Time Series Forecasting with Prophet

Rener Oliveira

February 22, 2021

Contents

1 Introduction

Prophet is a package available for R and Python that aims to facilitate the process of univariate time series modeling for those who doesn't have statistical background, but have knowledge from the data generation process. Model fitting can be done with just one line, but all the parameters can be tuned for a better suitability. This text aims to presents the mathematical backgroud of the packages and give soma practical examples.

2 The Prophet Forecasting Model

The Prophet package (S. J. Taylor and Letham 2018) assumes the data generating process of a time series y can be decomposed as:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t,$$

where $g(t)$ models non-seasonal time trend, $s(t)$ is the periodic effects in data, $h(t)$ models the impact of holidays, and ε_t is a noise term. In the following sections we describe in detail each model.

2.1 The Trend Model

For trend modeling, the authors propose two models that cover many of the company's applications.

The simpler is a linear regression against time:

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

where k is the growth rate, and m is the intercept. This model makes the strong assumption of a constant growth rate through the series. To fit variations in the growth rate through time, we define cut-points where such a rate would be able to change. Suppose, there are S changepoints at times s_j, $j = 1, 2, \dots, S$, and δ_j is the rate variation at s_j. Then, given t the rate will be $k + \sum_{j: s_j < t} \delta_j$. Representing all the changes δ_j in a $\mathbb{R}^S$ vector $\boldsymbol{\delta}$, we can represent the new regression model as:

$$g(t) = (k + \boldsymbol{a}(t)^\intercal \boldsymbol{\delta})t + m + \boldsymbol{a}(t)^\intercal \boldsymbol{\gamma},$$

where $\boldsymbol{a}(t)$ is a S-dimensional indicator vector such that:

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

and $\boldsymbol{\gamma}$ is another S-dimensional vector, which elements are $\gamma_j = -s_j \delta_j$. That's just an intercept correction to make the function continuous for all values of t, especially the changepoints.

The second trend model is a logistic growth model. The most basic form is:

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

where C is the carrying capacity, k is the growth rate, and m is the offset parameter. We can have a non-constant carrying capacity. For example, if Facebook is modeling its active user's time series, such a carrying capacity is the world population with internet access, certainly a time-increasing number. To address this problem, we replace C by a capacity function $C(t)$, which can be specified by the user with knowledge about its time series limits.

Another improvement in the function is to introduce variable growth rate, we use the vector of $\boldsymbol{\delta}$, representing S cut-points changes, as before. Then, the piecewise logistic model is:

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

where $\boldsymbol{\gamma}$ is an intercept correction to make g continuous in the changepoints, and it can be computed as:

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

2.1.1 Changepoint Selection and Uncertainty Forecast

The changepoints can be defined manually by the user who approximately knows the dates when of growth changes in his business. If not specified, Prophet package will automatically select S changepoints from the first $p\%$ of history data. By default $p = 80$ but it can be set by the user.

The parameters δ_j are set to have the prior distribution Laplace$(0, \tau)$. When $\tau \to 0$, the growth function converges to its standard non-piecewise version, with constant growth rate k.

To extrapolate changepoints for out of sample forecasts, we use the maximum likelihood estimator of the in-sample growth changes $\lambda = \sum_{j=1}^{S} |\delta_j|$ and for future values $j > T$, we randomly sample δ_j using the distribution:

$$\begin{cases} \delta_j = 0, \text{ with prob. } \frac{T-S}{T} \\ \delta_j \sim \text{Laplace}(0, \lambda), \text{ with prob. } \frac{S}{T} \end{cases}$$

With this distribution, we can make simulations and compute empirical uncertainty intervals. Note that as τ increases, the model has more flexibility to fit observed data, and consequently reducing in-sample error, but will have poor generalization power and simulations will produce large uncertainty intervals.

2.2 The Seasonality Model

A common characteristics of time series is the seasonal component, for example, a retail company might observe a higher sales in the end of the year, because of people buying Christmas gifts; That's a pattern that repeats every year, so we add a periodic function with 1-year period to the model.

Suppose that we have a P-periodic time series and we want to fit the seasonal pattern. The authors propose the following Fourier Series model:

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

This model has $2N$ parameters to fit. And we can summarize-it in the vector form $s(t) = X(t)\boldsymbol{\beta}$, where

$$\boldsymbol{\beta} = [a_1, b_1, \ldots, a_n, b_n]^\intercal, \text{ and}$$

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{P}\right), \sin\left(\frac{2\pi(1)t}{P}\right), \ldots, \cos\left(\frac{2\pi(n)t}{P}\right), \sin\left(\frac{2\pi(n)t}{P}\right) \right]$$

In a Bayesian framework, the parameters are set to have the prior distribution $\boldsymbol{\beta} \sim \mathcal{N}(0, \sigma^2)$. The number of Fourier terms N, has to be set with care, because a large N can potentially overfit the data.

2.3 Holidays

In many time series, holidays might cause big shocks and produce outliers. To address this issue, the function $h(t)$ is added to the general model. Suppose each year we have L holidays and/or special events that affects our data. Let i be a holiday, define D_i as the set of time indexes of all past and future dates of holiday i. Then we just add an indicator variable of t being in a holiday.

$$h(t) = \sum_{i=1}^{L} \kappa_i \cdot \mathbf{1}\{t \in D_i\}$$

We can fit the coefficients, by introducing a regressor matrix the same way as before:

$$\begin{gathered} h(t) = Z(t)\boldsymbol{\kappa}, \text{ where} \\ Z(t) = [\mathbf{1}\{t \in D_1\}, \ldots, \mathbf{1}\{t \in D_L\}], \text{ and} \\ \boldsymbol{\kappa} = [\kappa_1, \ldots, \kappa_L]^\intercal \end{gathered}$$

3 Bayesian Parameter Estimation

There are a lot of parameters in Prophet model, k and m for growth rate and offset, the vector of growth changes $\boldsymbol{\delta}$, seasonal coefficients $\boldsymbol{\beta}$, holidays coefficients $\boldsymbol{\kappa}$.

Using a Bayesian approach, we set normal priors to k and m, Laplace prior to $\boldsymbol{\delta}$ as seen before, and for the other parameters, we combine seasonality and holidays regressors in a single matrix $\tilde{X} = [X, Z]^\intercal$, then we define $\tilde{\boldsymbol{\beta}} = [\boldsymbol{\beta}, \boldsymbol{\kappa}]^\intercal$. The we have the priors:

$\boldsymbol{\delta} \sim \text{Laplace}(\mathbf{0}, \boldsymbol{\tau})$
$\tilde{\boldsymbol{\beta}} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$

We can interpret the hyperparameters τ and σ as regularization controls. If they are small, seasonal/holidays coefficients will be tightly distributed around zero and will only deviate from that if data is informative through the likelihood.

We assume a normal likelihood as a T-dimensional multivariate normal distribution $\mathcal{N}(G + \tilde{X}\tilde{\boldsymbol{\beta}}, \varepsilon^2 I)$, where G is a matrix where the row t in the growth function $g(t)$ which depends on $k, m, \boldsymbol{\delta}$, and whether the trend is linear or logistic. The variance ε^2 has its own normal prior too.

The fitting algorithm is L-BFGS a well-known optimization algorithm that is implemented on Stan (source of Prophet model) and here it's used to find the parameters that maximizes the posterior distribution, given the above described priors and likelihood.

4 Estimating out-of-sample error

Let $\hat{y}(t|T)$ be a prediction for $y(t)$ made with a model using data up to time T, and $d(y, y')$ be any regression error metric (such as mean squared error or mean absolute error). We define the error of a forecast h periods ahead of T by:

$$\phi(T, h) = d(\hat{y}(T + h|T), y(T + h))$$

If our data ends in time T, we don't have access to the real values of $y(T + h)$, then, our out-of-sample error for a given horizon h will be:

$$\xi(h) = \mathbb{E}[\phi(T, h)],$$

which can be estimated empirically through **simulated historical forecasts**.

The idea is to isolate a part of the dataset $[T_1, T_2]$ (with $T_2 - T_1 \geq h$) and use just the points up to T_1 to train the model. Then we estimate $\xi(h)$ as the average of forecast errors using different cut-off points for training:

$$\hat{\xi}(h) = \frac{1}{T_2 - T_1} \sum_{T=T_1}^{T_2} \phi(T, h)$$

This method is often called expanding window, or rolling window if we drop the first point from training when adding a new one. Prophet also gives the option to set a period in the interval $[T_1, T_2]$ to avoid computing $\phi(T, h)$ for every forecast date.

The following images, taken from (R. J. H. Hyndman 2016), illustrate this procedure. In the case of 1-step ahead forecast (Figure 1), we use the blue points to predict the red ones, and then compare with true values, computing error ϕ and averaging through time, to estimate $\xi(1)$.

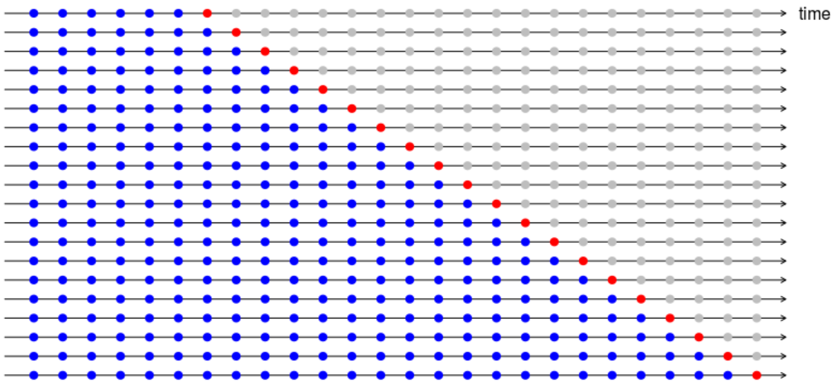

Figure 1: Expanding windows forecast for h=1

If we aim to estimate $\xi(4)$, we must place the red points 4 steps ahead the last blue training point (Figure 2)

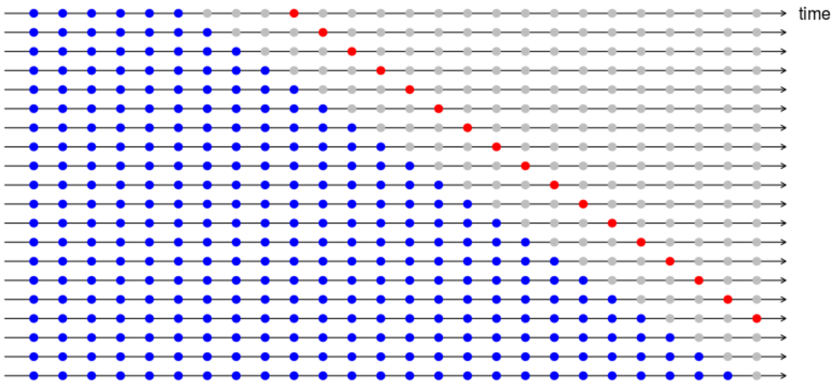

Figure 2: Expanding windows forecast for h=4

5 Practical Examples

Let's use the example data from Prophet's documentation (S. Taylor and Letham 2021), which is log daily page views of Peyton Manning's Wikipedia Page. Model fitting with all default parameters, is very simple in R (also in Python):

```
library(prophet)
model = prophet(df)
```

Forecasting is also simple, Figure 3, shows Prophet's 365-days-ahead forecast of page views.

```
future <- make_future_dataframe(model, periods = 365)
forecast <- predict(model, future)

plot(model, forecast)
```

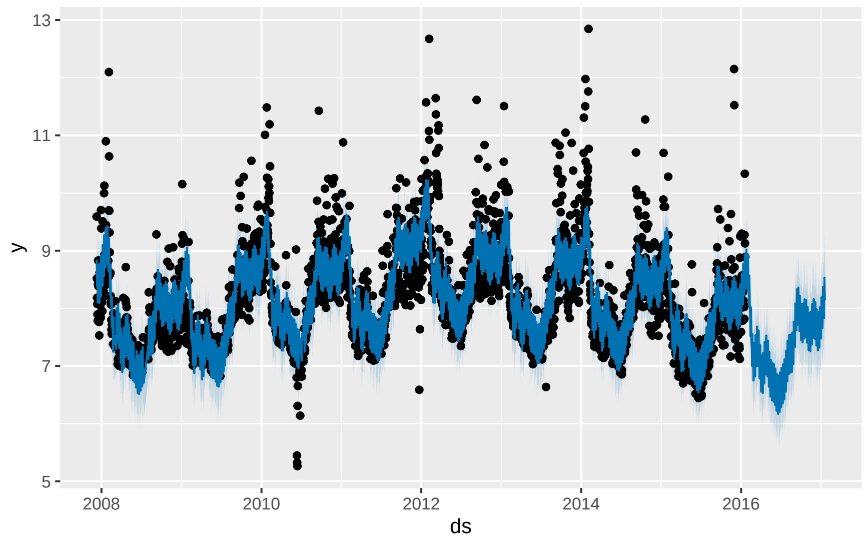

Figure 3: Prophet 1-year-ahead Forecasting

We also can plot the automatically selected changepoints (Figure 5).

```
plot(model, forecast) + add_changepoints_to_plot(model)
```

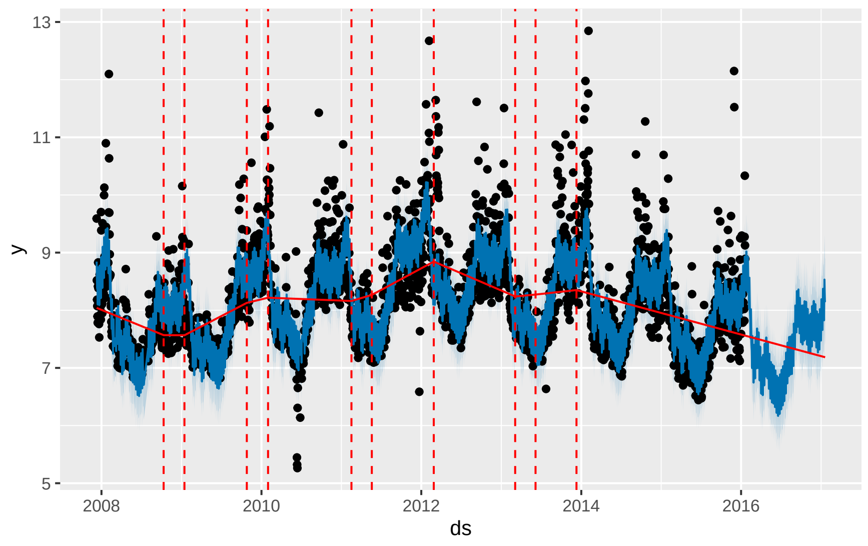

Figure 4: Changepoints plot

A note on generalization: Phophet uses by default 25 changepoints candidates that are fitted in the first 80% of the data using the sparse prior Laplace$(0, \tau)$, where $\tau = 0.05$ by default. Larger values of τ may lead to poor generalization and consequently a large uncertainty interval. Figure 4 shows for different values of tau, the distribution of interval width along the 1-year forecast. As expected, larger τ, produces more uncertainty.

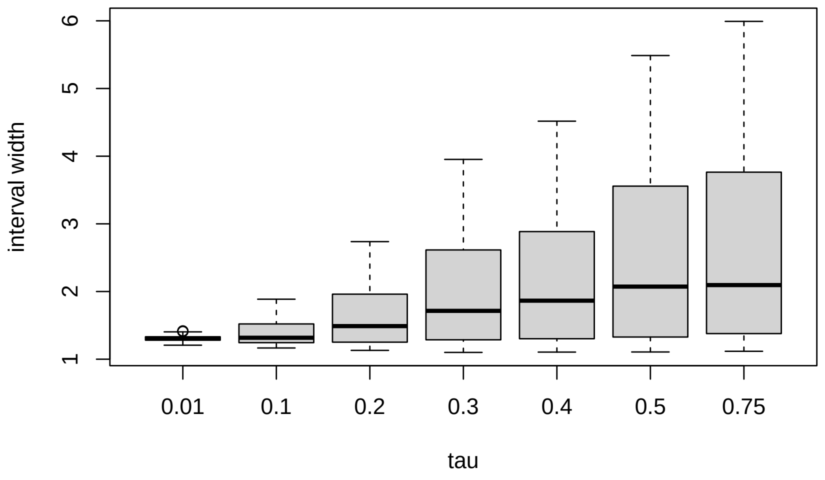

Figure 5: Uncertainty interval width by tau value

5.1 Benchmarks

In Prophet's paper the authors used Simulated Historical Forecasts in a time series of Facebook events, with has a lot of components well adressed by the package such as holiday shocks ans multiplie seasonality.

Prophet model beats all othe time seires models for every horizon

Here we used a simpler time series: quarterly expenditure on cafes, restaurants and takeway food services in Australia (Figure 6) provided by fpp R package (R. J. Hyndman 2013).

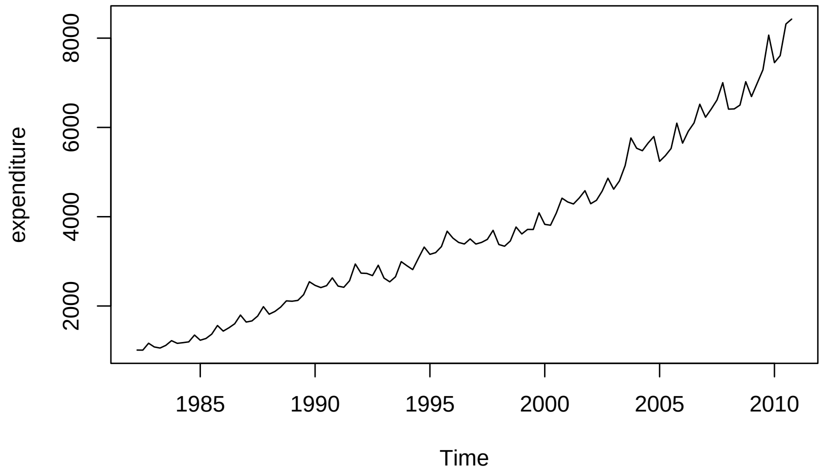

Figure 6: Australian Cafes expendirure

Figure 7 shows the results of Simulated Historical Forecasts (using log of timeseries), using RMSE, from horizons 2 to 20, and comparing Prophet with ARIMA and ETS models, two famous, simple and often powerful time series models. We can see that for low horizons, ARIMA is better than Prophet, which makes sense, since the time series has a very simple patters

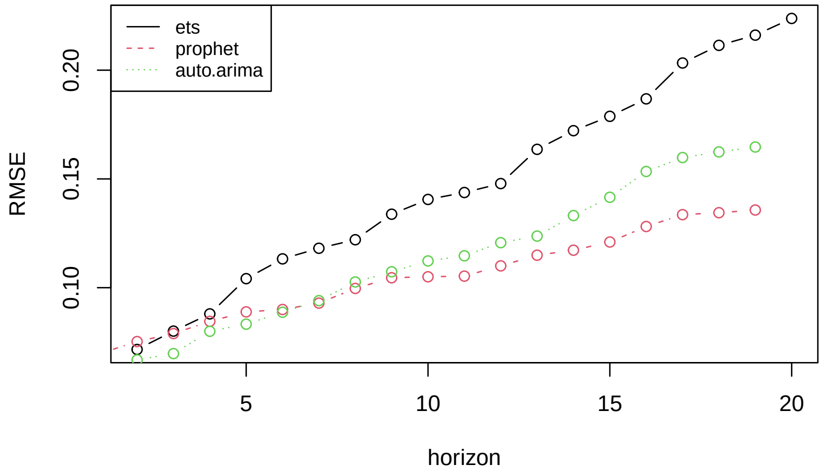

Figure 7: Benchmarks

References

Hyndman, R J H. 2016. “Cross-validation for time series.” https://robjhyndman.com/hyndsight/tscv/.

Hyndman, Rob J. 2013. *Fpp: Data for "Forecasting: Principles and Practice"*. https://CRAN.R-project.org/package=fpp.

Taylor, Sean J, and Benjamin Letham. 2018. “Forecasting at Scale.” *The American Statistician* 72 (1): 37–45.

Taylor, Sean, and Ben Letham. 2021. *Prophet: Automatic Forecasting Procedure*. https://CRAN.R-project.org/package=prophet.