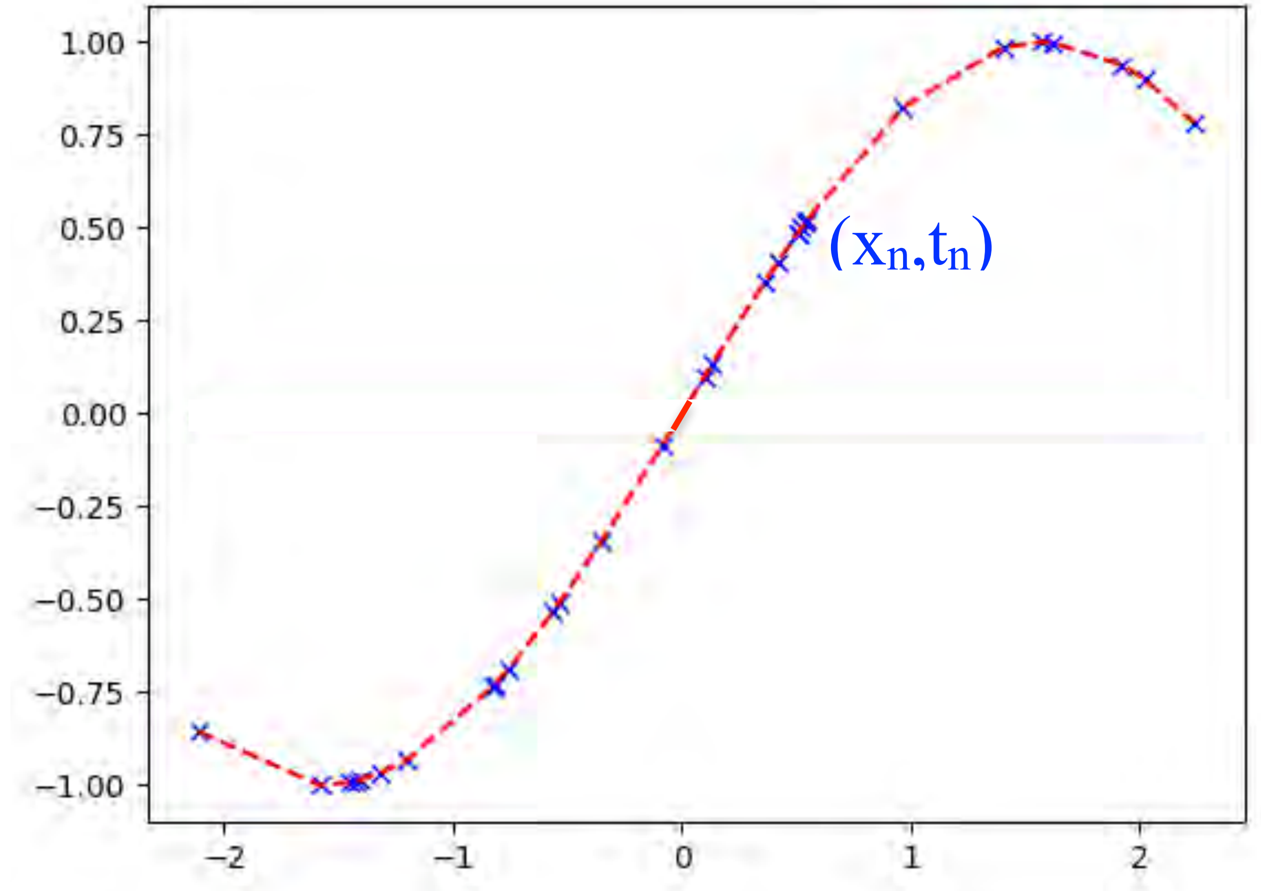
Non Linear Regression

Pascal Fua
IC-CVLab

Reminder: Polynomial Regression



For $1 \leq n \leq N$:

$$t_n = f(x_n) + \epsilon$$

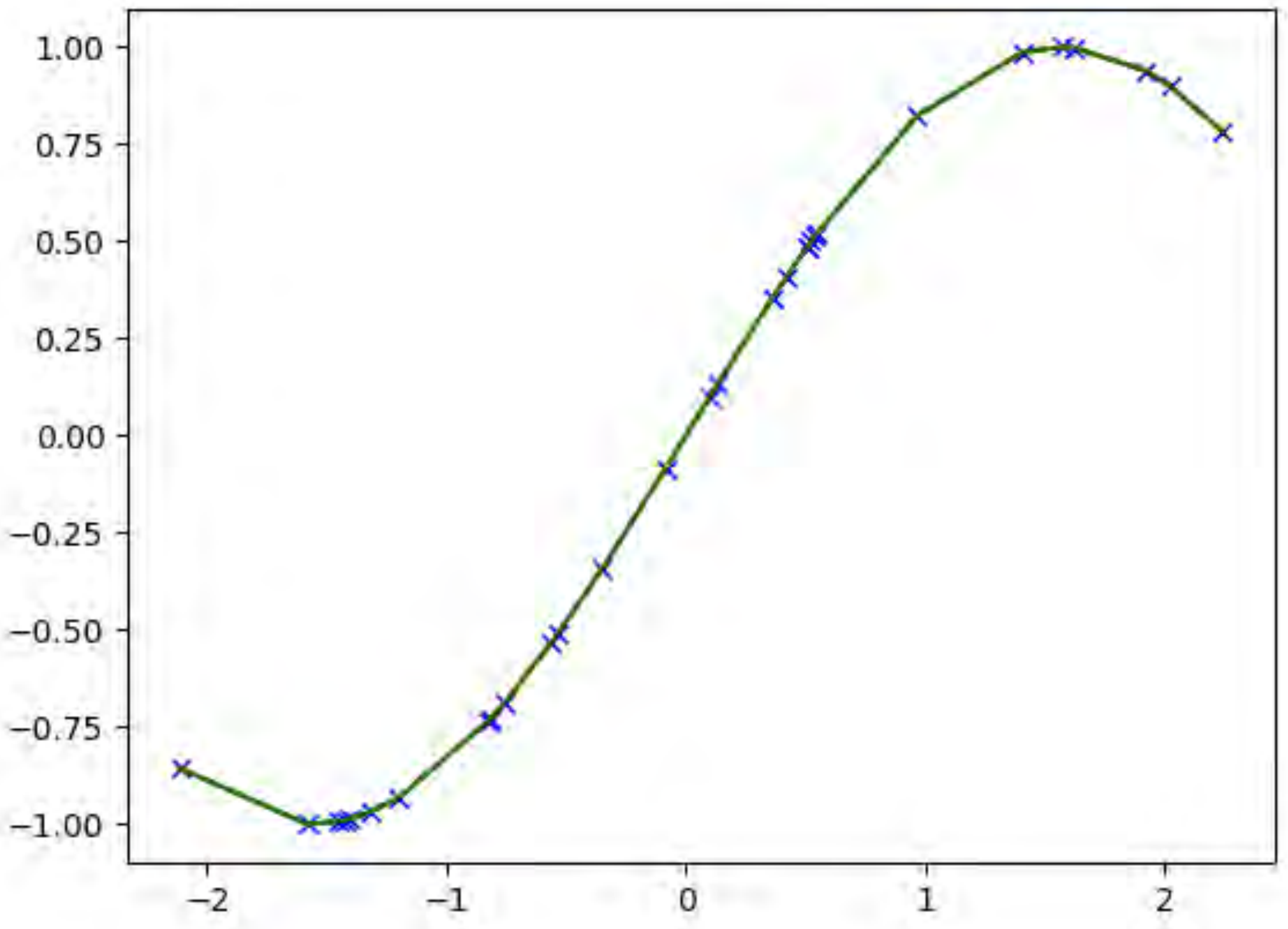
- The (x_i,t_i) are given.
- f is unknown.

- Find $\mathbf{w} = [w_0, w_1, \ldots, w_M]$ such that: $\forall x, f(x) \approx \sum_{i=0}^{M} w_i x^i$

- Least squares solution: $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \sum_n (t_n - \sum_{i=0}^{M} w_i x_n^i)^2$

- For M=1, reduces to linear regression.

Reminder: Polynomial Approximation



M=0

M=1

M=2

M=3

M=4

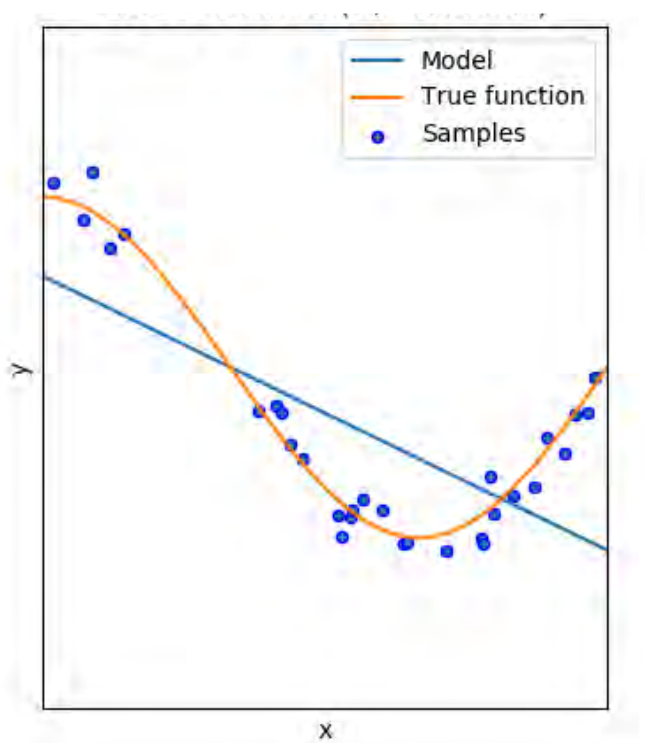
M=6

M=10

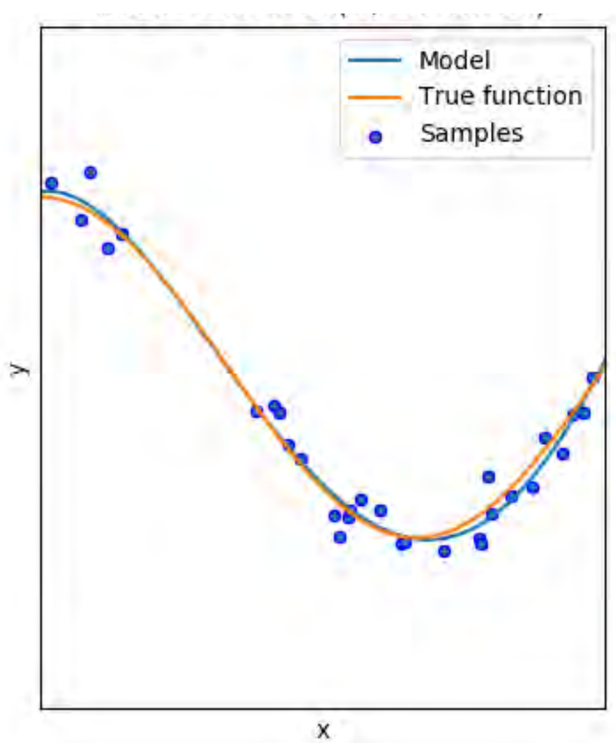
For a given M, we plot in green:

$$f_M(x) = \sum_{i=0}^{M} w_i^* x^i$$

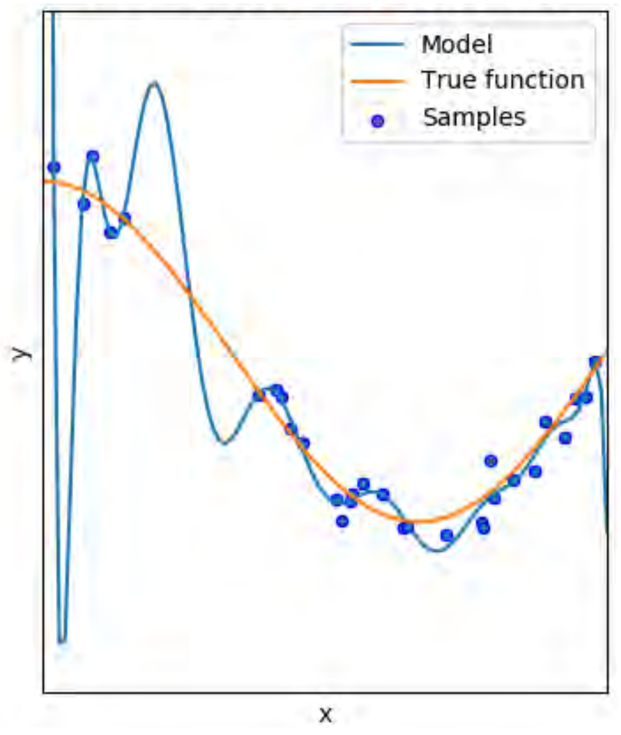
From Simple to Complex



Order 1



Order 4



Order 15

The trick is to find the best compromise between simplicity and goodness of fit.

1D Polynomial Feature Expansion

$$x \rightarrow \phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^M \end{bmatrix}$$

The polynomial can be rewritten as:

$$\sum_{i=0}^{M} w_i x^i = \mathbf{w} \cdot \phi(x) = \mathbf{w}^T \phi(x) \text{ with } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_M \end{bmatrix}$$

The least squares solution becomes:

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \sum_n (t_n - \mathbf{w}^T \phi(x_n))^2$$

Solving a Linear System

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_{n=1}^{N} (t_n - \mathbf{w}^T \phi(\mathbf{x}_n))^2$$
$$= \arg\min_{\mathbf{w}} \|\Phi \mathbf{w} - \mathbf{t}\|^2$$

with

$$\Phi = \begin{bmatrix} \phi(\mathbf{x}_1)^T \\ \phi(\mathbf{x}_2)^T \\ \vdots \\ \phi(\mathbf{x}_N)^T \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^M \\ 1 & x_2 & x_2^2 & \dots & x_2^M \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_N & x_N^2 & \dots & x_N^M \end{bmatrix}, \ \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \dots \\ w_M \end{bmatrix}, \ \text{ and } \mathbf{t} = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \\ \dots \\ t_N \end{bmatrix}.$$

Intuitively: $\Rightarrow \underset{N \times \tilde{M}}{\Phi} \underset{\tilde{M} \times 1}{\mathbf{w}^*} \approx \underset{N \times 1}{\mathbf{t}}$

Formally: $\Rightarrow \underset{\tilde{M} \times \tilde{M}}{(\Phi^T \Phi)} \underset{\tilde{M} \times 1}{\mathbf{w}^*} = \underset{\tilde{M} \times 1}{\Phi^T \mathbf{t}}$

Reminder: Proof Sketch

We want to minimize:

$$\begin{aligned} R &= \frac{1}{2}\|\Phi\mathbf{w} - \mathbf{t}\|^2 \\ &= \frac{1}{2}(\Phi\mathbf{w} - \mathbf{t})^T(\Phi\mathbf{w} - \mathbf{t}) \end{aligned}$$

The gradient or R w.r.t **w** is:

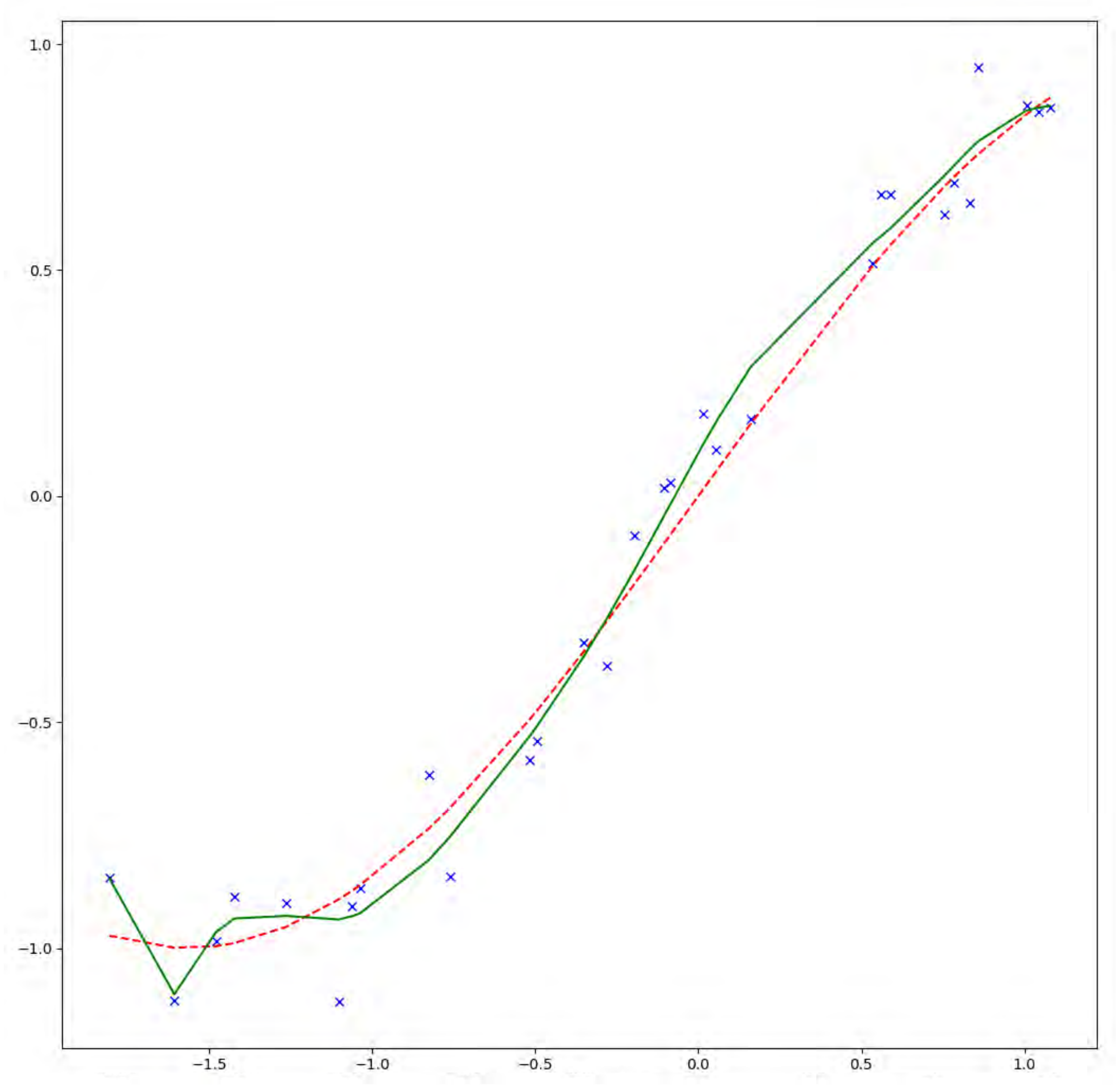
$$\nabla R = \Phi^T(\Phi\mathbf{w} - \mathbf{t})$$

At the minimum:

$$0 = \nabla R = \Phi^T(\Phi\mathbf{w} - \mathbf{t})$$

$$\Rightarrow \Phi^T\Phi\mathbf{w} = \Phi^T\mathbf{t}$$

Adding Noise



M=10

Regularization

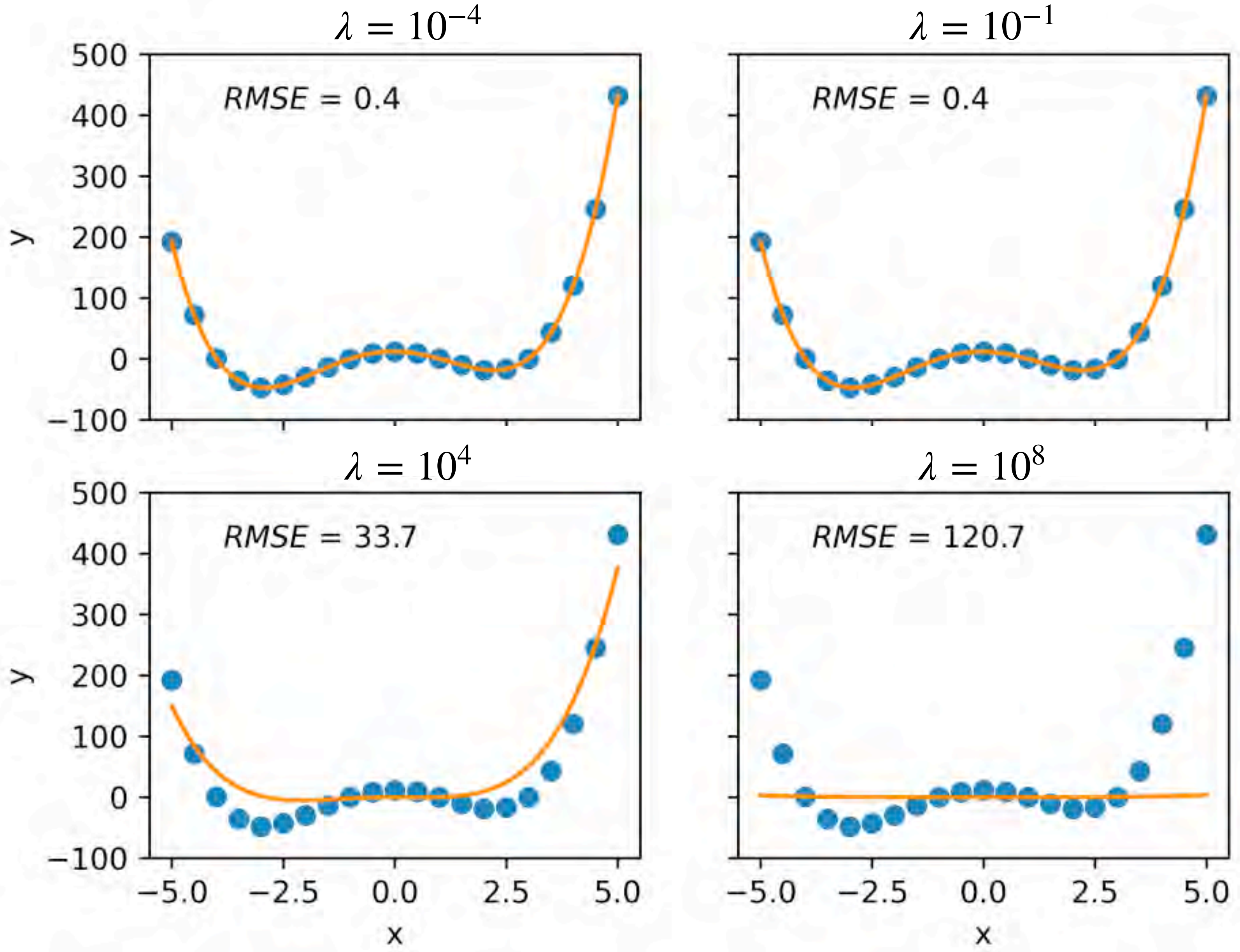
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\Phi\mathbf{w} - \mathbf{t}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

$$\Rightarrow \text{ Solve: } (\Phi^T\Phi + \lambda\mathbf{I})\mathbf{w} = \Phi^T\mathbf{t}$$

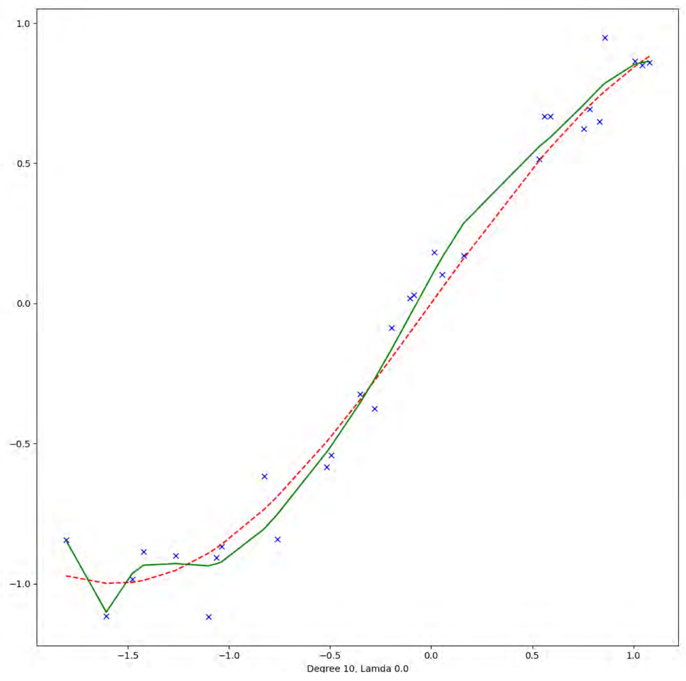
- This is known as weight decay because in iterative algorithms it encourages the weight values to decay to zero, unless supported by the data.
- It discourages large weights and therefore quick variations.

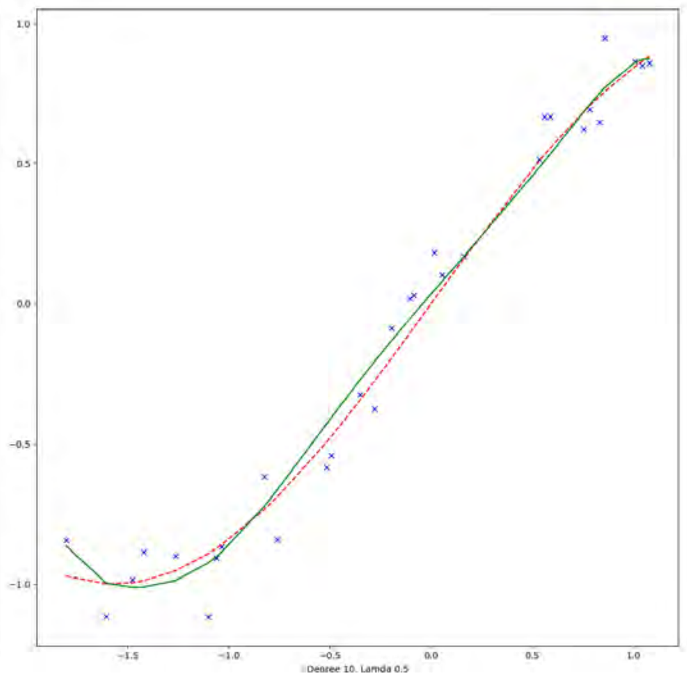
EPFL

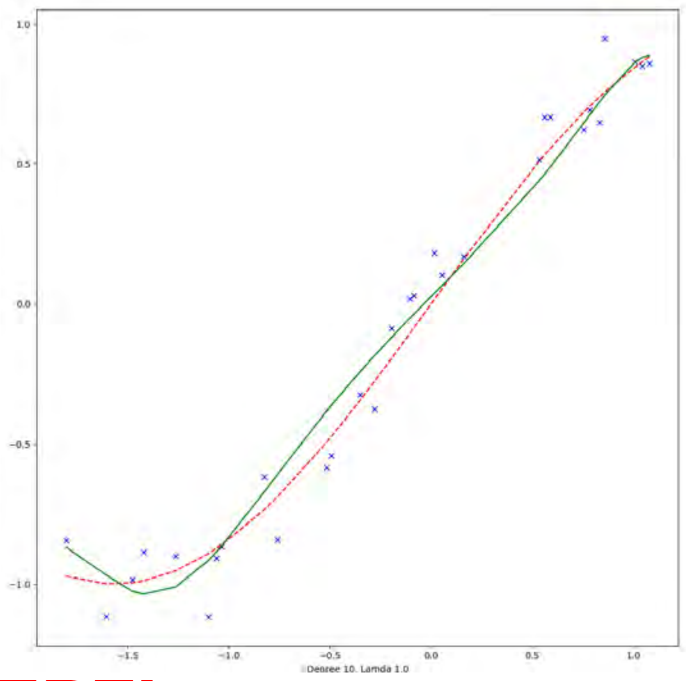
Increasing λ without Noise

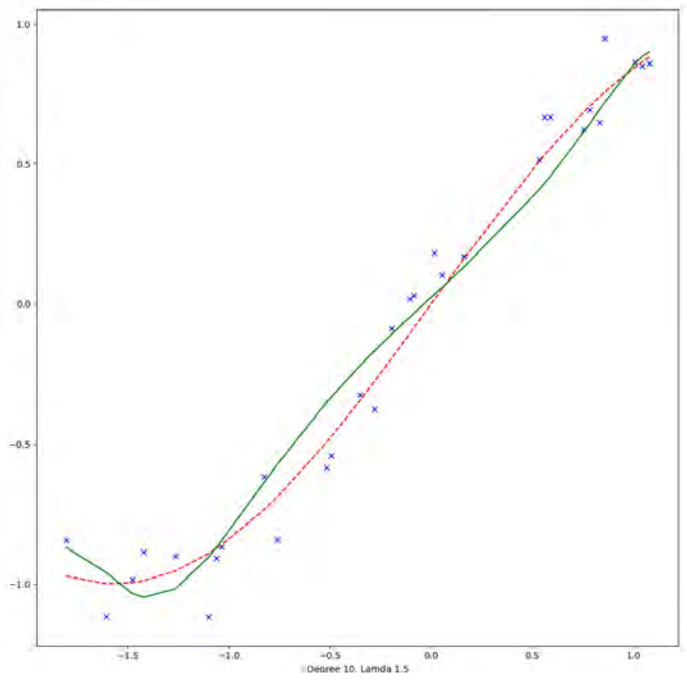


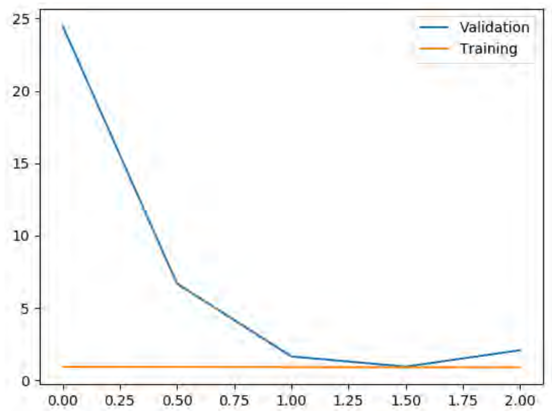
Increasing λ with Noise











Use cross-validation data to select the value of λ.

Into Higher Dimensions

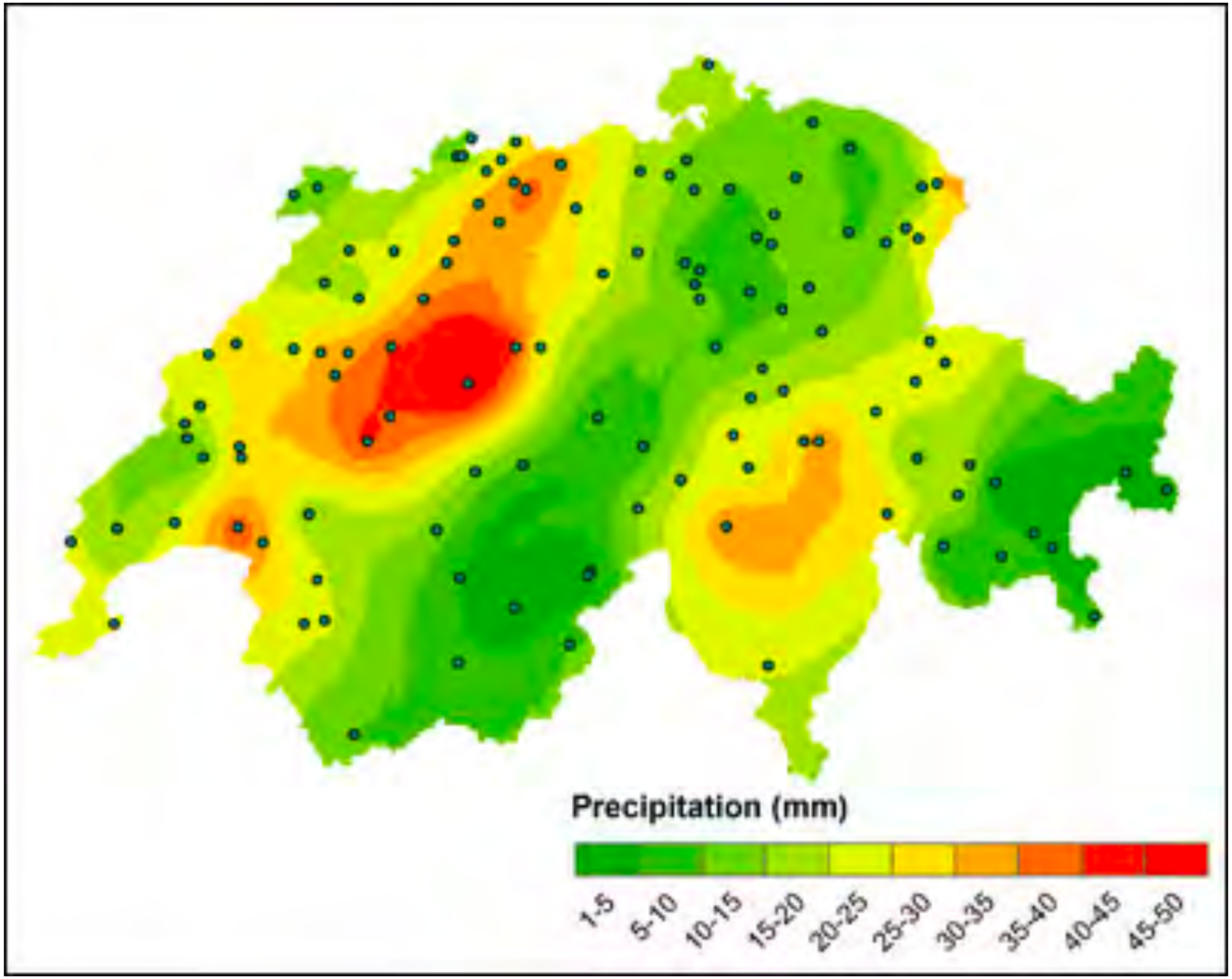
- Let $\{(\mathbf{x}_n \in \mathbb{R}^d, \mathbf{t}_n \in \mathbb{R}^D)_{1 \leq n \leq N}\}$ be N training pairs.
- Let ϕ be a function from $\mathbb{R}^d$ to $\mathbb{R}^M$.
- Let y be the function $\mathbf{x} \in \mathbb{R}^d \rightarrow y(\mathbf{x}) = \mathbf{W}^t \phi(\mathbf{x})$, where $\mathbf{W}$ is an $M \times D$ matrix.
- We seek to minimize

$$E(\mathbf{W}) = \frac{1}{2} \Sigma_{n=1}^{N} || \mathbf{W}^t \phi(\mathbf{x}_n) - t_n ||^2 + \frac{\lambda}{2} || \mathbf{W} ||^2$$

Data term

Regularization

Weather in Switzerland



The circles represent actual measurements

- Rain only:

$$d = 2$$
$$D = 1$$

- Rain
- Temperature
- Wind
- …

$$d = 2$$
$$D > 1$$

Polynomial Expansion

- Let $\{(\mathbf{x}_n \in \mathbb{R}^d, \mathbf{t}_n \in \mathbb{R}^D)_{1 \leq n \leq N}\}$ be a set N pairs.
- We seek to minimize

$$E(\mathbf{W}) = \frac{1}{2}\Sigma_{n=1}^{N} || \mathbf{W}^t \phi(\mathbf{x}_n) - t_n ||^2 + \frac{\lambda}{2} || \mathbf{W} ||^2$$

- $\phi(\mathbf{x})$ can be the polynomial expansion

$$\left[1, x_1, \ldots, x_d, x_1^2, \ldots, x_d^2, x_1^3, \ldots, x_d^3, \ldots, x_1 x_2, \ldots, x_1 x_d, \ldots, x_{d-1} x_d, x_1^2 x_2, \ldots\right]^t \quad M \times 1$$

- The least-squares solution satisfies

$$(\Phi\Phi^t + \lambda \mathbf{I})\mathbf{W}^* = \Phi \mathbf{T}$$

$$\Phi = [\phi(\mathbf{x}_1) | \ldots | \phi(\mathbf{x}_n)] \quad M \times N$$

$$\mathbf{T} = [\mathbf{t_1} | \ldots | \mathbf{t_n}]^t \quad N \times D$$

- The computational complexity is in $O(M^3)$ because $\Phi\Phi^t$ is of size $M \times M$.

—> Let's get rid of ϕ !

Kernel Trick

- Introduce dual variables

$$a_n = \frac{1}{\lambda}\{\mathbf{W}^T\phi(\mathbf{x}_n) - \mathbf{t}_n\}$$

- At the minimum

$$[a_1 | \ldots | a_N]^t = (\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{T}$$

$$K_{n,m} = \phi(\mathbf{x}_n)^T\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- The regressor becomes

$$\boxed{y(\mathbf{x}) = \mathbf{k}(\mathbf{x})(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{T}}$$

with

$$\mathbf{k}(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_N)]^t$$

$$k(\mathbf{x}, \mathbf{x}') = \theta_0(exp(-(\frac{\theta_1}{2}||\mathbf{x} - \mathbf{x}'||^2) + \theta_2 + \theta_3\mathbf{x}^t\mathbf{x}')$$

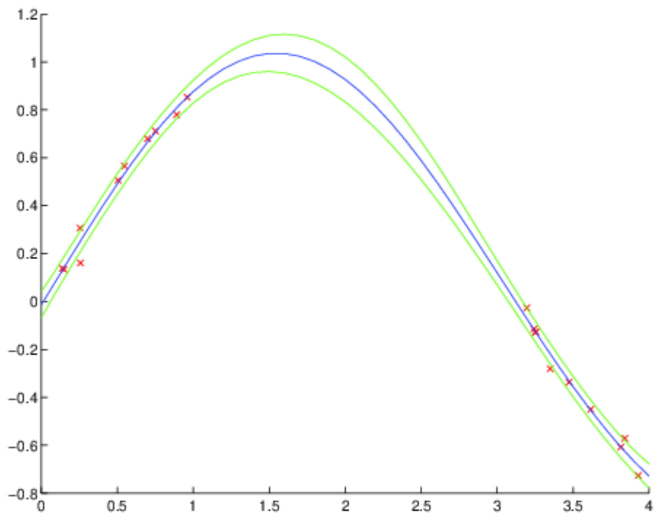
—>ϕ is never explicitly computed

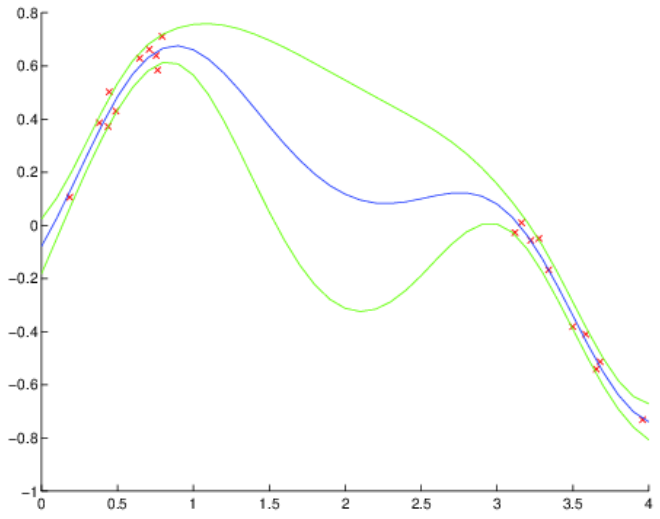
Kernel Ridge Regression

$$y(\mathbf{x}) = \mathbf{k}(\mathbf{x})(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{T}$$

- ϕ is never explicitly evaluated.
- $\mathbf{k}$ can be understood as a vector of distances to the training samples.
- Using $\mathbf{k}$ is tantamount to making the dimension of ϕ infinite.
- Complexity in $O(N^3)$ where N is the number of samples.
- Can be used to evaluate not only predictions but also uncertainties.

➡ Extremely effective when N is small.





Curse of Dimensionality

$$\mathbf{k}(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_N)]^t$$

$$k(\mathbf{x}, \mathbf{x}') = \theta_0(exp(- (\frac{\theta_1}{2}||\mathbf{x} - \mathbf{x}'||^2)) + \theta_2 + \theta_3\mathbf{x}^t\mathbf{x}')$$

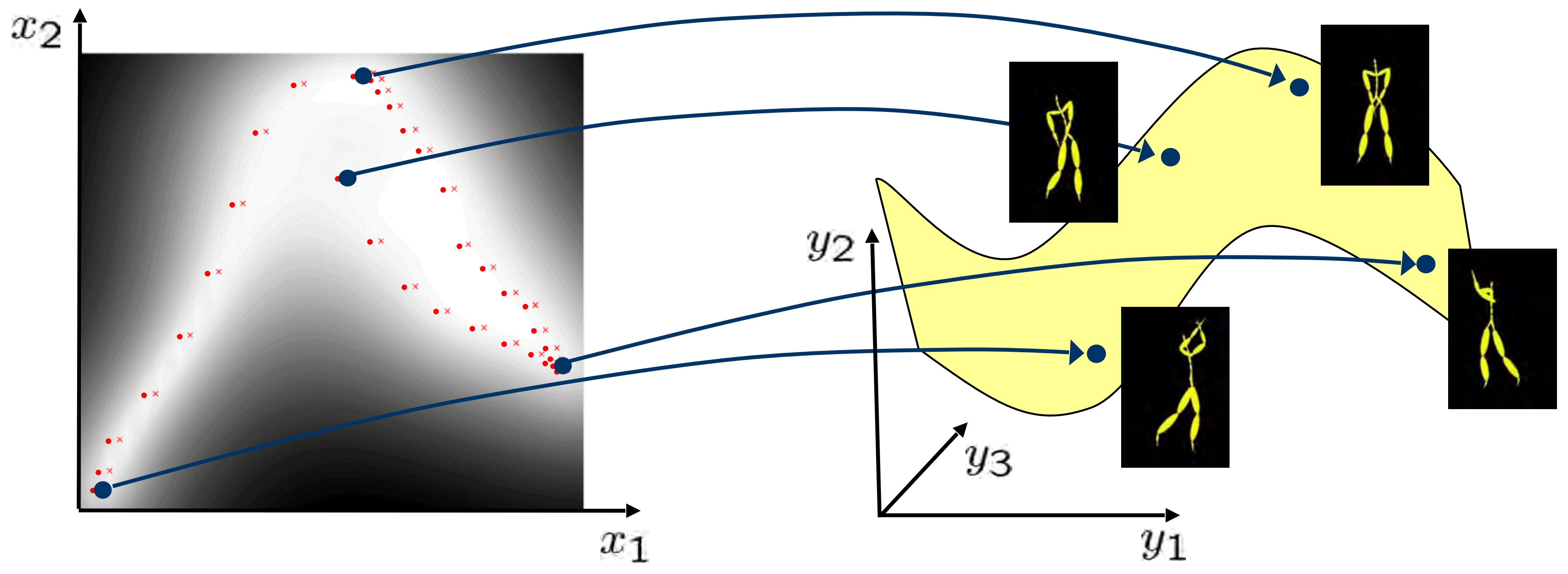
- In high dimensional spaces, the Euclidean distance stops being meaningful.
- For dimensions D > 40, KRR tends to lose some of its effectiveness.
- A similar problem occurs with kNNs.

Optional: Tracking Golf Swings

Can we recover the 3D pose from the positon of the joints?

[Urtasun et al., ICCV'05]

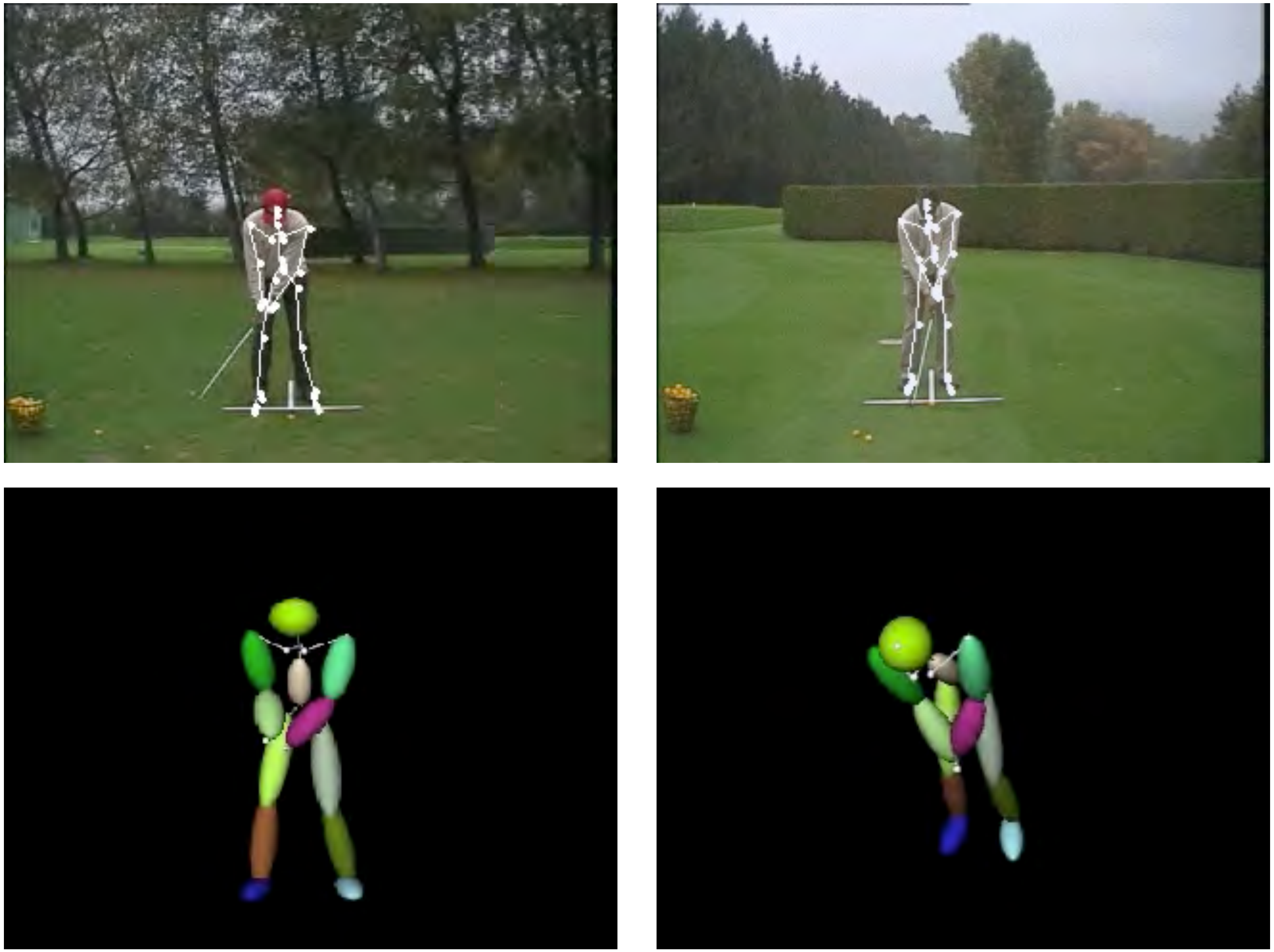
Optional: Latent Space of Swings



Latent Space ($\mathbf{X}$)   Pose Space ($\mathbf{Y}$)

- Use KRR to map from a 2D space to full body poses
- Fit the 2D model to image data

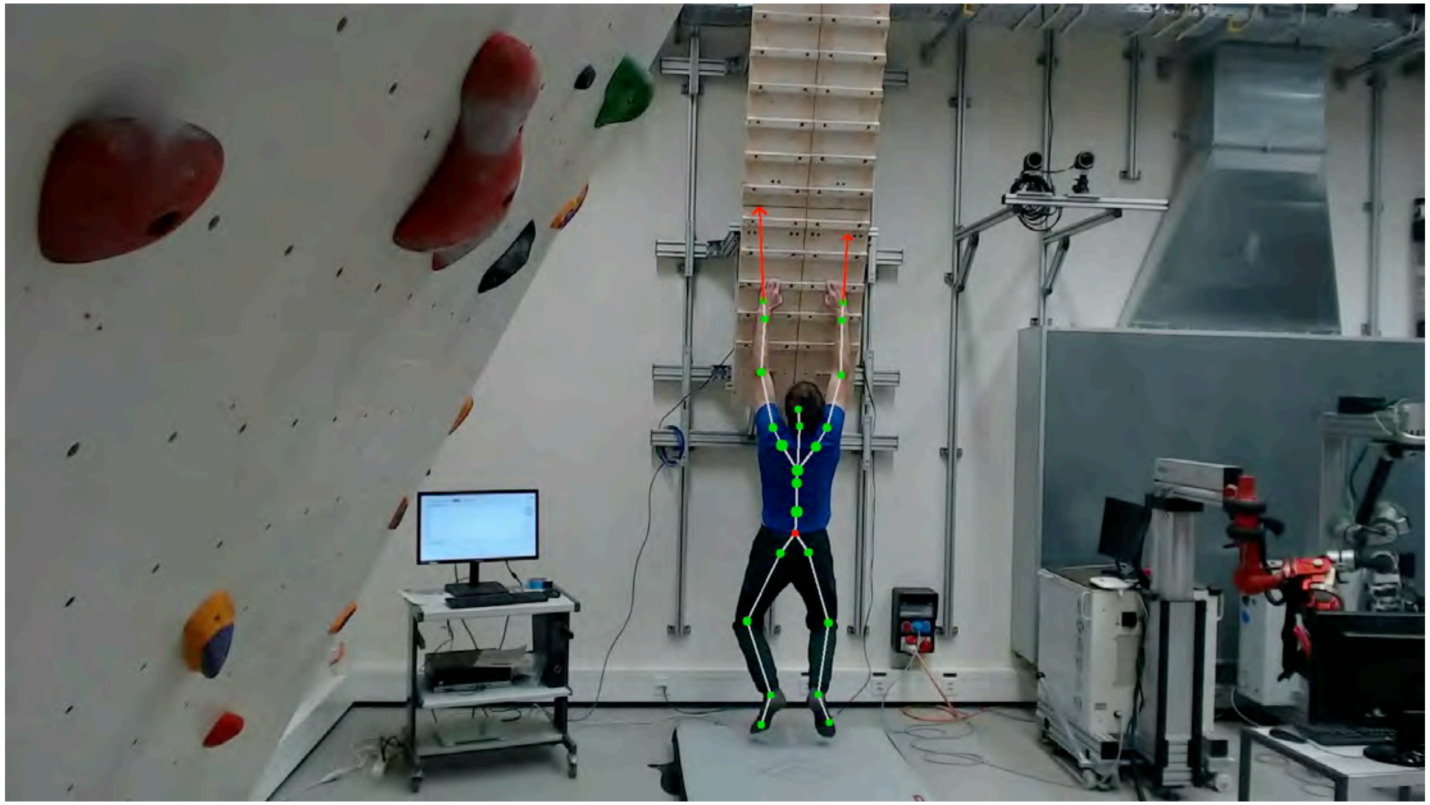
Optional: 3D Golf Swings



[Urtasun et al., ICCV’05]

Optional: KRR vs Deep Networks

Red arrows: Estimated forces exerted by the climber

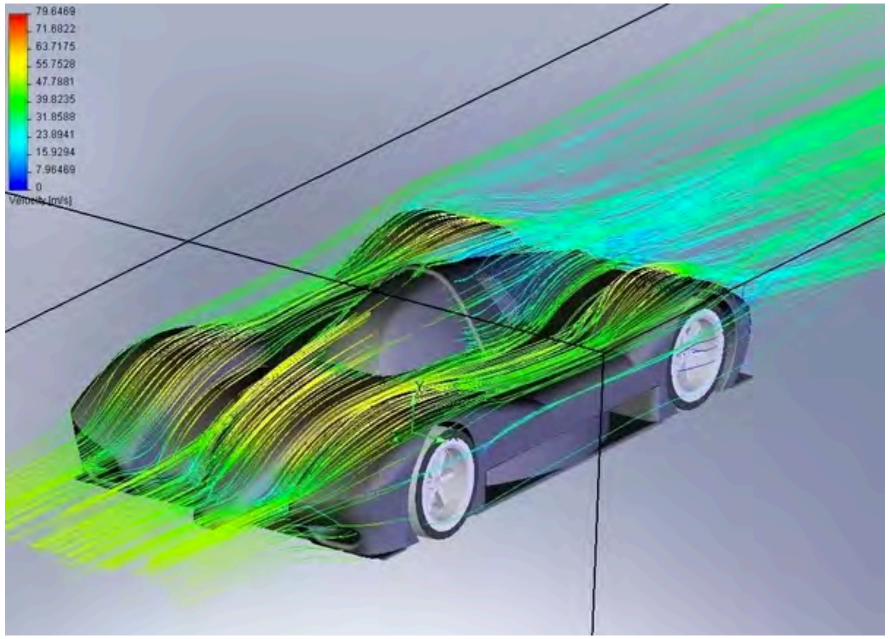


- Now we tend to use deep networks for 3D pose estimation.
- Deep Nets are not affected by the curse of dimensionality but require large training sets.

We will get back to that.

Optional: KRR for 3D Shape Design

- Design a shape.
- Simulate its performance.
- Redesign.



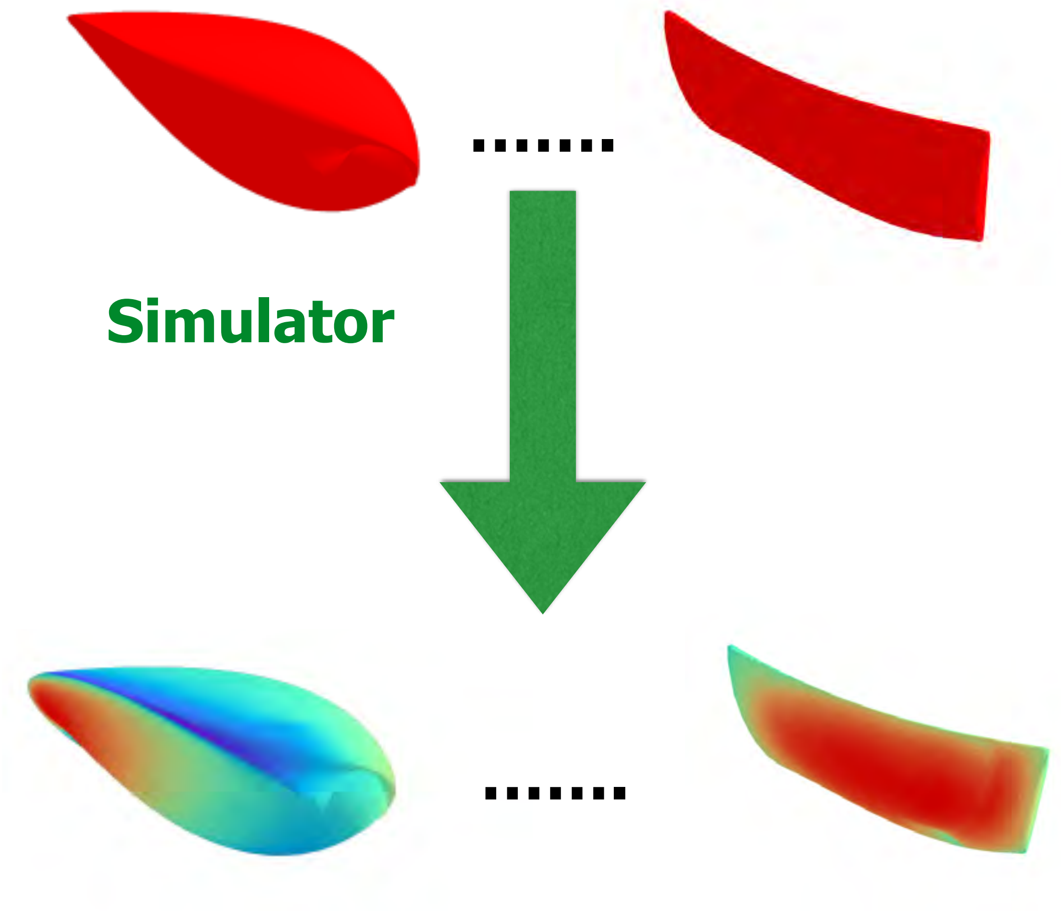
It works but:

It takes hours or days to produce a single simulation.

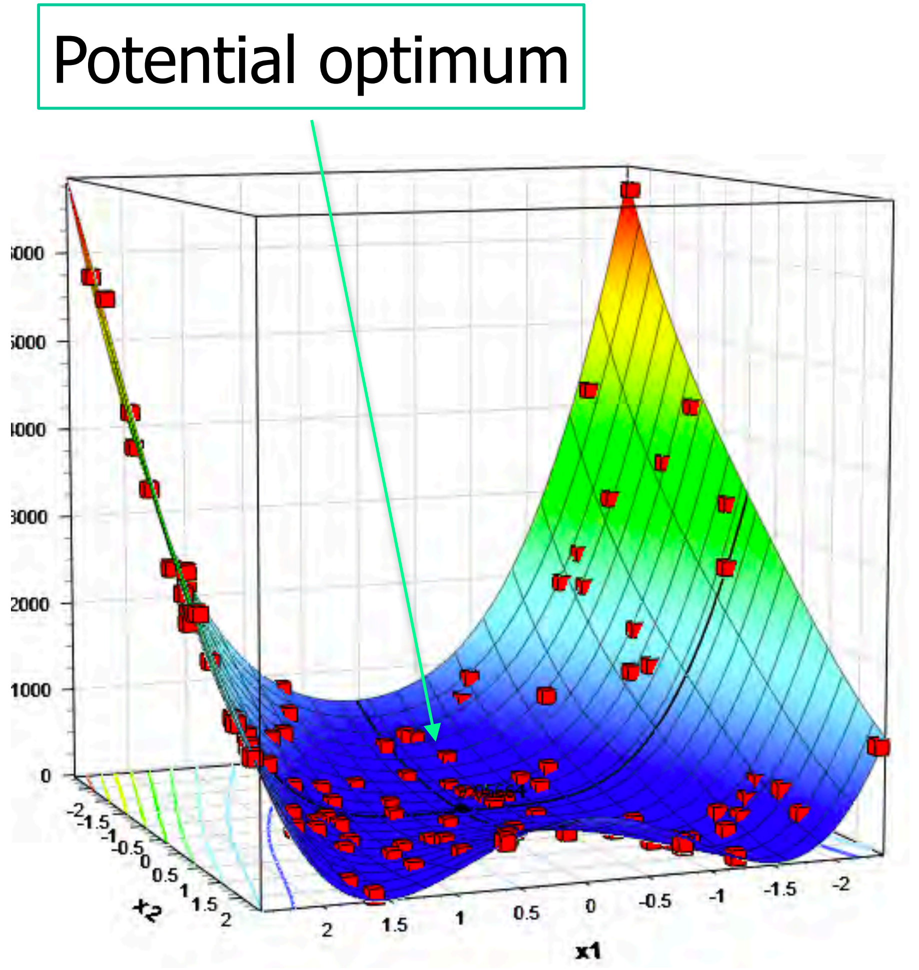
This constitutes a serious bottleneck in the exploration of the design space.

Designs are limited by humans' cognitive biases.

Optional: Surrogate Models

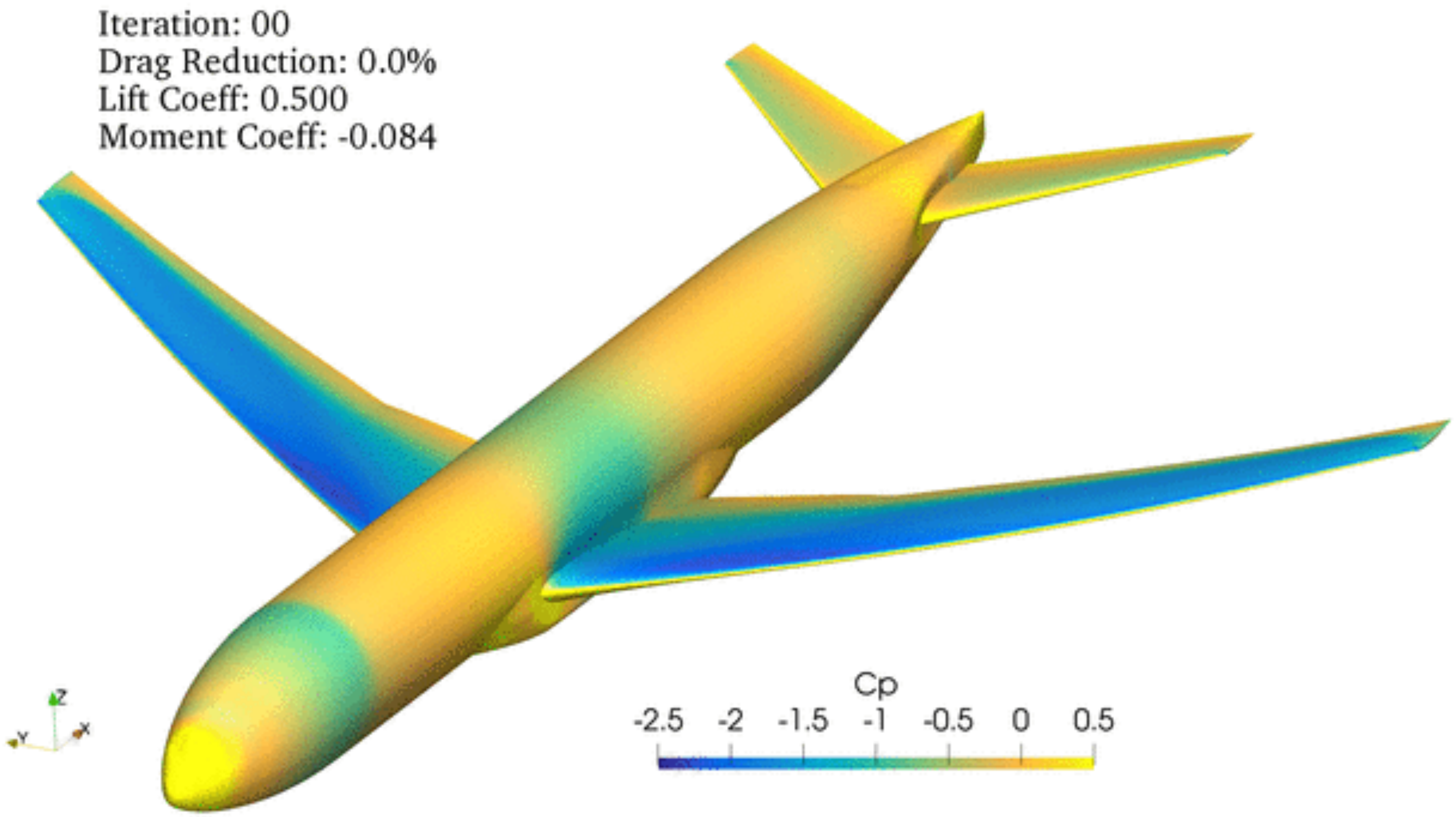


- Drag
- Pressure Coefficients
- Boundary Layer Velocities
- …



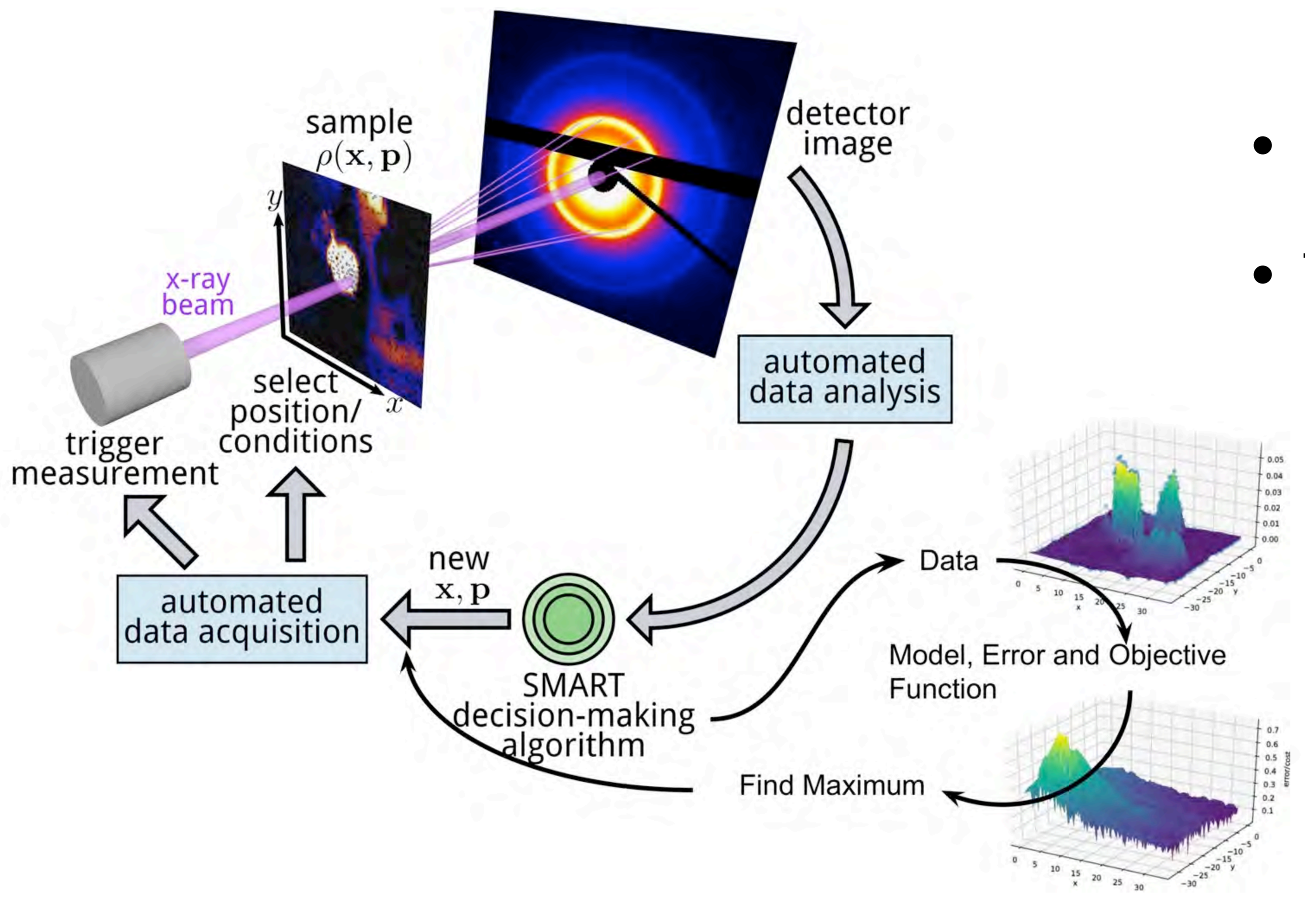
The response surface is approximated use Kernel Ridge Regression.

Optional: Aerodynamic Optimization

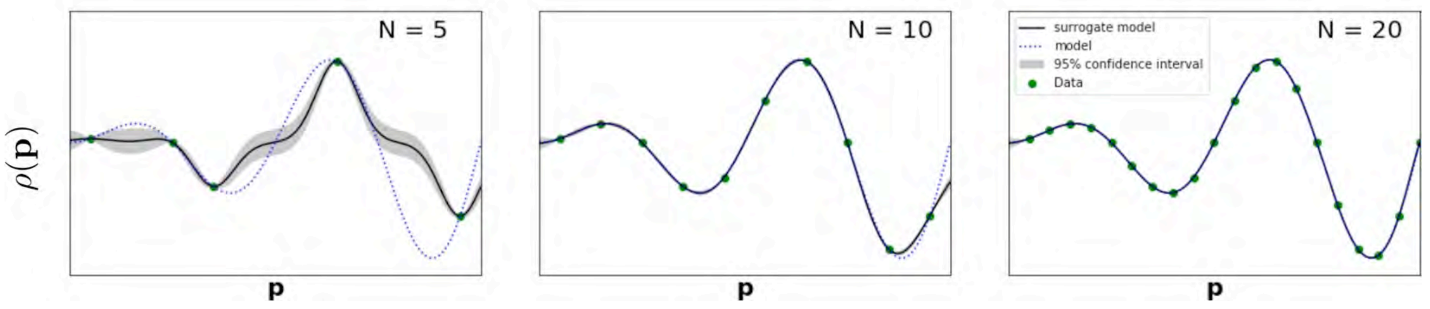


He, AIAA'19

Optional: Autonomous X-Ray Scattering

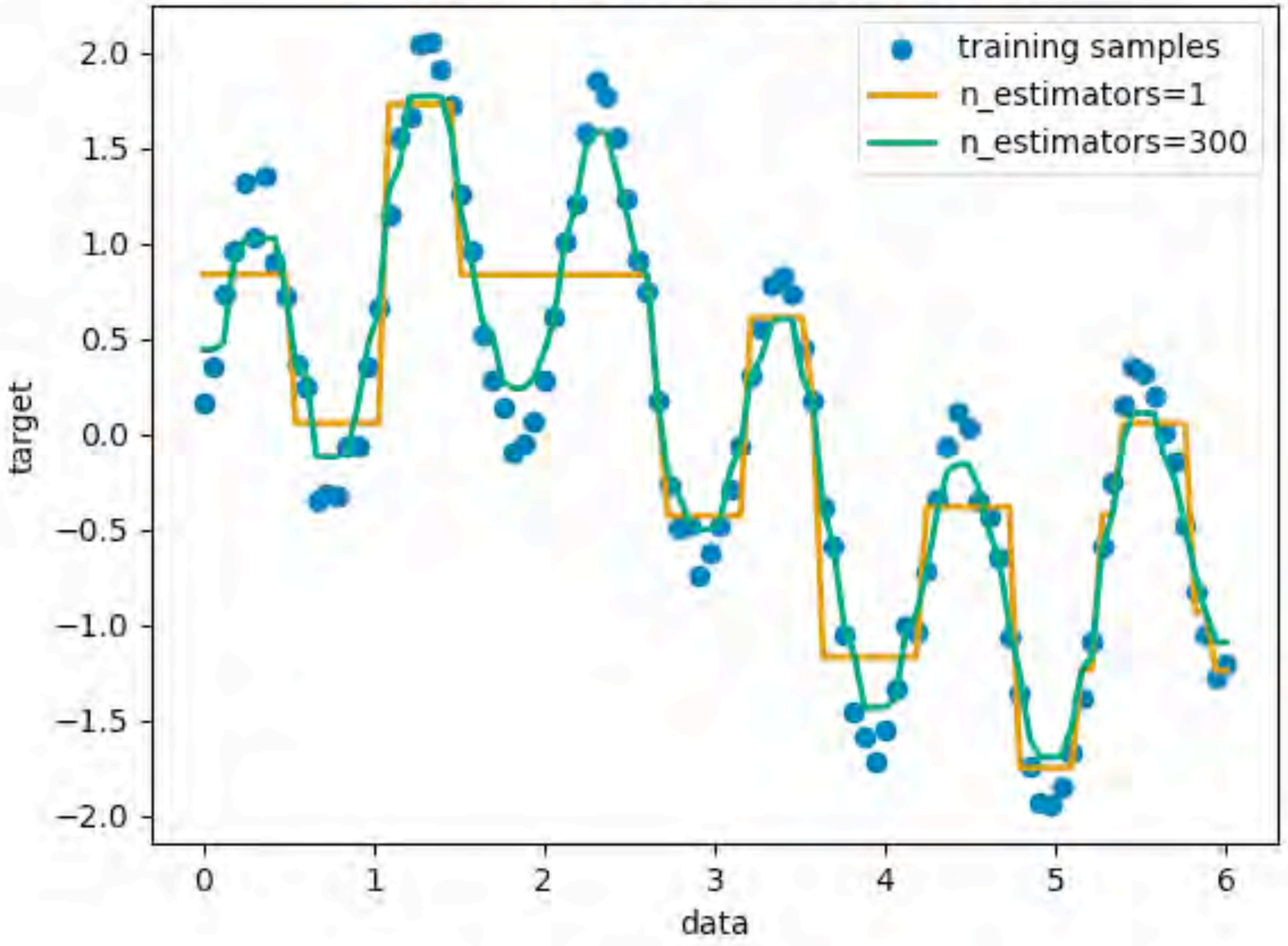

sample
$\rho(\mathbf{x}, \mathbf{p})$
detector image
x-ray beam
select position/ conditions
trigger measurement
automated data analysis
automated data acquisition
new $\mathbf{x}, \mathbf{p}$
SMART decision-making algorithm
Data
Model, Error and Objective Function
Find Maximum

- KRR is used to interpolate the measurements.
- The bean is then targeted at areas of large uncertainty.



Noack et al. Scientific Reports'20

Reminder: Boosted Decision Tree Regression



Piecewise constant predictions

EPFL

KRR vs Trees

Kernel Ridge Regression

Strengths:

- Yields smooth, continuous predictions
- Provides uncertainty estimates
- Naturally handles multi-output problems
- Strong theoretical foundations in statistical learning theory
- There is a closed-form solution

Weaknesses:

- Scales poorly with dataset size ($O(N^3)$ training complexity)
- Memory intensive for large datasets (large kernel matrix)
- Hyperparameter tuning can be challenging
- Can struggle with very high-dimensional data
- Not ideal for categorical features unless properly encoded
- Can overfit if regularization parameter isn't properly tuned

When to use:

- Small to medium dataset
- Dataset has more features than samples
- Uncertainty estimates are needed.

Gradient Boosted Trees

Strengths:

- Predictive performance on structured/tabular data
- Handles mixed data types naturally
- Works well with high-dimensional data
- Robust to outliers and missing values
- Can handle large datasets efficiently

Weaknesses:

- Many parameters to tune
- Produces discontinuous, piecewise constant predictions
- Not ideal for problems with smooth underlying functions
- Training is sequential (difficult to parallelize)
- May struggle with highly correlated features

When to use:

- Large dataset with mixed feature types
- It may contain outliers or missing values
- Domains like finance, marketing, or healthcare

There is no clear winner. You have to be aware of the existence of both.