université PARIS-SACLAY

TIME SERIES :

INTRODUCTION, TASKS, APPROACHES & ISSUES

Vincent Guigue

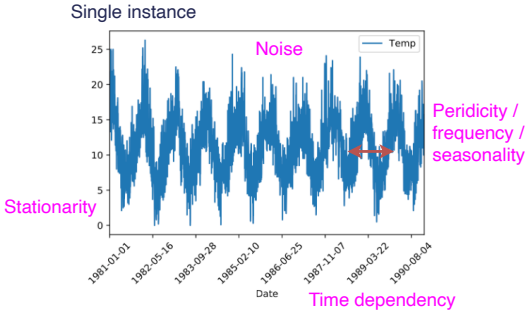
MIA PARIS-SACLAY
EKINOCS

AGROPARISTECH
Institut des Sciences et Industries du Vivant et de l'Environnement

Introduction

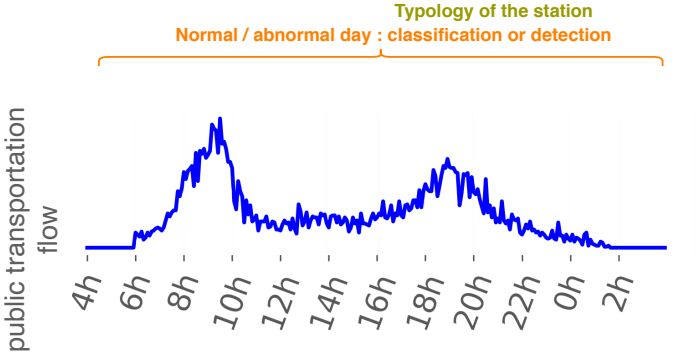
General schedule

1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning



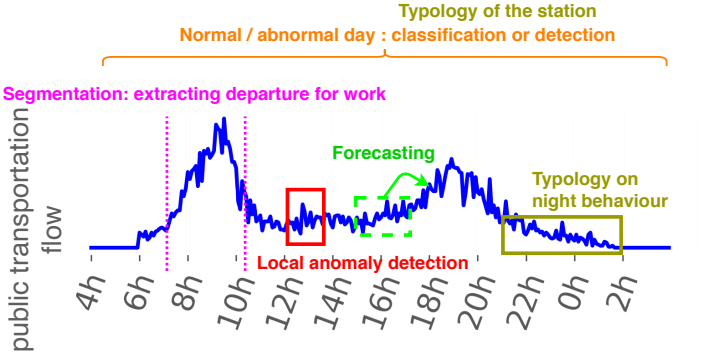
General schedule

1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning



General schedule

1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning

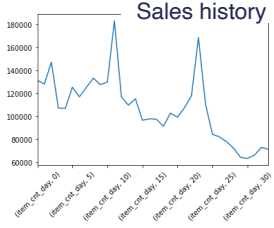


General schedule

1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning

Statistics

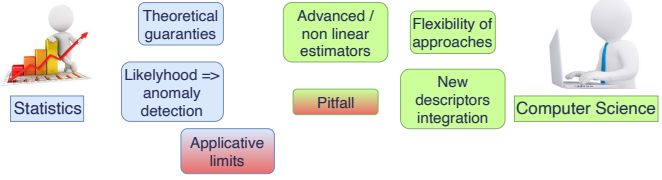
Data Science

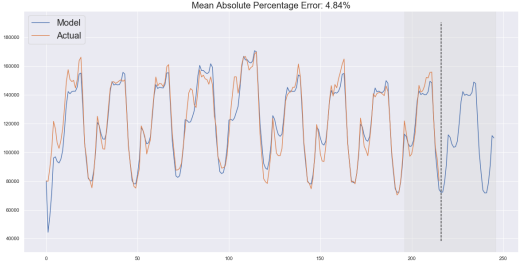


Computer Science

General schedule

1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning





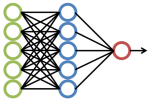
General schedule

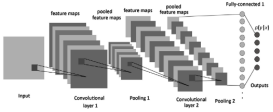
1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning

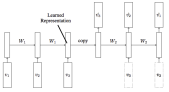
General schedule

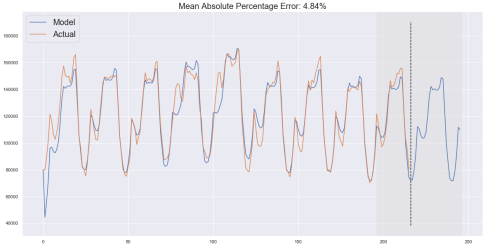
1. Signal specificities
2. Very different applications
3. Different points of view
4. Benefits/pitfalls of ML approaches
5. Benefits of deep learning

Which architecture for which application?
... And which benefits?

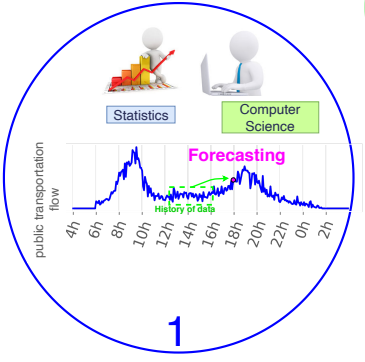


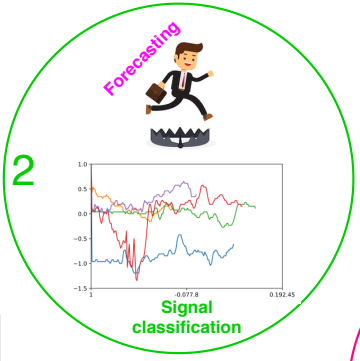


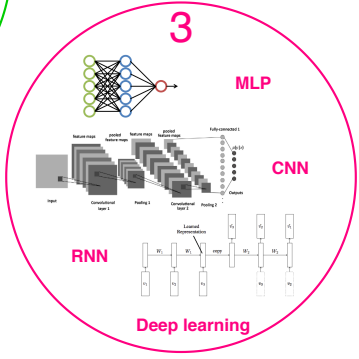




Courses organization

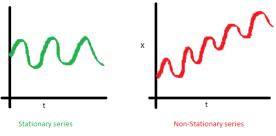




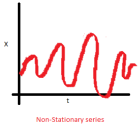


Time series = specific object

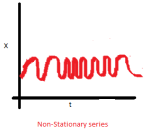
- Variable length
  - An issue... Or not, depending on the application
- Noise & de-noising
  - Specific de-noising strategy based on temporal dependency
- Stationarity / Periodicity
  - Constant statistical properties over time (mean, std. dev.)



Variable mean

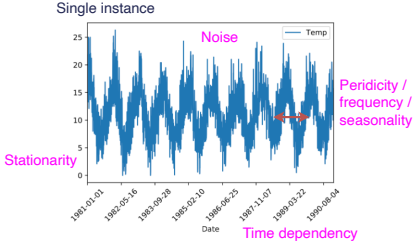


Variable std. dev.



Variable covariance

- Number of instances
  - May be one !

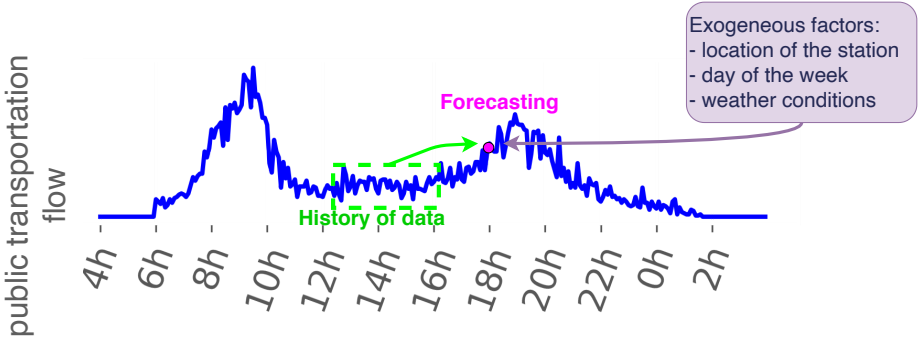


Statistical forecasting of next values

Forecasting task

1. Modeling prediction from history only
2. Adding context = exogenous factors

> Prediction is very difficult, especially about the future
>
> *Niels Bohr*



Can be applied on single/multiple instance problem.

Auto-Regressive approaches

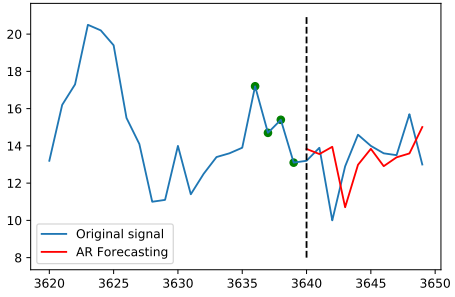
AR/ARMA

The historical answer to time-series forecasting (from statiticians)

AR : Auto-Regressive modeling:

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \ldots + \alpha_p Y_{t-p} + \varepsilon_1 \quad (1)$$

Complete Guide to Time Series Forecasting in Python, https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/



AR (order 4): 4 last measures are weighted by α to predict T

Auto-Regressive approaches

AR/ARMA

The historical answer to time-series forecasting (from statiticians)

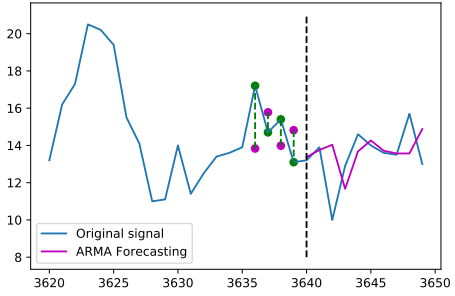
AR : Auto-Regressive modeling:

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \ldots + \alpha_p Y_{t-p} + \varepsilon_1 \quad (1)$$

ARMA : Auto-Regressive Moving Average modeling:

$$Y_t = \alpha + \alpha_1 Y_{t-1} + \alpha_2 Y_{t-2} + \ldots + \alpha_p Y_{t-p} + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \ldots + \beta_q \varepsilon_{t-q} \quad (2)$$

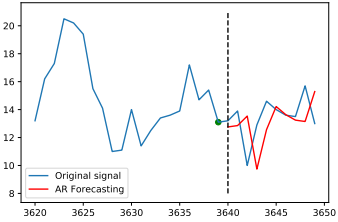
Complete Guide to Time Series Forecasting in Python, https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/



ARMA (order 4,4): 4 last measures are weighted by α & 4 errors ε are weighted by β to predict T

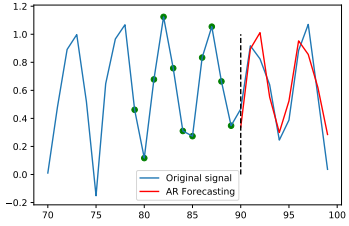
AR basic exemples & Periodicity discussion

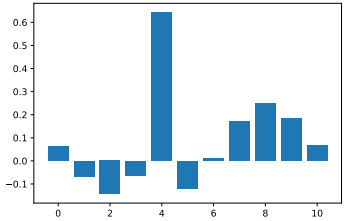
AR (order = 1): a very intuitive model:



- Order 1 = delayed version of the original signal
- Periodic signals: high coefficients on the period

AR on periodic time series (order = 10):





AR inference

AR:

- Prediction at t:

$$\hat{y}_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p}$$

- Dynamic Prediction at t (from $t-2$):

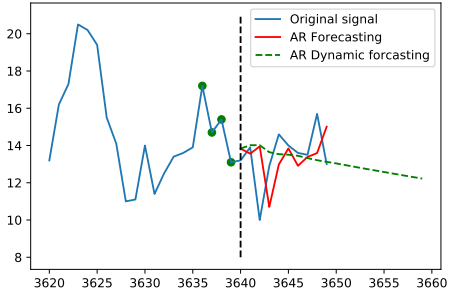
$$\hat{y}_t = \alpha + \alpha_1 \hat{y}_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p}$$

ARMA:

- Prediction at t:

$$y_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p} + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + \ldots + \beta_q \varepsilon_{t-q}$$

- Dynamic Prediction at t (from $t-2$): ε_{t-1} can no longer be computed



ε_t that we can't compute are set to 0

Parameter optimization

Problem formulation (MSE):

$$\mathcal{L} = \sum_t (y_t - \hat{y}_t)^2, \qquad \arg\min_{\alpha,(\beta)} \mathcal{L}$$

- AR problem admits a closed form solution (Yule Walker)
- ARMA is a convex problem that is solved by gradient descent

During training, $\hat{y}_t$ is estimated from real y_{t-p} values...
Our model is not dedicated to long term prediction.

Wikipedia,
https://en.wikipedia.org/wiki/Autoregressive_model

Finding AR optimal hyper-parameters

Statistician

- Information Criterion : AIC (/ BIC)
- Akaike information criterion:

$$\mathrm{AIC} = 2k - 2\ln(\mathcal{L})$$

k = nb estimated parameters
  - Maximizing likelyhood
  - while penalizing model complexity

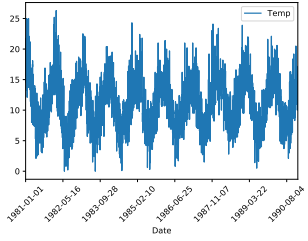
In practice: always very **low orders**

Computer scientist

- Cross validation (always)
  - Reconstruction criterion: MSE
- Estimating the generalization error on unseen data

ARIMA

- AR / ARMA approaches are dedicated to **stationary signals**
- Issue 1: **measuring** stationarity
- Issue 2: **improving** stationarity



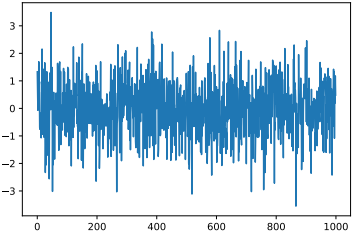
Hypothesis testing (e.g. Dicky Fuller Test):

```
Results of Dickey-Fuller Test:
Test Statistic                  -4.444805
p-value                          0.000247
#Lags Used                      20.000000
Number of Observations Used   3629.000000
Critical Value (1%)             -3.432153
Critical Value (5%)             -2.862337
Critical Value (10%)            -2.567194
```


mind the definition of the null hypothesis (Dickey-Fuller : null = non-stationary)

Stationarity & ARIMA model

- AR / ARMA approaches are dedicated to stationary signals
- Issue 1: **measuring** stationarity
- Issue 2: **improving** stationarity



Hypothesis testing (e.g. Dicky Fuller Test):

```
Results of Dickey-Fuller Test:
Test Statistic                  -31.448939
p-value                           0.000000
#Lags Used                        0.000000
Number of Observations Used     999.000000
Critical Value (1%)              -3.436913
Critical Value (5%)              -2.864437
Critical Value (10%)             -2.568313
```

mind the definition of the null hypothesis (Dickey-Fuller : null = non-stationary)

Improving stationarity = signal differencing

Differencing the signal:

Order 1 :

$\delta_t = y_t - y_{t-1}$ instead of y_t

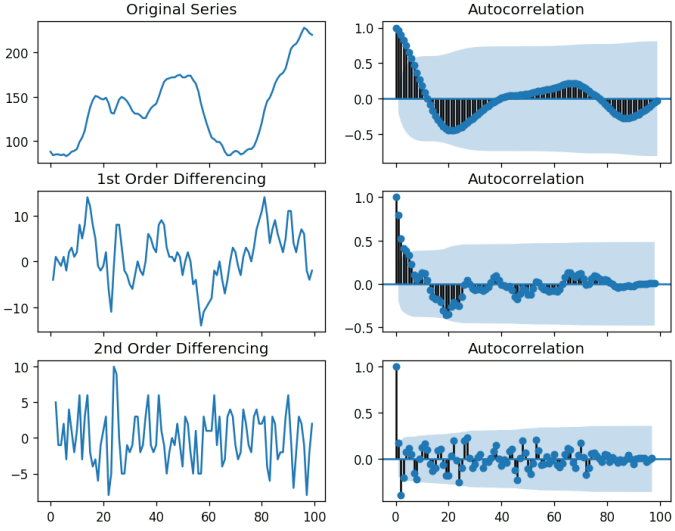
Order 2 :

$\delta_t^{(2)} = \delta_t - \delta_{t-1}$ instead of y_t

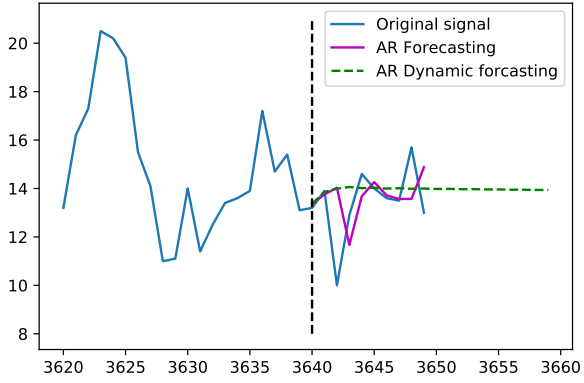
ARIMA :

Adding a (I)ntegrating parameter = order of differencing

No autocorrelation = stationary signal



What can we expect from AR modeling in practice?



In practice:

ARMA with low order = good local prediction

- Interesting modeling at $t + 1$
- Flat prediction at $t + N$

Main issue:

no **seasonality** is taken into account ($\approx$ a way to model long term dependency)

Differencing & seasonality

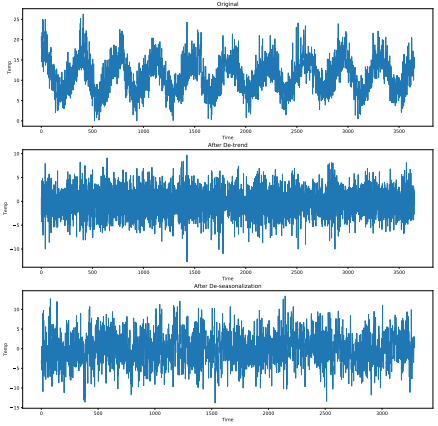
Knowing the seasonality s, it is easy to remove it:

$$\delta_t = y_t - y_{t-s}$$

And obviously to rebuild it: $y_t = \delta_t + y_{t-s}$

```
# create a differenced series
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interva
        diff.append(value)
    return Series(diff)

# invert differenced forecast
def inverse_difference(last_ob, value):
    return value + last_ob
```



Extracting trends & seasonality

1. Working at different scales: year, month, week, day, ... depending on the dataset
2. Seasonality extraction is done by convolution
   - denoting a trend t, a season s and a residue ε
     - period p must be provided
   - Additive model

$$y = t + s + \varepsilon$$

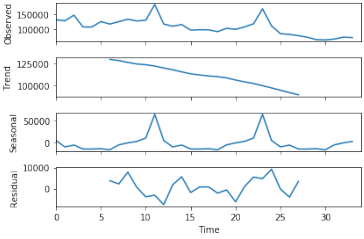
   - Multiplicative model

$$y = t \times s \times \varepsilon$$

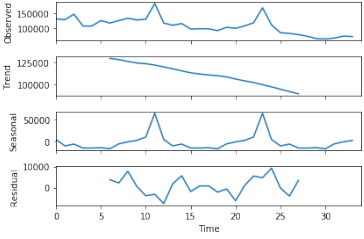
3. ARMA is performed on the residue

Time series Basics : Exploring traditional TS, G. Jagan
`https://www.kaggle.com/jagangupta/time-series-basics-exploring-traditional-ts`

Additive model:



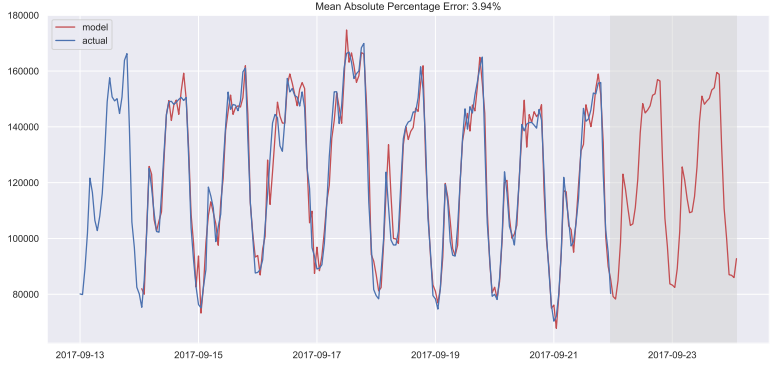
Multiplicative model:



SARIMA

Extension: Seasonality ARIMA

- Add a season length parameter
- Order(s) + Integration inside season
- + at the season level : $s_t = \alpha s_{t-1} + \dots$



Adding seasonality enables long term better predictions.

ARMA tends to 0. Seasonality & trend remain reasonable.

AR/ARMA/SARIMA & exogenous factors

Give side informations about:

- the day, the hour,
- the weather condition...

AR:

$$\hat{y}_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p}$$

AR + exogenous factors $e_1, e_2, \ldots$:

$$\hat{y}_t = \alpha + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_p y_{t-p} + \beta_1 e_{t,1} + \beta_2 e_{t,2} + \ldots$$

$\Rightarrow$ you must provide exogenous factor for the inference on the test set

Exponential Filtering (Holt-Winters predictor)

A well known alternative to SARIMA:

- Order 1:

$$\hat{y}_t = \alpha \cdot y_t + (1 - \alpha) \cdot \hat{y}_{t-1}$$

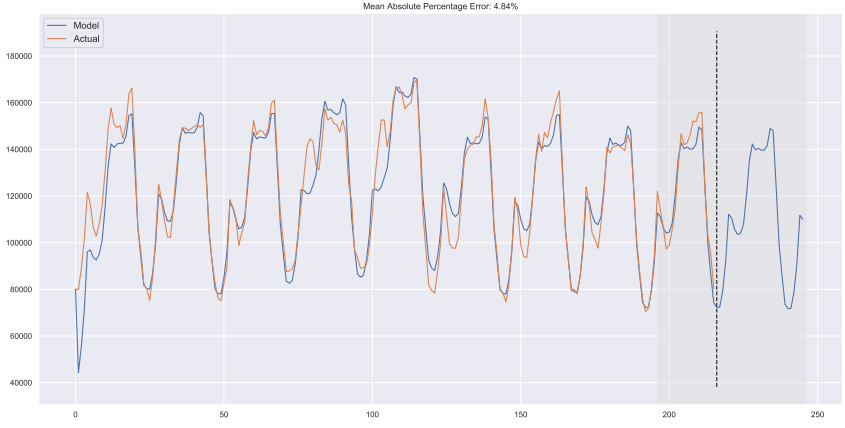
- Seasonality triple exponential filtering (=Holt-Winters)
  Prediction at horizon m, season $= s$, season length$=L$

$$\begin{aligned}
\text{1st order recursive model: } \ell_t &= \alpha\,(y_t - s_{t-L}) + (1-\alpha)\,(\ell_{t-1} + b_{t-1}) \\
\text{Differencing: } b_t &= \beta\,(\ell_t - \ell_{t-1}) + (1-\beta) b_{t-1} \\
\text{Seasonality: } s_t &= \gamma\,(y_t - \ell_t) + (1-\gamma) s_{t-L} \\
\text{Combination: } \hat{y}_{t+m} &= \ell_t + m b_t + s_{t-L+1+(m-1)\%L}
\end{aligned}$$

NB: the way to build this estimator is close to the gradient computation in ADAM

Exponential Filtering : results are close to SARIMA

Seasonality becomes more important than local modeling. . .



Greater horizon, simpler model

To look away an averaged season + trend is enough

Facebook Prophet model

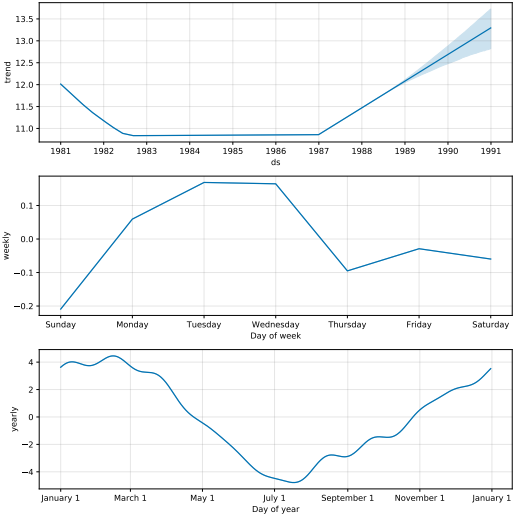
General formulation:

$$y(t) = g(t) + s(t) + h(t) + e(t)$$

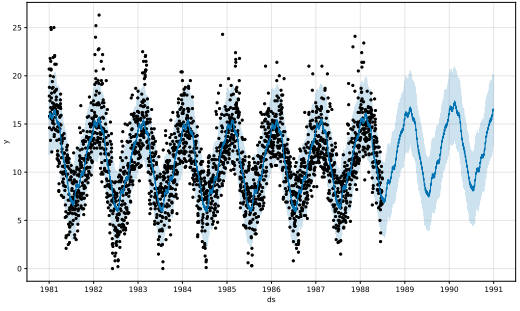
- $g(t)$: trends (for non periodic changes)
- $s(t)$: seasonality. In fact seasonality is multi-scale:
  - $s_h(t)$ hour, $s_d(t)$ day, $s_w(t)$ week, $s_m(t)$ month
- $h(t)$: holidays = prophet denomination for exogenous factors
- $e(t)$: residue

$\Rightarrow$ from statistics... Prophet $\approx$ SARIMA

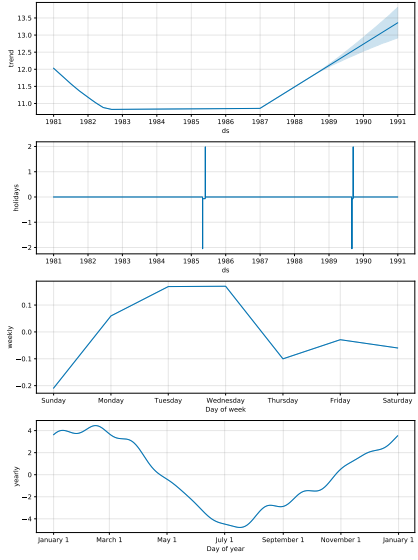
Prophet output



Australian daily minimum temperature



Prophet exogenous factor encoding



Simply define a `DataFrame` for your event & add it...

```
mod = Prophet(holidays=special_events)
#mod = Prophet()
```

Additional refinement:

- definition of overlapping special events
- possibilities to define additive or subtractive behaviour.

Prophet *vs* SARIMA

Prophet = a statistician tool in a computer science package

- more efficient (faster)
- more convenient ((almost) no parameter to set)
  - great integration with `pandas`
  - auto seasonality determination (relying on the calendar)
  - obvious counterpart: pandas is required
- better ML integration (scikit-learn / cross validation)

$\Rightarrow$ The statistical baseline to challenge ML approaches

Machine Learning Forecasting

From AR to standard ML chains

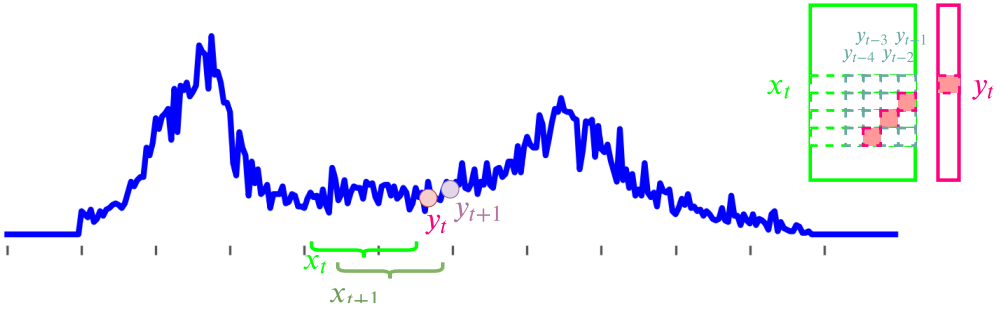
- Linear (& non linear) regression: not only an interpolator but also a predictor
- feature engineering
  - Statistical features (tsfresh) + time freq
  - Sales prediction case
  - example of features
- SVM, XGboost, ... Or neural networks
- **Easy & cheap**

some models will never be *production ready* as they demand too much time for the data preparation (for example, SARIMA),
or require frequent re-training on new data (again, SARIMA),
or are difficult to tune (good example - SARIMA),
so it's very often much easier to select a couple of features from the existing time series and build a simple linear regression or, say, a random forest. Good and cheap.

Dmitry Sergeyev

Rolling approaches

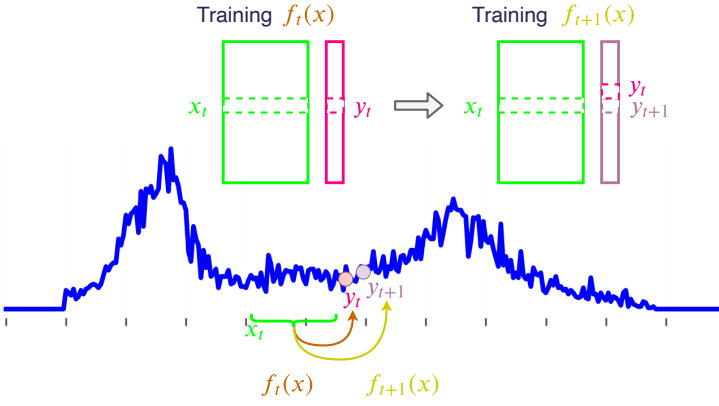
Introducing AR features in an ML environment:



Time	y	lag_6	lag_7	lag_8	lag_9	lag_10	lag_11	lag_12
2017-09-21 17:00:00	151790	132335.0	114380.0	105635.0	98860.0	97290.0	106495.0	113950.0
2017-09-21 18:00:00	155665	146630.0	132335.0	114380.0	105635.0	98860.0	97290.0	106495.0
2017-09-21 19:00:00	155890	141995.0	146630.0	132335.0	114380.0	105635.0	98860.0	97290.0

Predict further in time

- how to do it with ARMA?
  - Train your model at $t+1$ (always)
  - Apply learnt coefficient α on prediction $\hat{y}$
  - Expect poor results (without seasonality)
- how to do it with ML chain
  - Learning to predict further directly

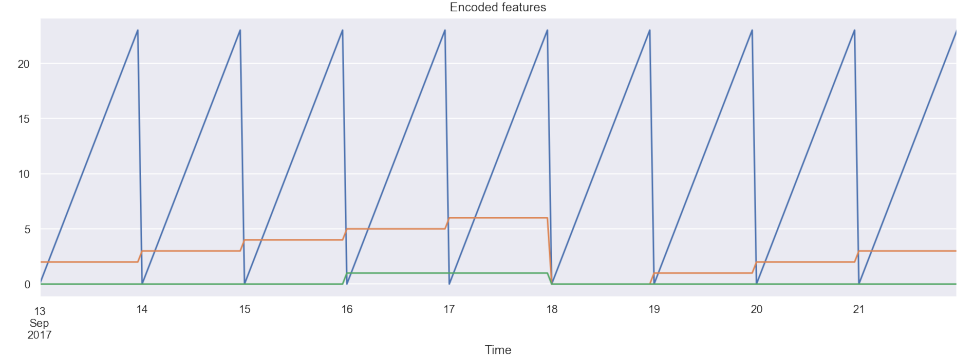


Exogenous factors / Feature engineering

Exogenous factors is straightforward in ML: just add features in the dataset

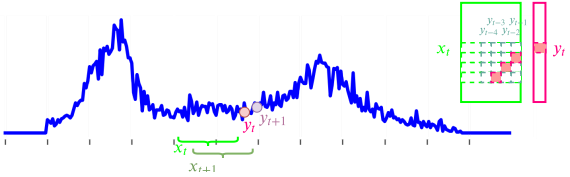
- Another way to encode seasonality
- Using pandas to catch up prophet functions
  - Types of days
  - Hours...

Example of time encoding: hour, day, week-end:



Feature engineering

- Depending on the application ⇒ discussions with experts
  [Often] more expert features ⇒ more performance
- Local statistics computations
- Statistic moment
- Frequency / power spectral density...
- `tsfresh`
  - Hundreds of features...
  - ... & test of relevant ones
  - ⚠ to be used only on time series (=lag variables)
- Classical feature engineering
  - Feature clustering, ...



Feature engineering

- Depending on the application ⇒ discussions with experts
  [Often] more expert features ⇒ more performance
- Local statistics computations

- Statistic moment
- Frequency / power spectral density...
- `tsfresh`
  - Hundreds of features...
  - ... & test of relevant ones
  -  to be used only on time series (=lag variables)

- Classical feature engineering
  - Feature clustering, ...

`abs_energy` (x)	Returns the absolute energy of the time se
`absolute_sum_of_changes` (x)	Returns the sum over the absolute value of
`agg_autocorrelation` (x, param)	Calculates the value of an aggregation func
`agg_linear_trend` (x, param)	Calculates a linear least-squares regression
`approximate_entropy` (x, m, r)	Implements a vectorized Approximate entr
`ar_coefficient` (x, param)	This feature calculator fits the unconditiona
`augmented_dickey_fuller` (x, param)	The Augmented Dickey-Fuller test is a hyp
`autocorrelation` (x, lag)	Calculates the autocorrelation of the specif
`binned_entropy` (x, max_bins)	First bins the values of x into max_bins equ
`c3` (x, lag)	This function calculates the value of
`change_quantiles` (x, ql, qh, isabs, f_agg)	First fixes a corridor given by the quantiles
`cid_ce` (x, normalize)	This function calculator is an estimate for a

Sales prediction use case

- Nature of the data
  - Shops
  - Items
- Target:
  - Fine grain: predicting the amount of each items in each Shops
  - General: sales revenue

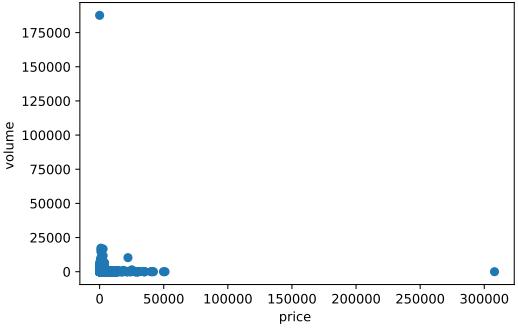
Idea: extracting features for all objects

K. Yacovlev, Kaggle notebook, 2019
`https://www.kaggle.com/kyakovlev/1st-place-solution-part-1-hands-on-data`

Sales prediction use case : classical features

- Items (aggregated over all shops)
  - Item category (from expert, from name, ...)
  - Item general trends
  - Binary: Is deprecated / Is new
  - Price/volume categorization :
    - removing (or separating outliers)
    - linspace separation
    - histogram
    - clustering
- Shops
  - Same features + co-clustering with items
- Time
  - Black Friday, holidays, ...

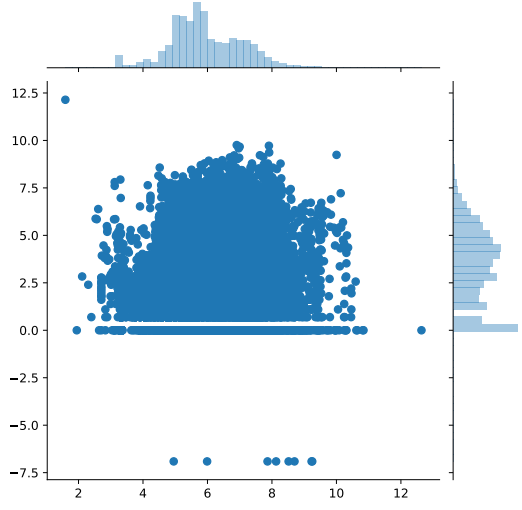
Items prices & volumes



Sales prediction use case : classical features

- Items (aggregated over all shops)
  - Item category (from expert, from name, ...)
  - Item general trends
  - Binary: Is deprecated / Is new
  - Price/volume categorization :
    - removing (or separating outliers)
    - linspace separation
    - histogram
    - clustering
- Shops
  - Same features + co-clustering with items
- Time
  - Black Friday, holidays, ...

Items **log** prices & volumes



Sales prediction use case : mono/multivariate approach

- Single shop prediction
  - with some features computed on multiple shops
- Multiple shop prediction
- Predicting all item per shop sales

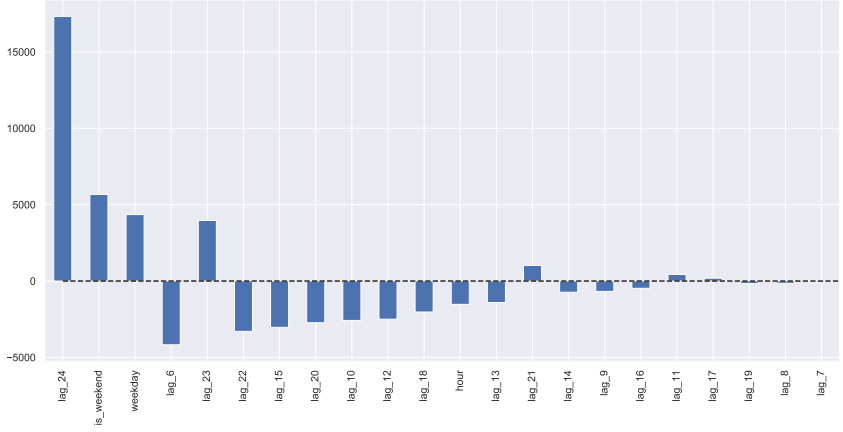
- Feature engineering makes monovariate prediction very strong
- Deep learning (try to) tackles multivariate prediction
  - To extract relevant feature automatically
  - To find fine correlation between shop/item dynamics

Variable weights normalization & interpretations

- Do not forget to normalize your data
  - Always in ML... But really mandatory when dealing with heterogenous variables
- Model introspection is always a good Idea

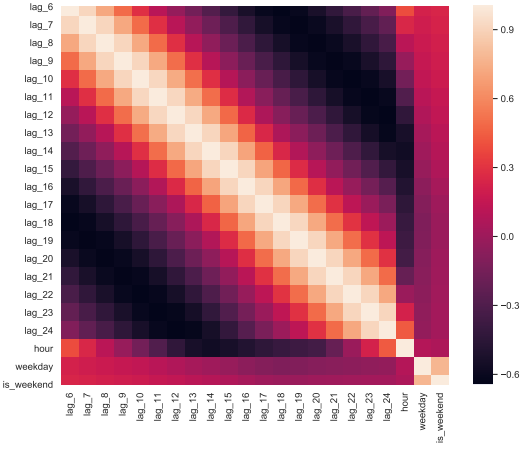
Variable weights normalization & interpretations

- Do not forget to normalize your data
  - Always in ML... But really mandatory when dealing with heterogenous variables
- Model introspection is always a good Idea



Regularization & variable selection

Extracting many features (or even several) lead to this kind of data shape:



- Ridge/regularized logistic regression
  - at least, to reduce the bad impact of correlated variables
- LASSO/Variable selection procedure
  - Reducing the number of features
  - scikit-learn...

XGBoost (everything?)

A nice algorithm...

And a **wonderful** implementation!

- like libSVM, word2vec, ...

Unsurprisingly $\Rightarrow$ Yes, it is a good idea

Model introspection: let's exploit all available functions

Amjad Abu-Rmileh, The Multiple faces of *Feature importance* in XGBoost
`https://towardsdatascience.com/be-careful-when-interpreting-your-features-impor`

Metrics

Metrics – How to evaluate if our predictions are relevant?

R squared

$s_y^2 =$ Empirical variance of Y :

$$s_y^2 = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2 + \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$s_y^2 =$ explained variance + residual variance

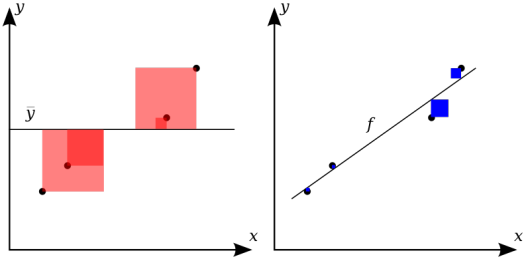
$$R^2 = \frac{\sum_i (\hat{y}_i - \overline{y})^2}{\sum_i (y_i - \hat{y}_i)^2} = \frac{\text{explained variance}}{\text{residual variance}}$$

Metrics – How to evaluate if our predictions are relevant?

R squared

coefficient of determination (in econometrics it can be interpreted as a percentage of variance explained by the model)

$$R^2 = 1 - \frac{SS_{residue}}{SS_{total}}$$



1 : all variance of the data explained $\Rightarrow$ best results

0 : worst model

Classical metrics

- Mean Absolute error

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

- Median Absolute Error (robust to outliers)

$$MedAE = \text{median}(|y_1 - \hat{y}_1|, ..., |y_n - \hat{y}_n|)$$

- Mean Absolute Percentage Error

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i}$$

Choose the metric adapted to your data to evaluate... Even if you (often) use MSE as a learning criterion

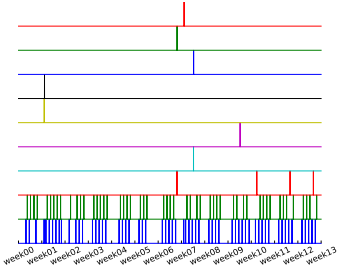
Side effects with MAPE, definition of SMAPE

Working on sparse data (e.g. validations of a single user in public transportation)
Lots of 0 in the ground truth:

- $MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i}$ diverges

Solutions:

- Rough aggregation over time to reduce sparseness
- Dedicated metrics
- ... SMAPE:



$$\text{SMAPE}_1 = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|y_t - \hat{y}_t|}{(|\hat{y}_t| + |y_t|)/2} \quad \text{or} \quad \text{SMAPE}_2 = \frac{\sum_{t=1}^{n} |y_t - \hat{y}_t|}{\sum_{t=1}^{n} (\hat{y}_t + y_t)}$$

Not magic... but sometimes robust enough to build an operational system

Anomalies

Anomaly definition

Something that is not supposed to append. Different cases:

- Outlier / error of measurement
- Distance between observations and predictions
- Anomaly tag labeled in the dataset

$\Rightarrow$ we focus on distances

Required for in-depth evaluation & for model monitoring in production

Anomaly detection: proposed implementation

- Computing bounds very easily:

```
mae = mean_absolute_error(series[window:], prediction[window:])
deviation = np.std(series[window:] - prediction[window:])
lower_bond = prediction - (mae + scale * deviation)
upper_bond = prediction + (mae + scale * deviation)
```

`scale = 1.96` (often)

- A more robust approach (still easy to implements)

```
cv = cross_val_score(model, series[window:], target[window:],
                             scoring="neg_mean_absolute_error")
mae = cv.mean() * (-1)
deviation = cv.std()
lower_bond = prediction - (mae + scale * deviation)
upper_bond = prediction + (mae + scale * deviation)
```

`scale = 1.96` (often)

Expected results

Anomalies are prediction outside the confidence bounds

Conclusion

Specific issues in time series

- Noise level
  - Discuss with expert
  - Filter
- Sampling
  - Subsampling (at least at the beginning, to reduce computations)
  - Average to reduce the noise (day $\Rightarrow$ week $\Rightarrow$ month)
  - How to deal with irregular Sampling?
    - Subsample to a regular & available rate
    - Fill missing values (EM / interpolation)

Statistician & computer scientist use different tools: make sure to exploit the best of each community proposals

Ideal combination:

(1) Naive baseline

(2) Prophet

(3) Xgboost on engineered features