Hyperparameter Optimization

28.10.2019 | Kai Krajsek | Jülich Supercomputing Centre

OUTLINE

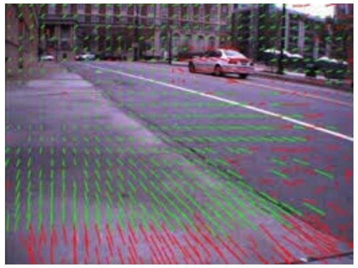
- Hyperparameters
- Cross validation
- Learning curve
- Grid search
- Random search
- Bayesian optimization
- Successive halving/Hyperband
- Challenges

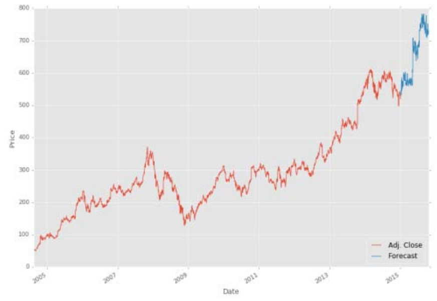
SUPERVISED LEARNING

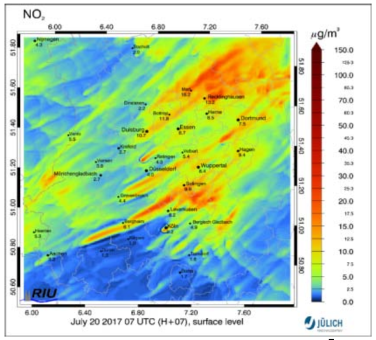
We consider supervised learning, i.e. features and labels are availabe for training

Examples:

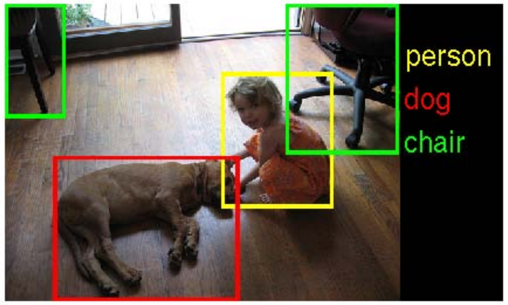
- **Regression:** Optical flow estimation, predicting stock prices, chemical weather forecasting

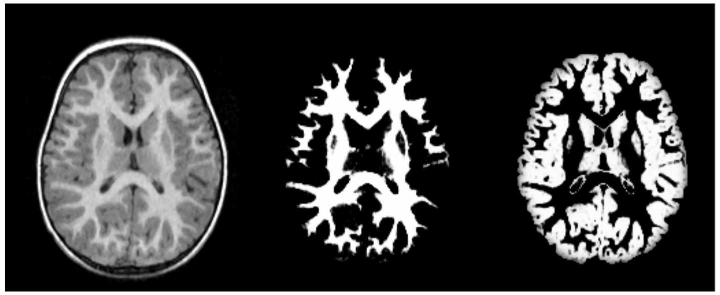






- **Classification:** Object recognition, image segmentation





SUPERVISED LEARNING

Formal definition: Find functional relationship

$$f : X \to Y$$

between infinite sets X (features) and Y (labels) based on a finite set of examples $x_i, y_i \sim p(x, y)$

➡ ill-posed problem ➡ requires additional assumptios

Function usually choosen from a parameterized set Λ, e.g.

- Linear regression: $y = ax + b, \; a, b \in \mathbb{R}$
- Neural networks: $y = g_N(g_{N-1}((...g_1(x))))$

$$g_i(x) = \sigma_i(A_i x + b_i),\, A_i \in \mathbb{R}^{m_i \times n_i},\, b_i \in \mathbb{R}^{n_i}$$

$$\sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$$

SUPERVISED LEARNING

Learning as an optimization problem:

$$\hat{f} = \arg\min_{f \in \Lambda} R(f)$$

Expected risk:

$$R(f) := \int L(y, f(x))p(x, y)dxdy$$

Loss function

Only finite number of samples $x_i, y_i \sim p(x, y)$ are availabe

➡ Approximation of $R(f)$ required

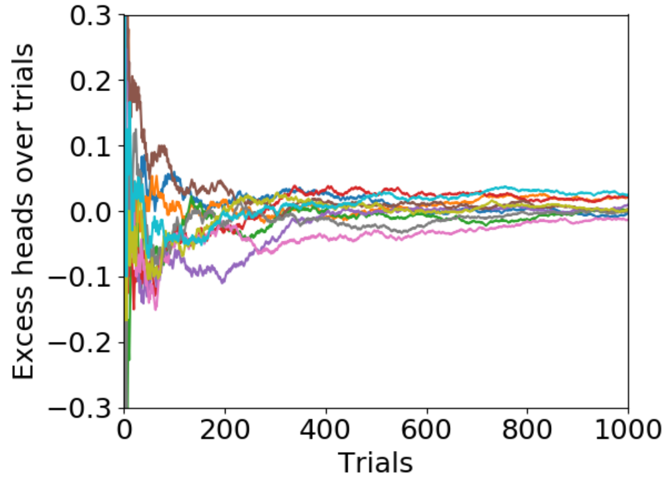
LAW OF LARGE NUMBERS

The law of large numbers indicates to **estimate** the expected risk by the **average** of individual losses

Emphirical risk

$$\hat{R}_N(f) := \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) \qquad R(f) = \lim_{N\to\infty} \hat{R}_N(f)$$

Example: Coin-toss experiment



Properties of **estimators:**

Bias:

$$\mathbb{B}(\hat{R}) := \mathbb{E}\left[\hat{R}(f)\right] - R(f)$$

Variance:

$$\mathrm{Var}(\hat{R}) := \mathbb{E}\left[\left(\hat{R}(f) - \mathbb{E}\left[\hat{R}(f)\right]\right)^2\right]$$

HYPER PARAMETERS

Hyper-parameters: All parameters that are not learned by minimizing the approximated risk

The approximation of the risk my itself depend on hyper-parameters, e.g. weight decay

$$\hat{R}(\phi) := \frac{1}{N}\sum_{i=1}^{N} L(y_i, f_\phi(x_i)) + \theta||\phi||^2$$

ϕ Model parameters
θ Hyperparameter

Model selection can be cast into HPO, e.g. one hot encoding

HYPER PARAMETERS

We have to make a couple of design decisions (select a model)

- Model complexity Λ_Θ , *e.g.* linear functions $\Theta = (a, b)$
- Loss function, *e.g.* quadratic loss $L(x) = x^2$
- Approximation of expected risk, *e.g.* $\hat{R}(f) := \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i))$
- Optimization strategy for $\hat{f} = \arg\min_{f \in \Lambda} \hat{R}(f)$

Decisions belong to model/hyperparameter selection

HYPER PARAMETERS

Hyper-parameters can be

- Continous, e.g. regularization parameter
- Discrete, e.g. number of network layers
- Categorical, e.g. optimization methods (SGD, Adam,…)
- Conditional, e.g. HPs in Adam if selected

EXAMPLE

ϕ : Model parameters θ : Hyperparameters

Neural networks

$$y = g_N(g_{N-1}((\ldots g_1(x))))$$ $$\theta := \{A_i, b_i\}$$

$$g_i(x) = \sigma_i(A_i x + b_i),\ A_i \in \mathbb{R}^{m_i \times n_i}, b_i \in \mathbb{R}^{n_i} \quad \sigma_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$$

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

Multi-label classification

Neural network hyperparameter θ :

Learning rate, loss function, mini-batch size, number of epochs, momentum, number of hidden units, weight decay, nonlinearity, weight initialization, layer types(full, convolution, pooling), batch normalization, layer connections, dropout…

HPO METHODS

- **Theoretical derivations**
- Information criteria (Bayesian, Akaike,…)
- **Fully Bayesian approach/Evidence framework**
- Ensemble methods
- **Grid search – cross validation**
- **Random search – cross validation**
- **Bayesian optimization**
- **Successive Halving / HyperBand**

THEORETICAL DERIVATIONS

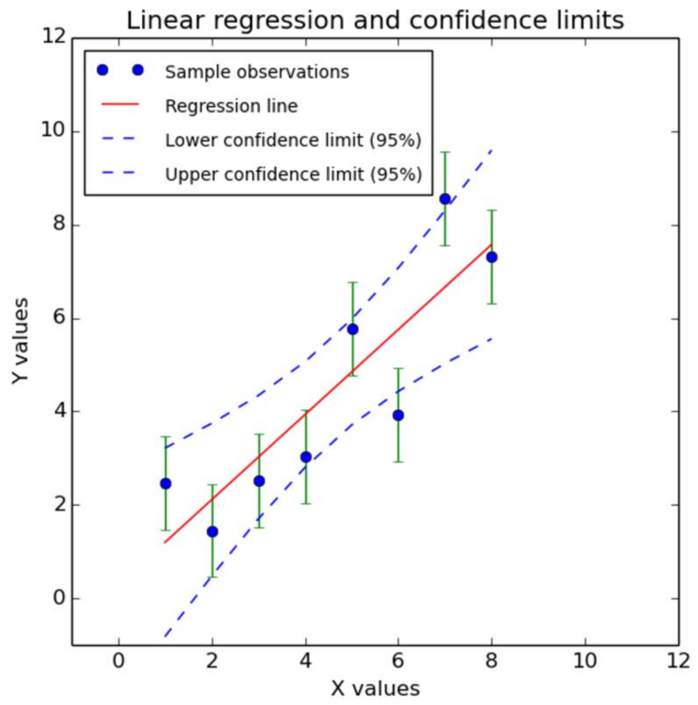
Example: Linear regression

$$y = ax + b + \varepsilon \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2) \quad \Rightarrow \quad L(x) = x^2$$

Emphirical risk:

$$\hat{R}_N(a,b) := \frac{1}{N}\sum_{i=1}^{N}(y_i - ax_i - b)^2$$

$$\hat{a}, \hat{b} = \arg\min_{a,b} \hat{R}(a,b)$$



FULLY BAYESIAN APPROACH

If HP can be modeled by random variable within a probabilistic approach, they could be integrated out

$$p(\tilde{y}|\tilde{x}) = \int p(\tilde{y}|\tilde{x}, \phi, \theta) p(\phi, \theta|\mathcal{D}) d\phi d\theta$$

Downside: Closed form solution only for few distributions

Alternatives: Numerical approximations with ensemble approaches

EVIDENCE FRAMEWORK

Alternative terms: Type-II likelihood, marginal likelihood

If HP can be modeled by random variable and the its likelihood can be derived, we can estiamte the hyperparamter

$$p(\mathcal{D}|\theta) = \int p(\mathcal{D}|\phi, \theta)p(\phi)d\phi$$

Downside: Closed form solution only for few distributions

Alternatives: Approximate distributions by simple pdfs first (e.g. variational approximation), then apply evidence approximation

MODEL SELECTION

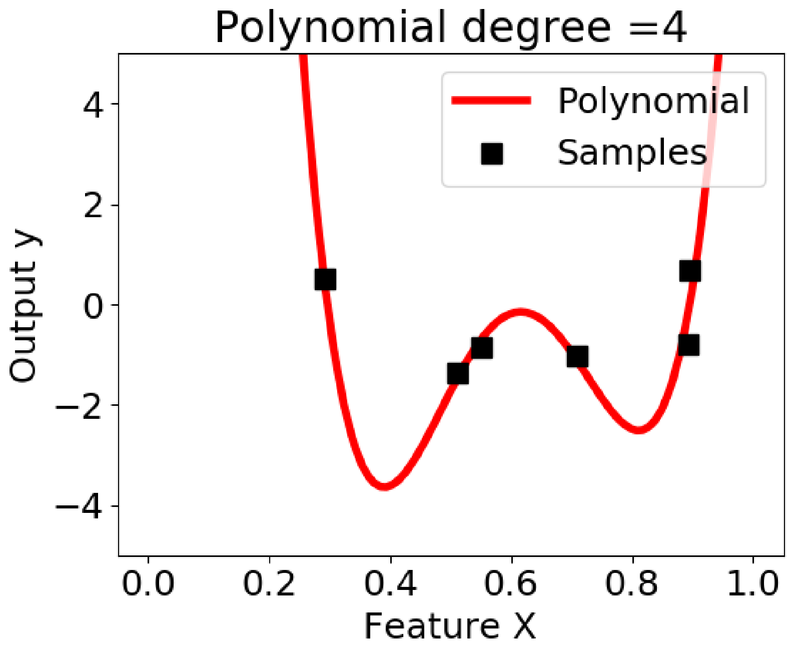
Example: Polynomial regression

$$y = a_0 + a_1 x + \dots + a_q x^q = \vec{a}^T \vec{x_q}, \quad \phi = (a_1, a_2, \dots, a_q)$$
$$\vec{x}_q := (1, x, \dots, x^q) \qquad \theta = q$$

Emphirical risk:

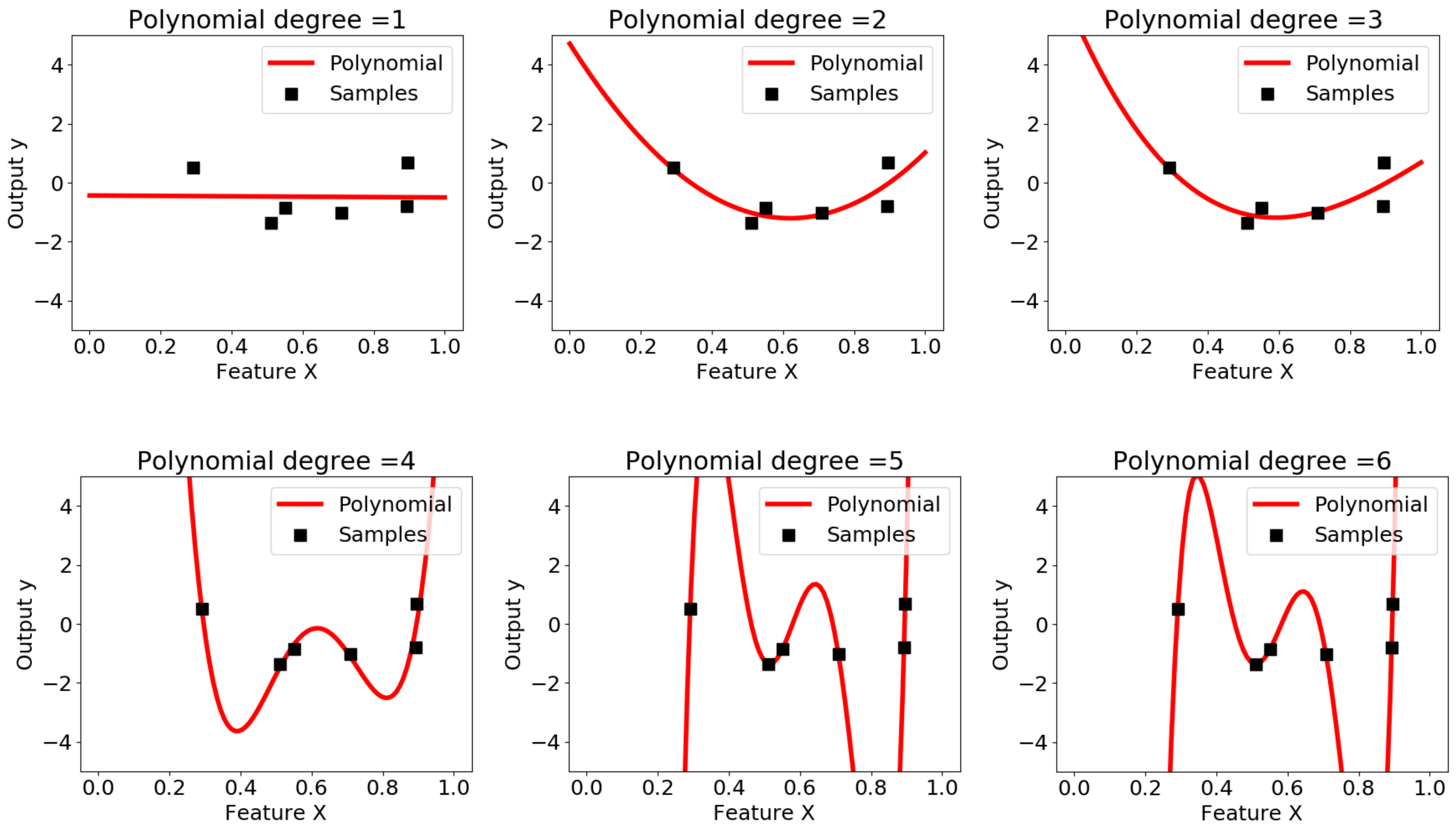
$$\hat{R}_N(\vec{a}) := \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \vec{a}^T\vec{x}_n\right)^2$$

$$\hat{\vec{a}} = \arg\min_{\vec{a}} \hat{R}(\vec{a})$$

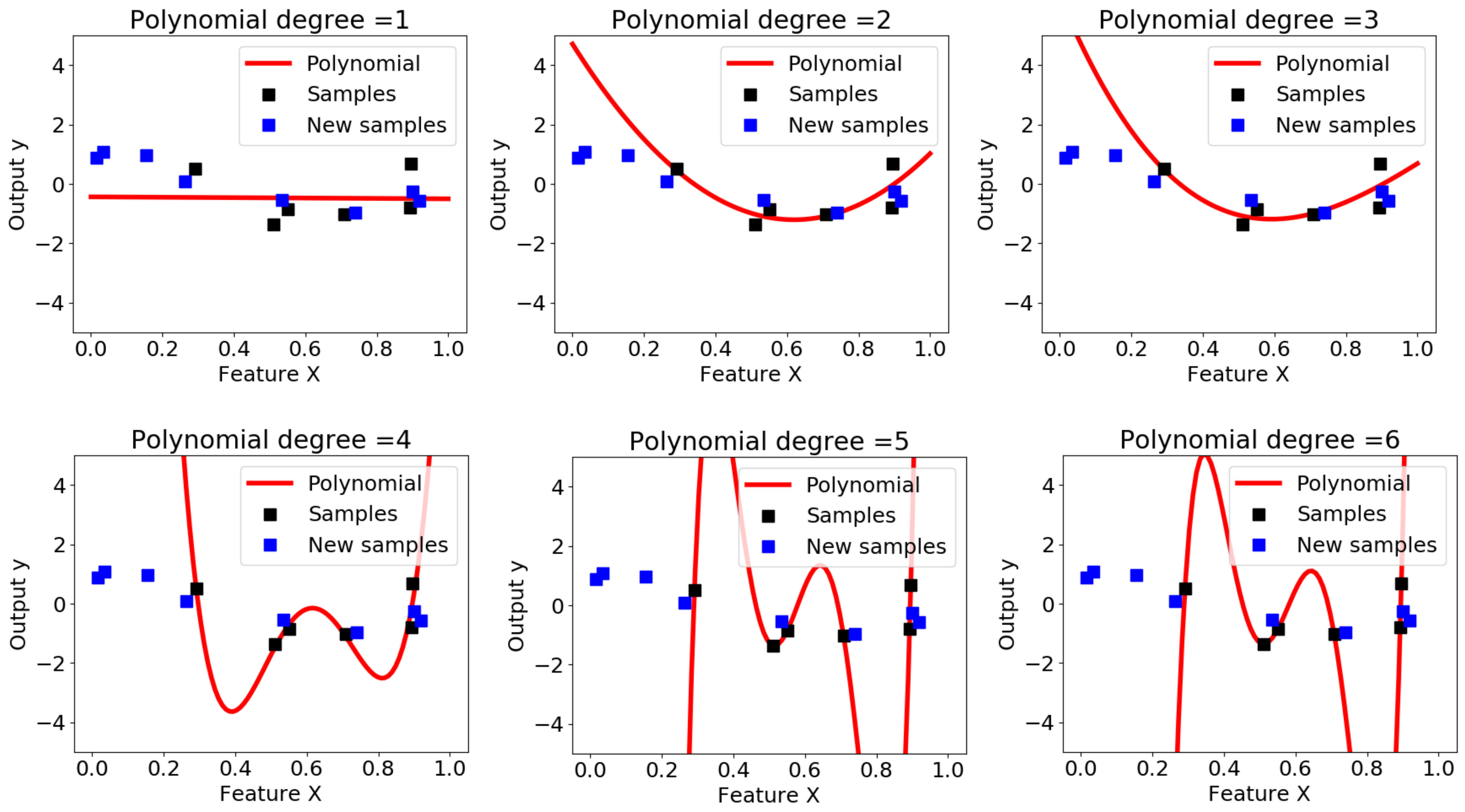


JÜLICH Forschungszentrum

EXAMPLE: POLYNOMIAL REGRESSION

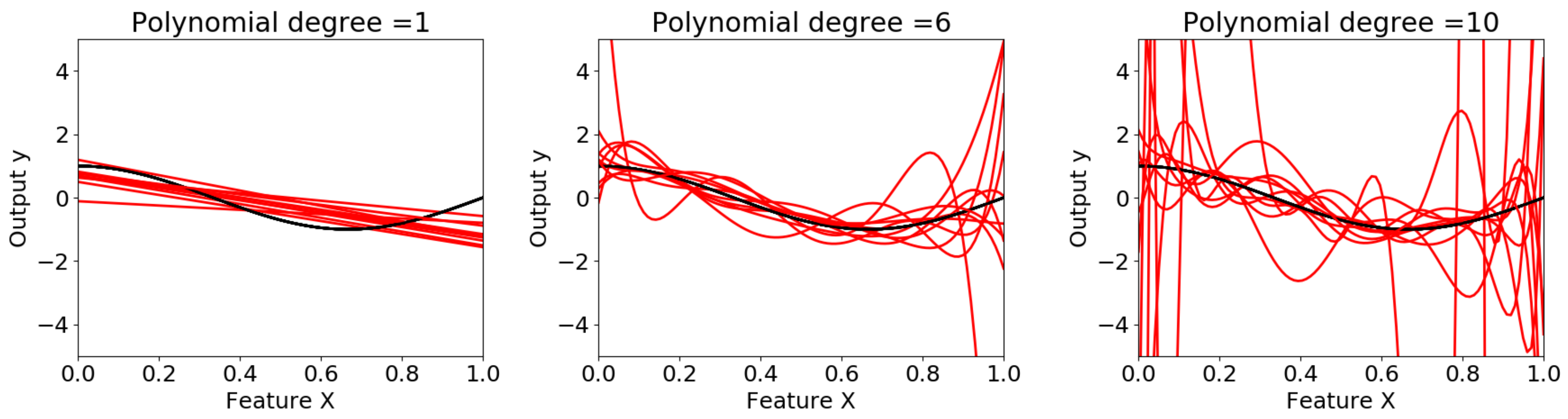


EXAMPLE: POLYNOMIAL REGRESSION

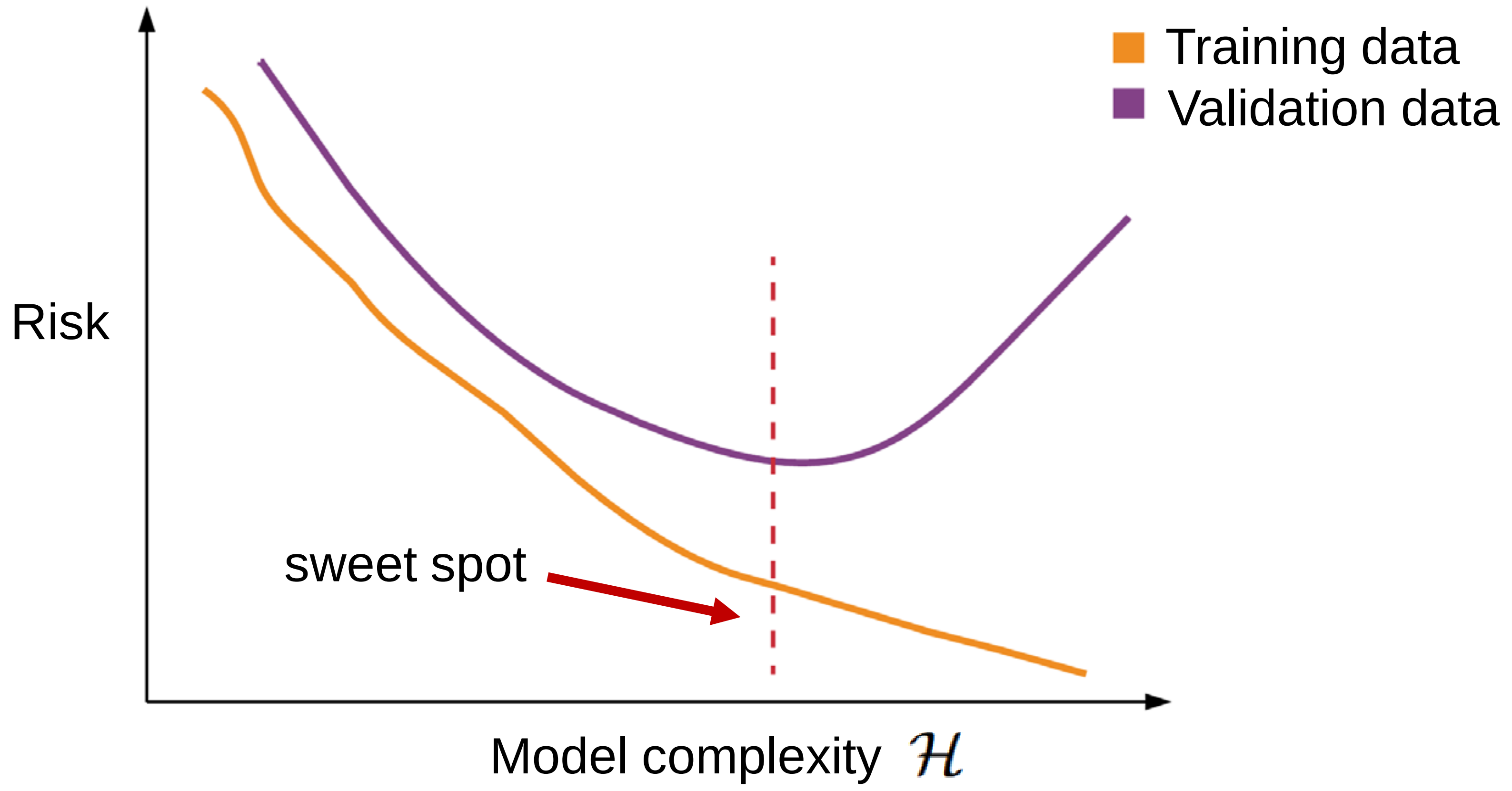


BIAS – VARIANCE TRADEOFF

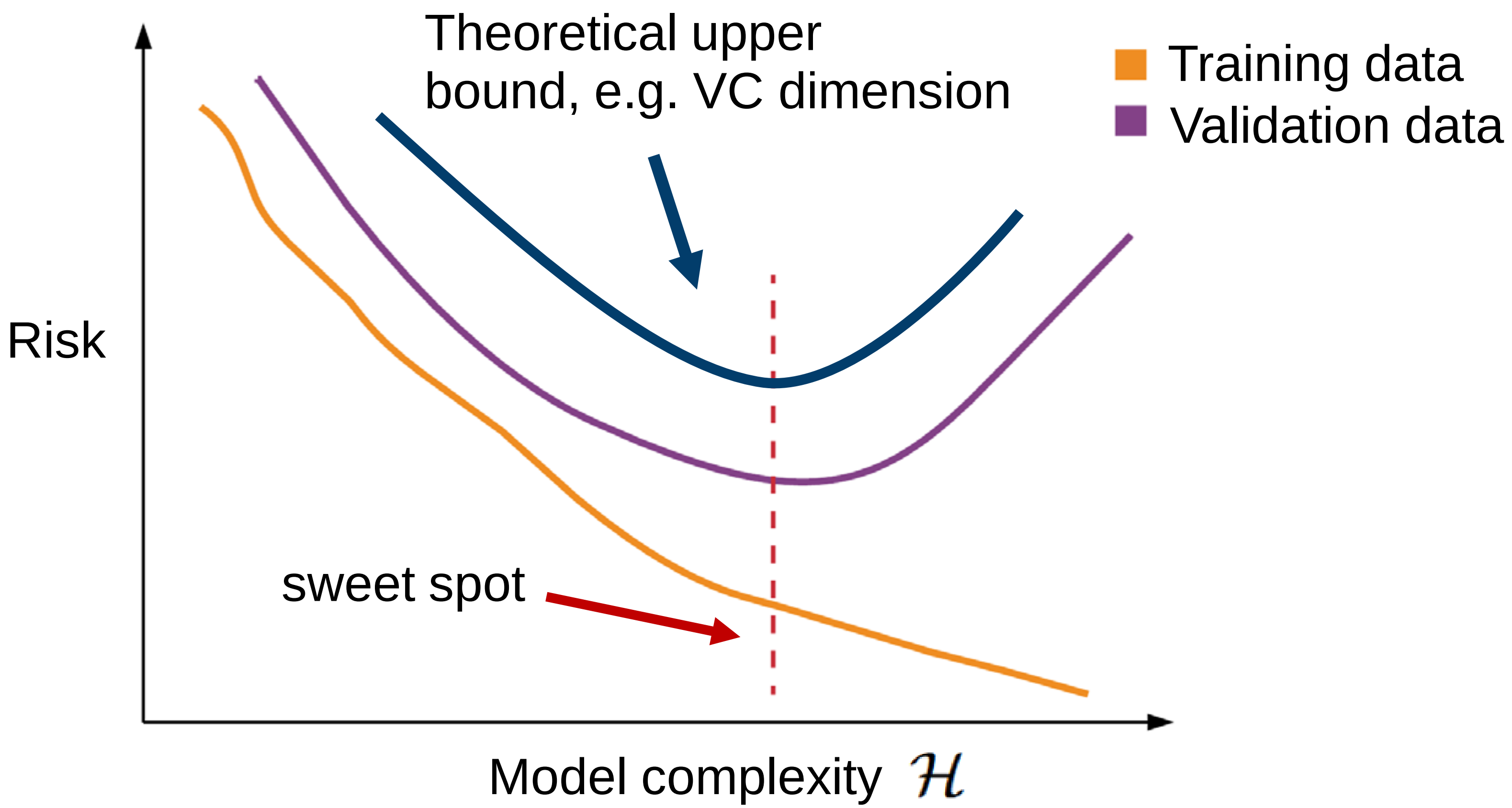
Learn the model on different training sets (#samples=6)



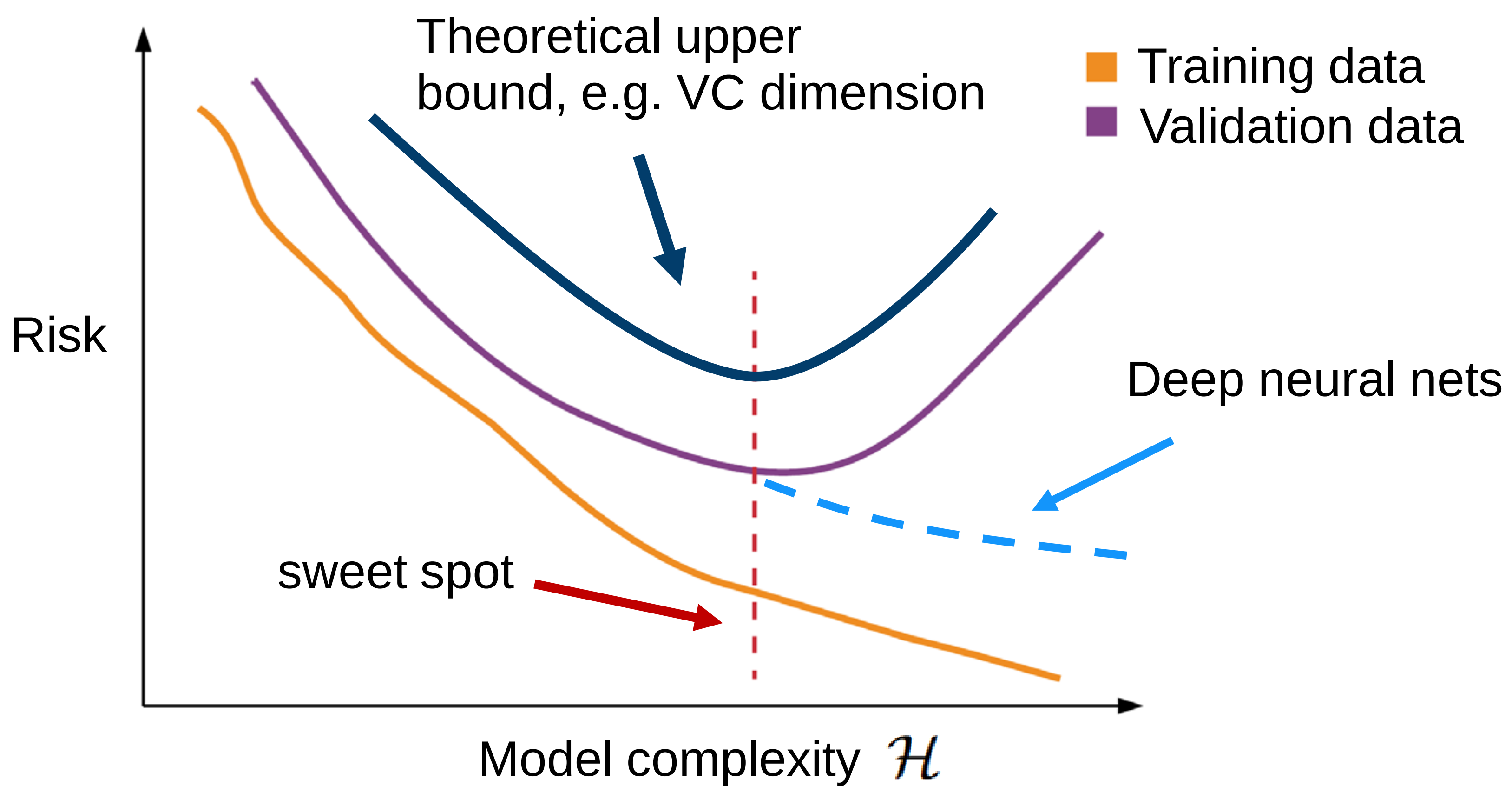
BIAS – VARIANCE TRADEOFF



BIAS – VARIANCE TRADEOFF

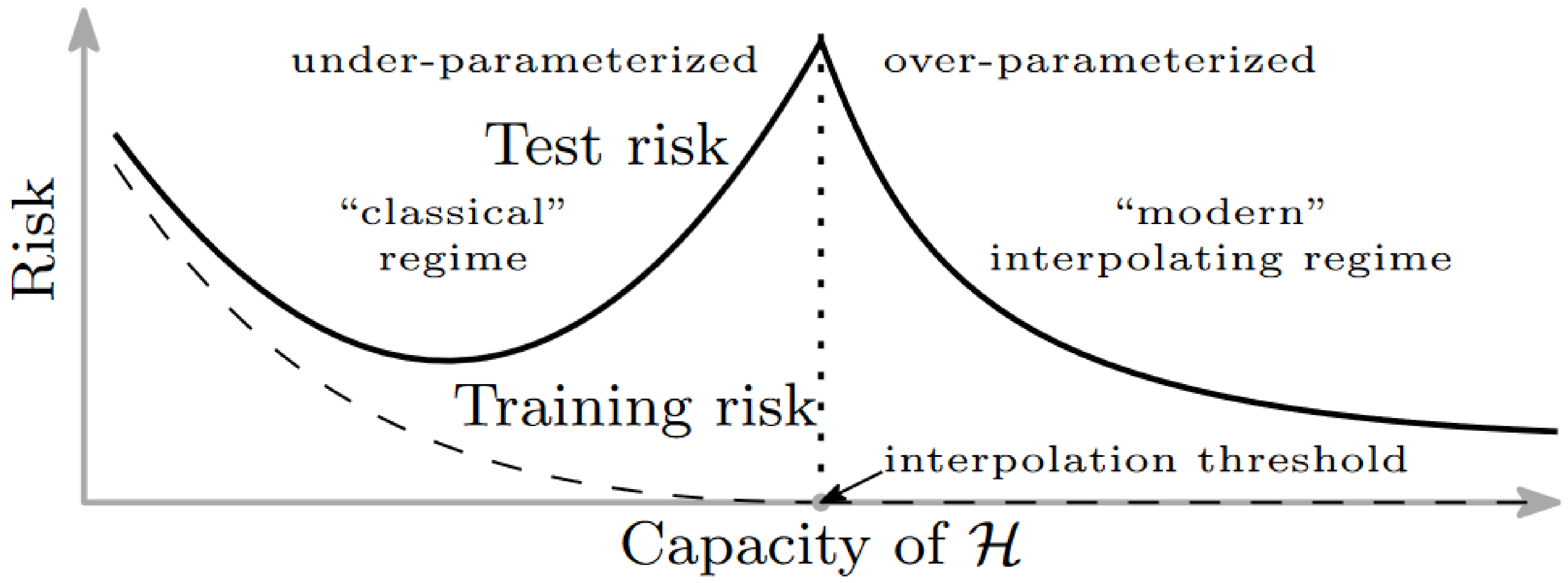


BIAS – VARIANCE TRADEOFF


Theoretical upper bound, e.g. VC dimension
🟧 Training data
🟪 Validation data
Risk
Deep neural nets
sweet spot
Model complexity $\mathcal{H}$

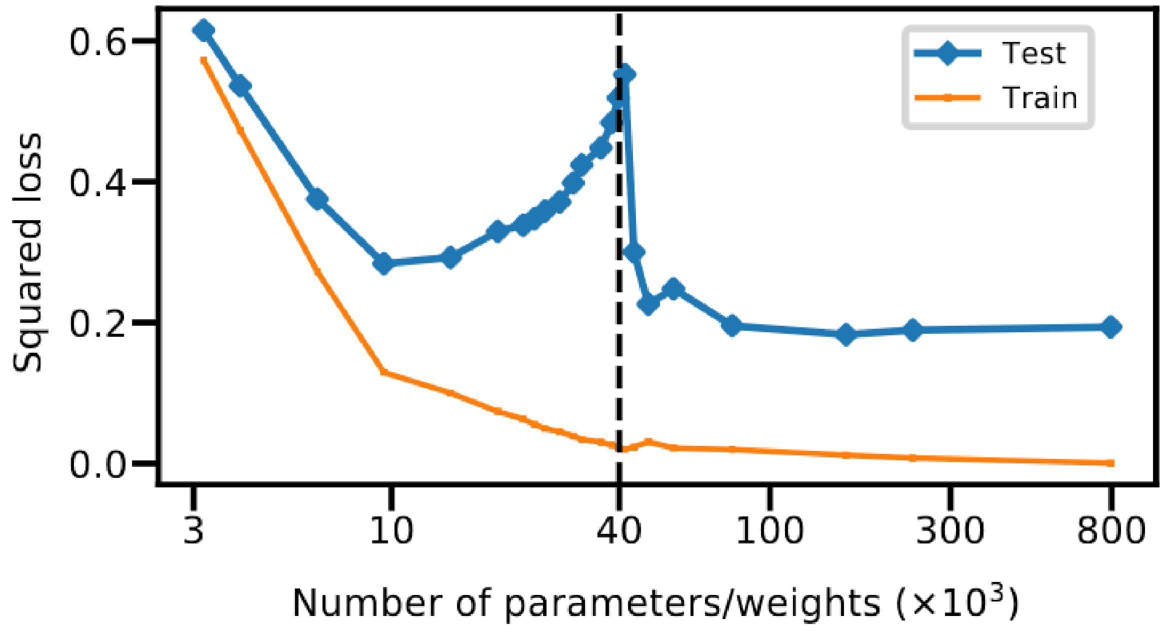
DOUBLE DESCENT CURVE

Belkin *et al.* (2019) proposed a double descent curve for high capacity models



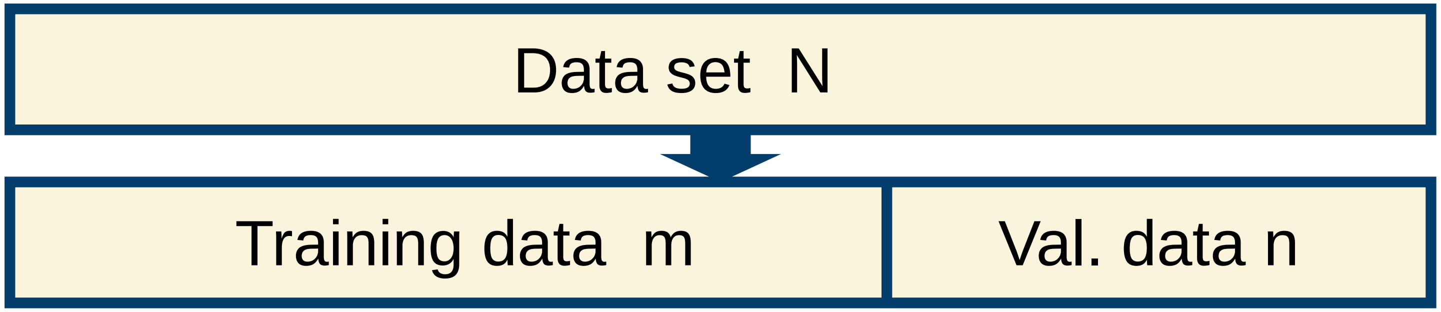
DOUBLE DESCENT CURVE

… and emphirically found them in a couple of models, e.g.
One hidden layer NN trained on 4000 MNIST images



HOLDOUT METHOD

Idea: Split the available data into training data and validation data (typical: validation data 20% to 40%)



1) Train parameters for fixed hyperparameters on training data according to the emphirical risk:

$$\hat{R}(f) := \frac{1}{m} \sum_{i=1}^{m} L(y_i, f(x_i))$$

2) Repeat 1.) for several sets of hyperparameter and compare the models on the validation data

$$\hat{R}(f) := \frac{1}{n} \sum_{i=m+1}^{m+n} L(y_i, f(x_i))$$

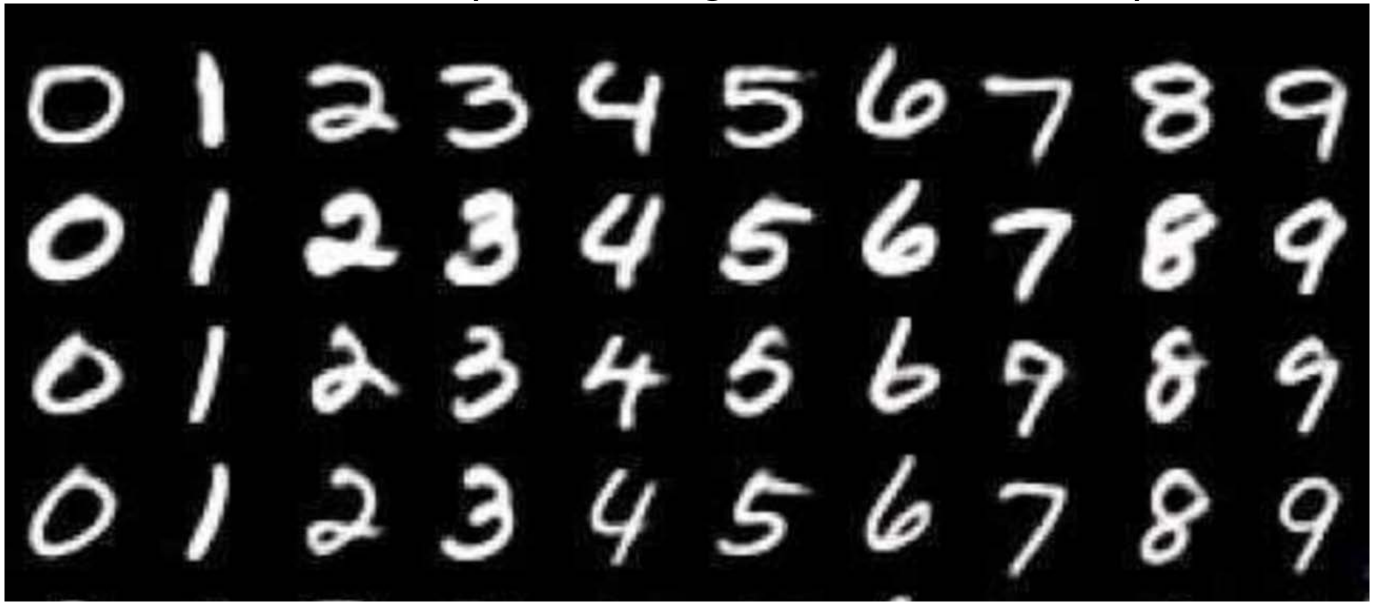
HOLDOUT METHOD

Example: Logistic ridge regression on MNIST Dataset

$$y = \sigma\left(\vec{a}^T\vec{x}\right), \; L(y,\hat{y}) = \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1-y_i)\log(\hat{y}_i) + \lambda||\vec{a}||^2$$

σ : Sigmoid function $\quad \Phi = \vec{a} \quad \Theta = \lambda \quad \vec{x} \in \mathbb{R}^{784}$

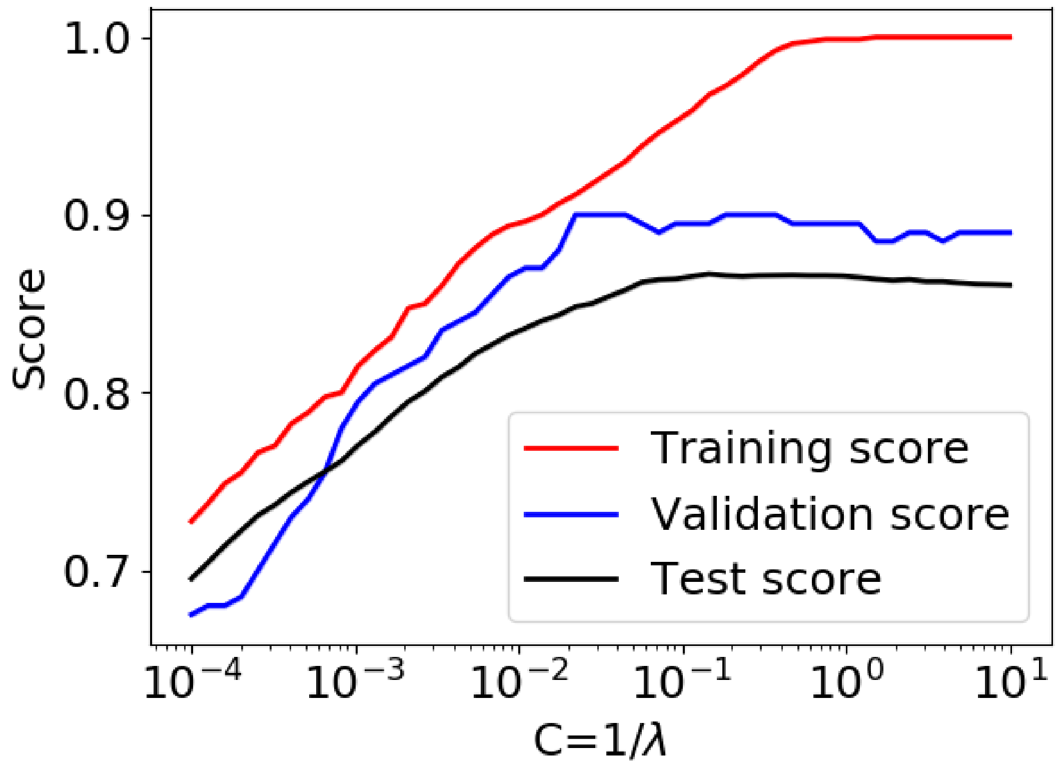
MNIST Data (60k training data, 10k test data)



JÜLICH Forschungszentrum

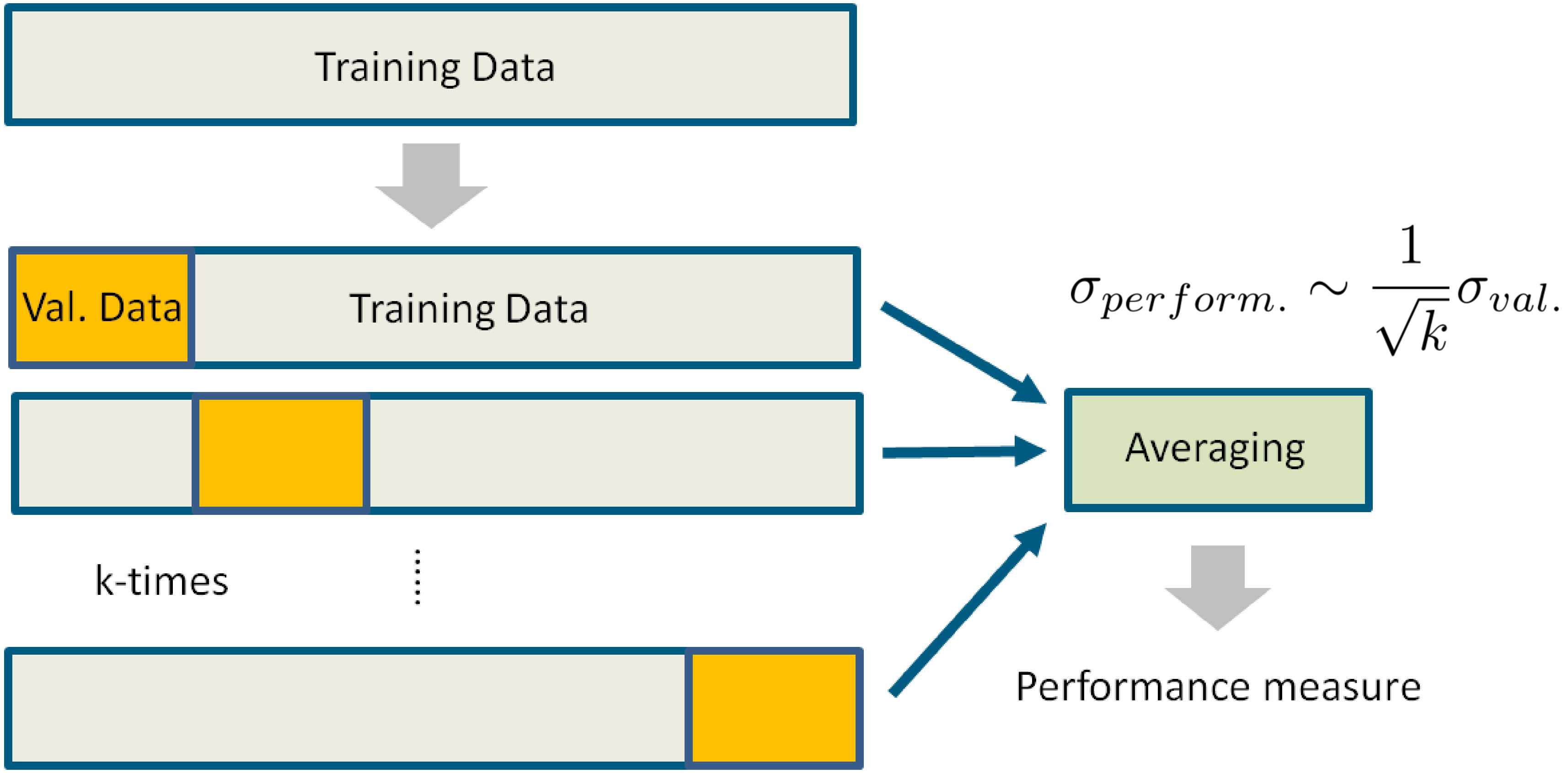
HOLDOUT METHOD

Example: Logistic ridge regression on MNIST Dataset
# of training images: 800, # of validation images: 200
# of test images: 9000



CROSS VALIDATION

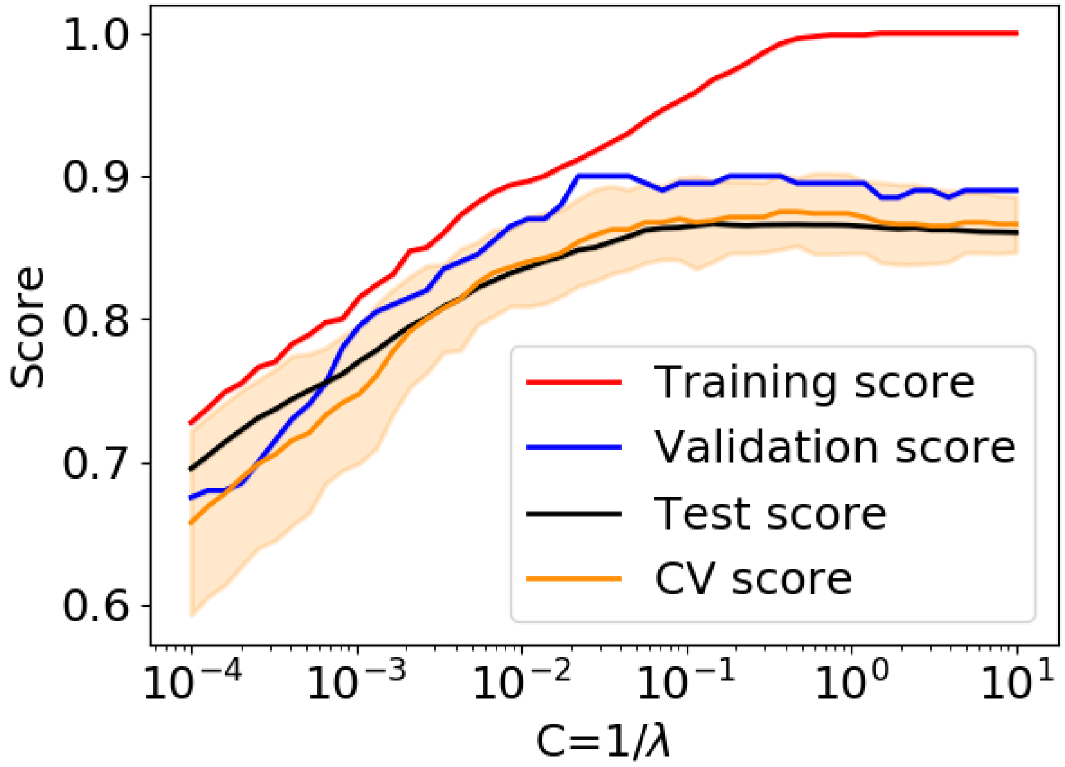
- Trainings data is divided into k non-overlapping patches (k-fold cross validation)



$$\sigma_{perform.} \sim \frac{1}{\sqrt{k}}\sigma_{val.}$$

K-FOLD CROSS VALIDATION

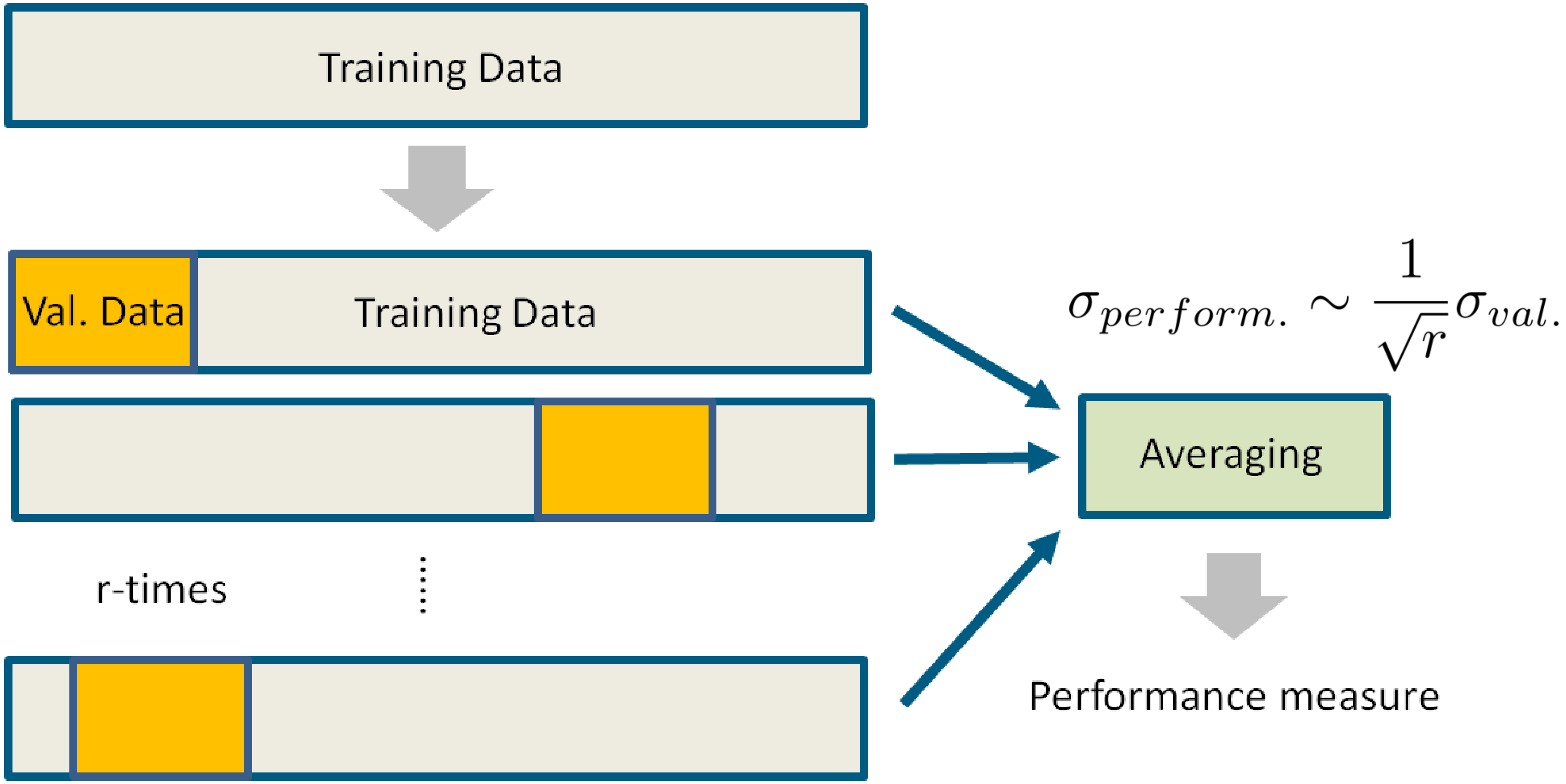
Example: Logistic ridge regression on MNIST Dataset
# of training images: 800, # of validation images: 200
# of test images: 9000
10-Fold cross-validation on 1000 samples



JÜLICH
Forschungszentrum

RANDOM SAMPLING

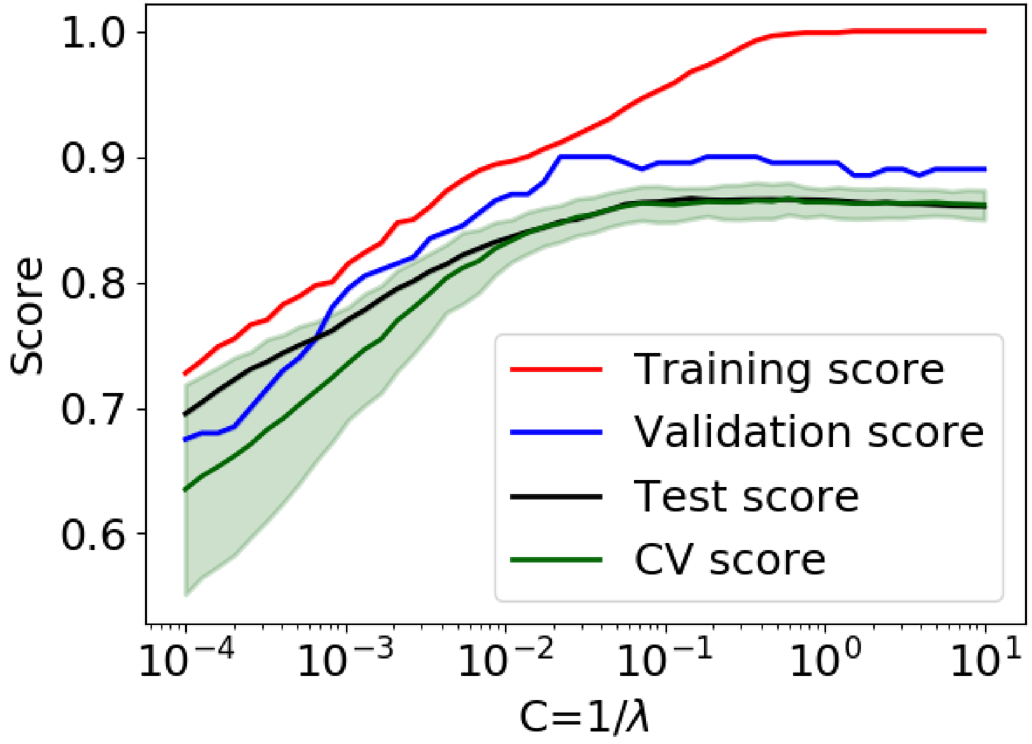
- Trainings data is divided into r random overlapping patches (repeated random sub-sampling validation)



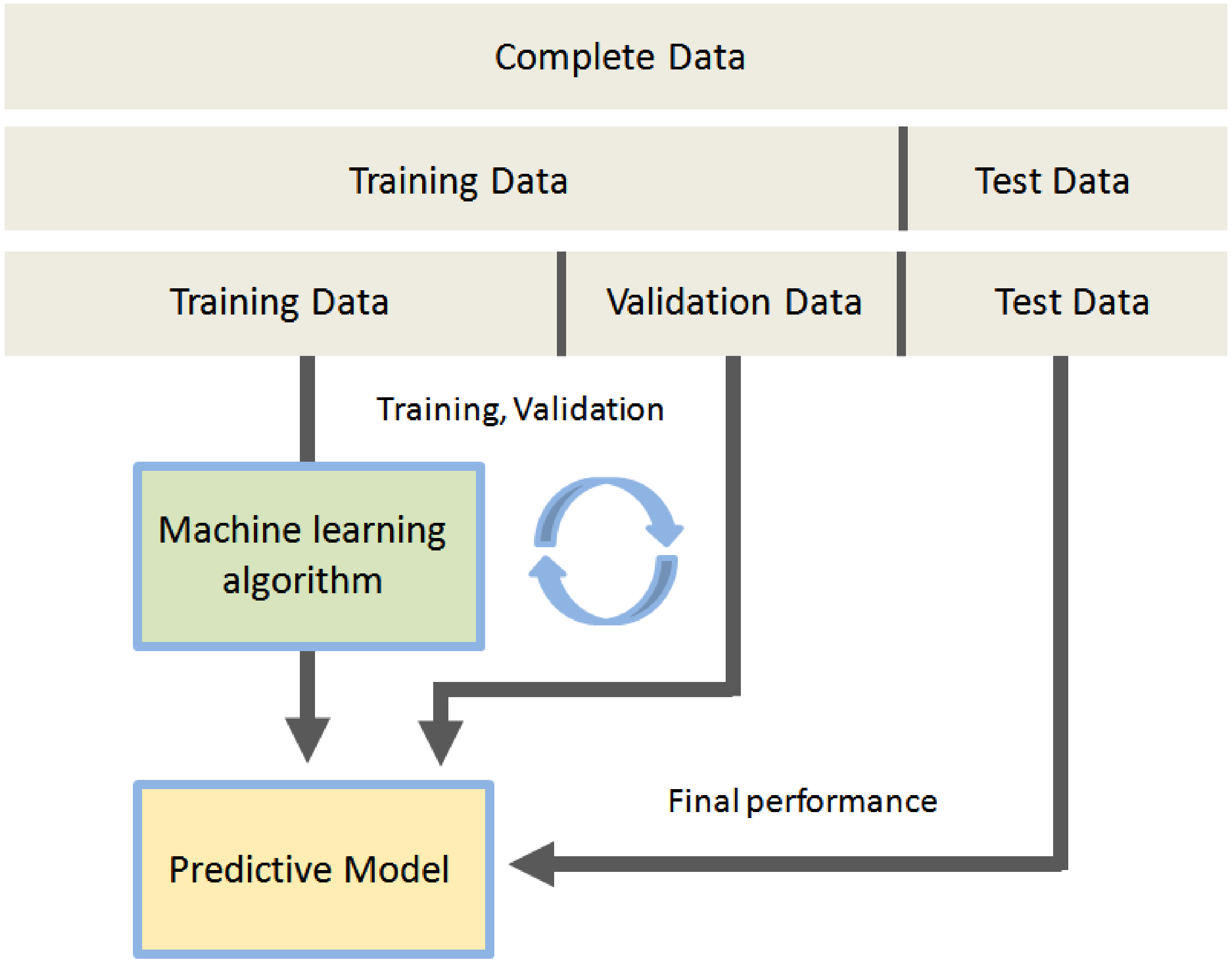
$$\sigma_{perform.} \sim \frac{1}{\sqrt{r}}\sigma_{val.}$$

RANDOM SAMPLING

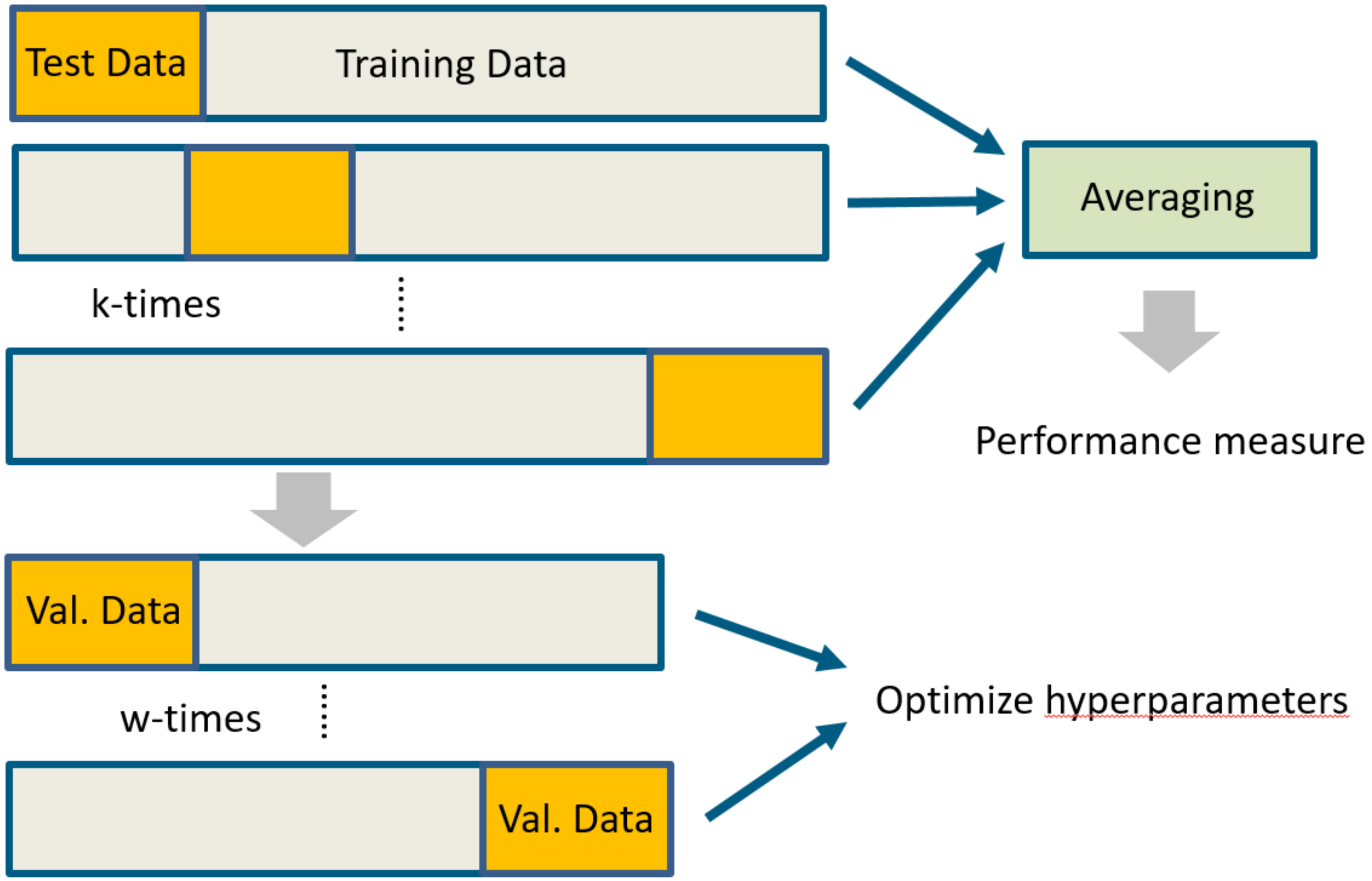
Example: Logistic ridge regression on MNIST Dataset
# of training images: 800, # of validation images: 200
# of test images: 9000
random sampling with 40% validation data of 1000



MODEL COMPARISON



NESTED CROSS-VALIDATION



LEARNING CURVE

Cross validation allows to estimate the „sweet spot“
but does not allow to judge if enough training data
is used

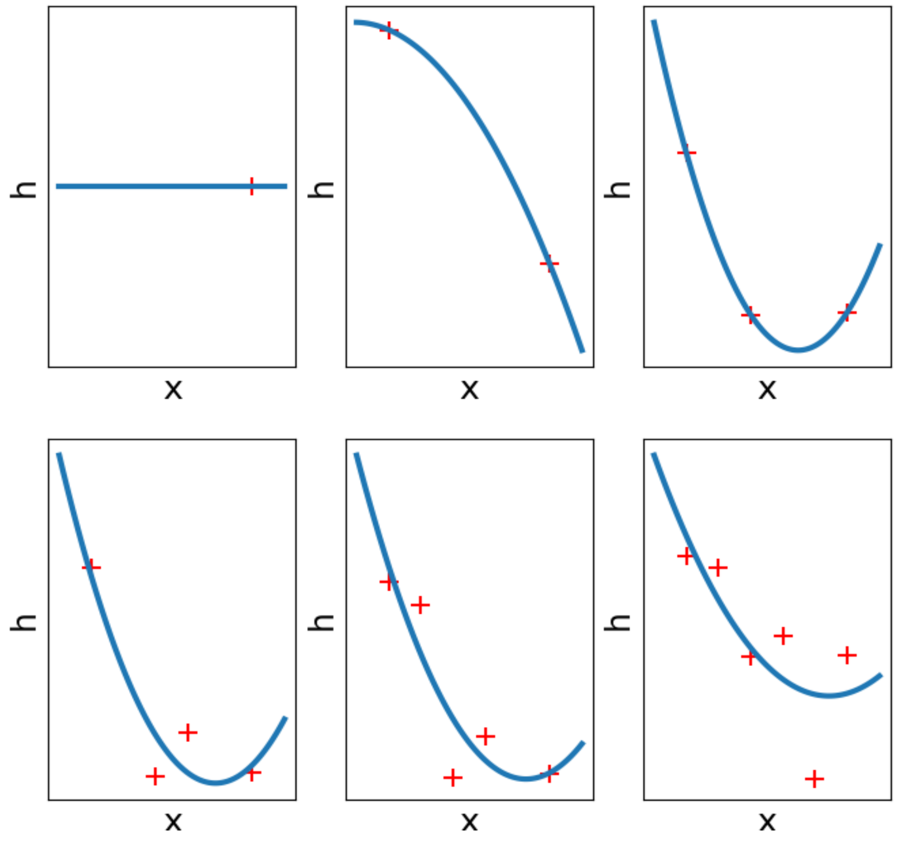
- optimal tradeoff bewtween training and validation/test data
- the need to obtain more data

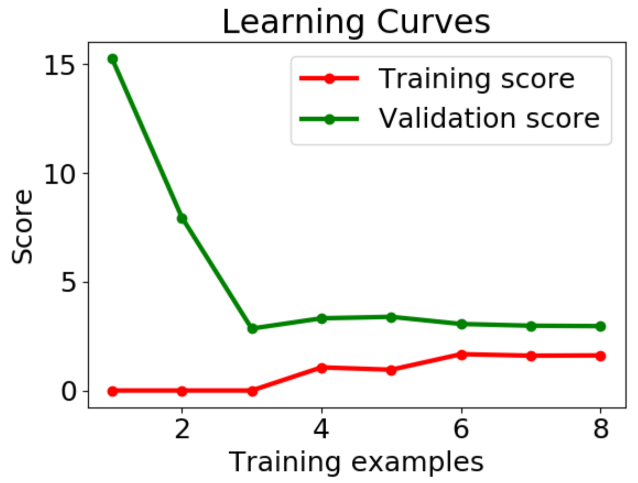
Idea: Train the model on subsets of the training data and
observe ist performance as amount of data
used for training increases

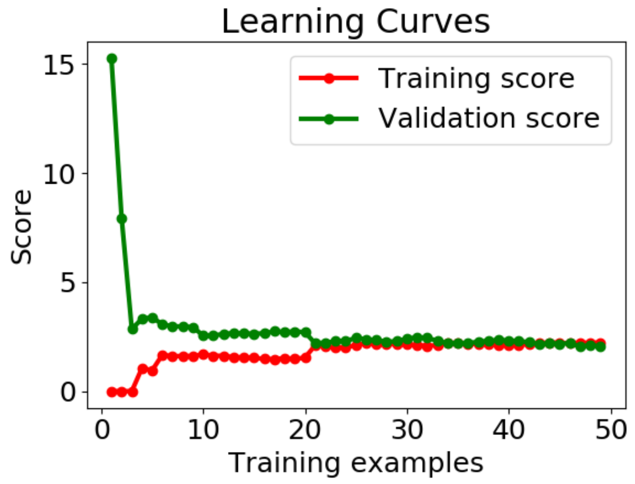
LEARNING CURVE

$$h_a(x) = a_0 + a_1 x + a_2 x^2$$

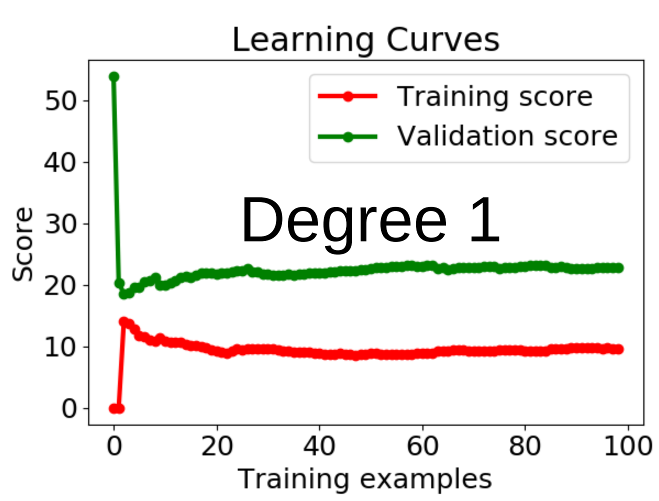
$$y = h_a(x) + \varepsilon$$

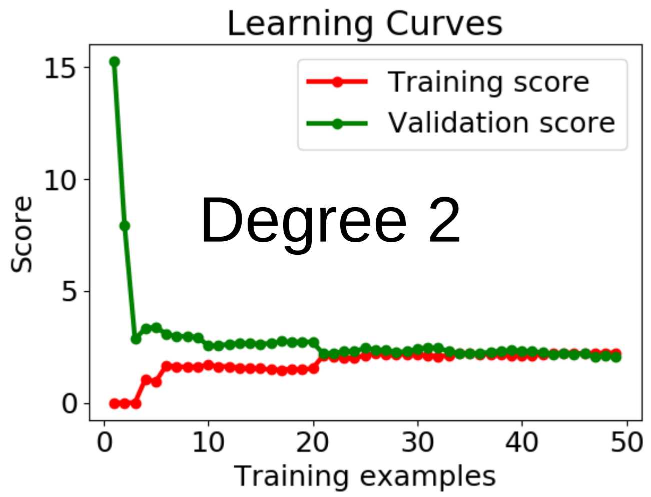


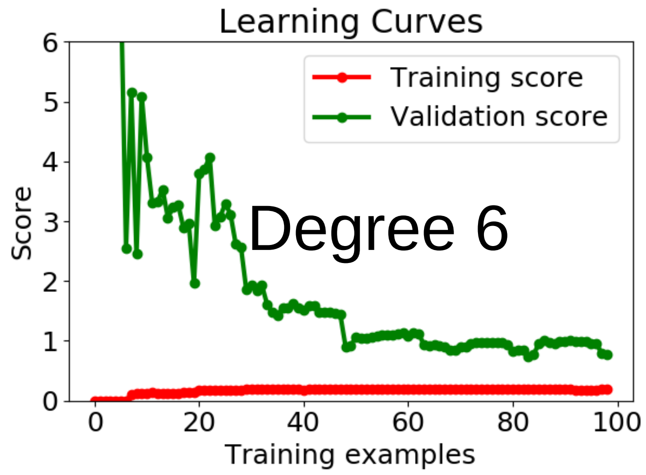


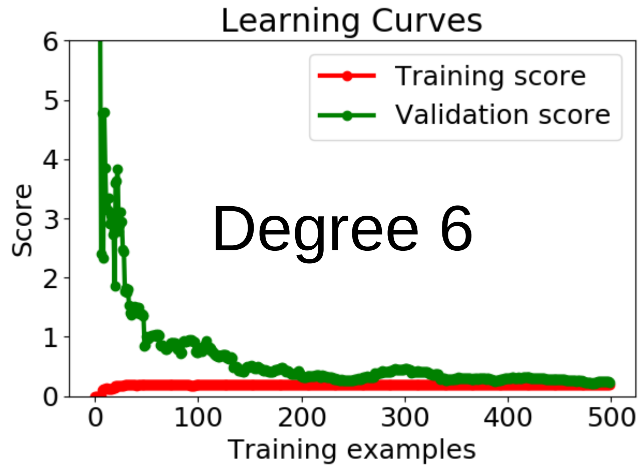


LEARNING CURVE



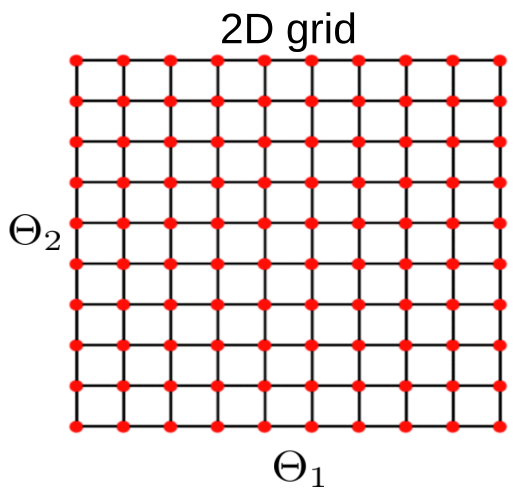






GRID SEARCH

- Exhaustive search through a manually specified subset of the hyperparameter space
- Apply CV on each hyperparameter set

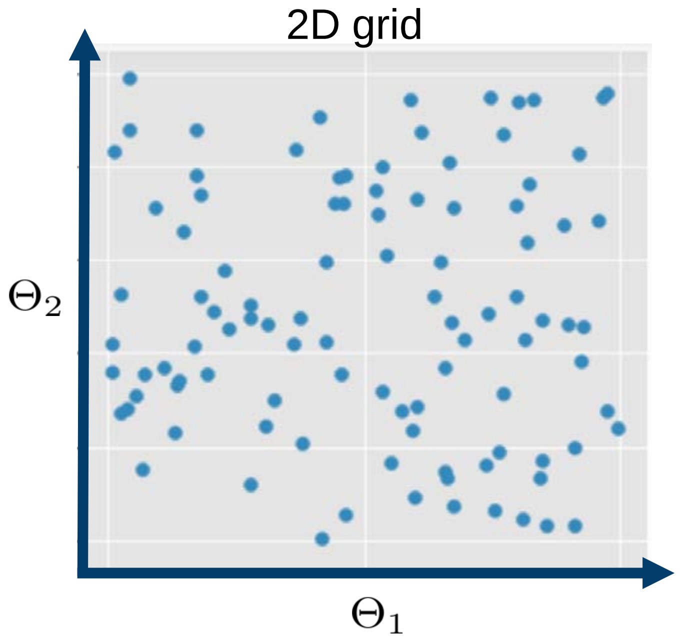


- Full parallelizable as model applied independently on each hyperparameter set
- Suffers from the curse of dimensionality, *e.g.* 10 hyper-parameter per dimension ➡ 10^{10} for dim=10

RANDOM SEARCH

Random search tend to be less expensive and time consuming because they do not examine every possible combination of parameters

Randomly selects a chosen number of hyperparameter sets from a given domain and tests only those



RANDOM SEARCH

Advantages:

- The experiment can be stopped any time and the trials form a complete experiment
- New trials can be added to an experiment without having to adjust the grid and commit to a much larger experiment
- Scales independent of the input dimension
- Good coverage, e.g. if the region of hyperparameters that are near optimal occupies at least 5% of the grid, then random search with 60 trials will find that region with high probability (94%).

RANDOM SEARCH - EXAMPLE

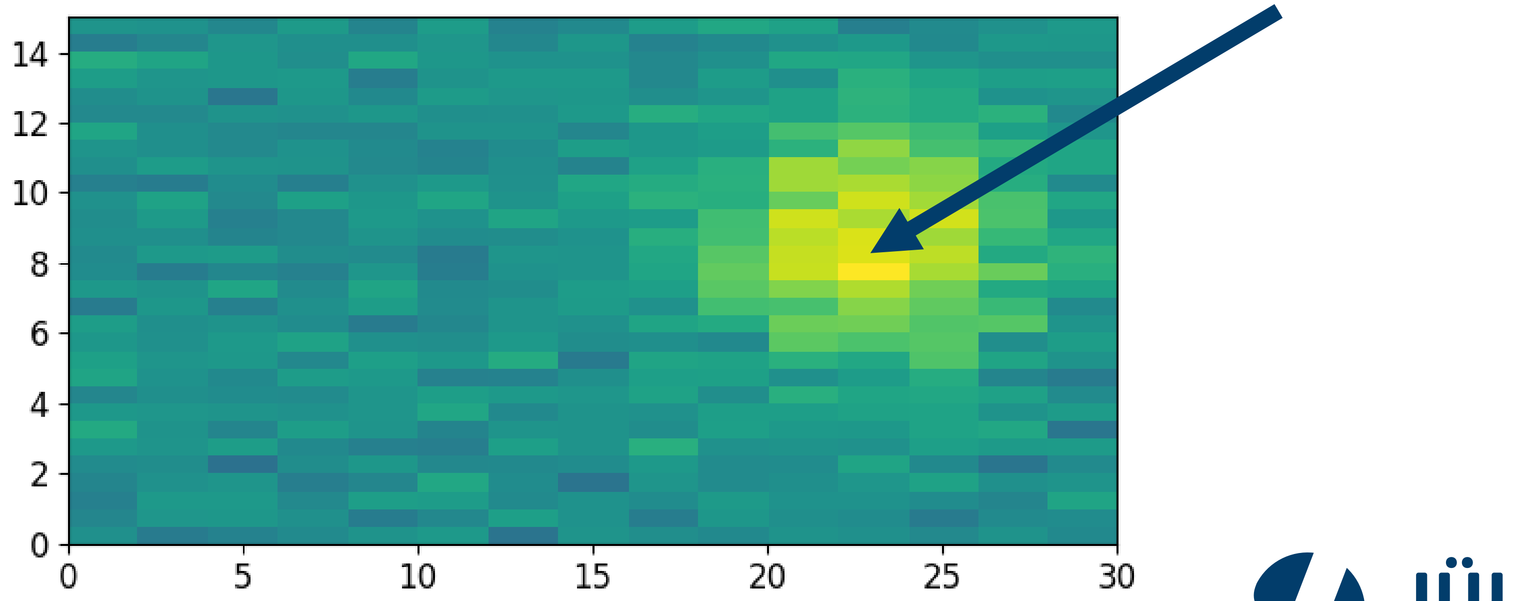
Highlighted in green are the 21 pairings
with the highest scores out of the 450 total combinations

Probability not to hit the yellow area with 60 trials

$$q = \left(1 - \frac{21}{450}\right)^{60} \approx 5.7\%$$

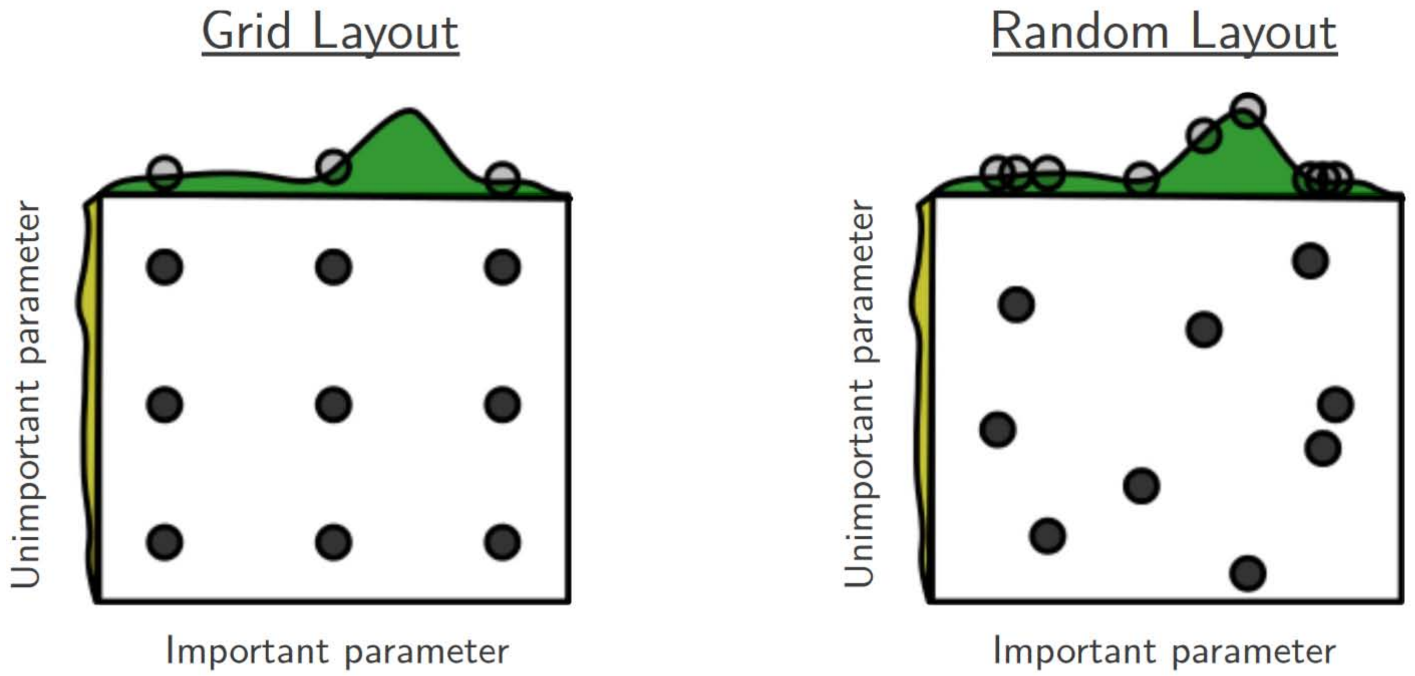
Probability to hit the green area with 60 trials

$$p = 1 - q \approx 94.3\%$$



RANDOM SEARCH – GRID SEARCH

If some hyper-parameters are unimportant for the model grid search waste computing time



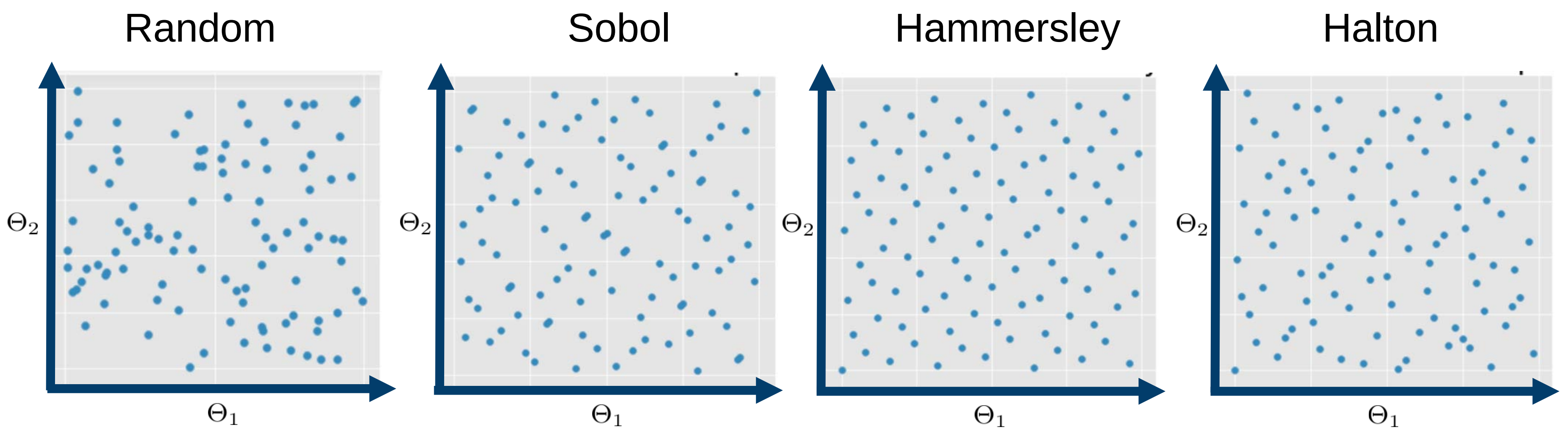
Bergstra and Bengio: Random Search for Hyper-Parameter Optimization, IJML, 2012

QUASI – RANDOM SEQUENCES

Few random samples tend to form clusters and wholes

➡ Domain not covered evenly

Quasi-random sequence is a deterministic irrelgular sequence with the property that for all values of N, its subsequence $x_1, ..., x_N$ has a low discrepancy



JÜLICH Forschungszentrum

BAYESIAN OPTIMIZATION

As the evaluation of the validation empirical risk is usually costly we want to use previous experience to choose the next HP configuration

Idea: Learn a surrogate of the validation expected risk that is cheap to evaluate and provide a confidence interval to determine the next grid point

$$S(\Theta) \approx \hat{R}(f_\Theta)$$

Search the hyperparameter space with

- **Exploration:** Seek places with high variance in expected risk
- **Exploitation:** Seek places with low expected risk

GAUSSIAN PROCESSES

A Gaussian process defines a probability distribution $p(f)$ over functions

$$S : \mathbb{R}^N \to \mathbb{R}$$

Notice: S is an infinite-dimensional quantity

➡ No explicit probability distribution can be defined

Consider the vector $\vec{S} := (S(x_1), S(x_2), ..., S(x_N))$ of function values evaluated at finite number of positions

Definition: A Gaussian Process GP is a collection of random variables of which each finite sample is a multivariate Gaussian distribution $\vec{S} \sim \mathcal{N}(\vec{m}, \Sigma)$

GAUSSIAN PROCESSES

Definition: A Gaussian Process (GP) is a collection of random variables of which each finite sample is a multivariate Gaussian distribution

A Gaussian process is fully determined by its *mean function*

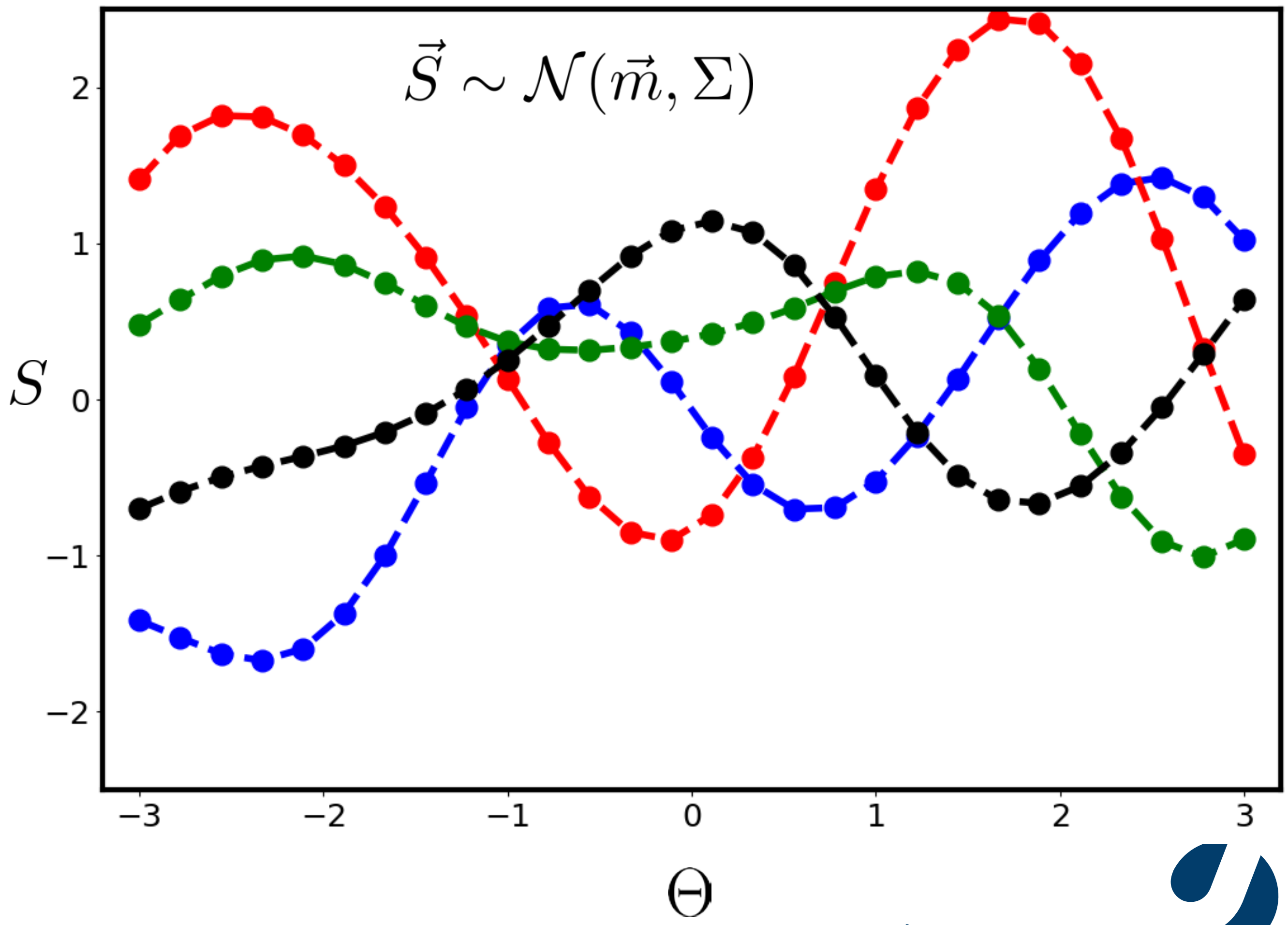
$$m(x) = \mathbb{E}[S(x)]$$

and *covariance function* (kernel)

$$K(x, x') = \mathbb{E}[(S(x) - m(x))\,(S(x') - m(x'))]$$

MULTIVARIATE GAUSSIAN

How to sample functions?
Lets start with sampling from multivariate Gaussian
and interpolate points



KERNEL FUNCTIONS

Exponential kernel

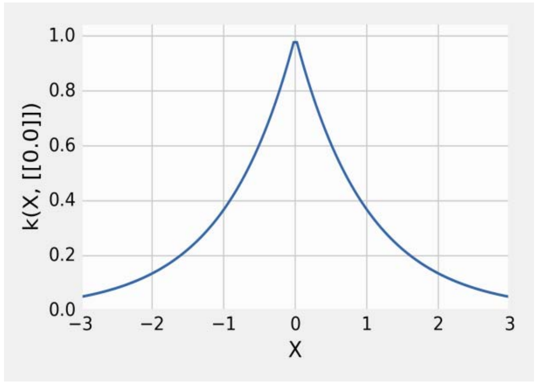
$$K(x,x') = \sigma^2 \exp\left(-\frac{d}{2L^2}\right)$$
$$d := \|x - x'\|$$

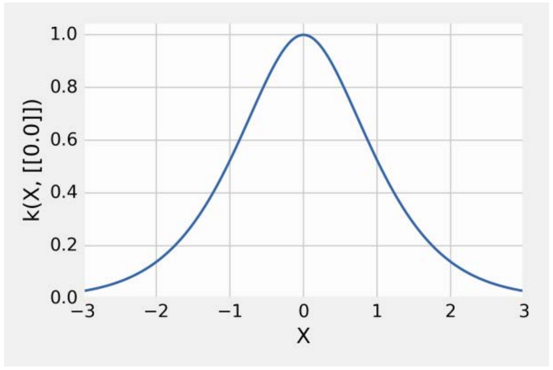
Matern kernel

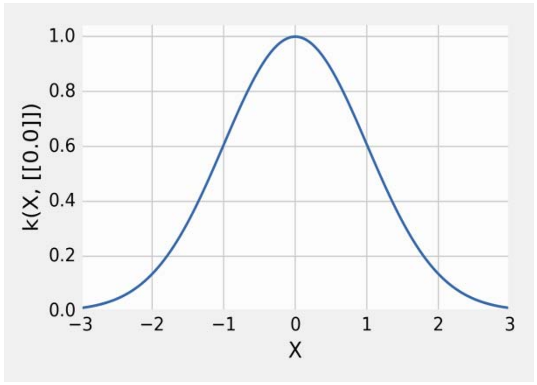
$$K(x,x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\mu}|d|}{L}\right)^{\mu} I_{\mu}\left(\frac{\sqrt{2\mu}|d|}{L}\right)$$

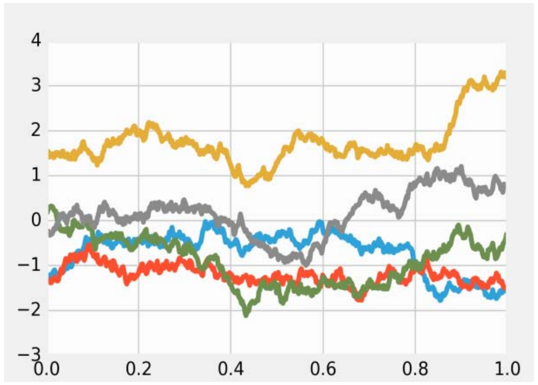
RBF kernel

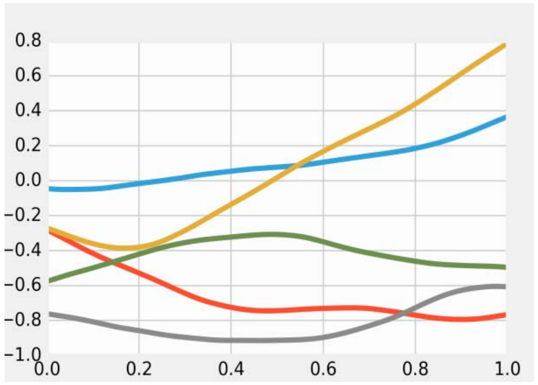
$$K(x,x') = \sigma^2 \exp\left(-\frac{d^2}{2L^2}\right)$$

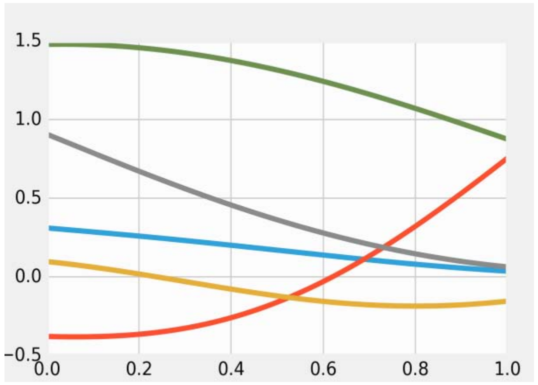












GAUSSIAN PROCESSES

$y \in \mathbb{R}^m$ Observed samples

$S \in \mathbb{R}^N$ Positions to be evaluated

$$(S, y) \sim \mathcal{N}\left(\vec{0}, \begin{pmatrix} K & K_* \\ K_* & K_{**} \end{pmatrix}\right) \Rightarrow S|y \sim \mathcal{N}(m_{post.}, K_{post.})$$

Posterior mean

$$m_{post.} = K_* K^{-1} y$$

Posterior Covariance

$$K_{post.} = K_{**} - K_* K^{-1} K_*^T$$

GAUSSIAN PROCESSES

Kernels need to be choosen (model selection) and kernel parameter have to be set

➡ We are faced with additional hyperparameters

But GP allow to use additional hyperparameter Optimization techniques

- Marginal likelihood (evidence framework)
- Fully Bayesian approach

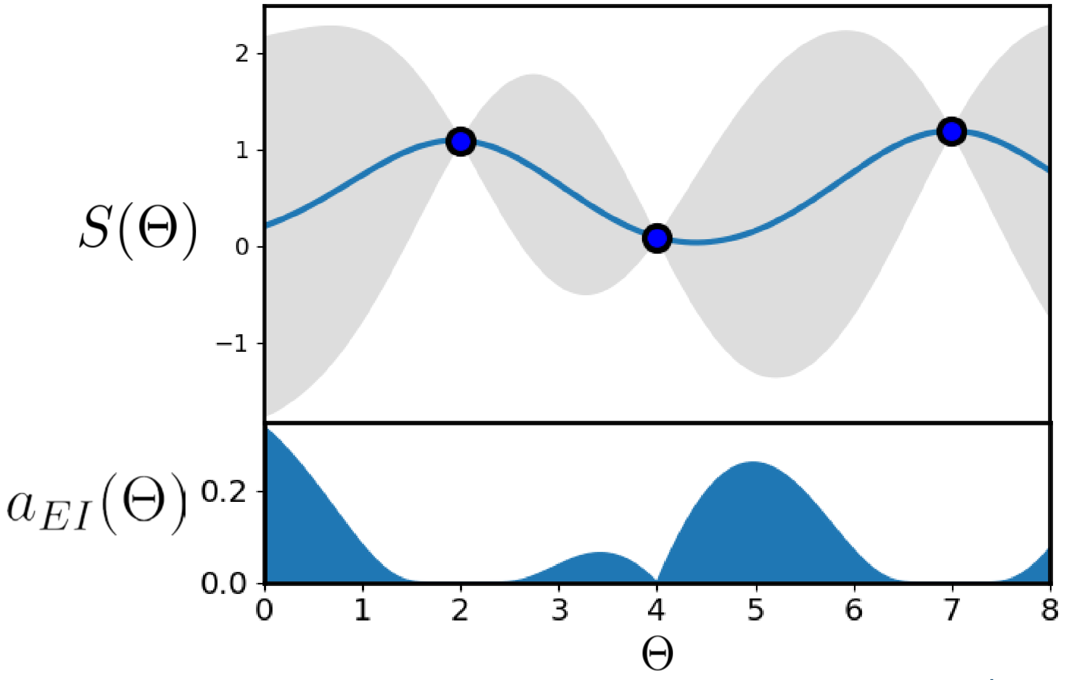
ACQUISITION FUNCTION

We need some criterion to select a new grid point based on our GP surrogate model

Expected Improvement (Mockus 1978):

$$a_{EI}(\Theta) = \int \max\left(S(\Theta_{min}) - S(\Theta)), 0\right) p(S(\Theta), m_{post.}, K(\Theta, \Theta))df$$

$$= \left(S(\Theta_{min}) - m_{post.}(\Theta)\right) \Psi + K(\Theta, \Theta) p(S(\Theta), m_{post.}, K(\Theta, \Theta))$$

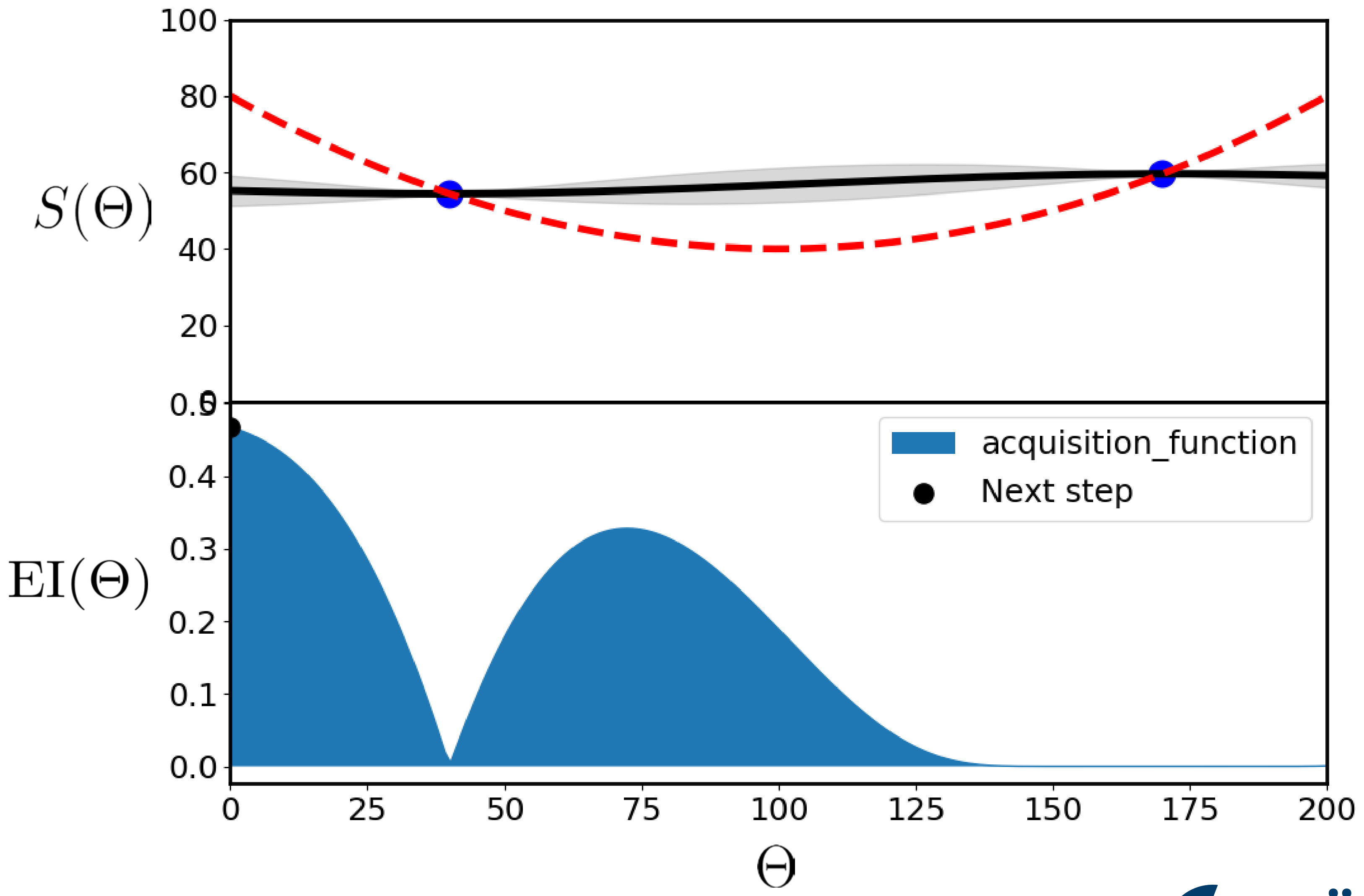


BAYESIAN OPTIMIZATION

We combine all steps to Bayesian optimization

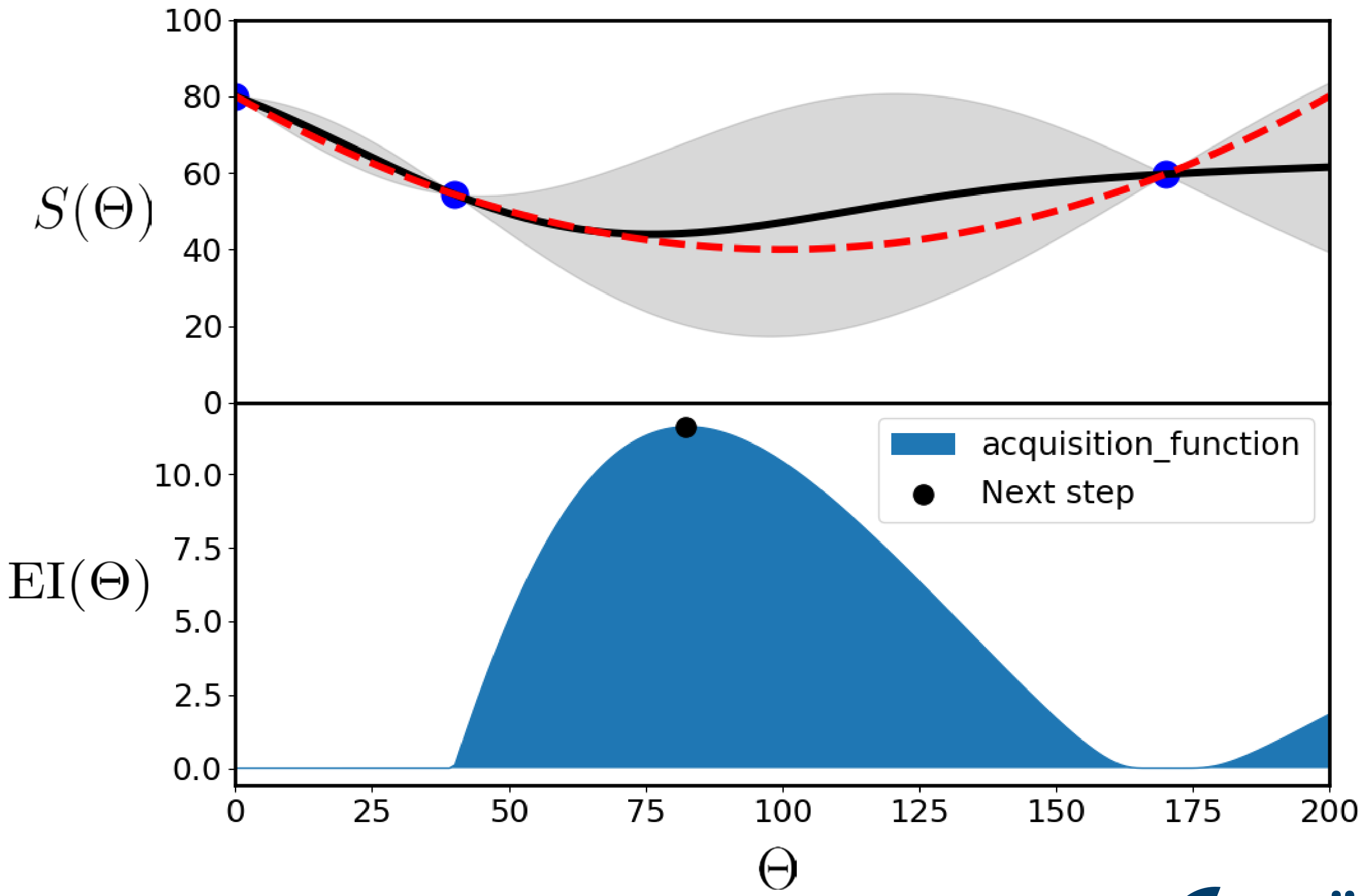
1) Evaluate $\hat{R}(f)$ for some grid points, e.g. random search
2) Estimate a GP surrogate model based on current grid points
3) Optimize GP hyperparameters by Evidence/Fully Bayesian
4) Compute an aquisition function
5) Select a new grid point by maximizing the aqusition function
6) Go to 2) or stop if $\hat{R}(f)$ does not change any more

EXAMPLE BO

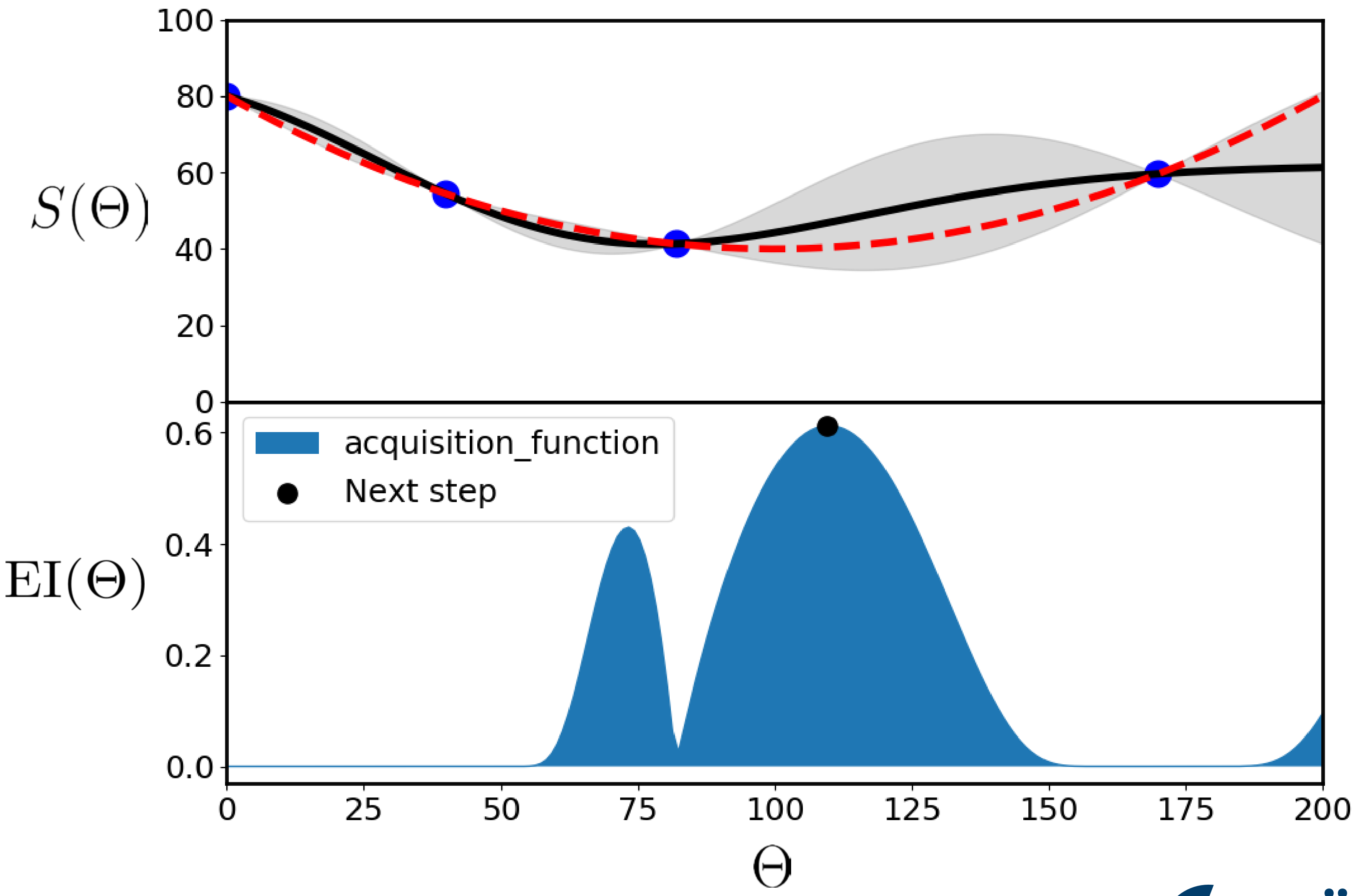


JÜLICH
Forschungszentrum

EXAMPLE BO

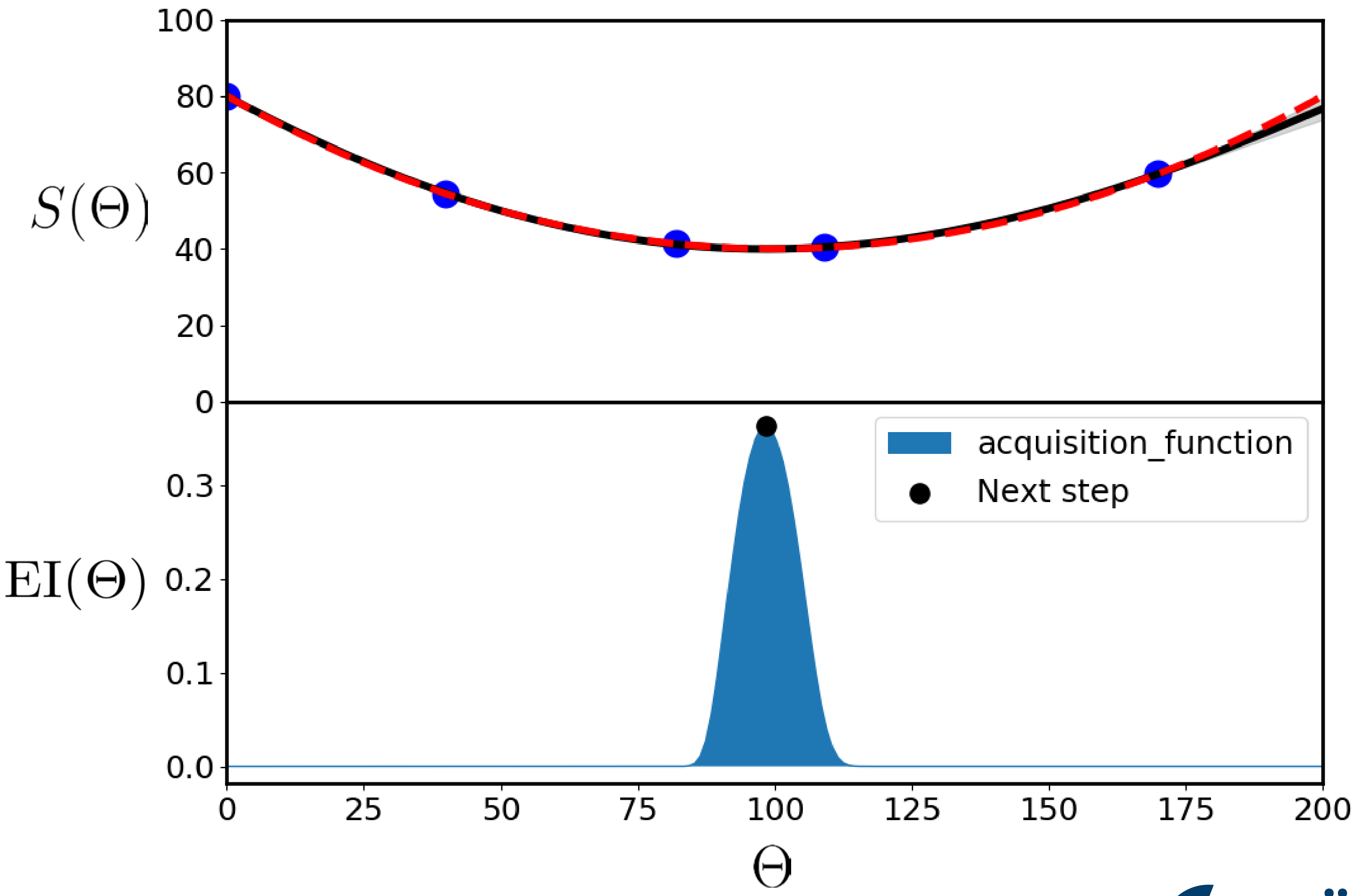

100
80
60
40
20
0
$S(\Theta)$
10.0
7.5
5.0
2.5
0.0
$\mathrm{EI}(\Theta)$
acquisition_function
Next step
0
25
50
75
100
125
150
175
200
Θ

EXAMPLE BO



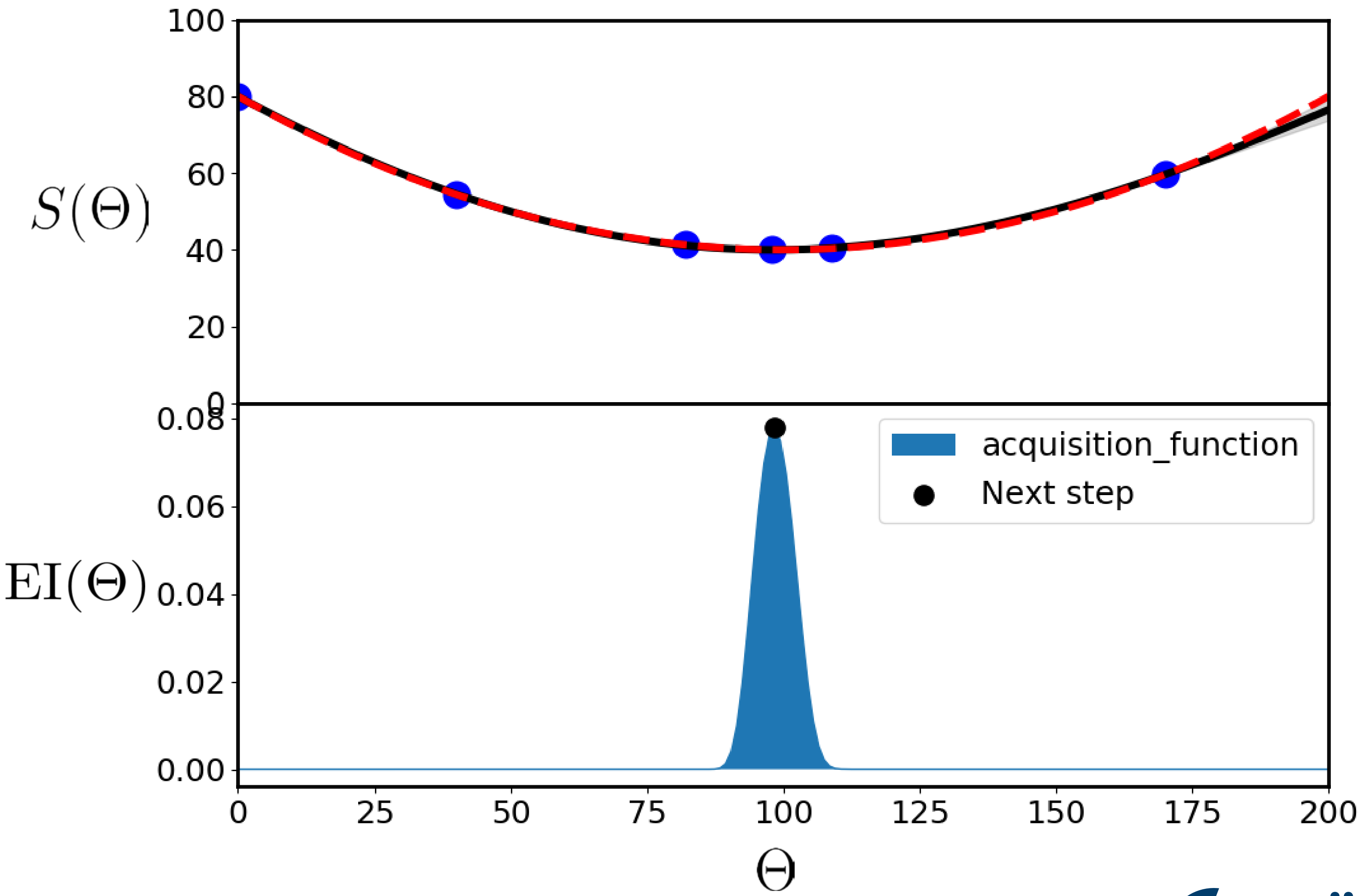
100
80
60
40
20
0
$S(\Theta)$
0.6
0.4
0.2
0.0
$\mathrm{EI}(\Theta)$
acquisition_function
Next step
0
25
50
75
100
125
150
175
200
Θ

EXAMPLE BO



$S(\Theta)$
100
80
60
40
20
0
$\mathrm{EI}(\Theta)$
0.3
0.2
0.1
0.0
acquisition_function
Next step
0
25
50
75
100
125
150
175
200
Θ

EXAMPLE BO



100
80
60
40
20
0
$S(\Theta)$
0.08
0.06
0.04
0.02
0.00
$\mathrm{EI}(\Theta)$
0
25
50
75
100
125
150
175
200
Θ
acquisition_function
Next step

EXTENSIONS

GPs do not scale well with the number of risk evaluations $O(n^3)$

Further developments:

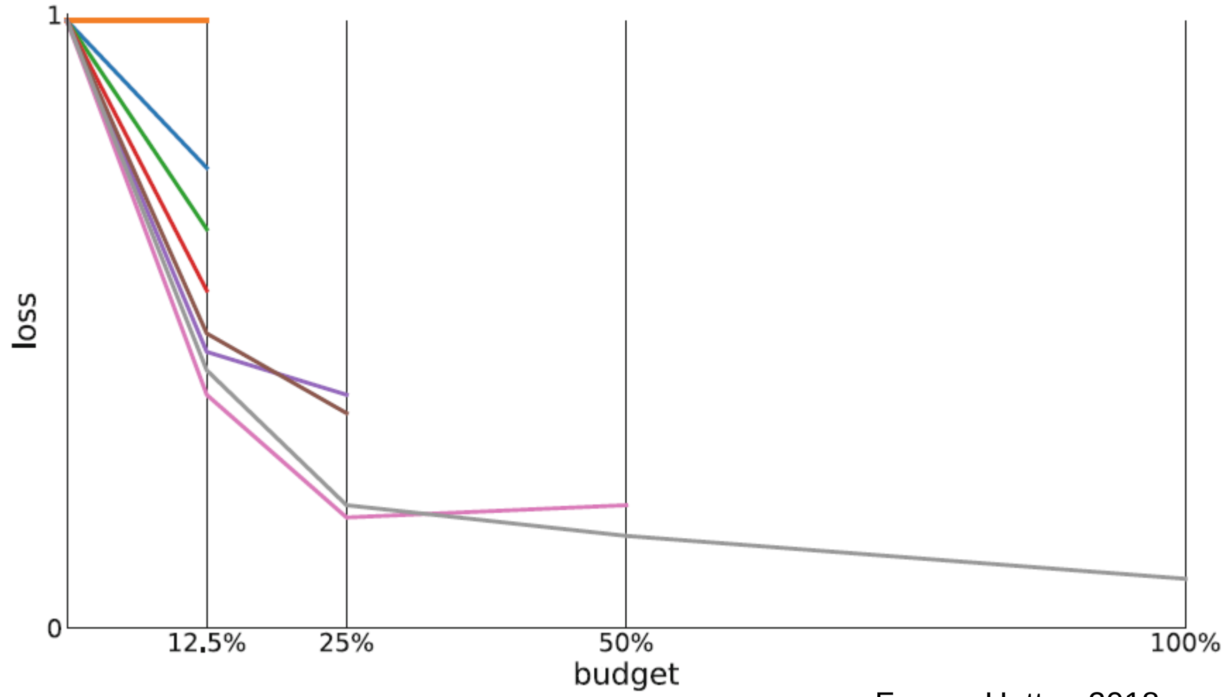
- Sparse Gaussian processes (Hutter et al. 2010)
- Neural nets combined with Bayesian linear regression (Snoek et al. 2015)
- Bayesian neural nets sampled with MCMC (Springenberg et al. 2016)
- Random forests (Hutter et al. 2011)
- Tree parzen estimator (Bergstra et al. 2011)

SUCESSIVE HALVING

Fully training ML algorithm does not allow many risk evaluations

Multi fidelity approach: Allow reduced training/modelling (e.g. train on a subset of data, train a few epochs etc.)

Sucessive halving: Start with low fidelities and many HP configurations
Sort out low performant instances and continue with remaining instances



Feurer, Hutter, 2018

HYPERBAND

Downside of sucessive halving: Budget (number of HP instances) is fixed

HyperBand: Divides the total budget into several combinations of number of configurations vs. budget for each, to then call successive halving as a subroutine on each set of random configurations.

BO + HyperBand: Select the HP configuration in HyperBand according to BO

META-LEARNING

Meta-Learning: Build a ML method based on data from other ML methods (e.g. loss function values, gradients etc.)

Application to HPO: Learn a ML model to learn HP based on the optimizee

Example: Consider the SGD update for the model f_w

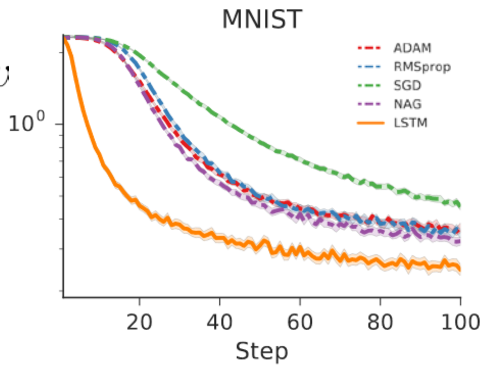
$$w^{t+1} = w^t - \eta_t \nabla L(w^t)$$

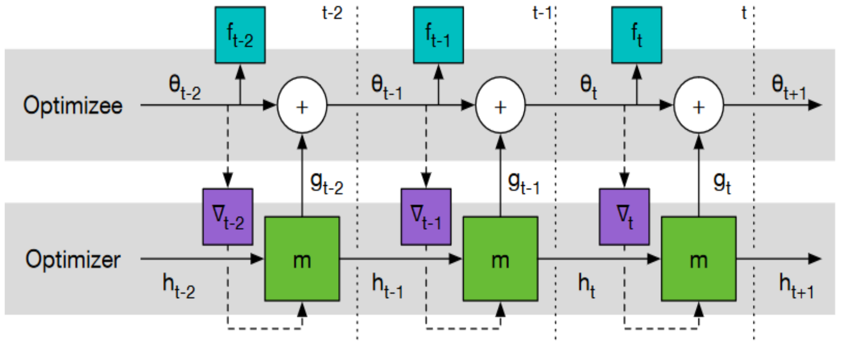
as a map of ML model

$$w^{t+1} = g_\phi(\nabla L(w^t))$$

Train by f_w minimizing

$$L(\phi) = \sum_{j=1}^{T} L(w_j(\phi))$$





Andrychowicz et al. 2016

CHALLENGES IN HPO

Benchmarks and Comparability

Scalability

Overfitting

Meta-Learning

Arbitrary sized ML pipelines