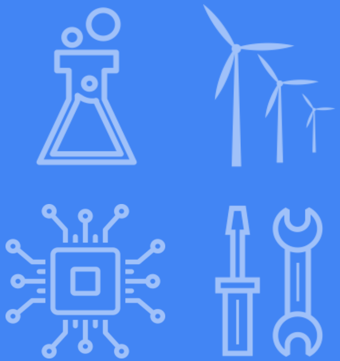
Neural Prophet

A simple time series forecasting framework

Oskar Triebe
Stanford University
Oct 12th, 2021

A time series is

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



When to use NeuralProphet

Task:	Forecasting. Predict future of observed variable.
Data:	100 to millions of samples. Unidistant, real-valued.
Dynamics:	Must: Future values depend on past values. Ideal: Seasonal, trended, repeating events, correlated variables.
Applications:	Human behavior, energy, traffic, sales, environment, server load, ...

Motivation

The Neural & Prophet

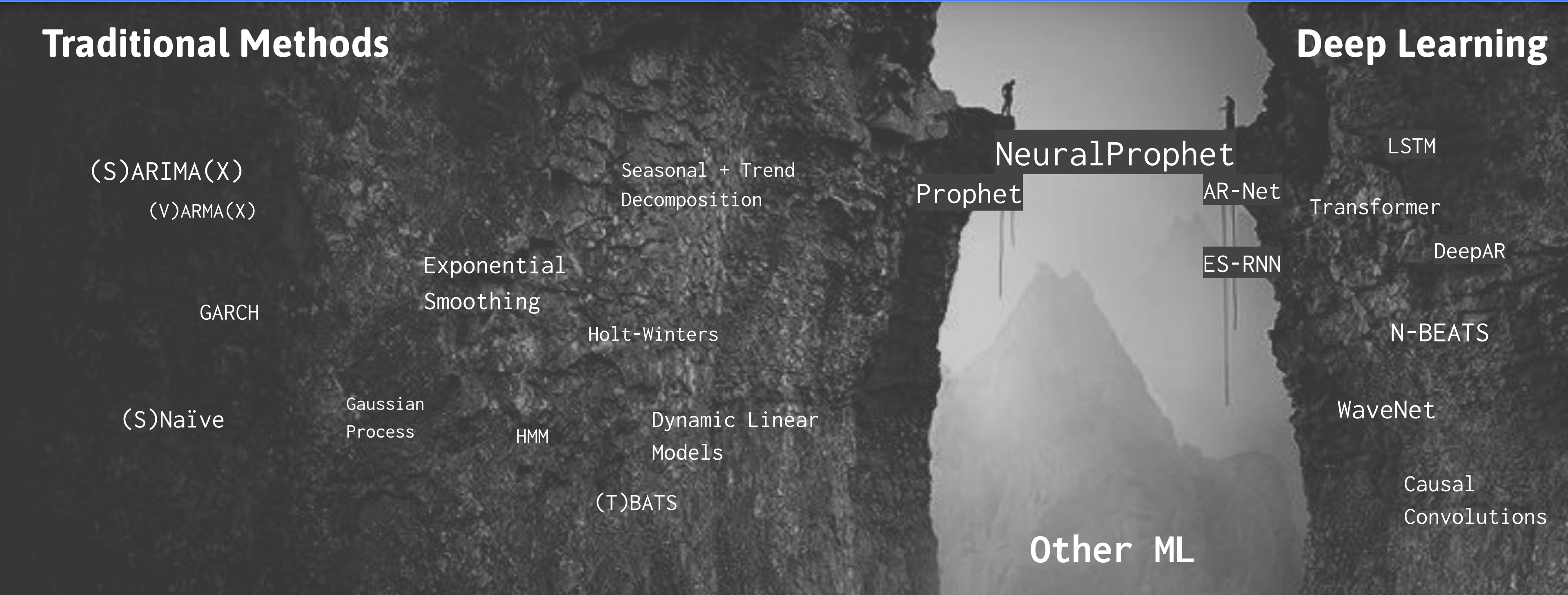
Model Components

Model Use

Example

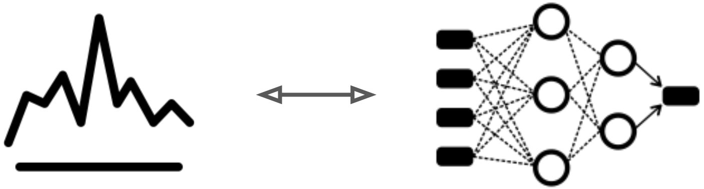
Outlook

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



Chasm

Time series - applied ML



Need expertise in both domains

Bridge

NeuralProphet

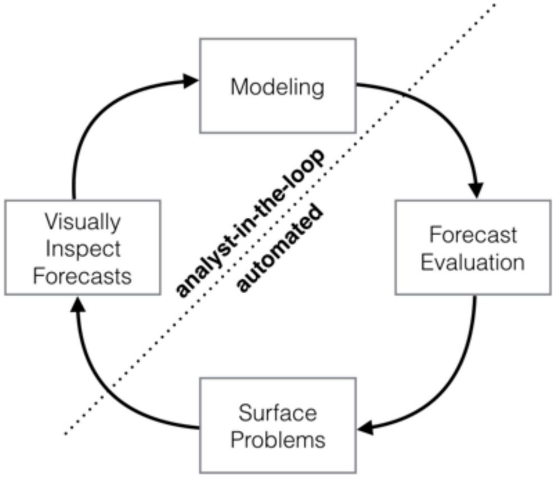
Abstracts time series and ML knowledge

Facebook Prophet is the most used forecasting package since 2017.

Quick from data to predictions.

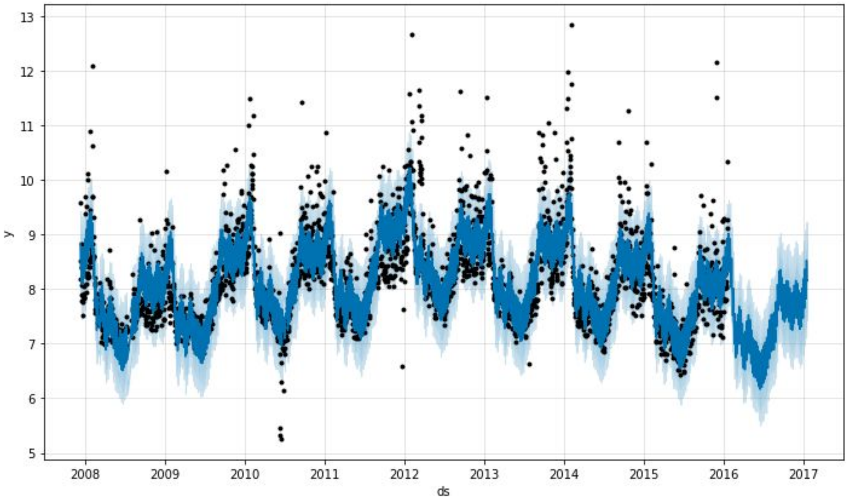
Gentle learning curve.

Customizable.



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

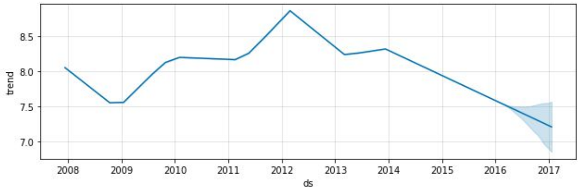
Prophet is an interpretable and decomposable model.



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

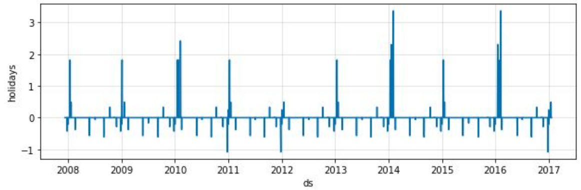
Trend

$g(t)$



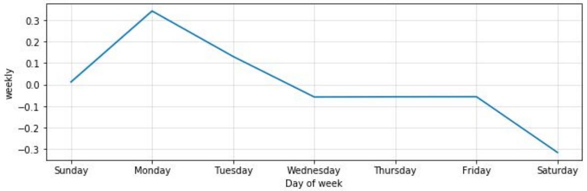
Events

$h(t)$



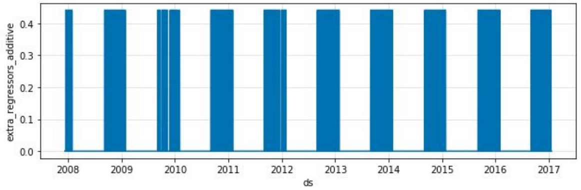
Seasonality

$s(t)$



Regressors

$X(t)$



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)

Neural Prophet

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

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*

Trend & Seasonality are identical to Prophet.

Piecewise linear trend

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

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

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

$$-s_j\delta_j$$

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

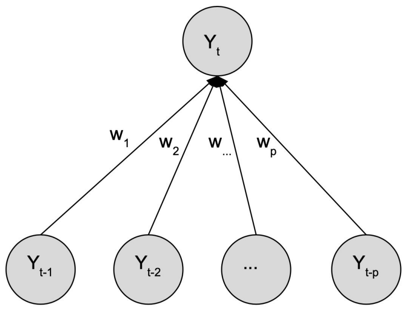
- Single weight
- Real-valued regressor
- Optional multiplicative mode

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

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 sparsification

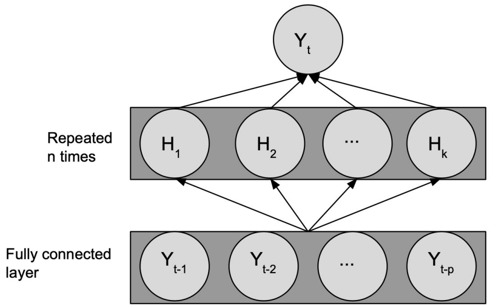


AR-Net(0)

Interpretable

(Lagged) Covariates

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



AR-Net(n)

Non-linear modeling

A user-friendly Python package

```
m = NeuralProphet()
```

Gentle learning curve.

Get results first, learn, and improve.

NeuralProphet has smart defaults.

Advanced features are optional.

```
growth="linear",
changepoints=None,
n_changepoints=10,
changepoints_range=0.9,
trend_reg=0,
trend_reg_threshold=False,
yearly_seasonality="auto",
weekly_seasonality="auto",
daily_seasonality="auto",
seasonality_mode="additive",
seasonality_reg=0,
n_forecasts=1,
n_lags=0,
num_hidden_layers=0,
d_hidden=None,
ar_sparsity=None,
learning_rate=None,
epochs=None,
batch_size=None,
loss_func="Huber",
optimizer="AdamW",
train_speed=None,
normalize="auto",
impute_missing=True,
```

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:

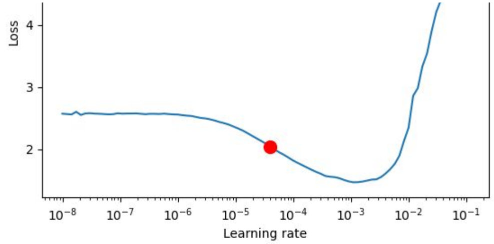
Name	Normalization Procedure
'auto'	'minmax' if binary, else 'soft'
'off'	bypasses data normalization
'minmax'	scales the minimum value to 0.0 and the maximum value to 1.0
'standardize'	zero-centers and divides by the standard deviation
'soft'	scales the minimum value to 0.0 and the 95th quantile to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

Hyperparameters have smart defaults.

Loss Function is Huber loss,
unless user-defined.

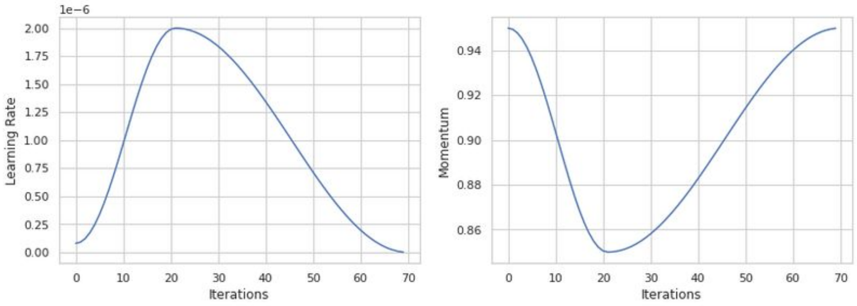
$$L_{huber}(y, \hat{y}) = \begin{cases} \frac{1}{2\beta}(y - \hat{y})^2, & \text{for } |y - \hat{y}| < \beta \\ |y - \hat{y}| - \frac{\beta}{2}, & \text{otherwise} \end{cases}$$

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.



There is a utility for that.

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
- ...

Examples

Yosemite Temperature prediction with Trend, Seasonality and Auto-Regression:

- 1 step ahead
- 36 steps ahead (with extras: AR Sparsity, 24 hours ahead)

Yosemite Temperature - Predict next observation.

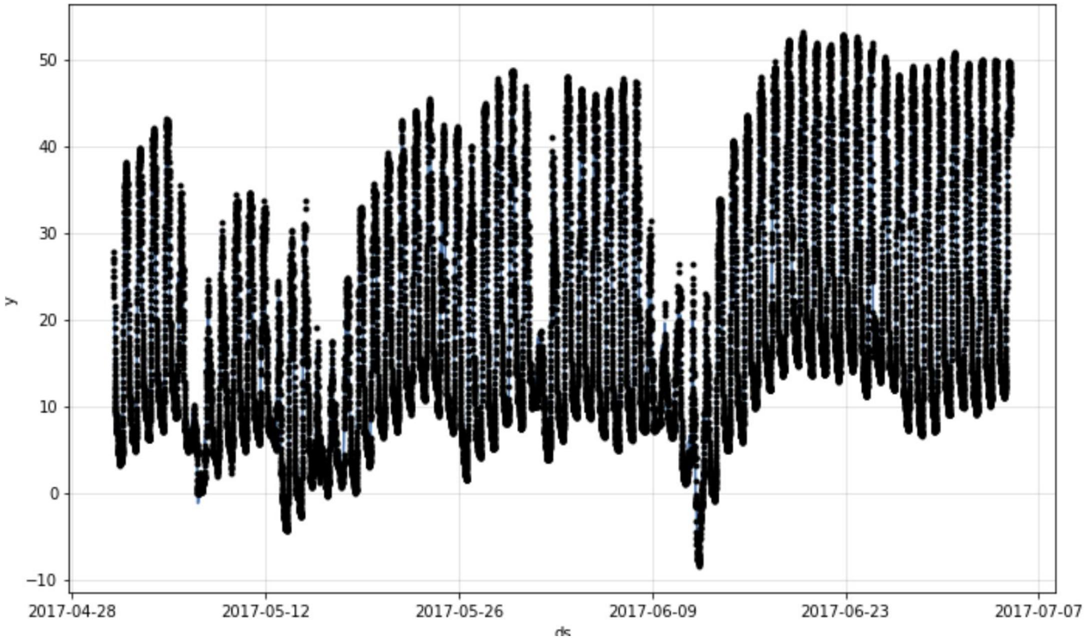
Example: Yos1

```python
import pandas as pd
from neuralprophet import NeuralProphet, set_log_level
# set_log_level("ERROR")
df = pd.read_csv(data_location + "example_data/yosemite_temps.csv")
```

```python
m = NeuralProphet(
    n_lags=12,
    changepoints_range=0.95,
    n_changepoints=30,
    weekly_seasonality=False,
    batch_size=64,
    epochs=10,
    learning_rate=1.0,
)
metrics = m.fit(df, freq='5min')
```

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

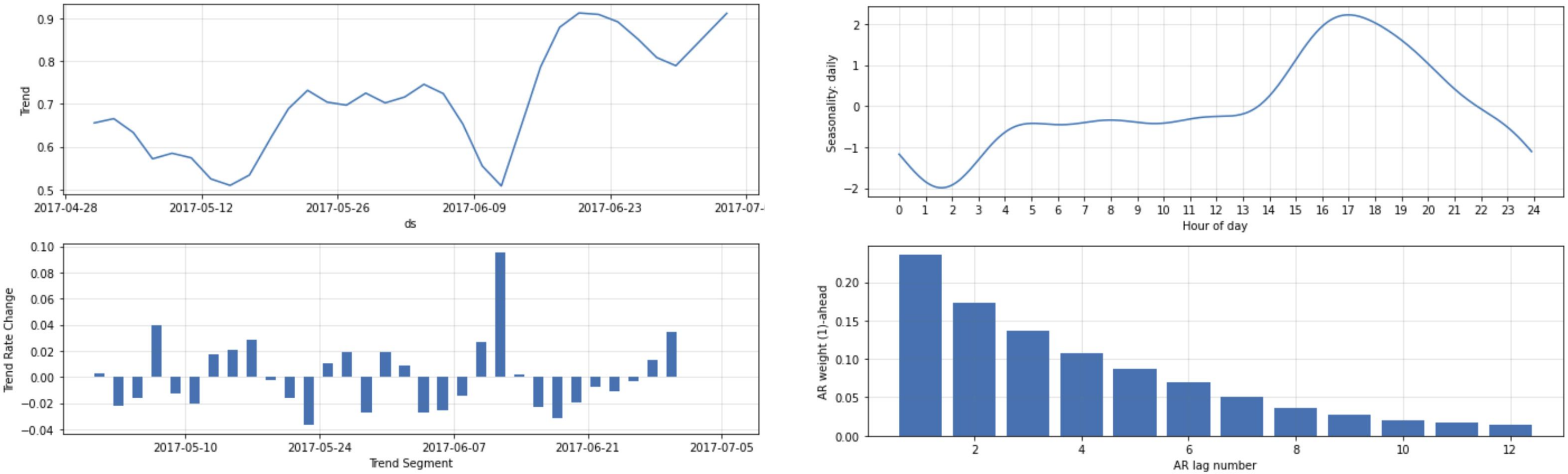
```python
future = m.make_future_dataframe(df, n_historic_predictions=True)
forecast = m.predict(future)
fig = m.plot(forecast)
```



Visualize model parameters with one line.

Example: Yos1

```
m.plot_parameters()
```

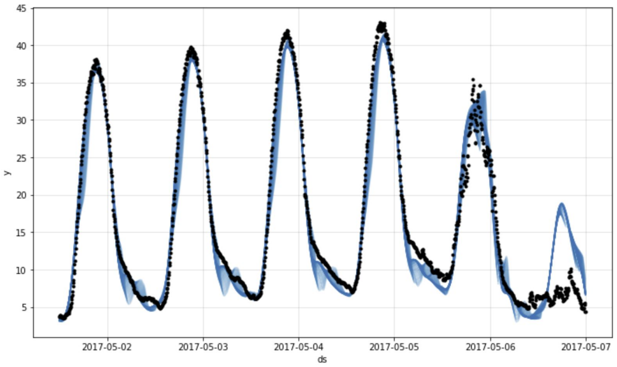


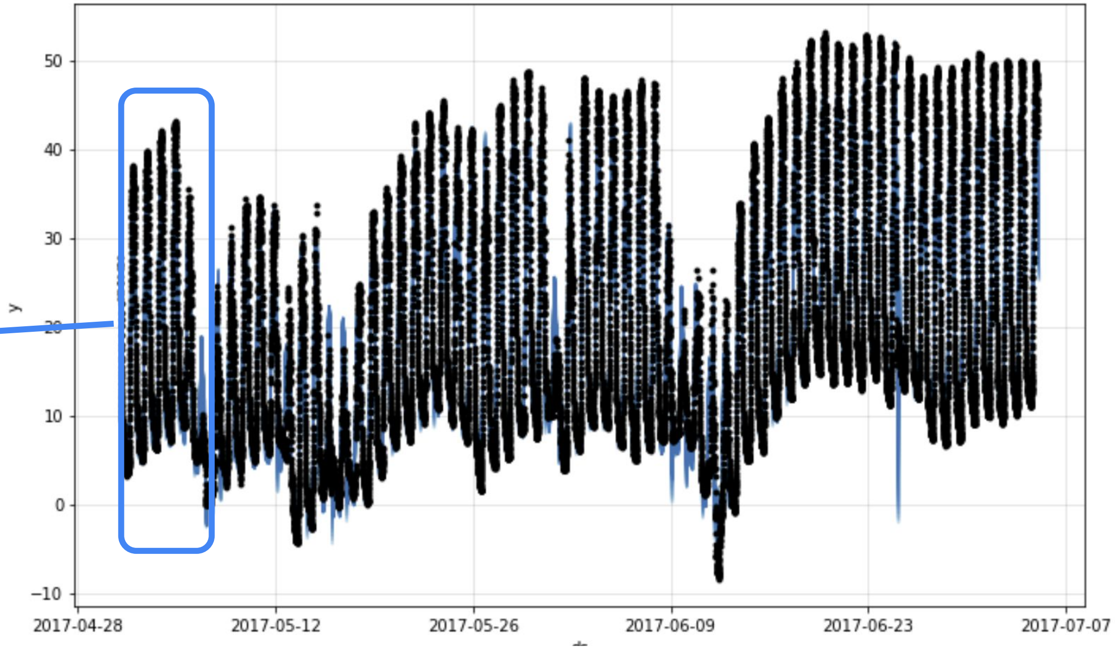
Yosemite Temperature - Predict next 36 observations.

Example: Yos36

```
m = NeuralProphet(
    n_lags=6*12,
    n_forecasts=3*12,
    changepoints_range=0.95,
    n_changepoints=30,
    weekly_seasonality=False,
    batch_size=64,
    epochs=10,
    learning_rate=1.0,
)
```

```
metrics = m.fit(df, freq='5min')
future = m.make_future_dataframe(df, n_historic_predictions=True)
forecast = m.predict(future)
fig = m.plot(forecast)
```



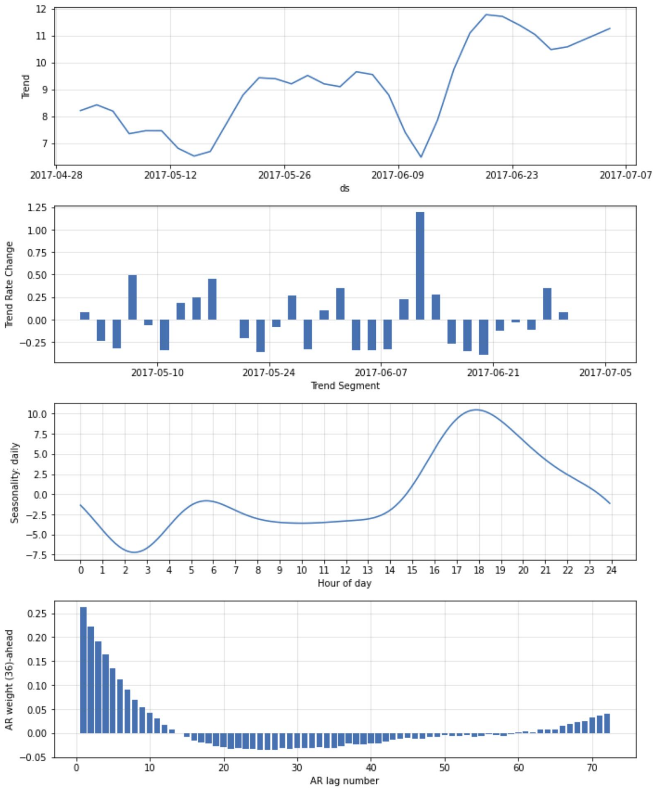


Dataset:
Observed temperature in Yosemite Valley, measured every 5 min over two months.

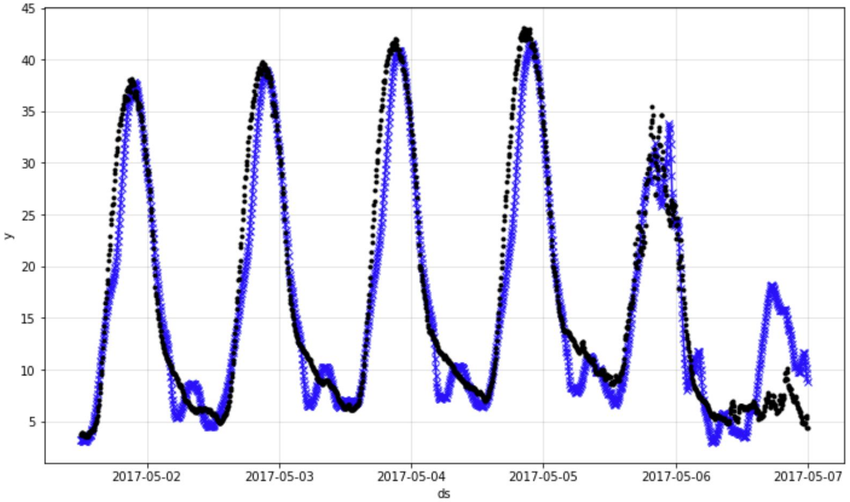
Analyze a specific forecast horizon. Sparsify.

Example: Yos36

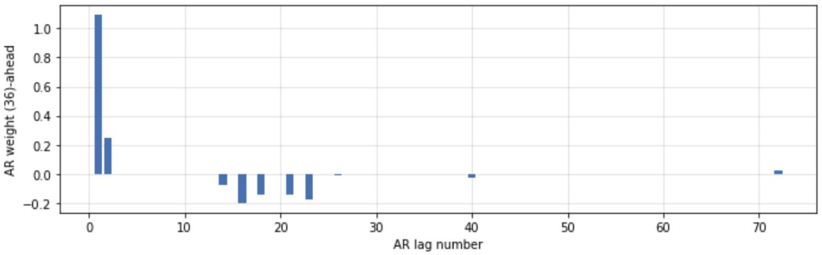
```
m = m.highlight_nth_step_ahead_of_each_forecast(3*12)
fig_param = m.plot_parameters()
```




```
fig = m.plot(forecast[144:6*288])
```



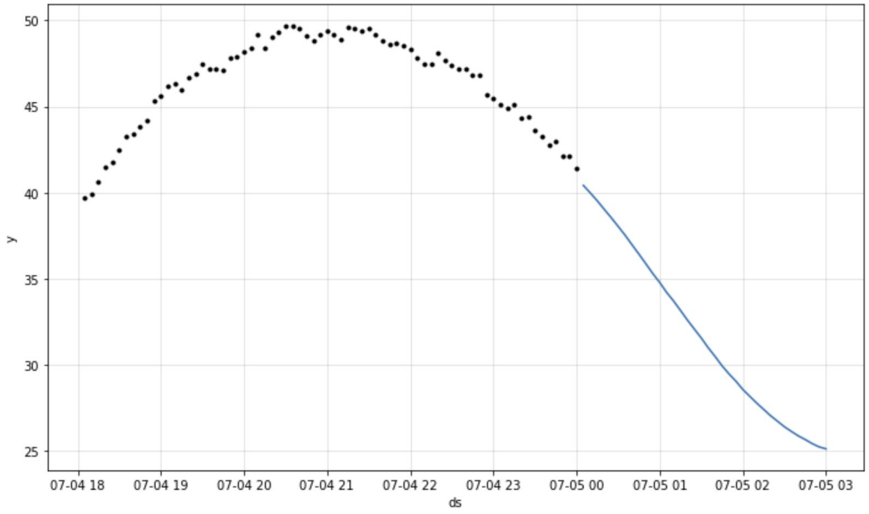

```
ar_sparsity=0.1,
```



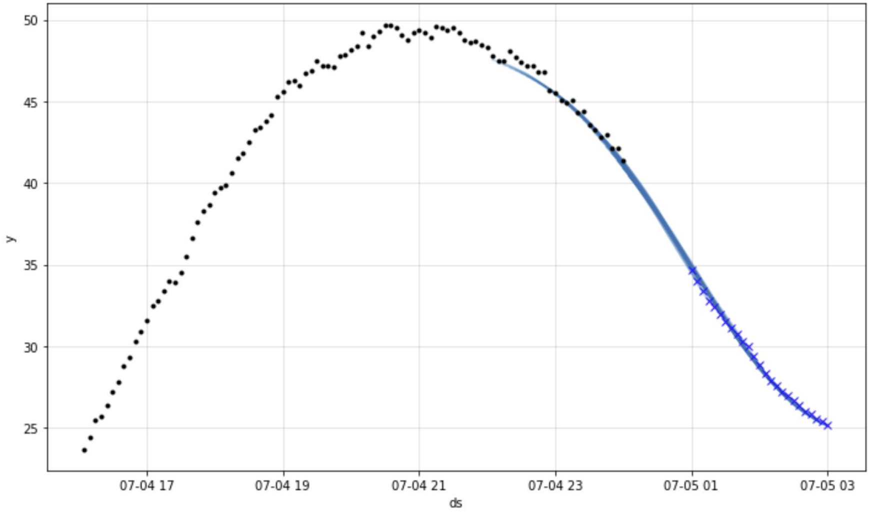
Predict. See how new data impacts the forecast.

Example: Yos36

```
m = m.highlight_nth_step_ahead_of_each_forecast(None) # reset highlight
fig = m.plot_last_forecast(forecast)
```




```
m = m.highlight_nth_step_ahead_of_each_forecast(3*12)
fig = m.plot_last_forecast(forecast, include_previous_forecasts=2*12)
```



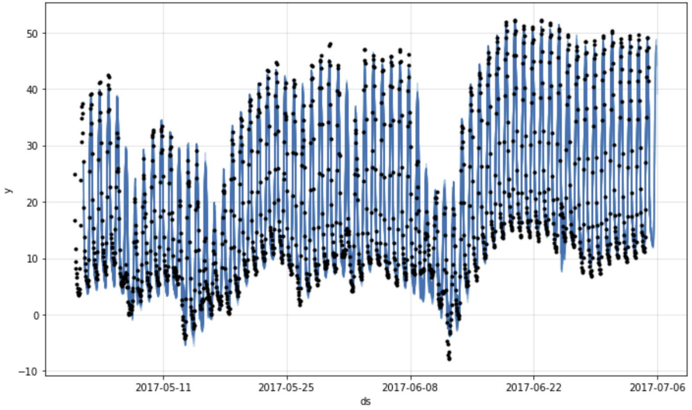
Want to forecast a larger horizon?

Example: Yos24

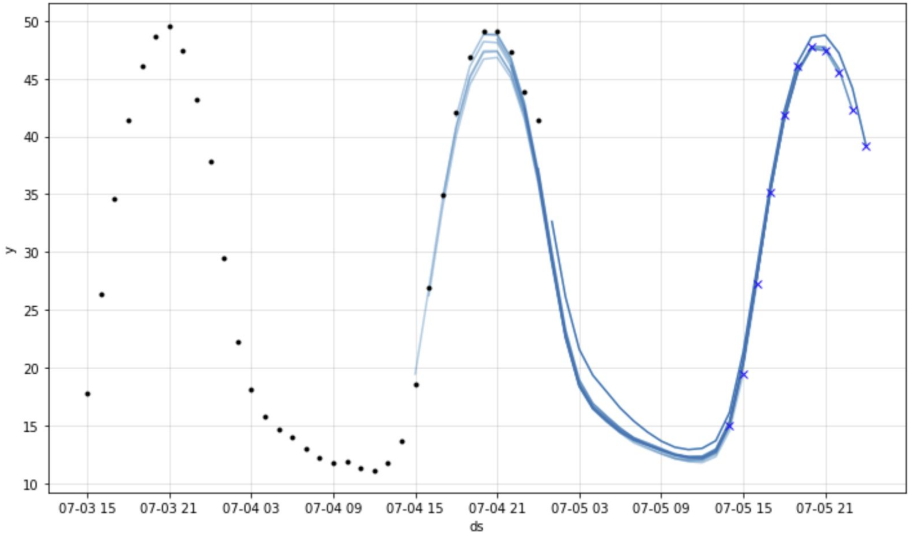
```
df.loc[:, "ds"] = pd.to_datetime(df.loc[:, "ds"])
df_hourly = df.set_index('ds', drop=False).resample('H').mean().reset_index()
len(df_hourly)
```

```
1561
```

```
m = NeuralProphet(
    n_lags=24,
    n_forecasts=24,
    changepoints_range=0.95,
    n_changepoints=30,
    weekly_seasonality=False,
    learning_rate=0.3,
)
metrics = m.fit(df_hourly, freq='H')
future = m.make_future_dataframe(df_hourly, n_historic_predictions=True)
forecast = m.predict(future)
fig = m.plot(forecast)
```




```
m = m.highlight_nth_step_ahead_of_each_forecast(24)
fig = m.plot_last_forecast(forecast, include_previous_forecasts=10)
```



Outlook

We are extending the framework to suit more forecasting needs.

Pull Request pending:

- Global modelling
- Quantile estimation (Uncertainty Interval)
- Classification
- Better documentation

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

Extensions [upcoming]

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

Improvements [upcoming]

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

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

Upgrade Prophet.

Task	Prophet	NeuralProphet
Very small dataset	✓	
Very large dataset		✓
Long range forecast (e.g. multiple years)	✓	✓
Short to medium range forecast (e.g. 1 to 1000 step 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 time (computationally)		✓

THANK YOU, dear collaborator, supporter and advisor!

Stanford University

TotalEnergies

facebook

MONASH University

Skoltech

Oskar Triebe

Hansika Hewamalage

Polina Pilyugina

Lluvia Ochoa

Nikolay Laptev

Mateus De Castro Ribeiro

Gonzague Henri

Ram Rajagopal

Christoph Bergmeir

Alessandro Panella

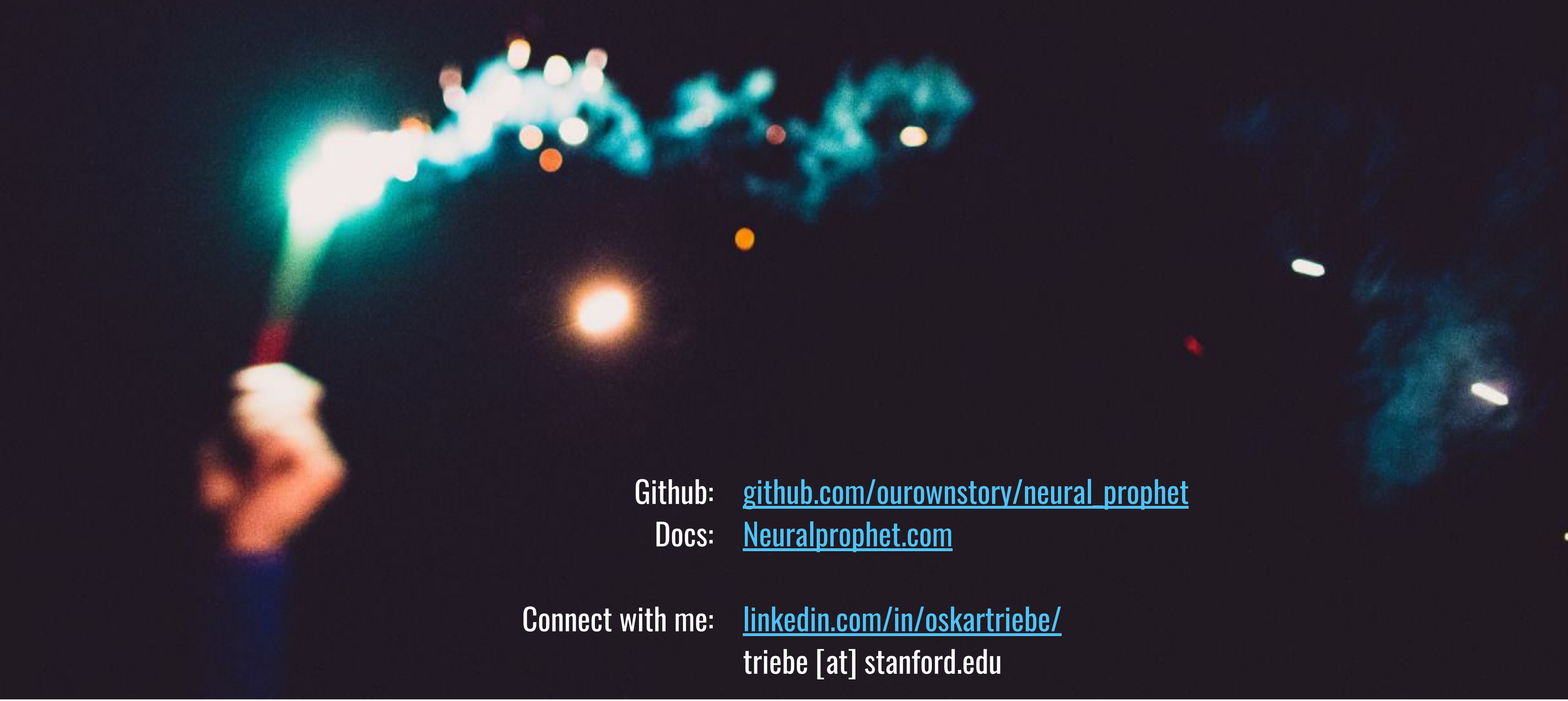
Evgeny Burnaev

Caner Komurlu

Italo Lima

Abishek Sriramulu

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

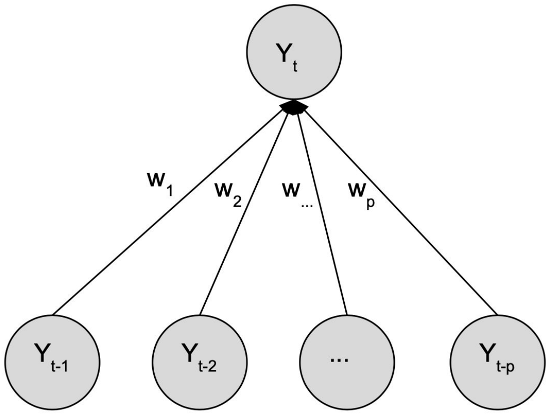
Thank You

Appendix: AR-Net

AR-Net is a Neural Network for autoregressive time series.

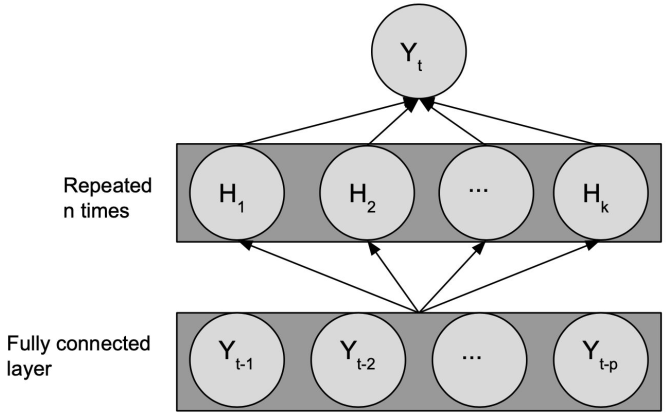
AR-Net(0)

Interpretable



AR-Net(n)

Stronger modeling ability



Skip estimating the AR process order.

Model: AR-Net

$$y_t = c + \sum_{i=1}^{i=p} w_i * y_{t-i} + e_t$$

$\downarrow$

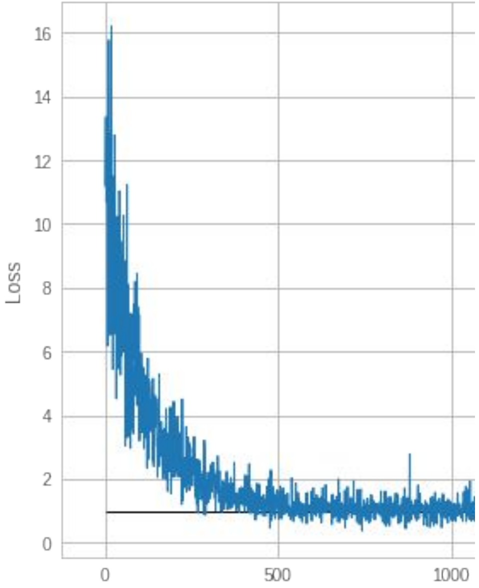
$$min_\theta \quad L(y, \hat{y}, \theta) + \lambda(s) \cdot R(\theta)$$

$$\lambda(s) = c_\lambda \cdot (s^{-1} - 1)$$

$$s = \frac{\hat{p}_{data}}{p_{model}}$$

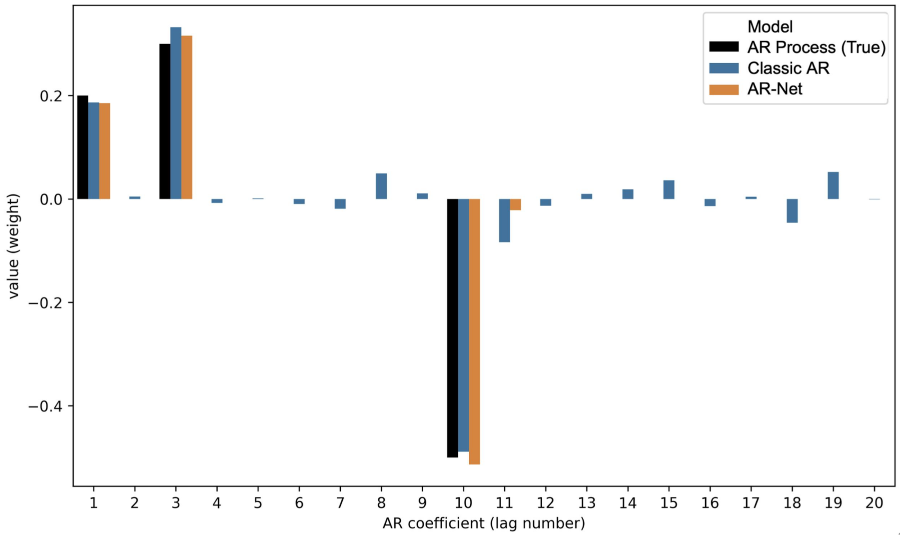
$$c_\lambda \approx \frac{\sqrt{\hat{L}}}{100}$$

$$R(\theta) = \frac{1}{p} \sum_{i=1}^{p} \frac{2}{1 + \exp(-c_1 \cdot |\theta_i|^{\frac{1}{c_2}})} - 1$$

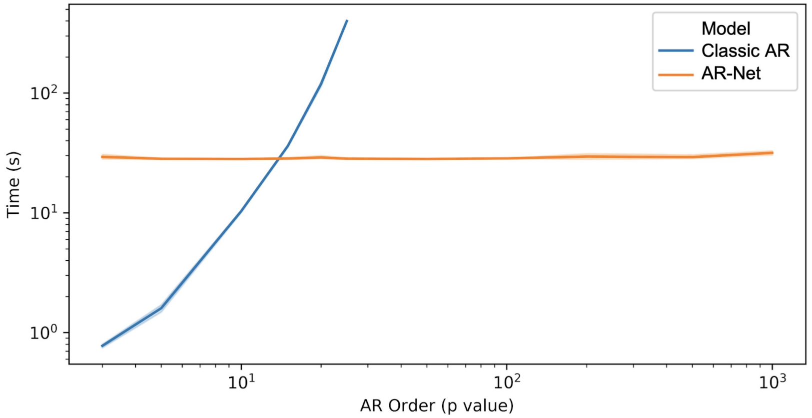


Trained with SGD (Adam)

Automatic AR-lag selection, yet faster.



Automatic Sparsity

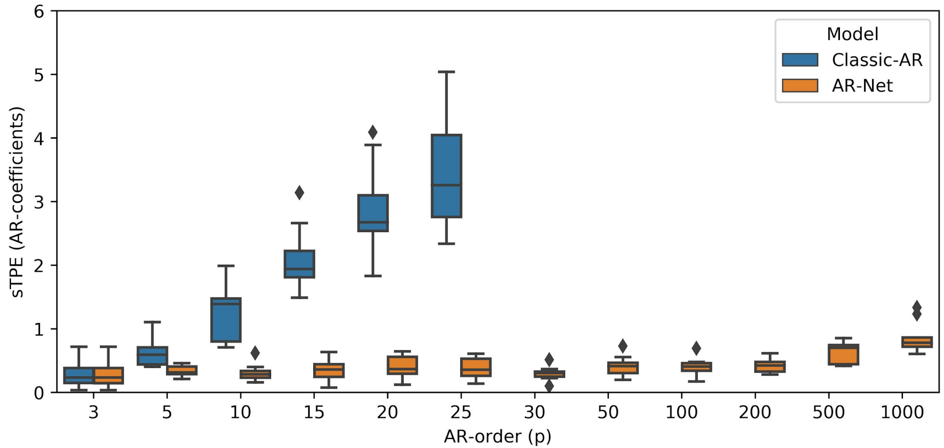


Quadratically faster

Sparse AR-Net surpasses Classic AR and scales to large orders.

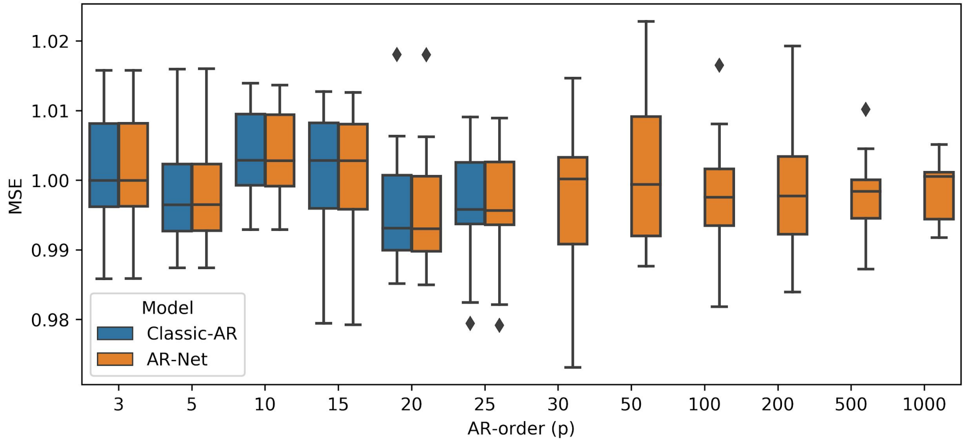
Closeness to true coefficients

$$sTPE = 100 \cdot \frac{\sum_{i=1}^{i=p} |\hat{w}_i - w_i|}{\sum_{i=1}^{i=p} |\hat{w}_i| + |w_i|}$$



MSE loss on forecast target

$$MSE = \frac{1}{n} \sum_{1}^{n} (y_t - \hat{y}_t)^2$$



Appendix: Model Use Details

Few lines from data to forecast.

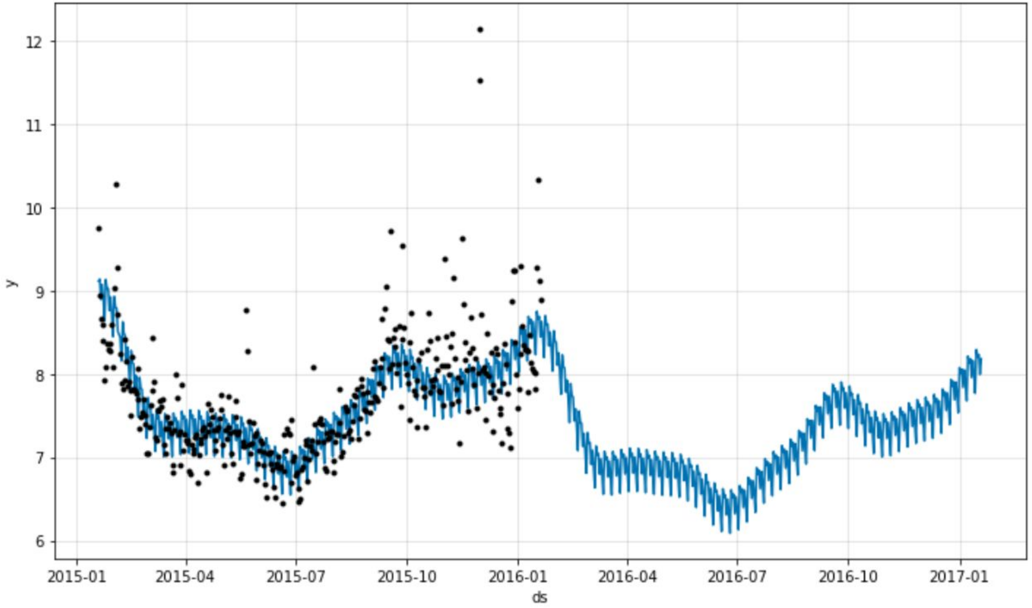
Hands-on

skip data preparation: → Just create a DataFrame with desired columns

```python
import pandas as pd
from neuralprophet.neural_prophet import NeuralProphet
```

```python
df = pd.read_csv('../data/example_wp_log_peyton_manning.csv')
```

```python
# linear time-dependent model
m = NeuralProphet()
metrics = m.fit(df)
future = m.make_future_dataframe(df, future_periods=365)
forecast = m.predict(future)
fig_fcst = m.plot(forecast[-730:])
```



Validate model performance without extra lines.

Hands-on

```
# or evaluate while training
m = NeuralProphet()
metrics = m.fit(df, validate_each_epoch=True, valid_p=0.2)
metrics.tail()
```

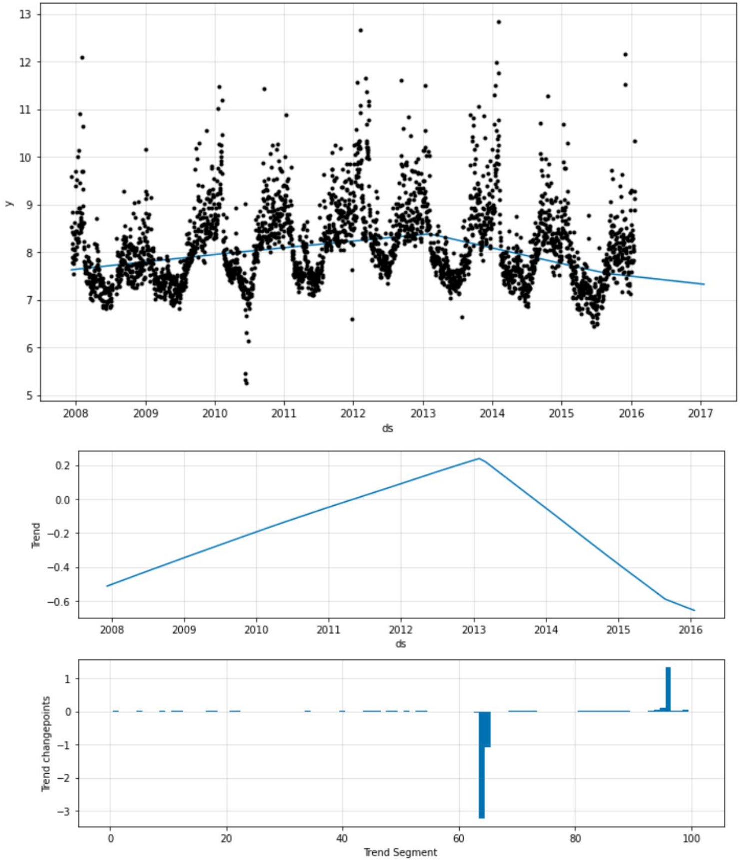
```
Disabling daily seasonality. Run NeuralProphet with daily_seasonality=True to override this.
```

	SmoothL1Loss	MAE	RegLoss	SmoothL1Loss_val	MAE_val
35	0.163102	0.371323	0.0	0.485371	0.779465
36	0.161851	0.369609	0.0	0.368921	0.648736
37	0.161122	0.369219	0.0	0.366328	0.648230
38	0.168598	0.376638	0.0	0.348269	0.627886
39	0.167961	0.375777	0.0	0.362161	0.642699

```
# split manually
m = NeuralProphet()
df_train, df_val = m.split_df(df, valid_p=0.2)
train_metrics = m.fit(df_train)
val_metrics = m.test(df_val)
```

Trend

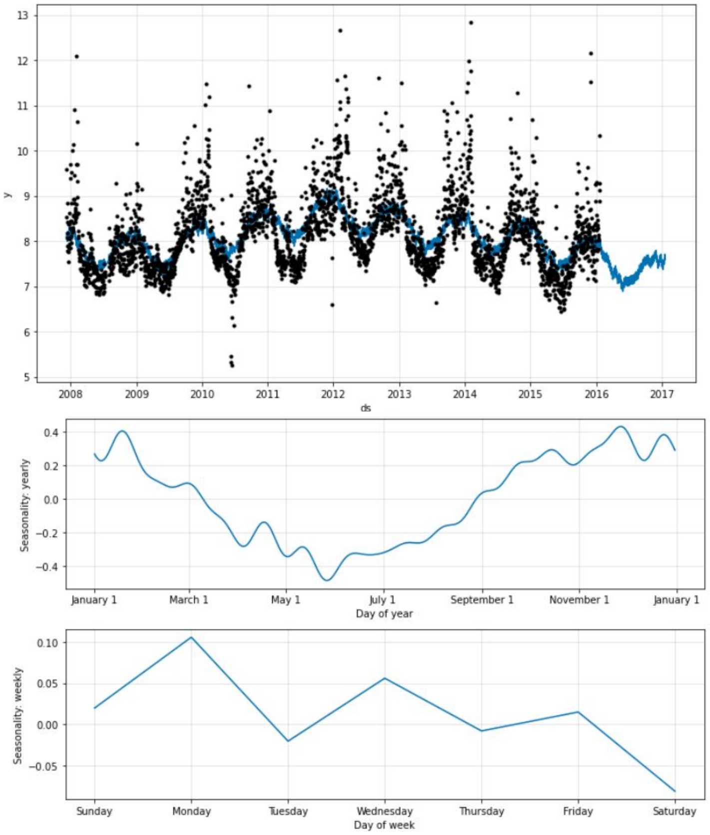

```
m = NeuralProphet(
    n_changepoints=100,
    trend_smoothness=2,
    yearly_seasonality=False,
    weekly_seasonality=False,
    daily_seasonality=False,
)
metrics = m.fit(df)
```



Seasonality

Hands-on

```
m = NeuralProphet(
    yearly_seasonality=16,
    weekly_seasonality=8,
    daily_seasonality=False,
    seasonality_reg=1,
)
metrics = m.fit(df)
```



Events can be added in different forms.

Hands-on

```
superbowls = pd.DataFrame({'event': 'superbowl',
'ds': pd.to_datetime(['2010-02-07', '2014-02-02', '2016-02-07'])})
```

```
events_df = pd.concat((superbowls, playoffs))
```

```
m = NeuralProphet()
```

```
m = m.add_country_holidays("US")
```

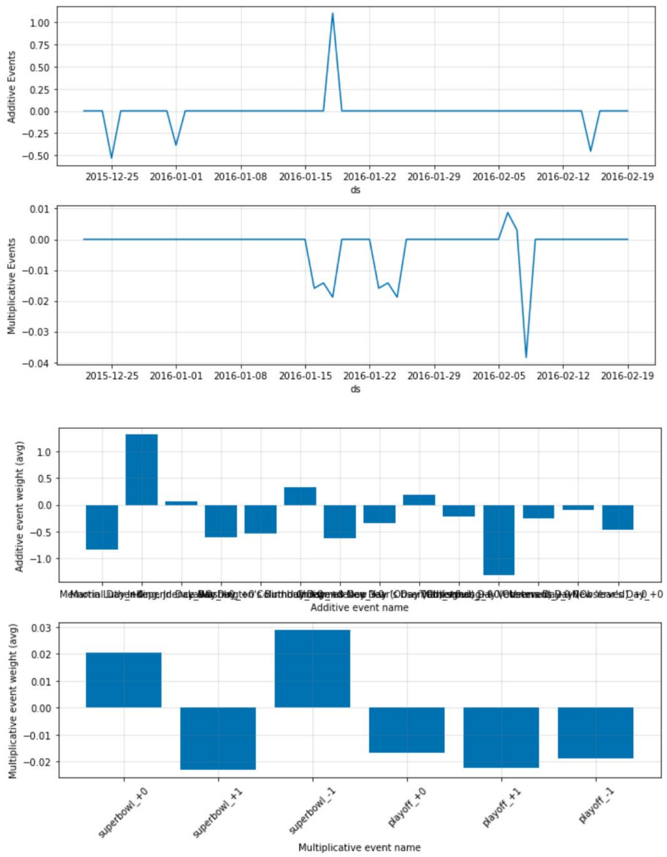
```
m = m.add_events(
    ["superbowl", "playoff"],
    lower_window=-1,
    upper_window=1,
    mode="multiplicative",
    regularization=0.5
)
```

```
history_df = m.create_df_with_events(df, events_df)
metrics = m.fit(history_df)
```

```
Disabling daily seasonality. Run NeuralProphet with daily_seasonality=
True to override this.
```

```
future = m.make_future_dataframe(
    history_df,
    events_df,
    future_periods=30,
    n_historic_predictions=30
)
```

```
forecast = m.predict(future)
```



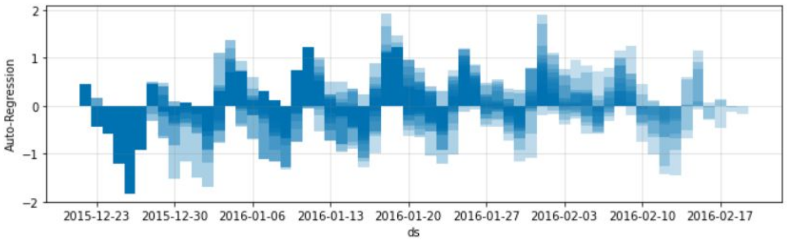
Auto-Regression


```
m = NeuralProphet(
    n_forecasts=30,
    n_lags=60,
    ar_sparsity=0.1,
    yearly_seasonality=False,
    weekly_seasonality=False,
    daily_seasonality=False,
)
```

Autoregression, here with sparsity

```
metrics = m.fit(df)
future = m.make_future_dataframe(df, n_historic_predictions=30)
forecast = m.predict(future)
```

```
fig_comp = m.plot_components(forecast[60:])
```



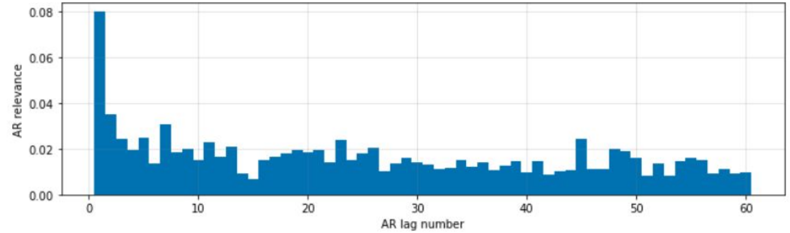
Recommended example notebook:

https://github.com/ourownstory/neural_prophet/blob/master/example_notebooks/autoregression_yosemite_temps.ipynb

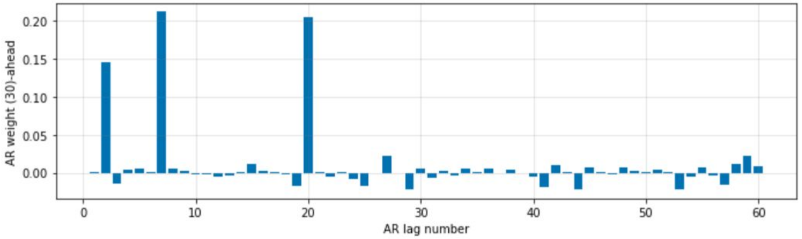
Focus on a specific forecast.

Hands-on

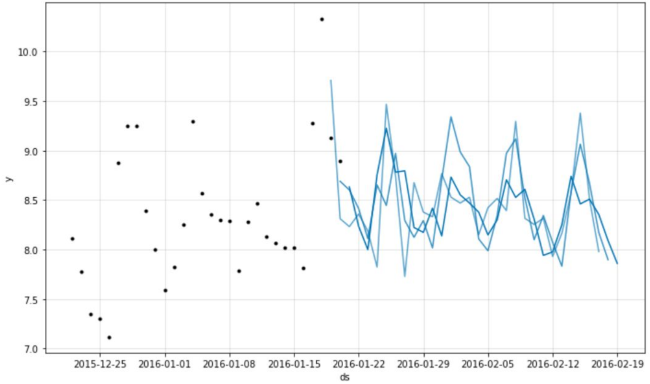
```
fig_param = m.plot_parameters()
```

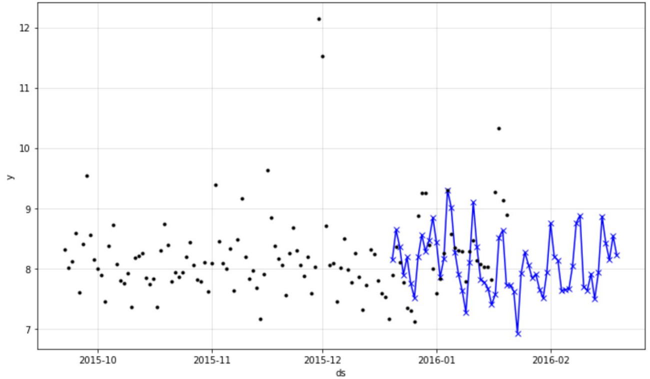



```
fig_param30 = m.highlight_nth_step_ahead_of_each_forecast(30).plot_parameters()
```




```
fig_prediction = m.plot_last_forecast(forecast[60:],
include_previous_forecasts=2)
```





Lagged Covariates & Future Regressors

Hands-on

```
m = NeuralProphet(
    n_forecasts=30,
    n_lags=60,
    ar_sparsity=0.1,
    yearly_seasonality=False,
    weekly_seasonality=False,
    daily_seasonality=False,
)
```

```
df = pd.read_csv('../data/example_wp_log_peyton_manning.csv')
df['A'] = df['y'].rolling(7, min_periods=1).mean()
df['B'] = df['y'].rolling(60, min_periods=1).mean()
m = m.add_covariate(name='A')
m = m.add_regressor(name='B')
```

```
metrics = m.fit(df)
future = m.make_future_dataframe(df, n_historic_predictions=30)
forecast = m.predict(future)
```

Covariates and regressors are similarly added

```
m = NeuralProphet(
    n_forecasts=30,
    n_lags=60,
    learning_rate=1.0,
    loss_func='Huber',
    normalize_y=True,
    num_hidden_layers=2,
    d_hidden=64,
)
```

Make it non-linear.

```
fig_comp = m.plot_components(forecast[60:])
```