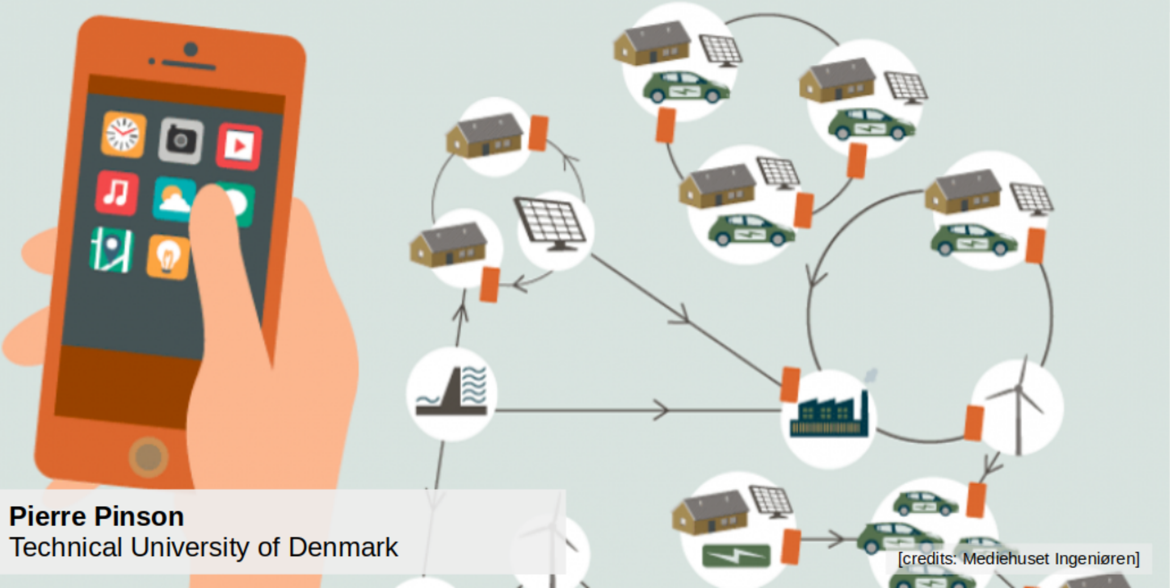
Module 10 – Renewable Energy Forecasting: Advanced Topics

10.2 Nonstationarity and time-adaptivity

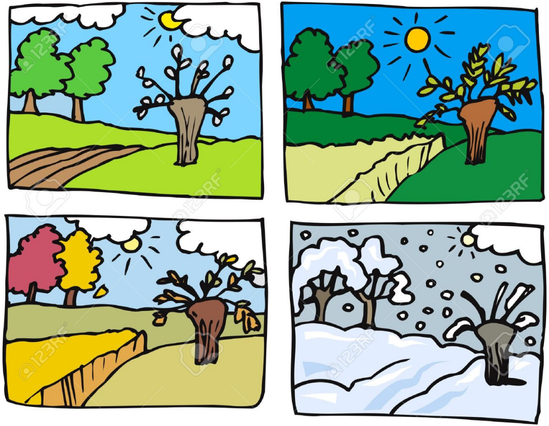


Pierre Pinson
Technical University of Denmark

[credits: Mediehuset Ingeniøren]

Why could there be nonstationarity?

- **Nonstationarity** broadly means that the characteristics of the underlying processes we consider may vary with time
- Examples:

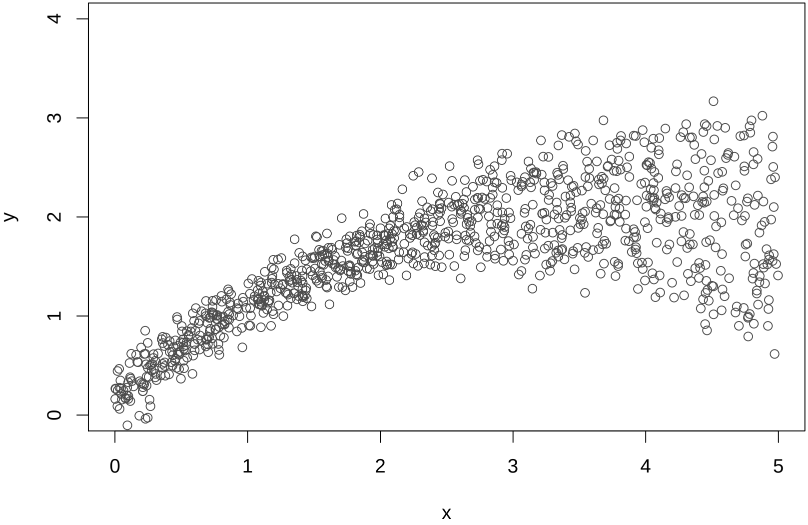


Seasons

Dirty blades

Let's look at an example

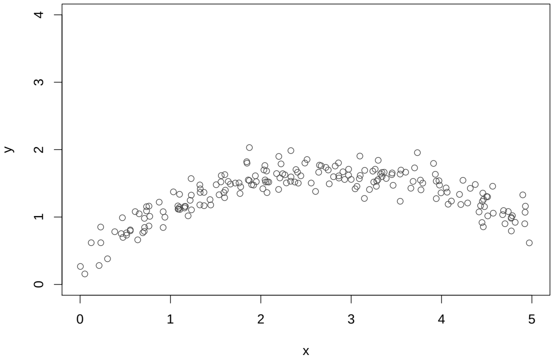
- We collect 1000 (x_t, y_t) pairs (say, over a period of 1000 hours, $t = 1, \dots, 1000$)

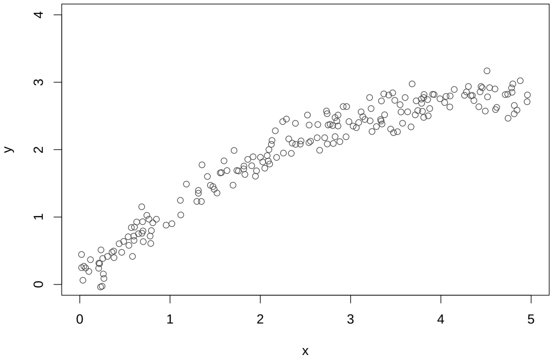


- It is just very noisy, right?

Digging into the data

- If we were to plot the data collected over the first 200 hours, and over the last 200 hours...





- So.. maybe it is not just noise

Estimation on sliding windows

- Instead of estimating model parameters once for all, one may estimate them on sliding windows

Given a window size n, The **Least-Squares (LS) estimate** $\hat{\beta}_t$ at time t is given by

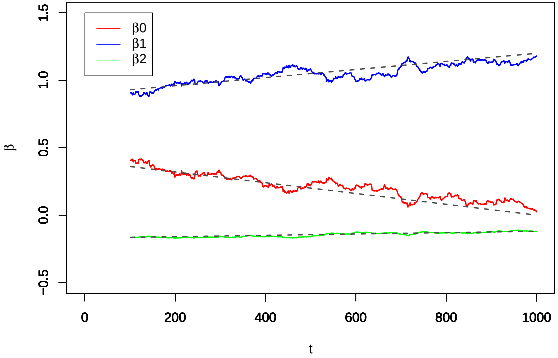
$$\hat{\boldsymbol{\beta}}_t = \arg\min_{\boldsymbol{\beta}} \sum_{i=t-n}^{t} \left(y_i - \boldsymbol{\beta}^\top \mathbf{x}_i \right)^2 = (\mathbf{X}_t^\top \mathbf{X}_t)^{-1} \mathbf{X}_t^\top \mathbf{y}_t$$

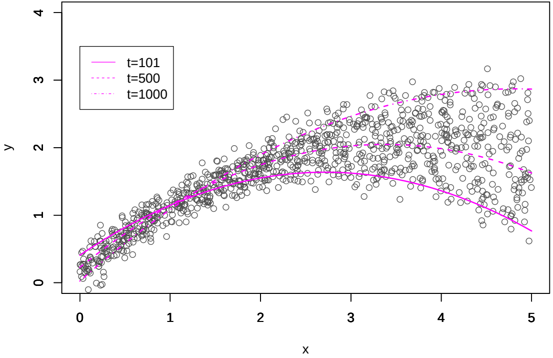
with

$$\hat{\boldsymbol{\beta}}_t = \begin{bmatrix} \hat{\beta}_{0,t} \\ \hat{\beta}_{1,t} \\ \cdots \\ \hat{\beta}_{P,t} \end{bmatrix}, \quad \mathbf{X}_t = \begin{bmatrix} 1 & x_{t-n} & x_{t-n}^2 & \dots & x_{t-n}^P \\ 1 & x_{t-n+1} & x_{t-n+1}^2 & \dots & x_{t-n+1}^P \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_t & x_t^2 & \dots & x_t^P \end{bmatrix}, \quad \mathbf{y}_t = \begin{bmatrix} y_{t-n} \\ y_{t-n+1} \\ \vdots \\ y_t \end{bmatrix}$$

Application and results

- Polynomial regression of order 2 (quadratic should be enough)
- Window size of $n = 100$
- Parameters are then estimated from $t = 101$ to $t = 1000$





- It works nicely but it may be a bit heavy to recalculate model parameters every time steps with such overlapping windows(!)
- Optimally, one would want to *lighten the computation burden* as much as possible, while *limiting the amount of data to store*

Online learning

- The fundamental principle of online learning relies on **recursivity**
- **At a given time** $t-1$:
  - A set of model parameters $\hat{\boldsymbol{\beta}}_{t-1}$ was estimated
  - All data $\{(x_i, y_i)\}_{i \le t-1}$ is considered as already "used", and hence dumped
  - It may be that some information Ω_{t-1} (of very limited size) is kept in memory
- **Then, at time** t:
  - Only data at time t, i.e., (x_t, y_t) is recorded and used as input
  - The model parameters are updated with

$$\hat{\boldsymbol{\beta}}_t = \hat{\boldsymbol{\beta}}_{t-1} + \mathcal{F}((x_t, y_t), \Omega_{t-1}, \tau)$$

  Optimally, $\mathcal{F}$ only involves simple operations e.g. matrix multiplications. τ includes useful parameters, e.g. memory

- Obviously, one needs an initialization for $\hat{\boldsymbol{\beta}}_0$

Online learning with Recursive Least Squares (RLS)

- Choose a forgetting factor ν, $\nu < 1$ (e.g., $\nu = 0.99$)
- Consider that the Least Squares estimation problem to be solved down-weight past observations, i.e.,

  $$\hat{\boldsymbol{\beta}}_t = \arg\min_{\boldsymbol{\beta}} \sum_{i<t} \nu^{t-i} \left(y_i - \boldsymbol{\beta}^\top \mathbf{x}_i \right)^2$$

- and we skip the necessary algebra to obtain the recursion for the RLS estimator with forgetting:

Given a forgetting factor ν, The **Recursive Least-Squares (LS) estimate** $\hat{\boldsymbol{\beta}}_t$ at time t is given by

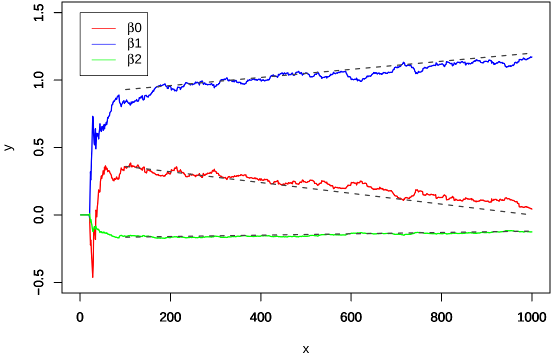
$$R_t = \nu R_{t-1} + \mathbf{x}_t \mathbf{x}_t^\top$$
$$\hat{\boldsymbol{\beta}}_t = \hat{\boldsymbol{\beta}}_{t-1} + R_t^{-1} \mathbf{x}_t (y_t - \hat{\boldsymbol{\beta}}_{t-1}^\top \mathbf{x}_t)$$

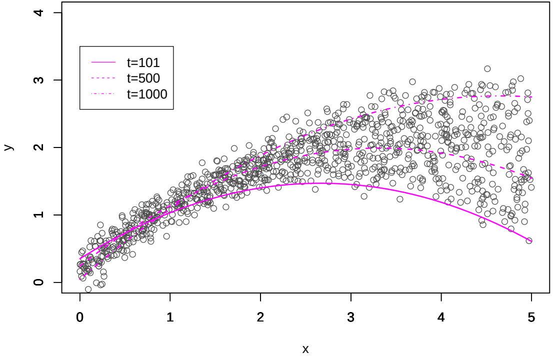
where

$$\mathbf{x}_t = \begin{bmatrix} 1 \\ x_{t-n} \\ x_{t-n}^2 \\ \cdots \\ x_{t-n}^P \end{bmatrix}$$

Application and results

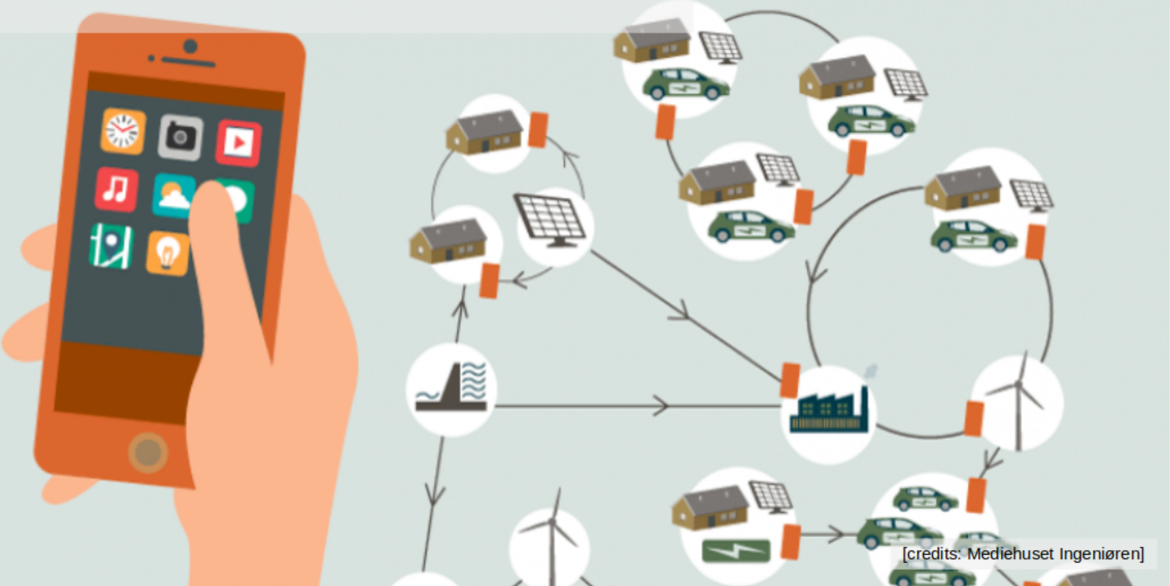
- Polynomial regression of order 2 (quadratic should be enough)
- Forgetting factor $\nu = 0.99$
- Initialization: $\hat{\beta}_0 = \mathbf{0}$





- It works as well as the sliding windows, while being (potentially) much faster and avoiding re-using a lot of data
- How does one decide on the forgetting factor ν to use?

Use the self-assessment quizz to check your understanding!



[credits: Mediehuset Ingeniøren]