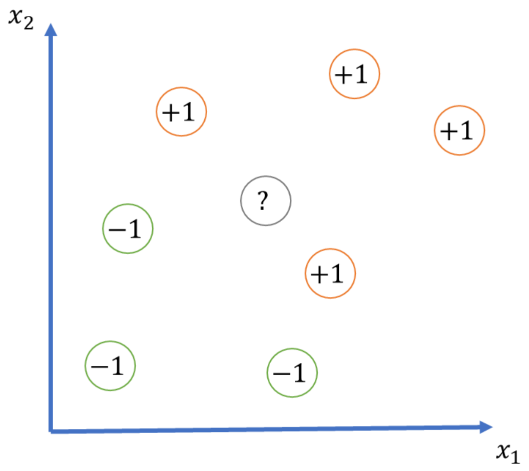
Ollscoil Teicneolaíochta na Mumhan
Munster Technological University

Machine Learning

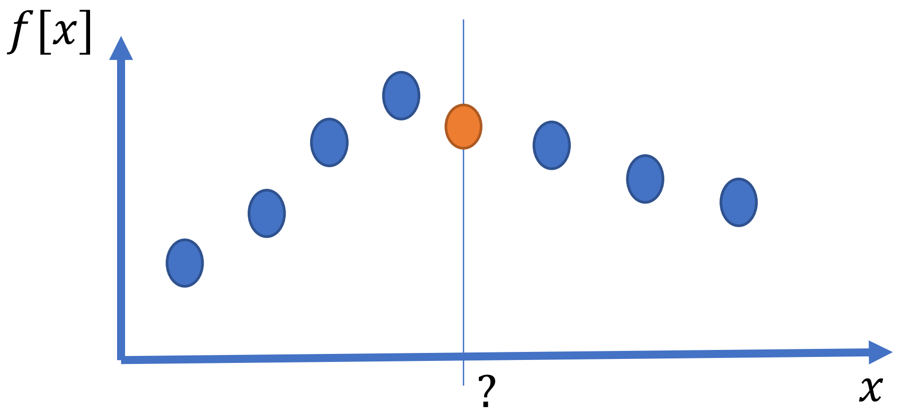
Lecture 18: Regression & parameter estimation

Regression vs. classification

- Up until now we have only looked at supervised classification problems, i.e. problems where the target was a discrete set of classes



- Regression aims at estimating real-valued functions from data instead



Regression

- More formally, we are given a training set of feature-value pairs

$$\{x_1 \rightarrow y_1, \dots, x_n \rightarrow y_n\}$$

- And want to determine a function $f\colon \mathbb{R}^n \rightarrow \mathbb{R}^m$ which allows us to compute the values of y from x, i.e.

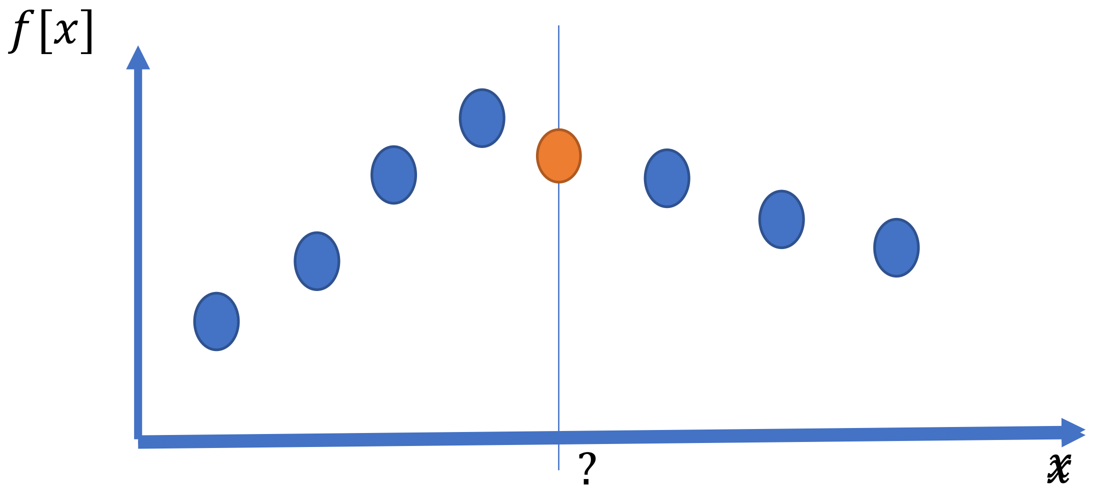
$$y = f[x]$$

- Typically, the function should perform well on the training set, i.e. the residual, the difference between prediction and actual value, should be small

$$\mathrm{r} = y - f[x]$$

Nearest-neighbour regression

- Some classification algorithms that we have seen can be generalised to also cover this type of problem
- For example the k-nearest neighbour algorithm could be used to return the average value instead of the class of the nearest neighbours



Parameter estimation

- In many applications we can assume that the functional relationship can be described by a parameterised known model function

$$y = f[x; \theta]$$

- In this case the problem reduces to estimating the set of parameters θ that best represents out training data

- If the model function f cannot be derived from the application domain, empirical model functions can be used
- For example we can fit a hyperplane through our data (linear regression), i.e. choose the function to be

$$f[x] = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

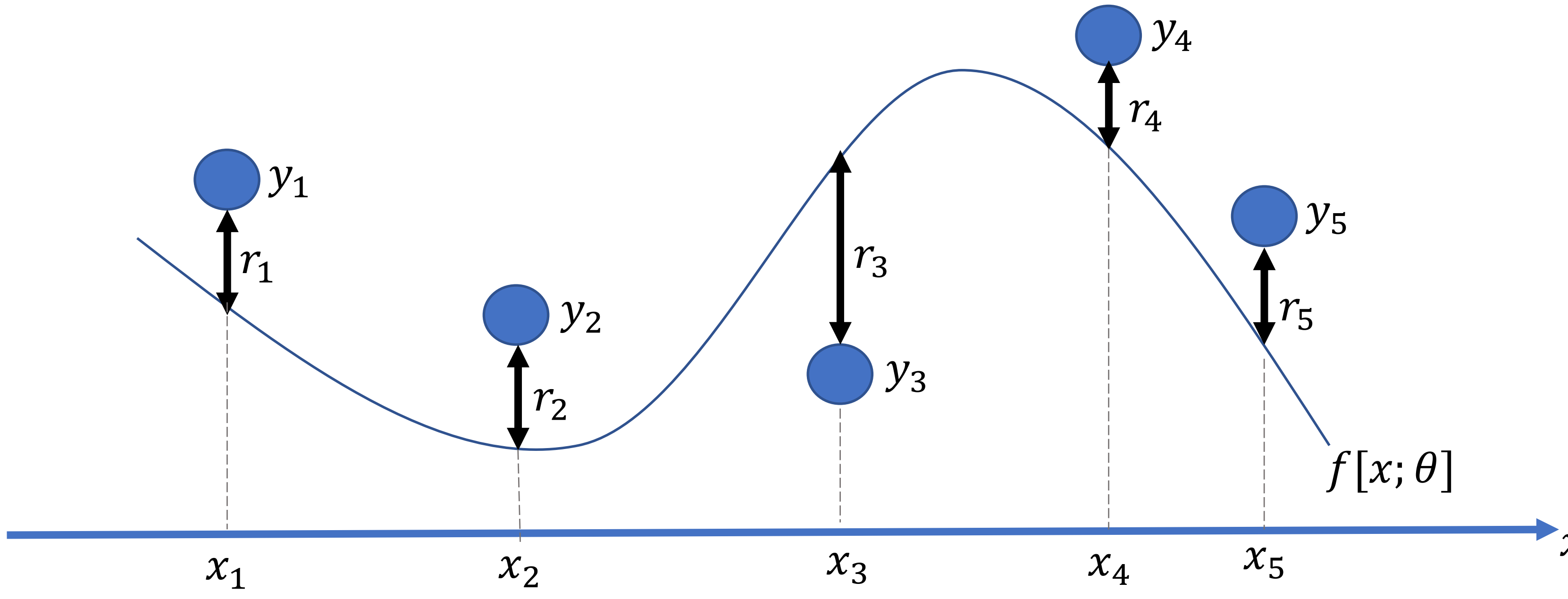
Model functions

- The model function f only provides the “shape” that needs to be fitted to the data in order to make sense
- The parameters are the used to calibrate the model and fit a particular instance
- If the physics of a problem is know, this can help improve accuracy and at the same time help reduce the number of parameters
- Many engineering sciences are creating models in their domain, for example:
    - Building energy model simulations
    - Geometric models for geodetic surveying applications
    - Flight dynamics modelling for aircraft control systems
    - Hydrodynamic models for marine applications
    - …
- Before blindly using an empirical model function, always evaluate if a domain specific model function is better suited

Parameter estimation

- A common approach for estimating the unknown parameters θ from the training data is to minimise the squared sum of residuals over the training set

$$\Omega = \frac{1}{2}\sum_i r_i^T r_i = \frac{1}{2}\sum_i (y_i - f[x_i;\theta])^T (y_i - f[x_i;\theta])$$



Parameter estimation

- A common approach for estimating the unknown parameters θ from the training data is to minimise the squared sum of residuals over the training set

$$\Omega = \frac{1}{2}\sum_i r_i^T r_i = \frac{1}{2}\sum_i (y_i - f[x_i;\theta])^T (y_i - f[x_i;\theta])$$

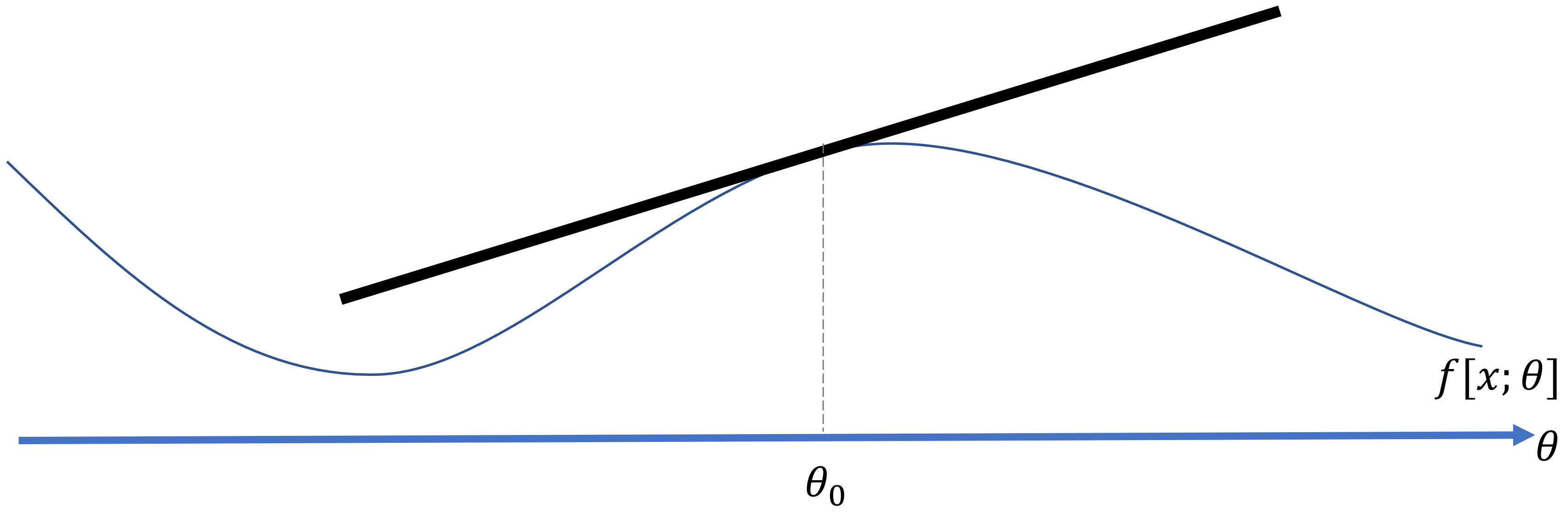
- The minimum of a function is where its derivative is zero, so in order to minimise Ω we look at the necessary condition

$$\frac{\partial\Omega}{\partial\theta} = \sum_i \frac{\partial}{\partial\theta}(f^T[x_i;\theta]f[x_i;\theta]) - 2y_i^T \frac{\partial}{\partial\theta} f[x_i;\theta] = 0$$

Linear approximation

- If the model function can be approximated by a Taylor expansion around some value for the parameter vector θ_0

$$f[x_i;\theta] \approx \underbrace{f[x_i;\theta_0]}_{f_{0_i}} + \underbrace{\left(\frac{\partial}{\partial\theta} f\right)[x_i;\theta_0]}_{J_i} \underbrace{(\theta-\theta_0)}_{\Delta\theta} = f_{0_i} + J_i \Delta\theta$$



Linear approximation

- If the model function can be approximated by a Taylor expansion around some value for the parameter vector θ_0

$$f[x_i;\theta] \approx \underbrace{f[x_i;\theta_0]}_{f_{0_i}} + \underbrace{\left(\frac{\partial}{\partial\theta}f\right)[x_i;\theta_0]}_{J_i}\ \underbrace{(\theta-\theta_0)}_{\Delta\theta} = f_{0_i} + J_i\Delta\theta$$

- Then the necessary condition for the minimum can be written as

$$\frac{\partial\Omega}{\partial\theta} = \sum_i \frac{\partial}{\partial\theta}\left(f_{0_i}^T f_{0_i} + 2f_{0_i}^T \mathrm{J}_i\Delta\theta + \Delta\theta^T J_i^T J_i\Delta\theta\right) - 2y_i^T\frac{\partial}{\partial\theta}\left(f_{0_i} + J_\mathrm{i}\Delta\theta\right) = 0$$

Linear approximation

- If the model function can be approximated by a Taylor expansion around some value for the parameter vector θ_0

$$f[x_i;\theta] \approx \underbrace{f[x_i;\theta_0]}_{f_{0_i}} + \underbrace{\left(\frac{\partial}{\partial\theta} f\right)[x_i;\theta_0]}_{J_i} \underbrace{(\theta - \theta_0)}_{\Delta\theta} = f_{0_i} + J_i \Delta\theta$$

- Then the necessary condition for the minimum can be written as

$$\frac{\partial\Omega}{\partial\theta} = \sum_i \frac{\partial}{\partial\theta}\left(f_{0_i}^T f_{0_i} + 2 f_{0_i}^T \mathrm{J}_i \Delta\theta + \Delta\theta^T J_i^T J_i \Delta\theta\right) - 2 y_i^T \frac{\partial}{\partial\theta}\left(f_{0_i} + J_\mathrm{i}\Delta\theta\right) = 0$$

- With this linear approximation it only contains quadratic and linear terms

Linear approximation

- Derivatives of linear and quadratic terms can be easily calculated, which allows to simplify the condition further to

$$\sum_i 2f_{0_i}^T J_i + 2\Delta\theta^T J_i^T J_i - 2y_i^T J_i = 0$$

- This can be re-structured to a linear equation system in $\Delta\theta$

$$\sum_i J_i^T J_i \, \Delta\theta = \sum_i J_i^T \left(y_i - f_{0_i}\right)$$

- The solution to this linear equation system can now be computed as

$$\Delta\theta = \left(\sum_i J_i^T J_i\right)^{-1} \sum_i J_i^T \left(y_i - f_{0_i}\right)$$

The learning algorithm

- Putting all this together we derived the following algorithm:

- Choose an initial parameter vector θ_0
- Iterate until convergence:
    - For each training sample $x_i \rightarrow y_i$
        - Calculate the model function f_{0_i} for x_i with parameters θ_0
        - Calculate the model function Jacobian J_i for x_i with parameters θ_0
    - Calculate the parameter update according to
$$\Delta\theta = \left(\sum_i J_i^T J_i\right)^{-1} \sum_i J_i^T \left(y_i - f_{0_i}\right)$$
    - Update the parameter vector
$$\theta_0 \coloneqq \theta_0 + \Delta\theta$$
    - If $|\Delta\theta| < \delta$ then break

Linearization

- We still have to understand how to compute the model function Jacobians J_i
- Typically the model is some black-box algorithm that we can call, taking as input some vector x and the parameters θ and computing as output some

$$y = f[x; \theta]$$

- The Jacobian matrix collects all the partial derivatives of that model function

$$J = \begin{pmatrix} \frac{\partial f}{\partial \theta_1} & \cdots & \frac{\partial f}{\partial \theta_m} \end{pmatrix}$$

- To compute it, we note that the derivative is defined as differential quotient

$$\frac{\partial f}{\partial \theta_k} = \lim_{\epsilon \to 0} \frac{f[x; \theta_1, \dots, \theta_k + \epsilon, \dots, \theta_m] - f[x; \theta]}{\epsilon}$$

Linearization algorithm

- The linearization algorithm for a black-box function f looks as follows

- We first execute the model function algorithm for the inputs x and θ (we need this anyway) and obtain

$$f_0 = f[x; \theta]$$

- Now we choose a small value ϵ (e.g. $\epsilon = 10^{-6}$)
- Then for each component θ_k of the parameter vector $\theta = [\theta_1, \dots, \theta_m]^T$
    - We execute the model function again, but instead of θ_k we change this single component slightly to $\theta_k + \epsilon$ and obtain a result

$$f_k = f[x; \theta_1, \dots, \theta_k + \epsilon, \dots, \theta_m]$$

    - The column of the Jacobian matrix is then calculated as

$$\frac{\partial f}{\partial \theta_k} = \frac{f_k - f_0}{\epsilon}$$

Regularisation

- The central formula is to compute the parameter update

$$\Delta\theta = \left(\sum_i J_i^T J_i\right)^{-1} \sum_i J_i^T \left(y_i - f_{0_i}\right)$$

- It requires the computation of the inverse of the normal equation matrix

$$\mathrm{N} = \left(\sum_i J_i^T J_i\right)$$

- How can we ensure that the matrix N is not singular?
- This singularity occurs when two parameters are linear dependent
- Because we consider the model function a black-box, there is not much we can do about this

Regularisation

- Idea: we could add an additional model forcing the parameter update to be small, i.e. $\lambda\Delta\theta \approx 0$
- This assumption is already linear, the Jacobian being

$$J = \lambda I$$

- The normal equation matrix is then augmented with this regularisation term

$$\mathrm{N} = \left(\lambda^2 I + \sum_i J_i^T J_i\right)$$

- This will slow down convergence, but it ensures that the matrix N can be inverted

Stochastic model

- We have derived a least-squares method minimising

$$\Omega = \frac{1}{2}\sum_i r_i^T r_i$$

- We could interpret the residuals as being Normal distributed random variables reflecting a noise distribution overlaying the model
- In this case their likelihood probability would be proportional to

$$p[y_i|\theta] \sim \exp\left[-\sum_i r_i^T \Sigma^{-1} r_i]\right]$$

- If we normalise our observation so that $\Sigma = \sigma_0^2 \mathrm{I}$, then maximising this log-likelihood is exactly the same as the procedure we derived
- Therefore, the least-squares estimate is equivalent to the maximum likelihood estimate under these assumptions

Stochastic model

- Under these assumptions of the stochastic model we can not only estimate the parameter vector $\hat{\theta}$, but also its covariance matrix
- The common factor for the covariance of the observation $\sigma_0^2 I$ can be estimated from the redundancy, i.e. the difference between number of observations and number of unknowns, and the residuals of the optimised parameter $\hat{\theta}$as

$$\hat{\sigma}_0^2 = \frac{1}{N-U} \sum_i r_i^T r_i$$

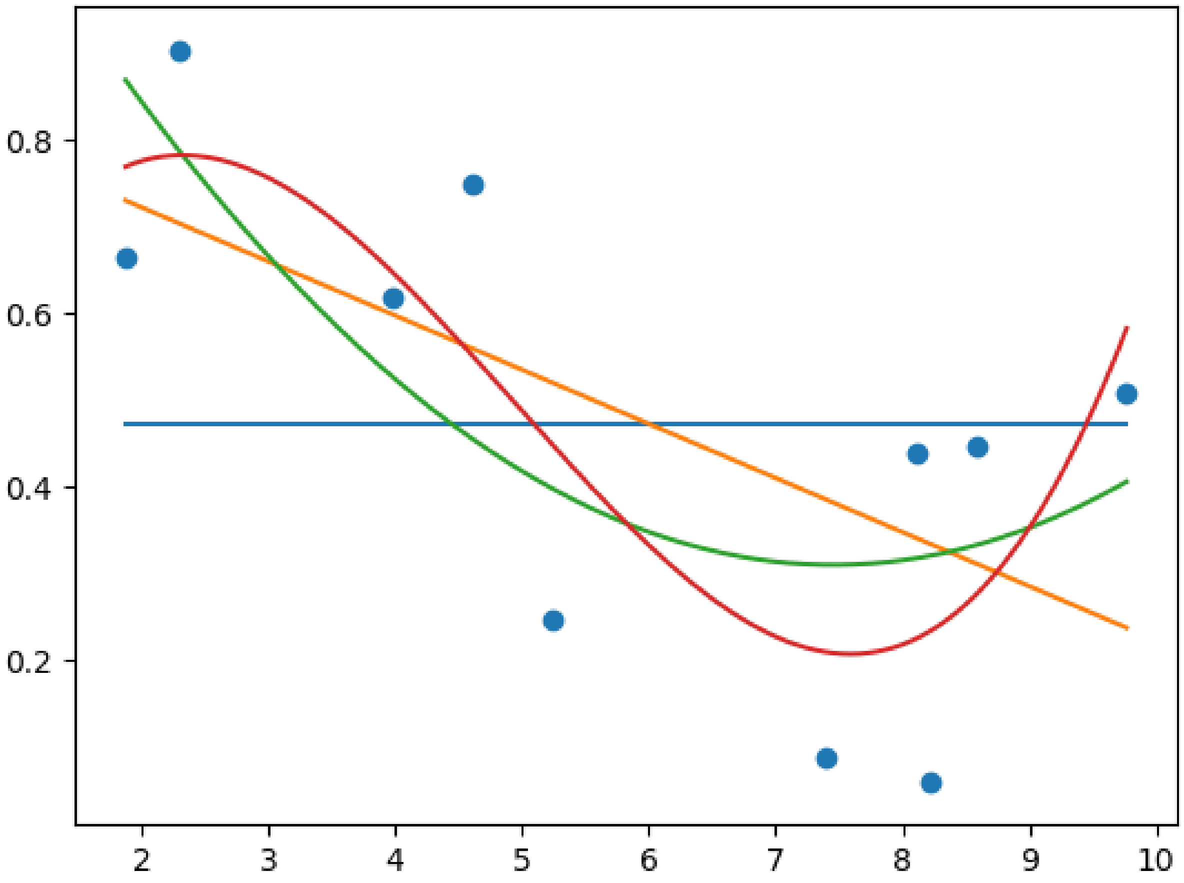
- This allows to estimate the covariance matrix of the parameter vector through linear error propagation as

$$\hat{\Sigma}_{\hat{\theta}\hat{\theta}} = \hat{\sigma}_0^2 \left(\sum_i J_i^T J_i \right)^{-1}$$

- This quantity describes the accuracy of our parameter vector estimate

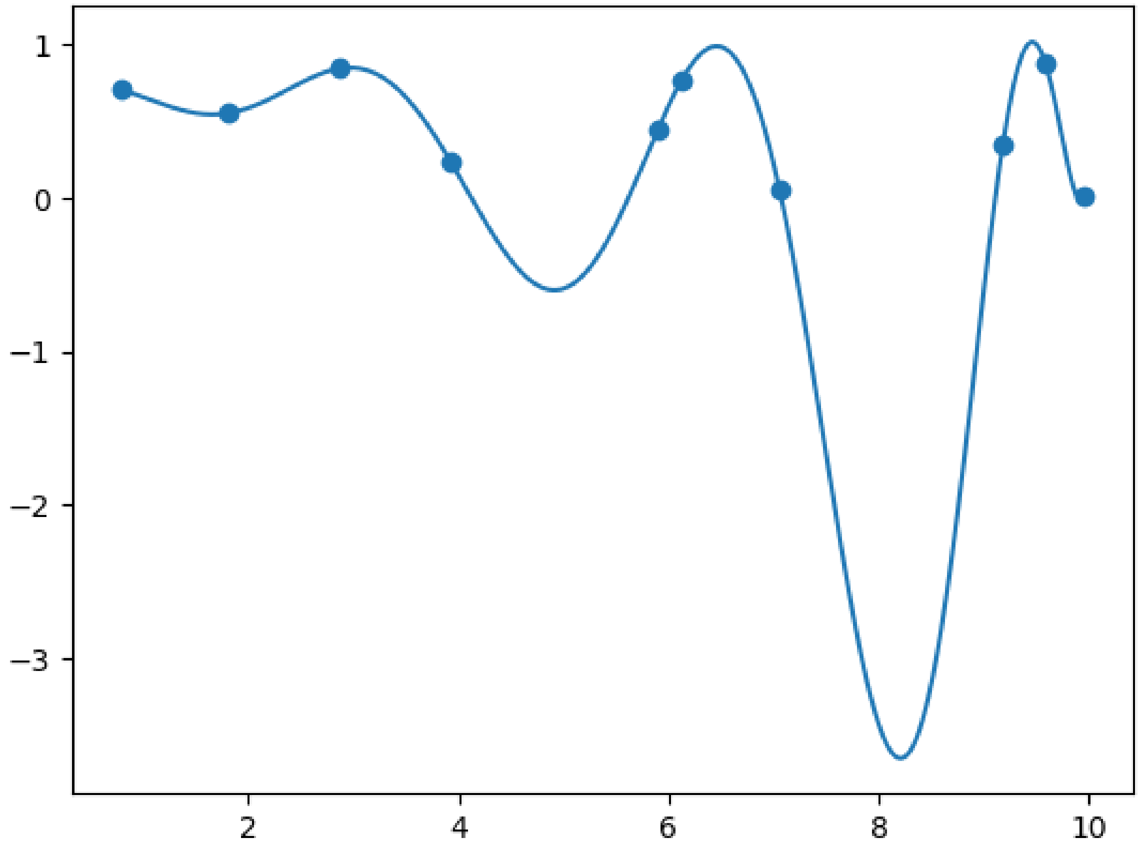
Polynomial regression for random points

With increasing number of parameters the model tries to better and better track the distribution of the training samples



Overfitting

As always, if the number of parameters is chosen too large, the model will perfectly fit the training data, but is increasingly unlikely generalise well to new data



Thank you for your attention