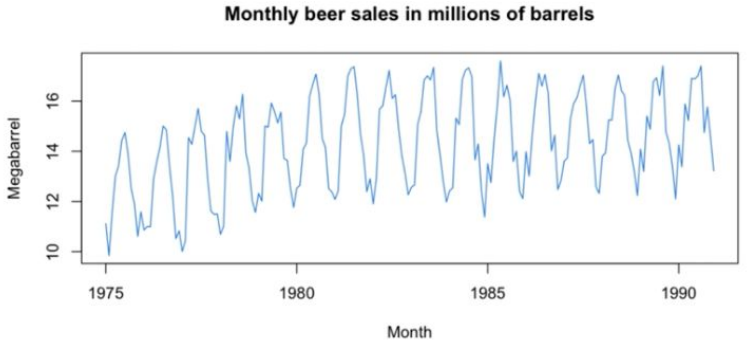
Time series Analysis

What is a time series

- Complex topic :
  - information theory
  - signal
  - probability/statistics
  - artificial intelligence/machine learning
  - optimization
  - Computer science (python)
  - Financial mathematics
- http://www.laurentoudre.fr/ast.html
- http://www.laurentoudre.fr/signalml.html
- Code
  - https://scikit-learn.org/stable/
  - https://github.com/sduprey/initiation_python_finance
  - https://github.com/sduprey/timeseries_ressources
- Books
  - https://www.amazon.fr/Probabilit%C3%A9s-pour-Licence-exercices-corrig%C3%A9s/dp/2807332234/
  - https://www.amazon.fr/Probabilit%C3%A9s-Statistiques-Mod%C3%A9lisation-lAgr%C3%A9gation-Math%C3%A9matiques/dp/2340013062
  - https://www.amazon.fr/Analyse-s%C3%A9ries-temporelles-5e-%C3%A9d/dp/2100835866
  - https://www.amazon.fr/dp/2340032997
- Online resources :
  - https://djalil.chafai.net/docs/M1/ (https://djalil.chafai.net/docs/M1/m1-series-temporelles-cours-2017-2018.pdf)
  - http://www.laurentoudre.fr/ast.html
  - https://www.ceremade.dauphine.fr/~roche/Enseignement/Series_temp/
  - https://github.com/fastai/course22
  - https://github.com/fastai/fastbook
  - https://www.imo.universite-paris-saclay.fr/~antoine.levitt/fourier/cours.pdf



Proper mathematics development

https://github.com/sduprey/timeseries_ressources/blob/main/m1-series-temporelles-cours.pdf

Modeling vs Predicting : narrowing down the determinism
A framework which encompasses AI and econometry

$$y_t = f(y_{t-1}, \ldots, y_{t-p}; \theta) + \varepsilon_t$$

- AI predilection domain: complex function with embeddings, features engineering and a small pure random part (see for example the Rossman sales kaggle competition)
- Econometrics predilection domain : function very simple (just a scalar for a random walk) (try for example to predict the bitcoin price)

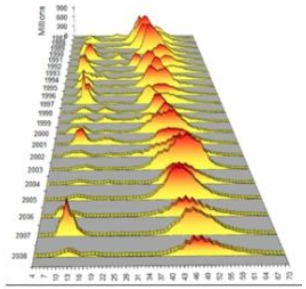
Modeling vs Predicting : narrowing down the determinism
A framework which encompasses AI and econometry

$$y_t = f(y_{t-1}, \ldots, y_{t-p}; \theta) + \varepsilon_t$$

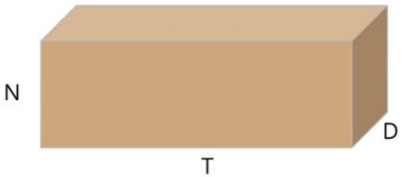
- Gives a functional form to your time series
- Gives you insight into the behavior of the time series
- “Why is the time series mean-reverting?”
- “Why does the time series grow unbounded?”
- “Is the time series predictable?”
- Someone who doesn’t know how to model might waste their time trying to predict something which is unpredictable! (e.g. a coin flip)

Dataframe (R, Pandas, PyTorch/TensorFlow tensors)

	New York City	London	Tokyo	Paris
1990-01-01	1	2	3	4
1990-01-02	5	6	7	8
1990-01-03	9	10	11	12
1990-01-04	13	14	15	16



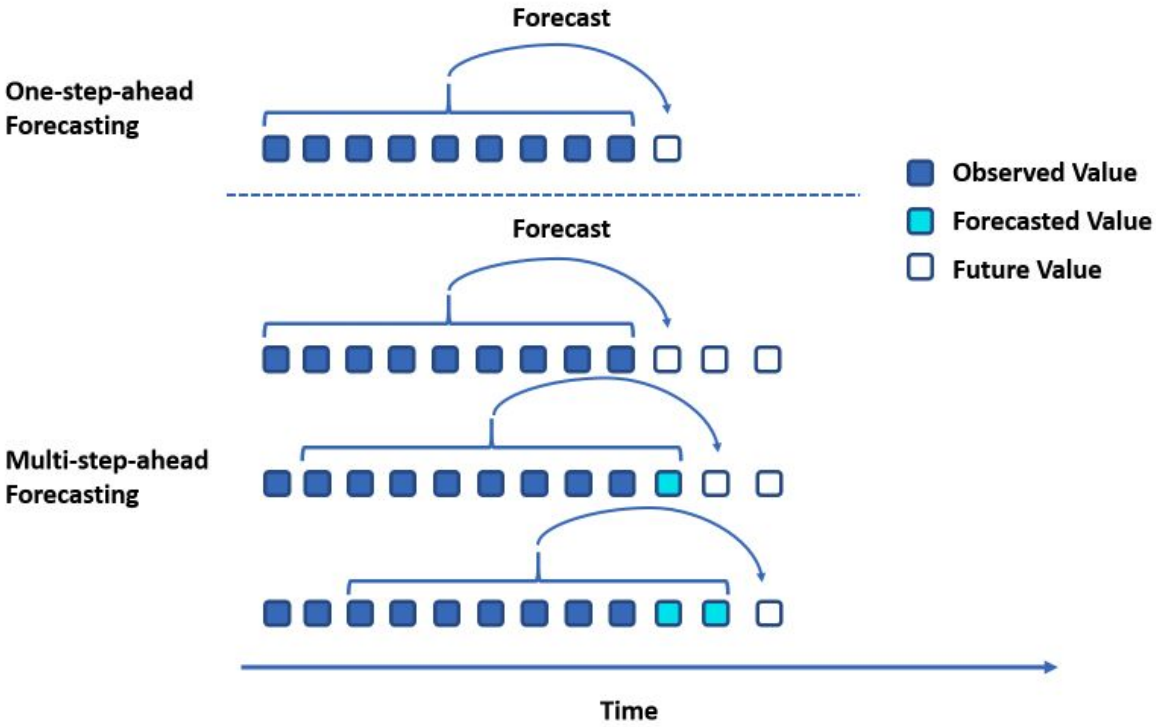
- N x T x D (a box in 3-D space)
- Automatically think of this, whenever you see / hear “N x T x D”
- It really helps, and makes it less abstract



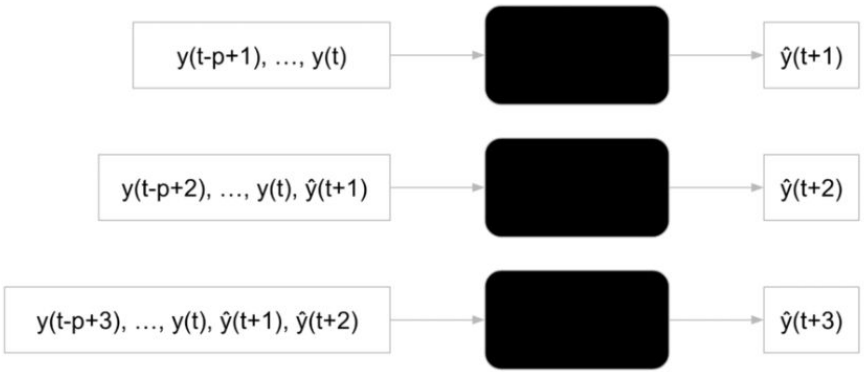
Generate random tensors/time series

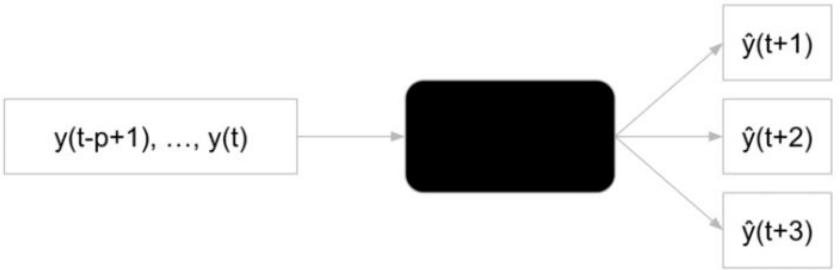
- np.random.randn(3,3,3)
- Generate from different probability distributions
- Distribution fitting with scikit-learn

1-step forecast versus multi-steps forecast



Incremental multi-steps forward versus multi-output





Stochastic processes in a nutshell

- Stochastic processes are processes that proceed randomly in time.
- Rather than consider fixed random variables X, Y, etc. or even sequences of i.i.d random variables, we consider sequences X_0, X_1, X_2, …. Where X_t represent some random quantity at time t.
- In general, the value X_t might depend on the quantity X_{t-1} at time t-1, or even the value X_s for other times $s < t$.
- Example: simple random walk .

Going more into maths:

- **Markov process**
- **Martingales**
- **Discretization of continuous processes**

Given an Itô process

$$\mathrm{d}X(t) = \mu(t, X(t))\,\mathrm{d}t + \sigma(t, X(t))\,\mathrm{d}W(t),$$

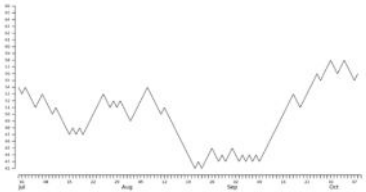
and a time discretization $\{t_i \mid i = 0, \ldots, n\}$ with $0 = t_0 < \ldots < t_n$, then the time-discrete stochastic process $\tilde{X}$ defined by

$$\tilde{X}(t_{i+1}) = \tilde{X}(t_i) + \mu(t_i, \tilde{X}(t_i))\,\Delta t_i + \sigma(t_i, \tilde{X}(t_i))\,\Delta W(t_i)$$

is called an *Euler-Maruyama scheme* of the process X (where $\Delta t_i := t_{i+1} - t_i$ and $\Delta W(t_i) := W(t_{i+1}) - W(t_i)$).

Random walk for instance

Discrete random walk



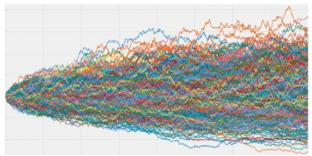
$p_0 = some\ initial\ value$

$p_1 = p_0 + e_1,\ where\ e_1 \in \{-1, +1\}$

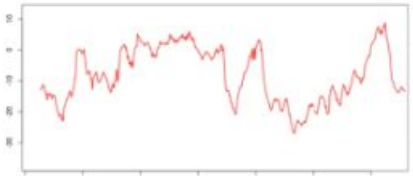
$p_2 = p_1 + e_2$

$\dots$

- Imagine yourself walking - you take one step **left** or **right** based on a coin flip - that's this random walk!
- Can't predict the future (50% chance of being correct)



Gaussian random walk

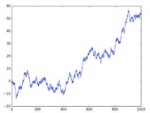


$p_0 = some\ initial\ value$

$p_1 = p_0 + e_1,\ where\ e_i \sim \mathcal{N}(0, \sigma^2)$

$p_2 = p_1 + e_2$

$\dots$



Log Prices

- Consider a random walk with drift

$$p_t = p_{t-1} + \mu + e_t,\ e_t \sim \mathcal{N}(0, \sigma^2)$$

- Take p(t-1) to the LHS - this is now the log return

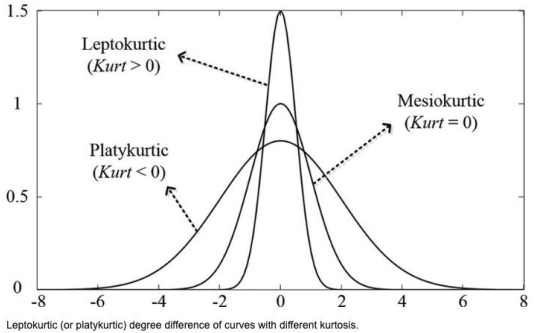
$$r_t = p_t - p_{t-1} = \mu + e_t$$

- The log return is therefore distributed as follows

$$r_t \sim \mathcal{N}(\mu, \sigma^2)$$

Stylized facts about financial time-series which invalidate the random walk hypothesis

- Volatility clustering
- Shock asymmetry
- Leptokurtic residuals



Leptokurtic (or platykurtic) degree difference of curves with different kurtosis.

Markov property

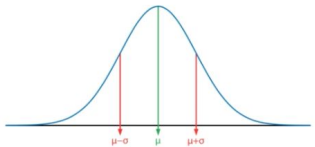
$$p(w_t \mid w_{t-1}, w_{t-2}, \ldots, w_0) = p(w_t \mid w_{t-1})$$

Gaussian random walk

$$x(t) = x(t-1) + e(t),\ e(t) \sim \mathcal{N}(0, \sigma^2)$$

$$x(t) \sim \mathcal{N}(x(t-1),\ \sigma^2)$$

Square root of time
growing confidence interval
Central limit theorem :
Converges to a gaussian



$$var\{\, x(t+\tau) \,\} = ?$$

$$x(t+1) = x(t) + e(t+1)$$

$$x(t+2) = x(t+1) + e(t+2) = x(t) + e(t+1) + e(t+2)$$

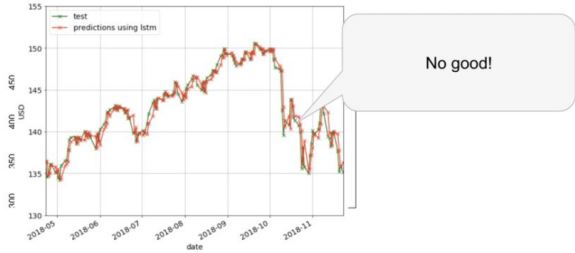
$$\ldots$$

$$x(t+\tau) = x(t) + e(t+1) + \ldots + e(t+\tau)$$

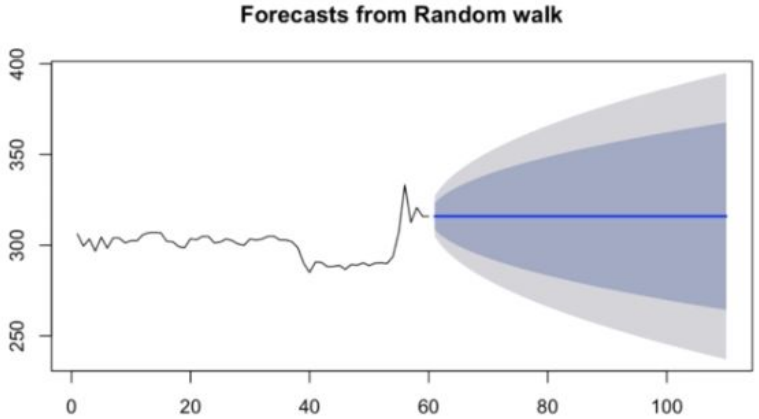
$$If\ var\{e(t)\} = \sigma^2,\ then\ var \sum_{k=1}^{\tau} e(t+k) = \tau\sigma^2,\ or\ sd\{x(t+\tau)\} = \sqrt{\tau}\sigma$$

Naive forecast or the importance of a baseline

- Compare your results against existing state of the art
- Be careful :
  - the naïve forecast looks good for martingale processes
  - In-sample versus out-of-sample forecasts
  - You can beat the naive forecast in-sample but not out-sample

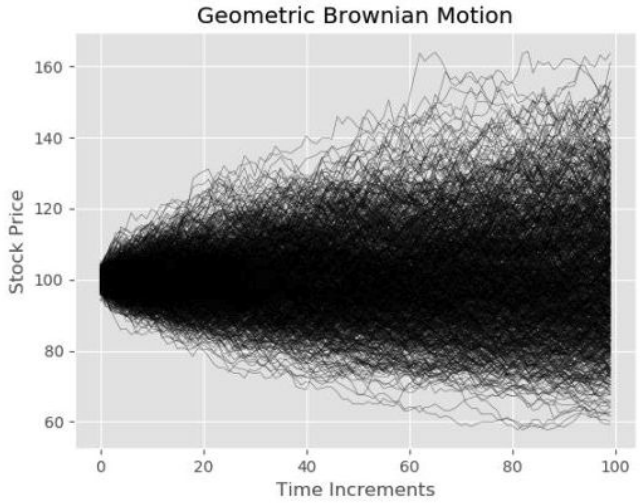


- **The naive forecast is the best for a random walk (for any martingales)**



Option pricing through Monte-Carlo simulation

- https://github.com/sduprey/timeseries_ressources/Black_Scholes_option_pricing_through_Monte_Carlo_simulation.ipynb



$$dS = S\mu dt + S\sigma dW(t)$$

$$dG = (\frac{\partial G}{\partial S}S\mu + \frac{\partial G}{\partial t} + \frac{1}{2}\frac{\partial^2 G}{\partial S^2}S^2\sigma^2)dt + \frac{\partial G}{\partial S}S\sigma dW(t)$$

$$\frac{\partial G}{\partial S} = \frac{1}{S} \quad , \quad \frac{\partial G}{\partial t} = 0 \quad , \quad \frac{\partial^2 G}{\partial S^2} = -\frac{1}{S^2}$$

$$dG = (\frac{1}{S}S\mu + 0 - \frac{1}{2}\frac{1}{S^2}S^2\sigma^2)dt + \frac{1}{S}S\sigma dW(t)$$

$$= (\mu - \frac{\sigma^2}{2})dt \ + \sigma dW(t)$$

$$lookback = e^{-rT}[\frac{1}{N}\sum_{i=1}^{N}(S_{max} - K)^+]$$

Forecasting metrics (like regression)

- Sum of squared errors (max likelihood when the errors are normally distributed) $E = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$
- Mean squared error $E = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \approx E\left((y - \hat{y})^2\right)$
- Root mean squared error (same unit) $E = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$
- Mean absolute error $E = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|$

R squared

- Not an error - we want it to be bigger, not smaller
- If your model makes perfect predictions, then MSE=0, R^2=1
- R^2=0 means your model does no better than predicting $\bar{y}$

$$E = 1 - \frac{SSE}{SST} = 1 - \frac{MSE}{Var(Y)}$$

$$where\ SST = \sum_{i=1}^{N} (y_i - \bar{y})^2,\ Var(Y) = \frac{1}{N} SST$$

Scikit-learn

- For classification models: model.score(X, Y) returns accuracy
- For regression models: model.score(X, Y) returns R^2

Problems of relativity

- Accuracy is not the best metrics for an imbalanced classifier
- MAPE (mean absolute percentage error)

$$E = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

- sMAPE

$$E = \frac{1}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|) / 2}$$

Common transformations : analog to features engineering in artificial intelligence

- Power transform $y'(t) = y(t)^{\gamma}$
- Log transform $y'(t) = \log y(t) \; or \; \log (y(t) + 1)$
- Box-Cox transform

$$y'(t) = \frac{y(t)^{\lambda} - 1}{\lambda} \quad if \; \lambda \neq 0$$
$$y'(t) = \log y(t) \quad if \; \lambda = 0$$

Since:

$$\lim_{\lambda \to 0} \frac{x^{\lambda} - 1}{\lambda} = \ln x$$

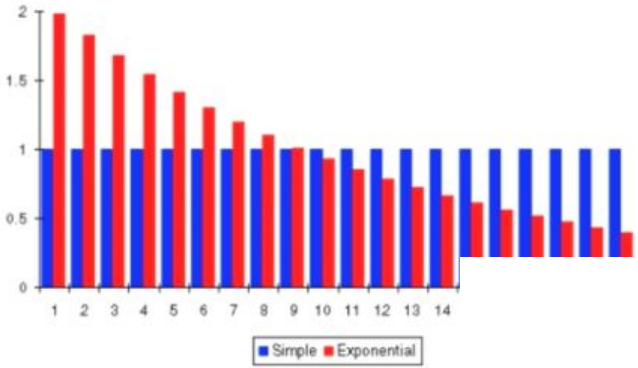
Why the log transform is fundamental

- Watch fast.ai on random forest : log uniformises large tail distributions
- In more standard econometrics, it is all about stationarity
- Rescaling data to more linear trends (decibels are the best example)
- It is part of a broader trend : features engineering for tabular data

https://github.com/fastai/fastbook/blob/master/09_tabular.ipynb

Simple/Exponentially weighted moving average

- The SMA is equally weighted, since for a window of size N, each point has a weight 1/N
- The EWMA gives each point a weight that decays exponentially



$$Arithmetic\ mean:\ \bar{x} = \frac{1}{T}\sum_{t=1}^{T} x_t$$

$$EWMA:\ \bar{x}_t = \alpha x_t + (1-\alpha)\bar{x}_{t-1},\ where\ 0 \le \alpha \le 1$$

EWMA is tailoring the importance you give to more recent samples

We make 1/t which dwindles constant !

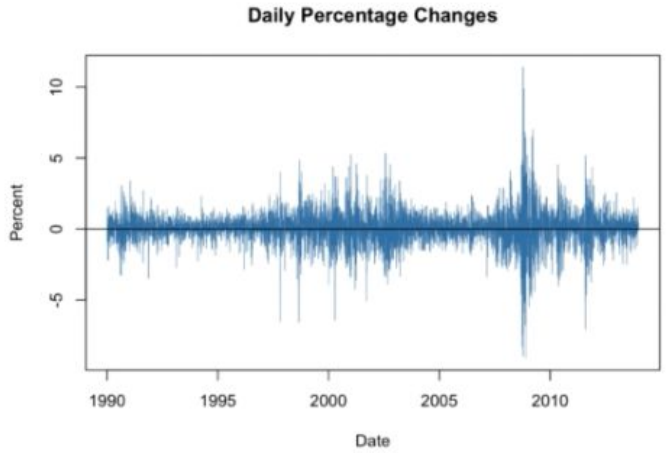
$$\bar{x}_t = \left(1 - \frac{1}{t}\right)\bar{x}_{t-1} + \frac{1}{t}x_t$$

$$\bar{x}_t = (1-\alpha)\,\bar{x}_{t-1} + \alpha x_t$$

$$\begin{aligned}
\bar{x}_t &= (1-\alpha)\bar{x}_{t-1} + \alpha x_t \\
&= (1-\alpha)[(1-\alpha)\bar{x}_{t-2} + \alpha x_{t-1}] + \alpha x_t \\
&= (1-\alpha)^2\bar{x}_{t-2} + (1-\alpha)\alpha x_{t-1} + \alpha x_t \\
&= (1-\alpha)^2[(1-\alpha)\bar{x}_{t-3} + \alpha x_{t-2}] + (1-\alpha)\alpha x_{t-1} + \alpha x_t \\
&= (1-\alpha)^3\bar{x}_{t-3} + (1-\alpha)^2\alpha x_{t-2} + (1-\alpha)\alpha x_{t-1} + \alpha x_t \\
&\cdots \\
&= (1-\alpha)^t\bar{x}_0 + \alpha\sum_{k=0}^{t-1}(1-\alpha)^k x_{t-k}
\end{aligned}$$

Why use moving average ?

Local assessment of mean/variance of a time serie (volatility clustering)

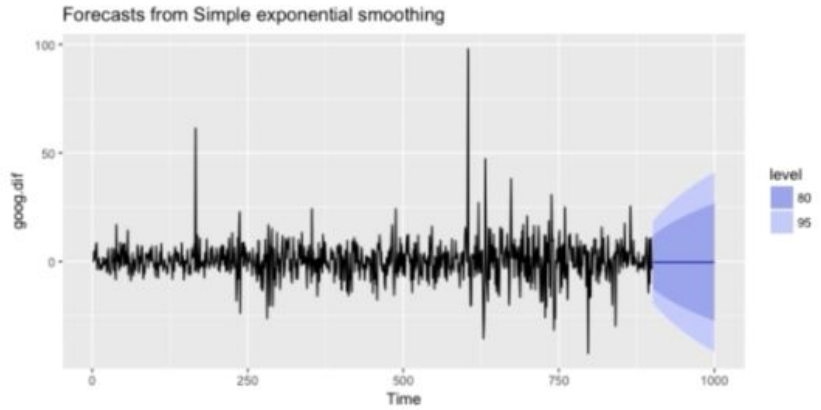



```
# returns a Rolling object
rolling_window = df['GOOG'].rolling(window_size)

# returns a series/dataframe of rolling means
# can also calculate min, max, sum, var, etc.
rolling_window.mean()

# multi-dimensional
covariance = df[['GOOG', 'AAPL']].rolling(50).cov()
```

Simple Exponential Smoothing

It makes the assumption that the process fluctuates around a mean value, which we assess through an EWMA (very efficient for martingales !)

Level(t+h) = EWMA(Time series from 1...t)



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

SES multi step forecast

$$Forecast\ Equation: \ \hat{y}_{t+h|t} = l_t, \ h = 1,2,3...$$

$$Smoothing\ Equation: \ l_t = \alpha y_t + (1-\alpha) l_{t-1}$$

- Just the EWMA
- Notice: the original time indices are back!

```
from statsmodels.tsa.holtwinters import SimpleExpSmoothing

# make the model - data is univariate
ses = SimpleExpSmoothing(data)

# 'fit' the model - returns a HoltWintersResults object
result = ses.fit(smoothing_level=alpha, optimized=False)

# in-sample prediction or out-of-sample forecast
result.predict(start=start_date, end=end_date)
```

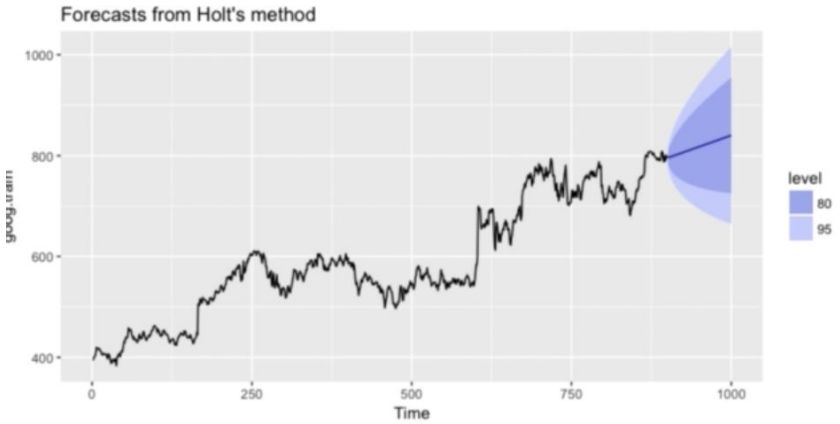
```
# get all in-sample predictions
result.fittedvalues

# forecast n steps ahead
result.forecast(n)
```

If your data goes from t=1...T, then the first forecast will be at time T+1

Holt's linear trend model

It makes the assumption that the process has a linear trend (level is the intercept, trend is the slope)

Level(t+h) = EWMA(Level of time series from 1...t)
Trend(t+h) = EWMA(Trend of time series from 1...t)



$$y = mx + b$$

$$y_t = slope \times t + y_0$$

Holt's linear trend model

$$Forecast\ Equation:\ \hat{y}_{t+h|t} = l_t + hb_t$$

$$Level\ Equation:\ l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1})$$

$$Trend\ Equation:\ b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

$$slope\ in\ one\ step = \frac{l_t - l_{t-1}}{1}$$

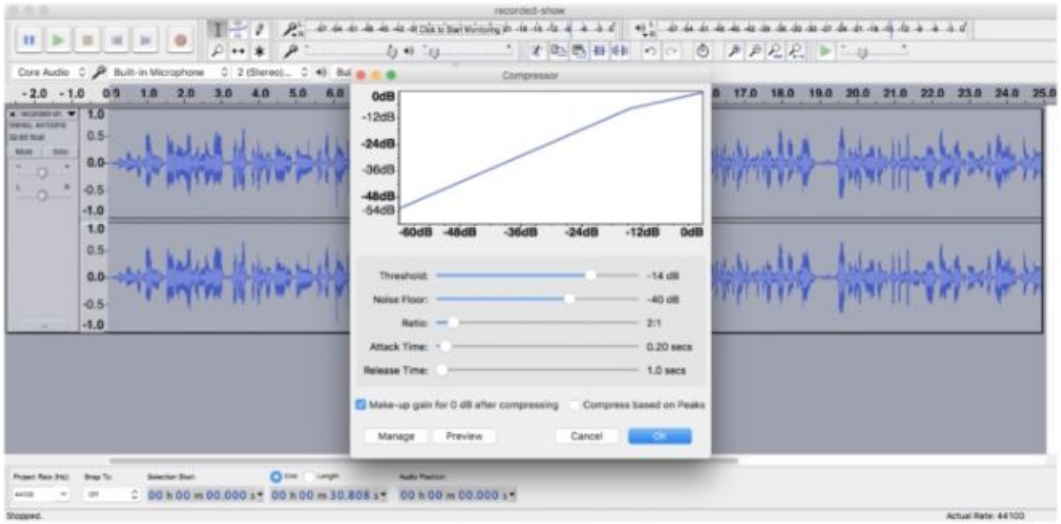
Slope = Vertical Change / Horizontal Change = Rise / Run

$$one\ step\ increase\ due\ to\ trend = l_{t-1} + 1 \times b_{t-1}$$

$$Error = \sum_{t=1}^{T} (y_t - \hat{y}_{t|t-1})^2$$

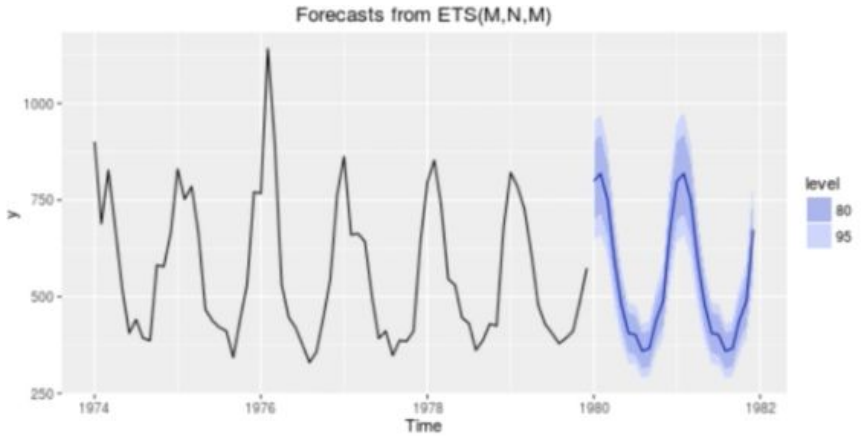
$$\alpha, \beta = \arg\min_{\alpha,\beta} Error$$

Alpha is a hyper parameter : like a frequency cut-off parameter in a signal filter



Holt-Winter model

Level(t+h) = EWMA(Level of time series from 1...t)
Trend(t+h) = EWMA(Trend of time series from 1...t)
Seasonal(t+h) = EWMA(Seasonal of time series from 1...t)



- Additive: y = level + trend + seasonal
- Multiplicative y = (level + trend) * seasonal

Holt-Winter model (additif)

$$Forecast: \hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-mk}$$

$$Level: \quad l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1})$$

$$Trend: \quad b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

$$Seasonality: \quad s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}$$

What is 'm'? The period of the cycle

What is 'k'?

The formula: $k = floor\left(\frac{h-1}{m}\right) + 1$

The intuition: finds the latest matching seasonal component (e.g. for March forecast, find the last-known March)

```
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# make the model - data is univariate
# trend/seasonal args can be 'add' or 'mul'
model = ExponentialSmoothing(
  data, trend='add', seasonal='add', seasonal_periods=12)

# 'fit' the model - returns a HoltWintersResults object
result = model.fit()

# in-sample prediction or out-of-sample forecast
result.fittedvalues
result.forecast(n)
```

Explaining the k formula

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-mk} \qquad k = floor\left(\frac{h-1}{m}\right) + 1$$

- t = 36, m = 12, h = 15
- k = floor[(15 - 1) / 12] + 1 = 2
- Index for s: t + h - mk = 36 + 15 - 12 * 2 = 27

Month	Mar	...	Dec	Jan	...	Jan	Feb	Mar
t	27	...	36	37	...	49	50	51

Holt-Winter model (multiplicatif)

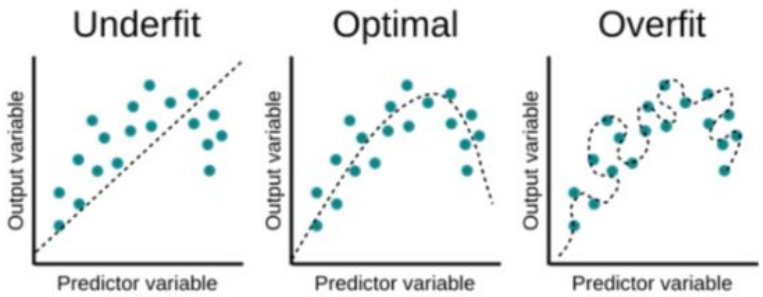
$$Forecast: \quad \hat{y}_{t+h|t} = (l_t + hb_t)s_{t+h-mk}$$

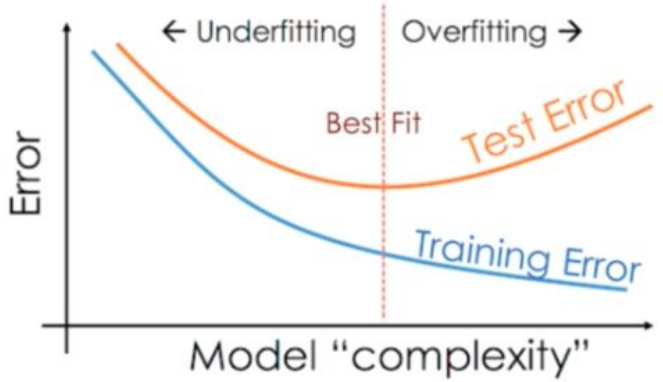
$$Level: \quad l_t = \alpha \frac{y_t}{s_{t-m}} + (1-\alpha)(l_{t-1} + b_{t-1})$$

$$Trend: \quad b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$$

$$Seasonal: \quad s_t = \gamma \frac{y_t}{l_{t-1}+b_{t-1}} + (1-\gamma)s_{t-m}$$

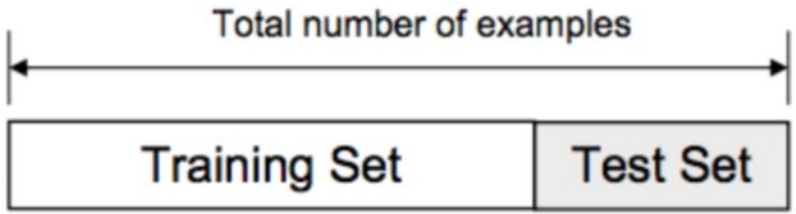
Walk-Forward validation





Single train-test splitting is wrong

- What happens when you try to optimize your model (via its configuration) by picking the best configuration for your test set?
- Your test set has effectively become “in-sample data”



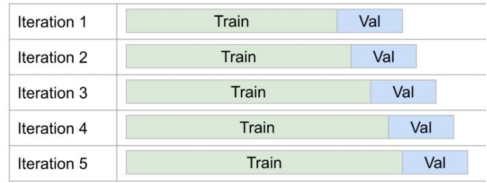
Cross-validation to mitigate overfitting the validation set

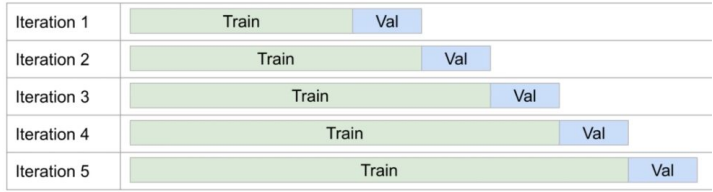
- How about multiple validation sets? (Note: I'm conflating validation/test)
- Think of your true test set as real future data
- E.g. Private data in Kaggle contest, new customers to your website
- K times: train on K-1 parts, evaluate on the remaining part

For time-series ?

- Since the data is split randomly, you'll mix past and future data
- In the real-world, your model cannot be trained on future data!
- We can only use past data to predict future data

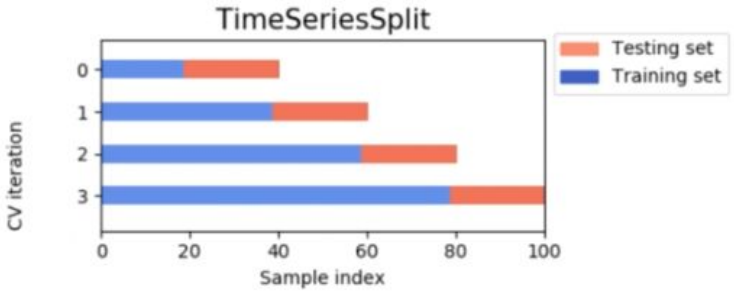
- Start with some minimum data to train on
- ... and validation forecast horizon h ≥ 1
- Walk forward one step (train set becomes +1 bigger)
- Validate on the next h data points, etc etc. until you reach the end





TimeSeriesSplit (Scikit-Learn)

- Might save you from a **bit** of work, under the right circumstances
- To use with cross validation functions, your model must conform to scikit-learn interface (i.e. not statsmodels)
- Another limitation: not flexible - must use non-overlapping blocks
- All blocks are equal size - first train set is teeny tiny
- Do these reflect real-world operation?



Do your own code according to your needs and intuition !

We won’t delve too much time using this approach into the different models we use : it complicates the script. We will just see an example script where you can plug your model afterwards.

Statistical tests & linear regression

1. **Magnitude and Sign**: The magnitude of the t-statistic indicates the strength of evidence against the null hypothesis (typically, that the coefficient is zero, meaning no effect). A higher absolute value of the t-statistic suggests stronger evidence. The sign indicates the direction of the relationship.

2. **Threshold or Critical Value**: **Compare the t-statistic to a critical value from the t-distribution,** which depends on the desired level of significance (commonly 0.05 for a 95% confidence level) and the degrees of freedom. If the absolute value of the t-statistic is greater than the critical value, the coefficient is considered statistically significant, and the null hypothesis can be rejected. There are several online t-distribution table which can be used to determine critical t-values for both one-tailed and two-tailed tests. Here is one of the them: T-Distribution Table of Critical Values.

3. **P-Value Association**: The **t-statistic is associated with a p-value**, which helps in determining significance. If the p-value is less than the chosen significance level (e.g., 0.05), the coefficient is statistically significant. This means there is sufficient evidence to suggest that the coefficient is different from zero. The null hypothesis can thus be rejected. There are several online **t-statistics / p-value calculator** for finding p-value from t-statistics value. Here is one of them: P-value from T-statistics – Calculator

```python
import statsmodels.api as sm
# Load the dataset
data = sm.datasets.get_rdataset('cars', 'datasets').data
# Fit the linear regression model
model = sm.formula.ols('dist ~ speed', data=data).fit()
# Print the t-test results
print(model.summary())
```

```
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   dist   R-squared:                       0.651
Model:                            OLS   Adj. R-squared:                  0.644
Method:                 Least Squares   F-statistic:                     89.57
Date:                Thu, 13 Apr 2023   Prob (F-statistic):           1.49e-12
Time:                        11:54:22   Log-Likelihood:                -206.58
No. Observations:                  50   AIC:                             417.2
Df Residuals:                      48   BIC:                             421.0
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    -17.5791      6.758     -2.601      0.012     -31.168      -3.990
speed          3.9324      0.416      9.464      0.000       3.097       4.768
==============================================================================
Omnibus:                        8.975   Durbin-Watson:                   1.676
Prob(Omnibus):                  0.011   Jarque-Bera (JB):                8.189
Skew:                           0.885   Prob(JB):                       0.0167
Kurtosis:                       3.893   Cond. No.                         50.7
===
```

1. **Estimate (Coefficient)**: The coefficient for **speed** is 3.9324. This means that for each additional unit of speed, the stopping distance (**dist**) increases by approximately 3.9324 units, assuming all other factors remain constant.

2. **Standard Error**: The standard error of the estimate for **speed** is 0.4155. This measures the accuracy of the coefficient (3.9324) estimate and indicates a relatively low level of uncertainty or variability in the estimate of the effect of **speed** on **dist**.

3. **t-value**: The t-value for **speed** is 9.464. This value is used to test the null hypothesis that the coefficient (effect of **speed** on **dist**) is zero. A larger absolute t-value indicates a greater deviation of the coefficient from zero, suggesting that it is highly unlikely to observe such a large t-value if the true coefficient were zero.

4. **p-value (Pr(>|t|))**: The p-value for **speed** is extremely small (1.49e-12), which is far below any standard significance level (like 0.05 or 0.01). This indicates that the effect of **speed** on **dist** is statistically significant, meaning that there is a very low probability that the observed relationship between **speed** and **dist** in the sample occurred by random chance.

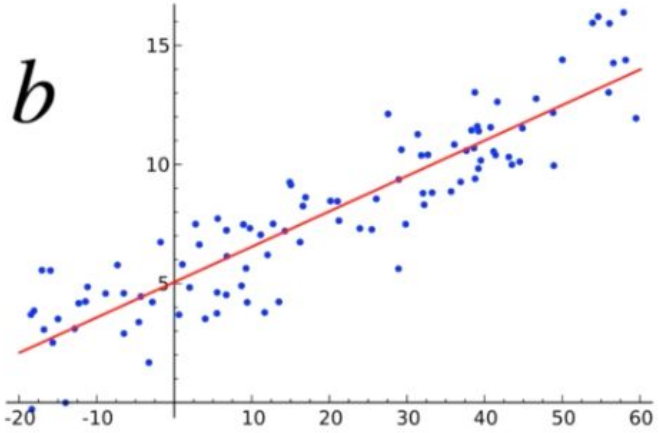
ARIMA vs Exponential smoothing

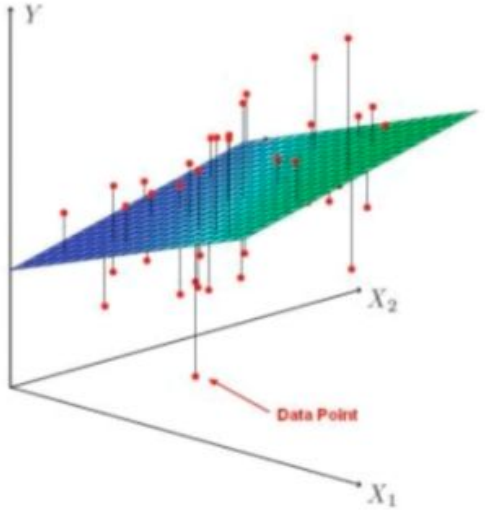
- Exponential smoothing is very specific (linear trends, seasonality)
- ARIMA imposes no such structure
- It is more “machine learning”-like

Autoregressive models

$$\hat{y} = w_1 x_1 + w_2 x_2 + b$$

$$\hat{y} = mx + b$$





AR(p) : auto-regressive processus

$$\hat{y}_t = b + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots + \varphi_p y_{t-p}$$

- Suppose our data is: $y_1, y_2, \ldots, y_{10}$
- In ML, we say X has shape N x D, but for ARIMA we'll stick with D == p

y1	y2	y3
y2	y3	y4
y3	y4	y5
y4	y5	y6
y5	y6	y7
y6	y7	y8
y7	y8	y9

This is our "X"

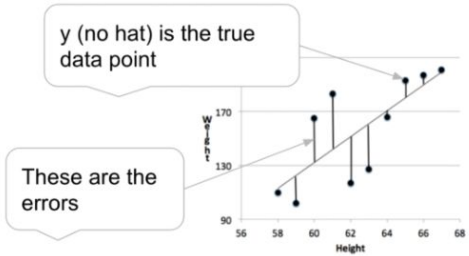
y4
y5
y6
y7
y8
y9
y10

This is our "Y"

$$y_t = b + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \ldots + \varphi_p y_{t-p} + \varepsilon_t$$

$$\varepsilon_t \sim \aleph(0, \sigma^2)$$

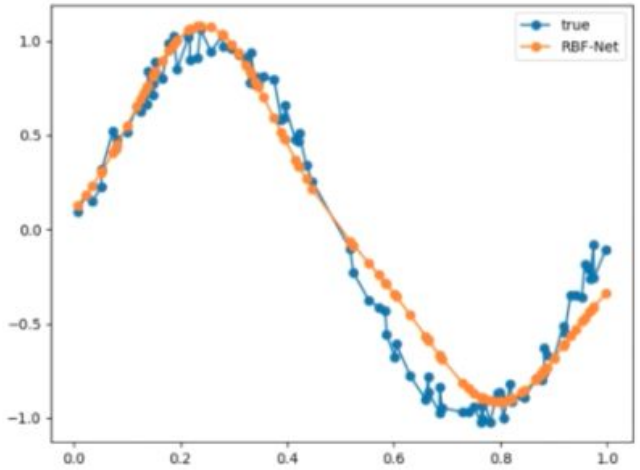
$$\hat{y}_t = E(y_t)$$



Machine Learning is the next step

- Linear models aren't that powerful (only lines or planes)
- Why stick to linear regression?
- ARIMA helps us understand the modeling and statistical properties

```
model = NeuralNetwork()
model.fit(X, Y)

model = RandomForest()
model.fit(X, Y)
```



MA(q)

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q}$$

Moving around the average c:

$$E(y_t) = E(c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \ldots + \theta_q \varepsilon_{t-q})$$

$$= c$$

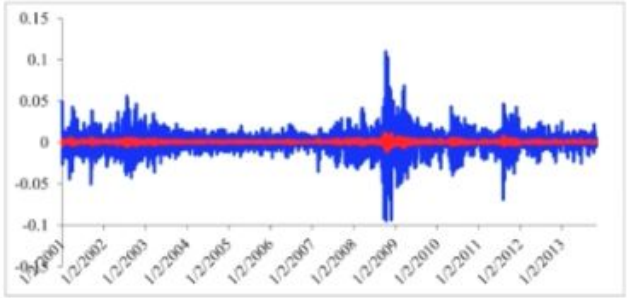
ARMA(p,q)

- ARMA(p, q) = AR(p) + MA(q)

$$y_t = b + \varphi_1 y_{t-1} + \ldots + \varphi_p y_{t-p} + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Differencing : detrending

- Where have we seen this?
  - Log returns - the difference of log prices: $r_t = p_t - p_{t-1}$
  - Holt's Linear Trend Model: $b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1}$

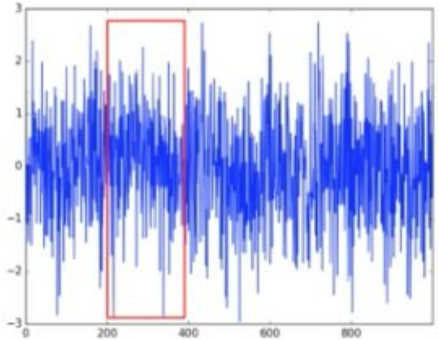


$$Given: \ \{y_t\} = \{y_1, y_2, ..., y_T\} \ (some\ time\ series)$$

$$Differenced\ Series: \ \Delta y_t = y_t - y_{t-1}$$

ARMA modelling needs stationary time series

- When fitting an ARMA model, we want the data to be close to stationary
- Stationary = Does not change over time
- Stationarity is nice: mean, variance, autocorrelation, ... will be constant over time
  - Recall: linear models fit well when there is strong correlation between inputs / output
- Each “window” of the time series is like a “training point” for fitting the model



I(d) and ARIMA(p,d,q)

- An I(d) process is a process that is stationary after differencing d times
- We say it's integrated to order d
- ARIMA(p, d, q) is just a model where we've differenced d times before applying ARMA(p, q)

- ARIMA(p, 0, 0) is AR(p)
- It's also ARMA(p, 0)
- ARIMA(0, 0, q) is ARMA(0, q) and MA(q)
- ARIMA(0, d, 0) is I(d)

Random Walk

- ARIMA(0, 1, 0) is I(1) and this is a random walk

Differenced time series

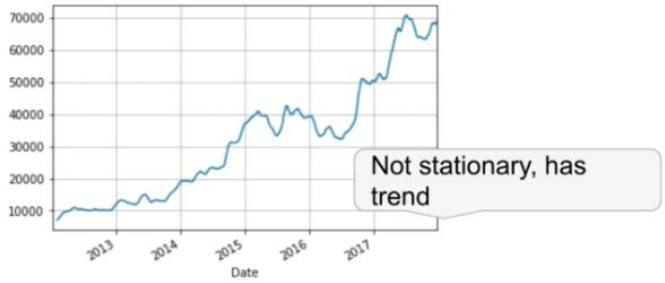
Absence of AR and MA components

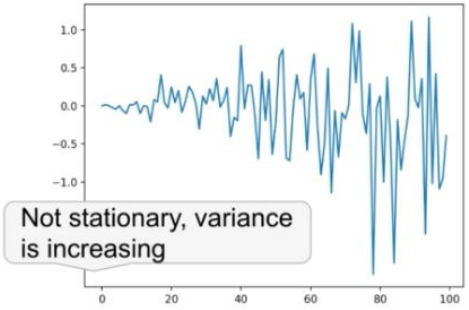
$$\Delta y_t = \varepsilon_t$$

$$y_t - y_{t-1} = \varepsilon_t$$

$$y_t = y_{t-1} + \varepsilon_t$$

Stationarity



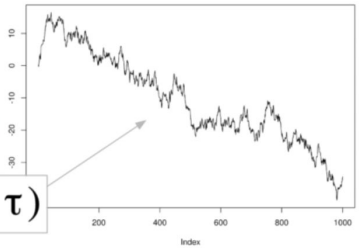


- Loosely, the distribution of the random variables in the time series does not change over time
- E.g. mean and variance will always be the same

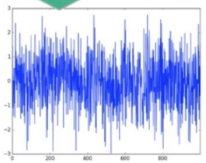
- If a time series is nonstationarity, then you might need different models at different points in time!
- For nonstationary time series, you can't treat data points at different times like "samples" (e.g. computing the mean, variance)

You can't compute "the mean" from the data because the mean is changing!

$$\mu_Y(t) \neq \mu_Y(t+\tau)$$



Here computing the mean across time is OK!



White noise

Définition 1.9 : Bruit blanc

Un processus stationnaire $(Z_t)_{t\in\mathbb{Z}}$ de moyenne μ et d'autocovariance γ_Z est un...

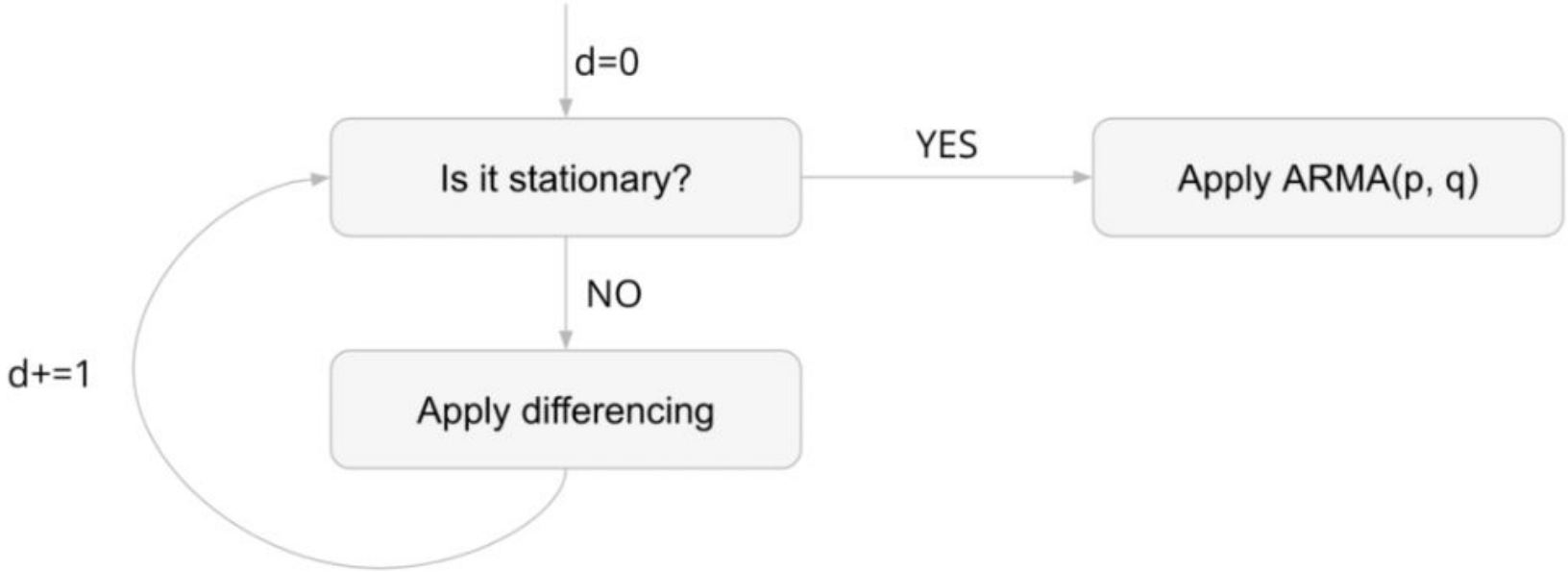
- **bruit blanc faible** ou **bruit blanc** si $\mathrm{Cov}(Z_s, Z_t) = 0$ pour tous $s \neq t$;
- **bruit blanc moyennement fort** si Z_s et Z_t sont indépendantes pour tous $s \neq t$, c'est-à-dire que $(Z_t)_{t\in\mathbb{Z}}$ sont deux à deux indépendantes ;
- **bruit blanc fort** lorsque les variables $(Z_t)_{t\in\mathbb{Z}}$ sont indépendantes ;
- **bruit blanc très fort** lorsque les variables $(Z_t)_{t\in\mathbb{Z}}$ sont i.i.d.

Dans tous les cas $\gamma_Z(h) = \sigma^2 \mathbf{1}_{h=0}$ et on note $\mathrm{BB}(\mu, \sigma^2)$. Dans ces notes de cours, lorsque la moyenne du BB n'est pas précisée, elle vaut zéro par convention.

Testing for stationarity

- We use the Augmented Dickey-Fuller Test (ADF Test)
- Think of it like an API:
  - Given: null hypothesis, alternative hypothesis
  - Input: time series, Output: p-value
  - Action: accept or reject the null hypothesis
- For ADF test:
  - Null: time series is non-stationary
  - Alternative: time series is stationary

How to use ADF test in selecting d in ARIMA



Strong vs Weak

- Strong: the entire distribution does not change over time

$$F_Y(y_{t_1+\tau}, y_{t_2+\tau}, \ldots, y_{t_n+\tau}) = F_Y(y_{t_1}, y_{t_2}, \ldots, y_{t_n}),\ \forall \tau, t_1, t_2, \ldots, t_n$$

- Weak : First order (mean) and second-order statistics (covariance) stay the same
  - The mean does not change over time

$$\mu_Y(t) = \mu_Y(t+\tau)\ for\ all\ \tau$$

  - The autocovariance does not change over time

$$K_{YY}(t_1, t_2) = K_{YY}(t_1 - t_2, 0)\ for\ all\ t_1, t_2$$

Main reasons for non stationarity

- Non stationarity in mean (trend stationary process examples)
- Non stationarity in variance (random walk with centered noise)

Trend stationary Definition:

A process $\{Y\}$ is said to be trend-stationary if[5]

$$Y_t = f(t) + e_t,$$

Examples linear, exponential, quadratic :

$$Y_t = a \cdot t + b + e_t$$

$$\mathrm{GDP}_t = Be^{at}U_t$$

$$Y_t = a \cdot t + c \cdot t^2 + b + e_t.$$

Methodology

1. **Remove deterministic trend by a regression**
   a. **Inspect statistical significance**
   b. **Remove deterministic trend**
2. **Apply canonical tests for a centered stationary process**

Proposition 6.1.1
Soit $(m_t)_t$ un polynôme (de degré d, de coefficient dominant α_d), $(B_t)_t$ un processus stationnaire centré et $Z_t = m_t + B_t$. Alors,

i) $\Delta(m_t)$ *est un polynôme de degré* $d-1$, *et donc* $\Delta^d(m_t)$ *est une constante ;*

ii) $\Delta(B_t)$ *reste un processus stationnaire centré ;*

iii) $\Delta^d(Z_t) = d!\,\alpha_d + \Delta^d B_t$.

Differentiating will remove any time polynomial trend

Augmented Dickey Fuller

A simple AR model is

$$y_t = \rho y_{t-1} + u_t$$

where y_t is the variable of interest, t is the time index, ρ is a coefficient, and u_t is the error term (assumed to be white noise). A unit root is present if $\rho = 1$. The model would be non-stationary in this case.

The regression model can be written as

$$\Delta y_t = (\rho - 1) y_{t-1} + u_t = \delta y_{t-1} + u_t$$

where Δ is the first difference operator and $\delta \equiv \rho - 1$. This model can be estimated, and testing for a unit root is equivalent to testing $\delta = 0$. Since the test is done over the residual term rather than raw data, it is not possible to use standard t-distribution to provide critical values. Therefore, this statistic t has a specific distribution simply known as the Dickey–Fuller table.

There are three main versions of the test:

1. Test for a unit root:

$$\Delta y_t = \delta y_{t-1} + u_t$$

2. Test for a unit root with constant:

$$\Delta y_t = a_0 + \delta y_{t-1} + u_t$$

3. Test for a unit root with constant and deterministic time trend:

$$\Delta y_t = a_0 + a_1 t + \delta y_{t-1} + u_t$$

Each version of the test has its own critical value which depends on the size of the sample. In each case, the null hypothesis is that there is a unit root, $\delta = 0$. The tests have low statistical power in that they often cannot distinguish between true unit-root processes ($\delta = 0$) and near unit-root processes (δ is close to zero). This is called the "near observation equivalence" problem.

The intuition behind the test is as follows. If the series y is stationary (or trend-stationary), then it has a tendency to return to a constant (or deterministically trending) mean. Therefore, large values will tend to be followed by smaller values (negative changes), and small values by larger values (positive changes). Accordingly, the level of the series will be a significant predictor of next period's change, and will have a negative coefficient. If, on the other hand, the series is integrated, then positive changes and negative changes will occur with probabilities that do not depend on the current level of the series; in a random walk, where you are now does not affect which way you will go next.

- **Modèle 1** noté **(M1)** (ni constante ni tendance linéaire)

$$Z_t = \rho Z_{t-1} + \mathcal{E}_t,$$

- **Modèle 2** noté **(M2)** (avec une constante mais pas de tendance linéaire)

$$Z_t = \mu + \rho Z_{t-1} + \mathcal{E}_t,$$

- **Modèle 3** noté **(M3)** (avec une constante et une tendance linéaire)

$$Z_t = \mu + \beta t + \rho Z_{t-1} + \mathcal{E}_t,$$

Dickey Fuller test conducting strategy ??

- Remove Deterministic trend
- Apply the test for centered process

STRATEGIE 5.1
Stratégie de test de non stationnarité basée sur la statistique de Dickey et Fuller : *Comme on ne sait pas a priori si le modèle doit comprendre une constante et/ou une tendance, il convient de suivre une stratégie de test.*

Etape 1 : *On commence par le modèle* ***(M3)****, qui est le plus général. On réalise la régression standard, avec constante, de Z_t sur les variables explicatives t et Z_{t-1}, puis on calcule la statistique de Student usuelle t_β du coefficient β. En revanche, on teste la significativité de la tendance linéaire (coefficient β) grâce aux valeurs critiques dans la Table 5.2.*

- *1.1 : Si la tendance n'est pas significative, on va à l'étape 2.*
- *1.2 : Si la tendance est significative, on effectue le test $(H_0) : \rho = 1$ en considérant la statistique $t_{\rho,3}$ donnée par (5.3), et la Table 5.1, modèle* ***(M3)****.*
  - *1.2.1 : Si (H_0) est rejetée, alors $(Z_t)_t$ est non stationnaire de type* ***TS****.*
  - *1.2.2 : Si (H_0) est non rejetée, alors $(Z_t)_t$ est non stationnaire en moyenne et en variance de type* ***TS*** *+* ***DS****.*

Etape 2 : *On considère alors le modèle* ***(M2)****. On réalise la régression standard, avec constante, de Z_t sur la variable retardée Z_{t-1}, puis on calcule la statistique de Student usuelle t_μ de la constante μ. En revanche, on teste la significativité de la constante (coefficient μ) grâce aux valeurs critiques dans la Table 5.3.*

- *2.1 : Si la constante est non significative, on va à l'étape 3.*
- *2.2 : Si la constante est significative, on effectue le test $(H_0) : \rho = 1$ en considérant la statistique $t_{\rho,2}$ donnée par (5.2), et la Table 5.1, modèle* ***(M2)****.*
  - *2.2.1 : Si (H_0) est rejetée, alors $(Z_t)_t$ est un processus stationnaire AR(1) avec constante.*
  - *2.2.2 : Si (H_0) est non rejetée, alors $(Z_t)_t$ est non stationnaire de type* ***DS*** *avec constante.*

Etape 3 : *On considère alors le modèle* ***(M1)****. On effectue le test $(H_0) : \rho = 1$ en considérant la statistique $t_{\rho,1}$ donnée par (5.1), et la Table 5.1, modèle* ***(M1)****.*

- *3.1 : Si (H_0) est rejetée, alors $(Z_t)_t$ est un processus stationnaire AR(1) sans constante.*
- *3.2 : Si (H_0) est non rejetée, alors $(Z_t)_t$ est non stationnaire de type* ***DS*** *sans constante.*

(voir la Figure 5.6 qui illustre cette stratégie)

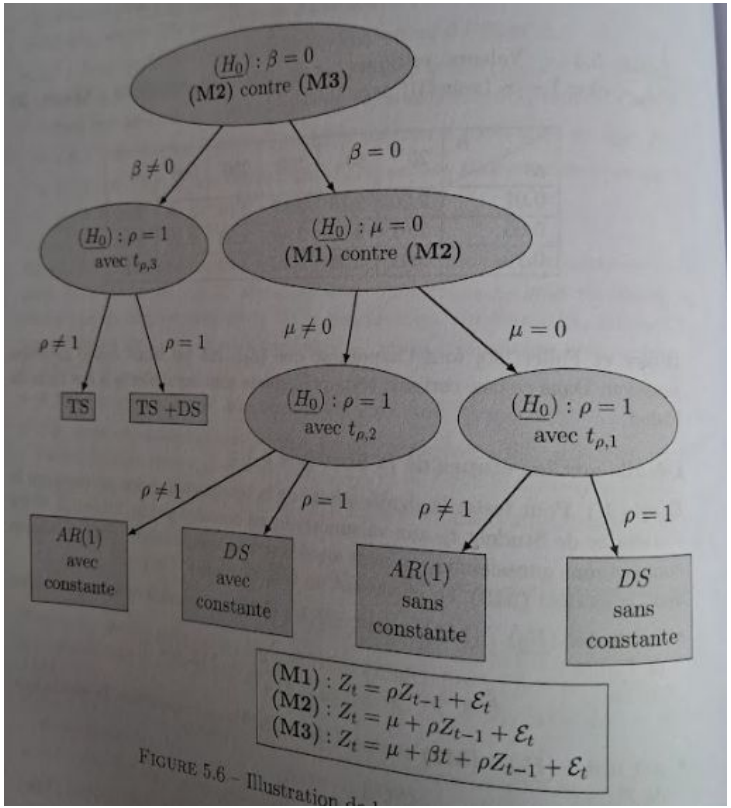


FIGURE 5.6 – Illustration d

KPSS test : same strategy

Soit $(\mathcal{E}_t)_t$ un bruit blanc fort de variance $\sigma_\mathcal{E}^2$, et soit $(X_t)_t$ la marche aléatoire associée :

$$\begin{cases} X_0 &= \mathcal{E}_0 \\ X_t &= X_{t-1} + \mathcal{E}_t . \end{cases}$$

Soit $(B_t)_t$ un processus stationnaire centré. On notera ${}^t v$ la transposée du vecteur v. Considérons le modèle

$$Z_t = {}^t\beta\, D_t + X_t + B_t ,$$

où D_t désigne

- soit la fonction constante égale à 1, et dans ce cas ${}^t\beta = \mathbb{E}(Z_t)$,
- soit le vecteur colonne ${}^t(1, t)$ et dans ce cas ${}^t\beta\, D_t$ représente une tendance linéaire.

Le test de stationnarité, proposé par Kwiatkowski-Phillips-Schmidt-Shin (KPSS) en 1992 [61] permet de tester :

$$(\underline{H_0}) : \sigma_\varepsilon^2 = 0 \ (\textit{et donc } X_t = 0) \text{ contre } (\underline{H_1}) : \sigma_\varepsilon^2 \neq 0.$$

i) Sous $(\underline{H_0})$, le processus $(Z_t)_t$ est stationnaire autour de ${}^t\beta\, D_t$ car $Z_t - {}^t\beta\, D_t = B_t$.

ii) Sous $(\underline{H_1})$, le processus $(Z_t)_t$ n'est pas stationnaire autour de ${}^t\beta\, D_t$ car $\text{var}(Z_t - {}^t\beta\, D_t) = t\,\sigma_\varepsilon^2 + \sigma_B^2$.

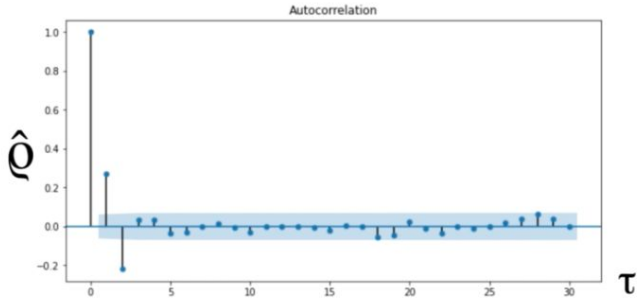
Autocorrelation function

$$Stationary: \ \varrho(Y(t_1), Y(t_2)) = \varrho(t_1 - t_2) = \varrho(\tau)$$

- Also known as correlogram
- Autocorrelation is to autocovariance as correlation is to covariance
- Auto = Self (both RVs come from the same time series)

$$Autocovariance: \ cov(Y_{t_1}, Y_{t_2})$$

$$Autocorrelation: \ \frac{cov(Y_{t_1}, Y_{t_2})}{\sigma_Y(t_1)\sigma_Y(t_2)}$$



How to determine q in MA(q)

- Assign q to be the maximum non-zero lag
- E.g. in below chart, q = 2
- Usually, the ACF for lags < q are also non-zero

Why does it work ?

- This can be derived mathematically! (we won't do it right now)
- For MA(1):

$$y_t = c + \theta_1 \varepsilon_{t-1} + \varepsilon_t, \ where \ \varepsilon_t \sim \aleph(0, \sigma^2)$$

$$\varrho(1) = \frac{\theta_1}{1+\theta_1{}^2}, \ \varrho(\tau) = 0 \ for \ \tau > 1$$

- For MA(2):

$$\varrho(1) = \frac{\theta_1+\theta_1\theta_2}{1+\theta_1{}^2+\theta_2{}^2}, \ \varrho(2) = \frac{\theta_2}{1+\theta_1{}^2+\theta_2{}^2}, \ \varrho(\tau) = 0 \ for \ \tau > 2$$

PACF of order p of AR(p)

- Definition: The PACF at lag τ is the autocorrelation between Y(t) and Y(t+τ), *conditioned on* Y(t+1), Y(t+2), ..., Y(t+τ-1)

$$\varphi(\tau,\tau) = corr(Y_{t+\tau} - \hat{Y}_{t+\tau}, Y_t - \hat{Y}_t)$$

$$\hat{Y}_{t+\tau} = \beta_0 + \beta_1 Y_{t+1} + \beta_2 Y_{t+2} + \ldots \beta_{\tau-1} Y_{t+\tau-1}$$

$$\hat{Y}_t = \beta_0' + \beta_1' Y_{t+1} + \beta_2' Y_{t+2} + \ldots \beta_{\tau-1}' Y_{t+\tau-1}$$

AIC & BIC criteria for hyperparameters selection

- Difference is in the penalty term
- Auto ARIMA uses AIC by default (but usually both give same answer)

$$AIC = 2(\#params) - 2\log L$$

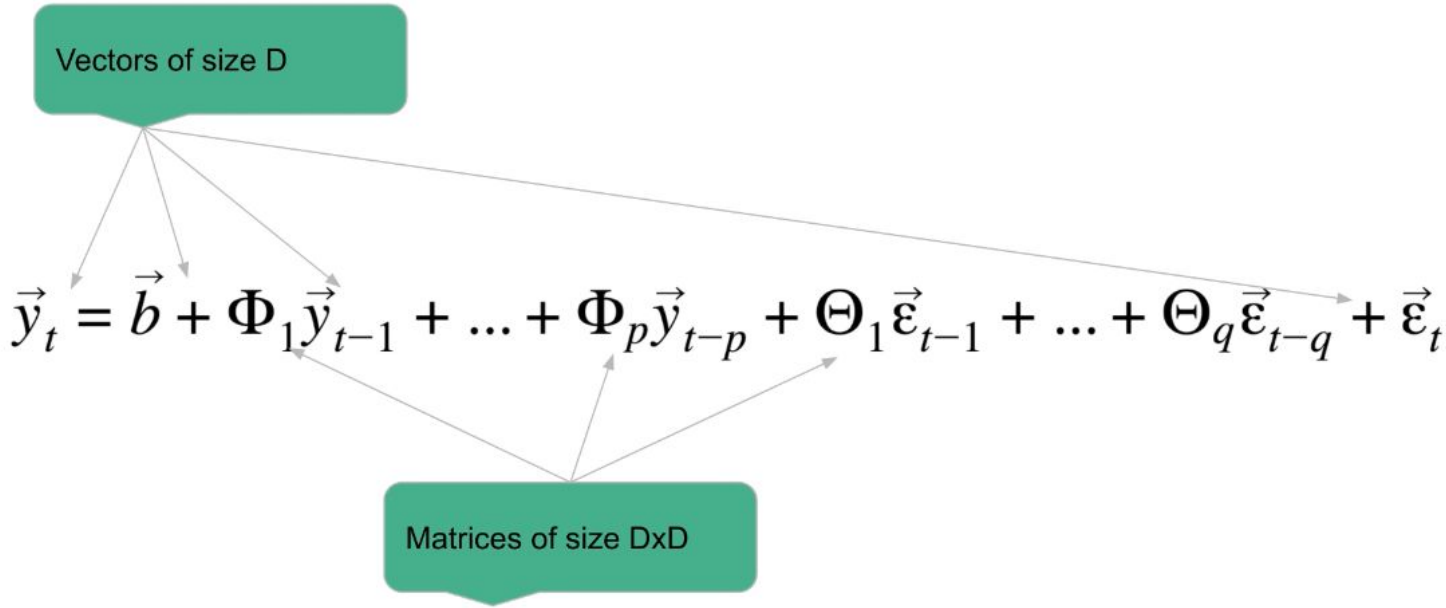
$$BIC = (\#params)\log T - 2\log L$$

$$L = likelihood$$

$$T = length\ of\ time\ series\ (number\ of\ data\ points)$$

Vector Autoregressive Moving Average

VARMA



Cross terms from multivariate series

- VAR(1)

$$y^{(1)}{}_t = b^{(1)} + \varphi_{11} y^{(1)}{}_{t-1} + \varphi_{12} y^{(2)}{}_{t-1} + \varepsilon^{(1)}{}_t$$
$$y^{(2)}{}_t = b^{(2)} + \varphi_{21} y^{(1)}{}_{t-1} + \varphi_{22} y^{(2)}{}_{t-1} + \varepsilon^{(2)}{}_t$$

- 2 separate AR(1)'s

$$y^{(1)}{}_t = b^{(1)} + \varphi_{11} y^{(1)}{}_{t-1} + \varepsilon^{(1)}{}_t$$
$$y^{(2)}{}_t = b^{(2)} + \varphi_{22} y^{(2)}{}_{t-1} + \varepsilon^{(2)}{}_t$$

Model Identifiability is a complex topic

- Fact: VARMA models are not unique
- Cannot uniquely determine p and q
- We'll see this warning in statsmodels

Sparse Identification and Estimation of Large-Scale Vector AutoRegressive Moving Averages

Ines Wilms[a]*, Sumanta Basu[b]*, Jacob Bien[c]†, and David S. Matteson[b]

[a] *Department of Quantitative Economics, Maastricht University, Maastricht, The Netherlands*

[b] *Department of Statistics and Data Science, Cornell University, Ithaca, NY, USA*

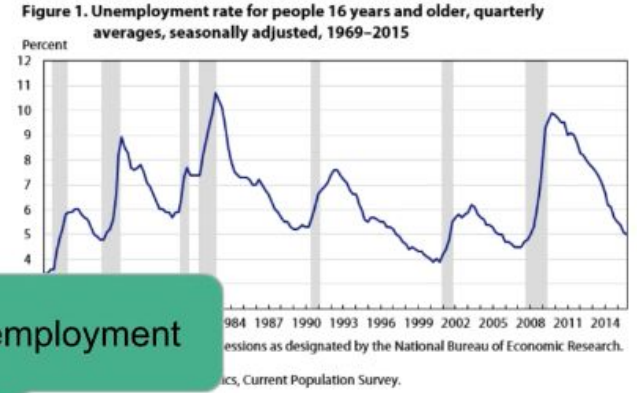
[c] *Data Sciences and Operations, University of Southern California, Los Angeles, CA, USA*

Abstract. The Vector AutoRegressive Moving Average (VARMA) model is fundamental to the theory of multivariate time series; however, identifiability issues have led practitioners to abandon it in favor of the simpler but more restrictive Vector AutoRegressive (VAR) model.

No VARIMA as I will change according to univariate series

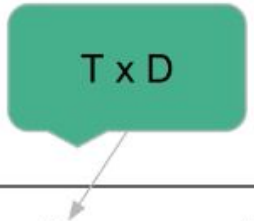
- We only want to use VARMA / ARMA if the time series is stationary
- Can find some packages / discussion for VARIMA, but not in statsmodels
- Each component of your time series may need to be differenced a different number of times than the others! (i.e. there isn't one "d" value)

GDP



Unemployment

VARMA(p,q) code + exogenous data




```
model = VARMAX(train, order=(p, q))
res = model.fit(maxiter=100)
fcast = res.get_forecast(Ntest)
train_pred = res.fittedvalues
test_pred = fcast.predicted_mean # df with 'col1', … 'colD'
ci = fcast.conf_int() # df with 'lower col1', 'upper col1', …
```

Granger causality

- Recall: in regression analysis, we determine whether
statistically significant by testing if it's "big enough" (

```
> summary(m)

Call:
lm(formula = y ~ u + v + w)

Residuals:
    Min      1Q  Median      3Q     Max
-3.3965 -0.9472 -0.4708  1.3730  3.1283

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.4222     1.4036   1.013  0.32029
u             1.0359     0.2811   3.685  0.00106 **
v             0.9217     0.3787   2.434  0.02211 *
w             0.7261     0.3652   1.988  0.05744 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.625 on 26 degrees of freedom
Multiple R-squared: 0.4981,     Adjusted R-squared: 0.4402
F-statistic: 8.603 on 3 and 26 DF,  p-value: 0.0003915
```

- Model 1:

$$y(t) = b + \phi_1 y(t-1) + \ldots \phi_p y(t-p) + \varepsilon(t)$$

- Model 2:

$$y(t) = b + \phi_1 y(t-1) + \ldots \phi_p y(t-p) + \alpha_3 x(t-3) + \alpha_7 x(t-7) + \varepsilon(t)$$

If any of these cross-coefficients are statistically significant, then we say x(t) "Granger causes" y(t) – note: you can pick any lags, these are just examples

Cointegration

Integration of order d [edit]

A time series is integrated of order *d* if

$$(1-L)^d X_t$$

is a stationary process, where L is the lag operator and $1-L$ is the first difference, i.e.

$$(1-L)X_t = X_t - X_{t-1} = \Delta X.$$

In other words, a process is integrated to order *d* if taking repeated differences *d* times yields a stationary process.

In particular, if a series is integrated of order 0, then $(1-L)^0 X_t = X_t$ is stationary.

Cointegration

Constructing an integrated series [edit]

An *I*(*d*) process can be constructed by summing an *I*(*d* – 1) process:

- Suppose X_t is *I*(*d* – 1)
- Now construct a series $Z_t = \sum_{k=0}^{t} X_k$
- Show that *Z* is *I*(*d*) by observing its first-differences are *I*(*d* – 1):

$$\Delta Z_t = X_t,$$

where

$$X_t \sim I(d-1).$$

Cointegration

Cointegration is a statistical property of a collection $(X_1, X_2, ..., X_k)$ of time series variables. First, all of the series must be integrated of order *d*. Next, if a linear combination of this collection is integrated of order less than d, then the collection is said to be co-integrated. Formally, if (*X*,*Y*,*Z*) are each integrated of order *d*, and there exist coefficients *a*,*b*,*c* such that $aX + bY + cZ$ is integrated of order less than d, then *X*, *Y*, and *Z* are cointegrated. Cointegration has become an important property in contemporary time series analysis. Time series often have trends—either deterministic or stochastic. In an influential paper,[1] Charles Nelson and Charles Plosser (1982) provided statistical evidence that many US macroeconomic time series (like GNP, wages, employment, etc.) have stochastic trends.

Engle-Granger methodologies for order one series

Engle–Granger two-step method [edit]

See also: Error correction model § Engle and Granger 2-step approach

If x_t and y_t both have order of integration d=1 and are cointegrated, then a linear combination of them must be stationary for some value of β and u_t . In other words:

$$y_t - \beta x_t = u_t$$

where u_t is stationary.

If β is known, we can test u_t for stationarity with an Augmented Dickey–Fuller test or Phillips–Perron test. If β is unknown, we must first estimate it. This is typically done by using ordinary least squares (by regressing y_t on x_t and an intercept). Then, we can run an ADF test on u_t. However, when β is estimated, the critical values of this ADF test are non-standard, and increase in absolute value as more regressors are included.[7]

If the variables are found to be cointegrated, a second-stage regression is conducted. This is a regression of Δy_t on the lagged regressors, Δx_t and the lagged residuals from the first stage, $\hat{u}_{t-1}$. The second stage regression is given as:
$\Delta y_t = \Delta x_t b + \alpha u_{t-1} + \varepsilon_t$

If the variables are not cointegrated (if we cannot reject the null of no cointegration when testing u_t), then $\alpha = 0$ and we estimate a differences model: $\Delta y_t = \Delta x_t b + \varepsilon_t$

Business oriented

- Built with business datasets in mind
- Model sales, CPU usage, adspend, etc.
- Do you run a business? Do you count anything? That's a time series!
- Easy to account for weekends, holidays, special events
- Easy to add custom exogenous inputs (e.g. NFL Sundays)
- Excellent choice for consultants and freelancers

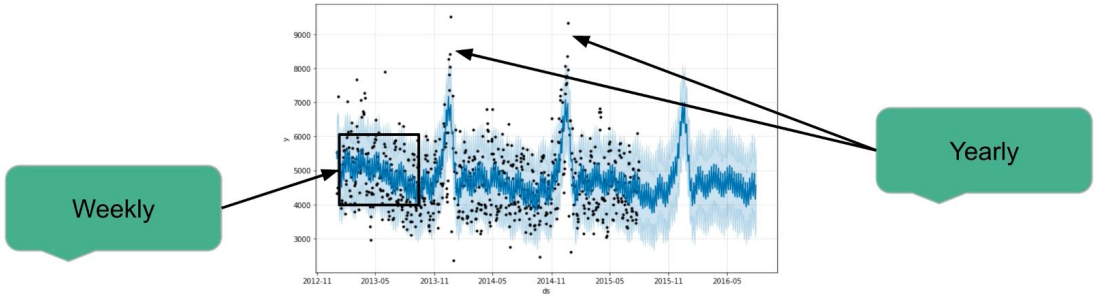
Pragmatic

- Just a few lines of code and relatively fast
- Recall: ARIMA can struggle with large T
- Most things (even cross-validation) are just 1 line of code

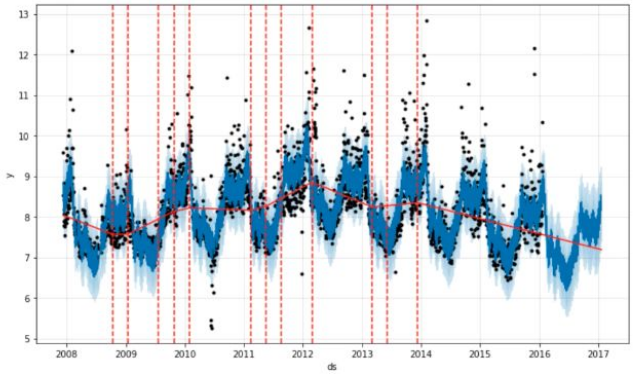
Interpretable, bayesian for client interaction

- Very interpretable
- Reminiscent of ETS methods
- Plot components with a function call
- Seasonality at multiple scales
  E.g. weekly / yearly
- E.g. decrease on Sunday, but increase on Black Friday

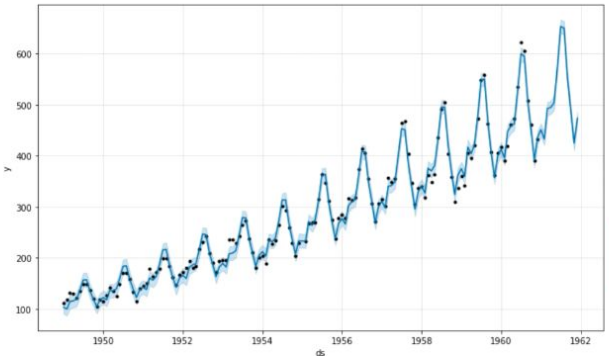
- Seasonality at multiple scales
- Weekly: less customers on Sundays
- Yearly: Christmas, New Years, Valentine's Day, etc.
- Cyclical but not exactly periodic: Thanksgiving, Black Friday, Easter, etc.
- Lunar Calendar: Lunar New Year



- Changes in trend
- Might be due to new website design, new product release, etc.



- Easily deal with outliers and missing data



Component of Prophet model

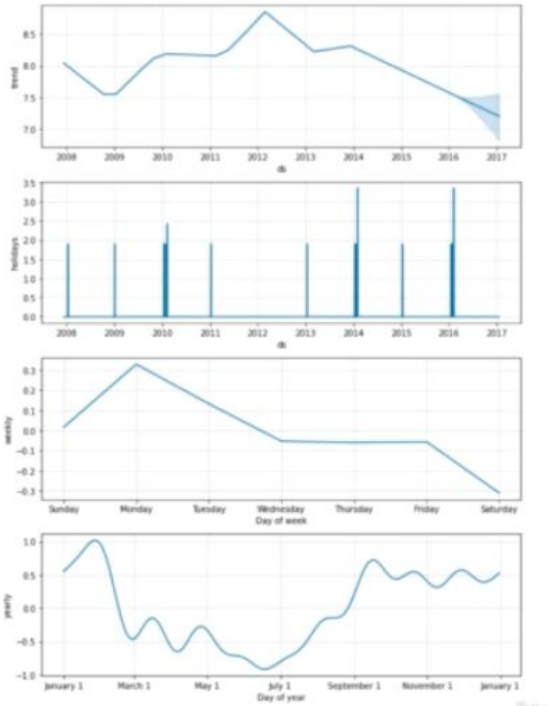
$$y(t) = g(t) + s(t) + h(t) + \varepsilon(t)$$

$g(t)$: *trend*

$s(t)$: *seasonal*

$h(t)$: *holiday*

$\varepsilon(t)$: *error*

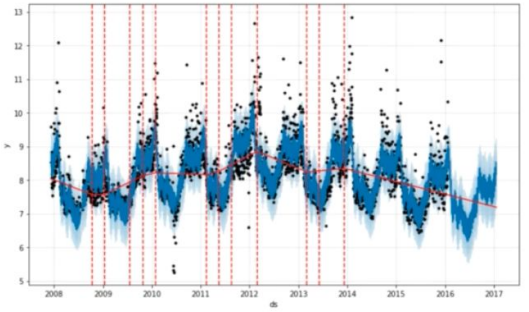


Trend g(t)

- Linear

slope x time + intercept

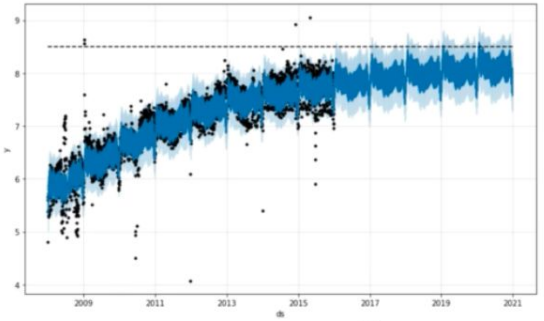
$$g(t) = (k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})t + (m + \mathbf{a}(t)^\intercal \boldsymbol{\gamma})$$



- Logistic

C(t) * sigmoid(linear(t))

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



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

Seasonal s(t)

- Fourier series - don't worry if you're not an engineer / physicist
- We know that sines and cosines are periodic
- P = 365.25 for yearly data, P = 7 for weekly, when time scaled in days

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

Holydays h(t)

- One-hot encode every holiday
- Influence of each holiday governed by components of κ (additive by default)
- Can also model surrounding days, but paper doesn't specify how

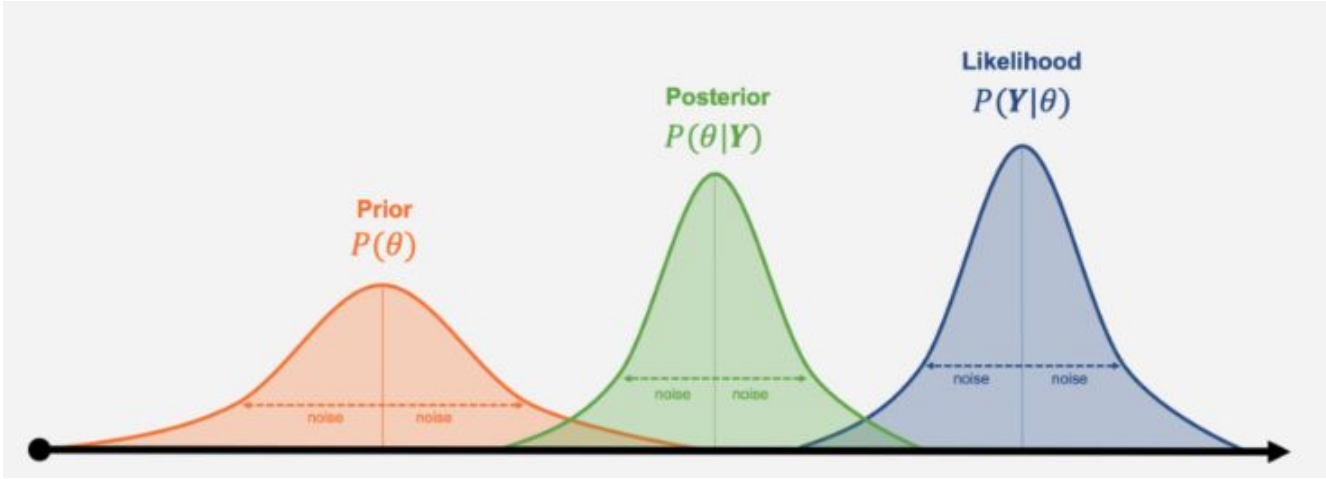
$$Z(t) = [\mathbf{1}(t \in D_1), \ldots, \mathbf{1}(t \in D_L)]$$

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

we use a prior $\boldsymbol{\kappa} \sim \text{Normal}(0, \nu^2)$.

Bayesian model

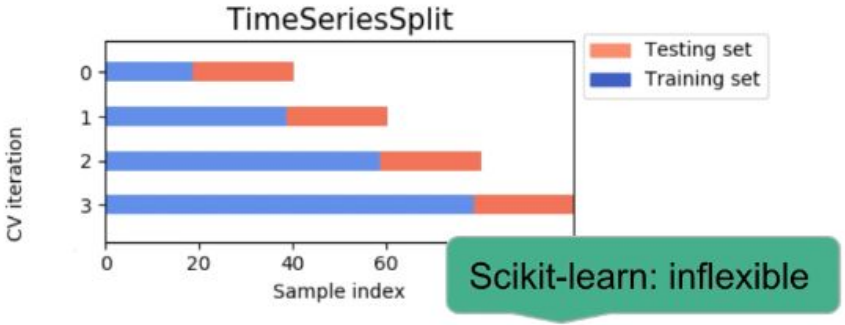
- Hinted at on previous slide - all parameters have priors
- Behind the scenes, Prophet uses Stan (a Bayesian ML library)
- Allows for "human in the loop" optimization
- i.e. include your client on the modeling process (e.g. modify changepoint prior to yield the appropriate number of trend changes)



Prophet Cross Validation : Walk-Forward validation

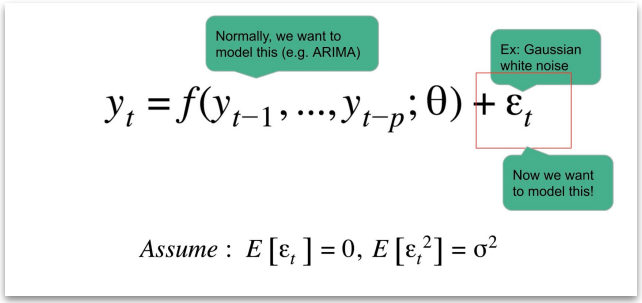
- By coincidence, it is exactly like I described it in "Walk-Forward Validation"

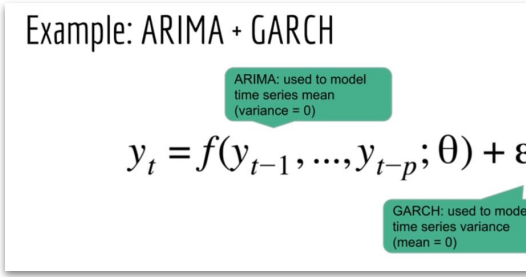
```
df_cv = cross_validation(
    m,
    initial='730 days',
    period='30 days',
    horizon='60 days')
```



Modeling heteroscedasticity

- ARCH
- GARCH
- NGARCH
- IGARCH
- EGARCH
- GARCH-M
- QGARCH
- GJR-GARCH
- TGARCH
- fGARCH
- COGARCH
- ZD-GARCH





ARCH

$$\epsilon_t = \sigma_t z_t$$

The random variable z_t is a strong white noise process. The series σ_t^2 is modeled by

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2,$$

where $\alpha_0 > 0$ and $\alpha_i \geq 0,\ i > 0$.

GARCH

$$y_t = x_t' b + \epsilon_t$$

$$\epsilon_t | \psi_{t-1} \sim \mathcal{N}(0, \sigma_t^2)$$

$$\sigma_t^2 = \omega + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_p \sigma_{t-p}^2 = \omega + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2$$

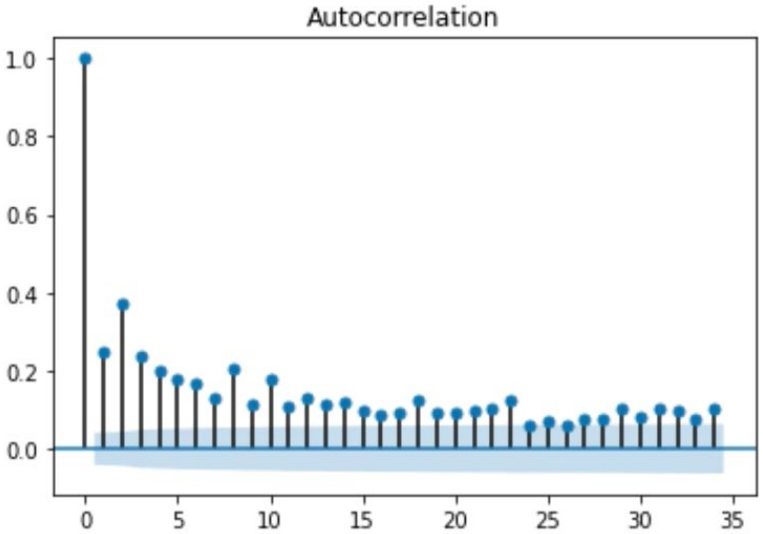
GJR-ARCH

$$\sigma_t^2 = K + \delta\, \sigma_{t-1}^2 + \alpha\, \epsilon_{t-1}^2 + \phi\, \epsilon_{t-1}^2 I_{t-1}$$

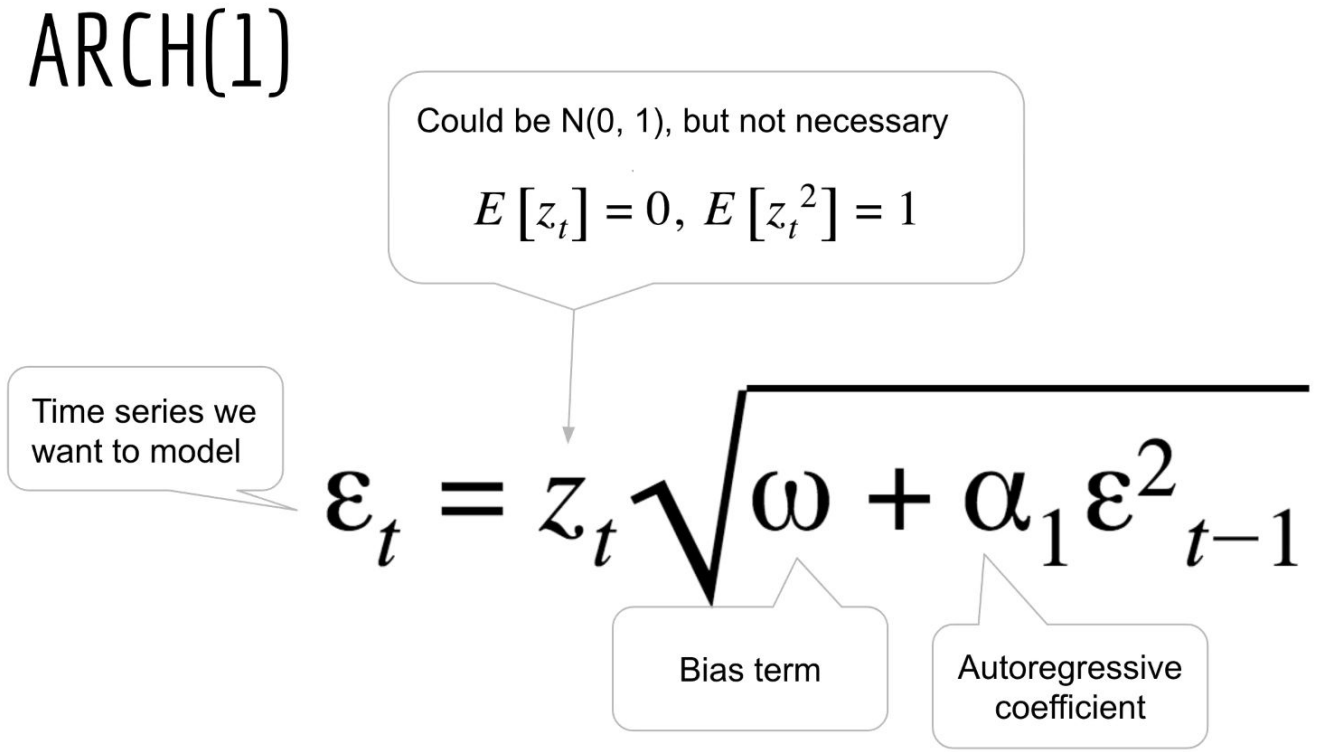
where $I_{t-1} = 0$ if $\epsilon_{t-1} \geq 0$, and $I_{t-1} = 1$ if $\epsilon_{t-1} < 0$.

ACF of squared returns show auto-regressivity in variance

- For stocks, log returns ACF shows randomness (I(1) is the best model)
- The ACF of squared log returns does not look random at all



ARCH(1)


ARCH(1)
Could be N(0, 1), but not necessary
$E[z_t] = 0,\ E[z_t^2] = 1$
Time series we want to model
$\varepsilon_t = z_t \sqrt{\omega + \alpha_1 \varepsilon^2_{t-1}}$
Bias term
Autoregressive coefficient

$$\varepsilon_t = z_t \sigma_t$$

=> **epsilon has variance sigma squared**

$$\varepsilon_t^2 = z_t^2(\omega + \alpha_1 \varepsilon^2_{t-1})$$

$$\varepsilon_t = z_t \sigma_t$$

$$\frac{\varepsilon_t^2}{z_t^2} = \sigma_t^2 = \omega + \alpha_1 \varepsilon^2$$

- There must be constraints, since variance cannot be negative

$$\varepsilon_t = z_t \sqrt{\omega + \alpha_1 \varepsilon^2_{t-1}}$$

- General constraints

$$\omega > 0,\ 1 \geq \alpha_1 \geq 0$$

- Constraint for stationarity and finite variance

$$\alpha_1 < 1$$

Conditional versus unconditional variance

$$E\left[\varepsilon_t^2 \mid \varepsilon_{t-1},\ \varepsilon_{t-2},\ ...,\ \varepsilon_{t-p}\right] = \sigma_t^2$$

$$E\left[\varepsilon_t^2\right] = \ ?\ (call\ it\ \sigma^2)$$

$$E\left[\varepsilon_t^2\right] = E\left[z_t^2\right] E\left[\omega + \alpha_1 \varepsilon_{t-1}^2\right]$$

$$\sigma^2 = 1 \times (\omega + \alpha_1 \sigma^2)$$

$$(1 - \alpha_1)\sigma^2 = \omega$$

$$\sigma^2 = \frac{\omega}{1 - \alpha_1}$$

We model the variance ! Mean c from an ARIMA

$$E\left[\varepsilon_t \mid \varepsilon_{t-1}, \varepsilon_{t-2}, ..., \varepsilon_{t-p}\right] = 0$$

$$E\left[\varepsilon_t\right] = 0$$

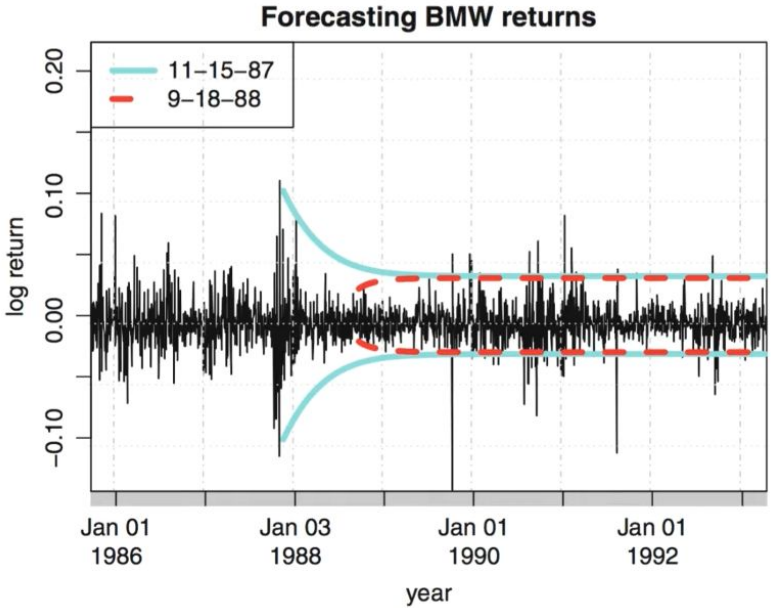
$$cov(\varepsilon_t, \varepsilon_{t-s}) = 0,\ \forall s \neq 0$$

$$\varepsilon_t \perp \varepsilon_{t-s} \ \rightarrow\ cov(\varepsilon_t, \varepsilon_{t-s}) = 0$$

$$cov(\varepsilon_t, \varepsilon_{t-s}) = 0 \ \not\rightarrow\ \varepsilon_t \perp \varepsilon_{t-s}$$

Long term forecast converge to the unconditional variance

- Our forecast of variance will also converge to unconditional variance
- Meaning: ARCH and GARCH are useful for short-term forecasts
- Makes sense: we cannot predict Elon Musk making a tweet about Bitcoin, or some country banning a cryptocurrency exchange
- Obvious example: COVID-19
- Imagine if time series models could predict world-scale events! (of course, this is unreasonable)
- Aside from insider trading, you probably cannot predict these

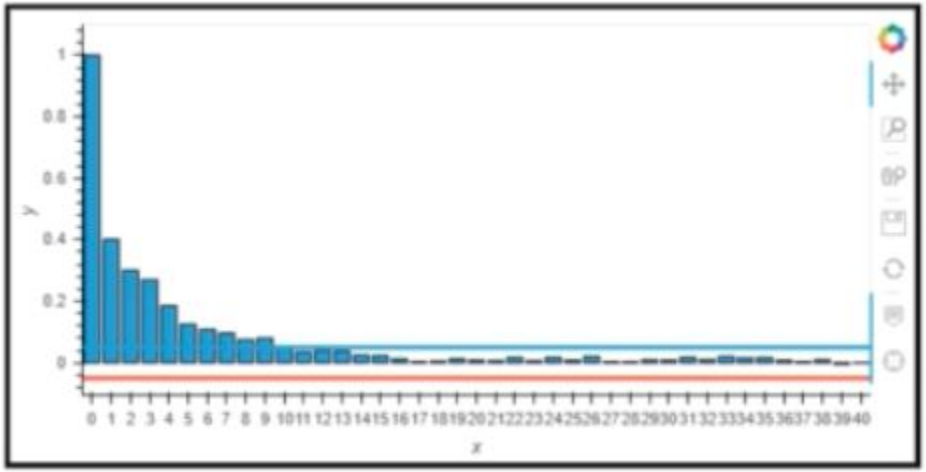


ARCH(p)

$$\varepsilon_t = z_t \sqrt{\omega + \sum_{\tau=1}^{p} \alpha_\tau \varepsilon^2{}_{t-\tau}}$$

ACF of squared epsilons : geometric decay

$$\varrho_{\varepsilon^2}(h) = \alpha_1^{|h|}$$



- Unconditional mean is zero
- Conditional mean is zero
- ACF of series shows no correlation with lags (just like stock returns)
- ACF of squared series does show correlation (just like stock returns)
- Allows us to model volatility clustering

GARCH(p,q) : persistent volatility, less HFT moves

ARCH(p) : AR(p)
GARCH(p,q) : ARMA(p,q)

There are other ways to form this analogy, but I think you get the idea...

$$\sigma_t^2 = \omega + \sum_{k=1}^{p} \alpha_k \varepsilon^2_{t-k} + \sum_{k=1}^{q} \beta_k \sigma^2_{t-k}$$

Persistent volatility

$$\varepsilon_t = z_t \sigma_t$$

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon^2_{t-1} + \beta_1 \sigma^2_{t-1}$$

- Consider a GARCH(1,1) for simplicity (actually, this is the most common)
- ε_{t-1} term is random, σ_{t-1} term is not random
- The "randomness" comes from z_t
- Imagine $\beta_1 = 0.9$ - then σ_t would decay slowly over time

GARCH(1,1) as ARMA(1,1)

$$\eta_t = \varepsilon_t^2 - \sigma_t^2$$

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon^2_{t-1} + \beta_1 \sigma^2_{t-1}$$

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon^2_{t-1} + \beta_1 (\varepsilon^2_{t-1} - \eta_{t-1})$$

$$\sigma_t^2 = \omega + (\alpha_1 + \beta_1) \varepsilon^2_{t-1} - \beta_1 \eta_{t-1}$$

$$\sigma_t^2 + \eta_t = \omega + (\alpha_1 + \beta_1) \varepsilon^2_{t-1} - \beta_1 \eta_{t-1} + \eta_t$$

$$\varepsilon_t^2 = \omega + (\alpha_1 + \beta_1) \varepsilon^2_{t-1} - \beta_1 \eta_{t-1} + \eta_t$$

$$\eta_t = \varepsilon_t^2 - \sigma_t^2$$

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon^2_{t-1}$$

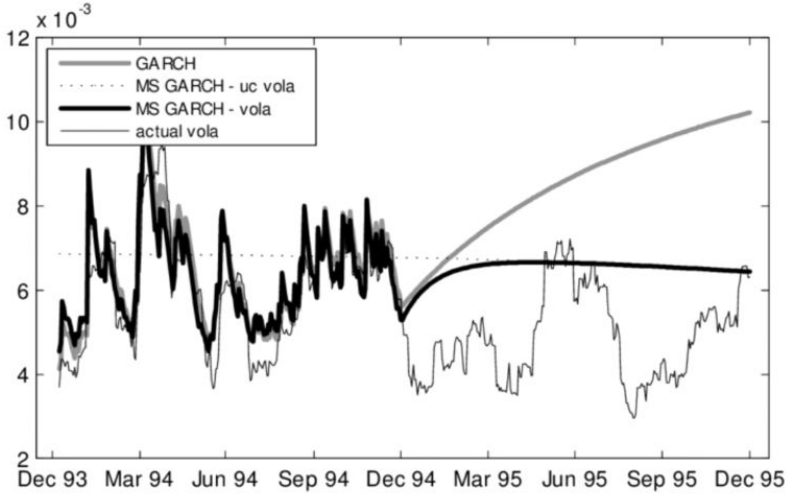
$$\sigma_t^2 + \eta_t = \omega + \alpha_1 \varepsilon^2_{t-1} + \eta_t$$

$$\varepsilon_t^2 = \omega + \alpha_1 \varepsilon^2_{t-1} + \eta_t$$

$$y_t^2 = \omega + \alpha_1 y^2_{t-1} + \eta_t, \text{ where } y_t = \varepsilon_t$$

Making prediction of variance

- We want to predict the statistics of ε_t (e.g. its mean and variance)
- We care about variance (or standard deviation - i.e. volatility)



A deep learning approach to GARCH

1) How does it make predictions? (i.e. go from input to output)
   Usually some equation involving model parameters (ω, α, β for GARCH)
2) How do we find those model parameters?
   E.g. Deep Learning: "call fit" / "gradient descent"

- For deep learning, ARIMA, and many other ML models, the objective is based on the log-likelihood
- Maximum Likelihood Estimation (MLE): maximize the likelihood wrt model parameters - this is the "training process"
- Regression: typically minimize $\sum_i (y_i - \hat{y}_i)^2$ (squared error)
- Equivalent to maximizing likelihood when errors are normal
- This requires another assumption! The variance of errors is constant

Maximum likelihood in case of homoscedasticity

$$Assumption: \; y_t \sim \mathcal{N}(\hat{y}_t, \sigma^2)$$

$$Equivalent: \; y_t = \hat{y}_t + \varepsilon_t, \; where \; \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$$

$$\hat{y}_t(\theta) = f(y_{t-1}, \ldots, y_{t-p}; \theta)$$

$$L(\theta) = \prod_{t=1}^{T} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(y_t - \hat{y}_t)^2}{\sigma^2}\right)$$

$$l(\theta) = \log L(\theta) = \sum_{t=1}^{T} \left\{-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\frac{(y_t - \hat{y}_t)^2}{\sigma^2}\right\}$$

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{T} \left\{-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2}\frac{(y_t - \hat{y}_t)^2}{\sigma^2}\right\}$$

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{T} \left\{-\frac{1}{2}\frac{(y_t - \hat{y}_t)^2}{\sigma^2}\right\}$$

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{T} \left\{-(y_t - \hat{y}_t)^2\right\}$$

$$\theta^* = \arg\min_{\theta} \sum_{t=1}^{T} \left\{(y_t - \hat{y}_t)^2\right\}$$

Max likelihood in case of hetero scedasticity

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{T} \left\{ -\frac{1}{2}\log(2\pi\sigma_t^2) - \frac{1}{2}\frac{(y_t - \hat{y}_t)^2}{\sigma_t^2} \right\}$$

$$\theta^* = \arg\max_{\theta} \sum_{t=1}^{T} \left\{ -\frac{1}{2}\log(\sigma_t^2) - \frac{1}{2}\frac{(y_t - \hat{y}_t)^2}{\sigma_t^2} \right\}$$

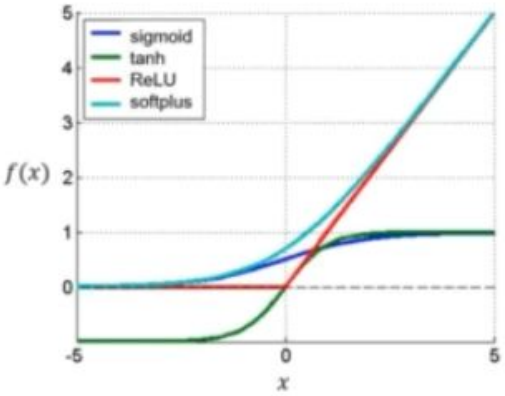
$$\theta^* = \arg\min_{\theta} \sum_{t=1}^{T} \left\{ \log(\sigma_t^2) + \frac{(y_t - \hat{y}_t)^2}{\sigma_t^2} \right\}$$

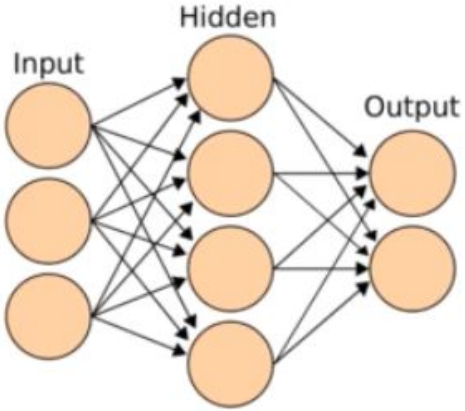
$$\theta^* = \arg\min_{\theta} \sum_{t=1}^{T} \left\{ \log(\sigma_t^2) + \frac{\varepsilon_t^2}{\sigma_t^2} \right\}$$

- Can plug into any stock optimizer (e.g. L-BFGS) - see Scipy docs
- We also need to include constraints on ω, α, β
- Easy to plug in with any optimizer
- Note: Derivation based on normal, but similar steps can be done for t-distribution, skewed t-distribution, etc.

The deep learning way

- Why stick with GARCH? (a linear model)
- We can parameterize σ_t using a neural network
- Just need to know how to make custom loss functions in TF2/PyTorch/...
- How to constraint output to be positive?
- Use positive-only activation (e.g. softplus)





Prerequisites for machine learning

Tensorflow 2.0

Tensorflow 2.0: Deep Learning and Artificial Intelligence

Lazy Programmer Inc., Lazy Programmer Team

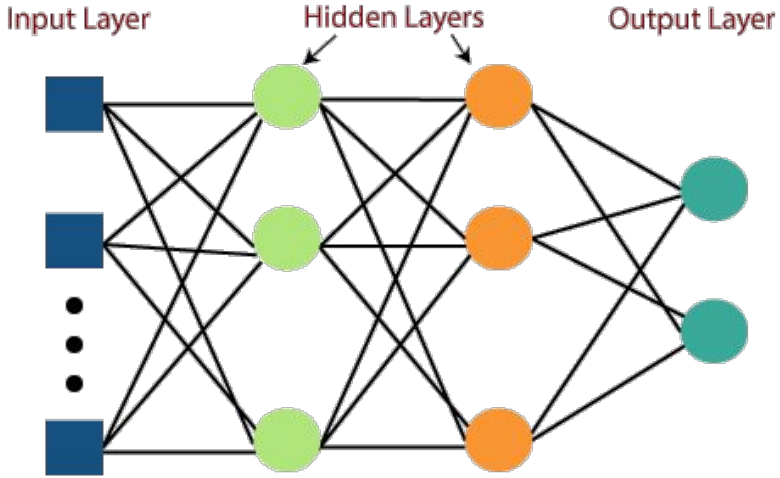
PyTorch

PyTorch: Deep Learning and Artificial Intelligence

Lazy Programmer Team, Lazy Programmer Inc.

In-depth deep learning series (up to CNNs and RNNs)

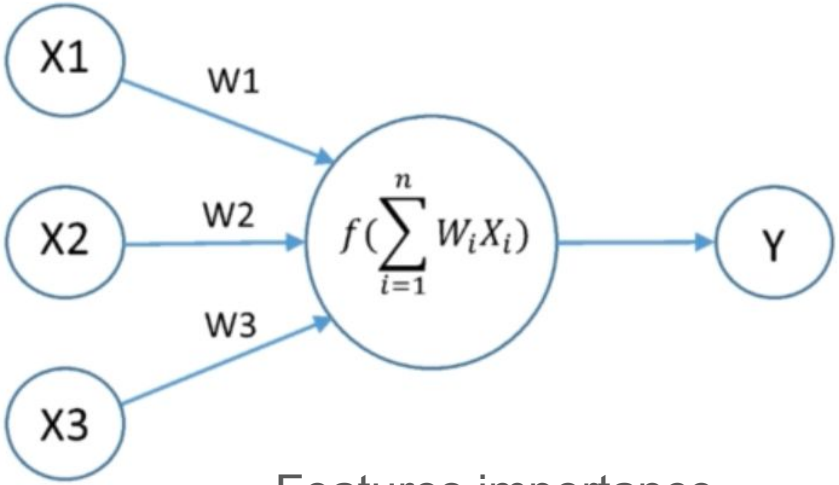
Multi-Layer perceptron



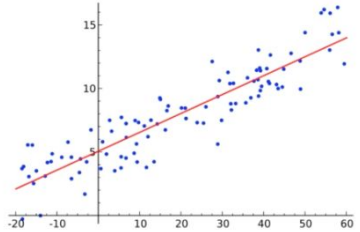
- https://github.com/fastai/course22/blob/master/04-how-does-a-neural-net-really-work.ipynb
- https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/video-lecture?hl=fr

$$A\ line:\ ax+b$$

$$A\ neuron:\ \sigma(w^T x+b)$$

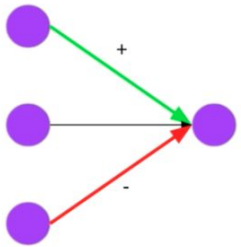


1d regression



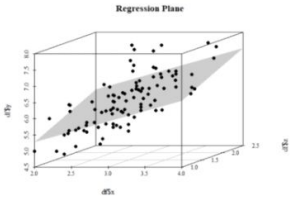
$$\hat{y} = mx + b\ (or\ ax + b)$$

Features importance



2d regression

$$\hat{y} = w_1 x_1 + w_2 x_2 + b$$

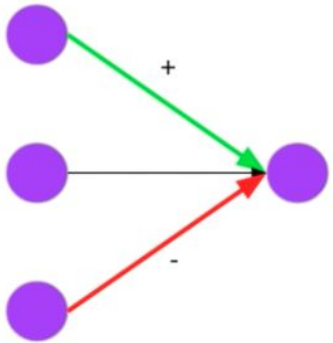


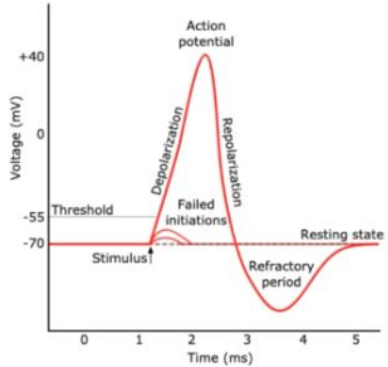
On the importance of introducing non-linearities through activation

- https://github.com/fastai/course22/blob/master/04-how-does-a-neural-net-really-work.ipynb
- Introduction aux réseaux de neurones (google)

The “threshold” in this case is -b (negative)

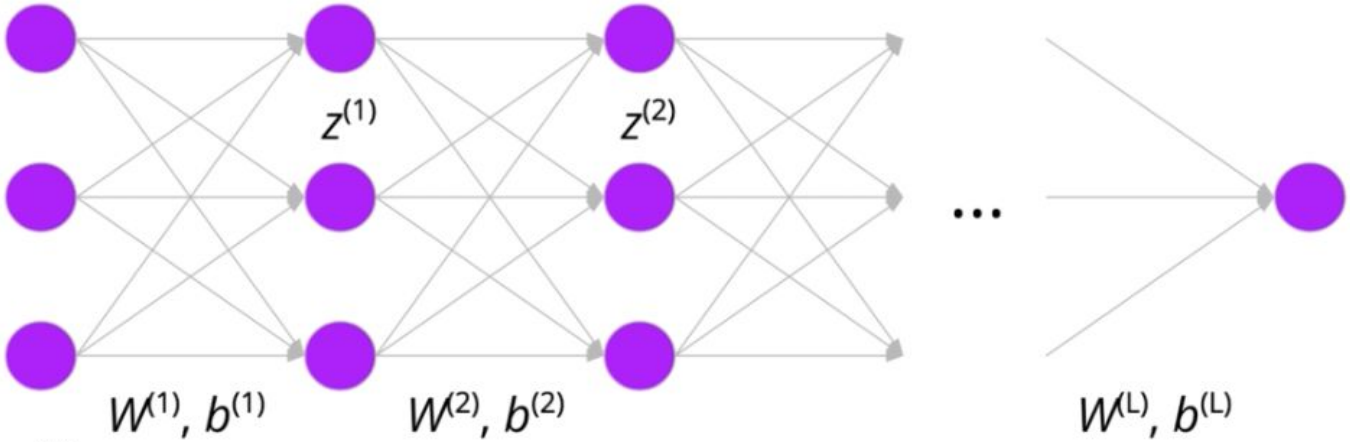
As long as $w^T x$ is > -b, then $w^T x + b > 0$, and the prediction is 1





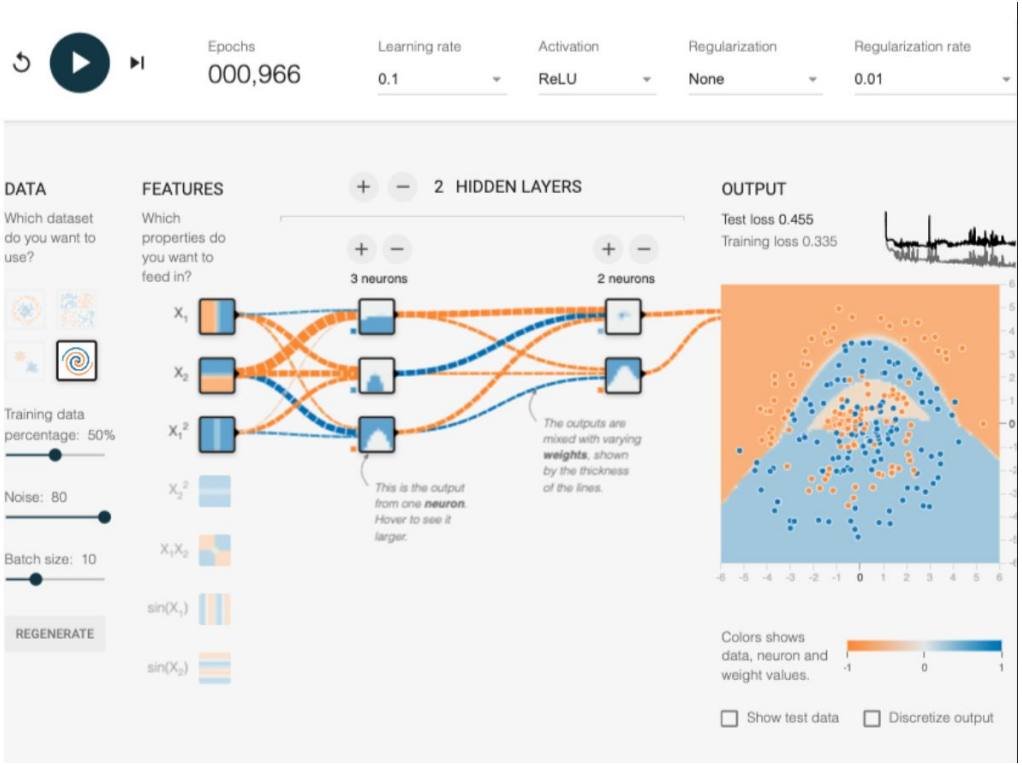
$$p(y = 1 \mid x) = \sigma(w^T x + b) = \sigma(\sum_{d=1}^{D} w_d x_d + b)$$

Stacking neurons



$$z^{(1)} = \sigma(W^{(1)T} x + b^{(1)})$$
$$z^{(2)} = \sigma(W^{(2)T} z^{(1)} + b^{(2)})$$
$$z^{(3)} = \sigma(W^{(3)T} z^{(2)} + b^{(3)})$$
$$\dots$$
$$p(y = 1 \mid x) = \sigma(W^{(L)T} z^{(L-1)} + b^{(L)})$$

Fitting complex patterns through non-linear activation function and multiple layers

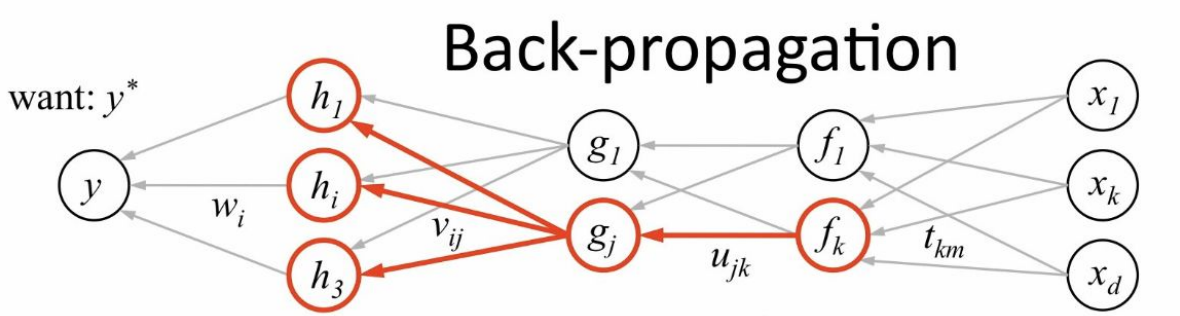


Building MLP from scratch (tensorflow and pytorch)

- https://github.com/fastai/course22/blob/master/05-linear-model-and-neural-net-from-scratch.ipynb
- https://colab.research.google.com/github/google/eng-edu/blob/main/ml/cc/exercises/linear_regression_with_synthetic_data.ipynb

The matrix calculus you need for deep learning

https://explained.ai/matrix-calculus



1. receive new observation $\boldsymbol{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

(a) compute error on g_j

$$\underbrace{\frac{\partial E}{\partial g_j}}_{\text{should } g_j \text{ be higher or lower?}} = \sum_i \underbrace{\sigma'(h_i) v_{ij}}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

(b) for each u_{jk} that affects g_j

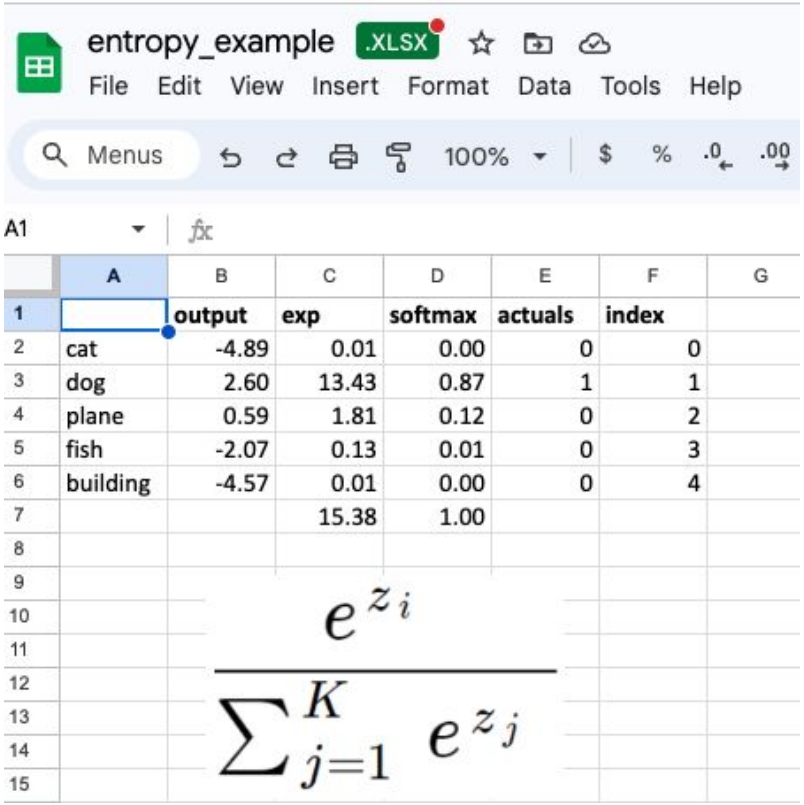
(i) compute error on u_{jk}

$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

(ii) update the weight

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

Multi-classification : soft-max and cross-entropy



	A	B	C	D	E	F	G
1		output	exp	softmax	actuals	index	
2	cat	-4.89	0.01	0.00	0	0	
3	dog	2.60	13.43	0.87	1	1	
4	plane	0.59	1.81	0.12	0	2	
5	fish	-2.07	0.13	0.01	0	3	
6	building	-4.57	0.01	0.00	0	4	
7			15.38	1.00			
8							
9							
10							
11							
12							
13							
14							
15							

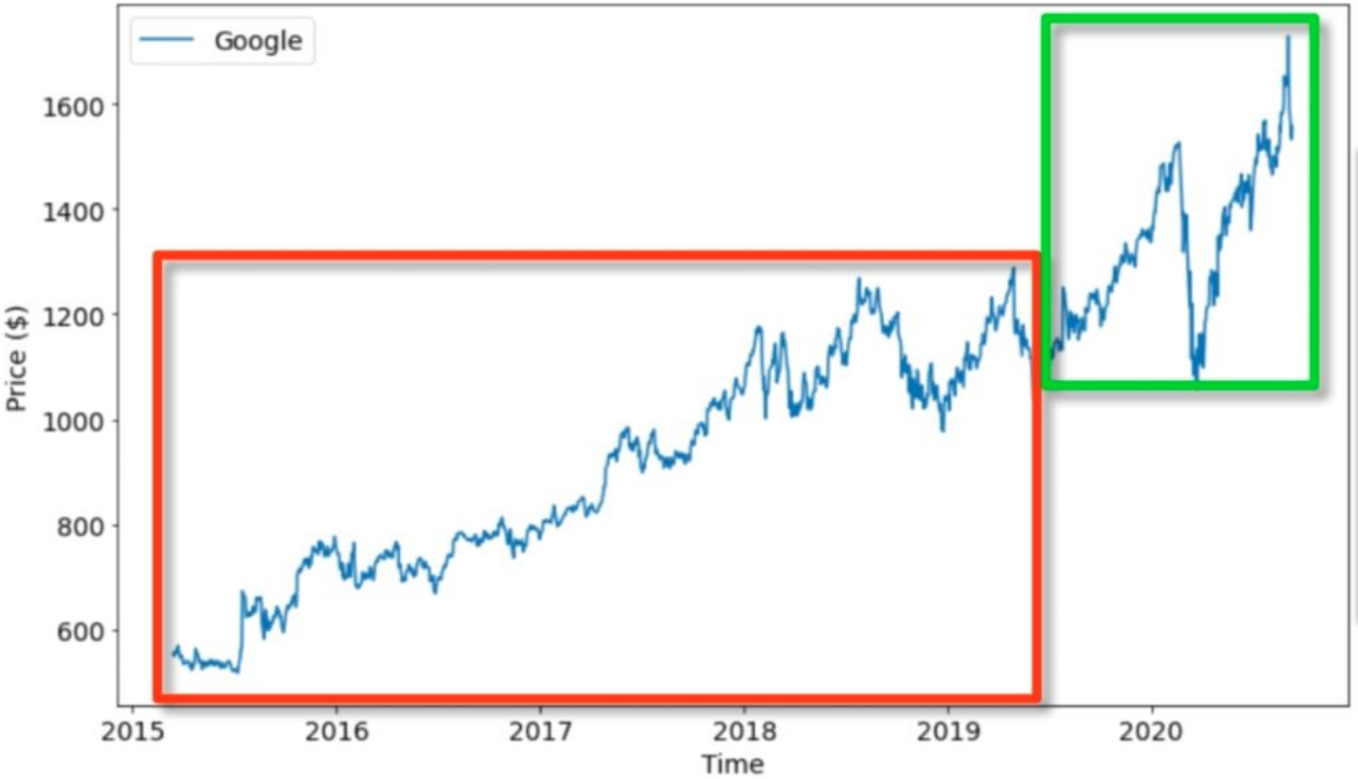
$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

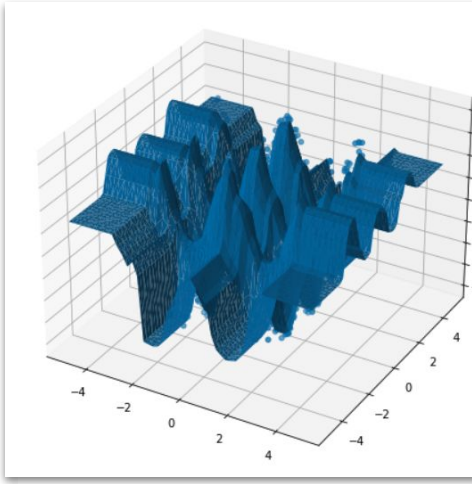
$$H(P^* \mid P) = -\sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTIRBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTIRBUTION}}$$

ML & time-series analysis

See ML_ notebooks

Extrapolation





Training a model on difference or not ?

$$\Delta y_t = y_t - y_{t-1}$$

$$Training\ data: \{\Delta y_1, \Delta y_2, \Delta y_3, ..., \Delta y_T\}$$

$$Train\ Predictions: \{\hat{\Delta} y_1, \hat{\Delta} y_2, ..., \hat{\Delta} y_T\}$$

Our prediction of th
first difference

$$\hat{\Delta} y_t \approx y_t - y_{t-1}$$

$$\hat{y}_t = y_{t-1} + \hat{\Delta} y_t$$

$$\hat{y}_{T+3} = y_T + \hat{\Delta} y_{T+1} + \hat{\Delta} y_{T+2} + \hat{\Delta} y_{T+3}$$

Fourier Transform

Convolution pour les signaux finis

Soient f, h des signaux discrets $f, h \in \mathbb{R}^{\mathbb{Z}}$. On peut définir la convolution :

$$(f * h)[n] = \sum_{p \in \mathbb{Z}} f[p]\, h[n-p]$$

Pour des signaux finis $\tilde{f}, \tilde{h} \in \mathbb{R}^N$, on périodise :

$$f[n] = \tilde{f}[n \bmod N], \qquad h[n] = \tilde{h}[n \bmod N]$$

On définit la **convolution circulaire** :

$$\boxed{(f \circledast h)[n] = \sum_{p=0}^{N-1} f[n]\, h[n-p].}$$

Vecteur propres de la convolution circulaire

Soit L un operateur de convolution circulaire :

$$Lf[n] = (f \circledast h)[n]$$

Les exponentielles complexes discrètes

$$e_k[n] = \exp\left(2i\pi k \frac{n}{N}\right).$$

sont les vecteurs propres de la convolution circulaire puisque :

$$\begin{aligned}(L\, e_k)[n] &= \exp\left(2i\pi k \frac{n}{N}\right) \sum_{p=0}^{N-1} h[p] \exp\left(-2i\pi k \frac{p}{N}\right) \\ &= \widehat{h}[k]\, e_k[n] \quad \text{pour} \quad \widehat{h}[k] = \sum_{p=0}^{N-1} h[p] \exp\left(-2i\pi k \frac{p}{N}\right).\end{aligned}$$

Discrete Fourier Transform

On note $\langle \cdot, \cdot \rangle$ le produit sequilinéaire à droite sur $\mathbb{C}^n$:

$$\langle f, g \rangle = \sum_{n=0}^{N-1} f[n]\overline{g[n]}.$$

Les $(e_k)_{k\in\{0,\dots,N-1\}}$ forment une base orthogonale de $\mathbb{C}^n$.
Par conséquent

$$f = \sum_{k=0}^{N-1} \frac{\langle f, e_k \rangle}{\|e_k\|_2^2} e_k.$$

Par définition

$$\widehat{f}[k] = \langle f, e_k \rangle = \sum_{n=0}^{N-1} f[n]\overline{e_k[n]} = \sum_{n=0}^{N-1} f[n] \exp\left(-2i\pi k \frac{n}{N}\right).$$

On a $\|e_k\|_2^2 = N$, ce qui donne la formule de reconstruction :

$$f[n] = \frac{1}{N} \sum_{k=0}^{N-1} \widehat{f}[k] \exp\left(2i\pi k \frac{n}{N}\right)$$

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{i2\pi k \frac{n}{N}}$$

To find the energy at a particular frequency, your signal around a circle at that frequency, average a bunch of points along that path.

Basics

https://github.com/sduprey/timeseries_ressources/blob/main/Box-Cox.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Price simulation.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Black_Scholes_option_pricing_through_Monte_Carlo_simulation.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Stationarity.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ACF_PACF.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ACF_PACF_Stocks.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Granger_Causality.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Naive Forecast.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/WalkForward.ipynb

Exponential models

https://github.com/sduprey/timeseries_ressources/blob/main/SMA.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/EMWA_HoltWinters.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ETS_CHAMPAGNE.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ETS_Stock.ipynb

ARIMA

https://github.com/sduprey/timeseries_ressources/blob/main/ARIMA.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ARIMA_Sales_forecast.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ARIMA_stock.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Auto_ARIMA.ipynb

Prophet

https://github.com/sduprey/timeseries_ressources/blob/main/Prophet_Airline.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Prophet_Rossman.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/Prophet_stockprice.ipynb

GARCH

https://github.com/sduprey/timeseries_ressources/blob/main/GARCH.ipynb

VARMA

https://github.com/sduprey/timeseries_ressources/blob/main/VARMA.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/VARMA_US_ECO.ipynb

Machine Learning (SVM-RF-Logistic regression)

https://github.com/sduprey/timeseries_ressources/blob/main/Extrapolation.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ML_Airline_No_Differencing.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ML_Airline_with_differencing.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ML_Champagne.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ML_Stocks_regression.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/ML_Stocks_Classification.ipynb

Deep learning models

https://github.com/sduprey/timeseries_ressources/blob/main/ANN_Airline.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/CNN_Airline.ipynb

https://github.com/sduprey/timeseries_ressources/blob/main/RNN_Airline.ipynb

Project proposition

- Advanced regression theory and statistical test (T-test, F-test, R squared)
- Rewriting the AI time series notebook in PyTorch
- Price a barrier knocking call using Monte-Carlo simulation of a stochastic process
- Apply Prophet to the airline passengers dataset
- Deep dive in the maths of ARMA process
- Spectral analysis
- Cointegration and stationarity statistical tests
- VEC for multivariate time series modeling
- Pattern recognition/signal filtering (outliers detection/regime change detection)