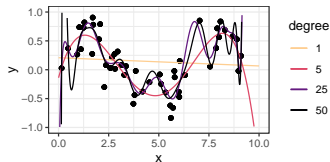


# Introduction to Machine Learning

## Supervised Regression

## Polynomial Regression Models

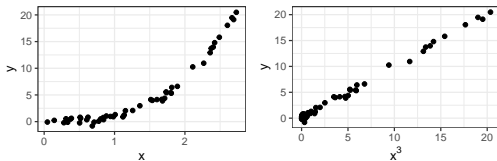


### Learning goals

- Learn about general form of linear model
- See how to add flexibility by using polynomials
- Understand that more flexibility is not necessarily better

# INCREASING FLEXIBILITY

- Recall our definition of LM: model  $y$  as linear combo of features
- But: isn't that pretty **inflexible**?
- E.g., here,  $y$  does not seem to be a linear function of  $x$ ...



... but relation to  $x^3$  looks pretty linear!

- Many other trafo's conceivable, e.g.,  $\sin(x_1)$ ,  $\max(0, x_2)$ ,  $\sqrt{x_3}$ , ...
- Turns out we can use LM much more **flexibly** (and: it's still linear)  
~> interpretation might get less straightforward, though



# THE LINEAR MODEL

- Recall what we previously defined as LM:

$$f(x) = \theta_0 + \sum_{j=1}^p \theta_j x_j = \theta_0 + \theta_1 x_1 + \cdots + \theta_p x_p \quad (1)$$

- Actually, just special case of "true" LM
- The linear model** with **basis functions**  $\phi_j$ :

$$f(\mathbf{x}) = \theta_0 + \sum_{j=1}^p \theta_j \phi_j(x_j) = \theta_0 + \theta_1 \phi_1(x_1) + \cdots + \theta_p \phi_p(x_p)$$

- In Eq. 1, we implicitly use identity trafo:  $\phi_j = \text{id}_x : x \mapsto x \quad \forall j$   
 $\rightsquigarrow$  we often say LM and imply  $\phi_j = \text{id}_x$



# THE LINEAR MODEL

- Are models like  $f(\mathbf{x}) = \theta_0 + \theta_1 x^2$  **really linear**?

- Certainly not in covariates:

$$\begin{aligned} a \cdot f(x, \theta) + b \cdot f(x_*, \theta) &= \theta_0(a + b) + \theta_1(ax^2 + bx_*^2) \\ &\neq \theta_0 + \theta_1(ax + bx_*)^2 \\ &= f(ax + bx_*, \theta) \end{aligned}$$

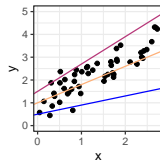
- Crucially, however, **linear in params**:

$$\begin{aligned} a \cdot f(x, \theta) + b \cdot f(x, \theta^*) &= a\theta_0 + b\theta_0^* + (a\theta_1 + b\theta_1^*)x^2 \\ &= f(x, a\theta + b\theta^*) \end{aligned}$$

- NB: we still call design matrix **X**, incorporating possible trafos:

$$\mathbf{X} = \begin{pmatrix} 1 & \phi_1(x_1^{(1)}) & \dots & \phi_p(x_p^{(1)}) \\ \vdots & \vdots & & \vdots \\ 1 & \phi_1(x_1^{(n)}) & \dots & \phi_p(x_p^{(n)}) \end{pmatrix}$$

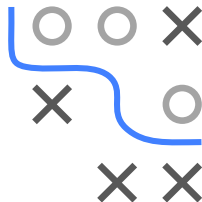
$\rightsquigarrow$  solution via normal equations as usual



$$\theta = (0.5, 0.4)^T$$

$$\theta = (1.0, 0.8)^T$$

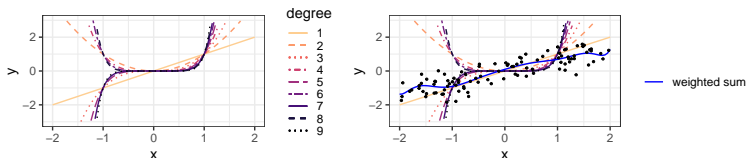
$$\theta = (1.5, 1.2)^T$$



# POLYNOMIAL REGRESSION

- Simple & flexible choice for basis funs: ***d*-polynomials**
- Idea: map  $x_j$  to (weighted) sum of its monomials up to order  $d \in \mathbb{N}$

$$\phi^{(d)} : \mathbb{R} \rightarrow \mathbb{R}, \quad x_j \mapsto \sum_{k=1}^d \beta_k x_j^k$$



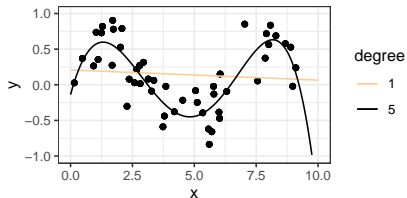
- How to estimate coefficients  $\beta_k$ ?
  - Both LM & polynomials **linear** in their params  $\rightsquigarrow$  merge
  - E.g.,  $f(\mathbf{x}) = \theta_0 + \theta_1 \phi^{(d)}(x) = \theta_0 + \sum_{k=1}^d \theta_{1,k} x^k$

$$\rightsquigarrow \mathbf{X} = \begin{pmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \dots & (x^{(1)})^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x^{(n)} & (x^{(n)})^2 & \dots & (x^{(n)})^d \end{pmatrix}, \quad \boldsymbol{\theta} \in \mathbb{R}^{d+1}$$



# POLYNOMIAL REGRESSION – EXAMPLES

Univariate regression,  $d \in \{1, 5\}$



- Data-generating process:

$$y = 0.5 \sin(x) + \epsilon,$$

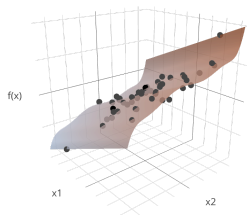
$$\epsilon \sim \mathcal{N}(0, 0.3^2)$$

- Model:

$$f(x) = \theta_0 + \sum_{k=1}^d \theta_{1,k} x^k$$



Bivariate regression,  $d = 7$



- Data-generating process:

$$y = 1 + 2x_1 + x_2^3 + \epsilon,$$

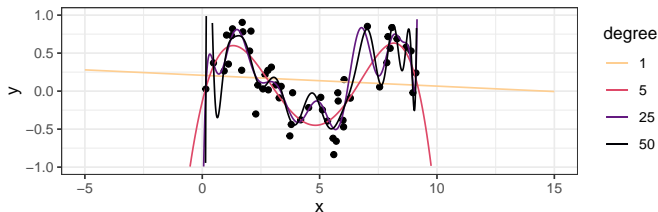
$$\epsilon \sim \mathcal{N}(0, 0.5^2)$$

- Model:

$$f(x) = \theta_0 + \theta_1 x_1 + \sum_{k=1}^7 \theta_{2,k} x_2^k$$

# COMPLEXITY OF POLYNOMIALS

- Higher  $d$  allows to learn more complex functions  
~> richer hyp space / higher **capacity**



- Should we then simply let  $d \rightarrow \infty$ ?
  - No**: data contains random **noise** – not part of true DGP
  - Model with overly high capacity learns all those spurious patterns ~> poor generalization to new data
  - Also, higher  $d$  can lead to oscillation esp. at bounds (Runge's phenomenon<sup>1</sup>)

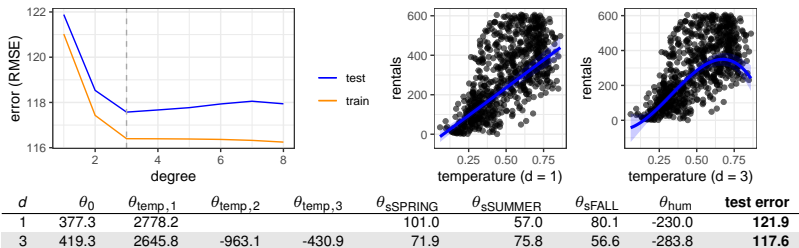
<sup>1</sup> Interpolation of  $m$  equidistant points with  $d$ -polynomial only well-conditioned for  $d < 2\sqrt{m}$ .  
Plot: 50 points, models with  $d \geq 14$  instable (under equidistance assumption).

# BIKE RENTAL EXAMPLE

- OpenML task `dailybike`: predict rentals from weather conditions
- Hunch: non-linear effect of temperature  $\rightsquigarrow$  include with polynomial:

$$f(\mathbf{x}) = \sum_{k=1}^d \theta_{\text{temperature},k} x_{\text{temperature}}^k + \theta_{\text{season}} x_{\text{season}} + \theta_{\text{humidity}} x_{\text{humidity}}$$

- Test error<sup>2</sup> confirms suspicion  $\rightsquigarrow$  minimal for  $d = 3$



- Conclusion: flexible effects can improve fit/performance

<sup>2</sup>Reliable insights about model performance only via separate test dataset not used during training (here computed via 10-fold *cross validation*). Much more on this in Evaluation chapter.

