j$ 对应的是第 j 个时刻的变化量。

$$a_j(t) = \begin{cases} 1, & if \quad t \geq s_j \\ 0, & otherwise \end{cases}$$

定义$a_j(t)$的目的是为了简化等式，那么$\mathbf{a(t)}$ =$(a_1(t), a_2(t), \cdots, a_S(t))$。

那么上面等式可以简写为：

$$k + \mathbf{a(t)^T}\delta$$

由于我们引入了离散的变换的δ，必然将导致$g(t)$的不连续，为了保证连续性，我们需要调整m的值，m的值在每一个时间节点上都会调整，需要注意的是m的调整值是由δ_j唯一确定。
$\gamma = (\gamma_1, \gamma_2, \cdots, \gamma_S)$，$\gamma_j$对应第$j$个时刻应该调整的偏移的量。

$$\gamma_j = (s_j - m - \sum_{l<j} \gamma_l)(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l\leq} \delta_l})$$

综合上面有：

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

线性增长项同理：

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

这里 $\gamma_j = -s_j\delta_j$，同样是 m 的调整值。

然后我们来解决第一个问题，$C(t)$这里可以也是一个关于t的量，一般可以使用外部数据或者模型来进行估计。（如世界银行的人口数据预测）

关于 $delta_j$ 的选择包含两个方面，一方面是时间节点 s_j 的选择，即我们要在哪个节点产生增长率的突变。另一方面是 δ_j 的大小应该是多少。

第一种思路，是由人分析历史数据，将一些重要的日期（e.g 产品发布日期）作为增长率突变的点。然后通过学习的方式得到具体 δ_j 的值。

第二种思路，先等频产生大量的候选点（比如每周产生一个），每个候选点会采样自拉普拉斯分布（先验分布）。$\delta_j \sim Laplace(0, \gamma)$ 由于拉普拉斯分布的性质，导致采样得到的 δ 是稀疏的。 γ 越小，则表明得到的 δ 越稀疏。然后让 δ_j 去学习，让他学到具体的值。这个值可能是 0 ，可能非 0。 0 就表示这个点在此处不是增长率的突变点。前面假设 δ_j 满足拉普拉斯分布，就是为了使得学习到的 δ 具有这种稀疏性。类似于线性回归中的 $lasso$ 回归。所以通过调整 γ 可以调整曲线的拟合情况，γ 越大意味着这个曲线分片的越多，越容易拟合。（后面我们可以看着这个东西本质上就是机器学习中的正则项，γ类似于正则项系数，调越大则越容易拟合，这个正则项也一定程度解决outliers的问题，回想一下前面讲的Prophet适应的时间序列的一些特点）。

对于 δ_j 在预测阶段的核心假设是未来的 δ_j 和历史的 δ_j 是一致的。假设在一共 T 个点中，一共有 S 个 δ_j。
那么对 δ_j 的预测采样的步骤如下：
1. 通过历史数据拟合出 δ_j 之后，计算$\lambda = \frac{1}{S}\sum_{j=1}^{S}|\delta_j|$。这个步骤其实就是对 δ_j 服从的拉普拉斯分布做一个最大似然估计。估计 γ 值。

1. 采样得到未来的δ_j

$$\forall j > T = \begin{cases} \delta_j = 0, & w.p\dfrac{T-S}{T} \\ \delta_j \sim Laplace(0, \lambda), & w.p\dfrac{S}{T} \end{cases}$$

这个等式表示一定的概率 δ_j 为0，一定的概率采样自拉普拉斯分布。

从上面等式可以看到，预测的结果是有一定的随机性的。而且如果再训练阶段 γ 的值设定很大的话，那么训练集可能会拟合得很好，但也会导致 λ 的值变大，从而导致预测的不确定性增加。

2.2 周期项

周期项是用傅立叶展开式来逼近：

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

其中P表示周期（年用365.25，周用7），N表示使用的逼近项的数目（N越大越精细）。
理论上来说我们需要学习a_n, b_n的值，在prophet中，将a_n, b_n均看作来自正态分布的随机变量。

比如：

$$X(t) = [cos(xxx), sin(xxx), cos(xxx), \cdots, sin(xxx)]$$

$$s(t) = X(t)\beta$$

这里β就是要学习的a_n, b_n

$$\beta \sim Normal(0, \sigma^2)$$

注意1：和上面一样，其实这里可以看成是用这组有周期的基去做拟合，然后之所以假设他服从随机变量也是为了防止过拟合（引入正则项）。

注意2：N 的大小直接决定了拟合情况的好坏，太大容易过拟合，太小容易欠拟合。文建议年季节性使用 N = 10，周季节性使用 N = 3 。

2.3 节假日项

节假日不一定具有季节性，且会有一种突然的效应（不平滑），所以单独考虑节假日选项，比如每年的春节是不一样的，而且在春假那前后会有一个很大的突变，前面的已经建模的几项都有很强的连续性。

Prophet 是支持自定义节假日的。他会把全球通用的节假日和该国所独有的节假日（自定义），一起纳入模型进行建模。

假设我们有节假日 i ， D_i 表示过去和未来这种节假日的日期集合。假设有 L 种不同节假日（相当于节假日有 L 类）。对于任何一个时间，我们可以通过虚拟变量来表示这个时间是否属于某个节假日:

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

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

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

注意：有时候节假日具有窗口效应，比如春节后某个连着的周末，按论文说法，他们会把节假日后续的窗口归并到这个节假日中。

注意：节假日这里建模的核心假设是每个节假日的影响是相互独立的。（很可能出现一天同时是多个节假日的情况，比如五一和周末算作两种不同节假日类别，但五一这天可能恰好是周末）。

3. 模型训练

训练的过程就是一个拟合的过程。prophet的建模是从概率建模的角度来进行的，模型训练的过程是使用概率编程的思想，通过概率编程工具Stan来求解：

```
model {
  // Priors
  k ~ normal(0, 5);
  m ~ normal(0, 5);
  epsilon ~ normal(0, 0.5);
  delta ~ double_exponential(0, tau);
  beta ~ normal(0, sigma);

  // Logistic likelihood
  y ~ normal(C ./ (1 + exp(-(k + A * delta) .* (t - (m + A * gamma)))) +
             X * beta, epsilon);
  // Linear likelihood
  y ~ normal((k + A * delta) .* t + (m + A * gamma) + X * beta, sigma);
}
```

这里提一下，实际上他和机器学习建模过程没有本质区别。如果将整个建模过程看成是参数模型的建模，那么前面需要学习的随机变量就是需要学习的参数，但是他加入了先验信息（先验分布）。那么这个问题实际上就转化为了最大后验估计（MAP）的问题（相当于最大似然估计进阶版，最大似然估计一句话概括就是“存在即合理”）。从机器学习的建模角度上来看MAP的问题等价于损失函数加上正则项。所以Prophet本质上仍然是机器学习建模的思路，只是把拟合项进行了很合理的拆解，选择了合理的正则项（先验分布）。

举个例子：从机器学习角度来看lasso回归就是MSE的损失函数加上L1正则化。从概率模型的角度上来看就是样本点服从均值为直线变化的高斯分布，且参数的先验分布是另一个高斯分布。

所以从宏观上来看，模型的训练过程就是一个曲线拟合的过程。所以 prophet 对于部分缺失值不会太敏感，但是对一些异常值可能会很敏感（想象一下用曲线去拟合一系列的点）。

但很讨巧的是，Prophet 从概率角度来建模的原因，更多是为了在预测阶段引入不确定性（因为时间序列的预测不仅仅要做点预测，更要做区间预测）比如再前面 2.1 小节中，通过采样方式生成 δ_j，这种预测阶段的不确定性保证了 Prophet 可以做区间预测。

4. 模型迭代流程

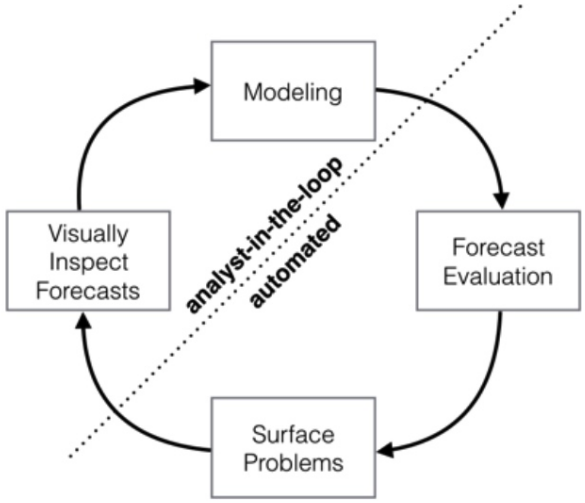
Propeht 模型中可以引入专家知识的地方，这几个也是用模型调整超参数的重要地方：

Capacities: Analysts may have external data for the total market size and can apply that knowledge directly by specifying capacities.

Changepoints: Known dates of changepoints, such as dates of product changes, can be directly specified.

Holidays and seasonality: Analysts that we work with have experience with which holidays impact growth in which regions, and they can directly input the relevant holiday dates and the applicable time scales of seasonality.

Smoothing parameters: By adjusting τ an analyst can select from within a range of more global or locally smooth models. The seasonality and holiday smoothing parameters (σ, ν) allow the analyst to tell the model how much of the historical seasonal variation is expected in the future.



5. 模型使用

python 版安装

```
pip install fbprophet
```

prophet 采用的是 sklearn 的 API 设计规范，使用 fit 训练模型，predict 预测模型。

propeht 的输入数据是面板数据(DataFrame)，包含 ds ， y 两列，ds 必须是 Pandas 的时间格式（ YYYY-MM-DD , YYYY-MM-DD HH:MM:SS） ， y 是数值类型，是我们需要预测的数据。

5.1 基本使用

```
import pandas as pd
from fbprophet import Prophet
# 训练
m = Prophet()
m.fit(df)
# 预测
# 获取需要预测的时间
future = m.make_future_dataframe(periods=365) # 这里默认是包含了历史的时间
forecast = m.predict(future)
# 预测的结果会包含 DS    YHAT    YHAT_LOWER  YHAT_UPPER

# 使用prophet的画图工具
fig1 = m.plot(forecast)

# 画出趋势，季节性
fig2 = m.plot_components(forecast)
```

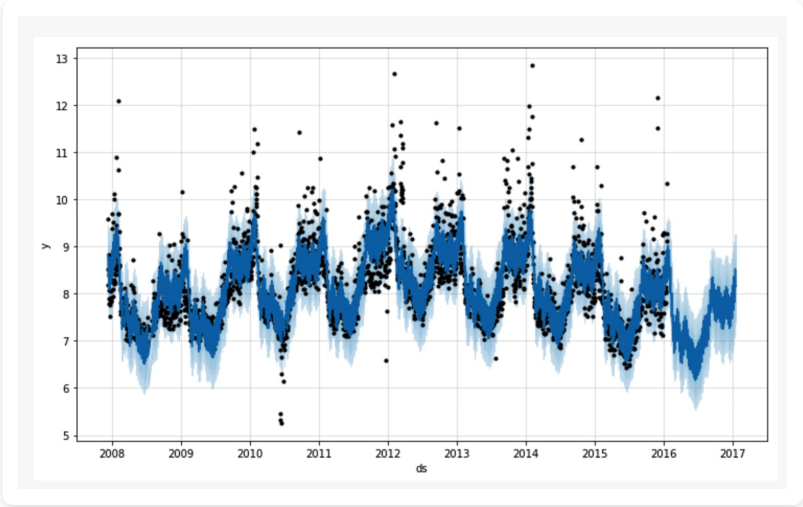
```
# 如果安装了 plotly 可以画互动的图
from fbprophet.plot import plot_plotly
import plotly.offline as py
py.init_notebook_mode()

fig = plot_plotly(m, forecast)
py.iplot(fig)
```

5.2 进阶操作

1.设定趋势增长上限和下降下限。

prophet 默认是用的线性模型。

```
# cap不一定是一个固定值，也可以是一个变化的值。
df['cap'] = 10
df['floor'] = 3
m = Prophet(growth='logistic')
m.fit(df)
future = m.make_future_dataframe(periods=1826)
future['cap'] = 10
future['floor'] = 3
fcst = m.predict(future)
fig = m.plot(fcst)
```



2.设定增长突变点。

打印出自动学习出的增长突变点。

```
from fbprophet.plot import add_changepoints_to_plot
fig = m.plot(forecast)
a = add_changepoints_to_plot(fig.gca(), m, forecast)
```

增长突变点主要有自动和手动两种方式：

手动：
设定参数 changepoints，为一个 list，list中元素为潜在的增长突变点的日期。

手动：
设定参数 n_changepoints 和 changepoint_range，changepoint_range为训练历史时间序列的比例，n_changepoints为均匀采样的个数。n_changepoints 会在选择的这段历史时间序列进行均匀采样作为增长突变点。

changepoint_prior_scale 调整的就是前面说的 γ，默认是 0.05 ， 这个值越大，得到的突变点就会越少。反之则越多。

3.设定节假日

自定义的节假日也是以 DatFrame 的形式提供给 prophet。

。

```
playoffs = pd.DataFrame({
  'holiday': 'playoff',
  'ds': pd.to_datetime(['2008-01-13', '2009-01-03', '2010-01-16',
                        '2010-01-24', '2010-02-07', '2011-01-08',
                        '2013-01-12', '2014-01-12', '2014-01-19',
                        '2014-02-02', '2015-01-11', '2016-01-17',
                        '2016-01-24', '2016-02-07']),
  'lower_window': 0,
  'upper_window': 1,
})
superbowls = pd.DataFrame({
  'holiday': 'superbowl',
  'ds': pd.to_datetime(['2010-02-07', '2014-02-02', '2016-02-07']),
  'lower_window': 0,
  'upper_window': 1,
})
holidays = pd.concat((playoffs, superbowls))
```

其中 DataFrame 有如下几个字段比较重要：

- ds: datetime类型的如期
- holiday: 对应的假期名字
- lower_window: 窗口效应
- upper_window: 窗口效应

- prior_scale: 对应前面说的 κ。不同节假日的可以不同。

注意1:相同的一天可以对应多个节假日。
注意2:节假日要尽量包含过去的历史数据和未来待预测数据。

```
m = Prophet(holidays=holidays,holidays_prior_scale = 0.05)
forecast = m.fit(df).predict(future)
```

holidays_prior_scale 对应的就是前面的 κ 默认值是10。这个值越小，假期效应越弱。

```
fig = m.plot_components(forecast)
plot_forecast_component(m, forecast, 'superbowl')
```

```
# 加入国家法定假期
m = Prophet(holidays=holidays)
m.add_country_holidays(country_name='US')
m.fit(df)
# 查看加入了哪些国家法定假期
m.train_holiday_names
```

部分国家简称：
– Brazil (BR)
– Indonesia (ID)
– India (IN)
– Malaysia (MY)
– Vietnam (VN)

– Thailand (TH)
– Philippines (PH)
– Turkey (TU)
– Pakistan (PK)
– Bangladesh (BD)
– Egypt (EG)
– China (CN)
– Russian (RU)

holiday python

4.设定季节项

yearly_seasonality (weekly_seasonality) 设定的是周期项公式中的 N。其中 p 已经确定是 365.25。 N 越大，则拟合会越精细。会存在过拟合风险。 prophet 中年季节默认用的是 N = 10。学习的参数个数是 2N 个。

```
from fbprophet.plot import plot_yearly
m = Prophet(yearly_seasonality=20).fit(df)
a = plot_yearly(m)
```

我们可以使用自定义的季节频率

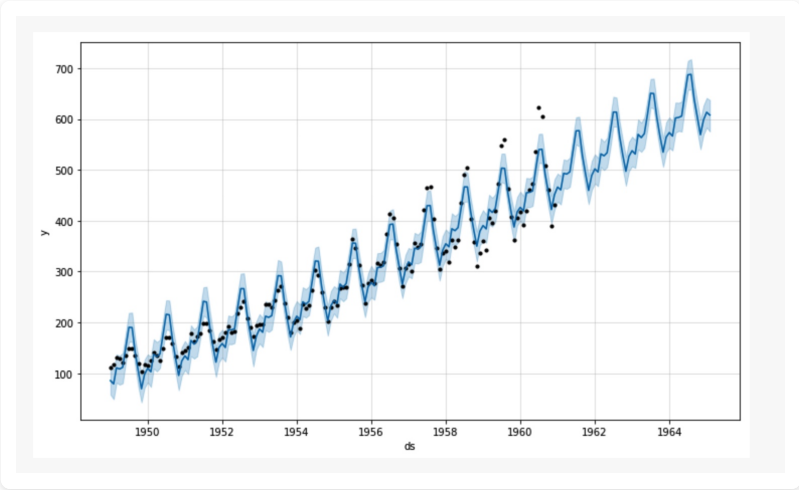
```
m = Prophet(weekly_seasonality=False)
m.add_seasonality(name='monthly', period=30.5, fourier_order=5)
forecast = m.fit(df).predict(future)
fig = m.plot_components(forecast)
```

这里的fourier_oder 对应的就是季节项中的 N 。
seasonality_prior_scale 调节季节性的先验。

```
m = Prophet()
m.add_seasonality(
name='weekly', period=7, fourier_order=3, prior_scale=0.1)
```

这里的 prior_scale 等价于 seasonality_prior_scale

5.加法模型和乘法模型




```
m = Prophet(seasonality_mode='multiplicative')
m.fit(df)
forecast = m.predict(future)
fig = m.plot(forecast)
```

```
m.add_seasonality('quarterly', period=91.25,
fourier_order=8, mode='additive')
```