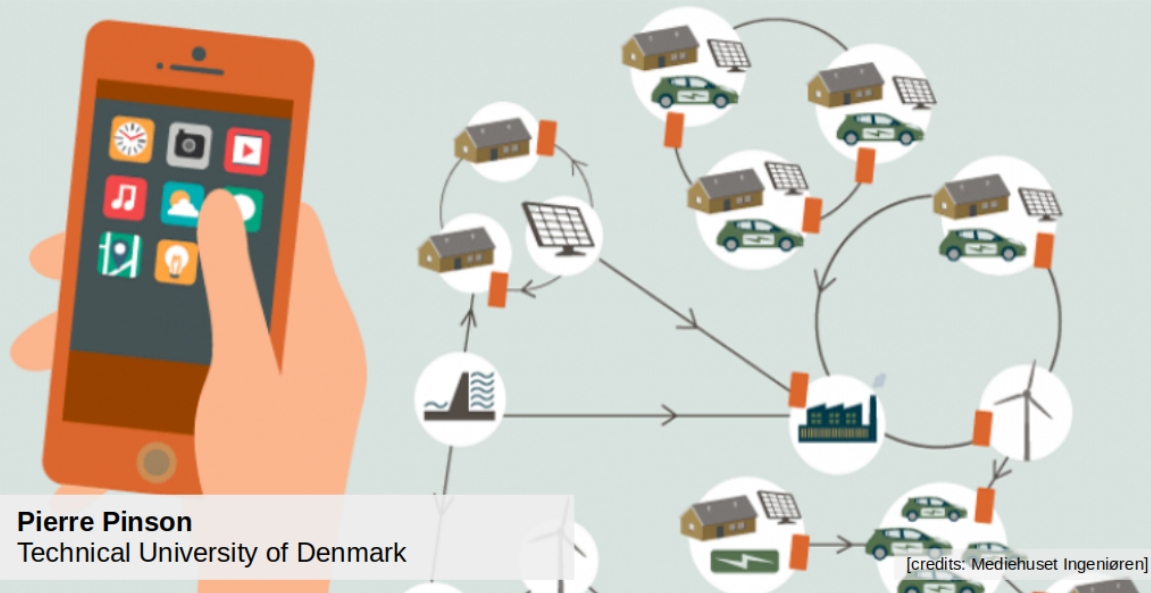


# Module 10 – Renewable Energy Forecasting: Advanced Topics

## 10.1 From linear to nonlinear regression

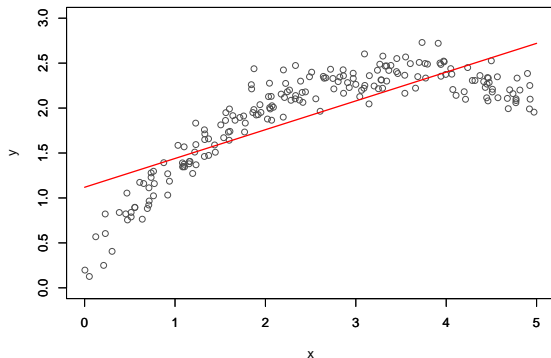
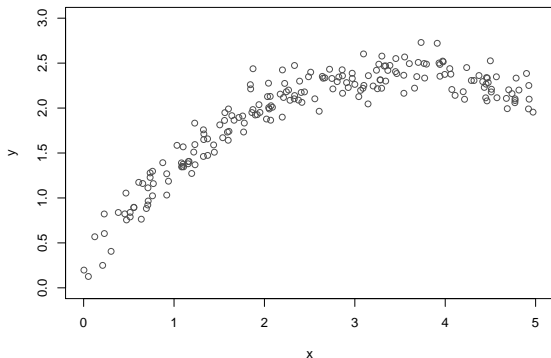


**Pierre Pinson**  
Technical University of Denmark

[credits: Mediehuset Ingeniøren]

# A motivation for polynomial regression

- We have obtained input-output pairs  $\{(x_t, y_t)\}_t$  over the last 200 time steps and aim to model their relationship



- Using linear regression does not look like such a good idea...

- A simple linear relation is assumed between  $x$  and  $y$ , i.e.,

$$y_t = \beta_0 + \beta_1 x_t + \varepsilon_t, \quad t = t_n - n, \dots, t_n$$

where

- $\beta_0$  and  $\beta_1$  are the model parameters (called *intercept* and *slope*)
- $\varepsilon_t$  is a noise term, which you may see as our forecast error we want to minimize

The linear regression model can be reformulated in a more compact form as

$$y_t = \boldsymbol{\beta}^\top \mathbf{x}_t + \varepsilon_t, \quad t = t_n - n, \dots, t_n$$

with

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \quad \mathbf{x}_t = \begin{bmatrix} 1 \\ x_t \end{bmatrix}$$

## Least Squares (LS) estimation

- Now we need to find the best value of  $\beta$  that describes this cloud of point
- Under a number of assumptions, which we overlook here, the (best) model parameters  $\hat{\beta}$  can be readily obtained with **Least-Squares (LS) estimation**

The **Least-Squares (LS) estimate**  $\hat{\beta}$  of the linear regression model parameters is given by

$$\hat{\beta} = \arg \min_{\beta} \sum_t \varepsilon_i^2 = \arg \min_{\beta} \sum_t \left( y_t - \beta^\top \mathbf{x}_t \right)^2 = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

with

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{t_n-n} \\ 1 & x_{t_n-n+1} \\ \vdots & \vdots \\ 1 & x_{t_n} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_{t_n-n} \\ y_{t_n-n+1} \\ \vdots \\ y_{t_n} \end{bmatrix}$$

## Extending to polynomial regression

- We could also assume more generally a polynomial relation between  $x$  and  $y$ , i.e.,

$$y_t = \beta_0 + \sum_{p=1}^P \beta_p x_t^p + \varepsilon_t, \quad t = t_n - n, \dots, t_n$$

where

- $\beta_p$ ,  $p = 0, \dots, P$  are the model parameters
- $\varepsilon_t$  is a noise term, which you may see as our forecast error we want to minimize

This polynomial regression can be reformulated in a more compact form as

$$y_t = \boldsymbol{\beta}^\top \mathbf{x}_t + \varepsilon_t, \quad i = t_n - n, \dots, t_n$$

with

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \dots \\ \beta_P \end{bmatrix}, \quad \mathbf{x}_t = \begin{bmatrix} 1 \\ x_t \\ \dots \\ x_t^P \end{bmatrix}$$

# Least Squares (LS) estimation

- As the model is linear we can still use LS estimation!

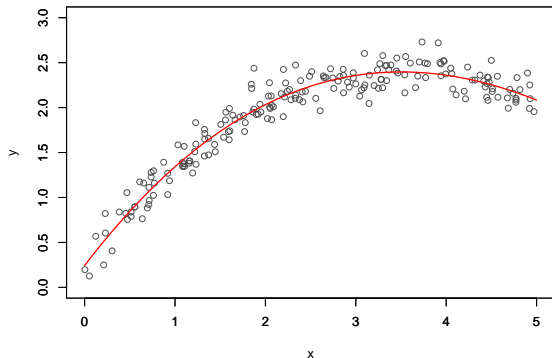
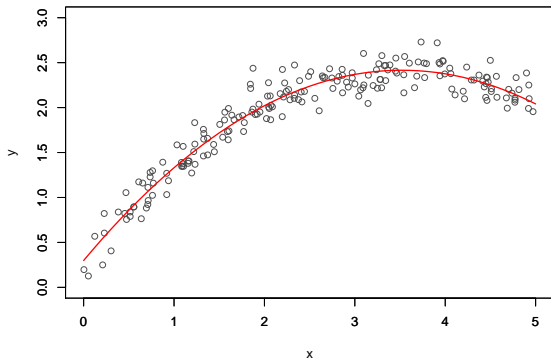
The **Least-Squares (LS) estimate**  $\hat{\beta}$  of the linear regression model parameters is given by

$$\hat{\beta} = \arg \min_{\beta} \sum_t \varepsilon_t^2 = \arg \min_{\beta} \sum_t \left( y_t - \beta^\top \mathbf{x}_t \right)^2 = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

with

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_P \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{t_n-n} & x_{t_n-n}^2 & \cdots & x_{t_n-n}^P \\ 1 & x_{t_n-n+1} & x_{t_n-n+1}^2 & \cdots & x_{t_n-n+1}^P \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{t_n} & x_{t_n}^2 & \cdots & x_{t_n}^P \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_{t_n-n} \\ y_{t_n-n+1} \\ \vdots \\ y_{t_n} \end{bmatrix}$$

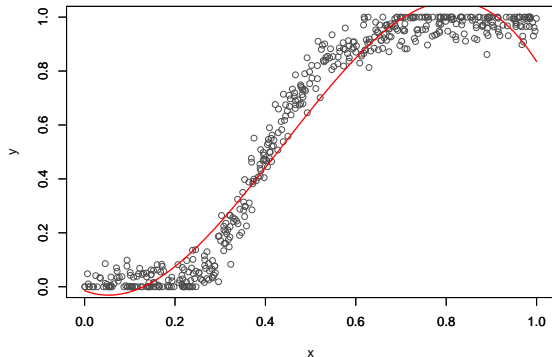
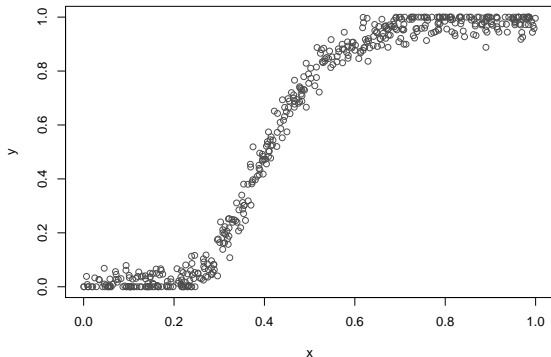
- We apply polynomial regression with  $P = 2$  (quadratic) and  $P = 3$  (cubic)



- They both look quite nicer than the simple linear fit
- We are lucky here that the relationship truly is quadratic... if fitting higher-order polynomials,  $\hat{\beta}_i = 0$ ,  $p > 2$
- In general, higher-order may yield spurious results(!)

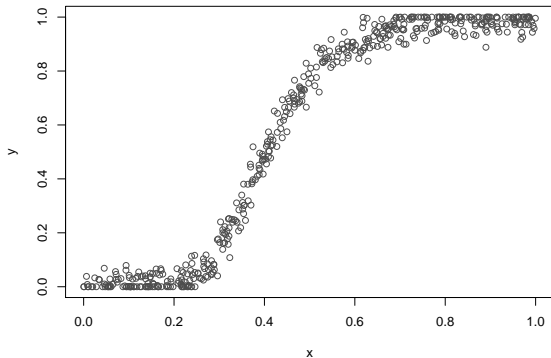
## With a more general nonlinear regression case

- Let's model something that looks more like a power curve, and try a cubic fit (polynomial regression with  $P = 3$ )



- Indeed we need to find something better than simply fitting polynomials that way
- **Ideas?**

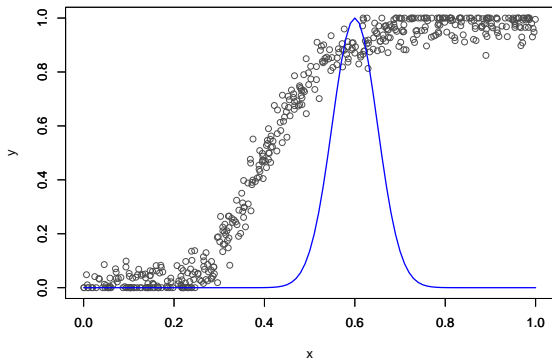
- Use polynomial regression, though locally fitting those models



- Consider a number of  $m$  of fitting points, e.g.,  $0, 0.1, \dots, 1$
  - Use some weighting function  $\omega$  to give more or less importance to the various data points
- After fitting those models, we can reconstruct the full nonlinear regression curve by connecting the values obtained at the fitting points

# Local polynomial regression

- Let us concentrate on a given fitting point  $x_u$ , e.g.  $x_u = 0.6$
- If aiming to fit a model that represents what happens in the neighborhood of  $x_u$ , more importance is to be given to data points close to  $x_u$



- For all data points  $\{(x_t, y_t)\}_t$ , the corresponding weight  $w_t$  can be defined as

$$w_t = \omega(x_t - x_u, \kappa)$$

- For instance with  $\omega$  a Gaussian kernel,

$$\omega(x_t - x_u, \sigma) = \exp\left(-\frac{(x_t - x_u)^2}{2\sigma^2}\right)$$

(Example Gaussian kernel with  $x_u = 0.6$  and  $\sigma = 0.05$ )

## Weighted Least Squares (WLS) estimation

- The previously introduced LS estimators can be generalized to account for weights given to data points

The **Weighted Least-Squares (WLS) estimate**  $\hat{\beta}$  of the polynomial regression model parameters fitted at  $x_u$  is given by

$$\hat{\beta} = \arg \min_{\beta} \sum_t w_t \varepsilon_t^2 = \arg \min_{\beta} \sum_t w_t \left( y_t - \beta^\top \mathbf{x}_t \right)^2 = (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{y}$$

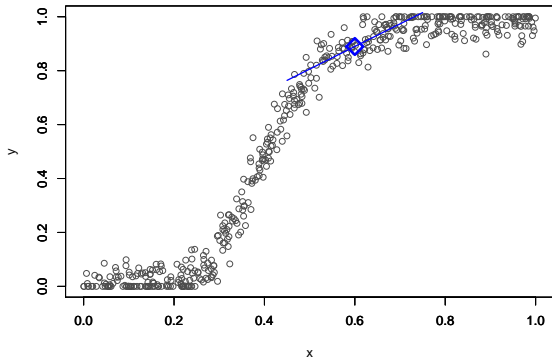
with

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_P \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 1 & x_{t_n-n} & x_{t_n-n}^2 & \cdots & x_{t_n-n}^P \\ 1 & x_{t_n-n+1} & x_{t_n-n+1}^2 & \cdots & x_{t_n-n+1}^P \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{t_n} & x_{t_n}^2 & \cdots & x_{t_n}^P \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_{t_n-n} \\ y_{t_n-n+1} \\ \vdots \\ y_{t_n} \end{bmatrix}$$

$$\text{and } \mathbf{W} = \begin{bmatrix} w_{t_n-n} & & & 0 \\ & w_{t_n-n+1} & & \\ & & \ddots & \\ 0 & & & w_{t_n} \end{bmatrix}$$

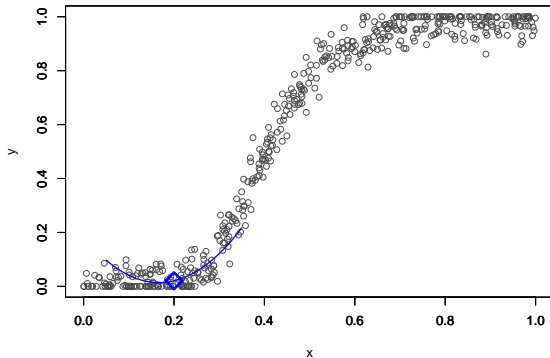
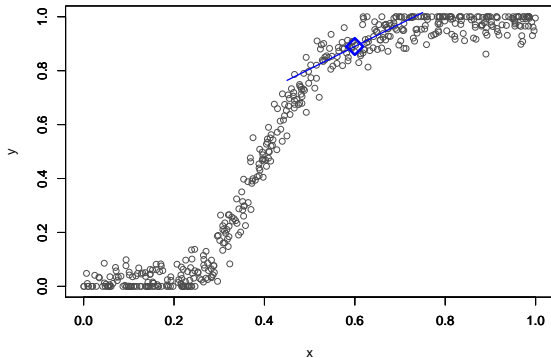
## Applying the idea to a few fitting points

- First for that we focused on, i.e.,  $x_u = 0.6$ , say with a polynomial of degree 1



# Applying the idea to a few fitting points

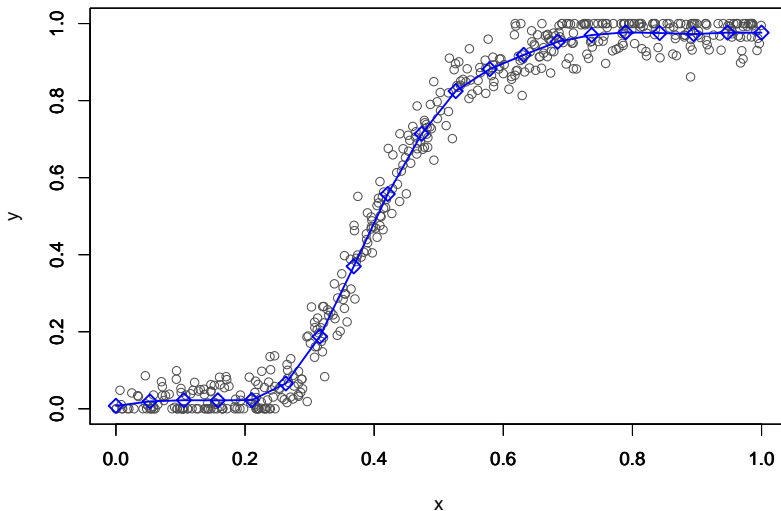
- First for that we focused on, i.e.,  $x_u = 0.6$ , say with a polynomial of degree 1



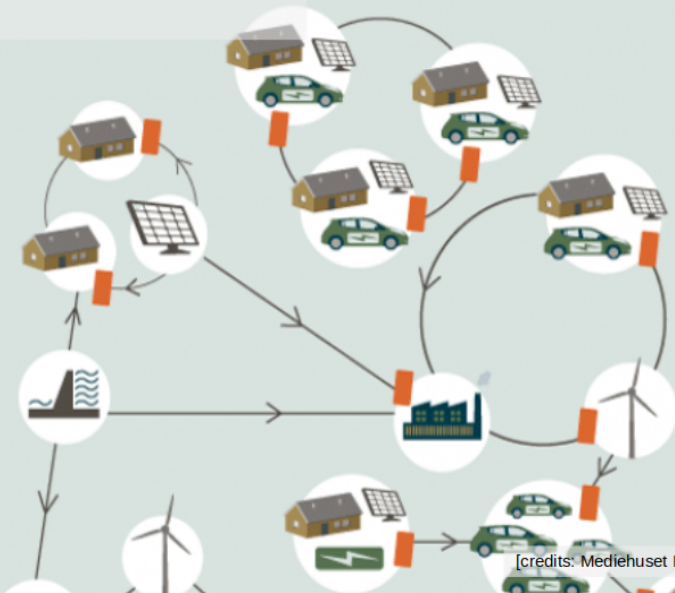
- And then for another fitting point,  $x_u = 0.2$ , say with a polynomial of degree 2

## The resulting power curve model

- We first fix a **polynomial order**, choice of **kernel and its parameters**, and **number of fitting points**,
- We then apply local polynomial regression at all fitting points and record the value at those points, and eventually connect all those points, e.g., with linear interpolation



**Use the self-assessment quizz to check your understanding!**



[credits: Mediehuset Ingeniøren]