# Hyperparameter Optimization

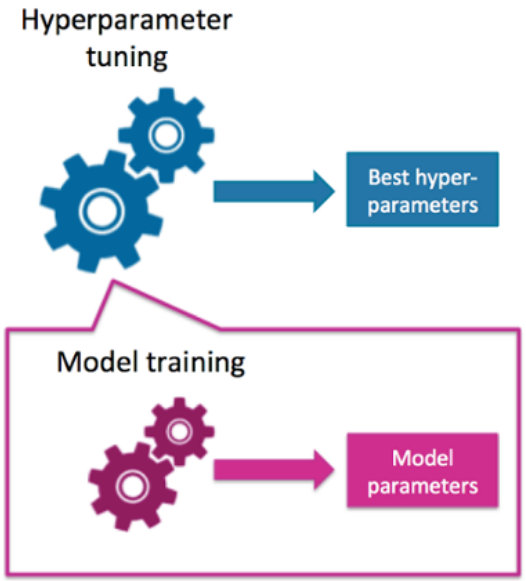
Dr. Josif Grabocka, Nicolas Schilling, Dr. Martin Wistuba

ISMLL, University of Hildesheim

Business Analytics

# Hyperparameter Optimization

- Model parameters are learned to minimize the loss on a training dataset
- Yet several hyper-parameters exist that control model complexity, characteristics and optimization choices




Courtesy: dato.com

# Learning Problems

The ultimate task of machine learning is to learn a model

$$\mathcal{A}(D^{\text{train}}) = \arg\min_{\hat{y} \in \mathcal{M}} \mathcal{L}(\hat{y}, D^{\text{train}}) + \mathcal{R}(\hat{y})$$

- $\mathcal{A}$ is a learning algorithm that returns a model
- $\mathcal{M}$ is the given model space
- $\mathcal{L}$ is a loss function, $D^{\text{train}}$ is training data
- $\mathcal{R}$ is a regularizer
- $\mathcal{A}$ depends on the correct setting of hyperparameters $\lambda$, which *usually* cannot be determined analytically

# Hyperparameter Optimization

We are interested in finding hyperparameters $\lambda^\star \in \Lambda$ for which holds:

$$\lambda^\star := \arg\min_{\lambda \in \Lambda} \mathcal{L}(\mathcal{A}_\lambda(D^{\text{train}}), D^{\text{val}}) := \arg\min_{\lambda \in \Lambda} f(\lambda, D) \ .$$

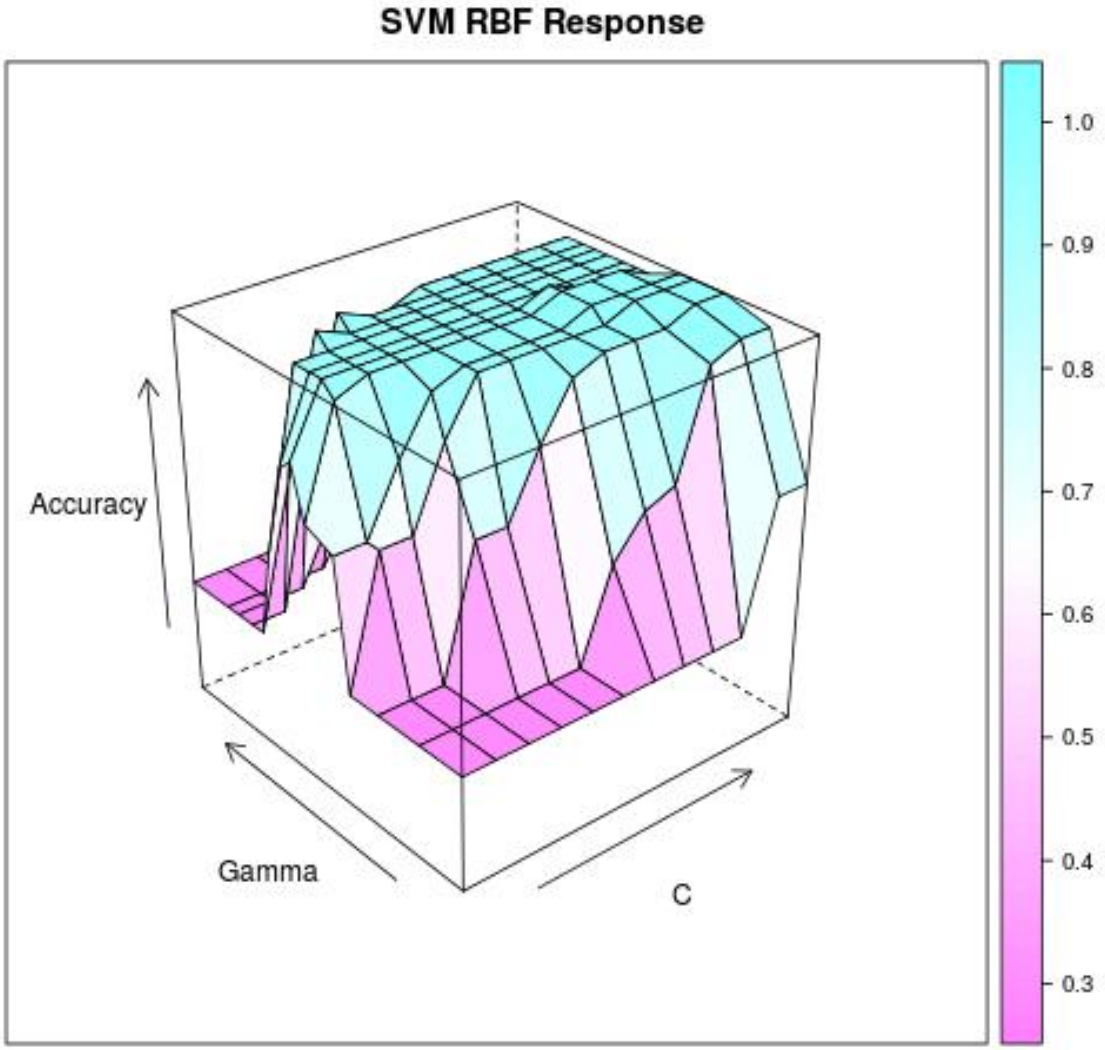
- search for the configuration yielding a learned model with best validation performance
- replace the loss of a learned model given $\lambda$ by the black-box function $f(\lambda, D)$
- $f$ is called **the response function**

# Example

- Model: SVM
- Loss/Response: Misclassification Rate / Accuracy
- Hyperparameters:
  - tradeoff $C$, choice of Kernel, kernel width $\gamma$, kernel degree $d$

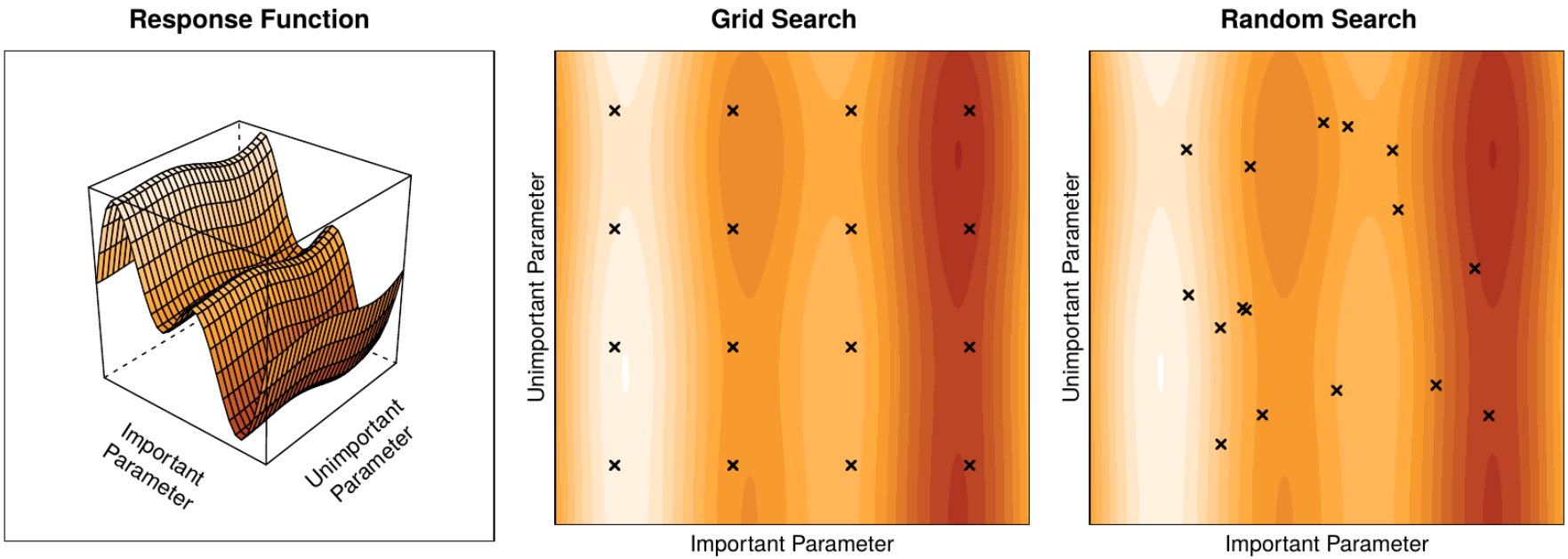
- Model: Neural Network
- Loss/Response: Misclassification Rate / Accuracy, MSE
- Hyperparameters:
  - Number of hidden layers, number of neurons per layer, choice of activation function, dropout rate, regularization penalty on weights, learning rate, choice of optimization algorithms

# Example (II)



# Grid and Random Search

- Need to try as few $\lambda \in \Lambda$, e.g.: in a grid, or randomly:



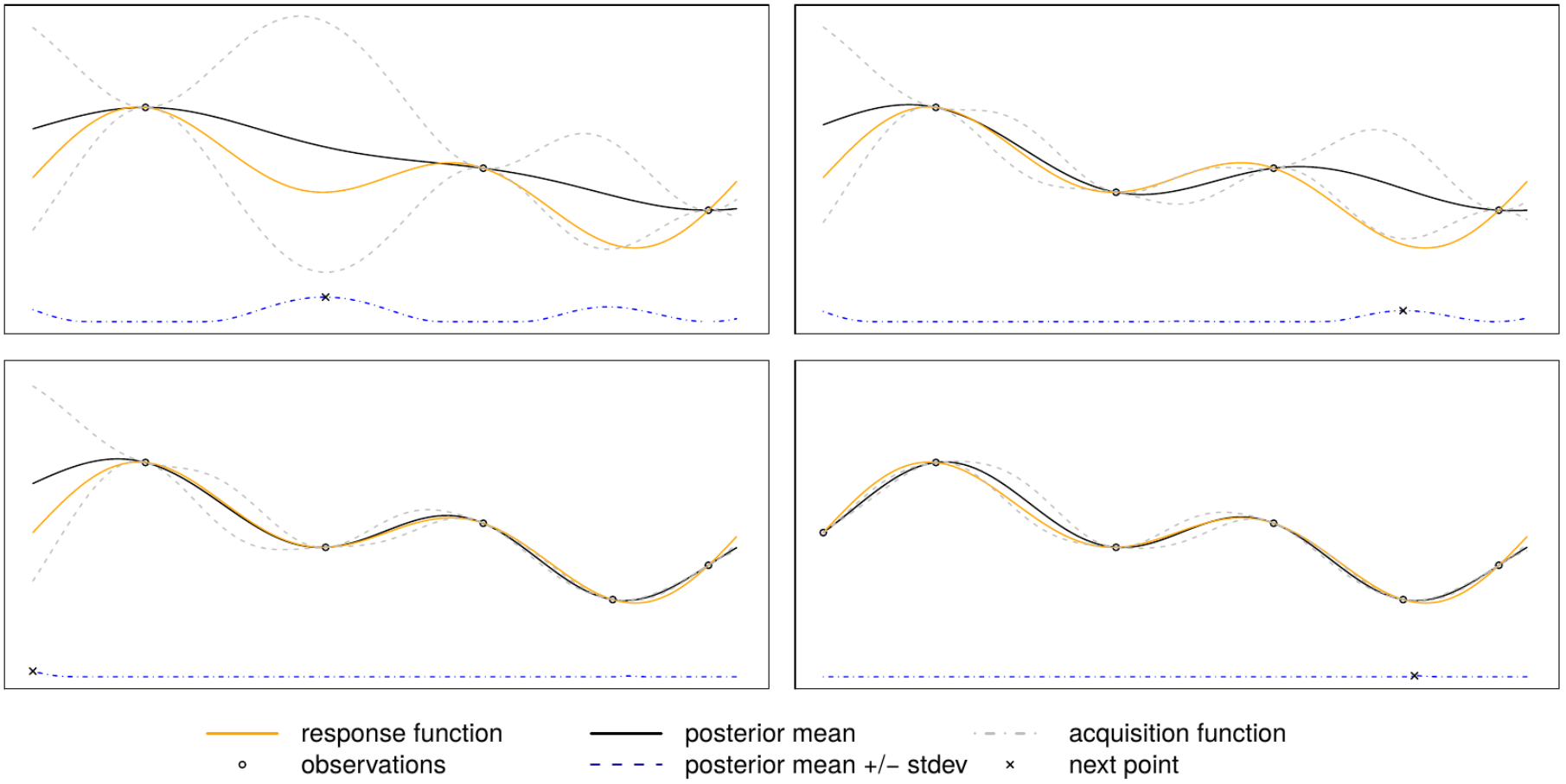

- and keep the hyper-parameters $\lambda^*$ yielding smallest $f(\lambda^*, D)$.
- Why are those approaches not sufficient?

# Sequential Model Based Optimization

Optimization of $f$ is costly as it involves many runs of the learning algorithm, therefore SMBO is used

- learn a **surrogate model** on known observations $\Psi \approx f$
- query $\Psi$ for regions of $\Lambda$ where **expected improvement** is high
- evaluate $f$ in these regions (run the learning algorithm)
- repeat the process until $f$ is optimized

# A sneak peek of SMBO ...



# The Surrogate Model $\Psi$:

- learns from the history of $N$ previous hyper-parameter evaluations

$$\mathcal{H} = \{(\lambda_1, f(\lambda_1, D)), (\lambda_2, f(\lambda_2, D)), \ldots, (\lambda_N, f(\lambda_N, D))\}$$

- is a parametric model $\Psi(\lambda; \theta)$ that approximates the true response $f(\lambda, D)$ over observed history $\mathcal{H}$:

$$\arg\min_\theta \sum_{n=1}^{N} \mathcal{L}_\Psi \left(\Psi(\lambda_n; \theta), f(\lambda_n, D)\right) + \mathcal{R}(\theta)$$

- $\Psi$ is often a Gaussian Process regressor

# Gaussian Processes (GP)

- Given our history of configurations $\lambda$ and responses $f(\lambda, D)$:

$$\Psi \sim \mathcal{N}(\mu(\lambda), k(\lambda, \lambda))$$

- And the covariance expressed as the squared exponential kernel:

$$k(\lambda, \lambda') = \exp\left(\frac{-||\lambda - \lambda'||^2}{2\sigma^2}\right)$$

- GP produces the mean and variance of a configuration $\lambda_*$:

$$\begin{aligned} \mu(\Psi(\lambda_*)) &= k_*^T K^{-1} f(\lambda) \\ \sigma^2(\Psi(\lambda_*)) &= k_{**} - k_*^T K^{-1} k_* \end{aligned}$$

  - $k = k(\lambda, \lambda)$, $k_* = k(\lambda, \lambda_*)$ and $k_{**} = k(\lambda_*, \lambda_*)$

# Sequential Model Based Optimization

**Require:** initial observation history $\mathcal{H}$, surrogate model $\Psi$, acquisition function $a$.

1: **return** Best hyperparameter configuration $\lambda^\star$
2: **for** $t = 1$ to $T$ **do**
3: Fit $\Psi$ to $\mathcal{H}$
4: $\lambda^{\text{new}} = \arg\max_{\lambda \in \Lambda} a(\lambda, \Psi(\lambda))$
5: Evaluate $f(\lambda^{\text{new}}, D)$
6: $\mathcal{H} = \mathcal{H} \cup (\lambda^{\text{new}}, f(\lambda^{\text{new}}, D))$
7: **end for**
8: **return** $\lambda^\star = \arg\min_{\lambda} f(\lambda, D)$

- Note: $f$ can also be classification accuracy and the problem becomes a maximization, i.e. $\lambda^\star = \arg\max_{\lambda} f(\lambda, D)$

# How to choose next hyper-parameter candidate?

- Knowing the mean $\mu(\Psi(\lambda))$ and $\sigma(\Psi(\lambda))$ of the surrogate model
- We use an acquisition function $a$ over all possible hyper-parameter configurations $\lambda^{\text{new}} = \arg\max_{\lambda \in \Lambda} a\left(\lambda, \Psi(\lambda)\right)$
- It is important to balance **exploration vs. exploitation** of the search space
- Alternative 1: The Lower Confidence Bound acquisition

$$a(\lambda) = -\mu(\Psi(\lambda)) + \beta\sigma(\Psi(\lambda))$$

# Acquisition Choice: Expected Improvement

- Define the standardized improvement of a configuration $\lambda$ as :

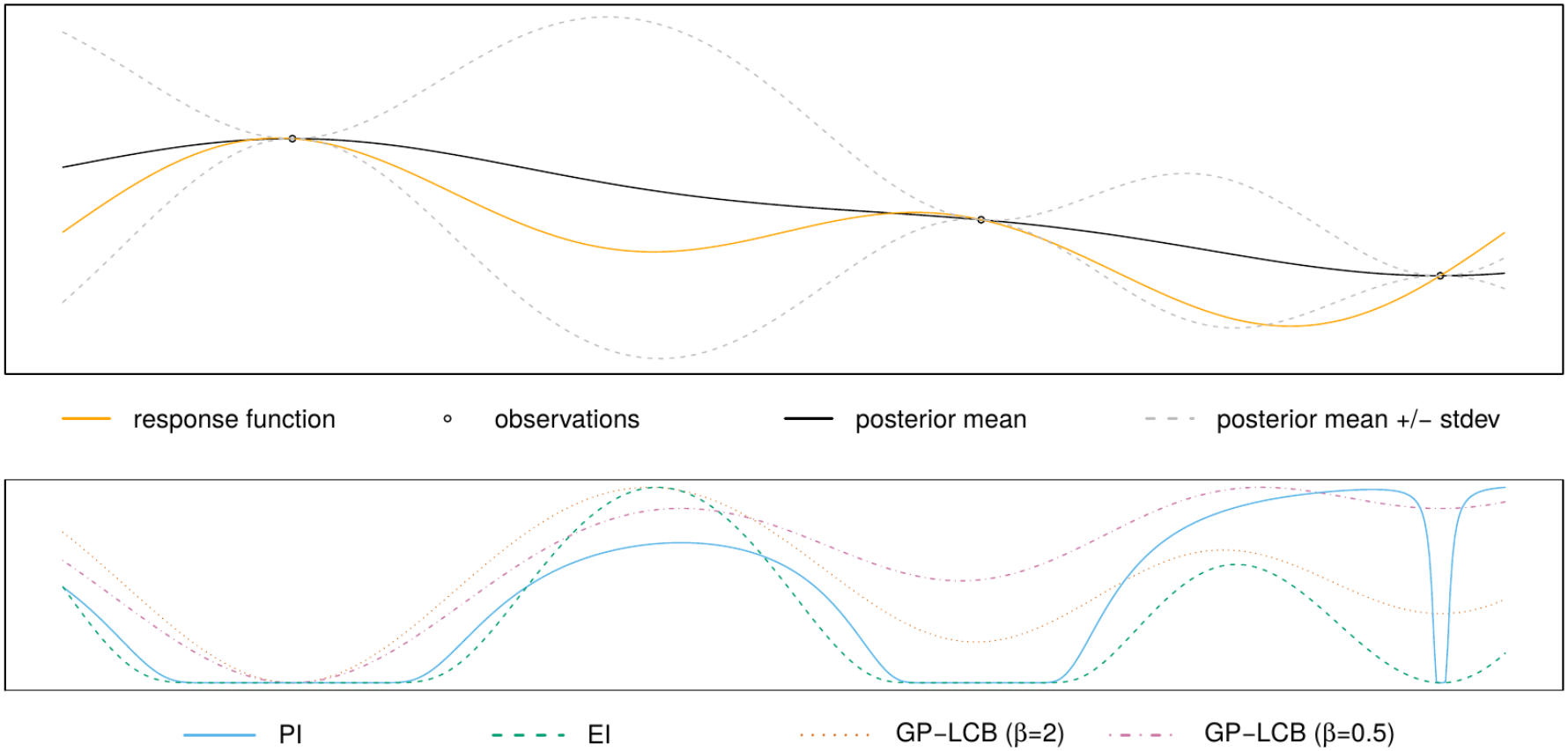
$$Z(\lambda) = \frac{f^{\min} - \mu(\Psi(\lambda))}{\sigma(\Psi(\lambda))}$$

- The expected improvement is:

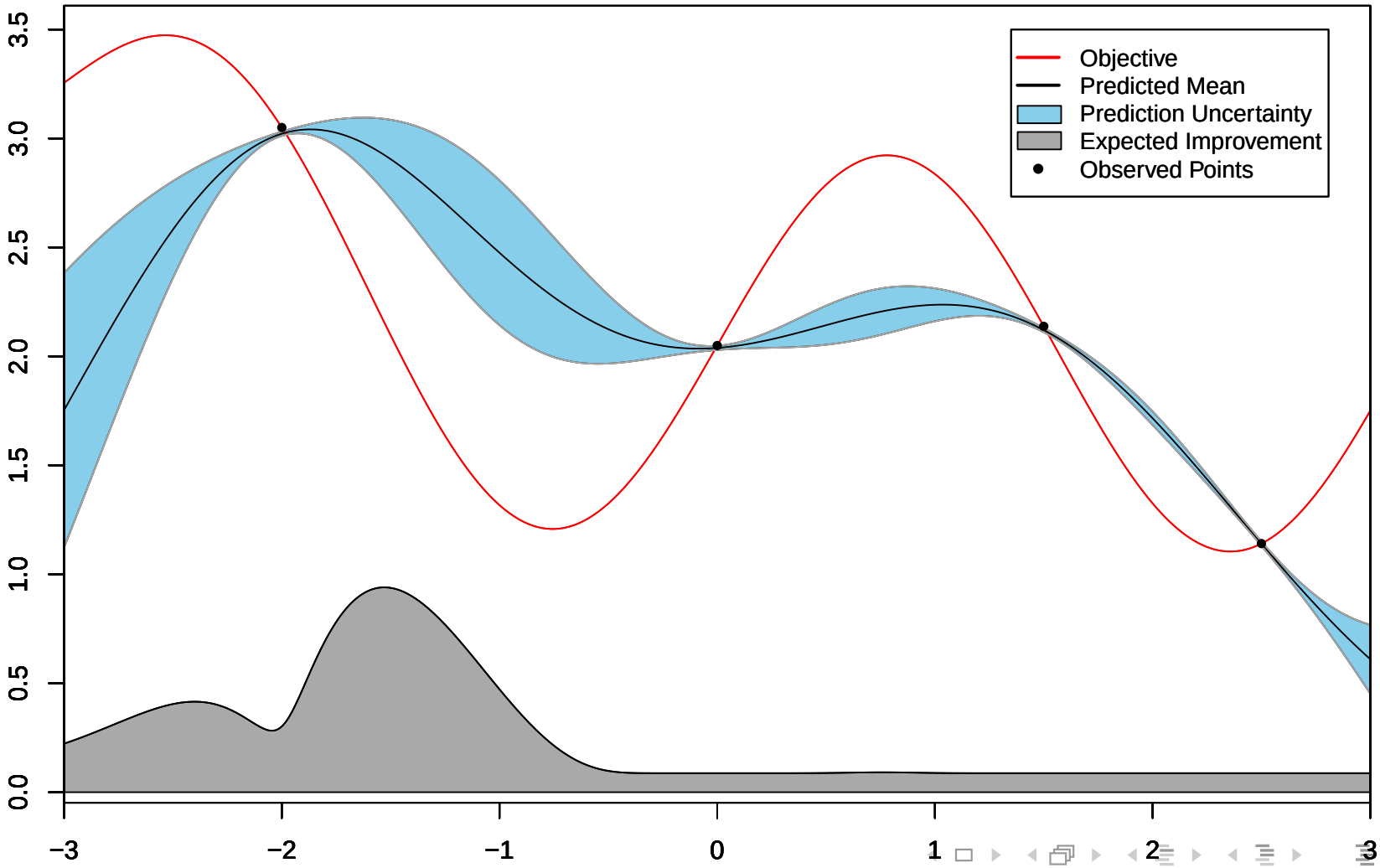
$$EI(\lambda) = \begin{cases} \sigma(\Psi(\lambda))\left(Z(\lambda) \cdot \Phi(Z(\lambda)) + \phi(Z(\lambda))\right) & \sigma^2(\Psi(\lambda)) > 0 \\ 0 & \text{otherwise} \end{cases}$$

- where $\Phi(Z(\lambda))$: Gaussian density, $\phi(Z(\lambda))$: Gaussian cumulative distribution function

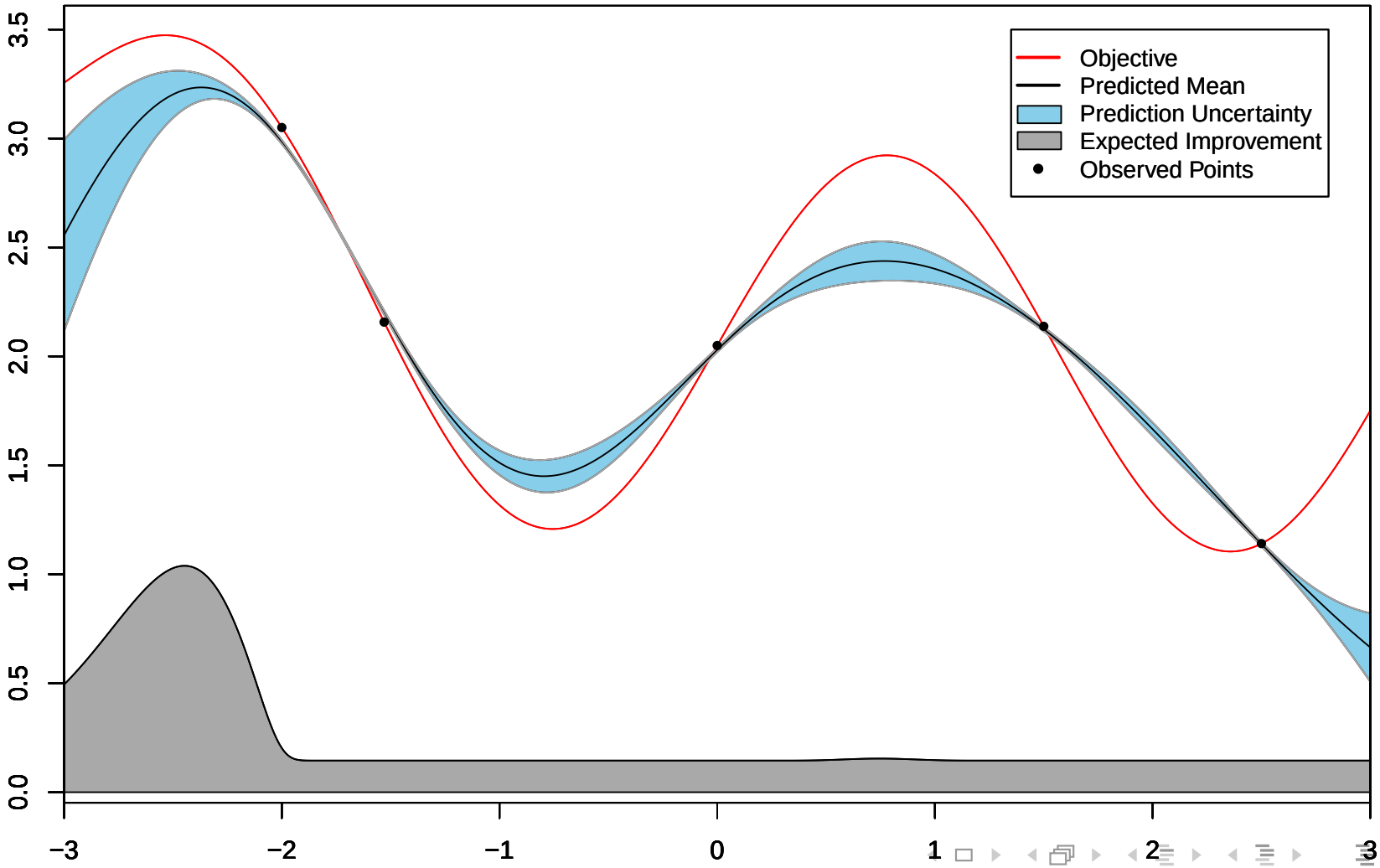
# Choice of acquisitions



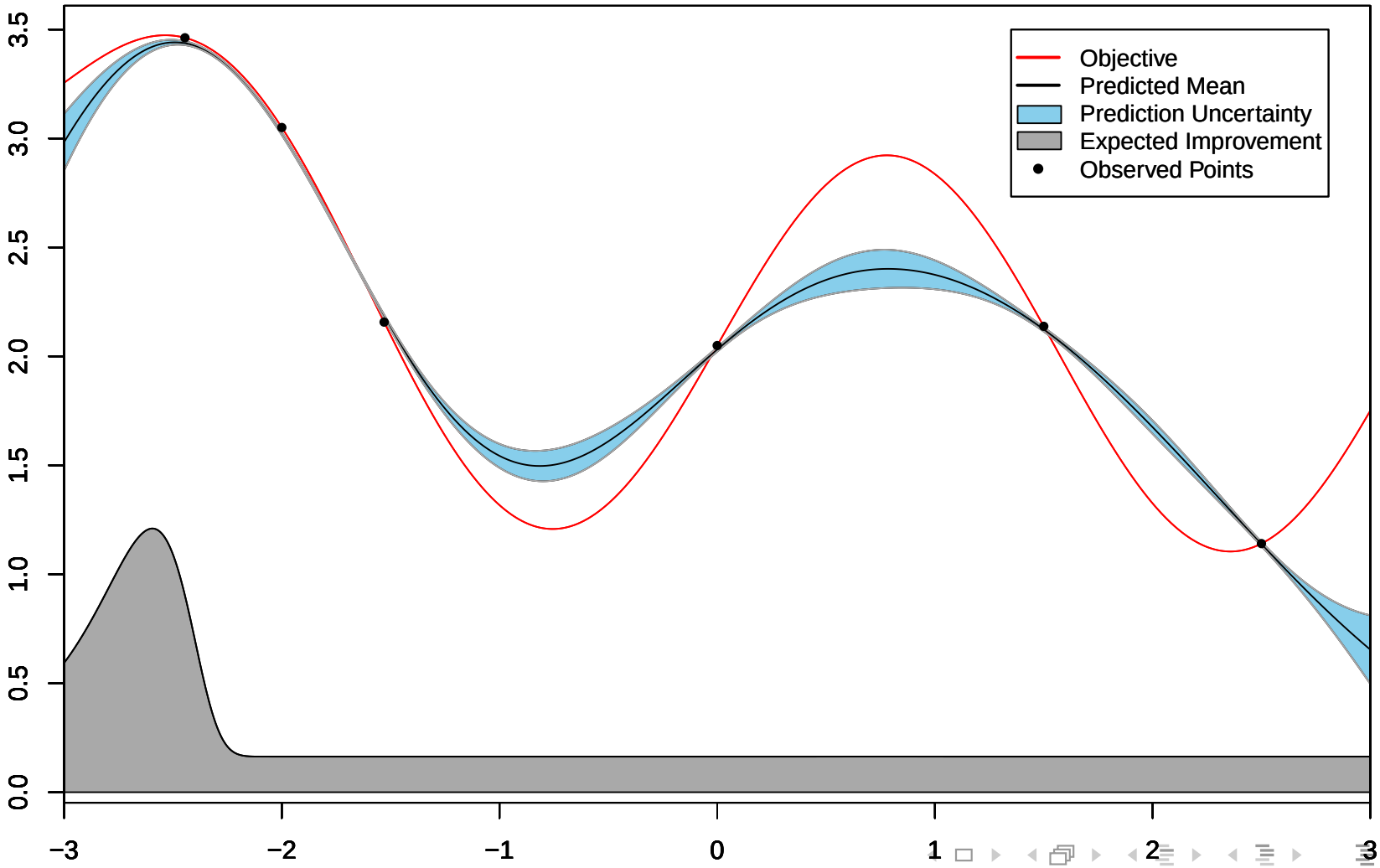
# Gaussian Process Learning with SMBO (I)



# Gaussian Process Learning with SMBO (II)



# Gaussian Process Learning with SMBO (III)



# Gaussian Process Learning with SMBO (IV)