Neural Prophet

Extendable and Scalable Forecasting

Oskar Triebe
Stanford University, October 26, 2020
40th International Symposium on Forecasting

Motivation

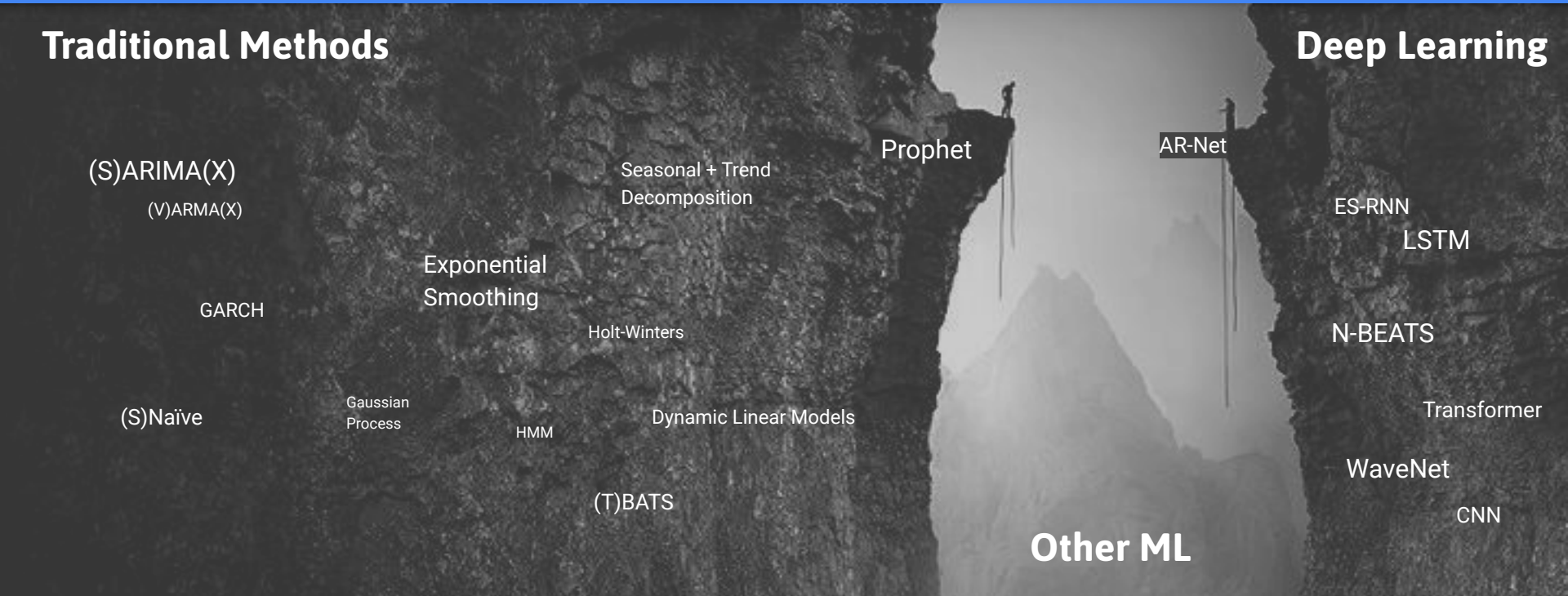
Introduction

Model

Hands-On

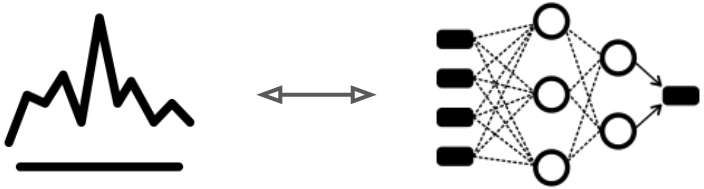
Outlook

Forecasting Landscape



Chasm

Time series - applied ML



Need expertise in both domains

Bridge

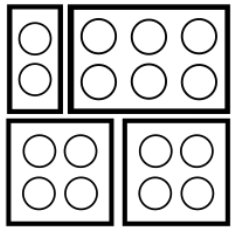
NeuralProphet

Abstracts time series and ML knowledge

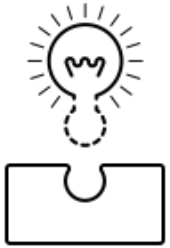
Introduction: NeuralProphet

- Beginner-friendly
- Customizable model
- Interpretable
- Scalable to large data
- Extendable code
- Workflow integration

What if: Typical tasks as one-liners.

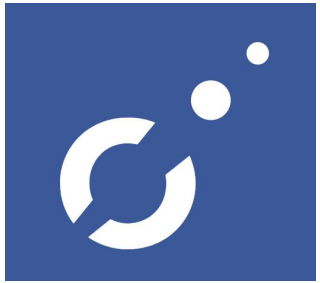


Add more data, features,
or modelling blocks



Explain model parameters,
Visualize prediction components

The original Prophet introduced Analyst-in-the-loop.

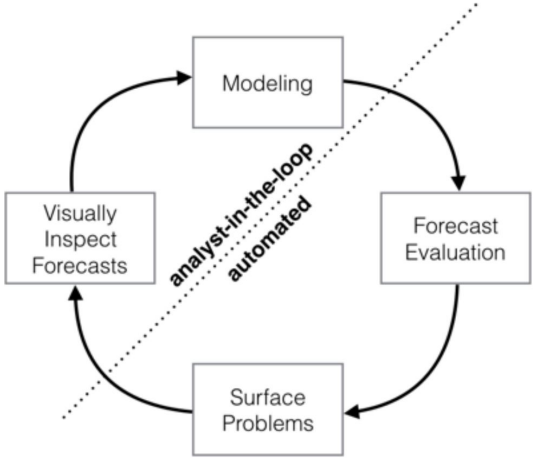


Forecasting at Scale

Sean J. Taylor*†
Facebook, Menlo Park, California, United States
sjt@fb.com

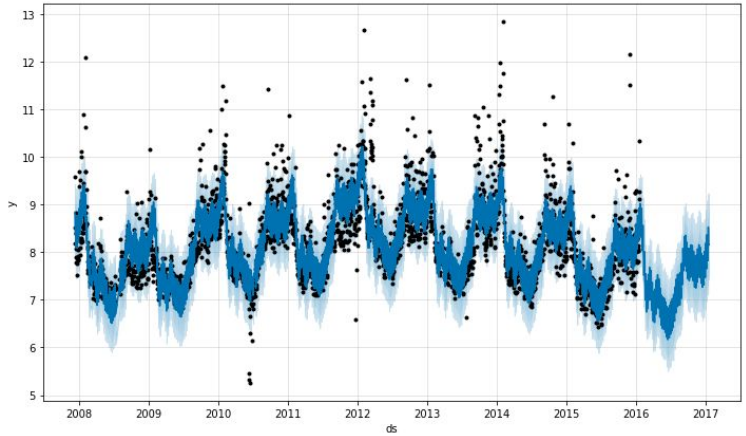
and

Benjamin Letham†
Facebook, Menlo Park, California, United States
bletham@fb.com



https://peerj.com/preprints/3190/

Prophet models interpretable time series components.



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

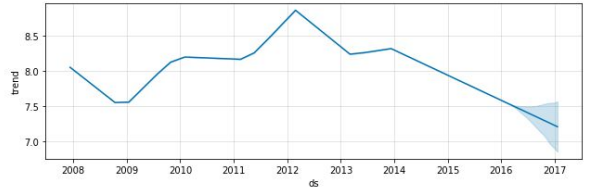
$g(t)$ Trend

$h(t)$ Events

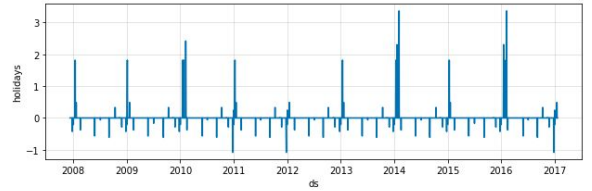
$s(t)$ Seasonality

$X(t)$ Regressors

$g(t)$

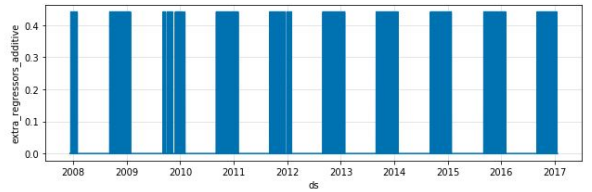


$h(t)$



$s(t)$

$X(t)$



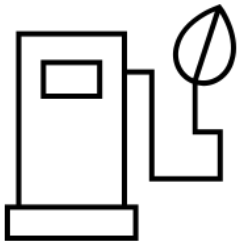
Why we need NeuralProphet.

Extensible code.

Models suited for larger datasets.

Lagged input variables.

A car is more than an engine.



Preprocessing

Diagnostics

Deployment

That’s why we include helpful tools to streamline your workflow.

Current Model Components

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

Piecewise linear trend

- N changepoints
- Segmentwise independent
- Automatic changepoint detection
- Optional logistic growth

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

$-s_j\delta_j$

Seasonality

- N Fourier terms
- Automatic yearly, weekly, daily

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

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

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

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

Events / Holidays

- Automatic for given country
- Various user-defined formats

$$h(t) = Z(t)\boldsymbol{\kappa}.$$

$$Z(t) = [\mathbf{1}(t \in D_1), \ldots, \mathbf{1}(t \in D_L)]$$

(Future-Known) Regressors

- Single weight

(Lagged) Covariates

- AR-Net (y as target)

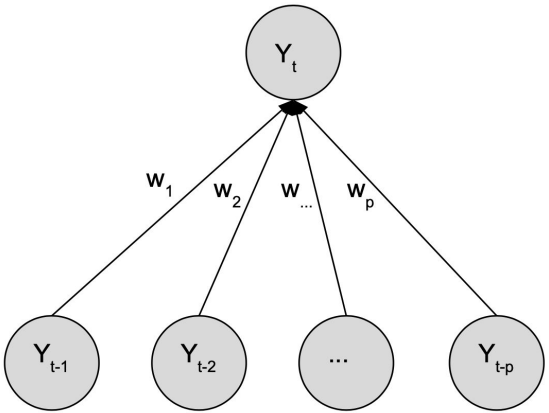
Auto-Regression

- AR-Net

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

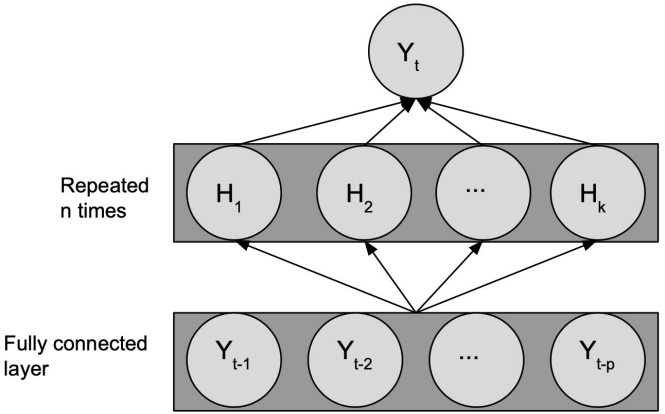
AR-Net(0)

Interpretable



AR-Net(n)

Stronger modeling ability



Skip estimating the AR process order.

Model: AR-Net

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

$$\downarrow$$

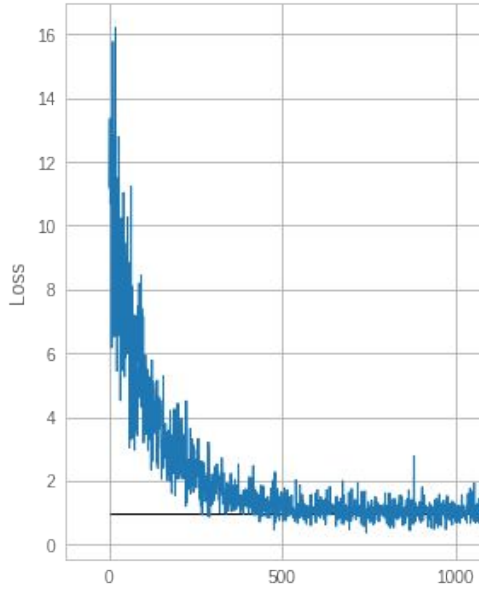
$$min_\theta \quad L(y, \hat{y}, \theta) + \lambda(s) \cdot R(\theta)$$

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

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

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

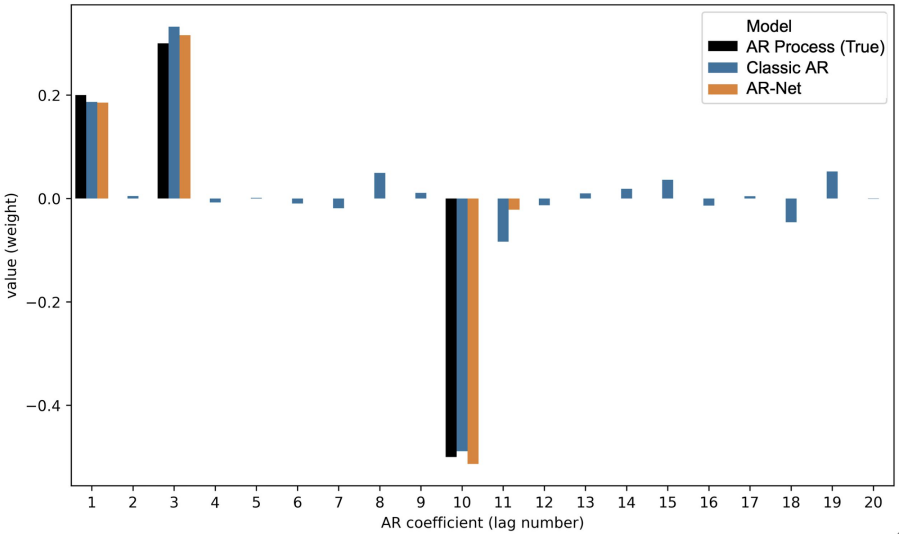
$$R(\theta) = \frac{1}{p} \sum_{i=1}^{p} \frac{2}{1 + \exp(-c_1 \cdot |\theta_i|^{\frac{1}{c_2}})} - 1$$



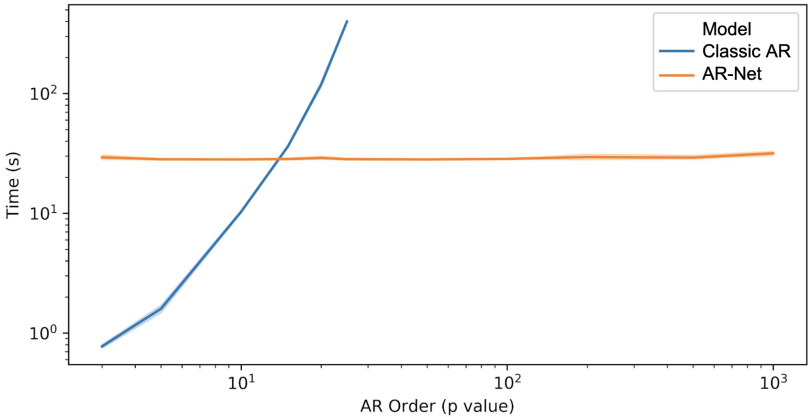
Trained with SGD (Adam)

Automatic AR-lag selection, yet faster.

Model: AR-Net



Automatic Sparsity



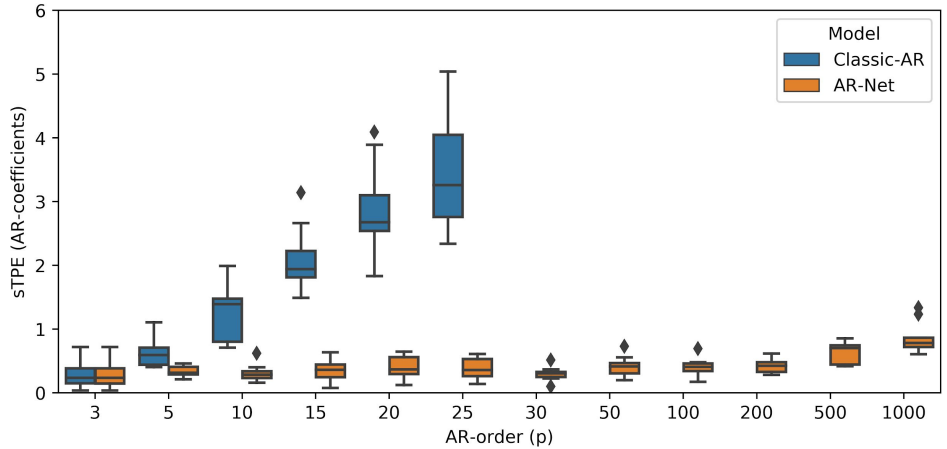
Quadratically faster

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

Model: AR-Net

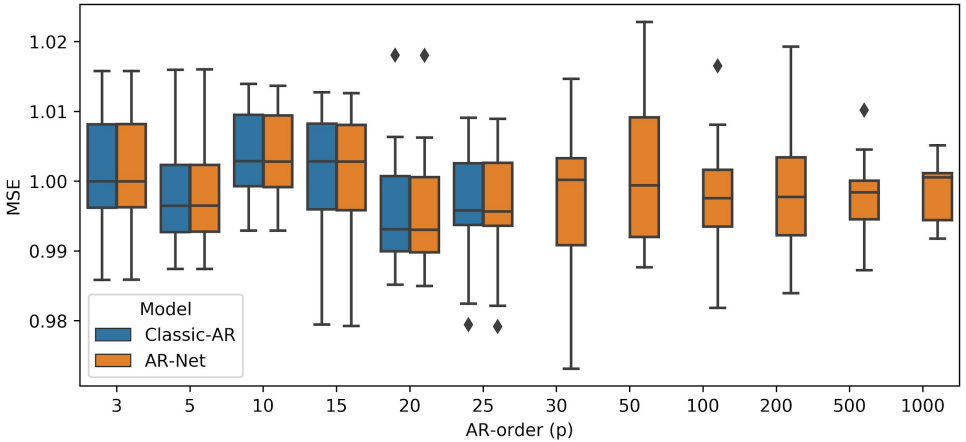
Closeness to true coefficients

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



MSE loss on forecast target

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



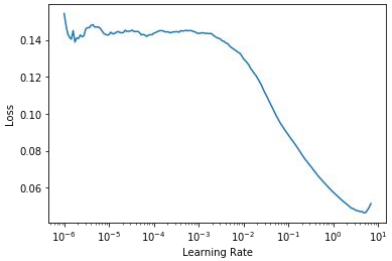
AR-Net v 1.0 builds on Fastai2 on PyTorch.

Model: AR-Net

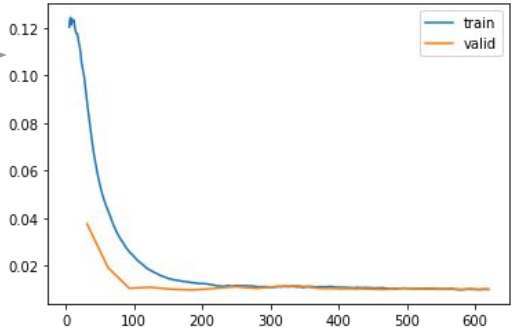
any time series

```
learn = init_ar_learner(
        series=df,
        ar_order=3,
        n_forecasts=1,
        valid_p=0.25,
        sparsity=1.0/2.0,
)

lr = learn.lr_find()[0]
learn.fit_one_cycle(n_epoch=10, lr)
print(coeff_from_model(learn.model))
```



epoch	train_loss	valid_loss	mae	time
0	0.091714	0.037677	0.165344	00:00
1	0.043905	0.019270	0.112051	00:00
2	0.026220	0.010630	0.083301	00:00
3	0.018274	0.011065	0.084113	00:00
4	0.014546	0.010218	0.081142	00:01
5	0.013065	0.009898	0.080138	00:00
6	0.012065	0.010537	0.083343	00:01
7	0.011674	0.011297	0.085578	00:00
8	0.011395	0.010602	0.082547	00:00
9	0.011284	0.011390	0.085972	00:00
10	0.011243	0.011658	0.086935	00:00



building on open-source Fastai2, PyTorch, Python

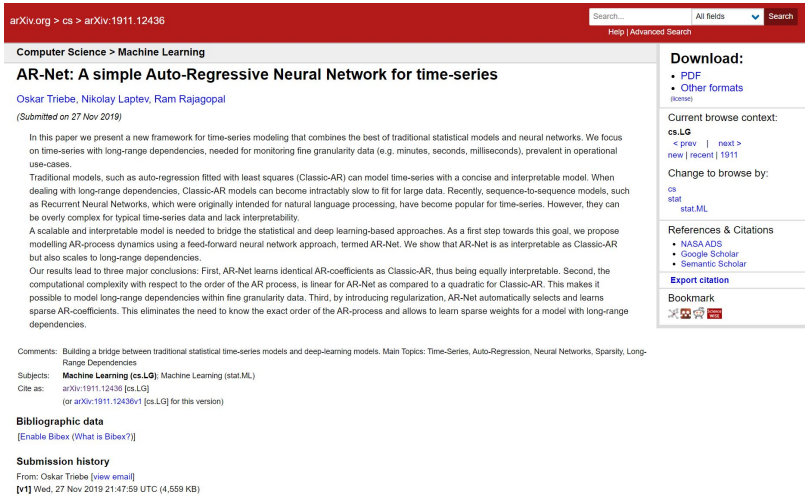
```
ar coeff [[0.19132832, 0.3154733, -0.5822663, -0.049190696, 0.0042145597, 0.00151443241]]
```

Available on Arxiv and Github.

Model: AR-Net

Preprint on Arxiv

https://arxiv.org/abs/1911.12436



arXiv.org > cs > arXiv:1911.12436

Search... All fields Search

Help | Advanced Search

Computer Science > Machine Learning

AR-Net: A simple Auto-Regressive Neural Network for time-series

Oskar Triebe, Nikolay Laptev, Ram Rajagopal

(Submitted on 27 Nov 2019)

In this paper we present a new framework for time-series modeling that combines the best of traditional statistical models and neural networks. We focus on time-series with long-range dependencies, needed for monitoring fine granularity data (e.g. minutes, seconds, milliseconds), prevalent in operational use-cases.
Traditional models, such as auto-regression fitted with least squares (Classic-AR) can model time-series with a concise and interpretable model. When dealing with long-range dependencies, Classic-AR models can become intractably slow to fit for large data. Recently, sequence-to-sequence models, such as Recurrent Neural Networks, which were originally intended for natural language processing, have become popular for time-series. However, they can be overly complex for typical time-series data and lack interpretability.
A scalable and interpretable model is needed to bridge the statistical and deep learning-based approaches. As a first step towards this goal, we propose modelling AR-process dynamics using a feed-forward neural network approach, termed AR-Net. We show that AR-Net is as interpretable as Classic-AR but also scales to long-range dependencies.
Our results lead to three major conclusions: First, AR-Net learns identical AR-coefficients as Classic-AR, thus being equally interpretable. Second, the computational complexity with respect to the order of the AR process, is linear for AR-Net as compared to a quadratic for Classic-AR. This makes it possible to model long-range dependencies within fine granularity data. Third, by introducing regularization, AR-Net automatically selects and learns sparse AR-coefficients. This eliminates the need to know the exact order of the AR-process and allows to learn sparse weights for a model with long-range dependencies.

Comments: Building a bridge between traditional statistical time-series models and deep-learning models. Main Topics: Time-Series, Auto-Regression, Neural Networks, Sparsity, Long-Range Dependencies
Subjects: **Machine Learning (cs.LG)**; Machine Learning (stat.ML)
Cite as: arXiv:1911.12436 [cs.LG]
(or arXiv:1911.12436v1 [cs.LG] for this version)

Bibliographic data

[Enable Bibex (What is Bibex?)]

Submission history

From: Oskar Triebe [view email]
[v1] Wed, 27 Nov 2019 21:47:59 UTC (4,559 KB)

Download:
- PDF
- Other formats
(license)

Current browse context:
cs.LG
< prev | next >
new | recent | 1911

Change to browse by:
cs
stat
stat.ML

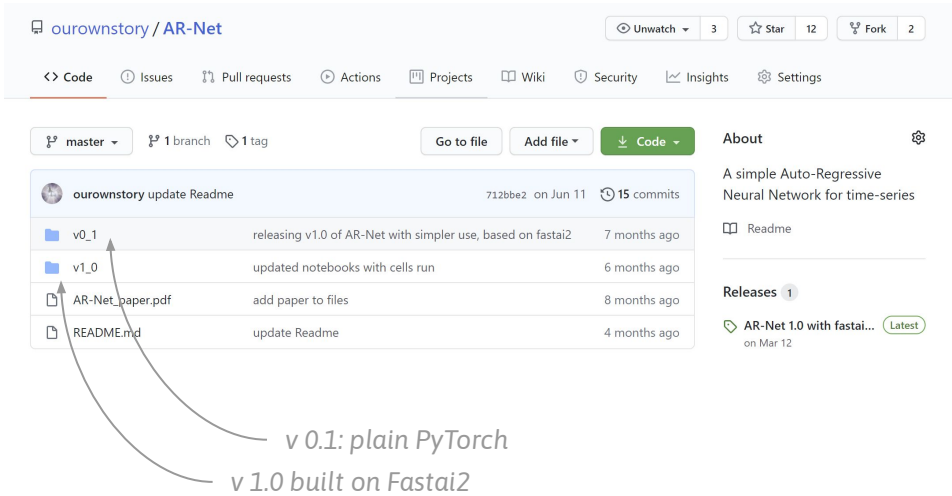
References & Citations
- NASA ADS
- Google Scholar
- Semantic Scholar

Export citation

Bookmark

Code on Github

https://github.com/ourownstory/AR-Net



ourownstory / AR-Net

Unwatch 3 | Star 12 | Fork 2

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 1 branch | 1 tag | Go to file | Add file | Code

ourownstory update Readme 712bbe2 on Jun 11 | 15 commits

v0_1	releasing v1.0 of AR-Net with simpler use, based on fastai2	7 months ago
v1_0	updated notebooks with cells run	6 months ago
AR-Net_paper.pdf	add paper to files	8 months ago
README.md	update Readme	4 months ago

About

A simple Auto-Regressive Neural Network for time-series

Readme

Releases 1

AR-Net 1.0 with fastai... Latest
on Mar 12

v 0.1: plain PyTorch

v 1.0 built on Fastai2

Hands-on beta NeuralProphet

Basics

Trend

Seasonality

Events

Autoregression

Covariates / Regressors

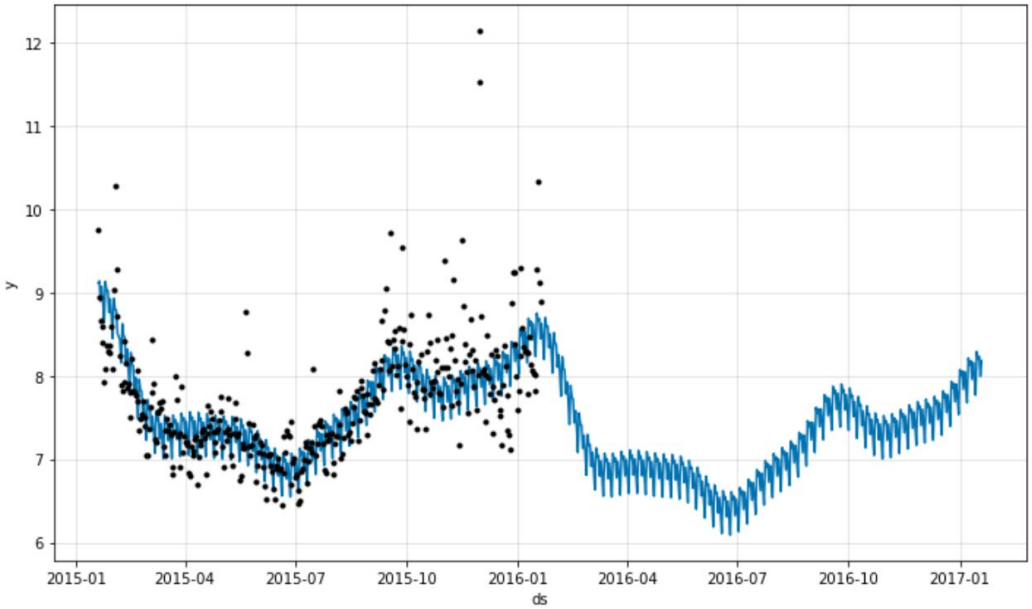
Few lines from data to forecast.

Hands-on

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

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

```
# 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:])
```



All arguments are optional.

Hands-on

```
m = NeuralProphet()
```

Gentle learning curve.

NeuralProphet has smart defaults.

Only use what you need and understand.

```
n_forecasts=1,
n_lags=0,
n_changepoints=5,
learning_rate=1.0,
loss_func='Huber',
normalize_y=True,
num_hidden_layers=0,
d_hidden=None,
ar_sparsity=None,
trend_smoothness=0,
trend_threshold=False,
yearly_seasonality='auto',
weekly_seasonality='auto',
daily_seasonality='auto',
seasonality_mode='additive',
seasonality_reg=None,
data_freq='D',
impute_missing=True,
verbose=False,
```

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.
```

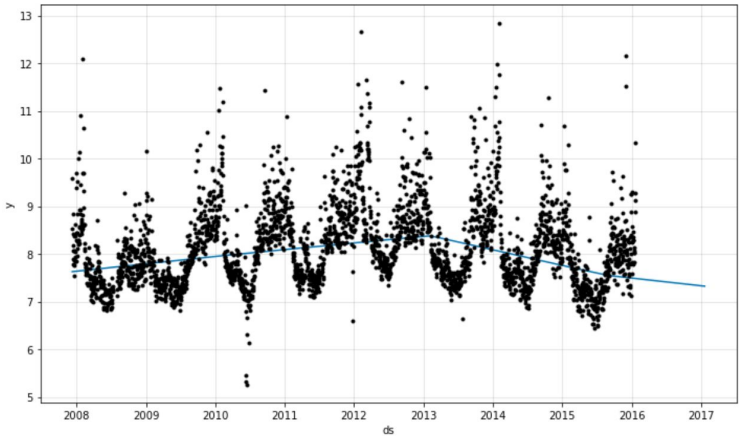
	SmoothL1Loss	MAE	RegLoss	SmoothL1Loss_val	MAE_val
35	0.163102	0.371323	0.0	0.485371	0.779465
36	0.161851	0.369609	0.0	0.368921	0.648736
37	0.161122	0.369219	0.0	0.366328	0.648230
38	0.168598	0.376638	0.0	0.348269	0.627886
39	0.167961	0.375777	0.0	0.362161	0.642699

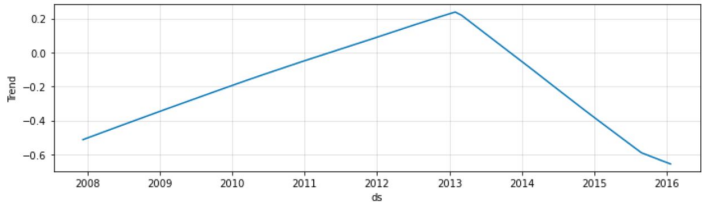
```
# split manually
m = NeuralProphet()
df_train, df_val = m.split_df(df, valid_p=0.2)
train_metrics = m.fit(df_train)
val_metrics = m.test(df_val)
```

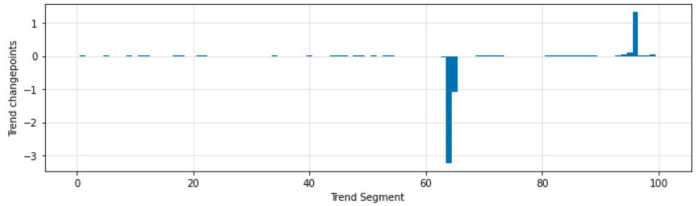
Trend

Hands-on

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



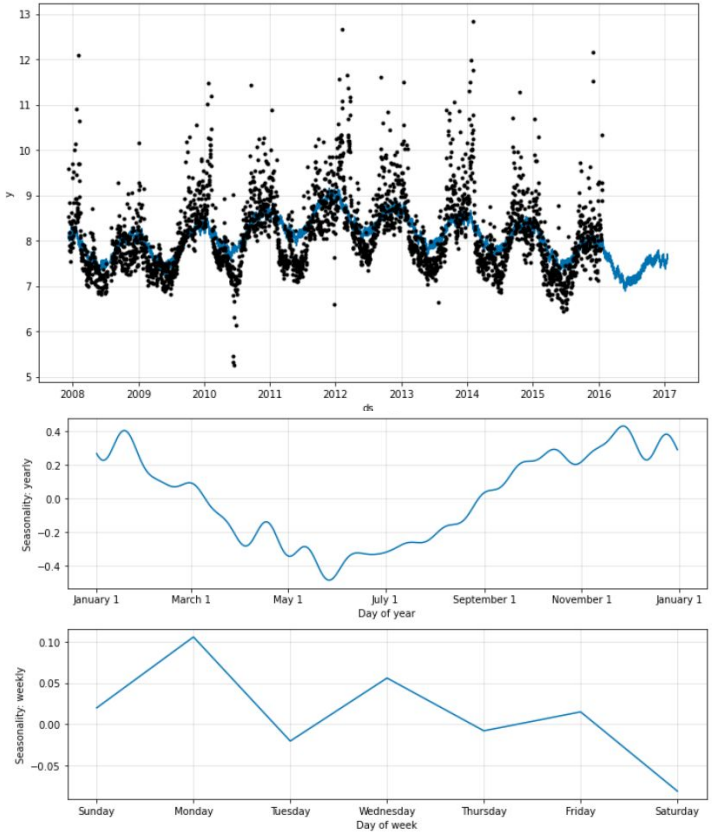




Seasonality

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

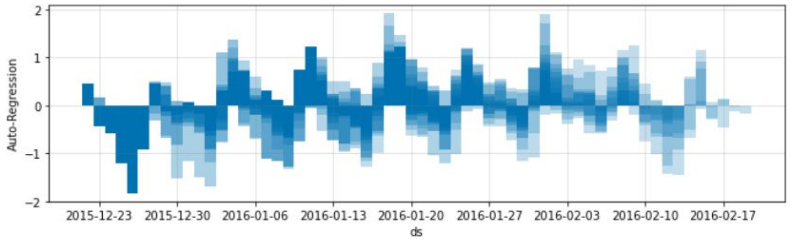
Hands-on

```
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:])
```



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')
```

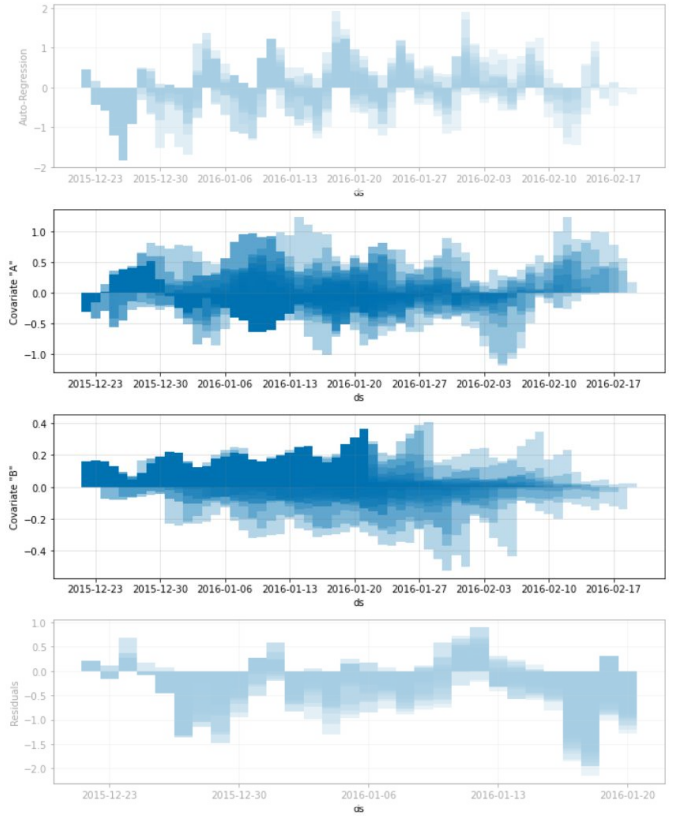
```
metrics = m.fit(df)
future = m.make_future_dataframe(df, n_historic_predictions=30)
forecast = m.predict(future)
```

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.

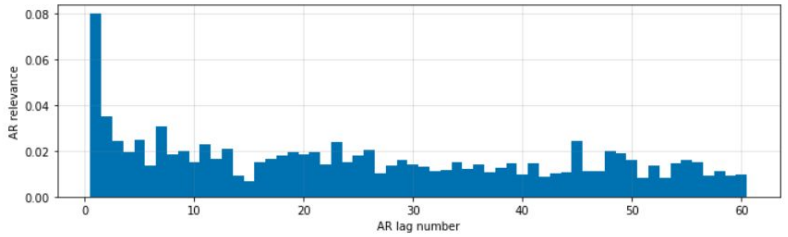
```
fig_comp = m.plot_components(forecast[60:])
```



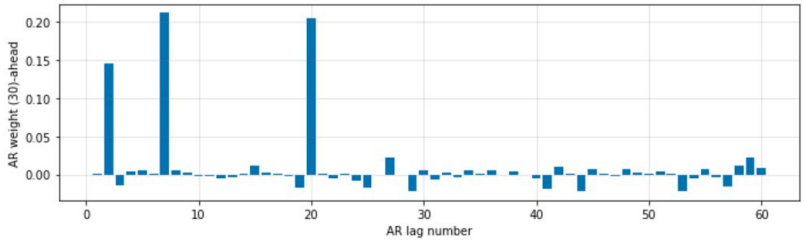
Focus on a specific forecast.

Hands-on

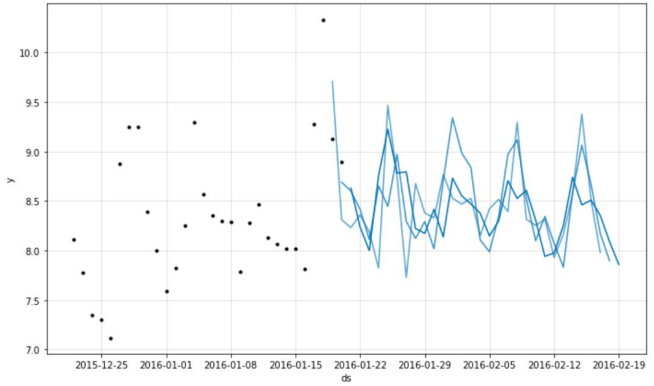
```
fig_param = m.plot_parameters()
```

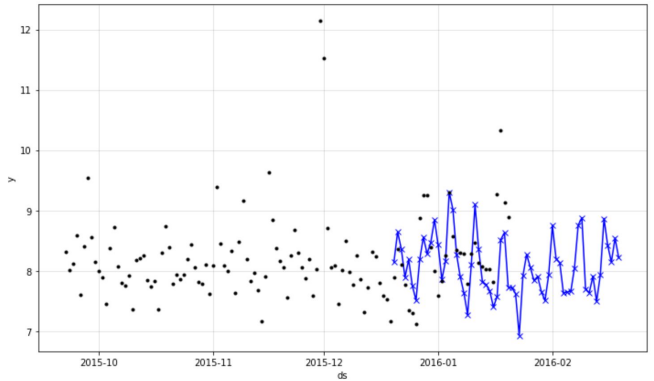


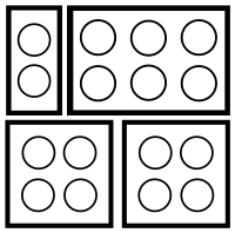

```
fig_param30 = m.highlight_nth_step_ahead_of_each_forecast(30).plot_parameters()
```




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







https://github.com/ourownstory/neural_prophet

Interpretable and customizable forecaster in PyTorch.

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

Workflow tools for diagnostics and more.

The future holds a lot of opportunities.

Upcoming:

- Beta release
- Documentation page
- Extensions (panel data, non-datetime data, …)

Future:

V 1.0	Workflow utilities, robustify training, model extensions
V 2.+	Modularize, integrate other frameworks
V 3.+	Web GUI, port to other languages

THANK YOU, dear collaborators, sponsors and advisors!

Oskar Triebe

Hansika Hewamalage

Nikolay Laptev

Ram Rajagopal

Riley De Haan

Gonzague Henri

Christoph Bergmeir

Italo Lima

Caner Komurlu

Stanford University

TOTAL

MONASH University

facebook

Connect with me: linkedin.com/in/oskartriebe/
triebe [at] stanford.edu

NeuralProphet: github.com/ourownstory/neural_prophet
AR-Net: github.com/ourownstory/AR-Net

Thank You