Contents

1. Time series decomposition: TSD, STL

Typical TSD (time series decomposition) looks like:

$$y_t = T_t + S_t + R_t$$

T_t — trend, S_t — seasonality component, R_t — random fluctuations, a.k.a. noise.

The decomposition can also take on the following forms:

$$y_t = T_t S_t R_t, \text{ or } \ y_t = (T_t + S_t) R_t.$$

1. 1. Classical TSD (using moving averages)

Moving average (MA) is given by the following expression:

$$\mathrm{MA}(y_t; m) = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j},$$

where $m = 2k + 1$ is called *window size* and has to be odd. Backward formula:

$$\mathrm{MA}(y_t; m) = \frac{1}{m} \sum_{j=-m}^{0} y_{t+j},$$

Forward formula:

$$\mathrm{MA}(y_t; m) = \frac{1}{m} \sum_{j=0}^{m} y_{t+j}.$$

For $m = 4$:

$$\mathrm{MA}(y_t; 4) = \frac{1}{4}(y_{t-1}, y_t, y_{t+1}, y_{t+2}).$$

Moving average over moving average:

$$\begin{aligned}
\mathrm{MA}(\mathrm{MA}(y_t, 4); 2) &= \frac{1}{2}[\mathrm{MA}(y_{t-1}; 4), \mathrm{MA}(y_t; 4)] = \\
&= \frac{1}{2}\left[\frac{1}{4}(y_{t-2}, y_{t-1}, y_t, y_{t+1}) + \frac{1}{4}(y_{t-1}, y_t, y_{t+1}, y_{t+2})\right] = \\
&= \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}.
\end{aligned}$$

MAs are used to: 1) smooth out the data; 2) extranct the trend.

Weighted moving average (WMA):

$$\mathrm{WMA}(y_t; m) = \sum_{j=-k}^{k} y_{t+j} \cdot w_j,\ \ w_j \geq 0,\ \ \sum w_j = 1.$$

The classical TSD algorithm is given as follows:

1. Compute trend component using $2 \times m$-MA if m is even and m-MA if it is odd.

$$\hat{T}_t = \begin{cases} \mathrm{MA}(y_t; m), \text{ if } m \text{ is odd,} \\ \mathrm{MA}(\mathrm{MA}(y_t; m); 2), \text{ if } m \text{ is even.} \end{cases}$$

2. Detrend the time series (TS):

$$y_t - \hat{T}_t = S_t + R_t.$$

3. Compute $\hat{S}_t$ by averaging detrended TS for a season (assuming that S_t does not change from season to season).
4. $\hat{R}_t = y_t - \hat{S}_t - \hat{T}_t$.

Note: TSD assumes that S_t is constant throughout the seasons and that the trend line itself is not sensitive to sharp fluctuations.

1. 2. STL decomposition

An alternative to classical TSD would be *STL decomposition* (Seasonal Trend decomposition via LOESS). Here LOESS (locally estimated scatterplot smoothing) is type of local regression for modeling and smoothing data $(x_i, y_i)_{i=1}^{m}$. Its key components are:

1. Kernel function. For example, Gaussian kernel

$$w_i = \exp\left(-\frac{(x_i - x)^2}{2\tau^2}\right).$$

2. Smoothing parameter τ. Smaller τ leads to narrower windows and more flexible models, larger τ — to wider windows and less flexible models and $\tau \to +\infty$ means that $w_i = 1$, hence model becomes a simple linear regression.

Given data $(x_i, y_i)_{t=1}^{m}$ or $(t, y_t)_{t=1}^{T}$, the LOESS algorithm step-by-step:

1. Choose a kernel function $\mathcal{F}$ and set smoothing parameter τ.

2. For each x in dataset:

   2.1. Calculate $w_i = \mathcal{F}(x_i, x, \tau)$

   2.2. Build weighted regression model. For example, weighted least squares:

$$L = \sum_{i=1}^{n} w_i (y_i - \Theta^T x_i)^2,$$

   where $\Theta = (X^T W X)^{-1} X^T W y$.

   2.3. Make predictions $\hat{y}(x)$ for x only.

   2.4. “Forget” the model.

1. 2. 1. STL algorithm

Input: $Y = \{y_1, ..., y_\tau\}$.

Parameters: n_p — # of outer iterations (1-2)

n_i — # of inner iterations (1-2)

n_l — trend smoothing parameter (smoothing parameter for LOESS)

n_s — seasonality smoothing parameter

n_o — residual smoothing parameter (optional, for residues R_t).

0. Outer loop: repeat the following steps n_p times.
1. Initialization:

   1.1. set trend $T^{(0)} = 0$ or other initial approximation (MA for example);

   1.2. set weights $w = \{1, 1, ...1\}$ (optional, for residues).

2. Inner loop: repeat n_i times

   2.1. Detrend time series: $D = Y - T$.

   2.2. Compute seasonal component:

      2.2.1. Split D subseries by seasons;

      2.2.2. For each subseries apply the LOESS smoothing with $\tau = n_s$ and weights w.

      2.2.3. Assemble the smoothed subseries into a seasonal component C.

2.2.4. Center the seasonal component C by subtracting moving average.

2.3. Update seasonal component $S = C$.

2.4. Deaseasonalize the data: $\tilde{Y} = Y - S$

2.5. Update the trend: apply LOESS for $\tilde{Y}$ with $\tau = n_l$ and "robust" weights w (obtain T).

3. Compute the residuals $R = Y - T - S$.
4. Update weights: recompute weights w based on residues R to reduce the influence of outliers usually using Tuikey's biweight function.

Post-processing:

1. Normalize seasonality: mean value of S for each season should be zero.
2. Smoothen the trend if needed.

Result: trend T, seasonality S, residual noise R

Pros:

- *flexiblity:* it is robust to outliers;
- *robustness:* it can model non-linear trends;
- *arbitrary period:* it can work with any seasonality.

1. 2. 2. Tuikey's biweight function

Tuikey's biweight function is used to update the weights w using the following algorithm:

1. Obtain the residuals $R = Y - S - T$
2. Compute MAD (median absolute deviation):

$$\text{MAD} = \text{median}(|r_i - \text{median}(R)|).$$

Normalize: $S \approx 1.4826 \cdot \text{MAD}$, since $\sigma = 1.4826$

3. Compute the normalized residuals:

$$u_i = \frac{r_i}{C \cdot S},$$

where C is a tuning constant ($C = 4.685$).

4. Bisquare function

$$w_i = \begin{cases} (1 - u_i)^2, \ |u_i| < 1, \\ 0, \ |u_i| \geq 1. \end{cases}$$

5. If $S = 0$, then $w_i = 0$ (all residuals are the same). If $\text{MAD} = 0$, but the residuals are not the same, we use standard deviation instead of MAD.

For example, if $R = [0.1, -0.2, 3.0, -0.1, 10.0]$:

1. $\text{median}(R) = 0.1$, hence $\text{MAD} = \text{median}(|R - 0.1|) = 0.3$
2. $S = 0.3 \cdot 1.4826 \approx 0.4448$
3. $C = 4.685 \Rightarrow C \cdot S = 2.083$
4. $r_3 = 3.0 : |u_3| = |\frac{3.0}{2.083}| \approx 1.44 > 1 \Rightarrow w_3 = 0$
5. $r_5 = 10.0 : |u_5| = 4.801 > 1 \Rightarrow w_5 = 0$
6. $r_1 = 0.1 : |u_1| \approx 0.04821 \Rightarrow w_1 = \left(1 - 0.048^2\right)^2 \approx 0.995$

2. Weak, strong stationarity. Stationarity tests: DF, ADF, KPSS. Reduction to stationary time series

2. 1. Stationarity and Ergoticity

Stationarity is a key feature of time series. There are several kinds of stationarity:

- *Strict stationarity:* joint distribution of any segment of time series $\left(y_{t_1}, y_{t_2}, \ldots, y_{t_k}\right)$ is equivalent to $\left(y_{t_1+\tau}, y_{t_2+\tau}, \ldots, y_{t_k+\tau}\right)$ $\forall \tau$.
- *Weak stationarity:*

  1. $\forall t\ \mathbb{E}[y_t] = \mu$,

  2. $\forall t\ \mathbb{D}[y_t] = \sigma^2 < +\infty$,

  3. $\forall t, s, \tau\ \ \operatorname{cov}(y_t, y_s) = \operatorname{cov}(y_{t+\tau}, y_{s+\tau}) = \gamma(|t-s|)$. Here $\gamma(\cdot)$ is a function that depends on distance between points.

2. 1. 1. Non-stationary time series examples

1. Time servies with determinitstic trend:

$$y_t = \alpha + \beta t + \varepsilon_t,\ \varepsilon_t \sim \mathcal{N}(0, \sigma^2).$$

Here, $\mathbb{E}[y_t] = \alpha + \beta t$ which is not a constant value.

2. $y_t = \sin t + \varepsilon_t,\ \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$. Here

$$\mathbb{E}[y_t] = \begin{cases} 1,\ t = \frac{\pi}{2} + 2\pi k \\ -1,\ t = -\frac{\pi}{2} + 2\pi k \end{cases}$$

and since it depends on t the TS is non-stationary.

3. Random Walk: $y_t = y_{t-1} + \varepsilon_t,\ \varepsilon_t \sim \mathcal{N}(0, \sigma^2),\ \ \operatorname{cov}(\varepsilon_t, \varepsilon_s) = 0,\ t \neq s$. Let us write out values of this TS:

$$\begin{aligned} y_1 &= y_0 + \varepsilon_1, \\ y_2 &= y_1 + \varepsilon_2 = y_0 + \varepsilon_1 + \varepsilon_2, \\ &\ldots \\ y_t &= y_0 + \sum_{i=1}^{t} \varepsilon_i \end{aligned}$$

Therefore, $\mathbb{E}[y_t] = y_0,\ \mathbb{D}[y_t] = t\sigma^2$.

2. 1. 2. Stationary time series examples

1. $y_t = \varepsilon_t, \varepsilon_t \underset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$ – white noise. In this case,

$$\forall t, s : t \neq s,\ \mathbb{E}[y_t] = 0,\ \mathbb{D}[y_t] = \sigma^2 < \infty \to \text{stationary}$$

2. $y_t = \beta_1 y_{t-1} + \varepsilon_t,\ \beta \in (-1, 1),\ \varepsilon_t \underset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$

$$\begin{aligned} y_t = \beta_1 y_{t-1} + \varepsilon_t &= \beta_1(\beta_1 y_{t-2} + \varepsilon_{t-1}) + \varepsilon_t = \\ &= \beta_1^t y_0 + \sum_{i=1}^{t} \beta_1^{t-i} \varepsilon_i. \end{aligned}$$

Here, since ε_i are independant from each other:

$$\begin{aligned} \mathbb{E}[y_t] &= \mathbb{E}\left[\beta_1^t y_0 + \sum_{i=1}^{t} \beta_1^{t-i} \varepsilon_i\right] = \beta_1^t y_0 + \sum_{i=1}^{t} \beta_1^{t-i} \mathbb{E}[\varepsilon_i] = \\ &= \beta_1^t y_0 \ \text{ if } \ t \to \infty,\ \beta_1^t \to 0. \\ \mathbb{D}[y_t] &= \mathbb{D}\left[\beta_1^t y_0 + \sum_{i=1}^{t} \beta_1^{t-i} \varepsilon_i\right] = \sum_{i=1}^{t} \beta_1^{2(t-i)} \mathbb{D}[\varepsilon_i] = \\ &= \left(\beta_1^{2t-2} + \beta_1^{2t-4} + \dots + 1\right) \cdot \sigma^2 \\ \operatorname{cov}(y_t,\ y_{t+1}) &= \operatorname{cov}\Big(\beta_1 y_{t-1} + \varepsilon_t,\ \beta_1 y_t + \varepsilon_{t_1}\Big) \\ &= \operatorname{cov}\left(\beta_1^t y_0 + \sum_{i=1}^{t} \beta_1^{t-i} \varepsilon_i,\ \beta_1^{t+1} y_0 + \sum_{i=1}^{t+1} \beta_1^{t+1-i} \varepsilon_i\right) = \\ &= \beta_1 \operatorname{cov}(\varepsilon_t,\ \varepsilon_t) + \beta_1^3 \operatorname{cov}(\varepsilon_{t-1},\ \varepsilon_{t-1}) + \dots + \beta_1^{2t-1} \operatorname{cov}(\varepsilon_1,\ \varepsilon_1) = \\ &= \sum_{i=1}^{t} \beta_1^{2i-1} \mathbb{D}[\varepsilon_{t+1-i}] \to \frac{\beta_1}{1 - \beta_1^2} \cdot \sigma^2 = \text{const}. \end{aligned}$$

A random stochastic process is called *ergodic* if its statistical properties can be estimated using a sample from it.

Note: any ergodic process is stationary and almost any stationary process is ergodic.

2. 2. Stationarity tests

2. 2. 1. Unit root

Time series with unit root do not have a constant average level and have stochastic trends.

Let us consider a simple model: $y_t = \varphi \cdot y_{t-1} + \varepsilon_t,\ \varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2),\ \varphi$ is constant.

1. $|\varphi| < 1$ means that the process is stationary;
2. $|\varphi| > 1$ is a non-stationary or explosive time series;
3. $|\varphi| = 1$ is the unit root case, not stationary, since:

$$y_t = y_{t-1} + \varepsilon_t = y_0 + \sum_{i=1}^{n} \varepsilon_i \Rightarrow \mathbb{D}[y_t] = t\sigma^2.$$

Why unit root?

Let us define a lag operator $Ly_t = y_{t-1}$. Then, $y_t = \varphi y_{t-1} + \varepsilon_t$ can be rewritten as $y_t = \varphi L y_t + \varepsilon_t$ hence $y_t(1 - \varphi L) = \varepsilon_t$.

Taking this into account, the characteristic equation would be

$$(1 - \varphi z) = 0 \Rightarrow z = \frac{1}{\varphi}$$

and if $\varphi = 1$ then $z = 1$ and $y_t = y_{t-1} + \varepsilon_t$.

2. 2. 2. Dickey-Fuller test (unit root test)

1. Consider a time series $y_t = \varphi y_{t-1} + \varepsilon_t$. Let $\Delta y_t = y_t - y_{t-1}$, then:

$$\Delta y_t = (\varphi - 1) y_{t-1} + \varepsilon_t = \gamma y_{t-1} + \varepsilon_t.$$

2. Formulate the hypotheses:

$$H_0 : \gamma = 0\ (\varphi = 1) \Rightarrow \text{unit root} \Rightarrow \text{non-stationary time series.}$$
$$H_1 : \gamma < 0\ (\varphi < 1) \Rightarrow \text{no unit root} \Rightarrow \text{stationary time series.}$$

3. Evaluate γ by fitting regression: $\Delta y_t = \gamma y_{t-1} + \varepsilon_t$. Estimate standard t-statistic for γ:

$$t_{\text{stat}} = \frac{\hat{\gamma}}{\text{SE}(\hat{\gamma})}$$

4. Dickey-Fuller distribution.

Significance level	**Critical value**
1%	-3.43
5%	-2.86
10%	-2.57

5. If $t_{\text{stat}} <$ crit. val. $\to H_0$ is rejected,

If $t_{\text{stat}} \geq$ crit. val. $\to H_0$ is not rejected.

2. 2. 3. Modification of DF test

Basic regression is very simple model. Instead, it is often epxanded:

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \varepsilon_t.$$

This model is able to perform stationarity checks around deterministic trends.

2. 2. 4. Augmented Dickey-Fuller test

DF test assumes that ε_t are not correlated. This issue can be solved by adding lagged differences to the regression. Those lagged differences will reduce auto-correlation in error terms ε_t.

$$\Delta y_t = \alpha + \beta t + \gamma y_{t-1} + \sum_{i=1}^{p} \delta_i \Delta y_{t-i}$$

How does the choice of p impact the model:

- if p is too small, then the correlation issue will not be solved,
- if p is too big, the power of test decreases.

How to choose p:

1. $p \approx \sqrt[3]{T},\ p \approx \sqrt{T}$.
2. Test different p, choose one that gives you the "best" regression: BIC, AIC, MQIC.

Interpretation of ADF is exactly the same.

2. 2. 5. KPSS (Kwiatkowski-Phillips-Schmidt-Shin) test

1. KPSS assumes that the time series can be decomposed into the following sum:

$$y_t = \xi_t + r_t + \varepsilon_t,$$

where:

- ξ_t is deterministic trend,
- r_t is stochastic trend such that $\mathbb{D}[r_t] = \sigma_r^2$,
- ε_t – white noise.

2. H_0: time series is stationary $\Rightarrow \sigma_r^2 = 0 \Rightarrow y_t = \xi_t + \varepsilon_t$,

H_1: time series is not stationary $\Rightarrow \sigma_r^2 > 0 \Rightarrow r_t \neq 0$.

3. Fit regression:

   3.1. $y_t = \alpha + \beta t + \varepsilon_t \Rightarrow$ residuals $e_t = y_t - \hat{\alpha} - \hat{\beta} t$.

   3.2. Accumulation of residuals $S_t = \sum_{i=1}^{t} e_i$.

   3.3. Calculate KPSS value:

$$\text{KPSS} = \sum_{i=1}^{T} \frac{S_t^2}{T^2 \sigma_\varepsilon^2},$$

   where σ_ε^2 is the variance of ε_t estimated using Newey-West method.

4. Decision logic: if KPSS $<$ crit. value, reject H_0. Otherwise, H_0 is not rejected.

3. Filtration problem. Deterministic methods of filtration: MA, SMA, EMA, polynomial smoothing

3. 1. Main methods of reduction to stationary time series (may be redundant)

There are two types of non-stationarity:

- Trend;
- Nonconsistent dispersion.

If there is a trend, we can use the following methods to standardize the time series:

1. Taking difference of time series:

$$y_i \to \Delta y_i,\ \Delta y_i = y_i - y_{i-1},\ i = 2, ..., \tau.$$

2. Substracting the trend component:

   2.1. TSD $\to$ Trend $\to$ y_i − Trend;

   2.2. Polynomial regression.

2.* Lagged difference:

$$y_i \to \Delta_k y_i,\ \Delta_k y_i = y_i - y_{i-k}$$

and adjust k for seasonality.

2.** Subtracting the seasonal component:

$$\text{TSD} \Rightarrow \text{Season} \Rightarrow y_i - \text{Season}$$

3. 1. 1. Dispersion stabilization

Box-cox transformation. Given $y = \{y_1, ..., y_\tau\},\ y_i > 0$:

$$\tilde{y}_i = \begin{cases} \frac{y_i^\lambda - 1}{\lambda},\ \lambda \neq 0, \\ \log y_i,\ \lambda = 0. \end{cases}$$

Note: if $\lambda > 1$ the inverse transform is taken, otherwise:

$$\lambda = \begin{cases} 1 \Rightarrow \text{no transformation;} \\ 0.5 \Rightarrow \text{square root, i.e. softer than log;} \\ 0 \Rightarrow \text{natural} \quad \log . \end{cases}$$

The λ value is chosen using a maximum likelyhood function by applying Box-Cox for different λ values and choosing which maximizes the likelyhood of transformed data following a normal distribution.

Normal distribution likelyhood function:

$$L = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z_i - \mu)^2}{2\sigma^2}\right).$$

Substituting $z_i = \tilde{y}_i = \text{Box-Cox}(y_i, \lambda)$ we get:

$$L = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(\tilde{y}_i - \mu)^2}{2\sigma^2}\right) \times \prod_{i=1}^{n} y_i^{\lambda-1}$$

$$\log L = -\frac{n}{2}\log\pi - \frac{n}{2}\log\sigma^2 - \sum_{i=1}^{n}\log\left(-\frac{(\tilde{y}_i - \mu)^2}{2\sigma^2}\right) +$$

$$+(\lambda - 1)\sum_{i=1}^{n}\log y_i .$$

Here, the term

$$\prod_{i=1}^{n} y_i^{\lambda-1}$$

is derivative of Jacobian matrix of Box-Cox trnasform. Note that Box-Cox works only for positive y_i, hence if $y_i \leq 0$, the data is shifted by $\alpha : y_i + \alpha > 0,\ i = 1, ...\tau$ and the transform itself is applied after that.

When to apply Box-Cox:

1. Graphical test: plot variance against mean. Use Box-Cox if there is a clear dependance.
2. Distribution is asymmetric.

3. 2. Data filtration and smooting

Data filtration is **not** smooting. Rather smoothing is a tool used in data filtration. Filtration is time series transformation aimed at highlinghting, analyzing or supressing certain characterstics of time series such as noise or artifacts.

Goals of filtration:

- Trend extraction;
- Noise supression;
- Artifact removal;
- Time series decomposition.

A problem that may arise during filtration is finding a compromise between precision and smoothing.

3. 2. 1. Deterministic methods of filtration

1. SMA (moving average):

$$\mathrm{SMA}(y_t, m) = \frac{y_{t-m} + y_{t-m+1} + \dots + y_{t+m}}{2m+1}.$$

Here the issue arises from last m observations missing, hence in most scenarios the formula will look like this:

$$\mathrm{SMA}\ (y_t, m) = \frac{y_{t-m} + y_{t-m+1} + \dots + y_t}{m+1}.$$

2. WMA (weighted moving average):

$$\mathrm{WMA} = \frac{\sum w_i y_i}{\sum w_i}.$$

3. EMA (exponential moving average).

The idea of this method is to construct a recurrent formula so that the weights of previous points would decrease exponentially. It is given by the following expression:

$$\mathrm{EMA}(y_t) = \alpha y_t + (1-\alpha)\ \mathrm{EMA}(y_{t-1}).$$

Here $\alpha \in (0,\ 1)$ is a smooting parameter.

4. Polynomial (Savitzky-Golay) filter.

Given data points, choose a window of size $n = 2m + 1$ and fit a polynomial line of a low degree then choose its value at i as TS value at i. Algorithm step-by-step (at a fixed point t):

1. Choose the window of size $n = 2m + 1$ and degree of polynomial k.

2. Fit a polinomial $P(i) = \alpha_0 + \alpha_1 i + \alpha_2 i^2 + ... + \alpha_k i^k,\ i = -m, ..., m.$

   In matrix multiplication from:

$$X\alpha \approx y.$$

   Here,

$$X = \begin{pmatrix} 1 & -m & (-m)^2 & ... & (-m)^k \\ 1 & -m+1 & (-m+1)^2 & ... & (-m+1)^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & m & m^2 & ... & m^k \end{pmatrix}.$$

3. Coefficients α_j can be obtained using least squares minimization:

$$\sum_{i=-m}^{m} \left(P(i) - y_{t-i}\right)^2 \to \min_{\alpha_j}$$

   its solution being $\hat{\alpha} = \left(X^T X\right)^{-1} X^T y$.

4. $P(0) = \hat{\alpha}_0 \to$ smoothed value for current y_t. Since

$$\hat{\alpha}_0 = c_0^T \hat{\alpha} = c_0^T \left(X^T X\right)^{-1} X^T y,\ c_0 = [1, 0, ..., 0]^T,$$

   designating $C = X\left(X^T X\right)^{-1} c_0$, the resulting expression is

$$\hat{\alpha}_0 = C^T y = c_{-m} y_{t-m} + ... + c_m y_{t+m}.$$

Downside. Polynomials have to be fitted for each point.

How to deal with corner points:

1. Asymmetric window
2. Use polynomials calculated for the first and last full window.

4. Filtration and smoothing using Fourier analysis: Fourier series, Fourier transform, DFT, FFT, SSFT

4. 1. Fourier transform

Fourier series is a decomposition of a function $f \in C[a, b]$ with a orthogonal function system $\{g_k\}_{k=0}^{+\infty}$ in some euclidean space:

$$f(x) = \sum_{k=1}^{+\infty} c_k g_{k(x)},\ (f, g_k) = \int_a^b f(x) g_k(x) dx = 0$$

If g_k is a trigonometric system:

$$g_k \in \left\{\frac{1}{2l}, \frac{1}{\sqrt{l}} \cos\left(\frac{\pi x}{l}\right), \frac{1}{\sqrt{l}} \sin\left(\frac{\pi x}{l}\right), ...\right\}$$

Then $f(x)$:

$$\begin{aligned} f(x) &= \frac{a_0}{2} + \sum_{k=1}^{+\infty} \left[a_k \cos\left(\frac{k\pi x}{l}\right) + b_k \sin\left(\frac{k\pi x}{l}\right)\right], \\ a_k &= \frac{1}{l} \int_{-l}^{l} f(x) \cos\left(\frac{k\pi x}{l}\right) dx,\ a_{-k} = a_k, \\ b_k &= \frac{1}{l} \int_{-l}^{l} f(x) \sin\left(\frac{k\pi x}{l}\right) dx,\ b_0 = 0,\ b_{-k} = -b_k. \end{aligned}$$

In a more general case:

$$f(x) = \sum_{k=-\infty}^{+\infty} c_k e^{i w_k x},\ w_k = \frac{\pi k}{l},\ c_k = \frac{1}{2l} \int_{-l}^{l} f(x) e^{-i w_k x} dx$$

Let us derive this statement. Since

$$\sin(kx) = \frac{e^{ikx} - e^{-ikx}}{2i} \ \text{ and } \ \cos(kx) = \frac{e^{ikx} + e^{-ikx}}{2},$$

$f(x)$ can expressed in a following manner:

$$f(x) = e^{iw_0 x} \cdot \frac{a_0}{2} + \sum_{k=1}^{+\infty} \left[a_k \frac{e^{iw_k x} + e^{-iw_k x}}{2} + b_k \frac{e^{iw_k x} - e^{-iw_k x}}{2i} \right] =$$

$$= \frac{a_0}{2} e^{iw_0 x} + \frac{1}{2} \sum_{k=1}^{+\infty} \left[a_k e^{iw_k x} + a_k e^{-iw_k x} - ib_k e^{iw_k x} + ib_k e^{-iw_k x} \right] =$$

$$= \frac{a_0}{2} e^{iw_0 x} + \frac{1}{2} \sum_{k=1}^{+\infty} (a_k - ib_k) e^{iw_k x} + \frac{1}{2} \sum_{k=1}^{+\infty} (a_k + ib_k) e^{-iw_k x} =$$

$$= \sum_{k=-\infty}^{+\infty} c_k e^{iw_k x}.$$

Then, since $a_{-k} = a_k$ and $b_{-k} = -b_k$,

$$c_k = \frac{1}{2}(a_k - ib_k) = \frac{1}{2l} \int_{-l}^{l} f(t) \left(\cos\left(\frac{k\pi t}{l} \right) - i \sin\left(\frac{k\pi t}{l} \right) \right) dt =$$

$$= \frac{1}{2l} \int_{-l}^{l} f(t) \left(\frac{e^{iw_k t} + e^{-iw_k t}}{2} - i \frac{e^{iw_k t} - e^{-iw_k t}}{2i} \right) dt =$$

$$= \frac{1}{4l} \int_{-l}^{l} f(t) \cdot 2e^{-iw_k t} dt = \frac{1}{2l} \int_{-l}^{l} f(t) e^{-iw_k t} dt.$$

4. 2. From Fourier series to Fourier transform

For $t \in [-l, l]$:

$$f(t) = \sum_{k=-\infty}^{+\infty} c_k e^{iw_k t}.$$

However, for $l \to +\infty$ following assumptions should be made:

1. $f(t)$ is piecewise continuous and has one-sided derivative in $[-l, l]$.
2. Limit function $f(t) = \lim_{l \to +\infty} \sum_{k=-\infty}^{+\infty} c_k e^{iw_k t}$ is absolutely integrable.
3. Limit function $f(t)$ is piecewise continuous and has one-sided derivatives at any point.

Let us define $\Delta w_k = w_{k+1} - w_k,\ k \in \mathbb{Z}$. Since $w_k = \frac{\pi k}{l}$, $\Delta w_k = \frac{\pi}{l}$ and $\frac{1}{l} = \frac{\Delta w_k}{\pi}$. Therefore, $f(t)$ can be represented as:

$$\begin{aligned}f(t) &= \sum_{k=-\infty}^{+\infty} \frac{1}{2l} \int_{-l}^{l} f(\tau) e^{-iw_k\tau} d\tau \cdot e^{iw_k t} = \\ &= \sum_{k=-\infty}^{+\infty} \frac{1}{2\pi} \int_{-l}^{l} f(\tau) e^{-iw_k\tau} d\tau \cdot e^{iw_k t} \Delta w_k = \\ &= \frac{1}{2\pi} \sum_{k=-\infty}^{+\infty} \hat{F}_l(w_k, t) \Delta w_k.\end{aligned}$$

And if $l \to +\infty$, then $\Delta w_k \to 0$ and

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{F}_l(w, t) dw,$$

where $\hat{F}_l(w, t) = \int_{-\infty}^{+\infty} f(\tau) e^{-iw\tau} d\tau \cdot e^{iwt}$.

Then, **Fourier transform** can be defined as:

$$\hat{f}(w) = \mathcal{F}(f(t)) = \int_{-\infty}^{+\infty} f(t) e^{-iwt} dt.$$

And **inverse Fourier transform** would be:

$$f(t) = \mathcal{F}^{-1}\left(\hat{f}(w)\right) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{f}(w) e^{iwt} dw.$$

Properties of Fourier transform:

1. Linearity.
2. $\mathcal{F}(f * g) = \hat{f}(w) \cdot \hat{g}(w)$, where $f * g = \int_{-\infty}^{+\infty} f(\tau) g(t - \tau) d\tau$ i.e. convolution.
3. $\mathcal{F}(f \cdot g) = \hat{f}(w) + \hat{g}(w)$.

4. 3. Discrete Fourier transform

DFT (discrete Fourier transform) is an operation that transforms $f(t)$ to $f_0, f_1, ..., f_n$. Direct and inverse DFT respectively:

$$\hat{f}_k = \sum_{j=0}^{n-1} f_j \exp\left(-i\frac{2\pi jk}{n}\right),$$

$$f_k = \sum_{j=0}^{n-1} \hat{f}_j \exp\left(i\frac{2\pi jk}{n}\right).$$

It has algorithmic complexity of $\mathcal{O}(n^2)$ and is essentially a matrix multiplication:

$$\begin{pmatrix} \hat{f}_0 \\ \vdots \\ \vdots \\ \vdots \\ \hat{f}_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & w_n & w_n^2 & \dots & w_n^{n-1} \\ 1 & w_n^2 & w_n^4 & \dots & w_n^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w_n^{n-1} & w_n^{2(n-1)} & \dots & w_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} f_0 \\ \vdots \\ \vdots \\ \vdots \\ f_{n-1} \end{pmatrix}$$

where $w_n = \exp\left(-\frac{2\pi i}{n}\right)$.

4. 4. Fast Fourier transform

FFT (fast Fourier transform) is a family of algorithms that arose from need for a… faster version of DFT. Let us consider Cooley-Tukey algorithm. It relies on two properties of DFT:

- $w_n^{jk} = \exp\left(-i\frac{2\pi jk}{n}\right)$ is periodic: $w_n^{jk} = w_n^{j(k+n)} = w_n^{k(j+n)}$.
- w_n^{jk} is symmetric: $w_n^{k+\frac{n}{2}} = -w_n^k$.

The algorithm step-by-step:

1. Split f into even and odd terms: $f_{\text{even}} = \{f_{2k}\}_{k=0}^{\frac{n}{2}-1}$ and $f_{\text{odd}} = \{f_{2k+1}\}_{k=0}^{\frac{n}{2}-1}$
2. Let $G(k) = \text{DFT}(f_{\text{even}})$ and $H(k) = \text{DFT}(f_{\text{odd}})$ which takes $\mathcal{O}\left(\frac{n^2}{4}\right)$ operations each and $\mathcal{O}\left(\frac{n^2}{2}\right)$ total.

Therefore,

$$\begin{aligned} \hat{f}_k &= \sum_{j=0}^{\frac{n}{2}-1} f_{2j} \exp\left(-i\frac{2\pi k(2j)}{n}\right) + \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1} \exp\left(-i\frac{2\pi k(2j+1)}{n}\right) = \\ &= \sum_{j=0}^{\frac{n}{2}-1} f_{2j} \exp\left(-i\frac{2\pi kj}{\frac{n}{2}}\right) + w_n^k \sum_{j=0}^{\frac{n}{2}-1} f_{2j+1} \exp\left(-i\frac{2\pi kj}{\frac{n}{2}}\right) = \\ &= G(k) + w_n^k H(k),\ k = 0, 1, \dots, \frac{n}{2} - 1. \end{aligned}$$

Taking the periodicity of w_n into account,

$$\hat{f}_{k+\frac{n}{2}} = G\Big(k + \frac{n}{2}\Big) + w_n^{k+\frac{n}{2}} H\Big(k + \frac{n}{2}\Big) = G(k) - w_n^k H(k)$$

which implies that $H(k + \frac{n}{2}) = H(k)$ and $G(k + \frac{n}{2}) = G(k)$.

3. Recursion. It can be used to calculate $H(k)$ and $G(k)$, moreover, when $n = 2^m$ recursion can be applied untill the end. Total number of recursions is $m = \log_2 n$, hence it is $\mathcal{O}(n \log_2 n)$.

Matrix form.

$$\hat{f} = F^{2^m} f = \begin{pmatrix} E^{2^{m-1}} & D^{2^{m-1}} \\ D^{2^{m-1}} & E^{2^{m-1}} \end{pmatrix} \begin{pmatrix} F^{2^{m-1}} & 0 \\ 0 & F^{2^{m-1}} \end{pmatrix} \begin{pmatrix} f_{\text{even}} \\ f_{\text{odd}} \end{pmatrix},$$

where E^n is $n \times n$ identity matrix and

$$D^n = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & w_n & 0 & \dots & 0 \\ 0 & 0 & w_n^2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \dots & w_n^{n-1} \end{pmatrix}.$$

4. 5. Short-time Fourier transform

Idea. Given a predefined window, drag it over the time series applying FFT locally each step. The values inside of the window would be weighted using a kernel function.

STFT (Gabor transform) is given by:

$$G(f)(t, w) = \hat{f}_g(t, w) = \int_{-\infty}^{+\infty} f(\tau) e^{-iw\tau} g(\tau - t) d\tau,$$

where $g(t)$ is a kernel function, e.g. Gaussian kernel function:

$$g(t) = \exp\left(-\frac{(t-\tau)^2}{\alpha^2}\right).$$

STFT can easily be discretized by applying FFT in each window. The result of STFT is a spectrogram: a plot of frequency against time.

5. Time series forecasting problem. Multi-step ahead forecasting: two main approaches

5. 1. One-step-ahead forecasting

Idea. Predict one next value of time series (usually denoted as $t + 1$) at a time using w previous observations.

Given time series $Y = \{y_1, ..., y_t\}$ and lookback window w, find f such that

$$\hat{y}_{t+1} = f(y_t, y_{t-1}, ..., y_{t-w+1}).$$

This task can be turned into an optimization problem by using a loss function, for example MAE, MSE, RMSE, MAPE or SMAPE.

5. 2. Multi-step-ahead forecasting

Idea. Predict multiple next values.

- **Recurrent approach.** Model learns how to make one-step-ahead forecasts adn then makes predictions multiple steps ahead by recursively applying one-step-ahead forecasting. This approach is easy to use and allows you to make predictions with any one-step-ahead model. However, since a model trained for one-step-ahead prediction is used recuresively, prediction errors add up the futher the prediction is.
- **Direct approach.** A separeate model is built for each step:

$$\begin{aligned} \hat{y}_{t+1} &= f_1(y_t, ..., y_{t-w+1}), \\ \hat{y}_{t+2} &= f_2(y_t, ..., y_{t-w+1}), \\ &\vdots \\ \hat{y}_{t+h} &= f_h(y_t, ..., y_{t-w+1}). \end{aligned}$$

Or in vector form:

$$\begin{pmatrix} \hat{y}_{t+h} \\ \hat{y}_{t+h-1} \\ \vdots \\ \hat{y}_{t+1} \end{pmatrix} = f(y_t, ..., y_{t-w+1}).$$

This approach does not have error accumulation issue, however it also does not use information form previous forecasts and is more computationally complex.

5. 3. Real-life approach

1. Determine the target variable.
2. Take into account exogenous factors like the number of woring days in a month for demand forecastiong.
3. Decide whether to use deterministic or probabilistic forecasting.

5. 4. Model testing

1. Train-test split
2. Train-test-val split, where val is for hyperparameter tuning
3. Cross-validation.

6. Exponential smoothing, Holt's linear model, ETS models

6. 1. Simple exponential smoothing

This model is given by

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

where $\hat{y}_{t+1 \mid t}$ is a forecast for y_{t+1} and α is smoothing parameter. Optimal α is a solution to the following optimization problem:

$$\sum_{i=0}^{T} (\hat{y}_i(\alpha) - y_i)^2 \to \min_{\alpha}.$$

It can be shown that the smoothed predicted value is equal to predicted value of x at $T + 1$ which solves the following problem:

$$\sum_{t=0}^{T} \beta^{T-1}(y_t - x)^2 \to \min_{x}.$$

Taking the derivative we get that:

$$\sum_{t=0}^{T} \beta^{T-t}(y_t - x) = 0,$$

it follows that:

$$\begin{aligned} x &= \frac{\sum_{t=0}^{T} \beta^{T-t} y_t}{\sum_{t=0}^{T} \beta^{T-t}} = \frac{\sum_{t=0}^{T} \beta^{T-t} y_t}{\frac{1}{1-\beta}} = \\ &= (1-\beta)\sum_{t=0}^{T} \beta^{T-t} y_t = (1-\beta)y_T + (1-\beta)\beta\sum_{t=0}^{T-1} \beta^{T-1-t} y_t = \\ &= (1-\beta)y_T + \beta\hat{y}_{T \mid T-1}. \end{aligned}$$

Designating $\alpha = 1 - \beta$ we get the initial expression.

6. 2. Holt's linear trend model

Consider additive trend model:

$$\hat{y}_{t+d \mid t'} = a_t + b_t \cdot d,$$

where level component a_t and slope of the linear trend b_t are given by

$$a_t = \alpha y_t + (1-\alpha)(a_{t-1} + b_t),$$
$$b_t = \beta(a_t - a_{t+1}) + (1-\beta)b_{t-1}.$$

Intuition. (very intuitive explanation ahead which is 100% easy to understand)

For

$$d = 0: \ \hat{y}_{t \mid t+1} = a_{t+1},$$
$$d = 1: \ \hat{y}_{t+1 \mid t} = a_t + b_t.$$

Note that since forecasts need to match, $a_{t+1} \approx a_t + b_t$, hence $a_{t+1} - a_t \approx b_t$.

Consider time series of differences $\Delta y_t = a_{t+1} - a_t$ and a problem of constant forecasting ($\Delta \hat{y}_{t| \ t-1} = b$) using exponential smoothing model:

$$\sum_{i=0}^{t} (1-\beta)^{t-i}(\Delta y_i - b)^2 \to \min_b .$$

Substituting the differences in definition of b_t gives the following result:

$$b_t = \beta \Delta y_t + (1-\beta)\Delta \hat{y}_{t \mid t-1}.$$

Since $a_{t+1} \approx a_t + b_t$, it follows that this is exponential smoothing for y_t where forecasted value is $a_t + b_t$:

$$a_t = \alpha y_t + (1-\alpha)\hat{y}_{t \mid t-1} = \alpha y_t + (1-\alpha)(a_{t-1} + b_{t-1}).$$

The resulting model is interpretable(bruh), but only for linear/exponential trends that lack a seasonal component. It is worth noting that this model is sensitive to outliers and assumes the trend to be constant.

6. 3. ETS models

ETS (error-trend-seasonality) models is a family of exponential smooting models which utilize time series decomposition and construct forecasts based on contributions of resulting components. Those components are:

- **Error.** Shows how random fluctuations affect the model. Error can be:
  1. A — additive (i.e. independant from time series level)
  2. M — multiplicative (i.e. scales with time series level).

- **Trend.** Designates the long-term direction of time series. There can be:
  1. N — no trend;
  2. A — additive trend;
  3. M — multiplicative trend;
  4. Ad, Md — additive / multiplicative dampened trend.
- **Seasonality.** Designates periodical fluctuations in time series. This component could be:
  1. N — no seasonality;
  2. A — linear seasonality;
  3. M — multiplicative seasonality.

Advantages:

1. Interpretable and intuitive;
2. Can be used for interval forecasting;
3. Stable and robust;
4. Fast.

Disadvantages:

1. Does not support inclusion of exogenous variables;
2. Very shaky assumptions which bar this family of models from many real-life scenarios.

7. Autocorrelation and partial autocorrelation. AR, MA, ARMA, ARIMA models.

7. 1. Autocorrelation and partial autocorrelation

ACF *(AutoCorrelation Function)* shows correlation of y_t with lagged component of time series y_{t-k} for different k's. It is given by the following expression:

$$\mathrm{ACF}(k) = \rho(y_t,\ y_{t-k}) = \frac{\operatorname{cov}(y_t,\ y_{t-k})}{\sigma(y_t)\sigma(y_{t-k})} \approx \frac{\sum_{\tau=k}^{T}(y_k - \overline{y})(y_{t-k} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})},$$

where $\overline{y} = \frac{1}{T}\sum_{t=1}^{T} y_t$ and $|\mathrm{ACF}(k)| \leq 1$.

ACF is used to identify:

1. **Trend.** Since trend is a long-term movement in a set direction, ACF will be positive and significant for long periods of time.
2. **Memory of the process.** Memory of the process is extent of the effect that previous values have on new observations. Therefore, the rate and nature of autocorrelation attenuation can signify the type of process: if it is fast, i.e. there are drops, the process has short memory; if attenuation is slow, i.e. the changes are exponential, the process has long memory.
3. **Seasonality.** Since seasonality is just oscillations at a fixed frequency, ACF plot will show spikes corresponding to seasonality period.

PACF *(Partial AutoCorrelation Function)* shows correlation between y_t and y_{t-k} but removes the effect of all intermediate lags $(y_{t-1}, y_{t-2}, \dots, y_{t-k+1})$.

$$\mathrm{PACF}(k) = \rho(y_t,\ y_{t-k} |\ y_{t-1}, \dots, y_{t-k+1}).$$

PACF is calculated by fitting a regression

$$y_t = \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \dots + \varphi_k y_{t-k} + \varepsilon_t$$

and then $\varphi_k = \mathrm{PACF}(k)$. Here the terms $\varphi_1, \dots, \varphi_{k-1}$ are responsible for removal of linear effect of intermediate lags.

7. 2. AR model description

Model $\mathrm{AR}(p)$ is given by a following expression

$$y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + \dots + \varphi_p y_{t-p} + \varepsilon_t,$$

where:

- y_t — value of TS @ time t;
- c — constant term to be determined;
- φ_t — parameters of the model;
- ε_t — model error term at time t, a.k.a. noise term.

Assumptions and limitations:

1. $\mathbb{E}(\varepsilon_t) = 0 \ \forall t$;
2. $\mathbb{D}(\varepsilon_t) = \text{const} = \sigma^2 \ \forall t$;
3. $\text{cov}(\varepsilon_t,\ \varepsilon_s) = 0 \ \forall t \neq s$;
4. $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$.

Key features:

1. Interpretability.

2. AR(p) can be applied only to stationary TS. This guarantiees that the influence of previous values fades over time, since otherwise the series will be explosive and model would not be suitable for forecasting.

3. ACF: decays over time.

4. PACF: breaks off after lag p, hence it can be used to find optimal p.

7. 2. 1. Training AR model

1. OLS

$$\sum_t \varepsilon_t^2 = \sum_t \left(y_t - c - \varphi_1 y_{t-1} - \ldots - \varphi_p y_{t-p}\right)^2 \to \min_{c, \varphi_i}$$

2. MLE

Given that $Y = (y_1, \ldots, y_n)$:

$$L(\theta \mid y) \approx \prod_{t=p+1}^{n} f\left(y_t \mid y_{t-1}, \ldots, y_{t-p}, \theta\right).$$

Considering that $y_t | y_{t-1}, \ldots, y_{t-p} \sim \mathcal{N}\left(c + \sum_{i=1}^{p} \varphi_i y_{t-i}, \sigma^2\right)$,

$$L(\theta \mid y) = \prod_{t=p+1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_t - c - \sum_{i=1}^{p} \varphi_i y_{t-i}}{\sigma}\right)^2\right) \to \max_{\theta},$$

where $\theta = \left\{c, \varphi_1, \ldots, \varphi_p\right\}$.

7. 3. MA model description

$$y_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \dots + \theta_q \varepsilon_q$$

where:

- θ_i — model parameters;
- ε_i — time series error term at time i;
- μ — configurable constant term.

This model has same assumptions as AR.

Key features:

1. Interpretability.
2. Always stationary.
3. ACF: breaks off after lag q, hence used to determine the optimal q value.
4. PACF: decays gradually.

7. 3. 1. Training MA model

Let us assume that $\varepsilon_i = 0,\ i = 0, \dots, q+1$. Then

1. Conditional LS. Denoting $\theta = \left\{\theta_1, \dots, \theta_q\right\}$ we get the following:

$$\sum_{t=1}^{n} \varepsilon_t^2 = \sum_t \left(y_t - \mu - \sum_{i=1}^{q} \varepsilon_{t-i} \theta_i \right) \to \min_{\mu, \theta}$$

2. MLE. Denoting $Y = (y_1, \dots, y_n)$, we get

$$L(\theta \mid y) \approx \prod_{t=1}^{n} f\left(y_t \mid \varepsilon_{t-1}, \dots, \varepsilon_{t-q}, \theta\right).$$

and since $y_t \mid \varepsilon_{t-1}, \dots, \varepsilon_{t-q}, \theta \sim \mathcal{N}\left(\mu + \sum_{i=1}^{q} \theta_i \varepsilon_i, \sigma^2\right)$, the optimization problem can be formulated in the following manner:

$$\log L(\theta \mid y) = -\frac{1}{2} n \log(2\pi) - \frac{1}{2} \log \sigma^2 - \frac{1}{2} \sum_{t=1}^{n} \frac{\left(y_t - \mu - \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}\right)^2}{\sigma^2} \to \max_{\theta}.$$

7. 4. ARMA model description

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

here p is defined as the first zero of PACF and q as the first zero of ACF.

7. 5. ARIMA model desciption

Denote

$$\begin{aligned}\Delta y_t &= y_t - y_{t-1} = (1-L)y_t,\ Ly_t = y_{t-1}\\ \Delta^2 y_t &= \Delta(y_t - y_{t-1}) = y_t - 2y_{t-1} + y_{t-2} = (1-L)^2 y_t\\ \Delta^d y_t &= (1-L)^d y_t.\end{aligned}$$

Taking an ARMA model:

$$\begin{aligned}y_t &= c + \varphi_1 y_{t-1} + \ldots + \varphi_p y_{t-p} + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}\\ \left(1 - \varphi_1 L - \ldots - \varphi_p L^p\right) y_t &= c + \left(1 + \theta_1 L + \ldots + \theta_q L^q\right)\varepsilon_t\end{aligned}$$

and applying it to $\Delta^d y_t$ results in the following model:

$$\begin{aligned}\left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)\Delta^d y_t &= c + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\varepsilon_t\\ \left(1 - \sum_{i=1}^{p} \varphi_i L^i\right)(1-L)^d y_t &= c + \left(1 + \sum_{i=1}^{q} \theta_i L^i\right)\varepsilon_t.\end{aligned}$$

Which is called ARIMA(p, q, d).

8. Predictive clustering

1. $Y = (y_1, ..., y_n)$
2. Find $z_j^{(m)}$ terms (todo: find out how they are called)

$$z_j^{(m)} = \left[y_{j-m+1}, ..., y_{j-1}, y_j\right] \to Z^m = \begin{pmatrix} z_m^{(m)} \\ z_{m+1}^{(m)} \\ \vdots \\ z_n^{(m)} \end{pmatrix}, Z^m \in \mathbb{R}^{n-m\times m}.$$

Inference:

3. $z_{n+1}^{(m)} = \left[y_{n-m+2}, ..., y_n, \hat{y}_{n+1}\right]$

   $\tilde{z}_{n+1}^{(m)} = z_{n+1}^{(m)}[: -1] = \left[y_{n-m+2}, ..., y_n\right]$

4. Collect a set of possible predictions:

$$S_{n+1} = \left\{z_i^{(m)}[m] \mid ||\tilde{z}_i^{(m)} - \tilde{z}_{n+1}|| < \varepsilon\right\}.$$

5. Choose single prediction:

- $\hat{y}_{n+1} = \text{mean}(S_{n+1})$
- $\hat{y}_{n+1} = \text{mode}(S_{n+1})$
- $\text{Cluster}(S_{n+1}) \to \{C_1, ..., C_k\}$

$\hat{y}_{n+1} = \text{mean}(C_i),\ i = \text{argmax}_j |C_j|.$

Multistep ahead prediction: $\tilde{z}_{n+2}^{(m)} = \left[y_{n-m+3}, ..., y_n, \hat{y}_{n+1}\right].$

2*. $Z^m \overset{\text{cluster}}{\to} C_1, ..., C_k$ clusters of $z^{(m)}$ vectors

$$X^m = \begin{pmatrix} \xi_1^{(m)} \\ \vdots \\ \xi_k^{(m)} \end{pmatrix},\ \xi_i = \text{"central element"}$$

3*. $\tilde{z}_{n+1}^{(m)} = \left[y_{n-m+2}, ..., y_n\right]$

4*. $S_n = \left\{\xi_i^{(m)}[m] |\ ||\tilde{\xi}_i^{(m)} - \tilde{z}_{n+1}^{(m)}|| < \varepsilon\right\}$

Let us modify the algorithm:

1. $Y = (y_1, ..., y_n)$

2. Given K, L : generate patterns (rus. *шаблоны*). For example:

$$K = 10, L = 4 : \ A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ \vdots & \vdots & \vdots & \vdots \\ 10 & 10 & 10 & 10 \end{pmatrix}$$

$$z_i^{(10,10,10,10)} = [y_{i-40}, ..., y_{i-10}, y_i]$$

Note that in this example A can have only elements with values from 1 to 10.

3. $\forall \alpha \in A : \alpha = (K_1, ..., K_L), K_i \in \overline{1, .., K}$

$$z_i^* = \left[y_{i-K_L-K_{L-1}-...K_1}, ..., y_{i-K_L-K_{L-1}}, y_{i-K_L}, y_i\right]$$

Generate $Z^\alpha = \begin{pmatrix} z^\alpha_{K_m-K_L} \\ z^\alpha_{K_m-K_{L+1}} \\ \vdots \\ z^\alpha_n \end{pmatrix} \ \forall \alpha \in A$

4. $\forall \alpha \in A$:

$$\tilde{z}^\alpha_{n+1} = \left[y_{n+1-K_L-K_{L-1}-...}, ..., y_{n+1-K_L}\right]$$

5. $\forall \alpha \in A$

$$S^\alpha_{n+1} = \left\{z^\alpha_i[i+1] \mid ||\tilde{z}^\alpha_i - \tilde{z}^\alpha_{n+1}||\right\}$$
$$S_{n+1} = \cup_\alpha \ S^\alpha_{n+1}$$

6. Classify point the point as predictable or not predictable.

6.1. Cluster $S_{n+1} \to C_1, ..., C_l, C_0$, where C_0 is a noise cluster and C_i, $i = \overline{1, .., l}$ are sorted by size from largest (C_1) to smallest (C_l).

$$\eta_1 = \frac{|C_1|}{|C_2|} \gg 1 \to \text{goto } 6.2$$

$$\eta_2 = \frac{|C_1|}{\sum_{i=1}^{l}|C_i|} > \varepsilon_1 \to \text{goto } 6.2$$

6.2. Estimate the variance of cluster C_1. If $\text{Var}(C_1) < \varepsilon_1 \to \text{goto } 7$, if $\text{IQR}(C_1) < \varepsilon_2 \to \text{goto } 7$

6.3. Identify the point as non-predictable and nove to the next point.

7. Obtain single prediction.

8. 1. Self-healing algorithms

These are used to make predictions if a value was skipped.

1. Forecast h steps ahead.
2. Run self-healing unill convergence or max iteration is reached.
3. Move to step 1.

9. Predictive clustering for trajectory forecasting

For $t = T + 1, ..., T + L$ generate a perturbed trajectory:

1. $\tilde{z}_t^\alpha$ – form truncated z vector for point $T + 1$ for pattern α.
2. $S_t^\alpha = \{z_i[-1] \mid \rho(\tilde{z}_i^\alpha,\ \tilde{z}_t^\alpha) < \varepsilon\}$
3. $S_t = \cup_\alpha S_t^\alpha$.
4. $\hat{y}^{(i)} = \text{mean}(S_t) + \mathcal{N}(0, \sigma^2)$.
5. Repeat for each $\alpha \in A$.

To formulate a forecast for each point t:

1. Create a set of forecasted values $S_t = \left\{\hat{y}_t^{(i)}\right\}_{i=1}^M$
2. Classify the point using respective algorithm.
3. If the point is forecastable, choose its value.

You can also substitute steps 4 and 5 in the previous algorithm with the following:

4. Cluster S_t obtaining $C_1, C_2, ..., C_l$.
5. For each cluster $\hat{y}_t^{(i)} = \text{mean}(C_i)$.

10. Clusterization for time series: DBSCAN, Wishart, metrics

10. 1. DBSCAN algorithm

10. 1. 1. Parameters

- ε
- minPts – minimum number of points in ε-neighbourhood to determine the core point.

10. 1. 2. Point types

- core point – a point that hase at least minPts points inside of its ε-neighbourhood including itself.
- border point – a point with less than minPts points in its ε-neighbourhood but is reachable from a core point
- noise point – every other point.

10. 1. 3. Reachability

q is *directly reachable* from p if q is in its ε-neighbourhood.

q is *reachable* from p if there exists a chain of points that includes p and q such that each next point is directly reachable from a previous one.

10. 1. 4. Algorithm

1. Initialization.

- All points are marked as unvisisted
- An empty list of clusters is created.

2. Marking the points one-by-one.

- For each point p:

  - if p is visited $\to$ skip

  - mark p as visited

  - calculate the number of points in ε-neighbourhood of p (including p itself).

  - if the # of neighbours $<$ minPts $\to$ mark p as noisy point

- otherwise:

    - create a new cluster;

    - add all neighbours to this cluster;

    - recuresively grow the cluster by adding points reachable from the core points.

3. CLuster expansion.

- For each point q in the current cluster

    - if q is not visited $\rightarrow$ mark it as visited, find its ε-neighbourhood and if it contains at least minPts points, add them to the cluster too.

10. 1. 5. Advantages

- DBSCAN can find clusters of any shape
- It does not require setting the number of clusters
- It is robust to outliers

10. 1. 6. Disadvantages

- This algorithm is very sensitive to its hyperparameter values
- The results vary depending in distance metric you choose
- Does not work well if data has differing cluster densities

10. 1. 7. Hyperparameter selection

- minPts is usually selected to be $\geq \#$ of features $+ 1$
- ε can be selected base on graph of distances to k-th nearest neighbours: it is mostly selected as the point of a sharp bend in the graph.

10. 2. Density and graph clusterization algorithm

Wishart algorithm modified by Lapko nad Chentsov.

10. 2. 1. Density estimation

$$p(x) = \frac{K}{V_{K(x)} \cdot n},$$

where:

$V_{K(x)}$ is the volume of K-dimensional hypersphere with center at x and containting K points,

$d_{K(x)}$ – radius of the spehere (a.k.a. dist. to the Kth nearest neighbour),

n – the total # of points.

10. 2. 2. Connectivity graph $G(Z_n, U_n)$

Here Z_n is the set of all vertices and U_n – the set of edges. So X_i is connected to X_j if

$$d\left(X_i, X_j\right) \leq d_K(x_i),\ i \neq j.$$

10. 2. 3. Connectivity subgraph $G(Z_i, U_i)$

Here, $Z_i = \{X_1, X_2, ..., X_i\}$ and U_i is the set of edges connecting points in Z_i.

10. 2. 4. Height significance of a cluster

Some cluster C_l is height significant w.r.t. some height $h > 0$ if

$$\max\left\{|p(X_i) - p\left(X_j\right)| \mid \forall X_i,\ X_j \in C_l\right\} \geq h.$$

10. 2. 5. Algorithm

1. Prepare the data.

- Calculate $d_K(X)\ \forall X \in$ data
- Sort the data based n $d_K(X)$ in ascending order

2. Initialization.

- Set counter $i = 1$
- Denote cluster label for i-th point as $\omega(X_i)$

3. Process points in sorted order.

For each point in subgraph $G(Z_i, U_i)$ do the following depending on the scenario:

A. Isolated vertex.

- If X is not connected to any other vertex create new cluster.

B. Connected to only one cluster C_l.

- If cluster is complete $\omega(X_i) = 0$ or in layman's terms we label X_i as noise.

- If cluster is not complete we label this point with class label.

C. Connected to multiple clusters.

- If all clusters as complete we lable the point as noise.
- Determine the number of significant cluters (in terms of height significance) $Z(h)$. If $Z(h) > 1$ then label all significant clusters as complete and delete insignificant clusters (by assigning them to noise) and assign the point to noise too: $\omega(X_i) = 0$. If there is only signle significant cluster combine all clusters into one C_{l_1}, $\omega(x) = l_1$

4. Update the counter $i = i + 1$, if $i \leq n$ goto step 3.

10. 2. 6. Advantages

- Algorithm itself determines the # of clusters
- Tends to label more points as noise the other algorithms do*
- Can find clusters of arbitrary form
- Not sensitive to noise

10. 2. 7. Disadvantages

- Sensitive to choice of hyperparameters
- Optimal hyperparameter values are hard to determine
- Slow (but we dgaf)

10. 2. 8. Hyperparameter selection

Obviously weuse grid search, but we need to compare clusterization results to each other. Let us discuss some metrics we can use.

1. Silhouette score $\mathcal{O}(n^2)$

Denote the avg. distance from point i to all other points in a cluster as $a(i)$.

Denote minimum avg. distance between i and other cluster(s?) as $b(i)$.

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$
$$S = \operatorname{mean}(S(i)).$$

2. Halinski-Harabaz index $\mathcal{O}(n)$

$$\mathrm{SSB} = \sum |C_k| \cdot \|\ C_k - C\ \|^2,$$
$$\mathrm{SSW} = \sum \|\ X - C_k\ \|^2,$$
$$\mathrm{CH} = \frac{\frac{\mathrm{SSB}}{K-1}}{\frac{\mathrm{SSW}}{N-K}},$$

where:

SSB – intercluster dispersion,

SSW – intracluster dispersion,

K – the number of clusters,

N – the number of points,

$|C_k|$ – the number of elements in the cluster C_k,

C – center of all data.

3. Davies-Bondeu index $\mathcal{O}(n)$

$$s_i = \frac{1}{|C_i|} \sum \|\ X - C_i\ \|$$
$$R_{ij} = \frac{s_i + s_j}{\|\ C_i - C_j\ \|}$$
$$D_i = \max_{j,\ i \neq j} R_{ij}$$
$$\mathrm{DB} = \frac{1}{K} \sum D_i, \quad \mathrm{DB} \in [0, \infty)$$

The lower DB is the better.

4. Vaname ratio criterion.

$$\mathrm{TSS} = \sum \|\ X_i - C\ \|^2$$
$$\mathrm{BSS} = \sum |\ C_k\ |\ \|\ C_k - C\ \|^2$$
$$\mathrm{VR} = \frac{\mathrm{TSS}}{\mathrm{BSS}} \in [0, 1]$$

The closer to 0, worse the clustering is.

11. Time series forecasting with neural networks

11. 1. Form dataset

- Use time windows:

$$X = [y_{t-n}, y_{t-n+1}, ..., y_{t-1}]$$
$$y = [y_t] - \text{one-step ahead prediction}$$
$$y = [y_t, y_{t+1}, ..., y_{t+h-1}] - \text{h-step ahead prediction}$$

- You can add other features

11. 2. Neural network acrhitectures

11. 2. 1. FNN

Take X as input of size n and make the output layer the size of number of steps ahead you want to predict.

Problem: these models do not account for time dependence.

11. 2. 2. RNN

Problem: RNNs do not have long-term memory, as earlier outputs decay and may not be accounted for in the end.

11. 2. 3. LSTM

<Import pic>

11. 2. 4. CNN

Idea: let us use 1D filters to extract local patterns (motives). Use dialation ot increase lookback period.

Example:

1. kernel $= 3, d = 1 : [y_{t-2}, y_{t-1}, y_t]$
2. kernel $= 3, d = 2 : [y_{t-4}, y_{t-2}, y_t]$

But here you can sometimes use data from the future thus breaking causality. To deal with this problem TCNN was developed.

11. 2. 5. TCNN

Idea: Combine properties of CNN and LSTM to make thing good.

- retain causality (solved by causal convolutions — left padding with size $k-1$ is used when applying filters instead of values to the right of one that the filter is applied to);
- enable parallelization;
- long effective memory (solved by using all sorts of different dialations).

Temporal (TCNN) block.

Input $\to$ Causal dialated convolution $\to$ Normalization $\to$ Activation function $\to$ Backprop $\to$ Causal dialated convolution $\to$ Normalization $\to$ Activation function $\to$ Output

Deep TCNN: skip connections.

TCNN Block gets some data as input and summed with it. Note that 1D convolution is used to adjuct the sizes.

11. 2. 6. Transformers

…

11. 2. 7. VAE

VAE (*variational autoencoder*) architecture:

1. Encoder: Input $\to$ CNN/RNN $\to$ Flatten $\to$ FC $\to$ $\mu,\ \log(\sigma^2)$, a.k.a. latent distribution of data
2. Sampling: $z = \mu + \sigma \cdot \varepsilon,\ \varepsilon \sim \mathcal{N}(0, 1)$
3. Decoder: z $\to$ FC $\to$ Reshape $\to$ CNN/RNN $\to$ output
4. ELBO loss (maximization task)

$$\text{ELBO} = \mathbb{E}[\log p(x \mid z)] - K \cdot L(q(z \mid x) \| p(z)) \to \max$$