NeuralProphet

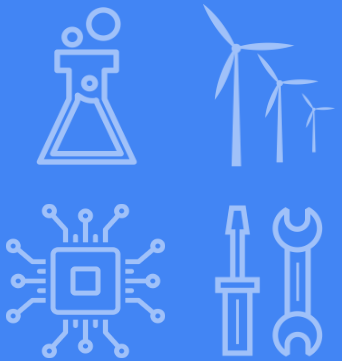
A simple time series forecasting framework

Oskar Triebe, Stanford University

Nov 19th, 2021

A time series is

a sequence of data points
that occur in successive order
over some period of time.



Neural Prophet

is	an open-source forecasting library.
tl;dr	Prophet in PyTorch + AR + Covar + NN + multistep + ...
Task:	Forecasting.
Data:	1E+2 to 1E+6 of samples. Unidistant, real-valued.
Dynamics:	Future values must depend on past observations. e.g. Seasonal, trended, events, correlated variables.
Applications:	Human behavior, energy, traffic, sales, environment, server load, ...

Motivation

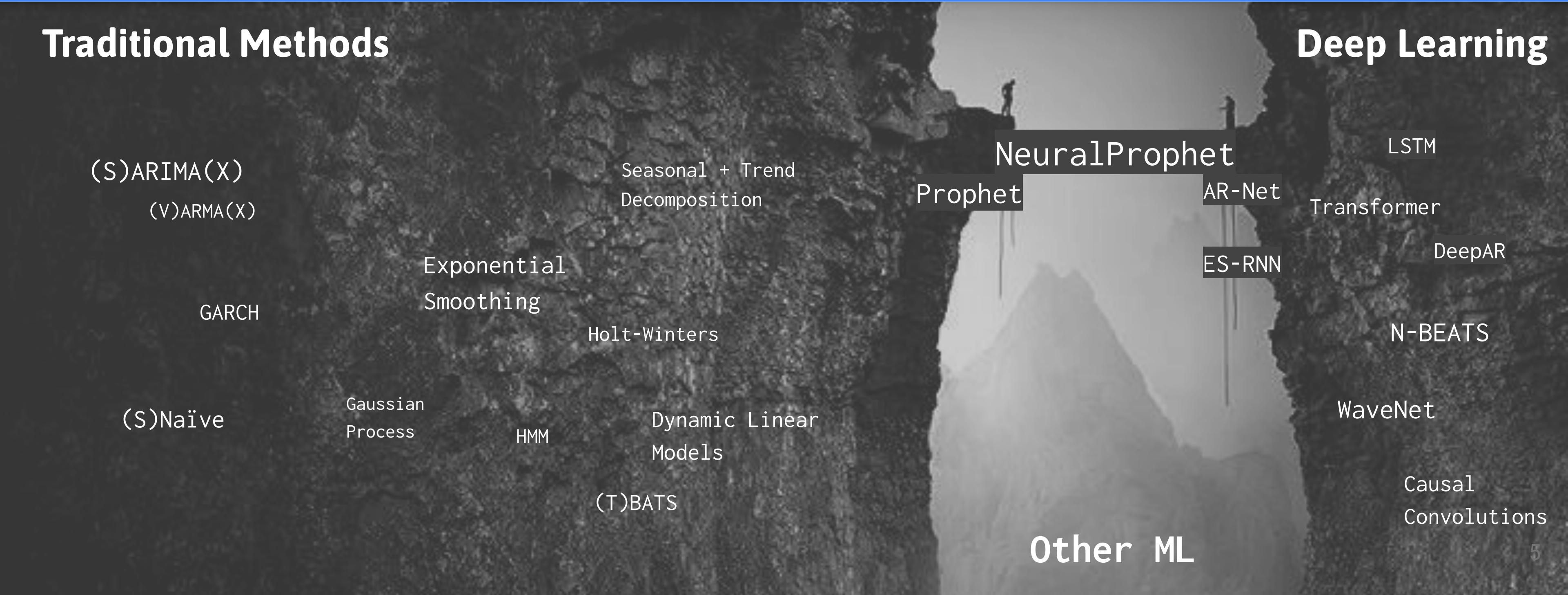
Model

Example

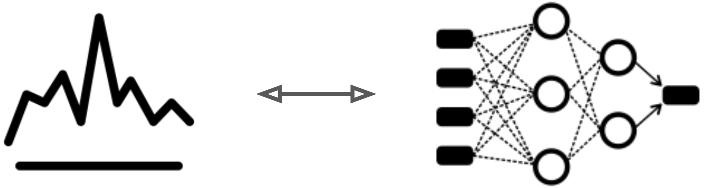
Outlook

Conclusion

Time series forecasting is messy.
We need hybrid models to bridge the gap.



Before



Need expertise in both

time series & machine learning

NeuralProphet

Abstracts time series

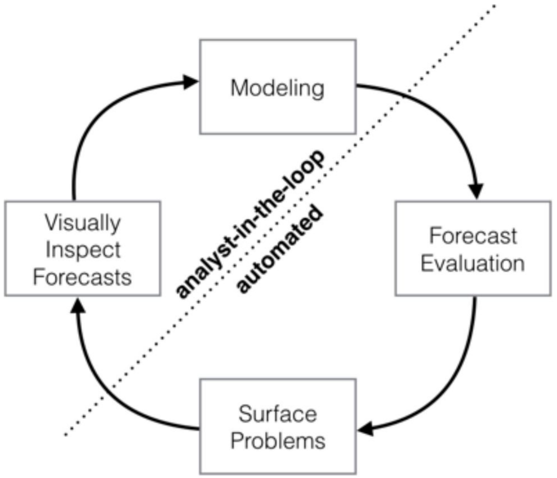
& machine learning knowledge

Facebook Prophet is still the most used forecasting package.

Quick from data to predictions.

Gentle learning curve.

Customizable.



Taylor, S. J., & Letham, B. (2017). Forecasting at scale, PeerJ. https://peerj.com/preprints/3190/

NeuralProphet is more than the Neural evolution of Prophet.

PROPHET

Prophet has three major shortcomings:

1. Missing local context for predictions
2. Acceptable forecast accuracy
3. Framework is difficult to extend (Stan)

NeuralProphet

NeuralProphet solves these:

1. Support for auto-regression and covariates.
2. Hybrid model (linear <> Neural Network)
3. Python package based on PyTorch using standard deep learning methods.

Stan → PyTorch

A user-friendly Python package

Gentle learning curve.

Get results first. Learn. Improve.

Powerful, customizable, extendable.

```
m = NeuralProphet()
metrics = m.fit(df, freq='D')
forecast = m.predict(df)
m.plot(forecast)
```

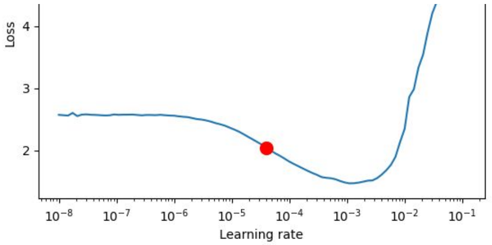
Hyperparameters have smart defaults.

Loss Function is **Huber loss**,
unless user-defined.

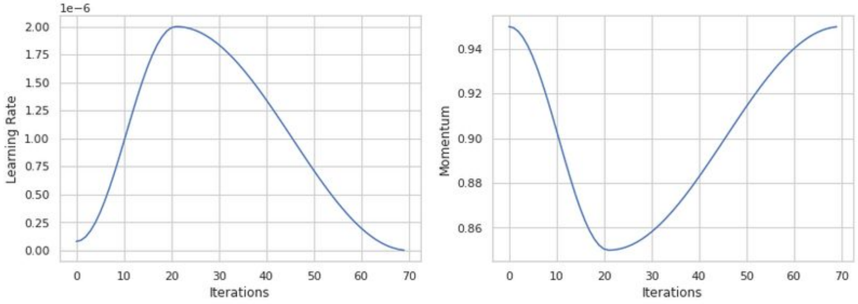
$$L_{huber}(y, \hat{y}) = \begin{cases} \frac{1}{2\beta}(y - \hat{y})^2, & \text{for } |y - \hat{y}| < \beta \\ |y - \hat{y}| - \frac{\beta}{2}, & \text{otherwise} \end{cases}$$

The learning rate is approximated
with a **learning-rate range test**.

Batch size and epochs are approximated
from the dataset size.



We use **one-cycle policy**
with AdamW as optimizer for simplicity.



Preprocessing happens mostly behind the scenes.

Missing Data is is automatically filled in:

1. bi-directional linear interpolation
2. centred rolling average

Data is automatically normalized:

Name	Normalization Procedure
'auto'	'minmax' if binary, else 'soft'
'off'	bypasses data normalization
'minmax'	scales the minimum value to 0.0 and the maximum value to 1.0
'standardize'	zero-centers and divides by the standard deviation
'soft'	scales the minimum value to 0.0 and the 95th quantile to 1.0
'soft1'	scales the minimum value to 0.1 and the 90th quantile to 0.9

We have utils ...

Visualize:

- Plot past and future predictions
- Decompose forecast components
- Interpret model parameters
- Plot most recent prediction
- Inspect a particular forecast horizon

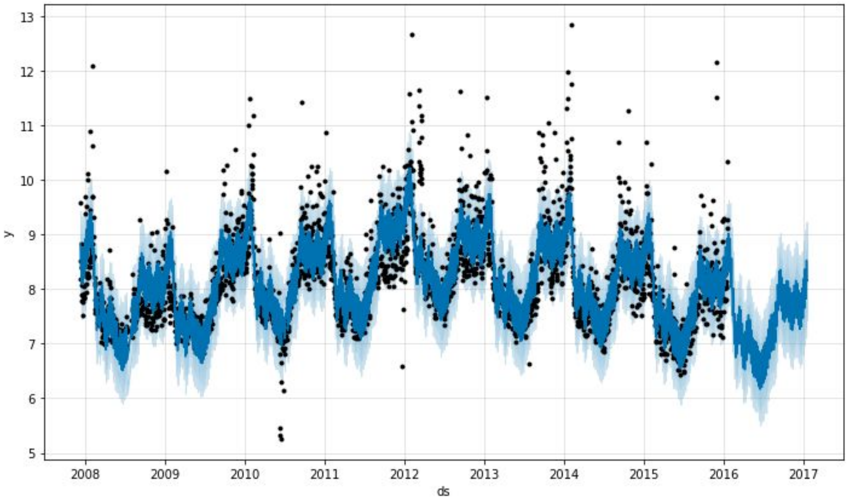
Other:

- Simple Split, cross validation, double cross validation
- Control logger verbosity
- Make fit reproducible
- Global modelling
- Benchmarking
- ...

Current Model Components

S	*Seasonality*
T	*Trend*
E/H	*Events / Holidays*
X	*Regressors*
AR	*Autoregression*
Cov	*Covariates*
_	*Sparsity / Regularization*
NN	*Nonlinear (deep) layers*
{ }	*Global Modelling*
?	*Uncertainty*

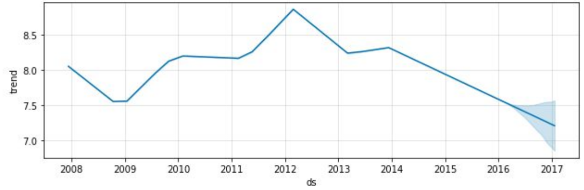
Prophet is interpretable and decomposable. So is NeuralProphet.



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

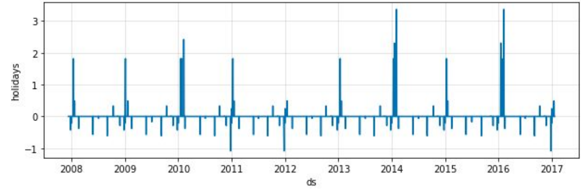
Trend

$g(t)$



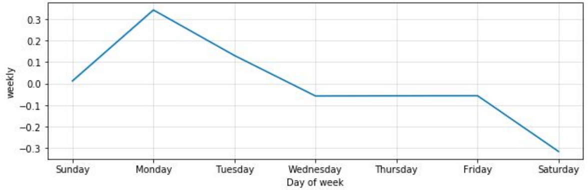
Events

$h(t)$



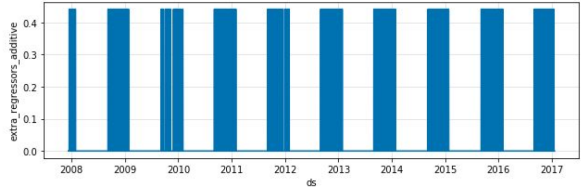
Seasonality

$s(t)$



Regressors

$X(t)$



Trend & Seasonality are identical to Prophet.

Piecewise linear trend

- N changepoints
- Segment-wise independent
- Automatic changepoint detection
- Optional logistic growth

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

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

$$-s_j\delta_j$$

Optional Regularization

$$R(\theta, \epsilon, \alpha) = \frac{1}{p}\sum_{i=1}^{p} log(\frac{1}{\epsilon \cdot e} + \alpha \cdot |\theta_i|) + log(\epsilon) + 1$$

Seasonality

- N Fourier terms
- Automatic yearly, weekly, daily
- Optional multiplicative mode

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

$$s(t) = X(t)\boldsymbol{\beta}.$$

$$X(t) = \left[\cos\left(\frac{2\pi(1)t}{365.25}\right), \ldots, \sin\left(\frac{2\pi(10)t}{365.25}\right)\right]$$

Events, Holidays and Regressors are identical to Prophet

Events / Holidays

- Automatic for given country
- Various user-defined formats
- Optional multiplicative mode

$$Z(t) = \sum_{i=1}^{m} c_i e_i(t)$$

(Future-Known) Regressors

- Single weight
- Real-valued regressor
- Optional multiplicative mode

$$R(t) = \sum_{i=1}^{l} d_i v_i(t)$$

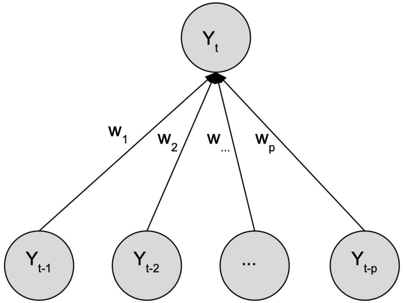
We model Covariates and Auto-Regression as AR-Nets.

Auto-Regression

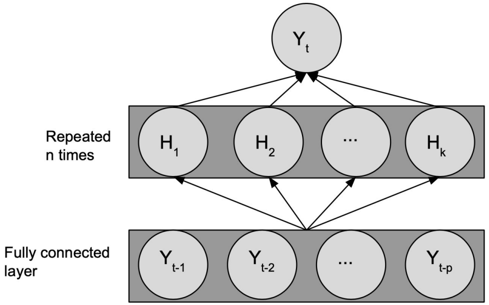
- By default AR-Net(0)
- Depth customizable AR-Net(n)
- Optional auto-AR via regularization

(Lagged) Covariates

- By default AR-Net(0) with y as target
- Depth customizable AR-Net(n)
- Optional lag-sparsification via regularization



AR-Net(0)
Interpretable

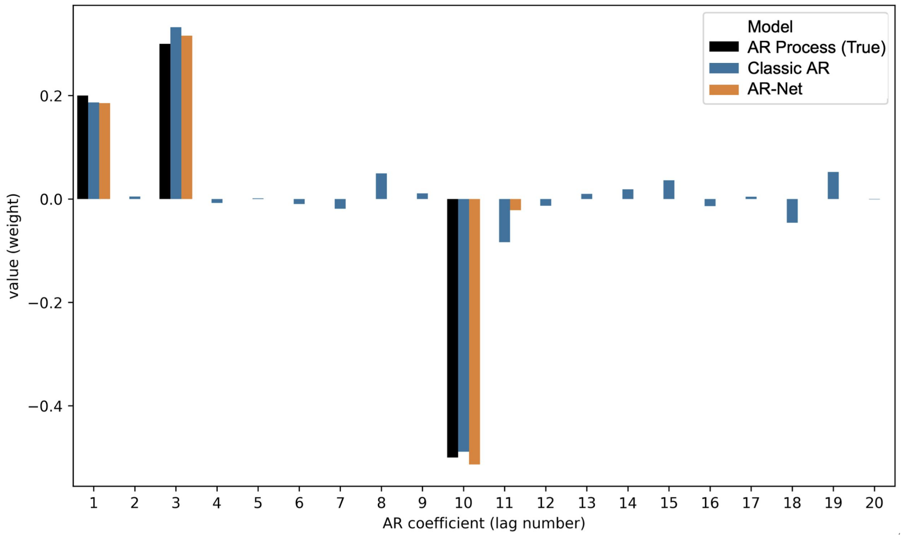


AR-Net(n)
Non-linear modeling

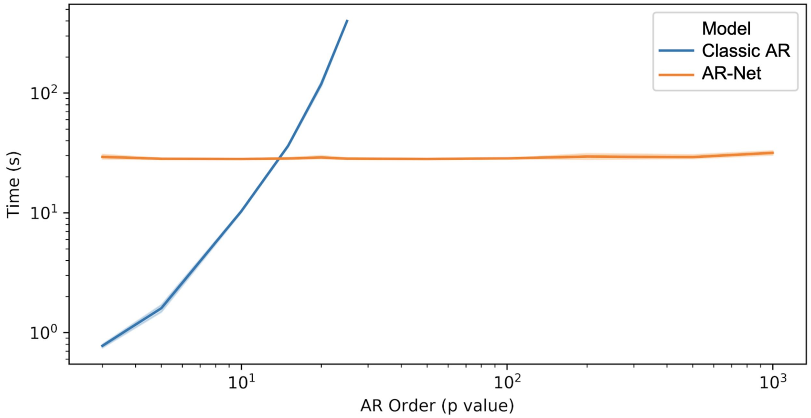
$$R_{AR}(\theta) = R(\theta, \epsilon = 3, \alpha = 1)$$
$$= \frac{1}{p}\sum_{i=1}^{p} log(\frac{1}{3 \cdot e} + |\theta_i|) + log(3) + 1$$

Optional Regularization

Automatic AR-lag selection, yet faster.



Automatic Sparsity



Quadratically faster

Quick Start Examples

The classic: Temperature forecast

Using Trend, Seasonality and Auto-Regression:

Example 1: 1 step ahead

Example 2: 36 steps ahead

Dataset:

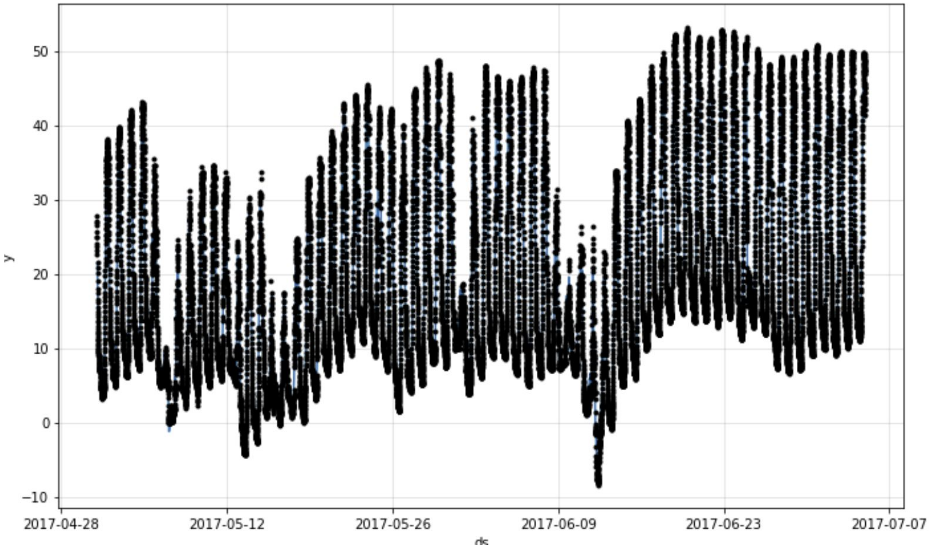
`Observed temperature in Yosemite Valley, measured every 5 min over two months.`

Yosemite Temperature - Predict next observation.

Example: Yos1

```
m = NeuralProphet(n_lags=12)
metrics = m.fit(df, freq='D')
forecast = m.predict(df)
```

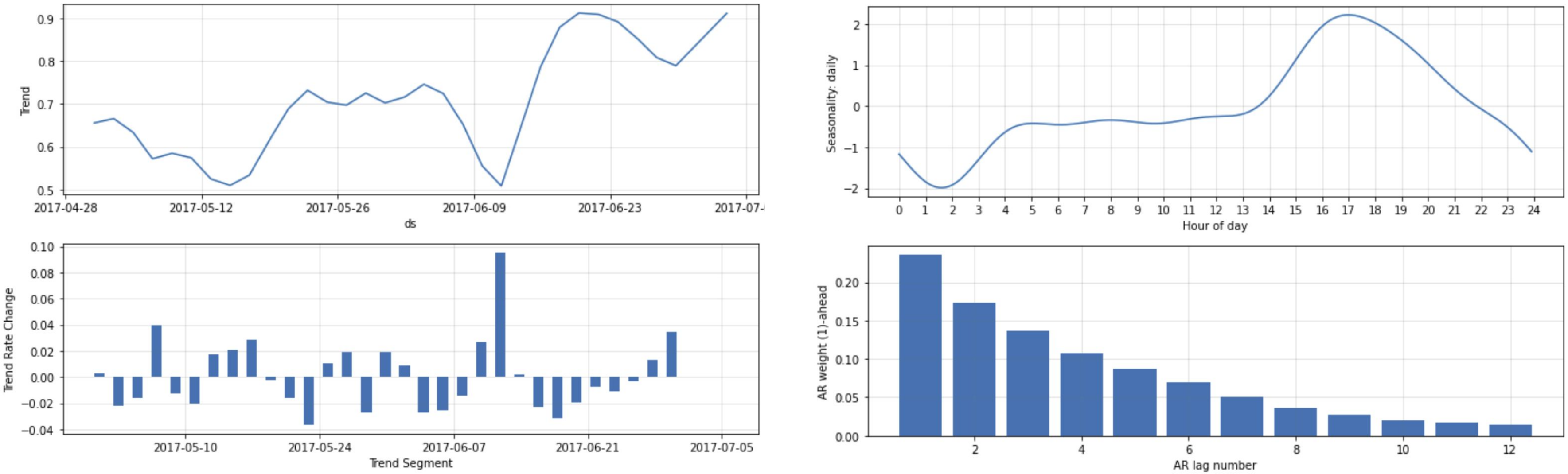
`m.plot(forecast)`



Visualize model parameters with one line.

Example: Yos1

```
m.plot_parameters()
```



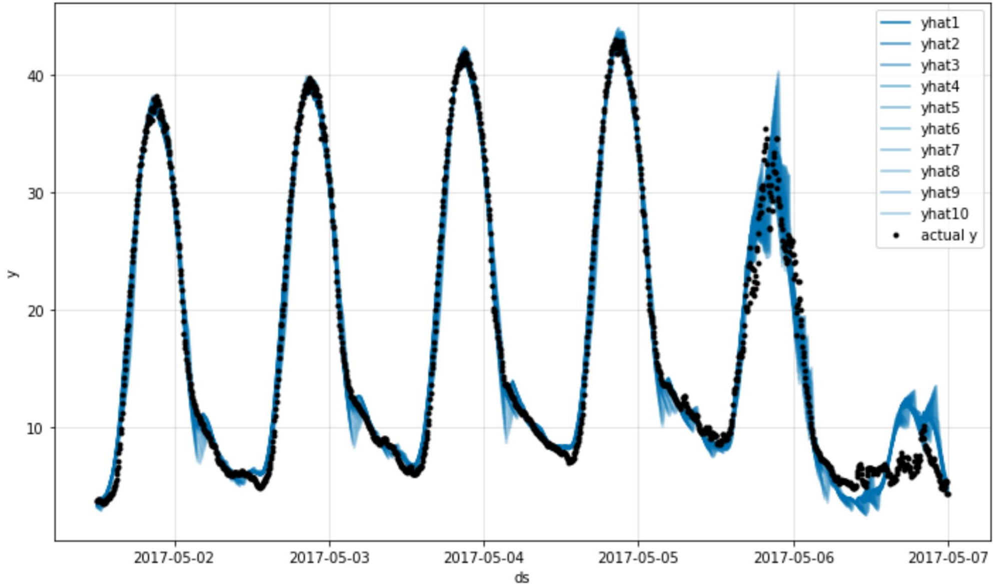
Yosemite Temperature - Predict next 36 observations. **Example: Yos36**

```
m = NeuralProphet(
      n_lags=72,
      n_forecasts=36,
)

metrics = m.fit(df, freq='D')

forecast = m.predict(df)
```

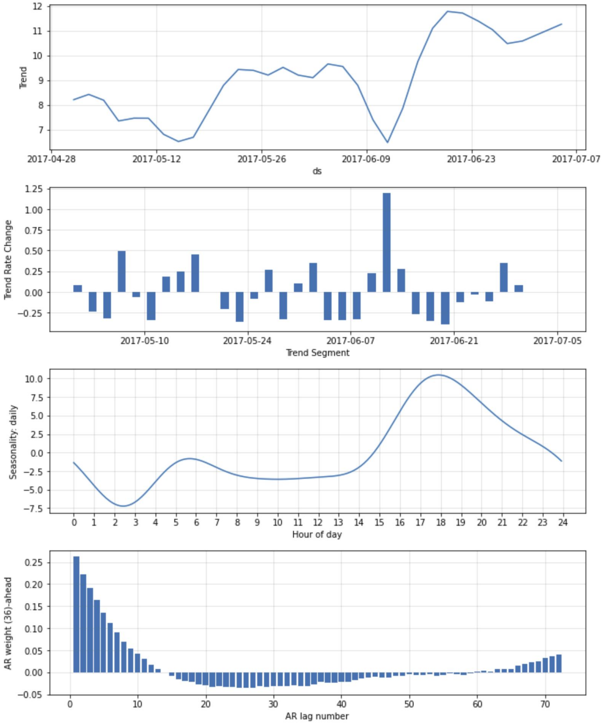
```
m.plot(forecast[:6*24*12])
```



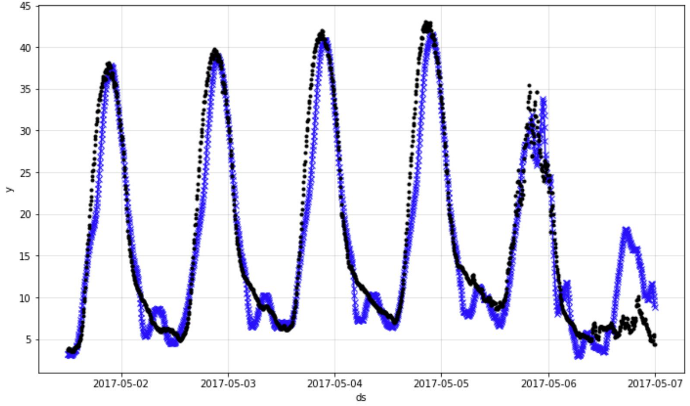
Analyze a specific forecast horizon. Sparsify.

Example: Yos36

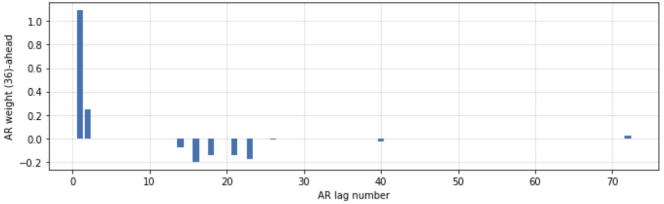
```
m.highlight_nth_step_ahead_of_each_forecast(36)
m.plot_parameters()
```




```
m.plot(forecast[:6*24*12])
```




```
NeuralProphet(ar_sparsity=0.1, ...)
```



We are working hard to extend the framework. Join us!

Extensions [upcoming]

- Hierarchical Forecasting
  & Global Modelling
- Quantifiable and Explainable Uncertainty
- Anomaly Prediction
  & Semi-Supervised Learning
- Attention: Automatic Multimodality
  & Dynamic Feature Importance

Improvements [upcoming]

- Improved NN
- Faster Training Time
  & GPU support
- Improved UI
- Diagnostic Tools for Deep Dives

`Anything trainable by gradient descent can be added as module`

NeuralProphet is a modern Prophet with a superset of its features.

Task	Prophet	NeuralProphet
Very small dataset (less than 100 samples)	✓	
Large dataset (more than 1000 samples)		✓
Long range forecast (e.g. multiple years)	✓	✓
Short to medium range forecast (e.g. 1 to 1000 steps ahead)		✓
Specific forecast horizon (e.g. next 24h)		✓
Auto-correlation (dependence on previous observations)		✓
Lagged regressors (observed covariates)		✓
Non-linear dynamics		✓
Global modelling of panel dataset		✓
Fast prediction (computational inference time)		✓

Conclusion: Upgrade Prophet.

Simple and powerful forecaster,

without compromising on interpretability.

1:1 replacement for Prophet,

with many new capabilities, superior for most applications.

Nothing but Python & PyTorch,

extensible to future state-of-the-art in forecasting.

THANK YOU, dear collaborator, supporter and advisor!

Stanford University

facebook

MONASH University

TotalEnergies

Skoltech

Oskar Triebe

Hansika Hewamalage

Mateus De Castro Ribeiro

Lluvia Ochoa

Nikolay Laptev

Polina Pilyugina

Abishek Sriramulu

Ram Rajagopal

Christoph Bergmeir

Alessandro Panella

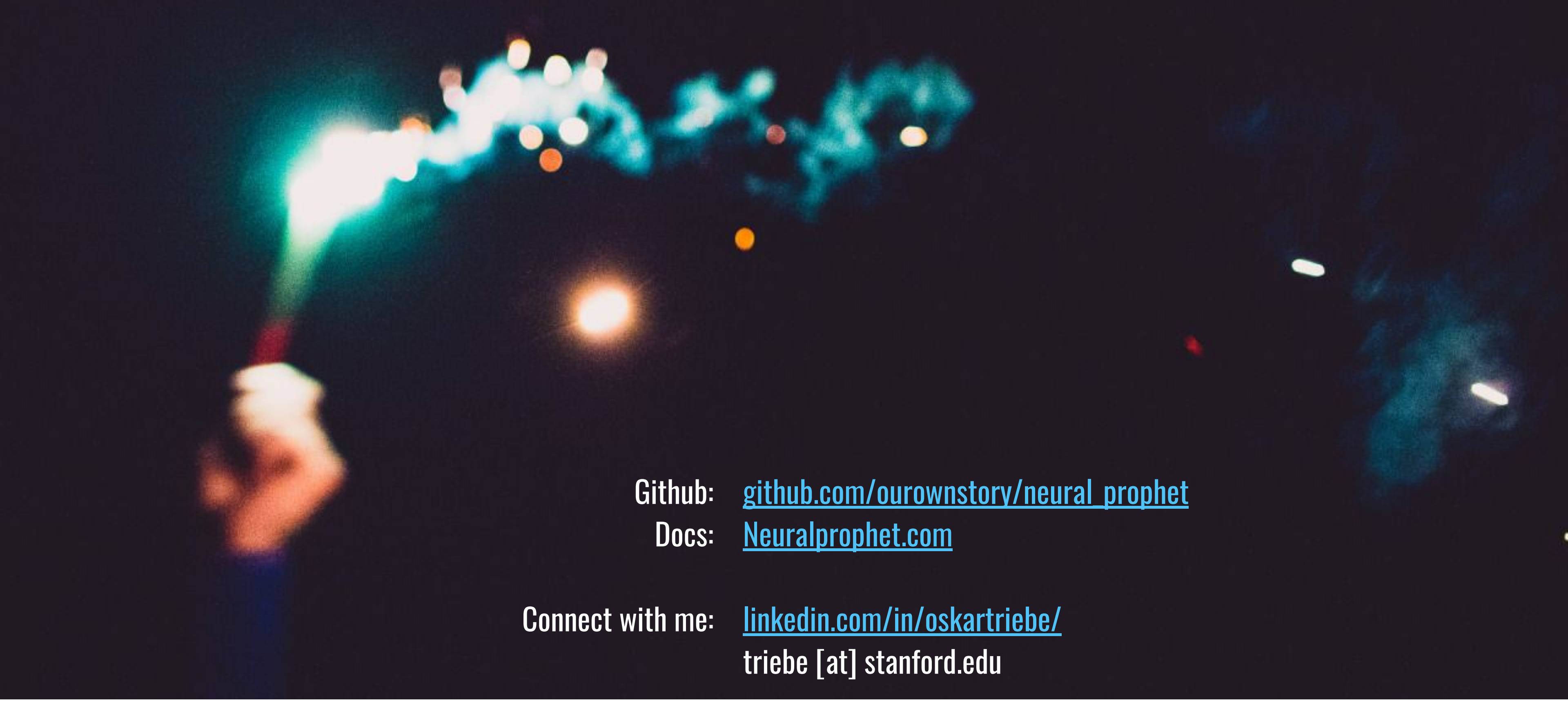
Evgeny Burnaev

Caner Komurlu

Italo Lima

Gonzague Henri

Bernhard Hausleitner



Github: github.com/ourownstory/neural_prophet

Docs: [Neuralprophet.com](https://neuralprophet.com)

Connect with me: linkedin.com/in/oskartriebe/

triebe [at] stanford.edu

Thank You