HYPERPARAMETER OPTIMIZATION

HELMHOLTZ ANALYTICS FRAMEWORK

DATA ANALYSIS METHODS WORKSHOP | KARLSRUHE | KIT

MARCH 23. 2018 I KAI KRAJSEK | JUELICH SUPERCOMPUTING CENTRE

OUTLINE

- Hyperparameters
- Cross validation
- Learning curve
- Grid search
- Random search
- Bayesian optimization
- Demo

HYPERPARAMETER OPTIMIZATION METHODS

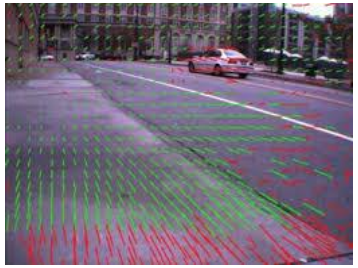
- **Evidence framework (Empirical Bayes)**
- Bayesian information criterion
- Akaike information criterion
- Fully Bayesian approach
- **Grid search – cross validation**
- **Random search – cross validation**
- **Bayesian optimization**
- „Leave one out" cross validation

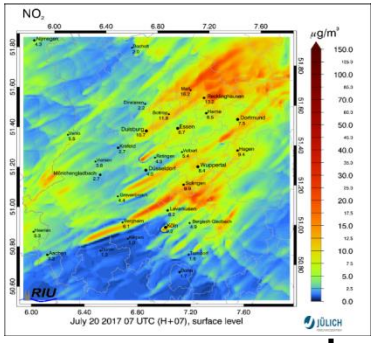
SUPERVISED LEARNING

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

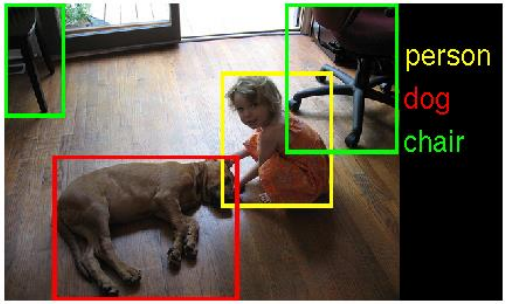
Application examples:

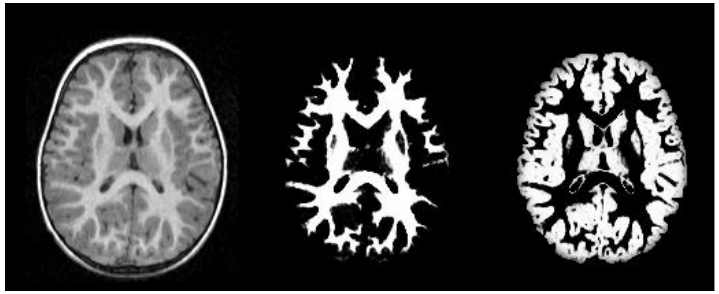
- **Regression:** Optical flow estimation, predicting stock prices, chemical weather forecasting





- **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:

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

Loss function

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

➡ Approximation of $R(f)$ required

MODEL SELECTION

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

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

Decisions belong to model/hyperparameter selection

LAW OF LARGE NUMBERS

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

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

Properties of **estimators:**

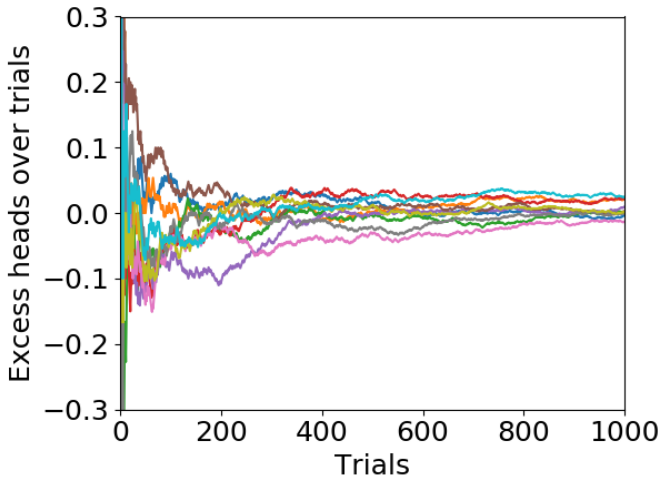
Bias:

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

Variance:

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

Example: Coin-toss experiment



LINEAR REGRESSION

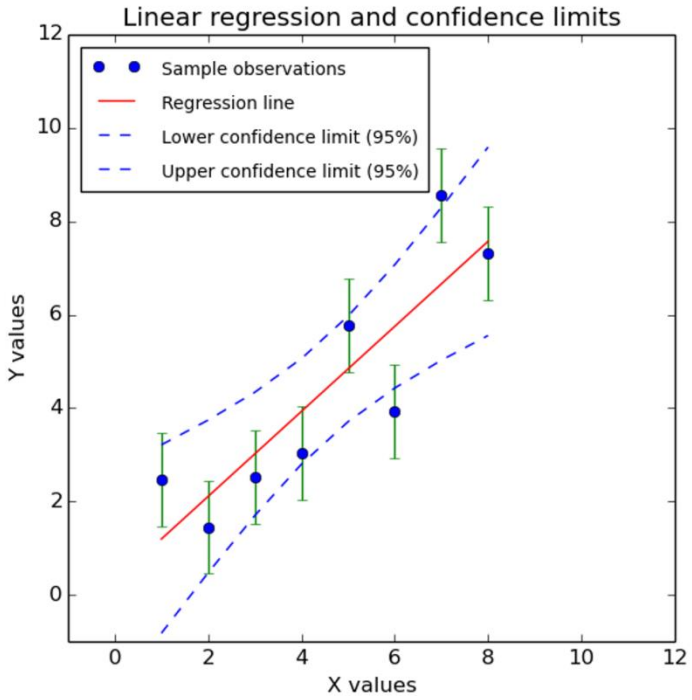
Example: Linear regression

$$y = ax + b, \quad \Theta = (a, b) \qquad L(x) = x^2$$

Emphirical risk:

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

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



EXAMPLE: POLYNOMIAL REGRESSION

Example: Polynomial regression

$$y = a_0 + a_1 x + ... + a_q x^q = \vec{a}^T \vec{x_q},$$

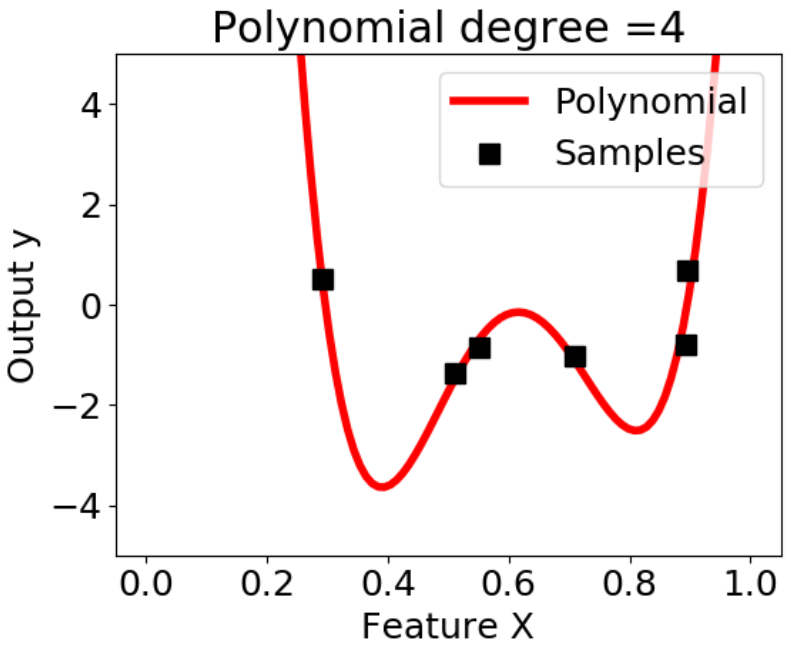
$$\Theta = (\vec{a} \in \mathbb{R}^q, q),\ L(\varepsilon) = \varepsilon^2$$

$$\vec{x}_q := (1, x, ..., x^q)$$

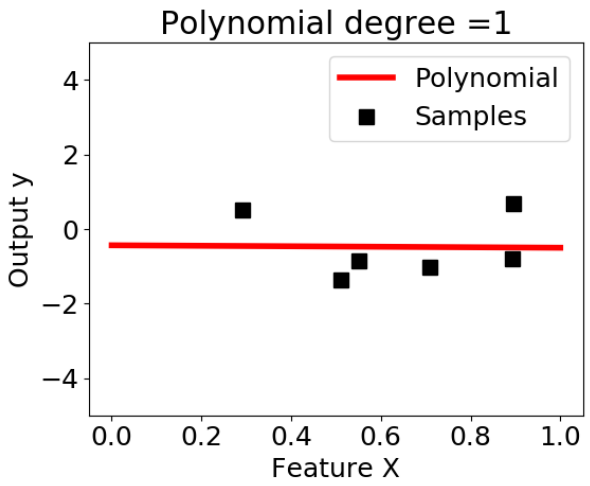
Emphirical risk:

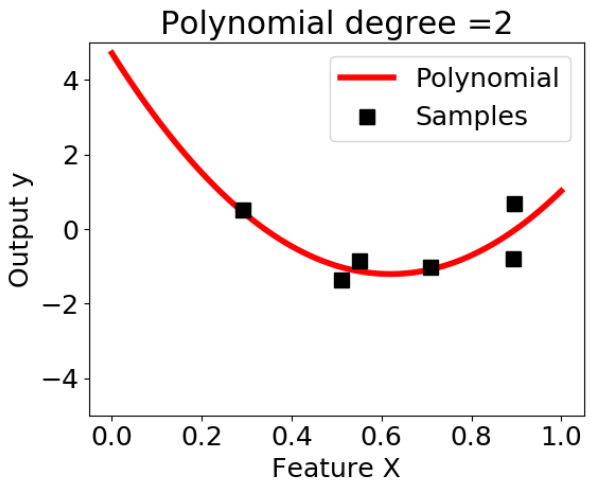
$$\hat{R}_N(\vec{a}) := \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \vec{a}^T \vec{x}_n\right)^2$$

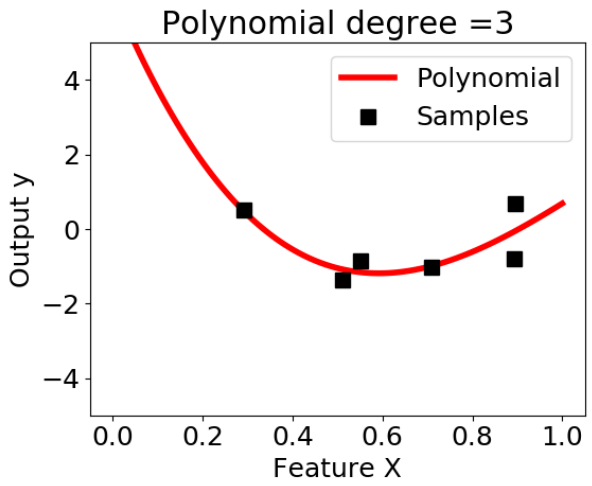
$$\hat{\vec{a}} = \arg\min_{\vec{a}} \hat{R}(\vec{a})$$

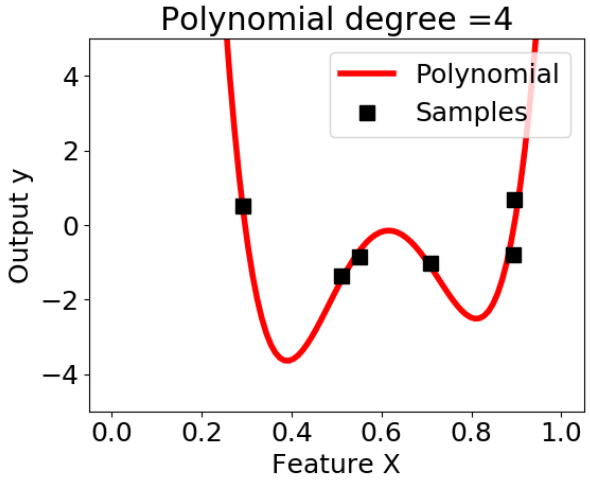


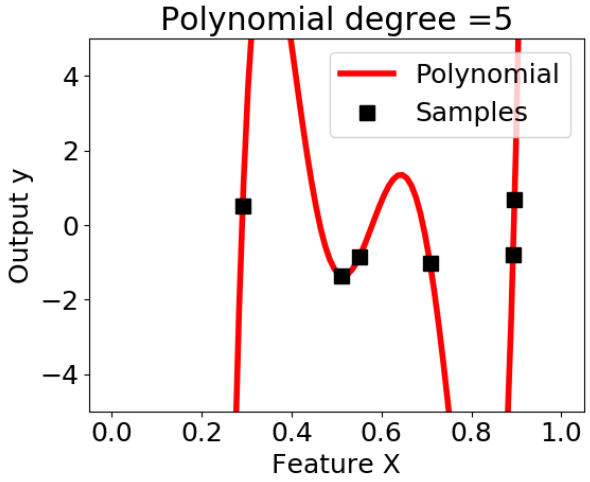
EXAMPLE: POLYNOMIAL REGRESSION

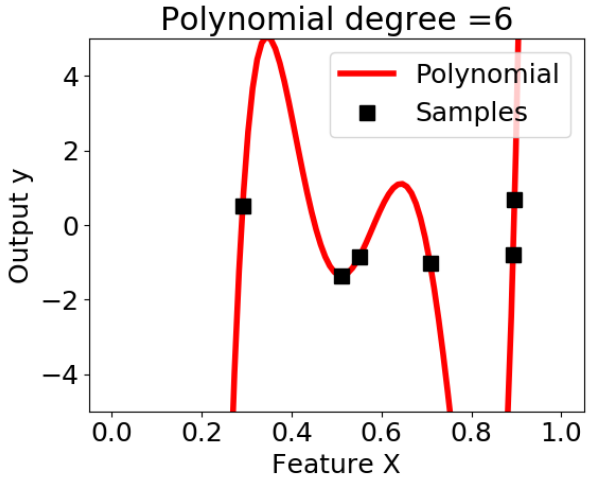




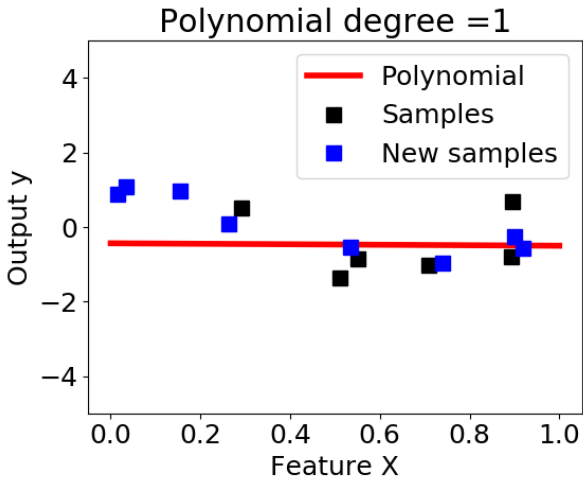


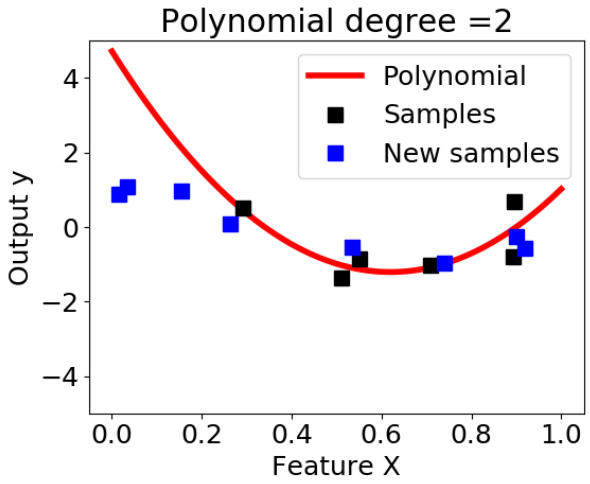


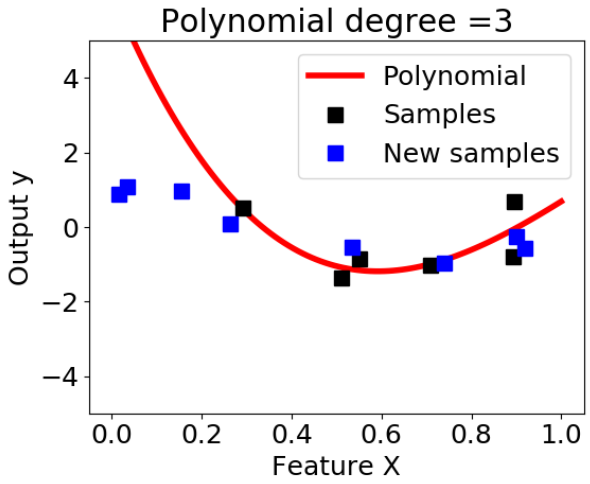


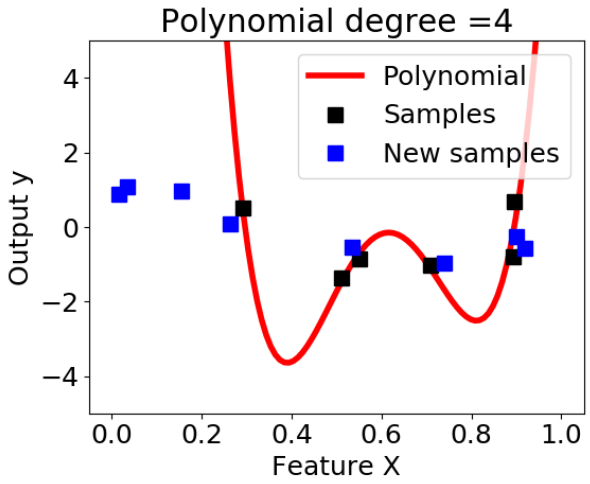


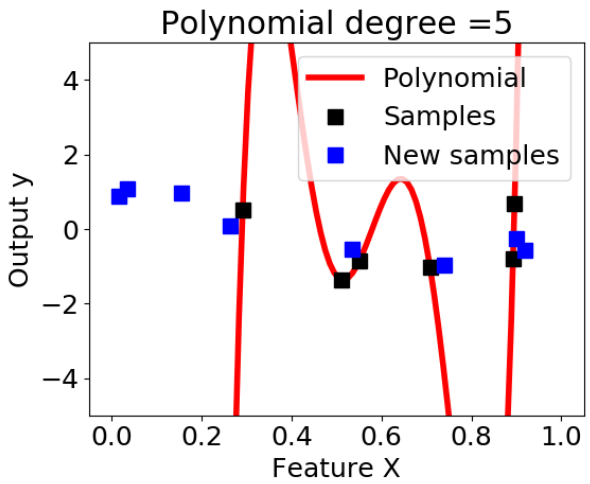
EXAMPLE: POLYNOMIAL REGRESSION

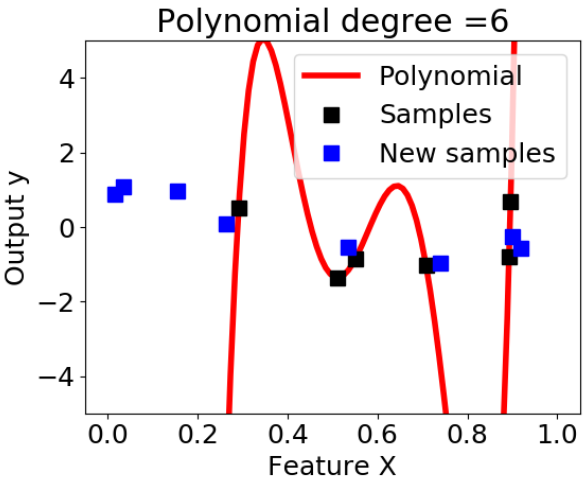






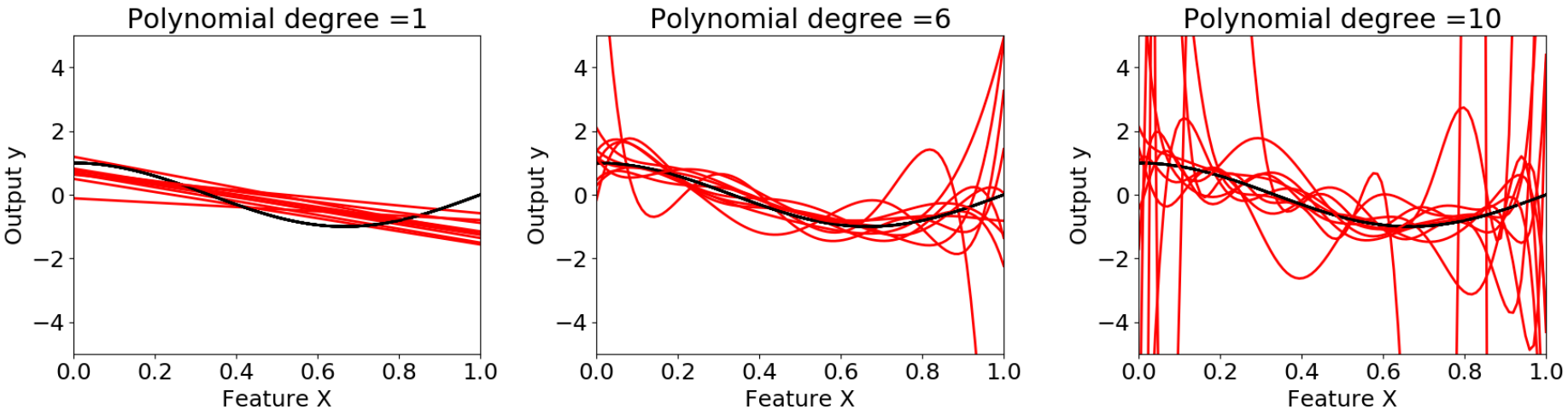






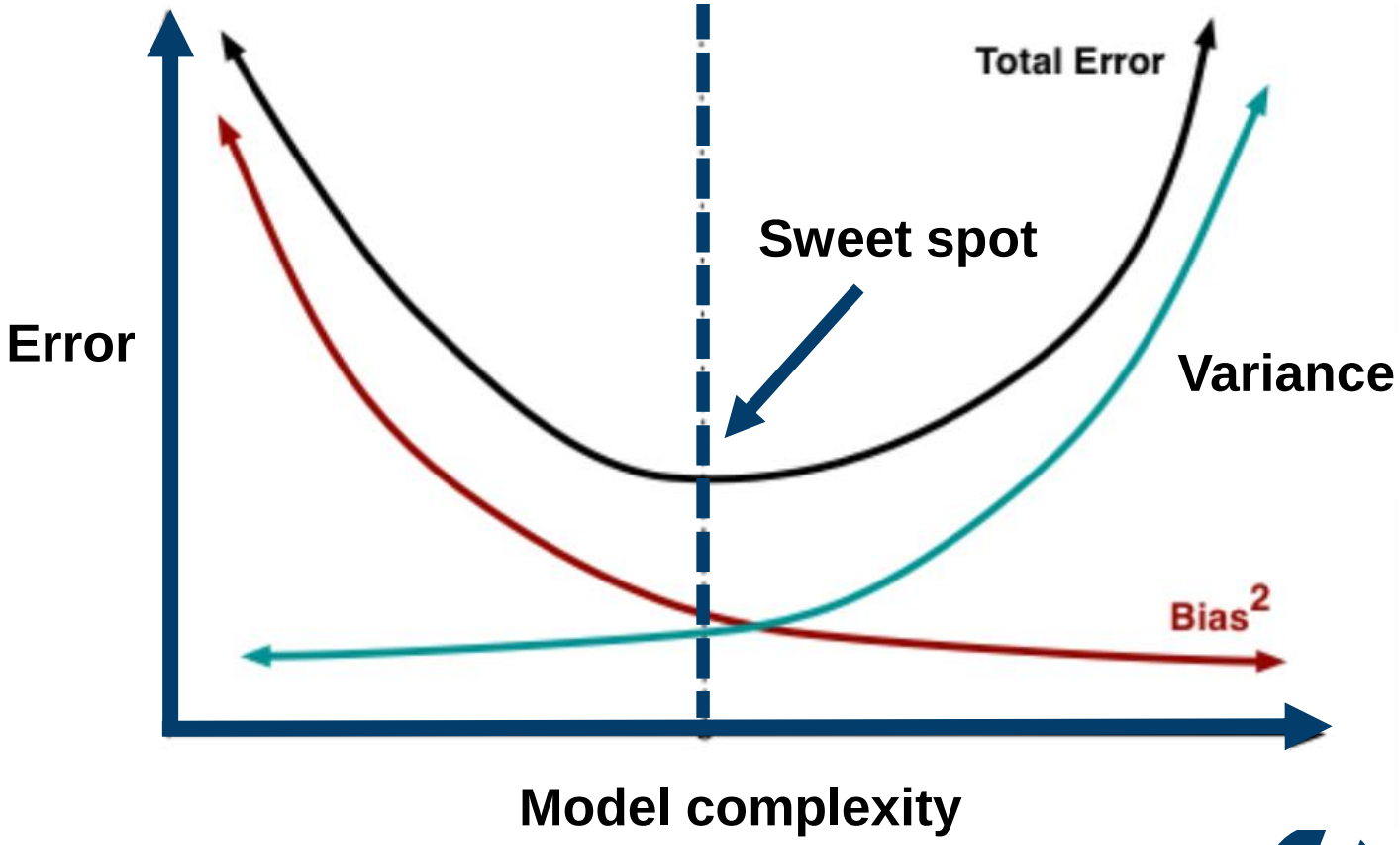
BIAS – VARIANCE TRADEOFF

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



Hyperparameter: Parameters determine the capacity/complexity of the model, parameters not part of the prediction model, *e.g.* optimization parameter, data-preprocessing etc.

BIAS-VARIANCE TRADEOFF



BIAS – VARIANCE TRADEOFF

Estimation theory: It can be shown that for quadratic loss functions and mean squared error:

$$\hat{f} = \arg\min_{f \in \Lambda} R(f) = \int yp(y|x)dy = \mathbb{E}\left[y|x\right]$$

The mean squared error (MSE) can be decompoised as:

MSE Bias Variance

$$\mathbb{E}\left[(f(x) - \mathbb{E}\left[y|x\right])^2\right] = (\mathbb{E}[f(x)] - \mathbb{E}[y|x])^2 + \mathbb{E}\left[(f(x) - \mathbb{E}[f(x)])^2\right]$$

A similar decomposition can be derrived for clasification scores (see e.g. *Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich (2008). Introduction to Information Retrieval. Cambridge University Press. pp. 308–314.*)

BIAS – VARIANCE TRADEOFF

Φ : Parameter space Θ : Hyperparameter space

Ridge polynomial regression

$$y = \vec{a}^T \vec{x_q}, \;\; L(\varepsilon) = \varepsilon^2 + \lambda \|\vec{a}\|^2 \qquad \Phi = \vec{a} \qquad \Theta = (q, \lambda)$$

Logistic ridge polynomial regression

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

σ : Sigmoid function $\Phi = \vec{a}$ $\Theta = (q, \lambda)$

EXAMPLE OF HYPERPARAMETERS

Φ : Parameter space Θ : Hyperparameter space

Neural networks

$$y = g_N(g_{N-1}((...g_1(x)))) \qquad \Phi = \{A_i, b_i\}$$

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

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

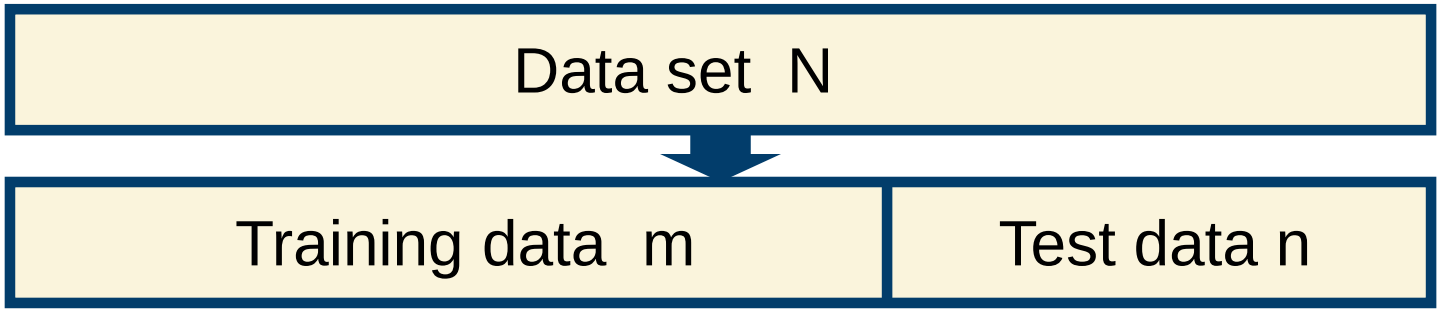
Multi-label classification

Neural network hyperparameter Θ :

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

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

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

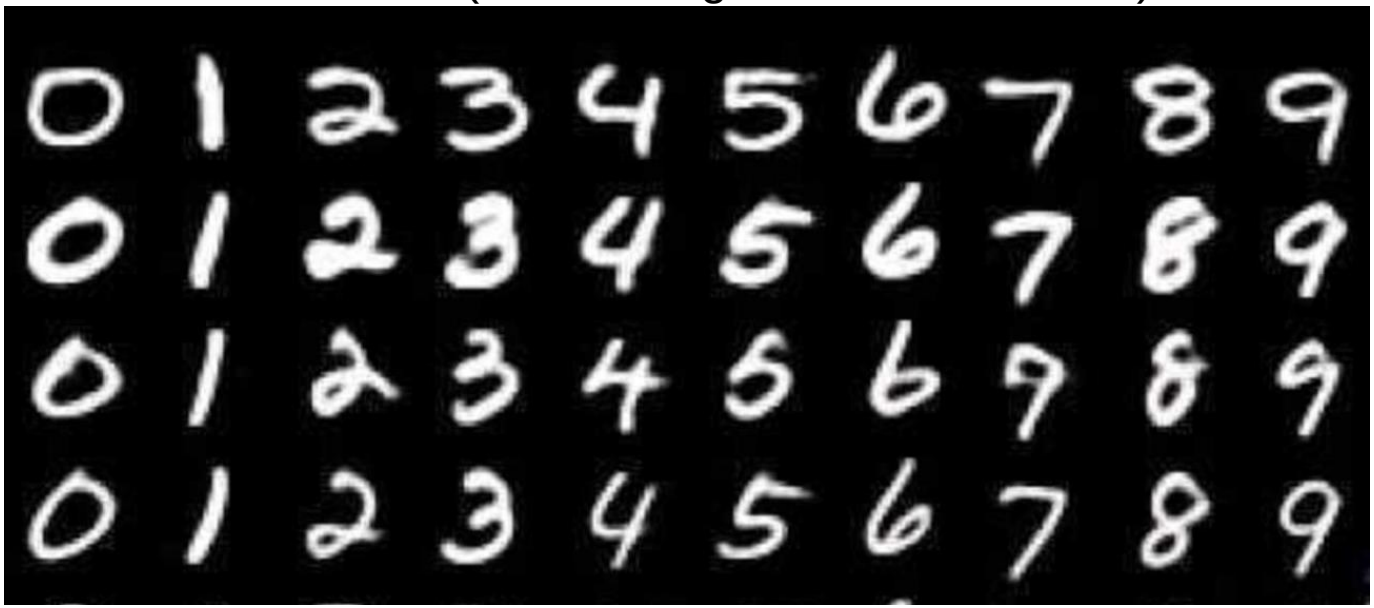
HOLDOUT METHOD

Example: Logistic ridge regression on MNIST Dataset

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

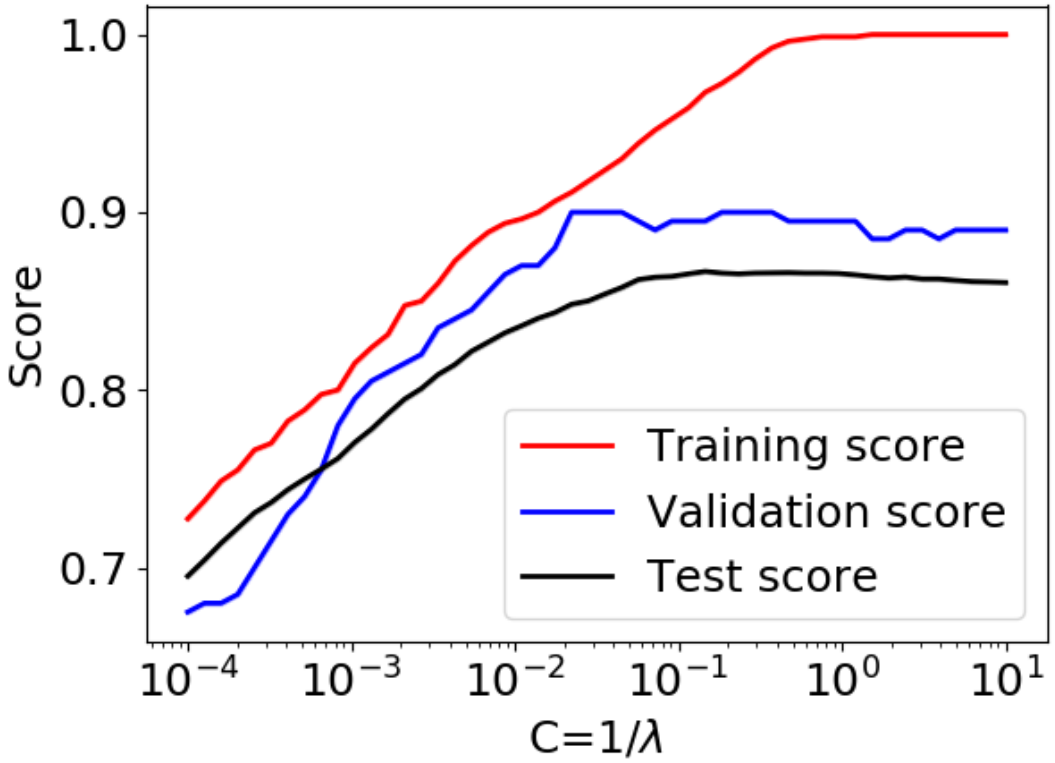
σ : Sigmoid function $\quad \Phi = \vec{a} \quad \Theta = \lambda \quad \vec{x} \in \mathbb{R}^{784}$

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



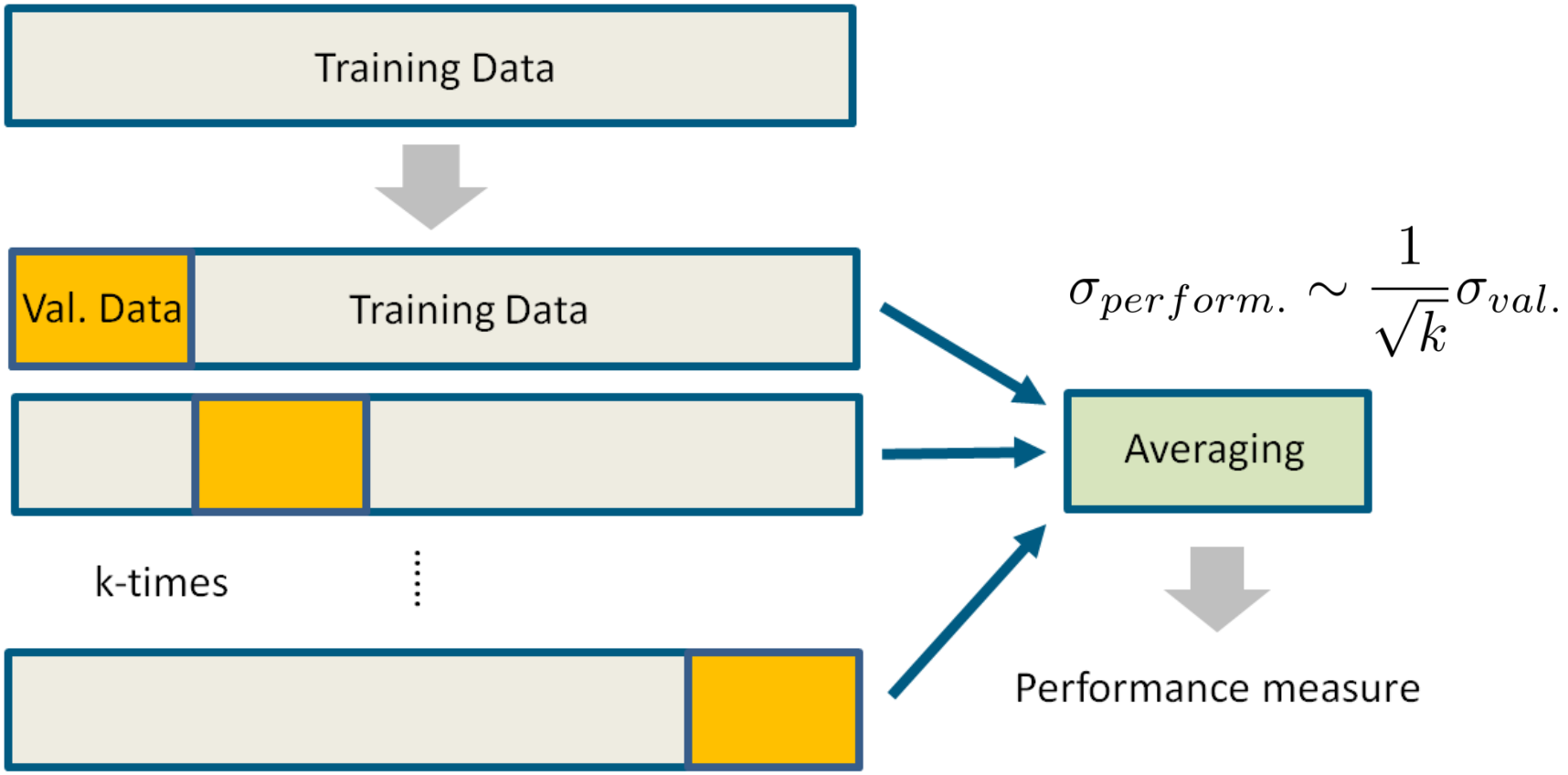
HOLDOUT METHOD

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



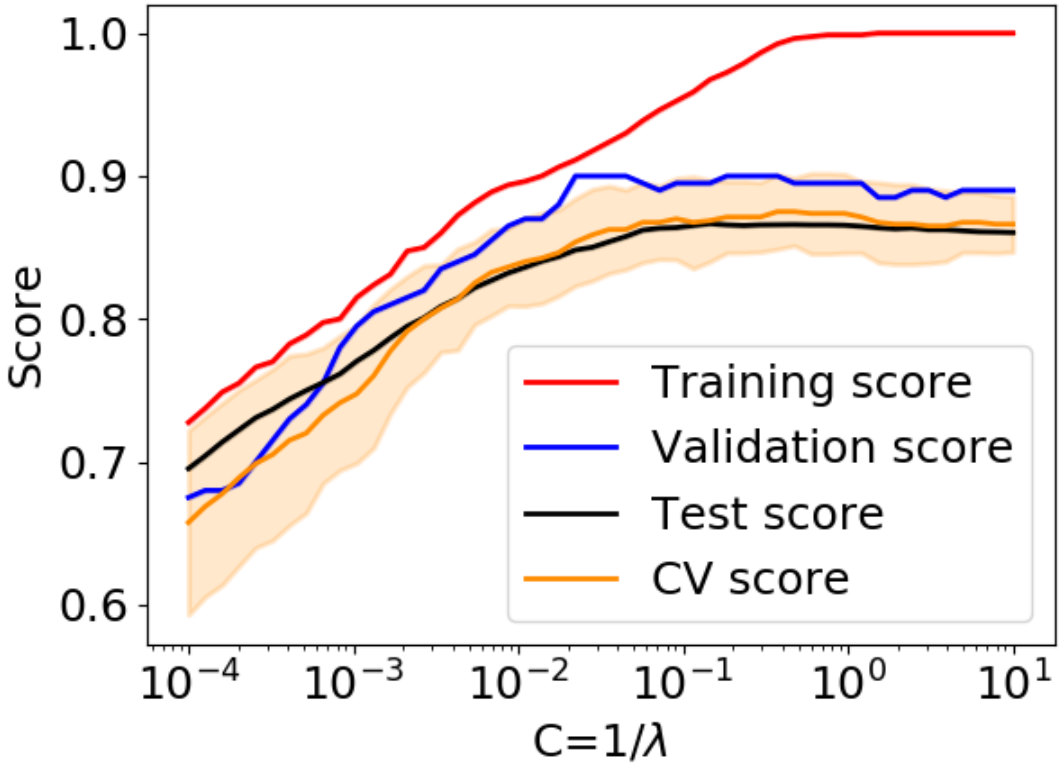
CROSS VALIDATION

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



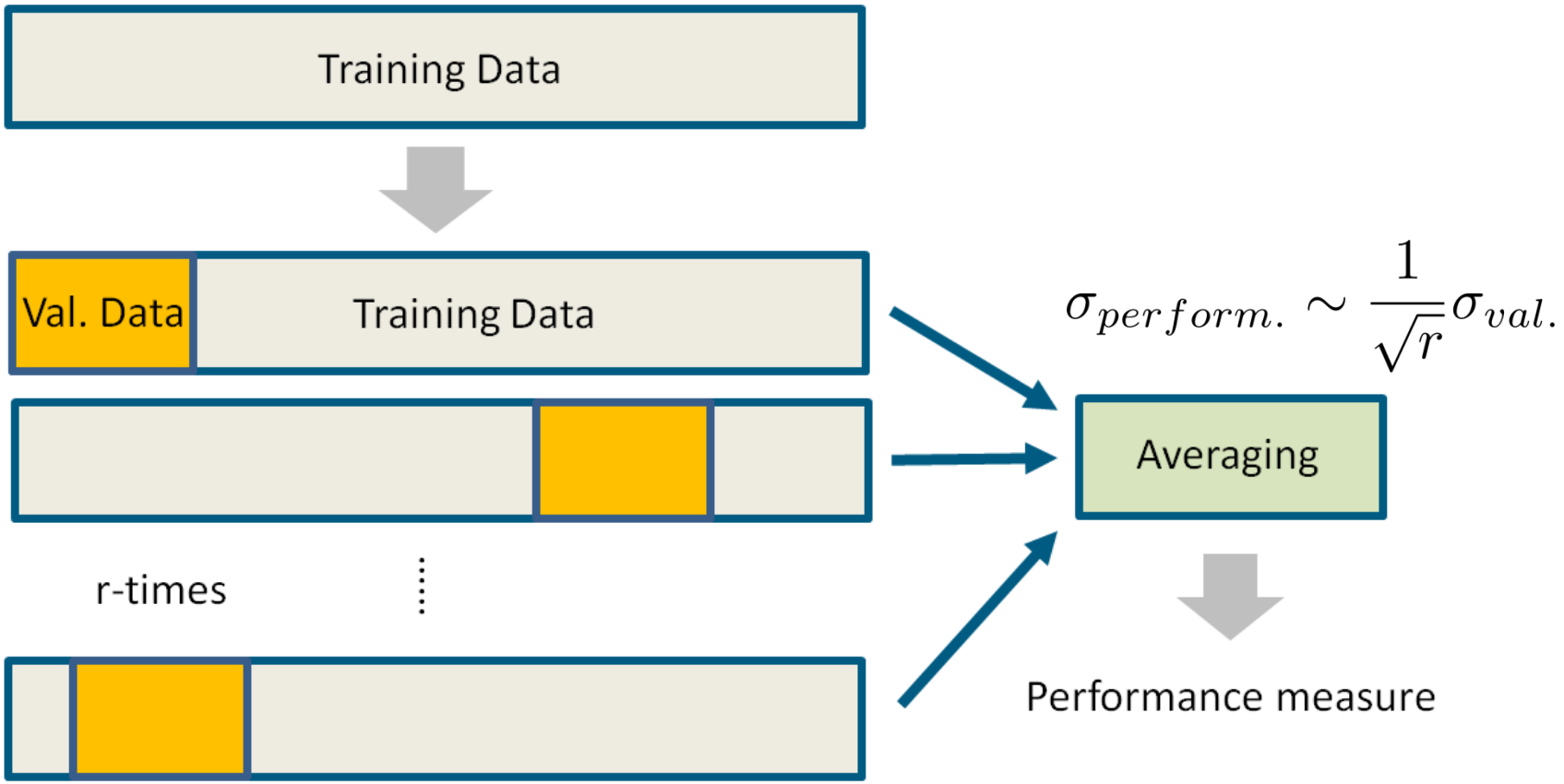
K-FOLD CROSS VALIDATION

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



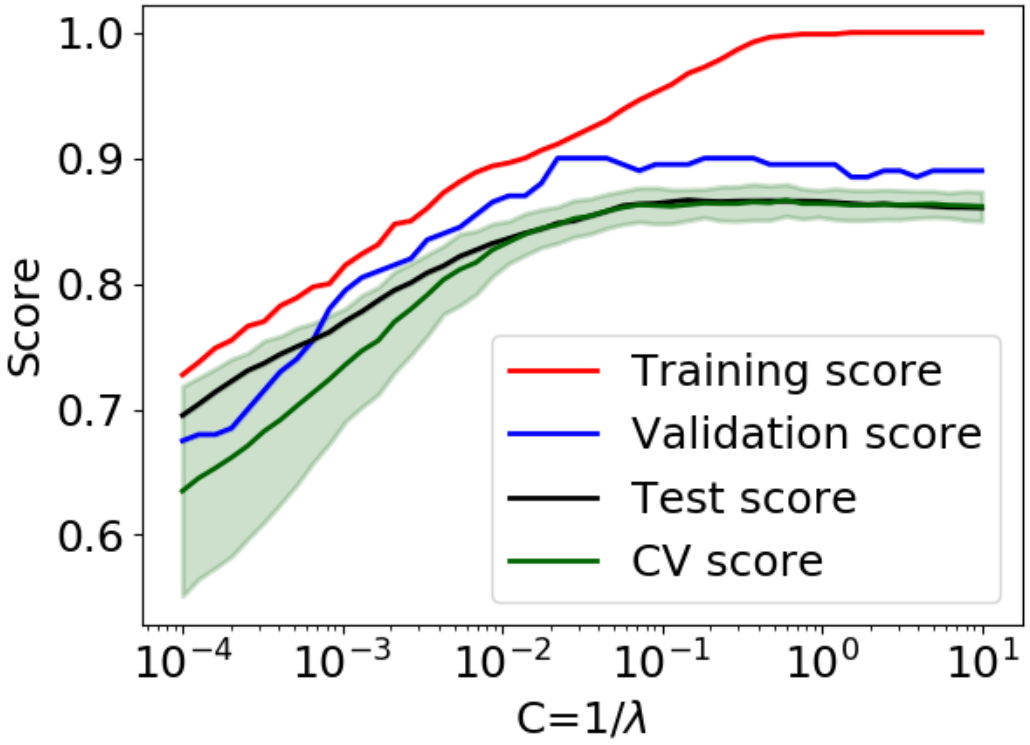
RANDOM SAMPLING

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



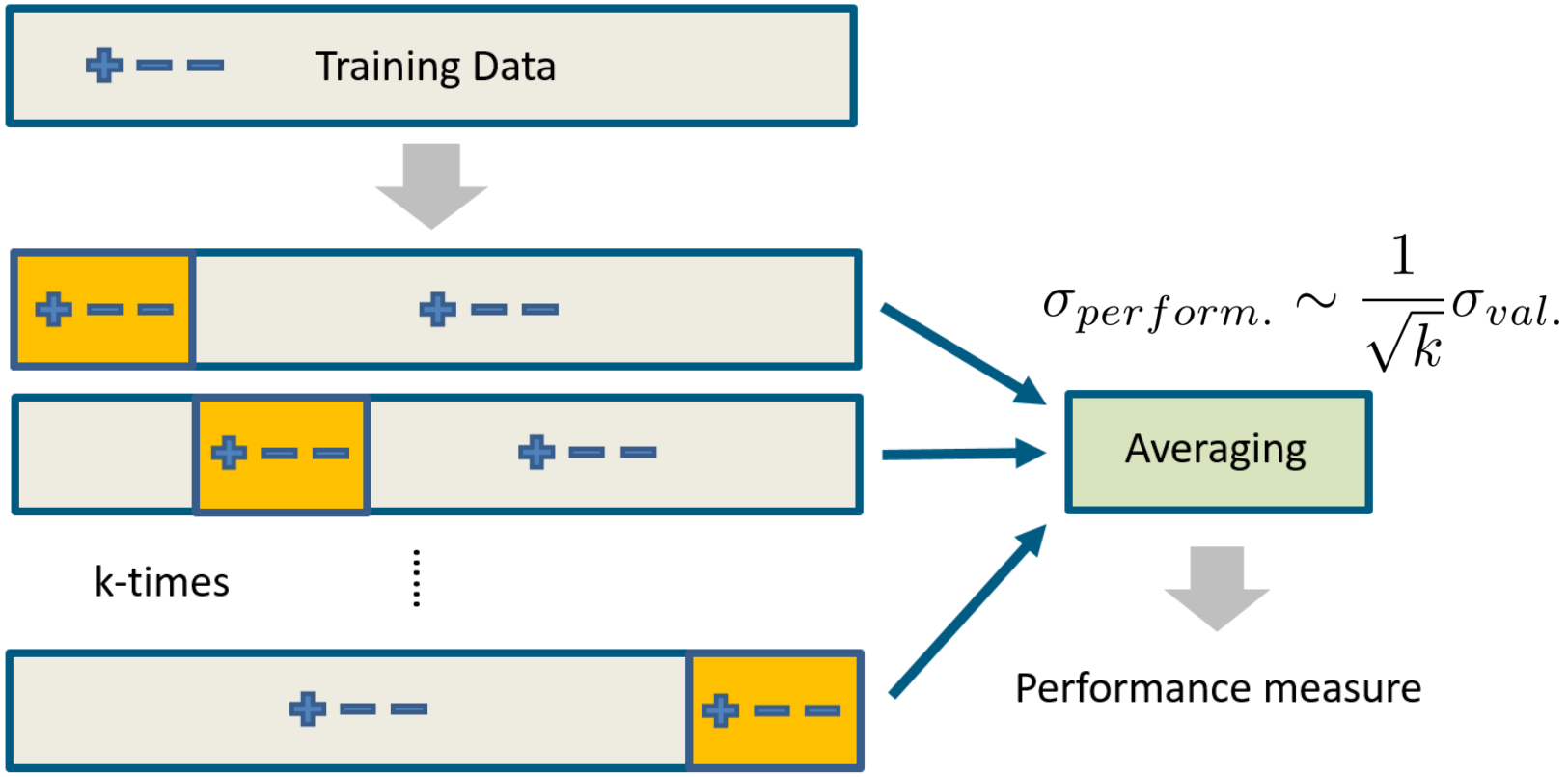
RANDOM SAMPLING

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

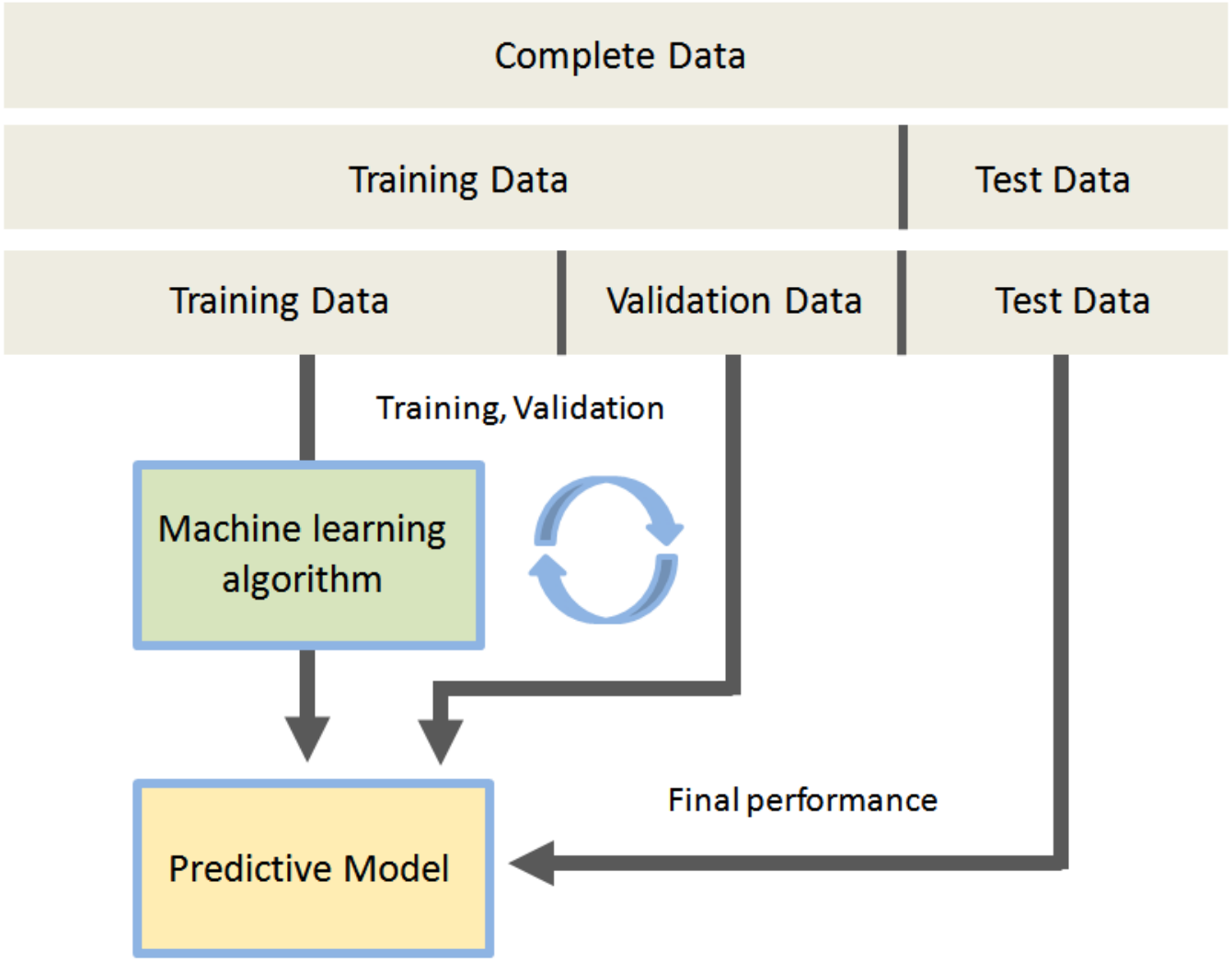


STRATISFIED CROSS-VALIDATION

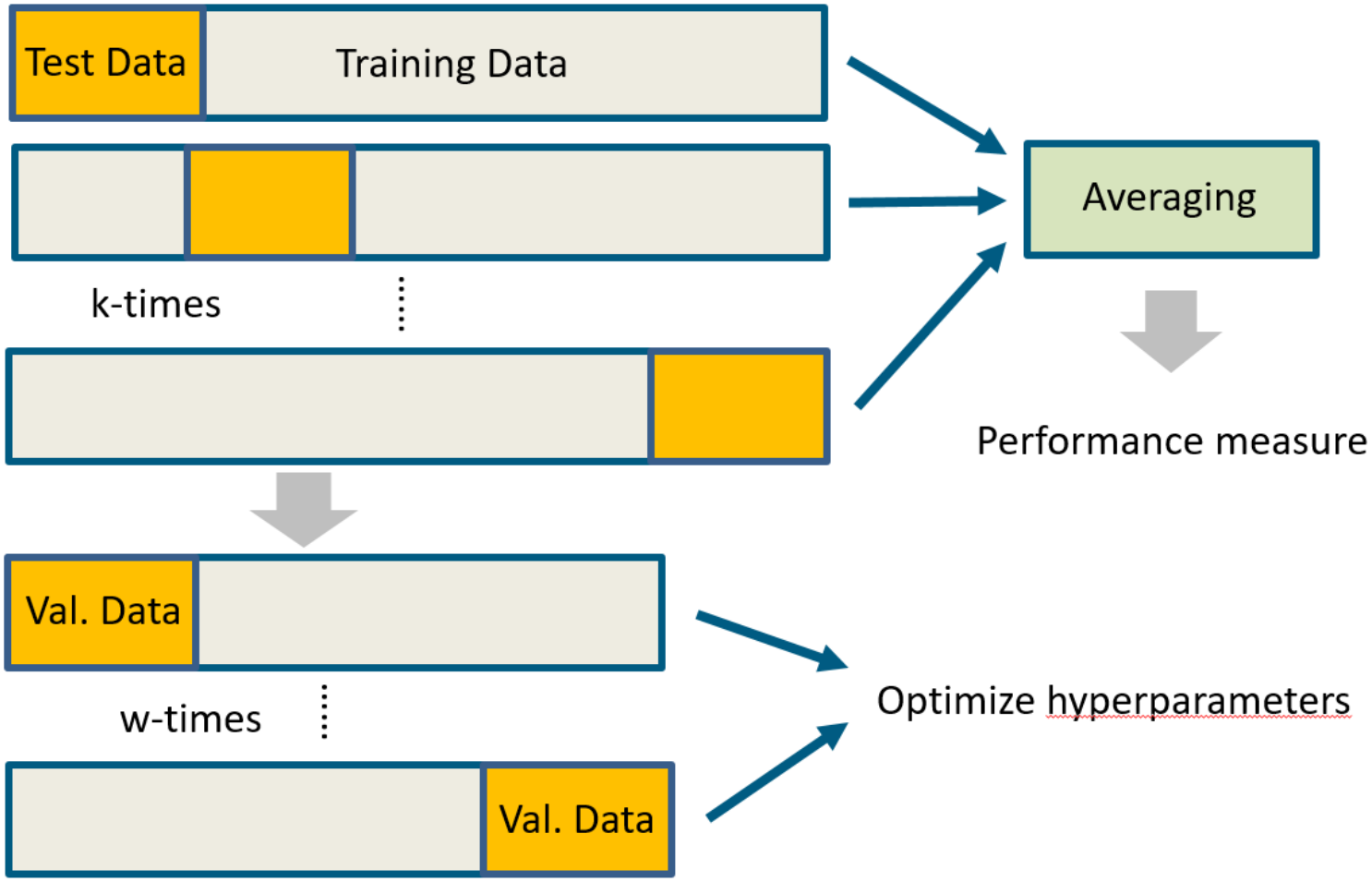
- Trainings data is divided into k non-overlapping patches (k-fold cross validation) with **stratified** folds



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

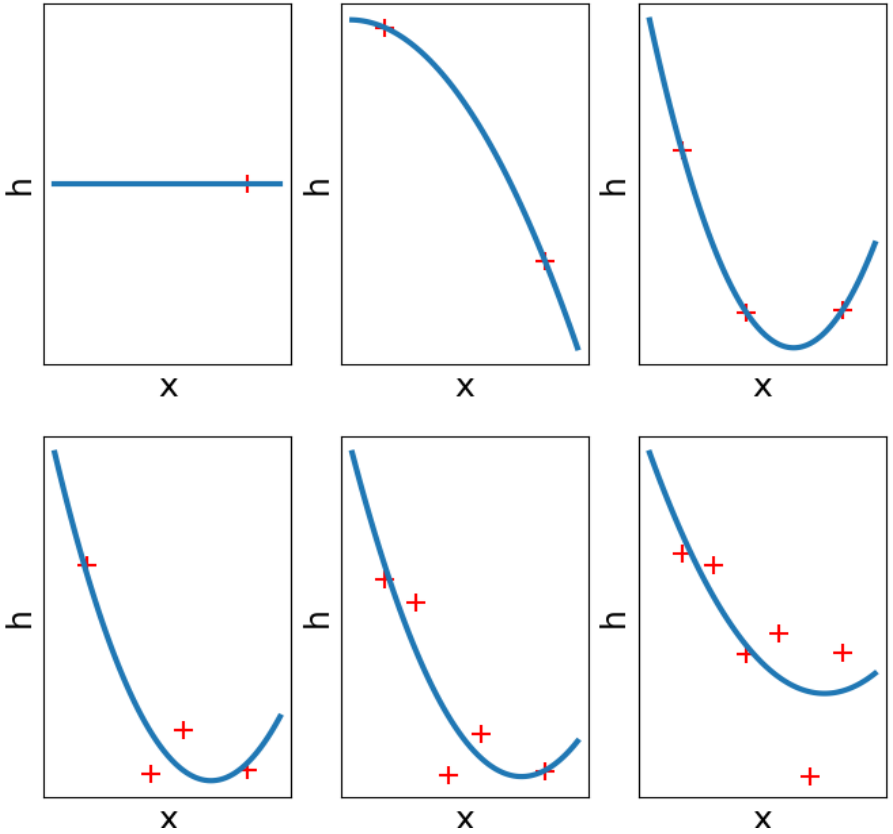
- optimal tradeoff bewtween training and validation/test data
- the need to obtain more data

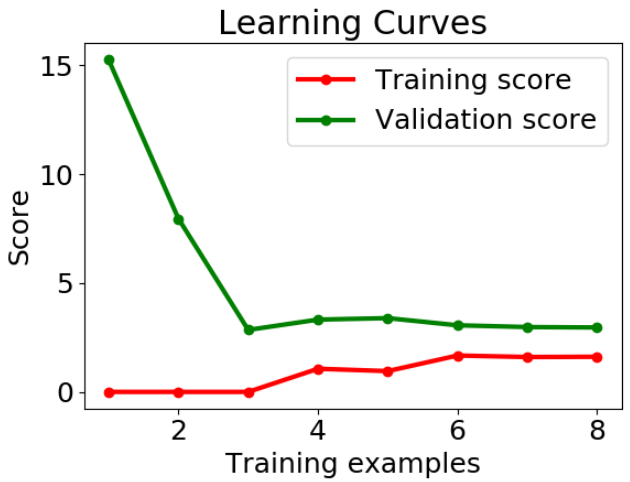
Idea: Train the model on subsets of the training data and observe ist performance as amount of data used for training increases

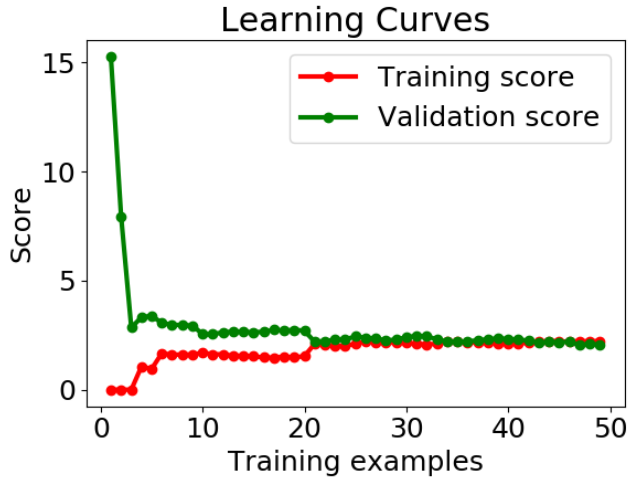
LEARNING CURVE

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

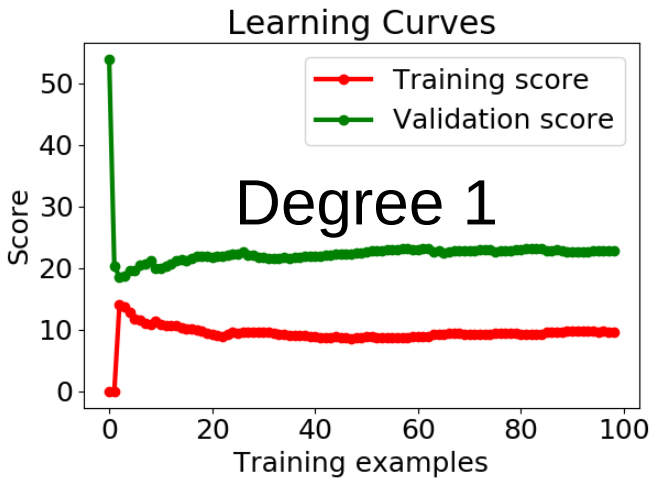
$$y = h_a(x) + \varepsilon$$

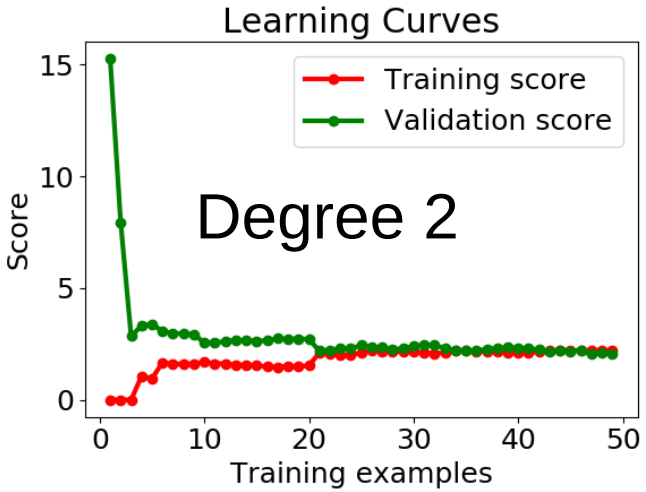


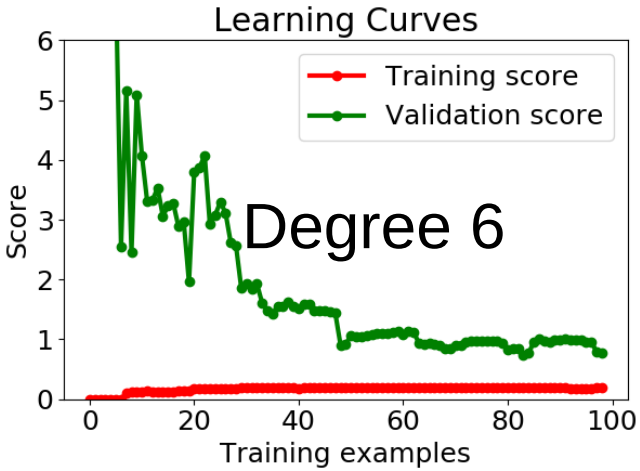


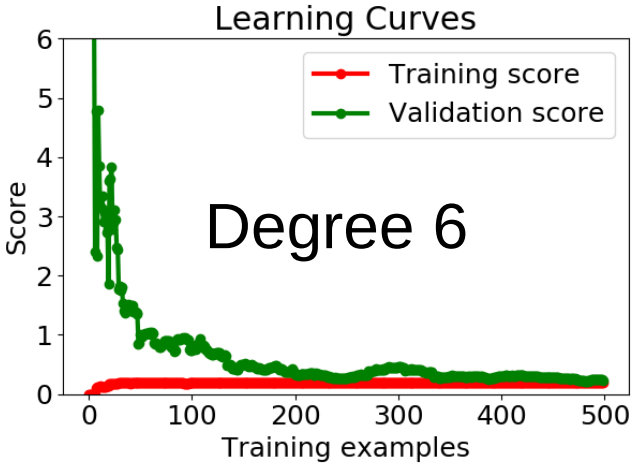


LEARNING CURVE



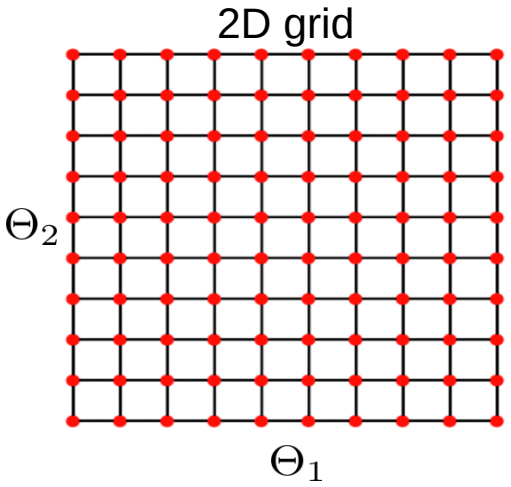






GRID SEARCH

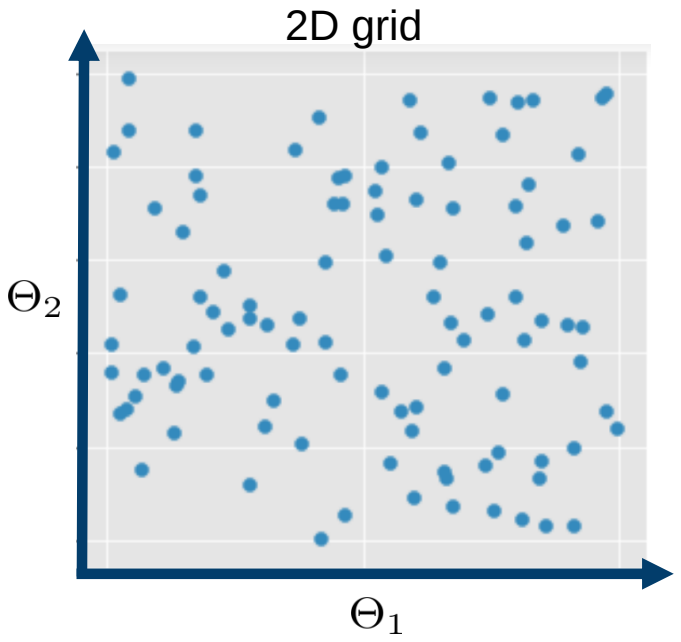
- Exhaustive search through a manually specified subset of the hyperparameter space
- Apply CV on each hyperparameter set
- Full parallelizable as model applied independently on each hyperparameter set
- Suffers from the curse of dimensionality, *e.g.* 10 hyper-parameter per dimension ➡ 10^{10} for dim=10



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

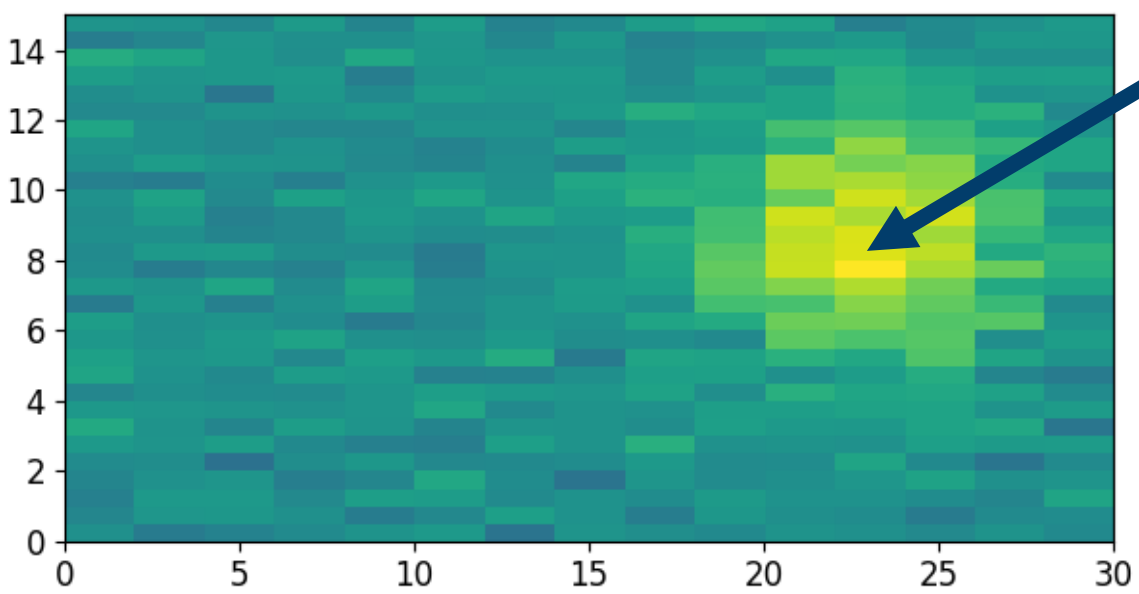
Probability not to hit the yellow area with 60 trials

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

Probability to hit the green area with 60 trials

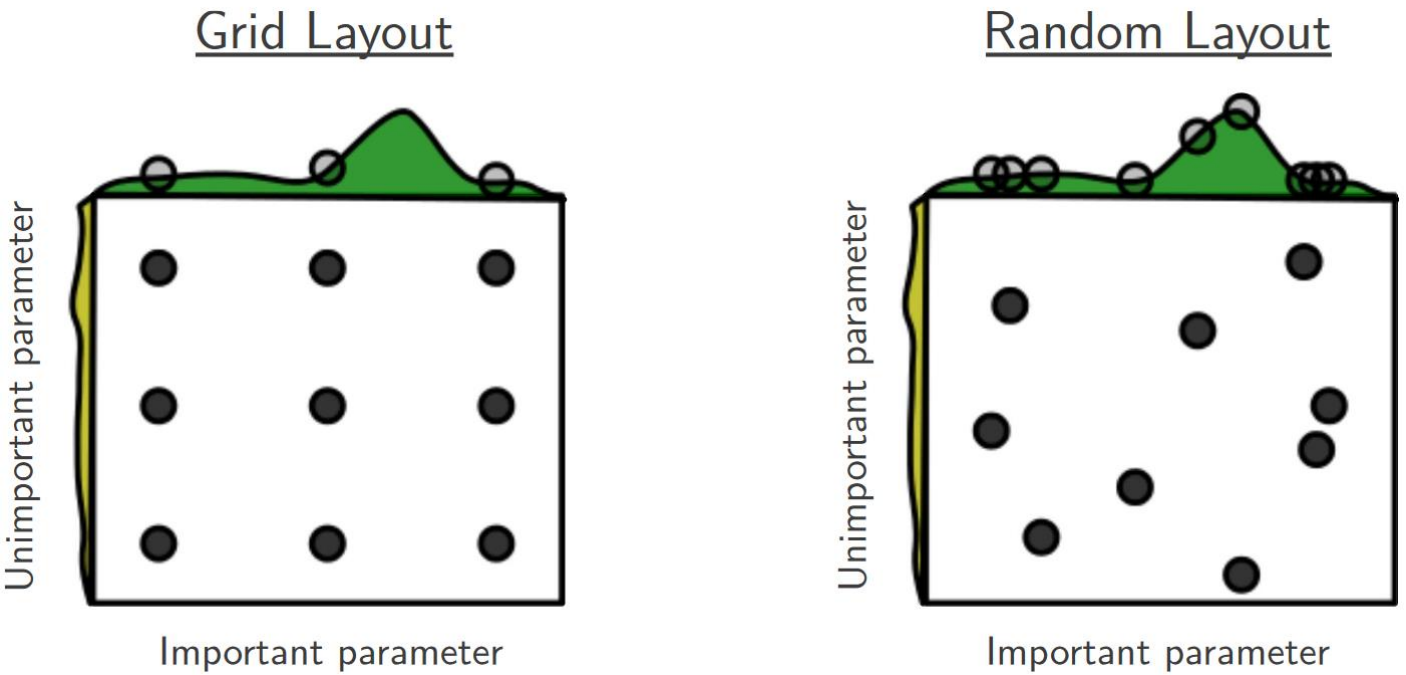
$$p = 1 - q \approx 94.3\%$$

5% highest scores



RANDOM SEARCH – GRID SEARCH

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



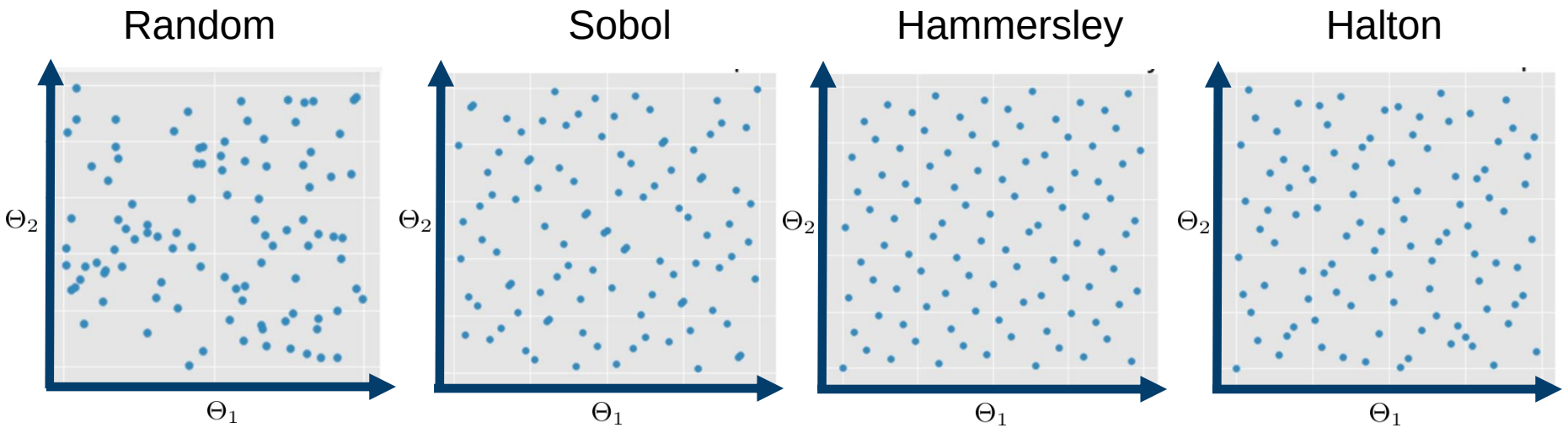
Bergstra and Bengio: Random Search for Hyper-Parameter Optimization, IJML, 2012

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

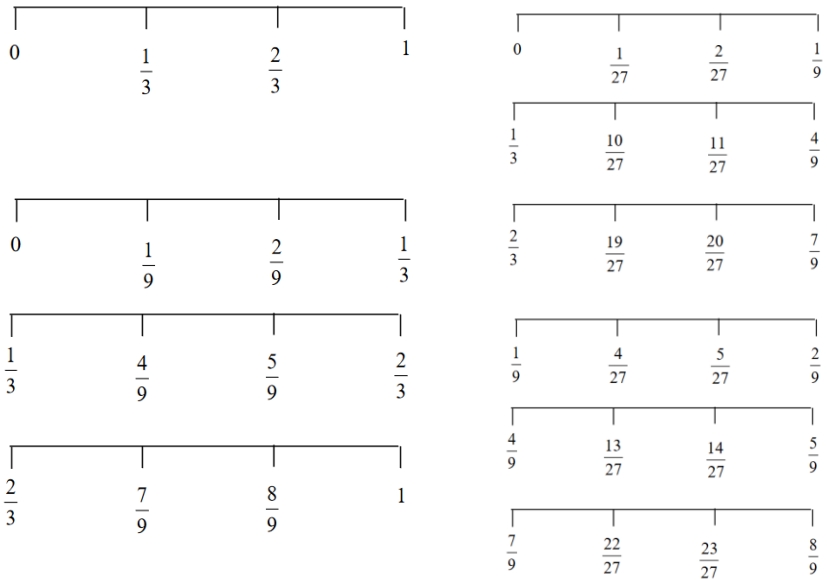


HALTON SEQUENCE

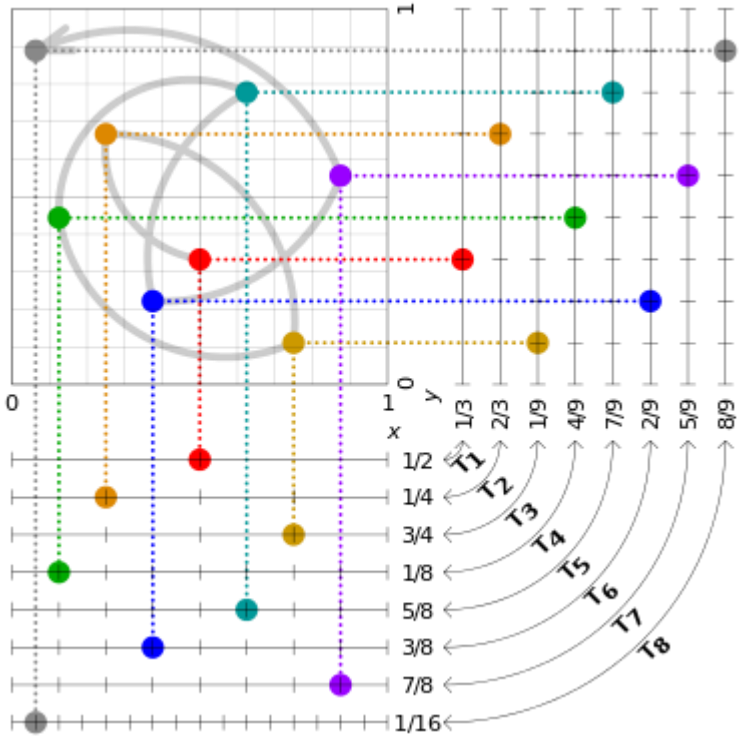
Sequence is constructed based on finer and finer prime based divisions of sub intervals of unit interval

Example: p=3

1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9,
1/27, 10/27, 19/27, 4/27, 13/27, 22/27,
7/27, 16/27, 25/27, 2/27, 11/27, 20/27,
5/27, 14/27, 23/27, 8/27, 17/27, 26/27



2D example with p=2,3



BAYESIAN OPTIMIZATION

We can evaluate the empirical risk pointwise, but do not have an easy functional form or gradients

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

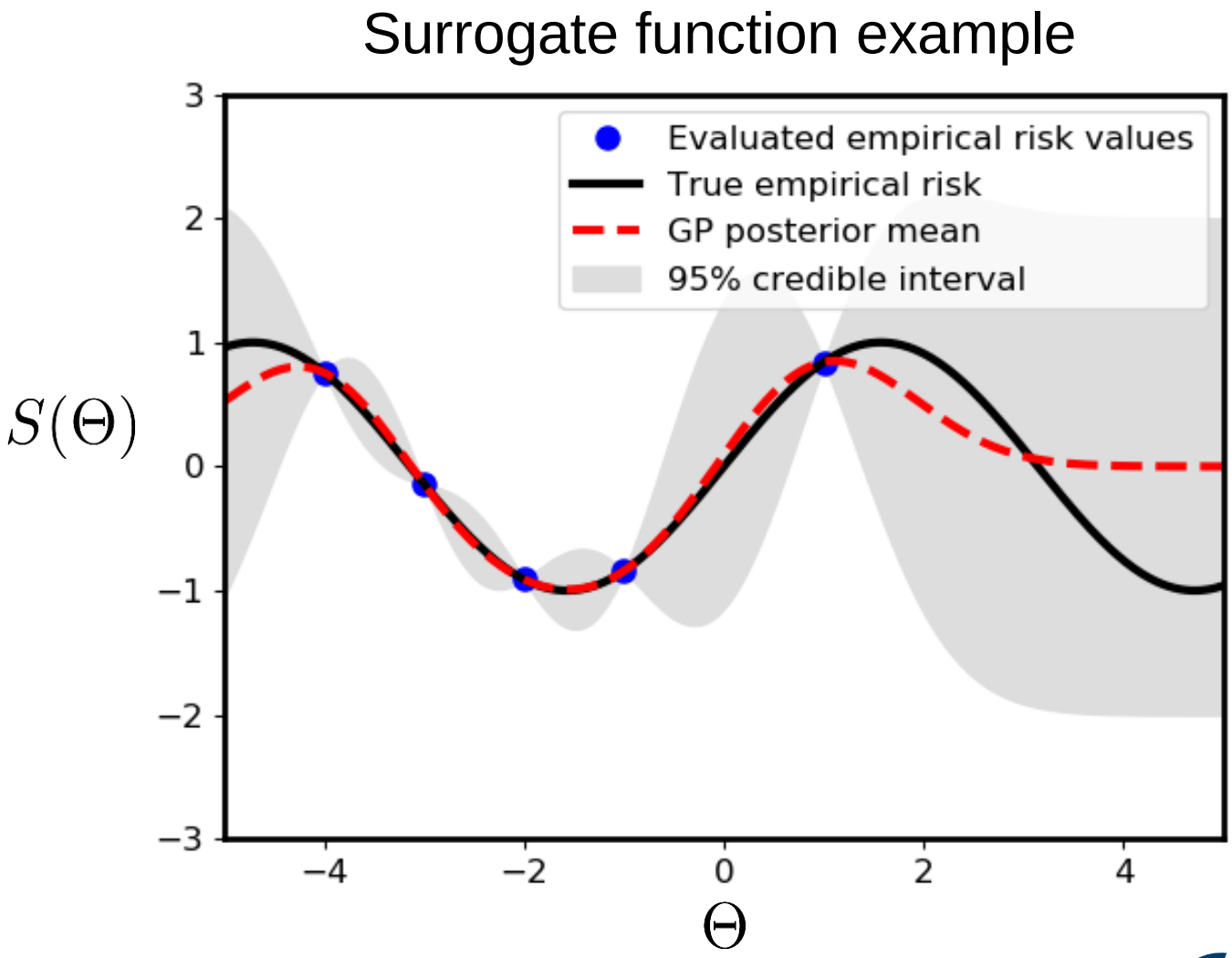
Idea: Learn a surrogate of the expected risk that is cheap to evaluate and provide an confidence interval to determine the next grid point

$$\hat{f} = \arg\min_{f \in \Lambda} S(f) \qquad \sigma^2\left(S(f)\right)$$

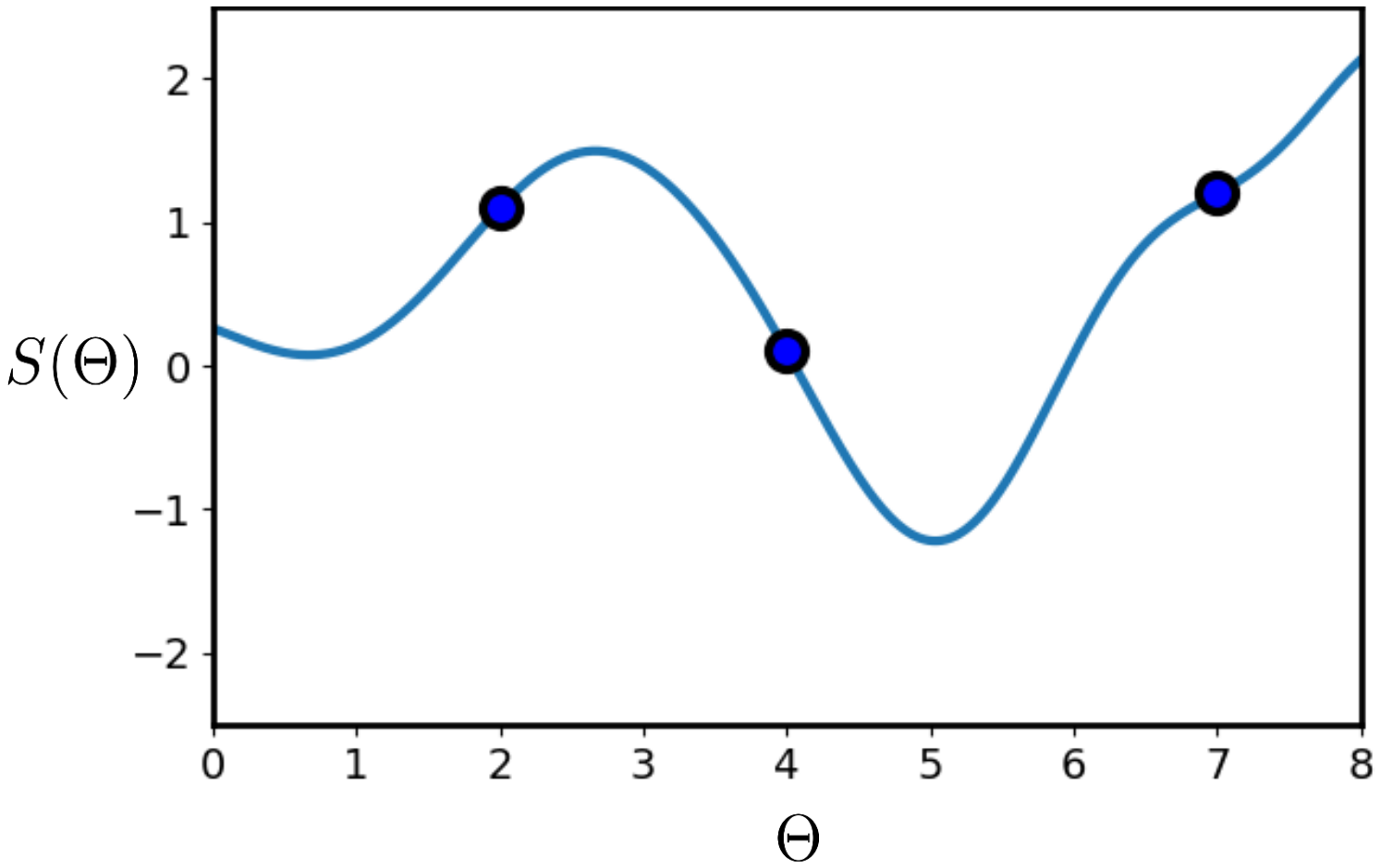
Search the hyperparameter space with

- **Exploration:** Seek places with high variance
- **Exploitation:** Seek places with low mean

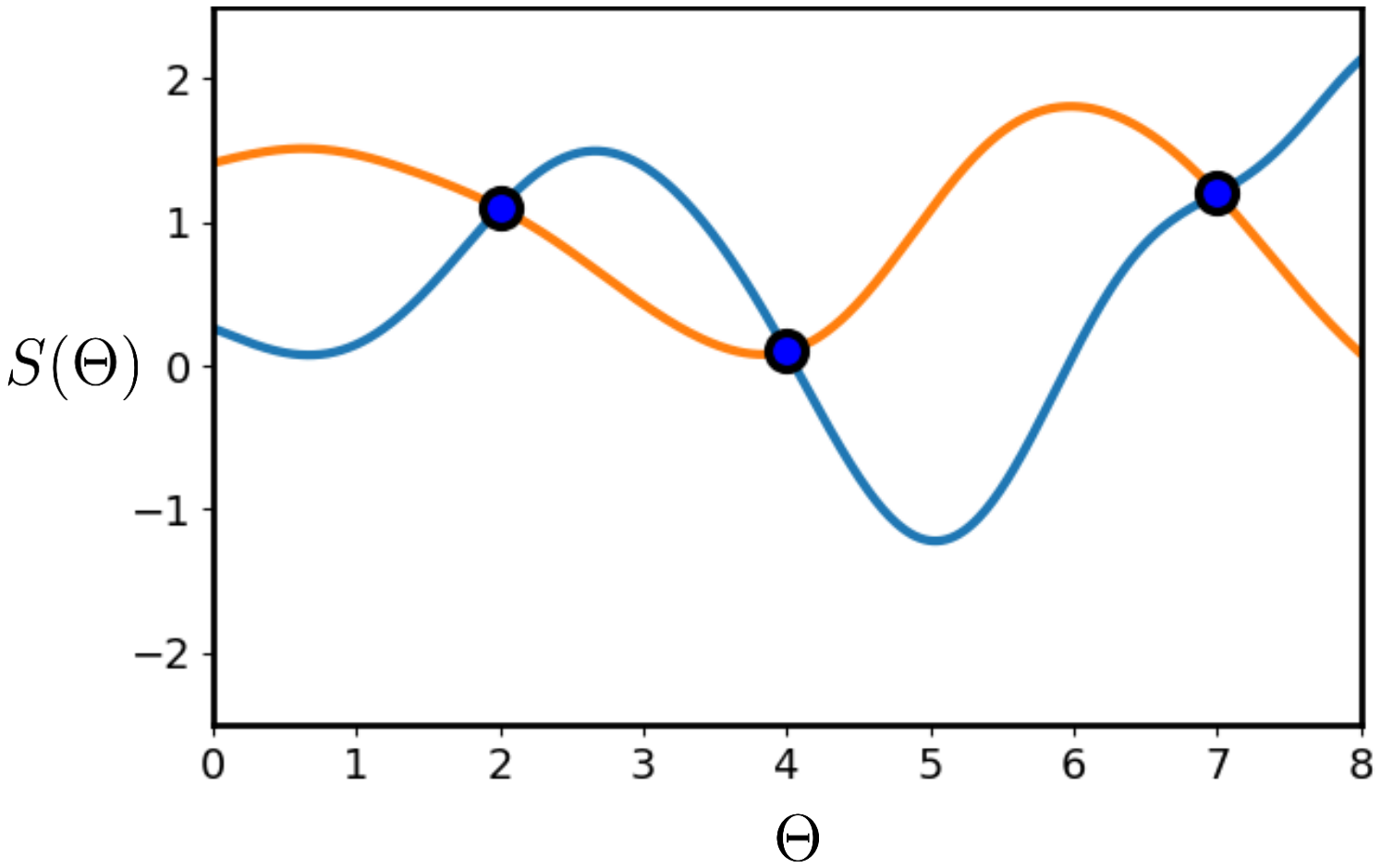
BAYESIAN OPTIMIZATION



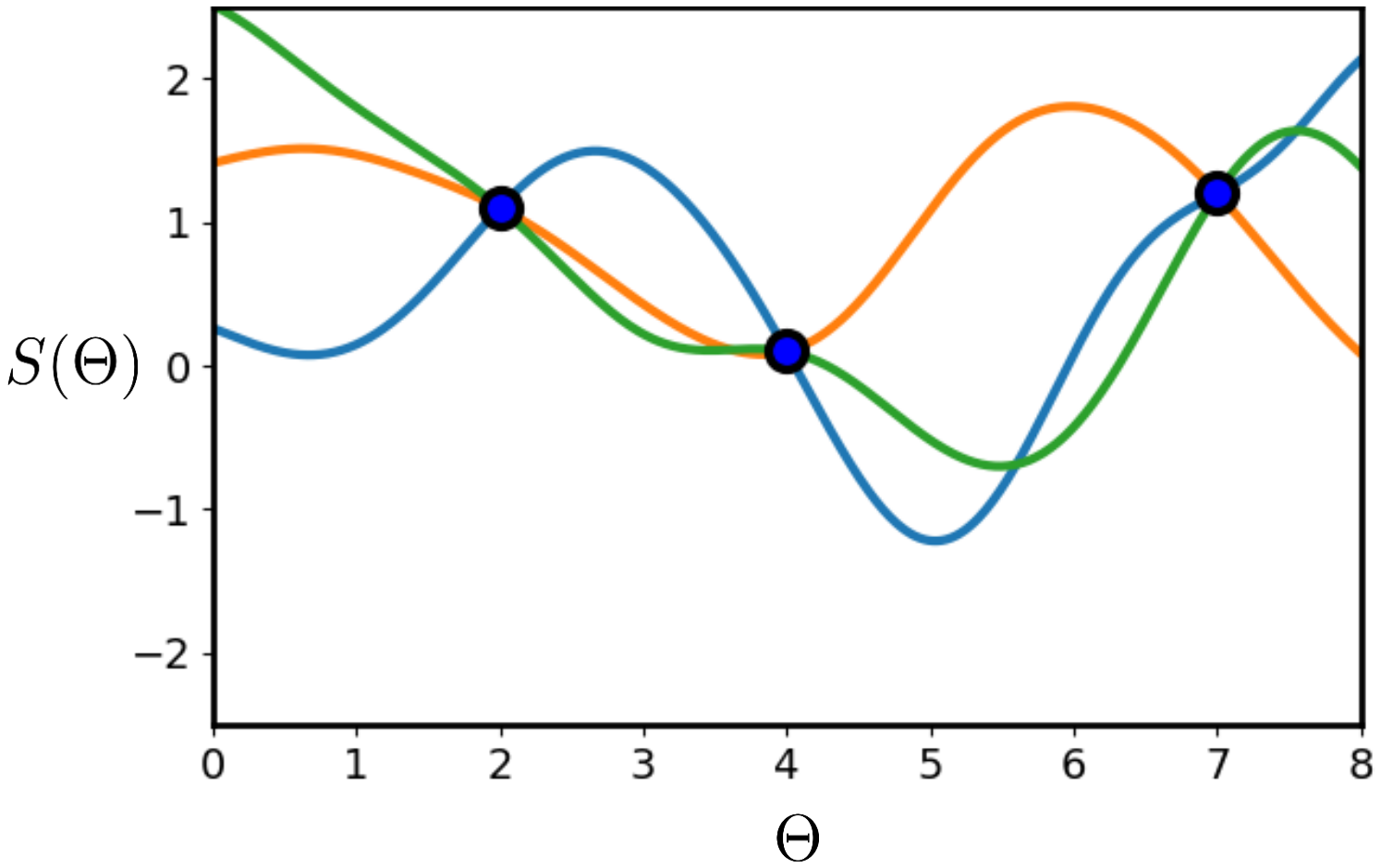
BAYESIAN OPTIMIZATION



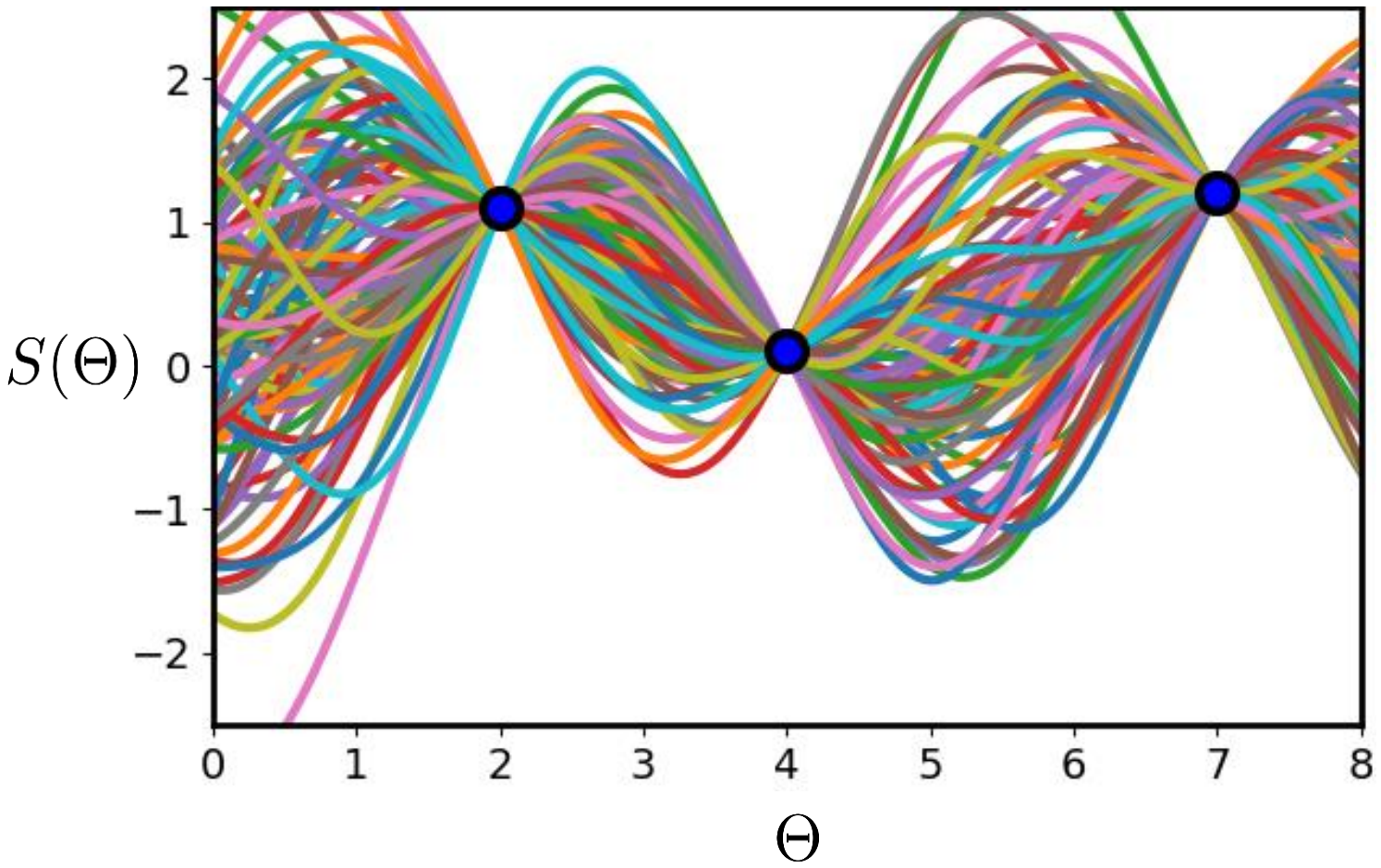
BAYESIAN OPTIMIZATION



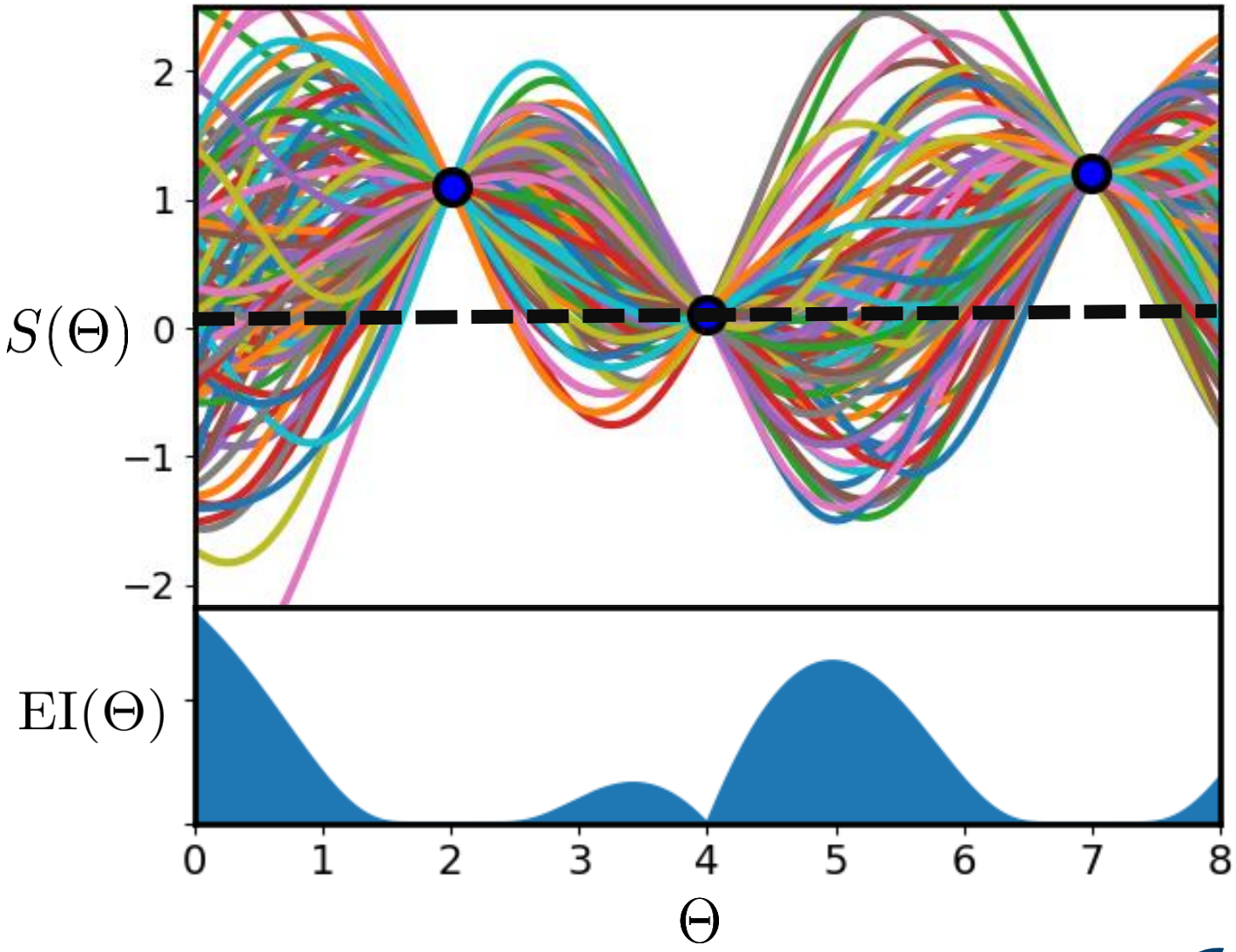
BAYESIAN OPTIMIZATION



BAYESIAN OPTIMIZATION

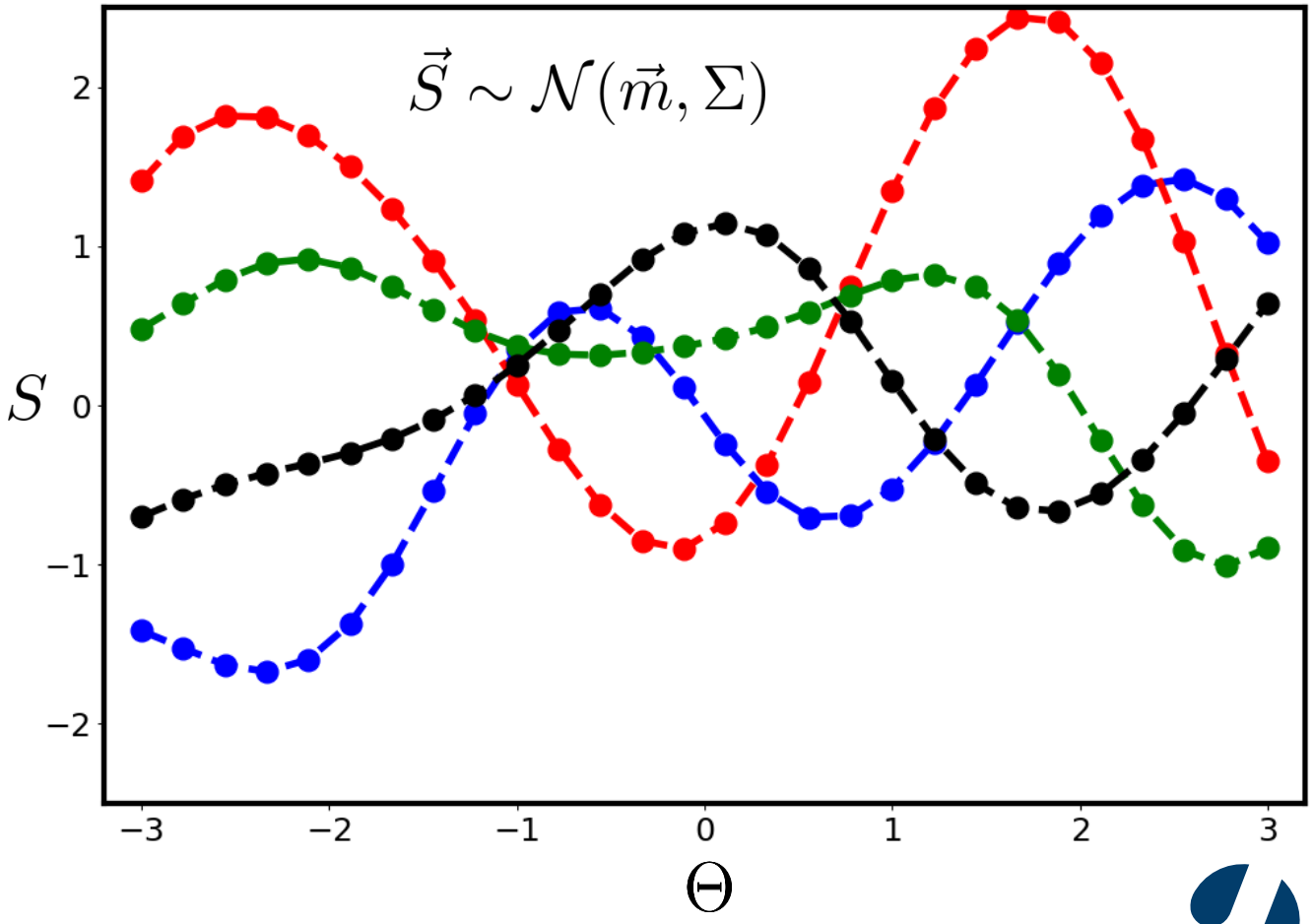


BAYESIAN OPTIMIZATION



MULTIVARIATE GAUSSIAN

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



GAUSSIAN PROCESSES

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

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

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)

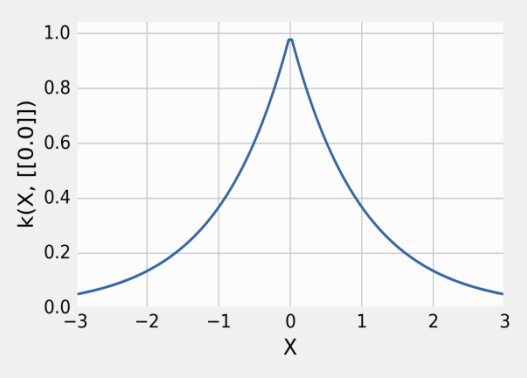
$$K(x, x') = \mathbb{E}[(S(x) - m(x))\,(S(x') - m(x'))]$$

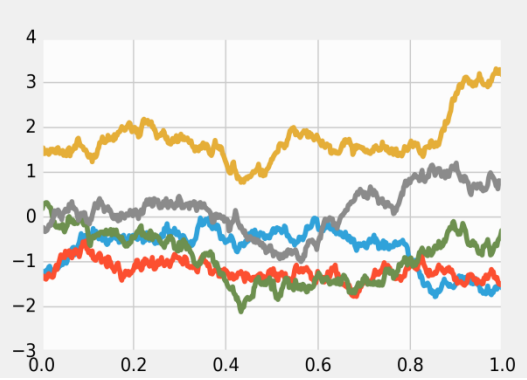
KERNEL FUNCTIONS

Exponential kernel

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

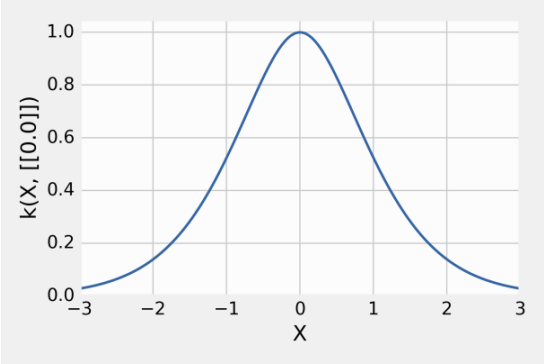
$$d := \|x - x'\|$$

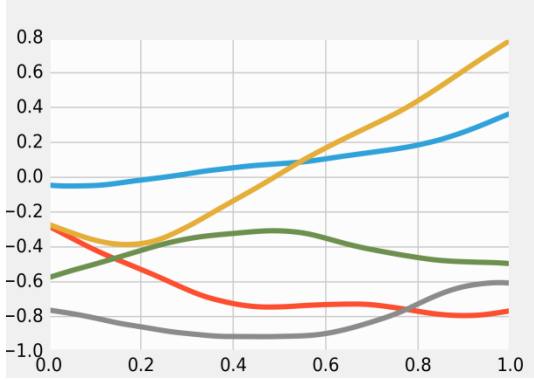




Matern kernel

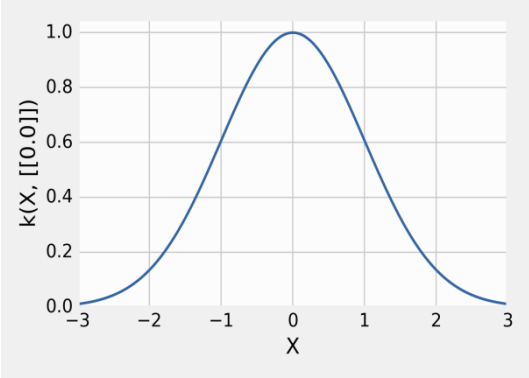
$$K(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\mu}|d|}{L}\right)^{\mu} I_{\mu}\left(\frac{\sqrt{2\mu}|d|}{L}\right)$$

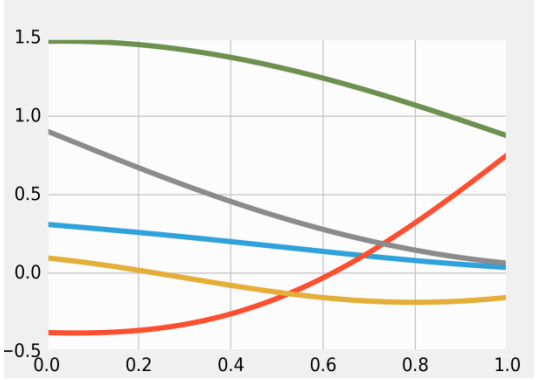




RBF kernel

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





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

MAXIMUM LIKELIHOOD

The emphirical risk can be interpreted as the energy of a conditional probability density function (sampling pdf)

$$p(y|\Theta,\Phi) \propto \exp\left(-\alpha \hat{R}_N\right) \qquad \hat{R}_N := \frac{1}{N}\sum_{i=1}^{N} L(y_i, f_{\Theta,\Phi}(x_i))$$

A *consistent* and *efficient* estimator of the model parameters is given by maximum likelihood

$$\hat{\Phi} = \arg\max_{\Phi} p(y|\hat{\Theta},\Phi)$$

Note: The hyperparameters are set to theit estimate $\hat{\Theta}$

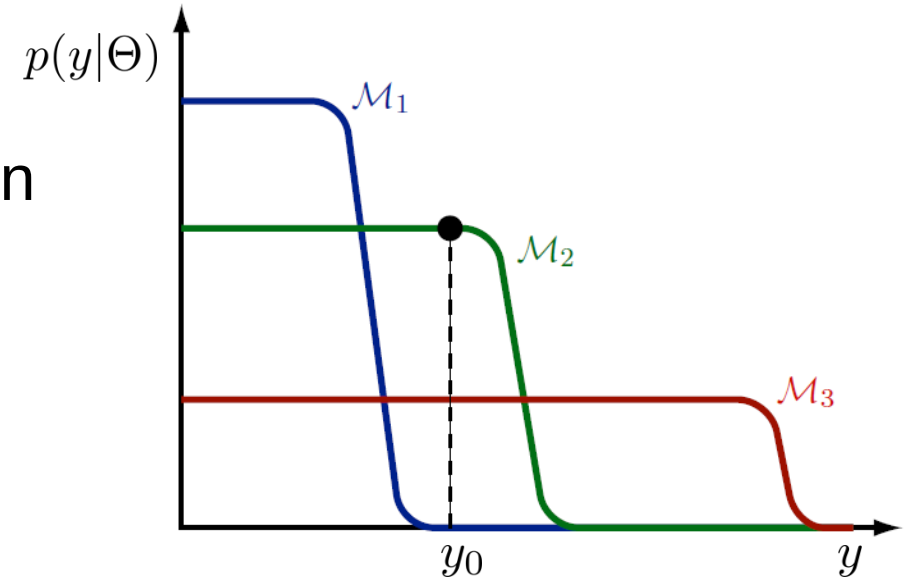
MARGINAL MAXIMUM LIKELIHOOD

In order to estimate the hyperparameters we can marginalize parameters of a GP in closed form

$$p(y|\Theta) = \int p(y|\Theta,\Phi)p(\Phi)d\Phi$$
$$= -\frac{N}{2}\ln 2\pi - \frac{1}{2}\ln\det K - y^T K^{-1} y$$

Maximum Likelihood estimation

$$\hat{\Theta} = \arg\max_{\Theta} p(y|\Theta)$$



ACQUISITION FUNCTION

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

Probability of Improvement (Kushner 1964):

$$a_{pi}(x) = \Psi(\gamma(x)) \qquad \gamma(x) = \frac{S(x_{min}) - m_{post.}(x)}{\sigma_{post.}}$$

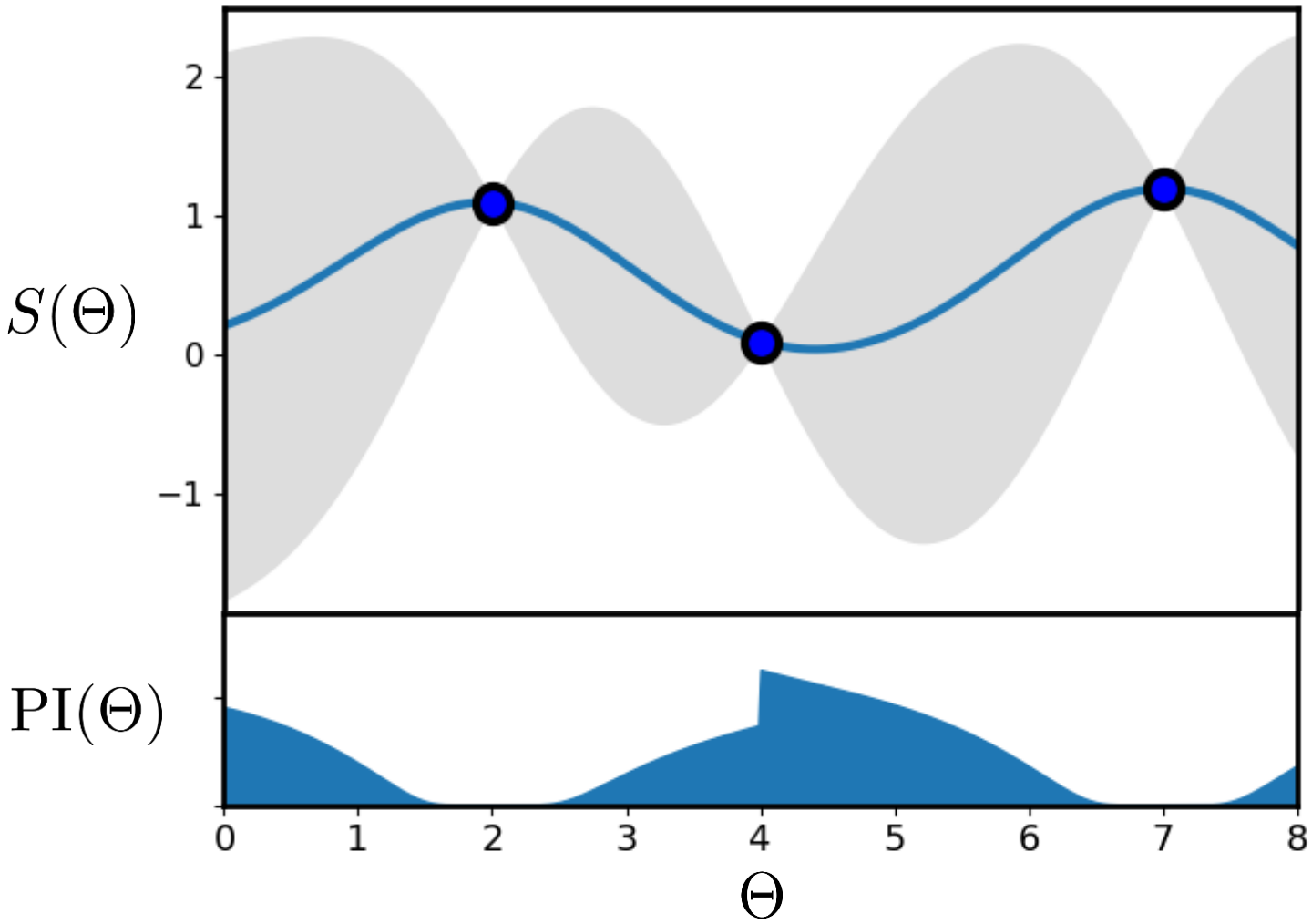
Expected Improvement (Mockus 1978):

$$a_{EI}(x) = \sigma_{post}\left(\gamma(x)\Psi(\gamma(x)) + \mathcal{N}(\gamma(x)|0,1)\right)$$

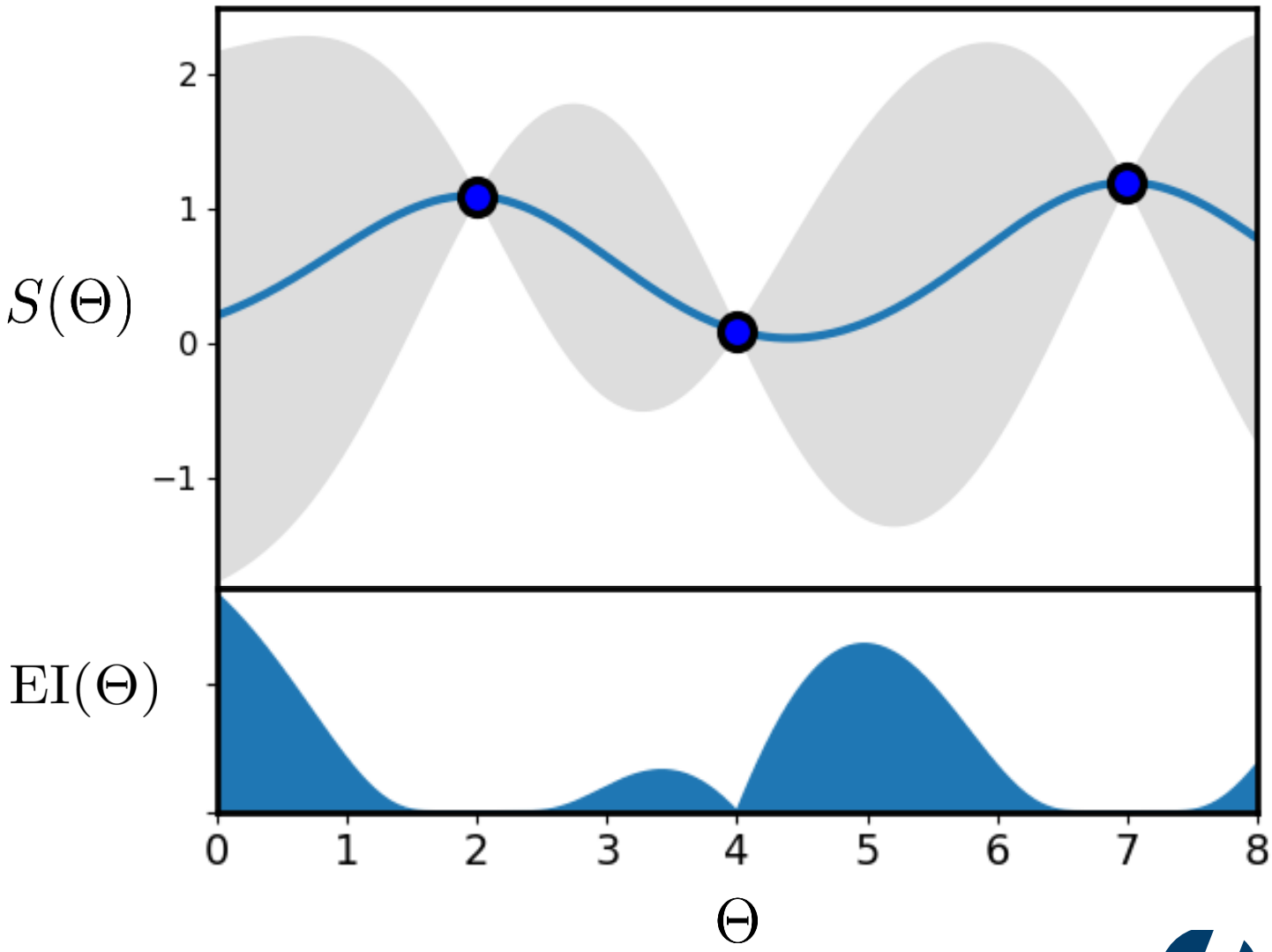
GP Upper Confidence Bound (Srinivas et al. 2010):

$$a_{UCB}(x) = \kappa\sigma_{post.} - m_{post.}$$

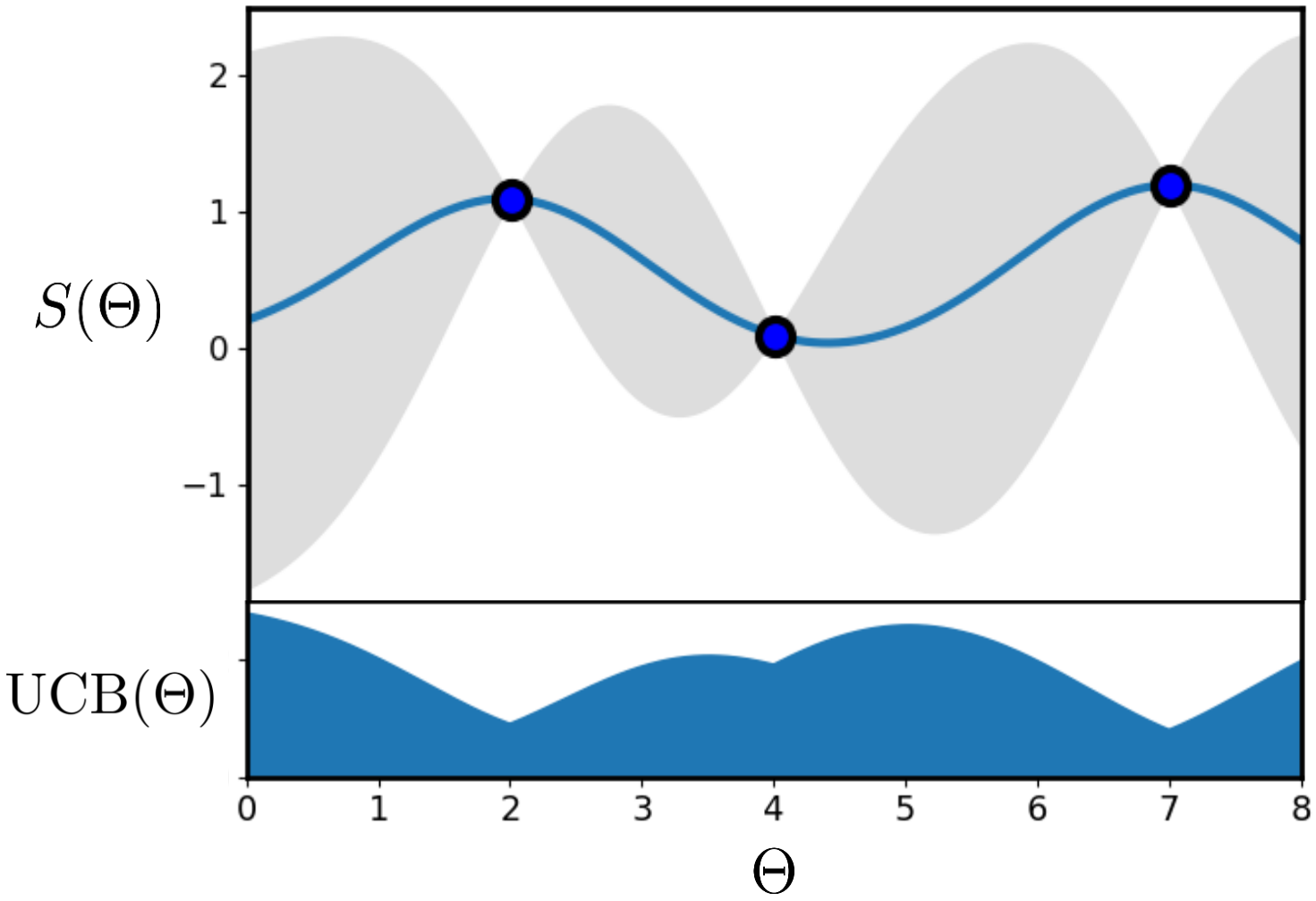
PROBABILITY OF IMPROVEMENT



EXPECTED IMPROVEMENT



UPPER CONFIDENCE BOUND

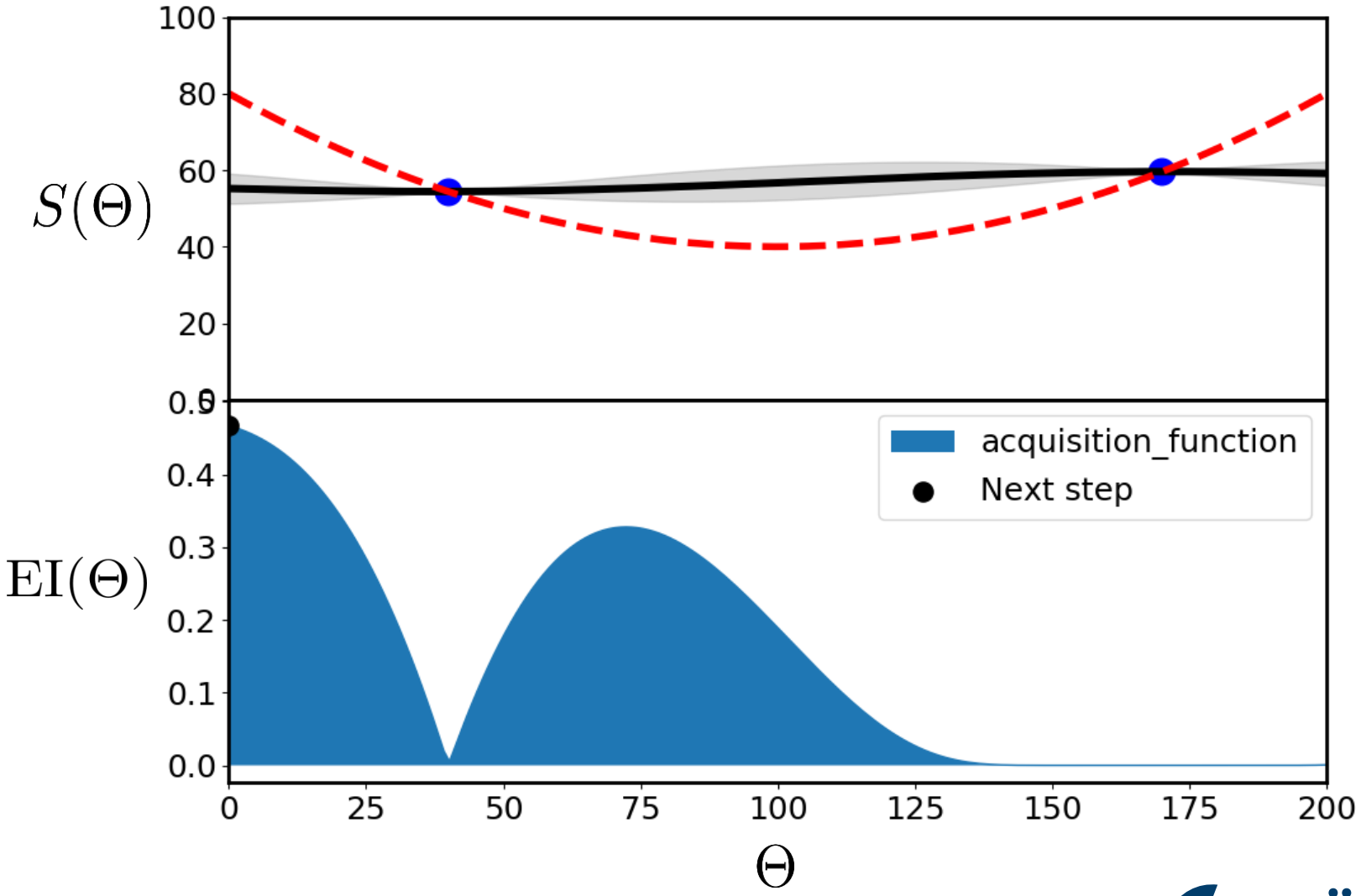


BAYESIAN OPTIMIZATION

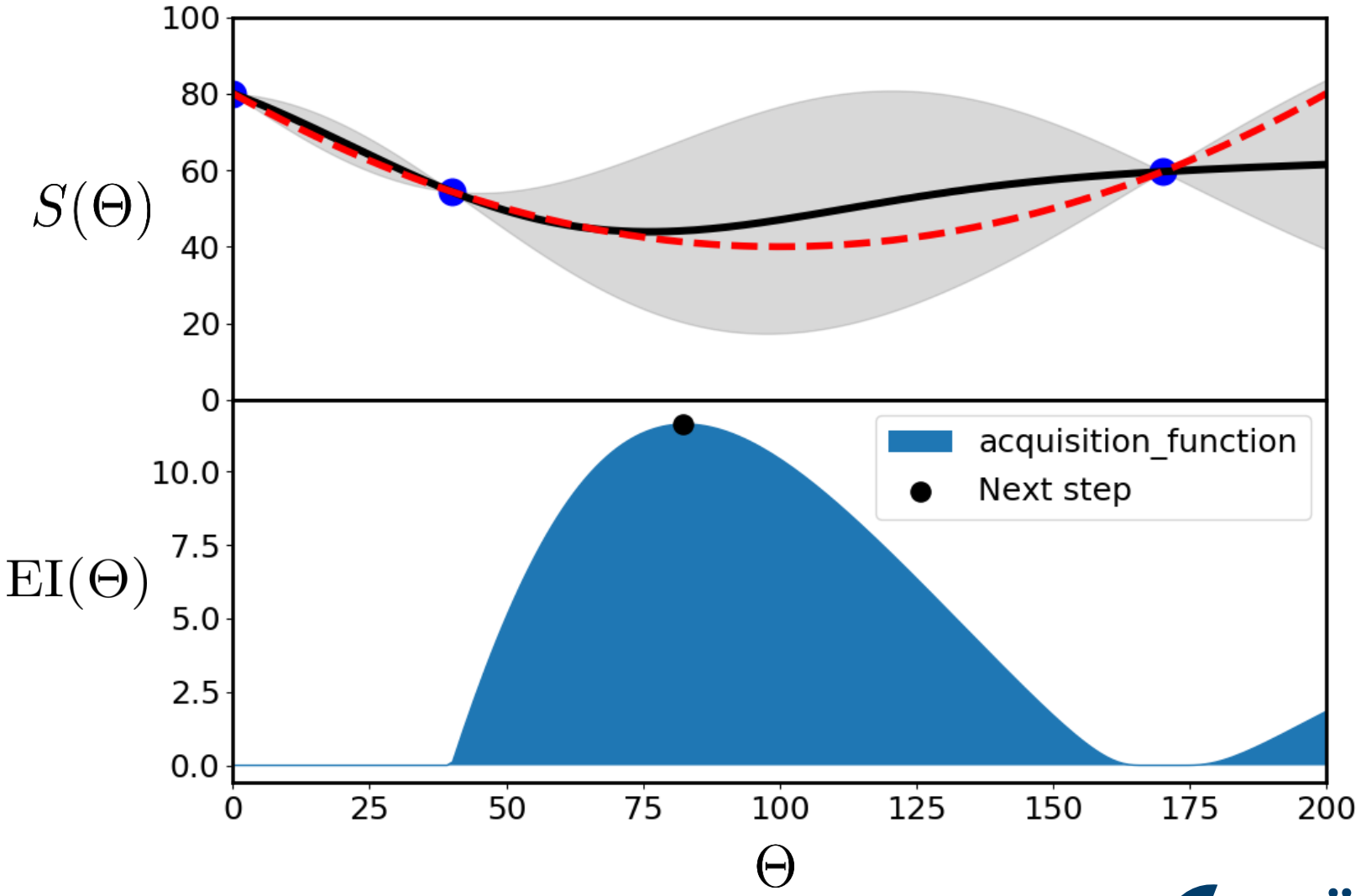
We combine all steps to Bayesian optimization

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 marginal ML
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

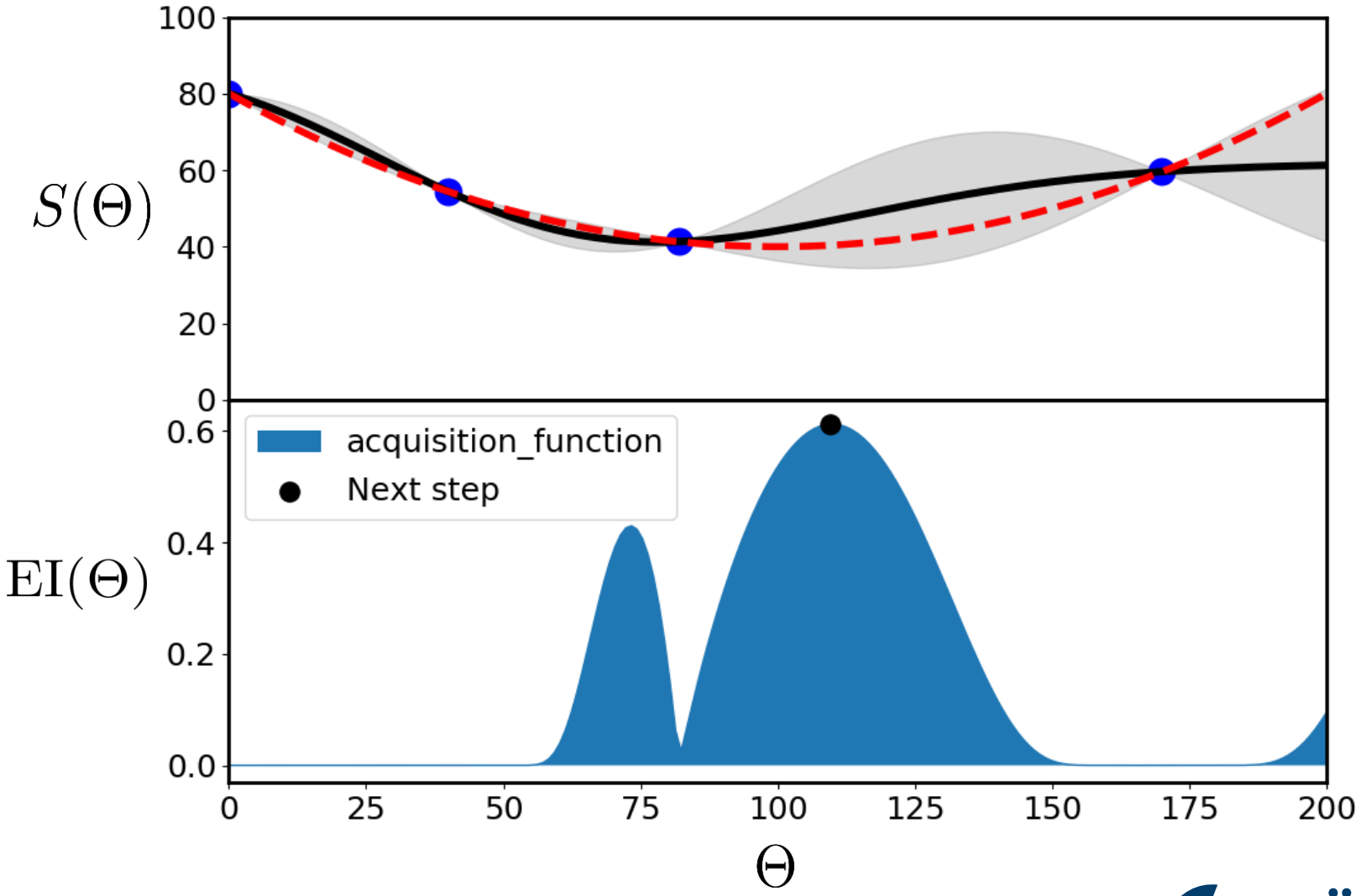
EXAMPLE BO


$S(\Theta)$
100
80
60
40
20
0
$\mathrm{EI}(\Theta)$
0.5
0.4
0.3
0.2
0.1
0.0
acquisition_function
Next step
0
25
50
75
100
125
150
175
200
Θ

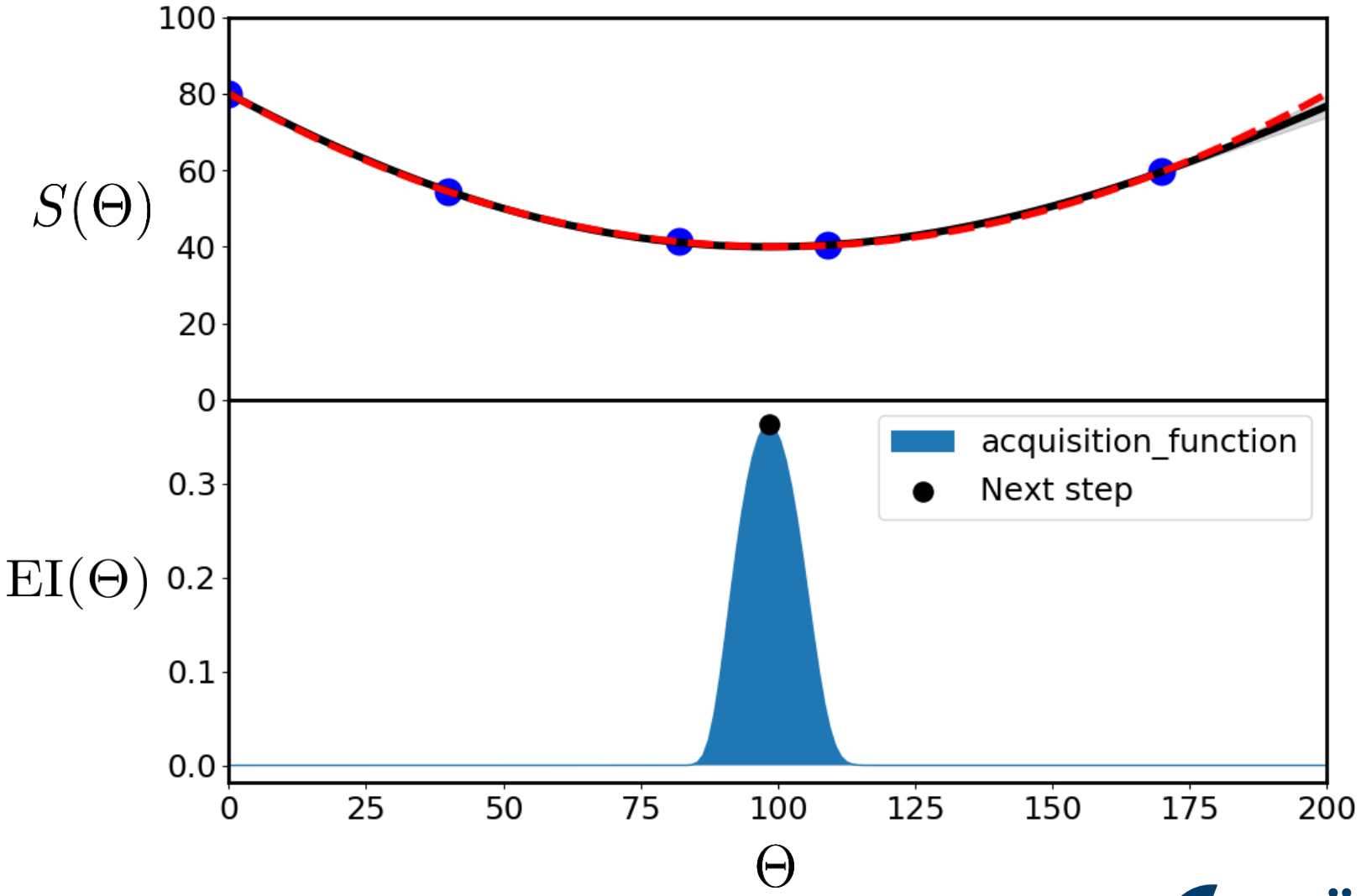
EXAMPLE BO


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

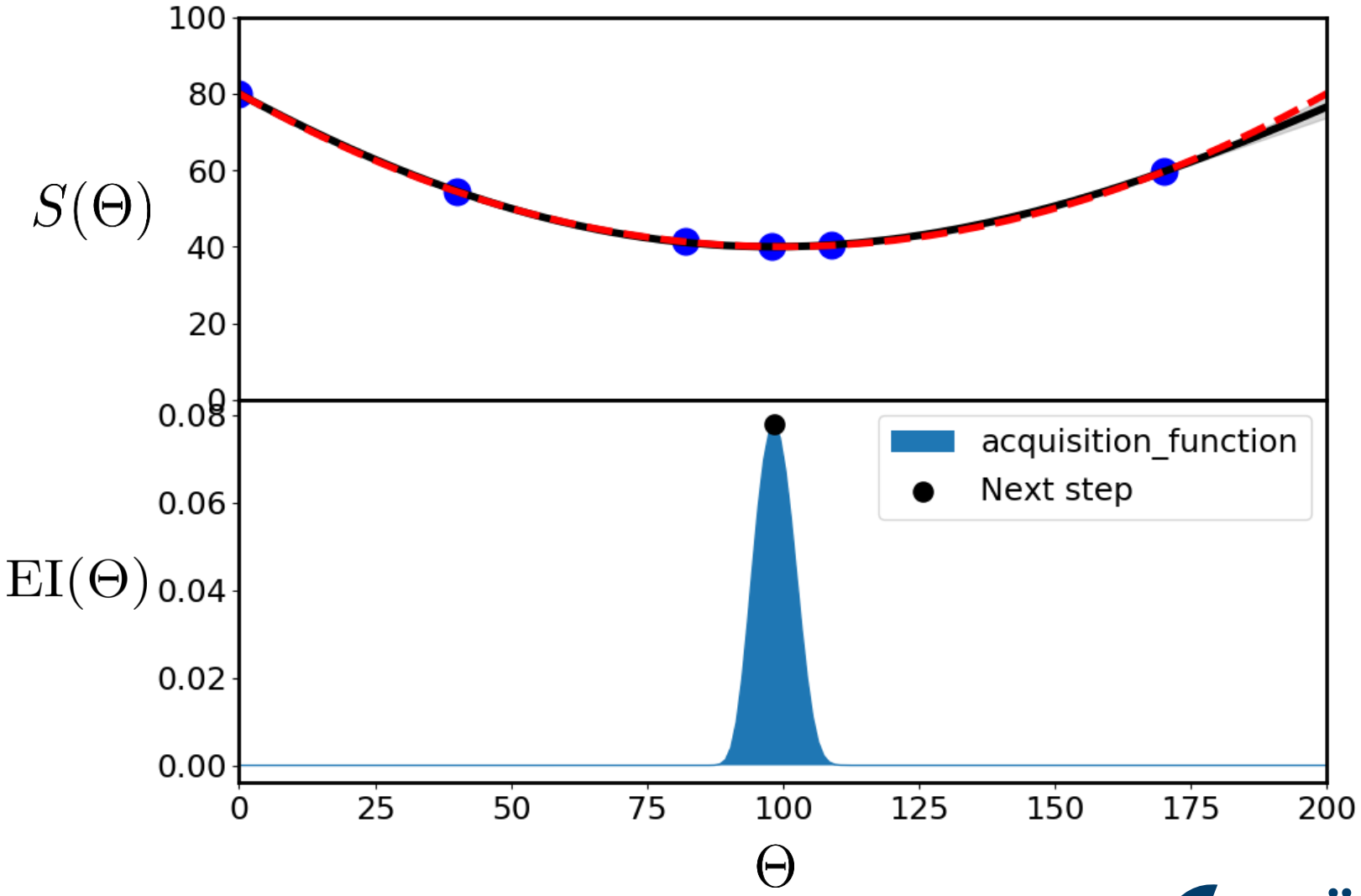
EXAMPLE BO


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

EXAMPLE BO


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

EXAMPLE BO


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

DEMO