Bayesian Optimization and Hyperparameter Search

Marc Deisenroth

Department of Computing
Imperial College London

@mpd37
m.deisenroth@imperial.ac.uk

Deep Learning Indaba
Kenyatta University
Nairobi, Kenya

August 27, 2019

Reading Material

- Brochu et al.: *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*, arXiv:1012.2599, 2012
- Shahriari et al.: *Taking the Human Out of the Loop: A Review of Bayesian Optimization*, Proceedings of the IEEE, 2016

Overview

Some Experiments are Expensive

- Practical optimization problems: oil drill locations, malaria prevention strategies, drug design ...
- Testing every setting costs much money or time

Machine Learning Meta-Challenges

- Machine learning models are getting more and more complicated
  ⏩ Usually more parameters (e.g., deep neural networks)
- Non-convex and stochastic optimization methods have meta-parameters that are difficult to tune (learning rates, momentum parameters, ...)

⏩ Generally hard to apply modern techniques or reproduce results

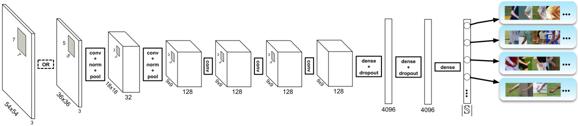
Machine Learning Meta-Challenges

- Machine learning models are getting more and more complicated
  - ⏩ Usually more parameters (e.g., deep neural networks)
- Non-convex and stochastic optimization methods have meta-parameters that are difficult to tune (learning rates, momentum parameters, ...)

⏩ Generally hard to apply modern techniques or reproduce results

Goal: Automate the selection of critical meta-parameters
(see also: Automated Machine Learning (AutoML))

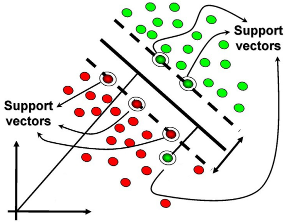
Example: Deep Neural Networks

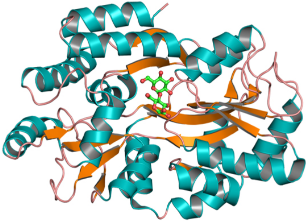


Huge interest in large neural networks

- When well-tuned, very successful for visual object identification, speech recognition, computational biology, ...
- Huge investments by Google, Facebook, Microsoft, etc.
- Many choices: number of layers, weight regularization, layer size, which nonlinearity, batch size, learning rate schedule, stopping conditions

Example: Classification of DNA Sequences





- Objective: Predict which DNA sequences will bind with which proteins
- Miller et al. (2012): Latent Structural Support Vector Machine
- Hyper-parameters: margin/slack parameter, entropy parameter, convergence criterion

Search for Good Hyper-parameters

- Define an objective function to evaluate the quality of the hyper-parameters
  - Usually, we care about generalization performance
  - Cross validation to measure parameter quality

Search for Good Hyper-parameters

- Define an objective function to evaluate the quality of the hyper-parameters
  - Usually, we care about generalization performance
  - Cross validation to measure parameter quality
- Standard search procedures:
  - Manual tuning (requires expert knowledge)
  - Grid search (does not scale to high dimensions)
  - Random search (very simple, works surprisingly well)
  - Magic

Search for Good Hyper-parameters

- Define an objective function to evaluate the quality of the hyper-parameters
  - Usually, we care about generalization performance
  - Cross validation to measure parameter quality

- Standard search procedures:
  - Manual tuning (requires expert knowledge)
  - Grid search (does not scale to high dimensions)
  - Random search (very simple, works surprisingly well)
  - Magic

- Painful:
  - Evaluating the quality of the objective may be very expensive (e.g., time or money)
    ▶▶ For example, running a GPU/TPU cluster for weeks
  - Noisy observations

Alternative Approach: Bayesian Optimization

Setting

Globally optimize a black-box objective that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

- Build a probabilistic proxy model for the objective using outcomes of past experiments as training data

  ▸▸ Probabilistic Regression

Alternative Approach: Bayesian Optimization

Setting

Globally optimize a black-box objective that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

- Build a probabilistic proxy model for the objective using outcomes of past experiments as training data
  - ▸▸ Probabilistic Regression
- The proxy model is much cheaper to evaluate than the original objective

Alternative Approach: Bayesian Optimization

Setting

Globally optimize a black-box objective that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

- Build a probabilistic proxy model for the objective using outcomes of past experiments as training data
  ▸▸ Probabilistic Regression
- The proxy model is much cheaper to evaluate than the original objective
- Optimize cheap proxy function to determine where to evaluate the true objective next

Alternative Approach: Bayesian Optimization

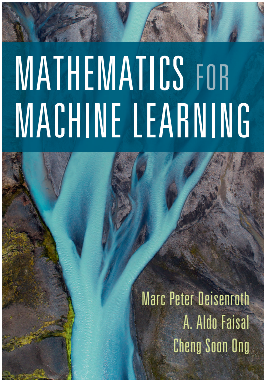
Setting

Globally optimize a black-box objective that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

- Build a probabilistic proxy model for the objective using outcomes of past experiments as training data
  ⏩ Probabilistic Regression
- The proxy model is much cheaper to evaluate than the original objective
- Optimize cheap proxy function to determine where to evaluate the true objective next
- Standard proxy: Gaussian process

Overview

Crashcourse on Linear Regression

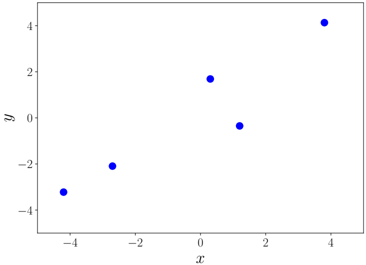


https://mml-book.com

Regression Problems

Regression (curve fitting)

Given inputs x and corresponding observations y find a function f that models the relationship between $\boldsymbol{x}$ and y.

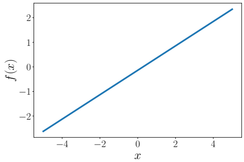


- Typically parametrize the function f with parameters $\boldsymbol{\theta}$
- Linear regression: Consider functions f that are **linear in the parameters**

Linear Regression Functions

- Straight lines

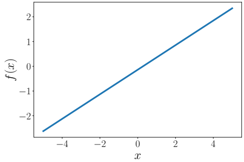
$$y = f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}$$



Linear Regression Functions

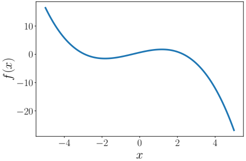
- Straight lines

$$y = f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}$$



- Polynomials

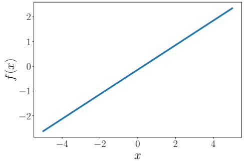
$$y = f(x, \boldsymbol{\theta}) = \sum_{m=0}^{M} \theta_m x^m = \begin{bmatrix} \theta_0 & \cdots & \theta_M \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ x^M \end{bmatrix}$$



Linear Regression Functions

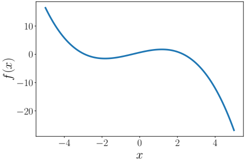
- Straight lines

$$y = f(x, \boldsymbol{\theta}) = \theta_0 + \theta_1 x = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix}$$



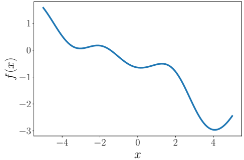
- Polynomials

$$y = f(x, \boldsymbol{\theta}) = \sum_{m=0}^{M} \theta_m x^m = \begin{bmatrix} \theta_0 & \cdots & \theta_M \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ x^M \end{bmatrix}$$



- Radial basis function networks

$$y = f(x, \boldsymbol{\theta}) = \sum_{m=1}^{M} \theta_m \exp\left(-\tfrac{1}{2}(x - \mu_m)^2\right)$$



Linear Regression Model and Setting

$$y = \boldsymbol{x}^\top \boldsymbol{\theta} + \epsilon \,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)$$

- Given a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ we seek optimal parameters $\boldsymbol{\theta}^*$

Linear Regression Model and Setting

$$y = \boldsymbol{x}^\top \boldsymbol{\theta} + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)$$

- Given a training set $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$ we seek optimal parameters $\boldsymbol{\theta}^*$

  ▶▶ **Maximum Likelihood Estimation**

  ▶▶ **Maximum a Posteriori Estimation**

Maximum Likelihood

- Define $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$
- Find parameters $\boldsymbol{\theta}^*$ that maximize the likelihood

Maximum Likelihood

- Define $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y} = [y_1, \dots, y_N]^\top \in \mathbb{R}^N$
- Find parameters $\boldsymbol{\theta}^*$ that maximize the likelihood

$$p(y_1, \dots, y_N | \boldsymbol{x}_1, \dots, \boldsymbol{x}_N, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\left(y_n \mid \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right)$$

Maximum Likelihood

- Define $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$
- Find parameters $\boldsymbol{\theta}^*$ that maximize the likelihood

$$p(y_1, \ldots, y_N | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{\theta}) = p(\boldsymbol{y} | \boldsymbol{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\big(y_n \,|\, \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\big)$$

- Log-transformation ⏩ **Maximize the log likelihood**

Maximum Likelihood

- Define $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$
- Find parameters $\boldsymbol{\theta}^*$ that maximize the likelihood

$$p(y_1, \ldots, y_N | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\big(y_n \,|\, \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\big)$$

- Log-transformation ⏩ **Maximize the log likelihood**

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{n=1}^{N} \log \mathcal{N}\big(y_n \,|\, \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\big)\,,$$

$$\log \mathcal{N}\big(y_n \,|\, \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\big) = -\frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + \text{ const}$$

Maximum Likelihood

- Define $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$ and $\boldsymbol{y} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$
- Find parameters $\boldsymbol{\theta}^*$ that maximize the likelihood

$$p(y_1, \ldots, y_N | \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_{n=1}^{N} \mathcal{N}\left(y_n \,|\, \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right)$$

- Log-transformation ⏩ **Maximize the log likelihood**

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{n=1}^{N} \log \mathcal{N}\left(y_n \,|\, \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right),$$

$$\log \mathcal{N}\left(y_n \,|\, \boldsymbol{x}_n^\top \boldsymbol{\theta}, \sigma^2\right) = -\frac{1}{2\sigma^2}(y_n - \boldsymbol{x}_n^\top \boldsymbol{\theta})^2 + \text{ const}$$

- Computing the gradient with respect to $\boldsymbol{\theta}$ and setting it to $\boldsymbol{0}$ gives the **maximum likelihood estimator** (least-squares estimator)

$$\boldsymbol{\theta}^{\text{ML}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

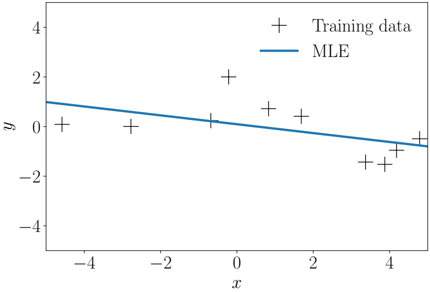
Making Predictions

$$y = \boldsymbol{x}^\top \boldsymbol{\theta} + \epsilon, \quad \epsilon \sim \mathcal{N}\big(0, \sigma^2\big)$$

Given an arbitrary input $\boldsymbol{x}_*$, we can predict the corresponding observation y_* using the maximum likelihood parameter:

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{\theta}^{\mathrm{ML}}) = \mathcal{N}\big(y_* \,|\, \boldsymbol{x}_*^\top \boldsymbol{\theta}^{\mathrm{ML}}, \sigma^2\big)$$

Example 1: Linear Functions



$$y = \theta_0 + \theta_1 x + \epsilon, \quad \epsilon \sim \mathcal{N}\big(0, \sigma^2\big)$$

- At any query point x_* we obtain the mean prediction as

$$\mathbb{E}[y_* | \boldsymbol{\theta}^{\mathrm{ML}}, x_*] = \theta_0^{\mathrm{ML}} + \theta_1^{\mathrm{ML}} x_*$$

Nonlinear Functions

$$y = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta} + \epsilon = \sum_{m=0}^{M} \theta_m x^m + \epsilon$$

- Polynomial regression with features

$$\boldsymbol{\phi}(x) = [1, x, x^2, \dots, x^M]^\top$$

- Maximum likelihood estimator:

Nonlinear Functions

$$y = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta} + \epsilon = \sum_{m=0}^{M} \theta_m x^m + \epsilon$$

- Polynomial regression with features

$$\boldsymbol{\phi}(x) = [1, x, x^2, \dots, x^M]^\top$$

- Maximum likelihood estimator:

$$\boldsymbol{\theta}^{\mathrm{ML}} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y}$$

Example 2: Polynomial Regression

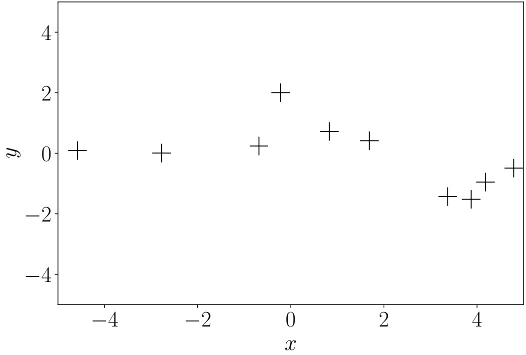


Figure: Training data

Example 2: Polynomial Regression

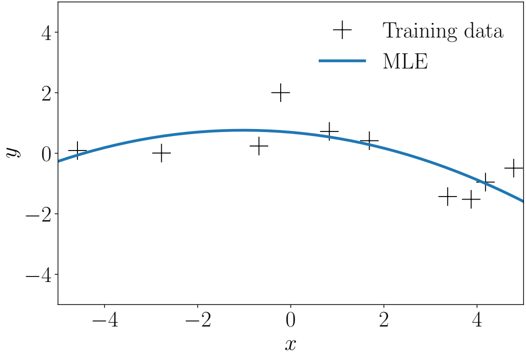


Figure: 2nd-order polynomial

Example 2: Polynomial Regression

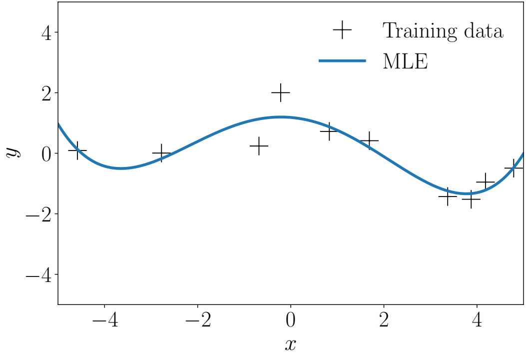


Figure: 4th-order polynomial

Example 2: Polynomial Regression

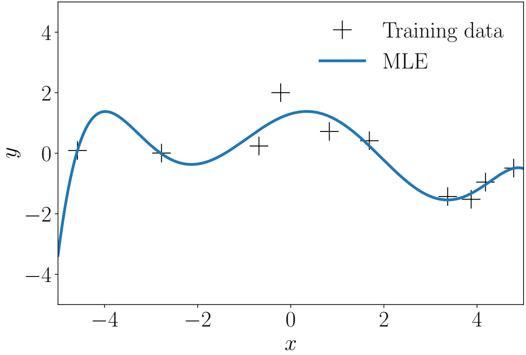


Figure: 6th-order polynomial

Example 2: Polynomial Regression

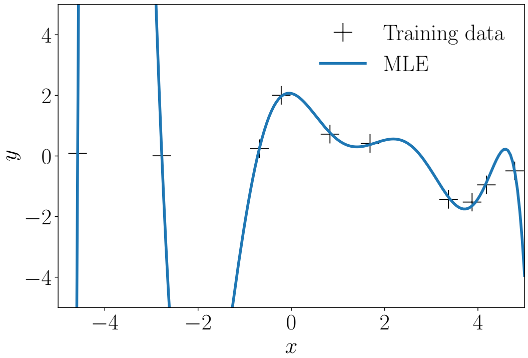


Figure: 8th-order polynomial

Example 2: Polynomial Regression

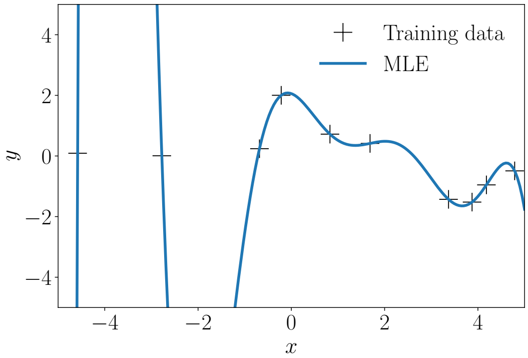
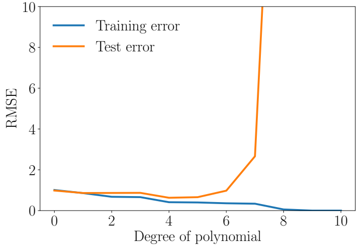


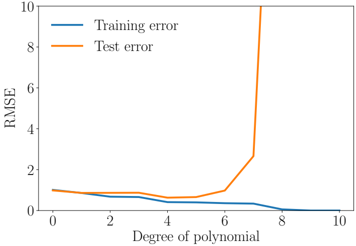
Figure: 10th-order polynomial

Overfitting



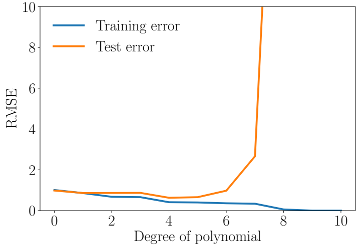
- Training error decreases with higher flexibility of the model

Overfitting



- Training error decreases with higher flexibility of the model
- We are not so much interested in the training error, but in the **generalization error**: How well does the model perform when we predict at previously unseen input locations?

Overfitting



- Training error decreases with higher flexibility of the model
- We are not so much interested in the training error, but in the **generalization error**: How well does the model perform when we predict at previously unseen input locations?
- Maximum likelihood often runs into **overfitting** problems, i.e., we exploit the flexibility of the model to fit to the noise in the data

MAP Estimation

- Observation: Parametric models that overfit tend to have some extreme (large amplitude) parameter values

MAP Estimation

- Observation: Parametric models that overfit tend to have some extreme (large amplitude) parameter values
- Mitigate the effect of overfitting by placing a prior distribution $p(\boldsymbol{\theta})$ on the parameters

  ▶▶ Penalize extreme values that are implausible under that prior

MAP Estimation

- Observation: Parametric models that overfit tend to have some extreme (large amplitude) parameter values
- Mitigate the effect of overfitting by placing a prior distribution $p(\boldsymbol{\theta})$ on the parameters
  - ▸▸ Penalize extreme values that are implausible under that prior
- Choose $\boldsymbol{\theta}^*$ as the parameter that maximizes the (log) parameter posterior

$$\log p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \underbrace{\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}_{\text{log-likelihood}} + \underbrace{\log p(\boldsymbol{\theta})}_{\text{log-prior}} + \text{const}$$

MAP Estimation

- Observation: Parametric models that overfit tend to have some extreme (large amplitude) parameter values
- Mitigate the effect of overfitting by placing a prior distribution $p(\boldsymbol{\theta})$ on the parameters
  - Penalize extreme values that are implausible under that prior
- Choose $\boldsymbol{\theta}^*$ as the parameter that maximizes the (log) parameter posterior

$$\log p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \underbrace{\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}_{\text{log-likelihood}} + \underbrace{\log p(\boldsymbol{\theta})}_{\text{log-prior}} + \text{const}$$

- Log-prior induces a direct penalty on the parameters

MAP Estimation

- Observation: Parametric models that overfit tend to have some extreme (large amplitude) parameter values
- Mitigate the effect of overfitting by placing a prior distribution $p(\boldsymbol{\theta})$ on the parameters
  - Penalize extreme values that are implausible under that prior
- Choose $\boldsymbol{\theta}^*$ as the parameter that maximizes the (log) parameter posterior

$$\log p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \underbrace{\log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})}_{\text{log-likelihood}} + \underbrace{\log p(\boldsymbol{\theta})}_{\text{log-prior}} + \text{const}$$

- Log-prior induces a direct penalty on the parameters
- **Maximum a posteriori estimate** (regularized least squares)

$$\boldsymbol{\theta}^{\text{MAP}} = (\boldsymbol{X}^\top\boldsymbol{X} + \tfrac{\sigma^2}{\alpha^2}\boldsymbol{I})^{-1}\boldsymbol{X}^\top\boldsymbol{y}$$

Example: Polynomial Regression

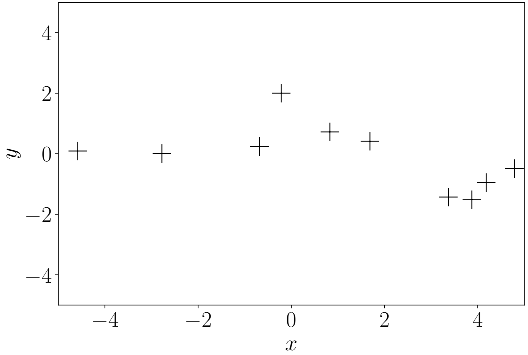


Figure: Training data

Mean prediction:

$$\mathbb{E}[y_* | \boldsymbol{x}_*, \boldsymbol{\theta}^*_{\text{MAP}}] = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\theta}^*_{\text{MAP}}$$

Example: Polynomial Regression

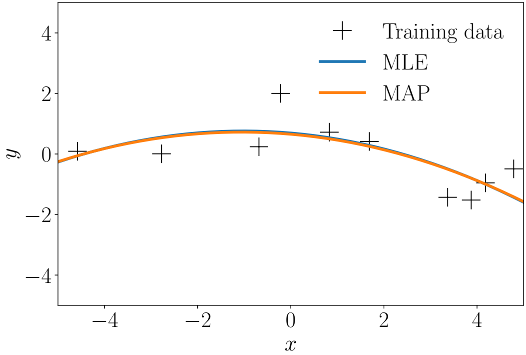


Figure: 2nd-order polynomial

Mean prediction:

$$\mathbb{E}[y_* | \boldsymbol{x}_*, \boldsymbol{\theta}^*_{\text{MAP}}] = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\theta}^*_{\text{MAP}}$$

Example: Polynomial Regression

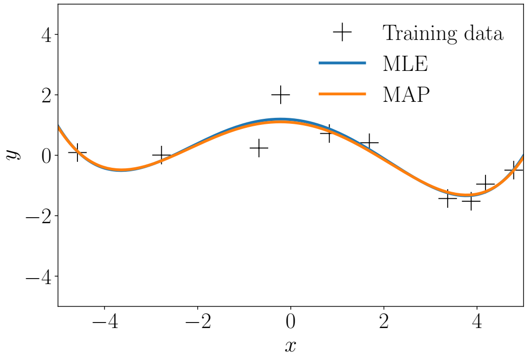


Figure: 4th-order polynomial

Mean prediction:

$$\mathbb{E}[y_* | \boldsymbol{x}_*, \boldsymbol{\theta}^*_{\text{MAP}}] = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\theta}^*_{\text{MAP}}$$

Example: Polynomial Regression

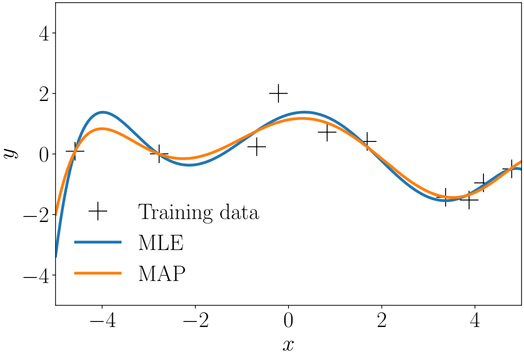


Figure: 6th-order polynomial

Mean prediction:

$$\mathbb{E}[y_* | \boldsymbol{x}_*, \boldsymbol{\theta}^*_{\text{MAP}}] = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\theta}^*_{\text{MAP}}$$

Example: Polynomial Regression

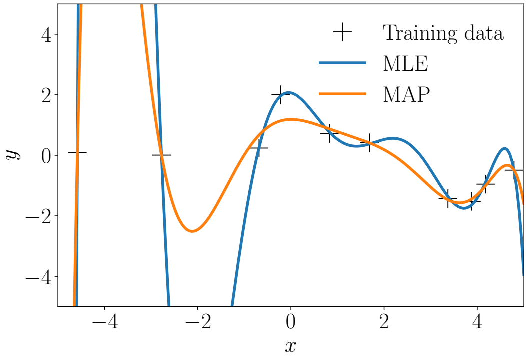


Figure: 8th-order polynomial

Mean prediction:

$$\mathbb{E}[y_* | \boldsymbol{x}_*, \boldsymbol{\theta}^*_{\text{MAP}}] = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\theta}^*_{\text{MAP}}$$

Example: Polynomial Regression

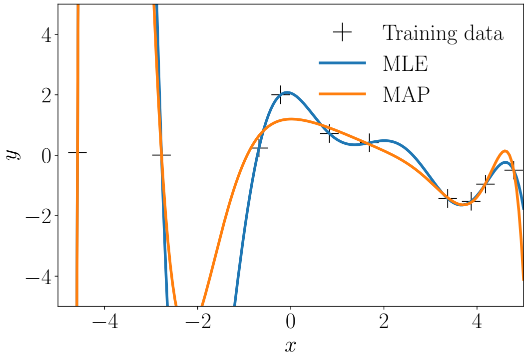
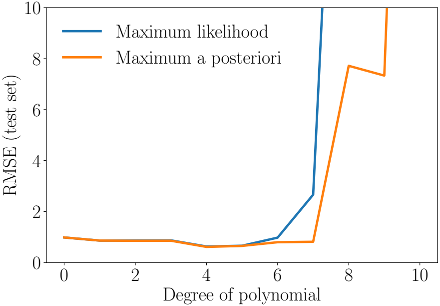


Figure: 10th-order polynomial

Mean prediction:

$$\mathbb{E}[y_* | \boldsymbol{x}_*, \boldsymbol{\theta}^*_{\text{MAP}}] = \boldsymbol{\phi}(\boldsymbol{x}_*)^\top \boldsymbol{\theta}^*_{\text{MAP}}$$

Generalization Error



- MAP estimation “delays” the problem of overfitting
- It does not provide a general solution

▶▶ Need a more principled solution

Bayesian Linear Regression

$$y = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta} + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0, \sigma^2\big)$$

- Avoid overfitting by not fitting any parameters:
  ⏩ Integrate parameters out instead of optimizing them

Bayesian Linear Regression

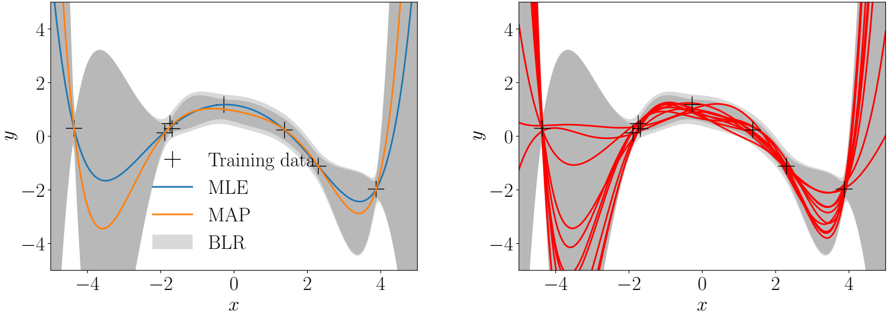
$$y = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\theta} + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma^2\big)$$

- Avoid overfitting by not fitting any parameters:
  ⏩ Integrate parameters out instead of optimizing them
- Use a full parameter distribution $p(\boldsymbol{\theta})$ (and not a single point estimate $\boldsymbol{\theta}^*$) when making predictions:

$$p(y_*|\boldsymbol{x}_*) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}$$

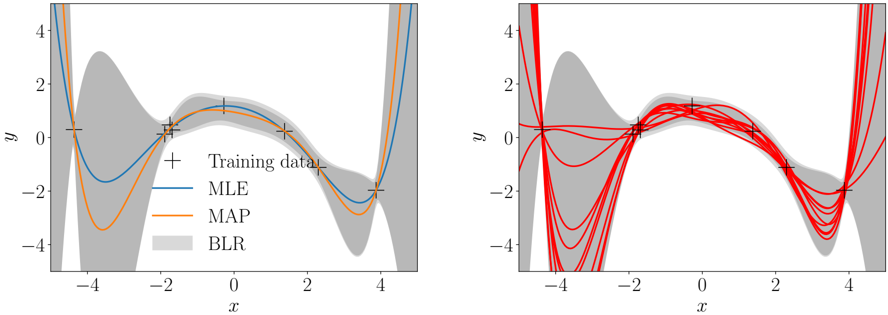
  ⏩ Prediction no longer depends on $\boldsymbol{\theta}$
- Predictive distribution reflects the uncertainty about the "correct" parameter setting

Example



- Light-gray: uncertainty due to noise
- Dark-gray: uncertainty due to parameter uncertainty

Example



- Light-gray: uncertainty due to noise
- Dark-gray: uncertainty due to parameter uncertainty
- Right: Plausible functions under the parameter distribution (every single parameter setting describes one function)

Model for Bayesian Linear Regression

$$\text{Prior} \qquad p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_0,\, \boldsymbol{S}_0\big)\,,$$
$$\text{Likelihood} \quad p(y|\boldsymbol{x},\boldsymbol{\theta}) = \mathcal{N}\big(y \,|\, \boldsymbol{\phi}^\top(\boldsymbol{x})\boldsymbol{\theta},\, \sigma^2\big)$$

- Parameter $\boldsymbol{\theta}$ becomes a latent (random) variable
- Prior distribution induces a distribution over plausible functions
- Choose a conjugate Gaussian prior
  - Closed-form computations
  - Gaussian posterior

Parameter Posterior and Predictions

- Prior $p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_0,\, \boldsymbol{S}_0\big)$ is Gaussian ⏩ posterior is Gaussian:

$$
\begin{aligned}
p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) &= \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big) \\
\boldsymbol{S}_N &= (\boldsymbol{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1} \\
\boldsymbol{m}_N &= \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y})
\end{aligned}
$$

Parameter Posterior and Predictions

- Prior $p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_0,\, \boldsymbol{S}_0\big)$ is Gaussian ⏩ posterior is Gaussian:

$$
\begin{aligned}
p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) &= \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big) \\
\boldsymbol{S}_N &= (\boldsymbol{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1} \\
\boldsymbol{m}_N &= \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y})
\end{aligned}
$$

- Mean $\boldsymbol{m}_N$ identical to MAP estimate

Parameter Posterior and Predictions

- Prior $p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_0,\, \boldsymbol{S}_0\big)$ is Gaussian ⏩ posterior is Gaussian:

$$
\begin{aligned}
p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) &= \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big) \\
\boldsymbol{S}_N &= (\boldsymbol{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1} \\
\boldsymbol{m}_N &= \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y})
\end{aligned}
$$

- Mean $\boldsymbol{m}_N$ identical to MAP estimate
- Assume a Gaussian distribution $p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big)$. Then

$$
p(y_*|\boldsymbol{x}_*) = \mathcal{N}\big(y\,|\,\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{m}_N,\, \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*) + \sigma^2\big)
$$

Parameter Posterior and Predictions

- Prior $p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_0,\, \boldsymbol{S}_0\big)$ is Gaussian ⏩ posterior is Gaussian:

$$
\begin{aligned}
p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) &= \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big) \\
\boldsymbol{S}_N &= (\boldsymbol{S}_0^{-1} + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{\Phi})^{-1} \\
\boldsymbol{m}_N &= \boldsymbol{S}_N(\boldsymbol{S}_0^{-1}\boldsymbol{m}_0 + \sigma^{-2}\boldsymbol{\Phi}^\top\boldsymbol{y})
\end{aligned}
$$

- Mean $\boldsymbol{m}_N$ identical to MAP estimate
- Assume a Gaussian distribution $p(\boldsymbol{\theta}) = \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big)$. Then

$$
p(y_*|\boldsymbol{x}_*) = \mathcal{N}\big(y \,|\, \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{m}_N,\, \boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*) + \sigma^2\big)
$$

- $\boldsymbol{\phi}^\top(\boldsymbol{x}_*)\boldsymbol{S}_N\boldsymbol{\phi}(\boldsymbol{x}_*)$: Contribution to predictive variance due to parameter uncertainty

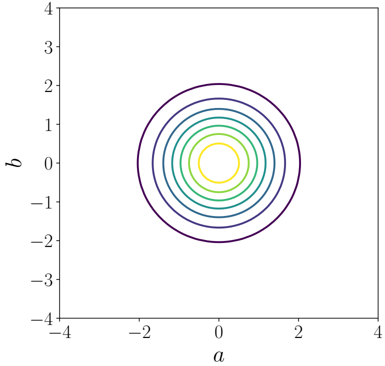
  More details ⏩ `https://mml-book.com`, Chapter 9

Distribution over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$

$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$



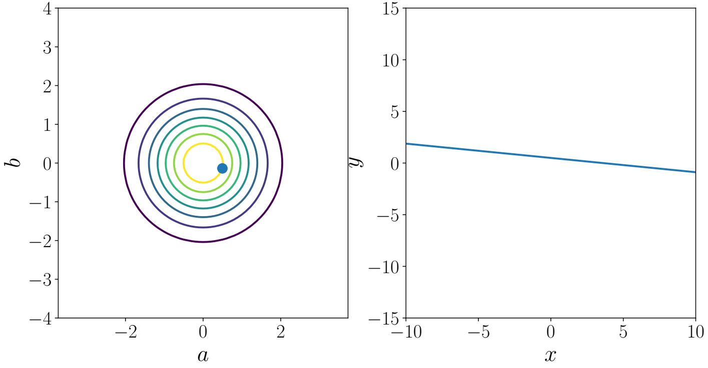
Sampling from the Prior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$

$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$

$$f_i(x) = a_i + b_i x, \quad [a_i, b_i] \sim p(a, b)$$



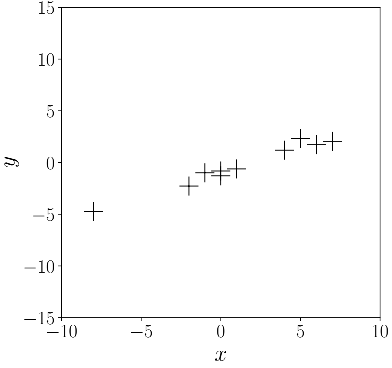
Sampling from the Posterior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$

$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$

$$\boldsymbol{X} = [x_1, \ldots, x_N],\ \boldsymbol{y} = [y_1, \ldots, y_N] \quad \text{Training inputs/targets}$$



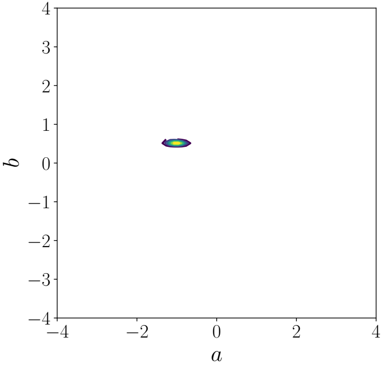
Sampling from the Posterior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$

$$p(a, b) = \mathcal{N}\big(\mathbf{0},\, \boldsymbol{I}\big)$$

$$p(a, b|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{m}_N,\, \boldsymbol{S}_N\big) \qquad \text{Posterior}$$



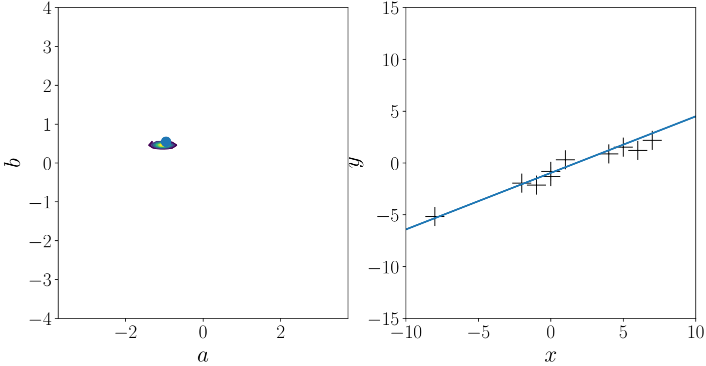
Sampling from the Posterior over Functions

Consider a linear regression setting

$$y = f(x) + \epsilon = a + bx + \epsilon\,, \quad \epsilon \sim \mathcal{N}\big(0,\, \sigma_n^2\big)$$

$$[a_i, b_i] \sim p(a, b|\boldsymbol{X}, \boldsymbol{y})$$

$$f_i = a_i + b_i x$$



Fitting Nonlinear Functions

- Fit nonlinear functions using (Bayesian) linear regression:
  Linear combination of nonlinear features

Fitting Nonlinear Functions

- Fit nonlinear functions using (Bayesian) linear regression: Linear combination of nonlinear features
- Example: Radial-basis-function (RBF) network

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \phi_i(\boldsymbol{x}), \quad \theta_i \sim \mathcal{N}\big(0, \sigma_p^2\big)$$

Fitting Nonlinear Functions

- Fit nonlinear functions using (Bayesian) linear regression:
  Linear combination of nonlinear features
- Example: Radial-basis-function (RBF) network

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \phi_i(\boldsymbol{x}), \quad \theta_i \sim \mathcal{N}(0, \sigma_p^2)$$

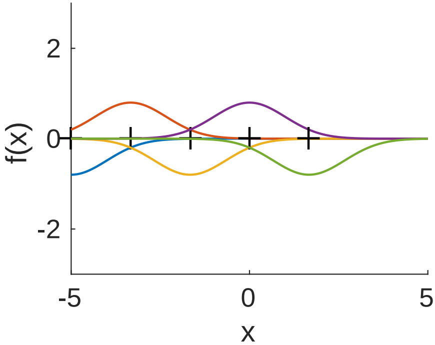
  where

$$\phi_i(\boldsymbol{x}) = \exp\left(-\tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^\top (\boldsymbol{x} - \boldsymbol{\mu}_i)\right)$$

  for given "centers" $\boldsymbol{\mu}_i$

Illustration: Fitting a Radial Basis Function Network

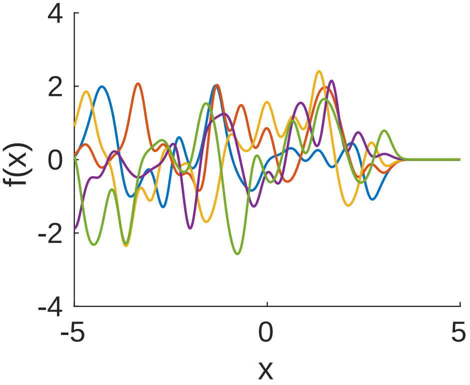
$$\phi_i(\boldsymbol{x}) = \exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i)\right)$$



- Place Gaussian-shaped basis functions ϕ_i at 25 input locations μ_i, linearly spaced in the interval $[-5, 3]$

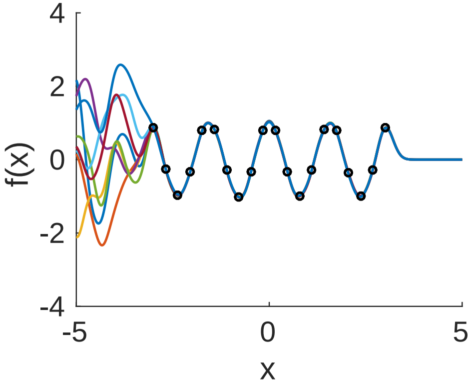
Samples from the RBF Prior

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \phi_i(\boldsymbol{x}), \quad p(\boldsymbol{\theta}) = \mathcal{N}\big(\mathbf{0}, \boldsymbol{I}\big)$$

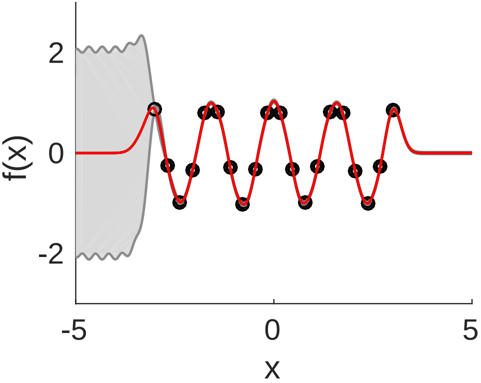


Samples from the RBF Posterior

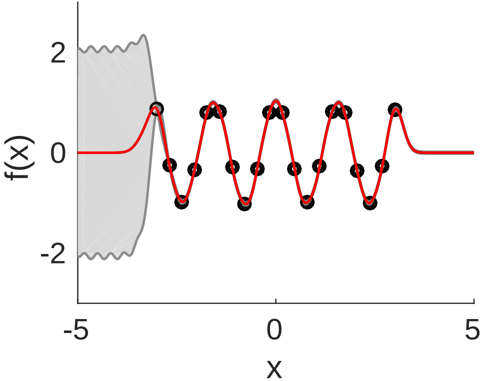
$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \theta_i \phi_i(\boldsymbol{x}), \quad p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(\boldsymbol{m}_N, \boldsymbol{S}_N\big)$$



RBF Posterior



Limitations



- Feature engineering (what basis functions to use?)
- Finite number of features:
  - Above: Without basis functions on the right, we cannot express any variability of the function
  - Ideally: Add more (infinitely many) basis functions

Approach

- Instead of sampling parameters, which induce a distribution over functions, sample functions directly
  - Place a prior directly on functions
  - Make assumptions on the distribution of functions

Approach

- Instead of sampling parameters, which induce a distribution over functions, sample functions directly
  - ⏩ Place a prior directly on functions
  - ⏩ Make assumptions on the distribution of functions
- Intuition: function = infinitely long vector of function values
  - ⏩ Make assumptions on the distribution of function values

Approach

- Instead of sampling parameters, which induce a distribution over functions, sample functions directly
  - ⏩ Place a prior directly on functions
  - ⏩ Make assumptions on the distribution of functions
- Intuition: function = infinitely long vector of function values
  - ⏩ Make assumptions on the distribution of function values

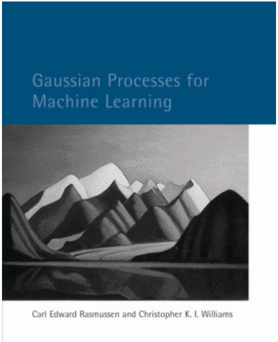
Approach

- Instead of sampling parameters, which induce a distribution over functions, sample functions directly
  - ⏩ Place a prior directly on functions
  - ⏩ Make assumptions on the distribution of functions
- Intuition: function = infinitely long vector of function values
  - ⏩ Make assumptions on the distribution of function values

⏩ **Gaussian process**

Overview

Two-Slide Introduction to Gaussian Processes



www.gaussianprocess.org

- GP Summer School http://gpss.cc
- Video lecture by Richard Turner

  https://tinyurl.com/y5l6dzsa

Gaussian Processes

- Very flexible Bayesian regression method
- Implements a probability distribution over functions
- Fully specified by
  - Mean function m (average function)
  - Covariance function k (assumptions on structure)

$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \mathrm{Cov}[f(\boldsymbol{x}_p), f(\boldsymbol{x}_q)]$$

Gaussian Processes

- Very flexible Bayesian regression method
- Implements a probability distribution over functions
- Fully specified by
  - Mean function m (average function)
  - Covariance function k (assumptions on structure)

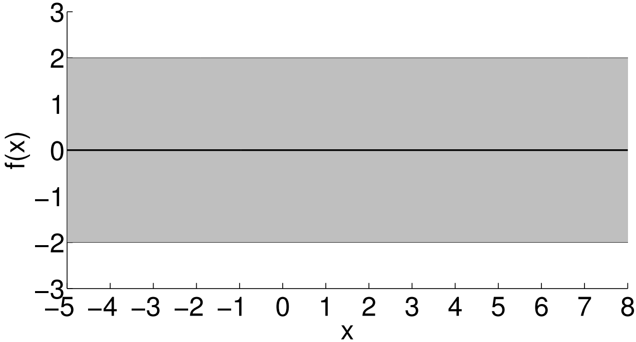
$$k(\boldsymbol{x}_p, \boldsymbol{x}_q) = \mathrm{Cov}[f(\boldsymbol{x}_p), f(\boldsymbol{x}_q)]$$

- Posterior predictive distribution at $\boldsymbol{x}_*$ is Gaussian (Bayes' theorem):

$$p(f(\boldsymbol{x}_*) | \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\big(f(\boldsymbol{x}_*) \,|\, m(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*)\big)$$

Test input ($\boldsymbol{x}_*$)   Training data ($\boldsymbol{X}, \boldsymbol{y}$)

Intuitive Introduction to Gaussian Processes

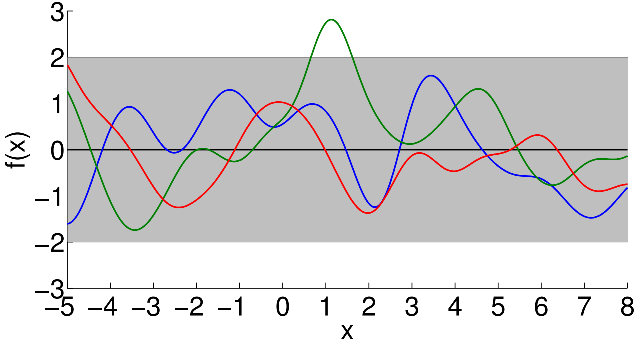


Prior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \varnothing] = m(\boldsymbol{x}_*) = 0$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \varnothing] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

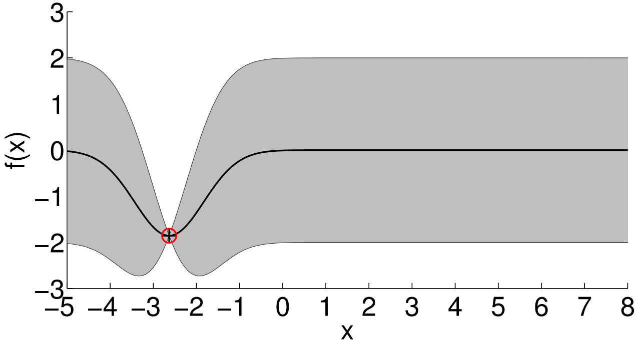


Prior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \varnothing] = m(\boldsymbol{x}_*) = 0$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \varnothing] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

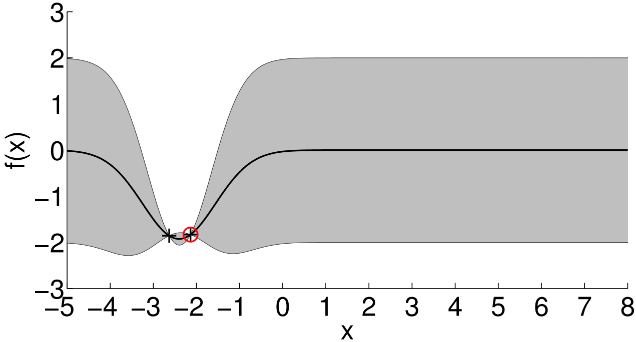


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

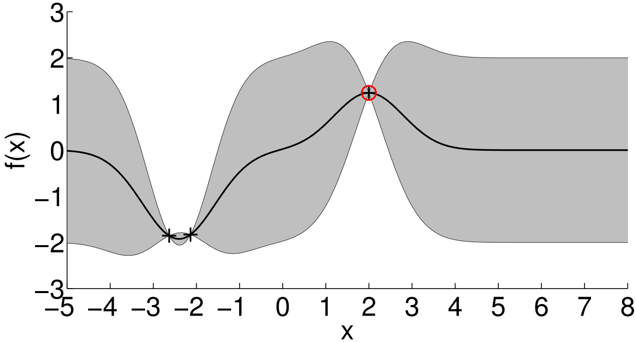


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

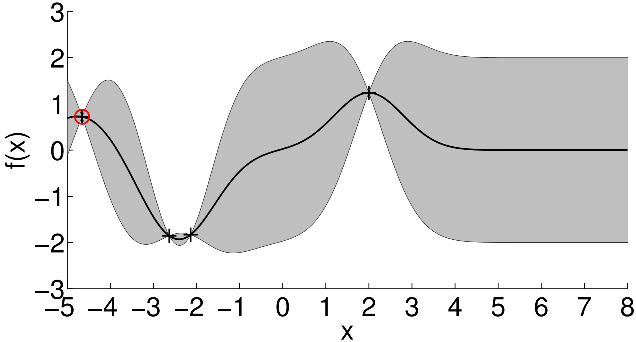


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

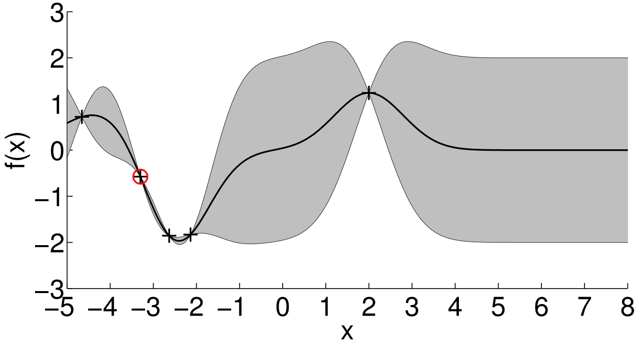


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

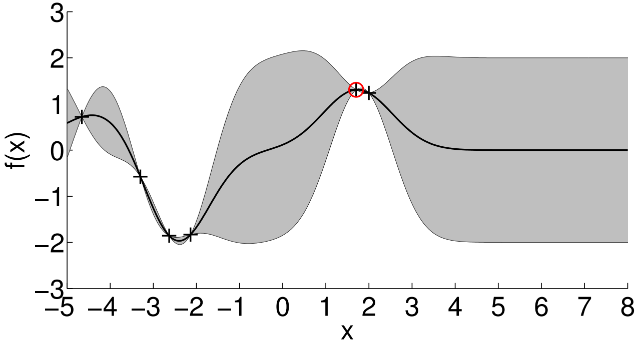


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

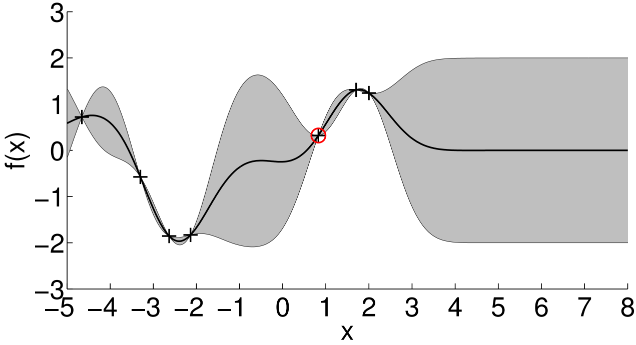


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes

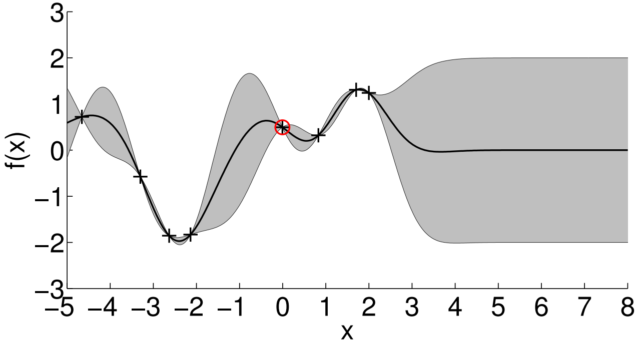


Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes



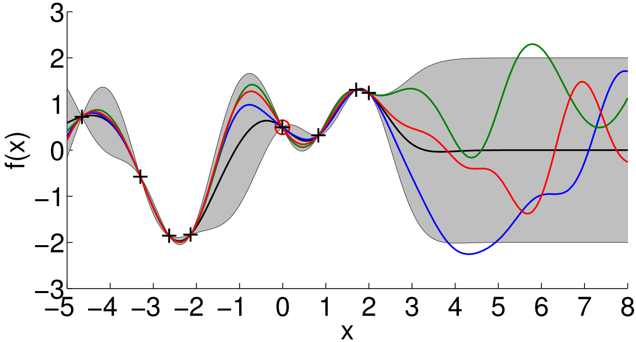
Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$

$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Intuitive Introduction to Gaussian Processes



Posterior belief about the function

Predictive (marginal) mean and variance:

$$\mathbb{E}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = m(\boldsymbol{x}_*) = k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{y}$$
$$\mathbb{V}[f(\boldsymbol{x}_*)|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}] = \sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - k(\boldsymbol{X}, \boldsymbol{x}_*)^\top k(\boldsymbol{X}, \boldsymbol{X})^{-1} k(\boldsymbol{X}, \boldsymbol{x}_*)$$

Overview

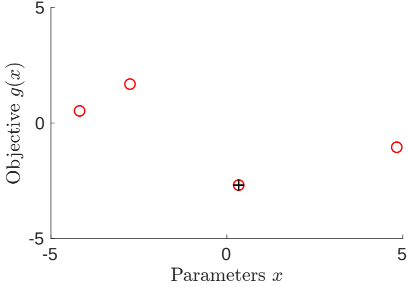
Bayesian Optimization with Gaussian Processes

Bayesian Optimization Setting

- Objective: Find global minimum of objective function g:

$$\boldsymbol{x}_* = \arg\min_{\boldsymbol{x}} g(\boldsymbol{x})$$

- We can evaluate the objective g pointwise, but do not have an easy functional form or gradients; observations may be noisy
- Evaluating g is costly (e.g., train a massive deep network)



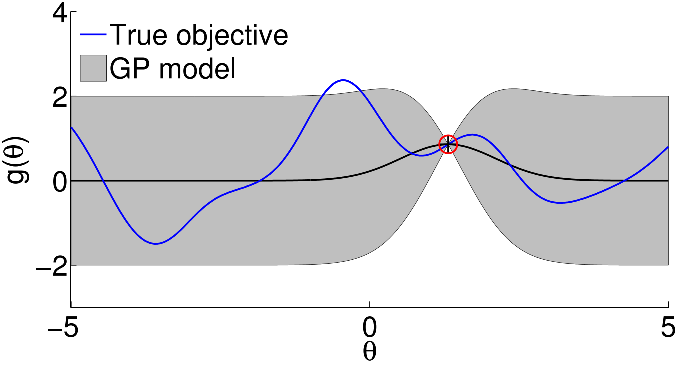
Key Steps

- To avoid evaluating g an excessive number of times, approximate it using a proxy function $\tilde{g}$ (which is cheap to evaluate)
  ⏩ Gaussian process
- Find a global optimum $\tilde{g}(\boldsymbol{x}_*)$ of proxy function $\tilde{g}$
- Evaluate true objective g at $\boldsymbol{x}_*$
- Overall: Evaluate g only once

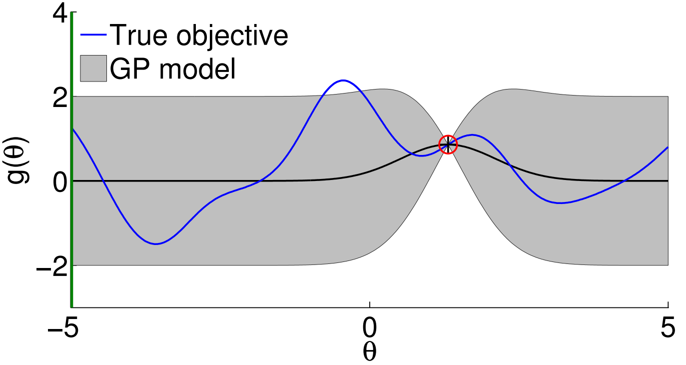
Key Steps

- To avoid evaluating g an excessive number of times, approximate it using a proxy function $\tilde{g}$ (which is cheap to evaluate)
  ⏩ Gaussian process
- Find a global optimum $\tilde{g}(\boldsymbol{x}_*)$ of proxy function $\tilde{g}$
- Evaluate true objective g at $\boldsymbol{x}_*$
- Overall: Evaluate g only once
- Works well if $\tilde{g} \approx g$.
- Usually not the case ⏩ Repeat this cycle and keep updating $\tilde{g}$

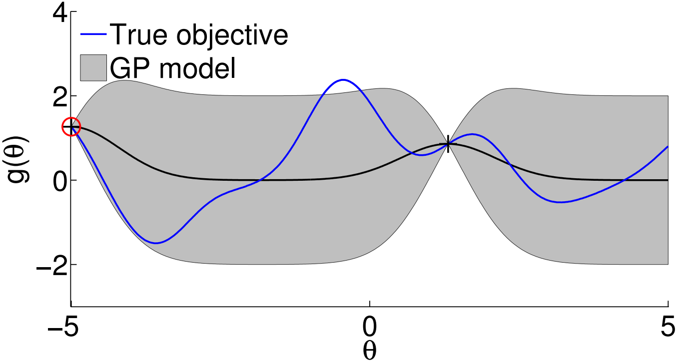
Bayesian Optimization: Illustration



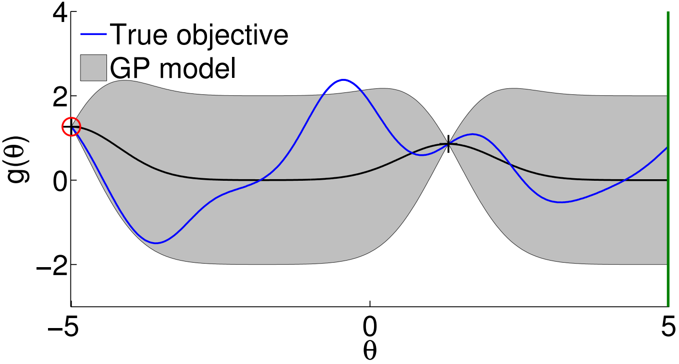
Bayesian Optimization: Illustration



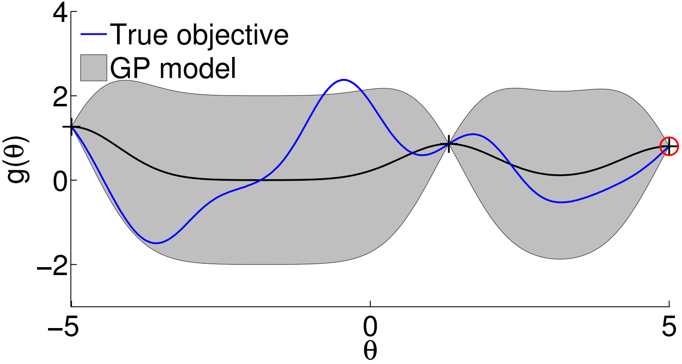
Bayesian Optimization: Illustration



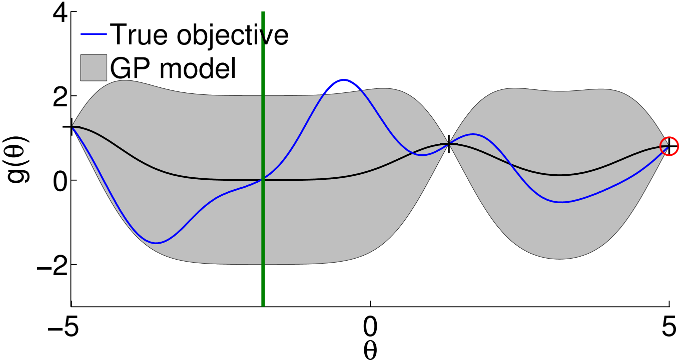
Bayesian Optimization: Illustration



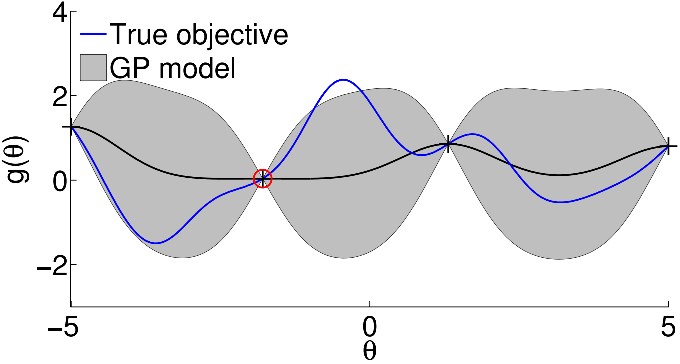
Bayesian Optimization: Illustration



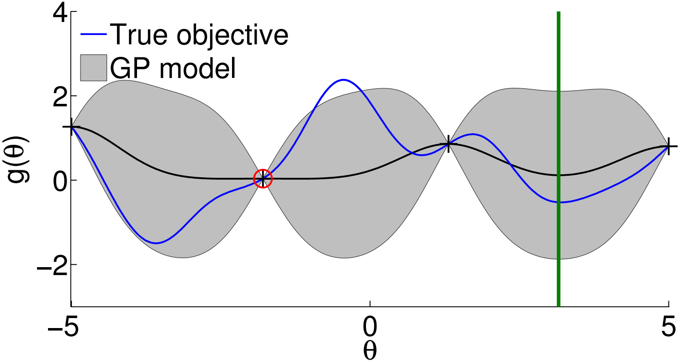
Bayesian Optimization: Illustration



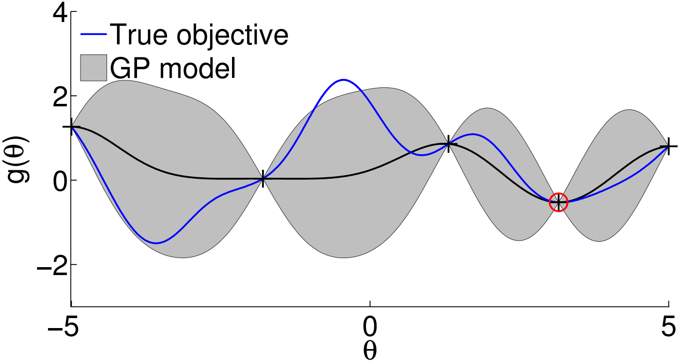
Bayesian Optimization: Illustration



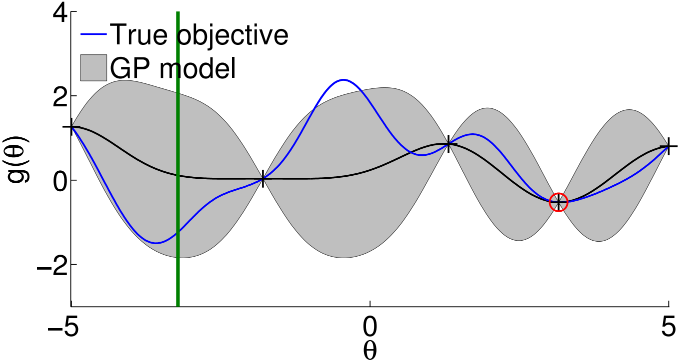
Bayesian Optimization: Illustration



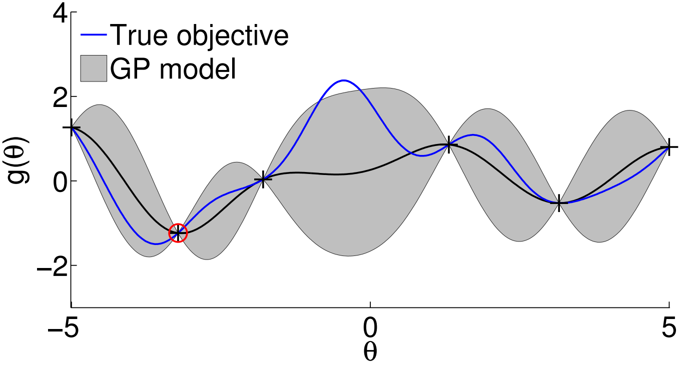
Bayesian Optimization: Illustration



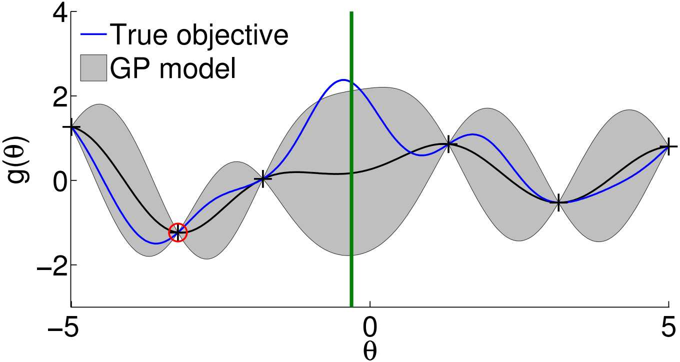
Bayesian Optimization: Illustration



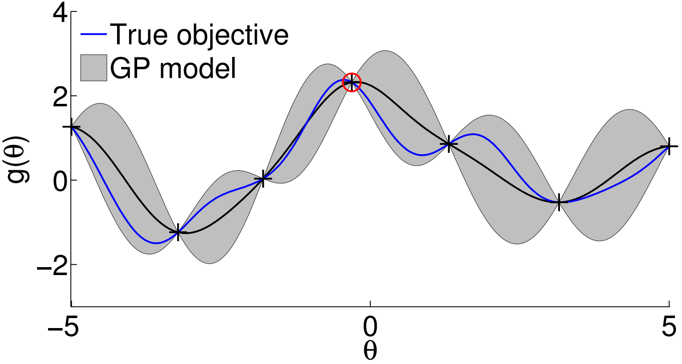
Bayesian Optimization: Illustration



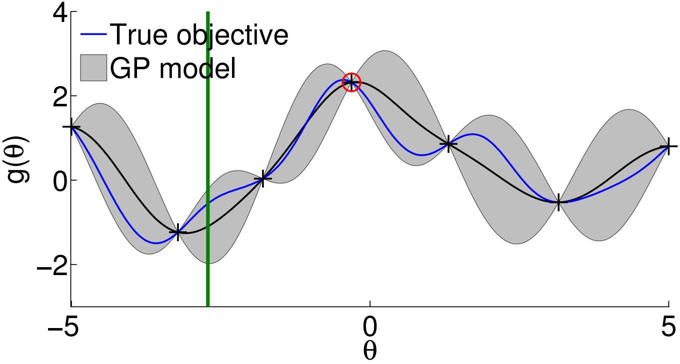
Bayesian Optimization: Illustration



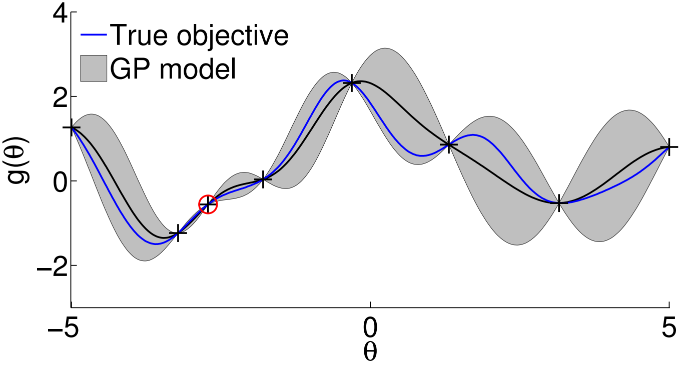
Bayesian Optimization: Illustration



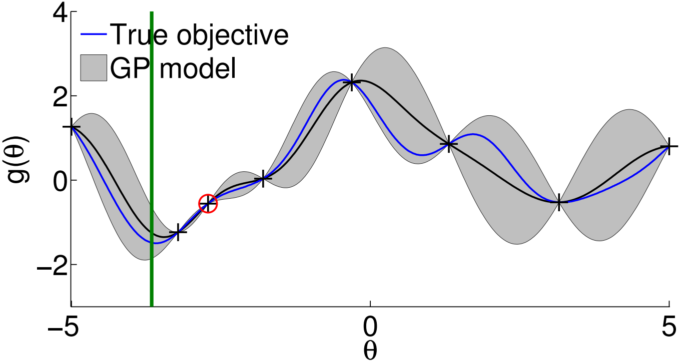
Bayesian Optimization: Illustration



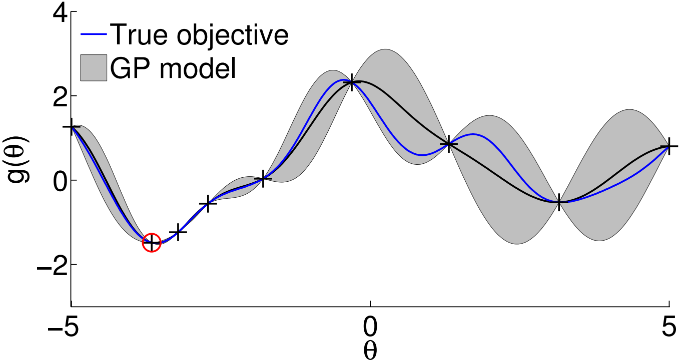
Bayesian Optimization: Illustration



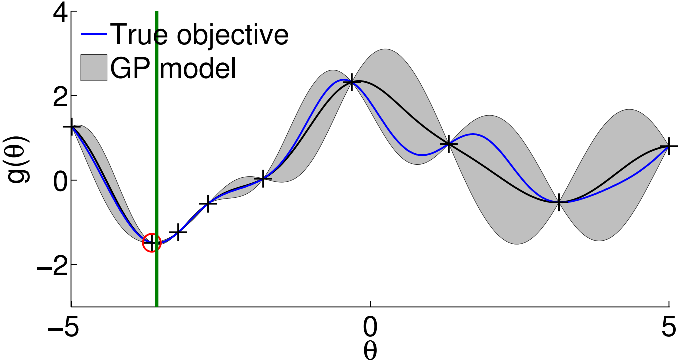
Bayesian Optimization: Illustration



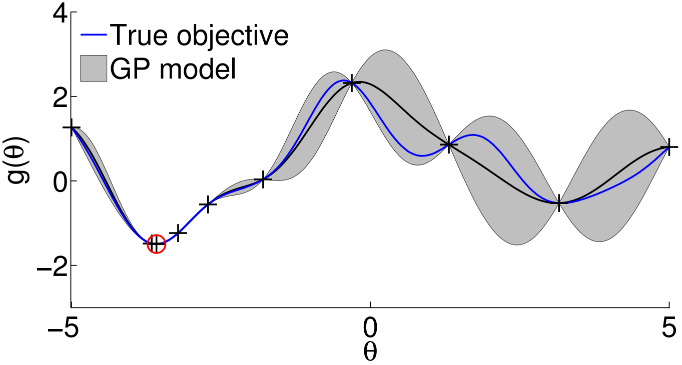
Bayesian Optimization: Illustration



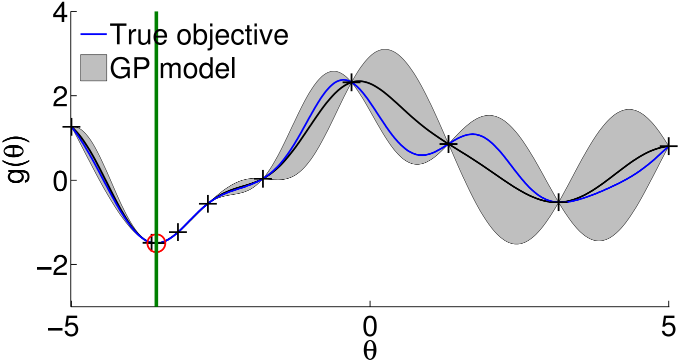
Bayesian Optimization: Illustration



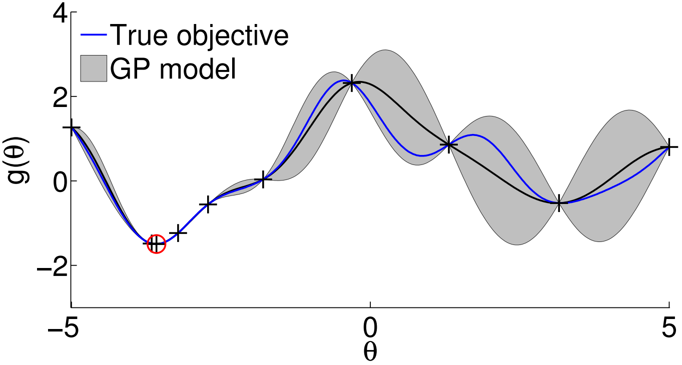
Bayesian Optimization: Illustration



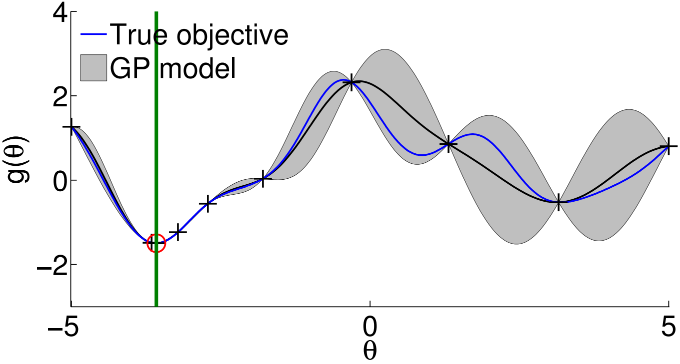
Bayesian Optimization: Illustration



Bayesian Optimization: Illustration

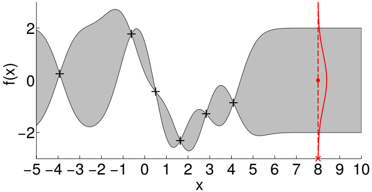


Bayesian Optimization: Illustration



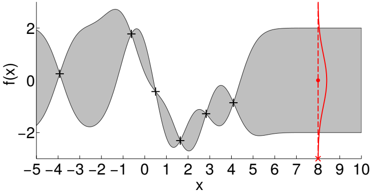
Choosing the Next Point to Evaluate the True Objective: Acquisition Functions

Using Uncertainty in Global Optimization



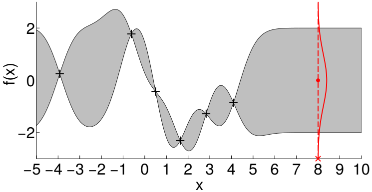
- Find a good (global) optimum
  - Need to get out of local optima

Using Uncertainty in Global Optimization



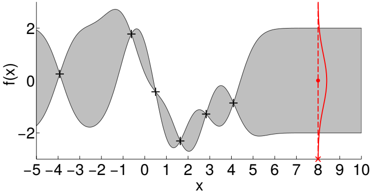
- Find a good (global) optimum
  - Need to get out of local optima
- Extrapolate from collected knowledge

Using Uncertainty in Global Optimization



- Find a good (global) optimum
  - ▶▶ Need to get out of local optima
- Extrapolate from collected knowledge
- GP gives us closed-form means and variances
  - ▶▶ Trade off exploration and exploitation
    - **Exploration:** Seek places with high variance/uncertainty
    - **Exploitation:** Seek places with low mean

Using Uncertainty in Global Optimization

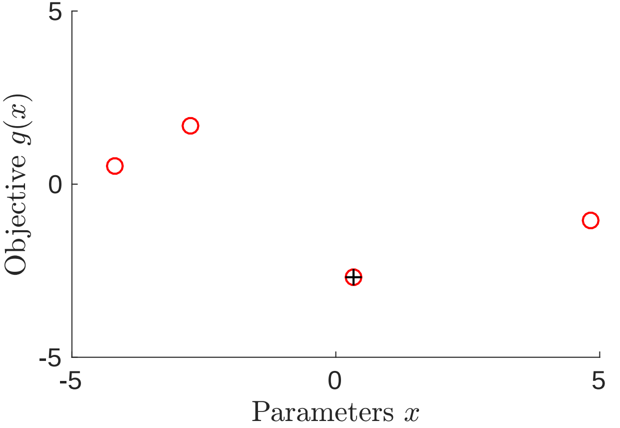


- Find a good (global) optimum
  - ▶▶ Need to get out of local optima
- Extrapolate from collected knowledge
- GP gives us closed-form means and variances
  - ▶▶ Trade off exploration and exploitation
    - **Exploration:** Seek places with high variance/uncertainty
    - **Exploitation:** Seek places with low mean
- Acquisition function α trades off exploration and exploitation for our proxy optimization

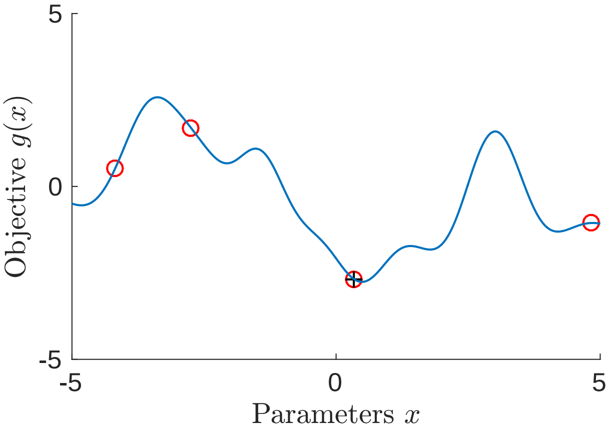
Key Steps (Pseudo-Code)

Init: Data set $\mathcal{D}_0 = \{\boldsymbol{X}_0, \boldsymbol{y}_0\}$
for iterations $t = 1, 2, \dots$ **do**
    Update GP using data $\mathcal{D}_{t-1}$
    Select $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}} \alpha(\boldsymbol{x})$ by optimizing acquisition function
    Query true objective g at $\boldsymbol{x}_t$
    Augment data set $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\boldsymbol{x}_t, y_t)\}$
end for
Return best input in data set: $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} y(\boldsymbol{x})$

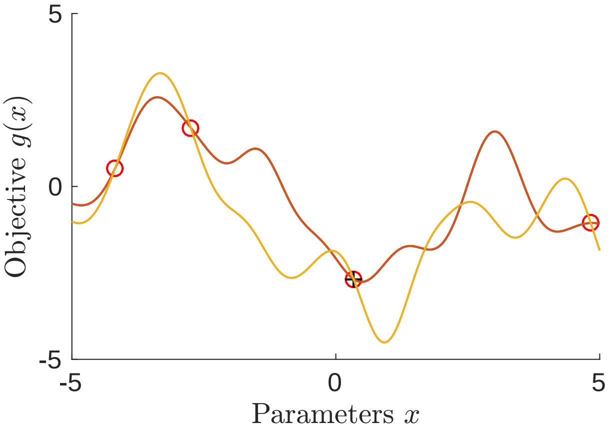
Where to Evaluate Next?



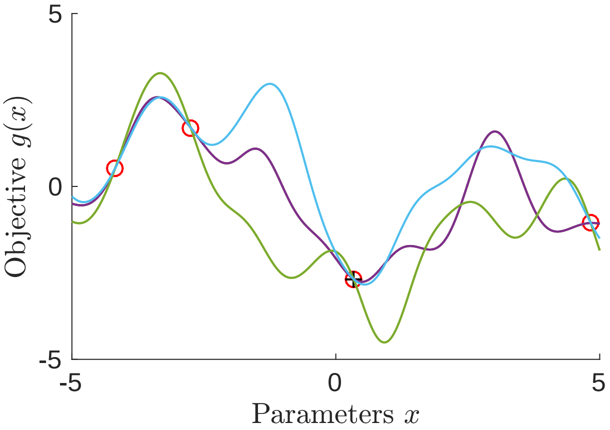
Where to Evaluate Next?



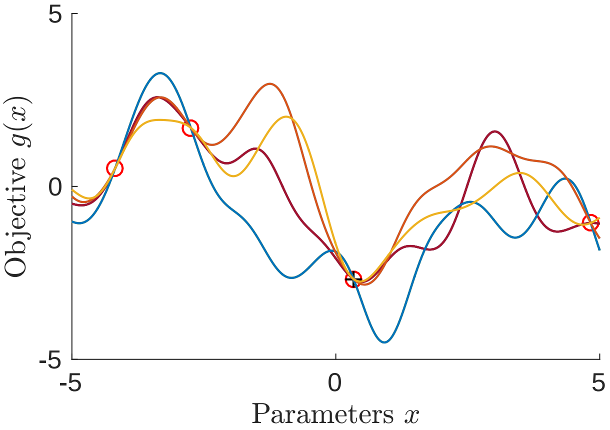
Where to Evaluate Next?



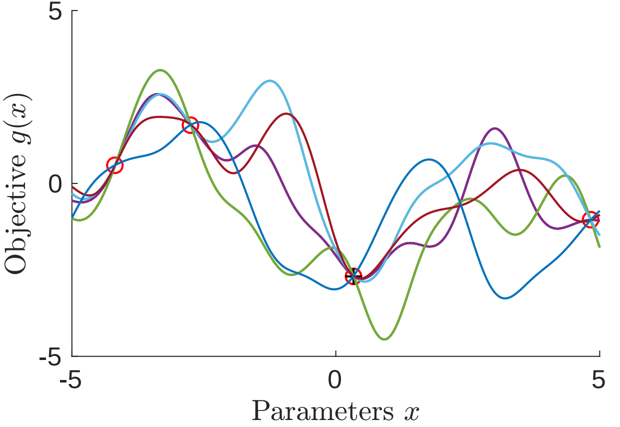
Where to Evaluate Next?



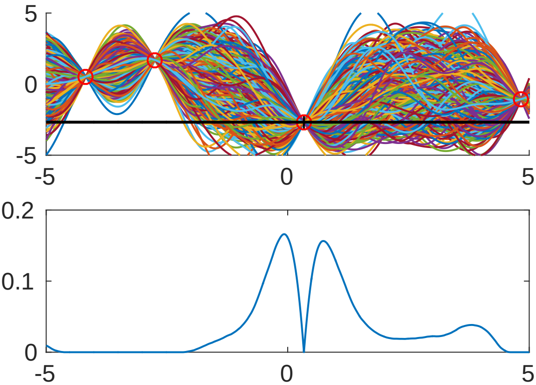
Where to Evaluate Next?



Where to Evaluate Next?

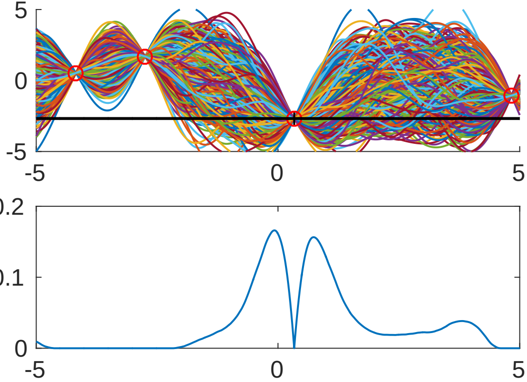


Where to Evaluate Next to Improve Most?



- Upper panel: Samples from a probabilistic proxy $\tilde{g}$

Where to Evaluate Next to Improve Most?



- Upper panel: Samples from a probabilistic proxy $\tilde{g}$
- Lower panel: Corresponding expected improvement over the best solution so far (black cross)
  - ▶▶ Evaluate g at the maximum of the expected improvement

Closed-Form Acquisition Functions

- For all $\boldsymbol{x} \in \mathbb{R}^D$ the GP posterior gives a predictive mean $\mu(\boldsymbol{x})$ variance $\sigma^2(\boldsymbol{x})$ of $g(\boldsymbol{x})$
- Define

$$\gamma(\boldsymbol{x}) = \frac{g(\boldsymbol{x}_{\text{best}}) - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}$$

Closed-Form Acquisition Functions

- For all $\boldsymbol{x} \in \mathbb{R}^D$ the GP posterior gives a predictive mean $\mu(\boldsymbol{x})$ variance $\sigma^2(\boldsymbol{x})$ of $g(\boldsymbol{x})$
- Define

$$\gamma(\boldsymbol{x}) = \frac{g(\boldsymbol{x}_{\text{best}}) - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}$$

- **Probability of Improvement (Kushner 1964):**

$$\alpha_{\text{PI}}(\boldsymbol{x}) = \Phi(\gamma(\boldsymbol{x}))$$

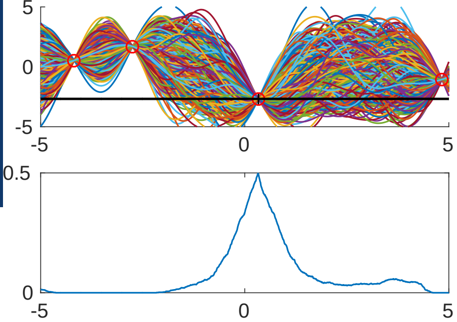
- **Expected Improvement (Mockus 1978):**

$$\alpha_{\text{EI}}(\boldsymbol{x}) = \sigma(\boldsymbol{x})\big(\gamma(\boldsymbol{x})\Phi(\gamma(\boldsymbol{x})) + \mathcal{N}\big(\gamma(\boldsymbol{x})\,|\,0,\,1\big)\big)$$

- **GP Lower Confidence Bound (Srinivas et al., 2010):**

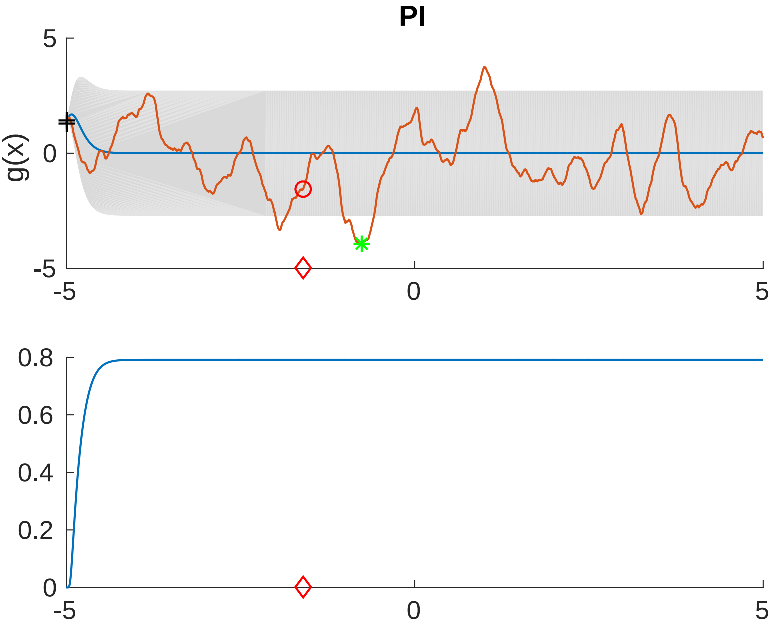
$$\alpha_{\text{LCB}}(\boldsymbol{x}) = -(\mu(\boldsymbol{x}) - \kappa\sigma(\boldsymbol{x}))\,, \quad \kappa > 0$$

Probability of Improvement

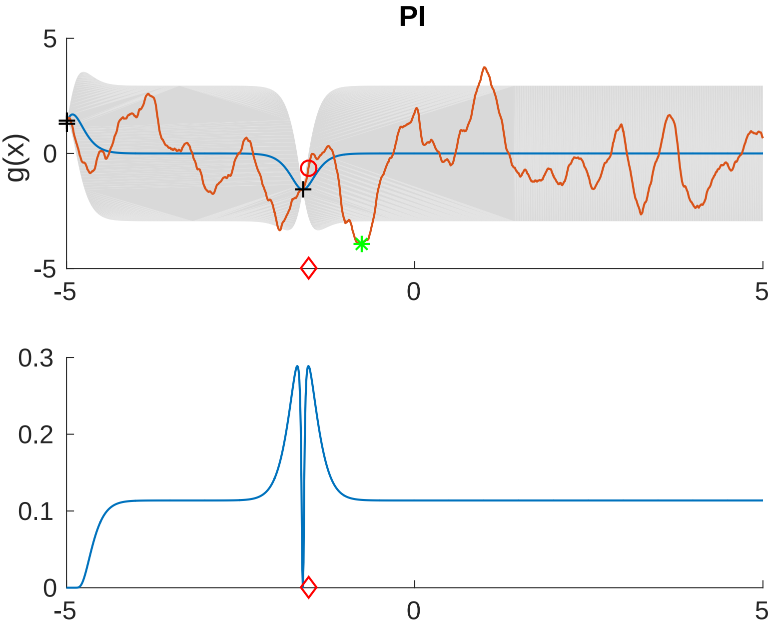


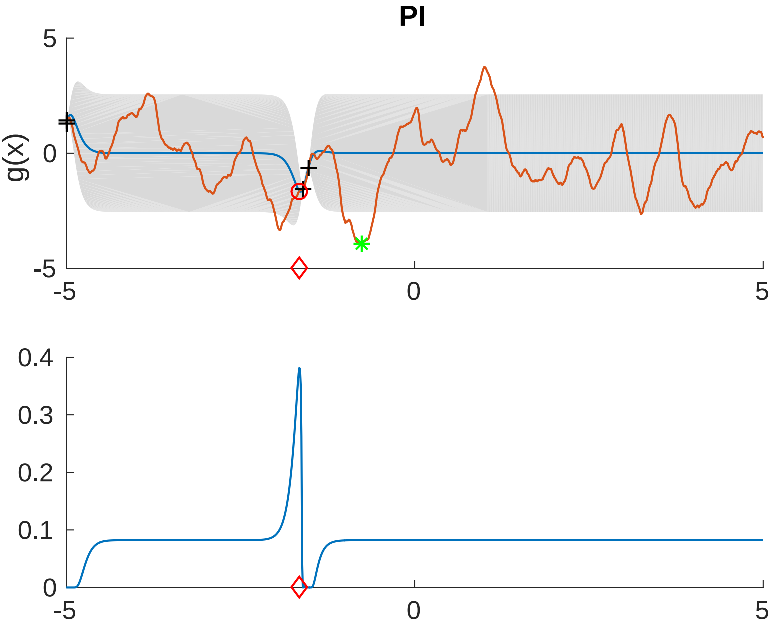
- Idea: Determine the probability that $\boldsymbol{x}_*$ leads to a better function value than the currently best one $g(\boldsymbol{x}_{\text{best}})$
- Sampling-based setting: Sample N functions g_i; at every input $\boldsymbol{x}$ compute a Monte-Carlo estimate

$$\alpha_{\text{PI}}(\boldsymbol{x}) = p(g(\boldsymbol{x}) < g(\boldsymbol{x}_{\text{best}})) \approx \frac{1}{N} \sum_{i=1}^{N} \delta\big(g_i(\boldsymbol{x}) < g(\boldsymbol{x}_{\text{best}})\big)$$

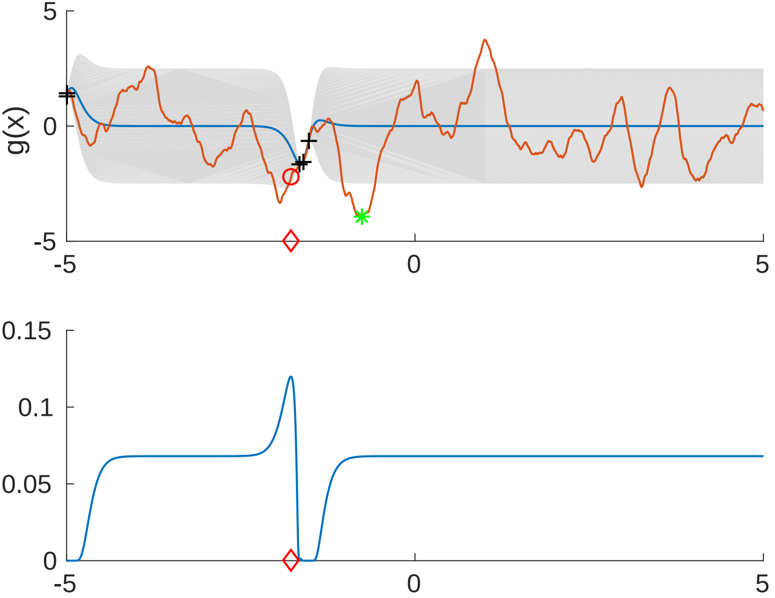


PI
5
0
-5
g(x)
-5
0
5
0.8
0.6
0.4
0.2
0
-5
0
5

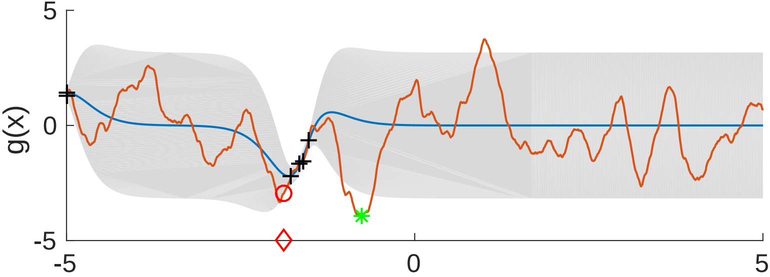

PI
g(x)
5
0
-5
-5
0
5
0.3
0.2
0.1
0
-5
0
5

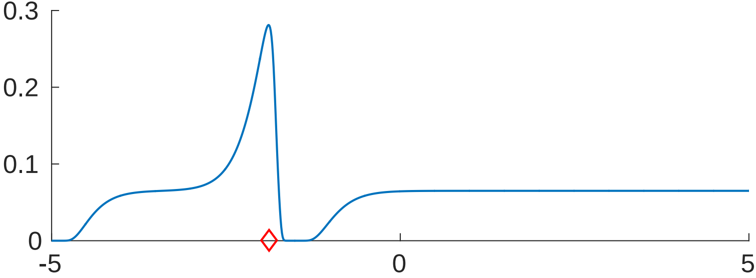

PI
5
0
-5
g(x)
-5
0
5
0.4
0.3
0.2
0.1
0
-5
0
5

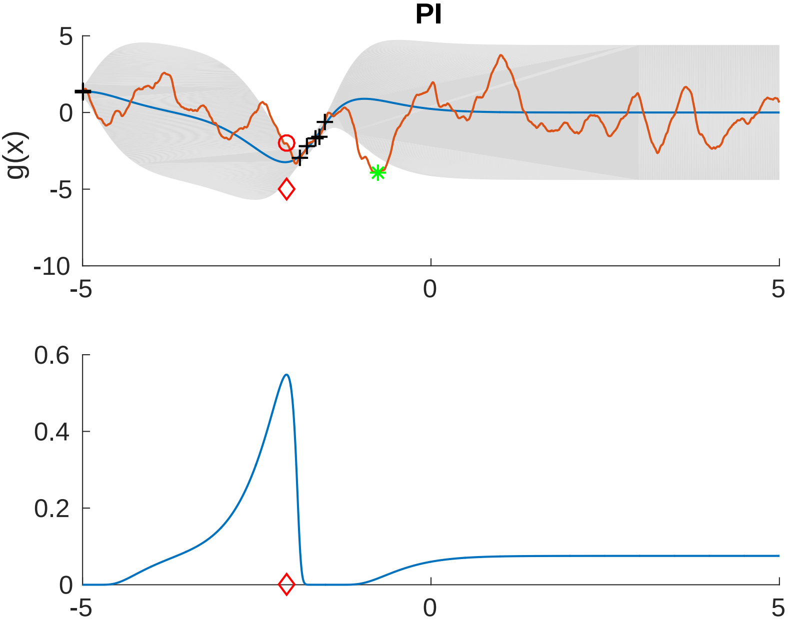
PI

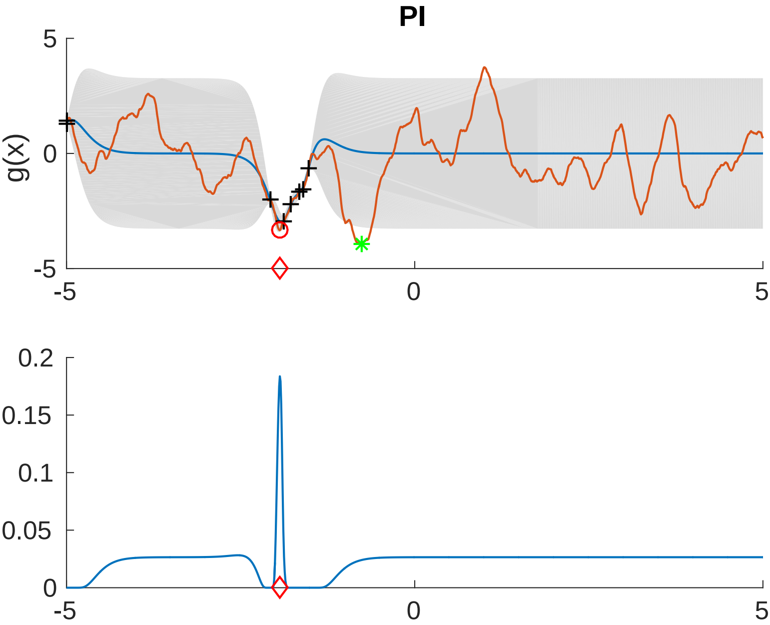


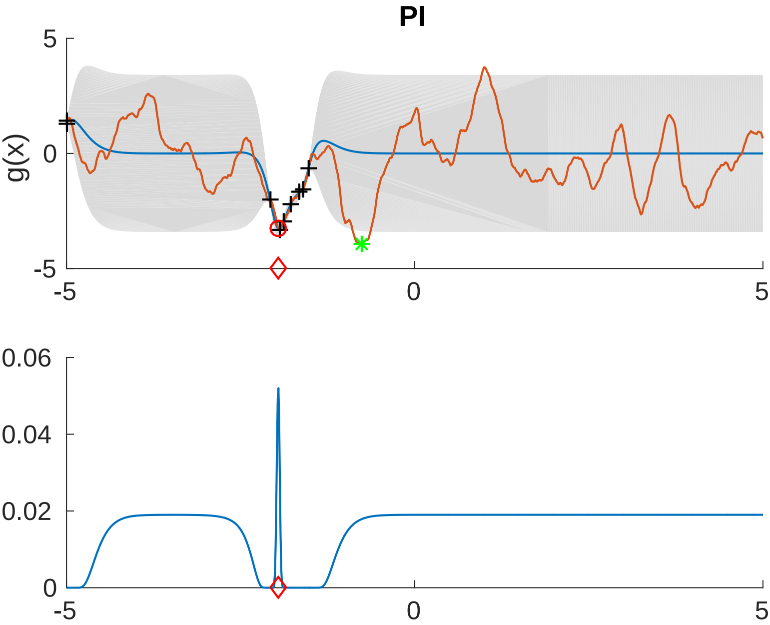
PI



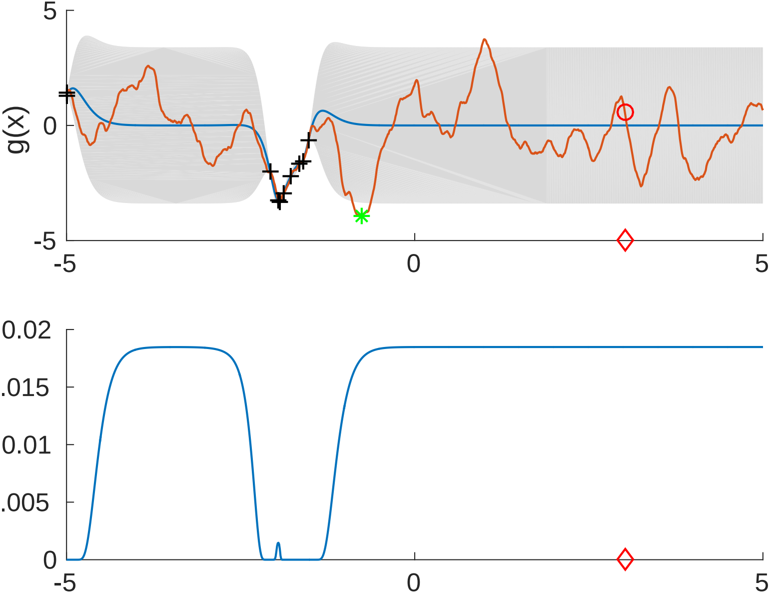



PI
g(x)
5
0
-5
-10
-5
0
5
0.6
0.4
0.2
0
-5
0
5

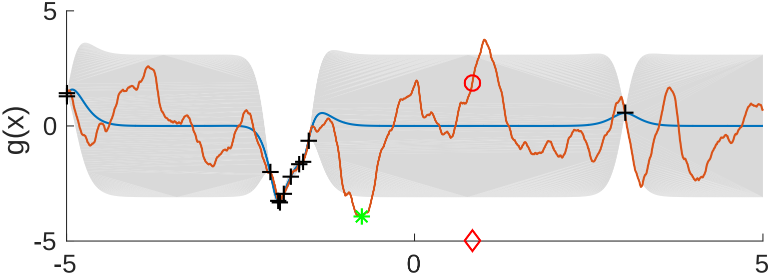

PI
g(x)
5
0
-5
-5
0
5
0.2
0.15
0.1
0.05
0
-5
0
5

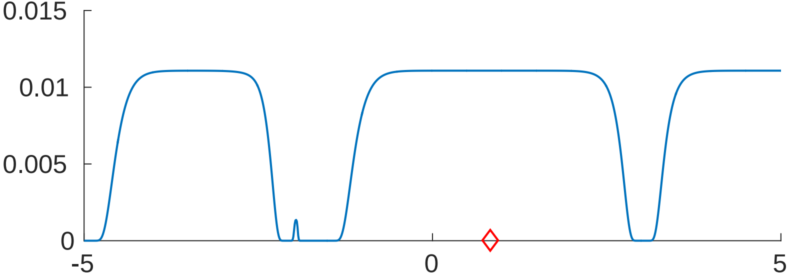

PI
g(x)
5
0
-5
-5
0
5
0.06
0.04
0.02
0
-5
0
5

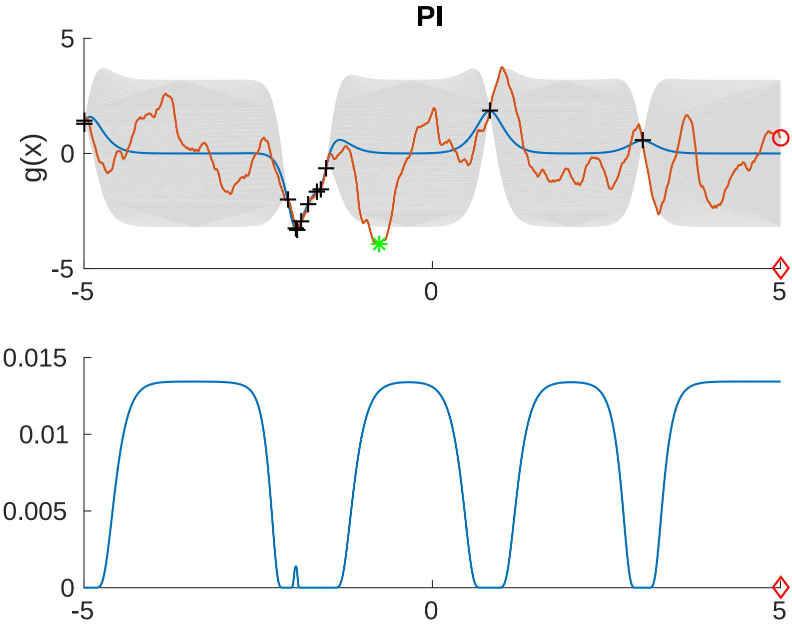
PI

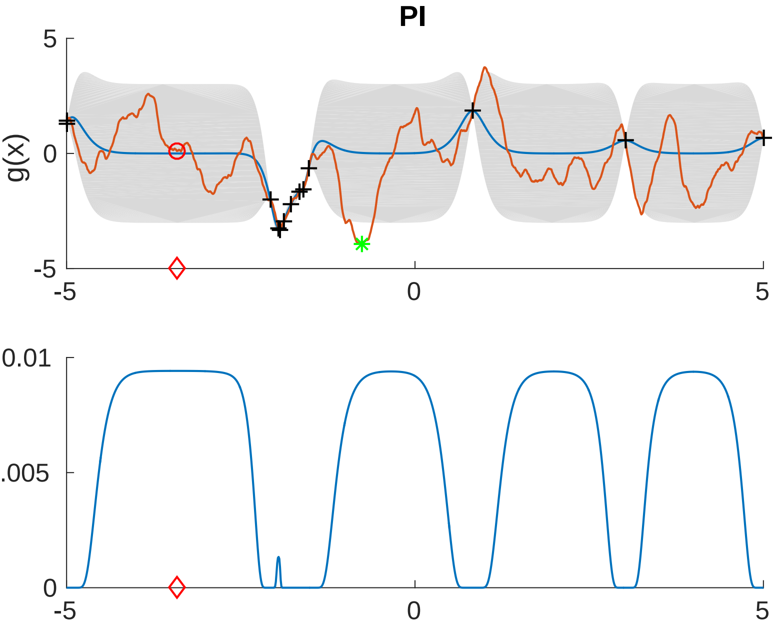


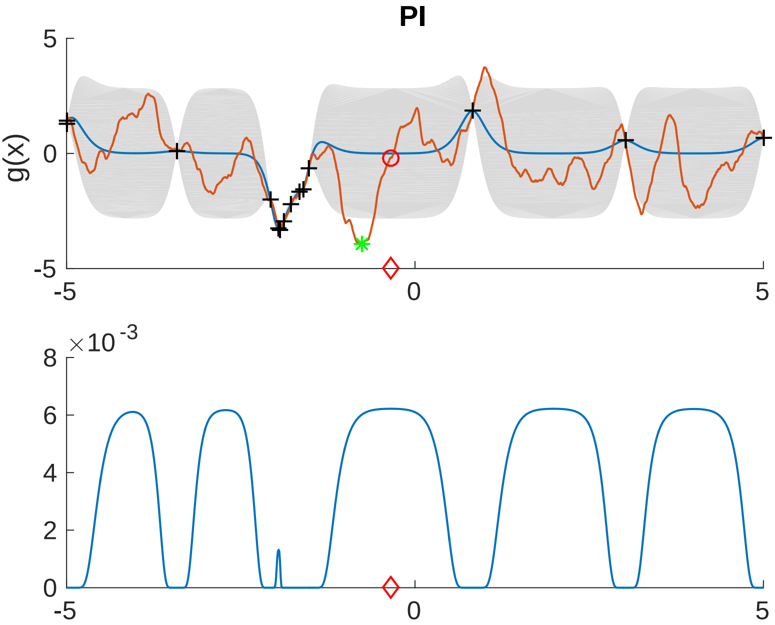
PI

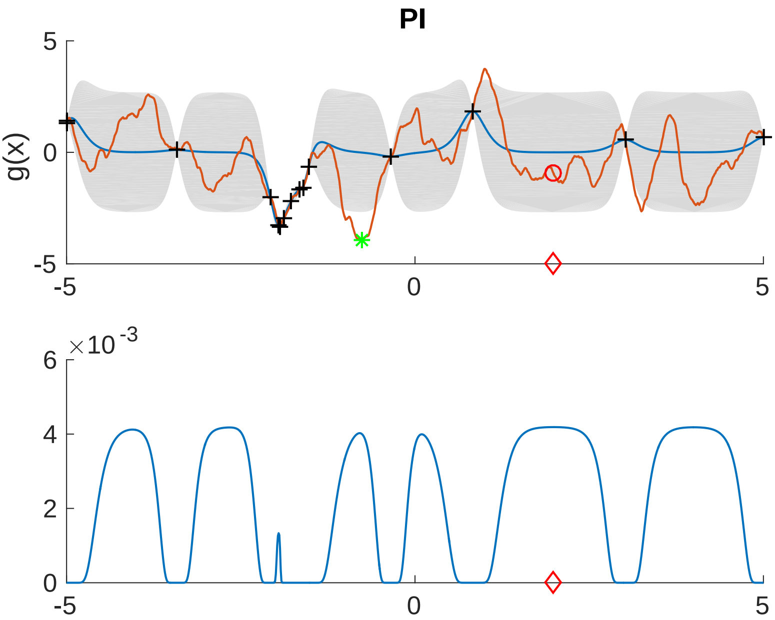


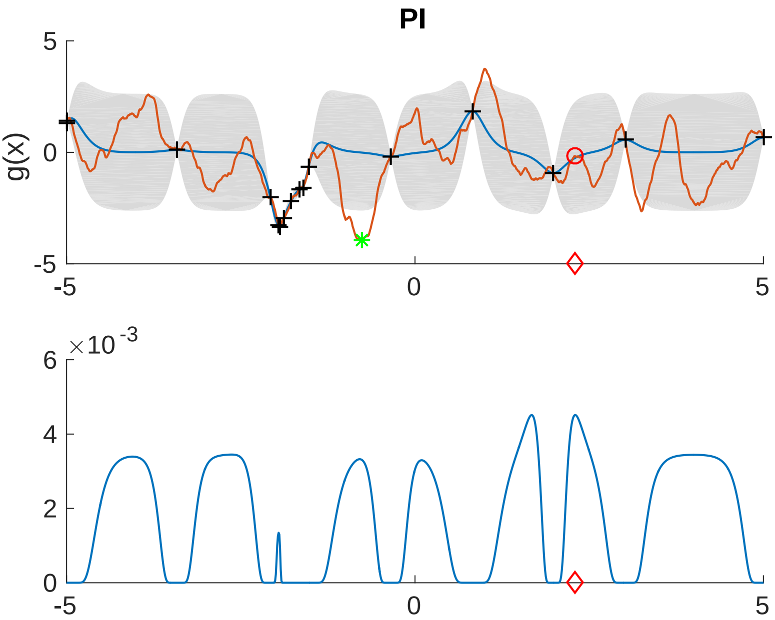


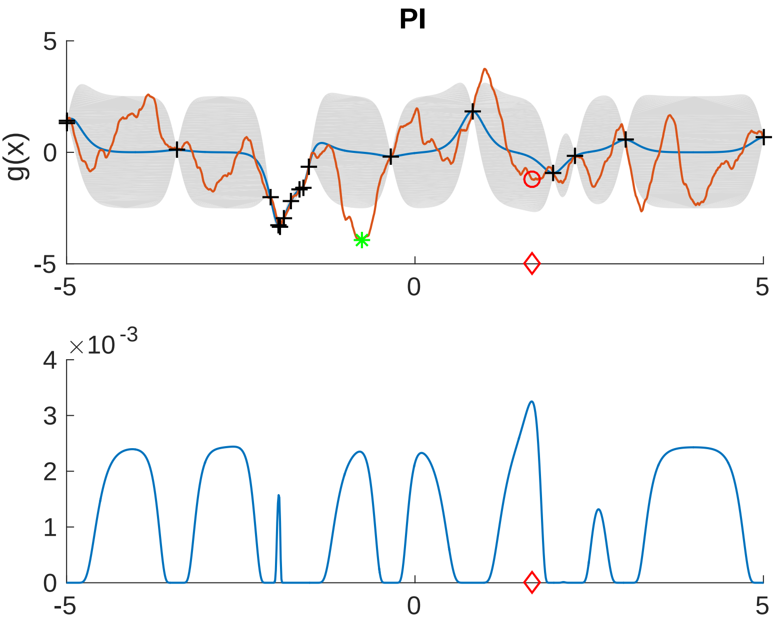

PI
g(x)
5
0
-5
-5
0
5
0.015
0.01
0.005
0
-5
0
5

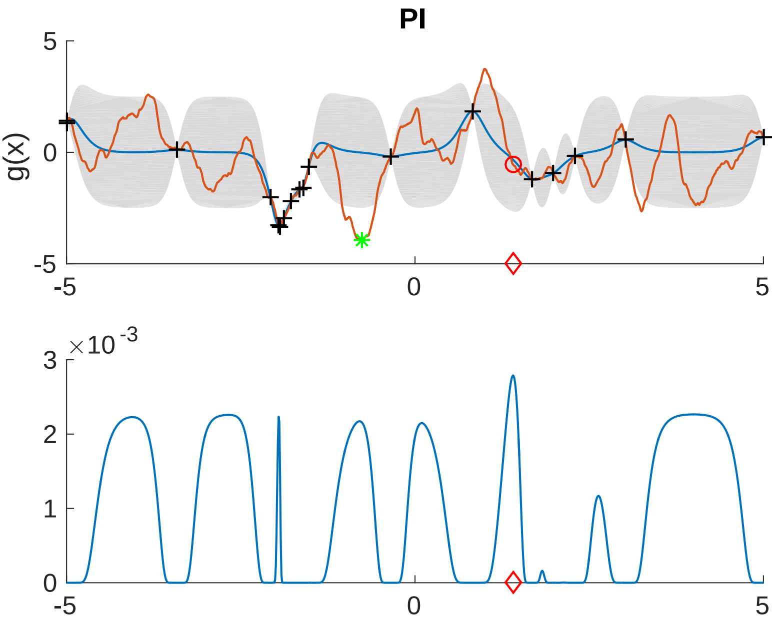

PI
g(x)
5
0
-5
-5
0
5
0.01
0.005
0
-5
0
5

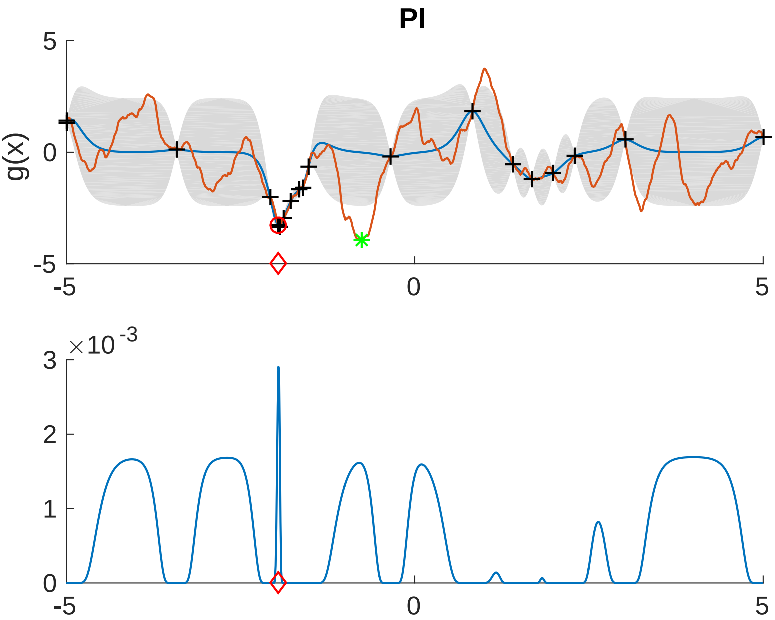

PI
5
0
-5
g(x)
-5
0
5
×10 -3
8
6
4
2
0
-5
0
5

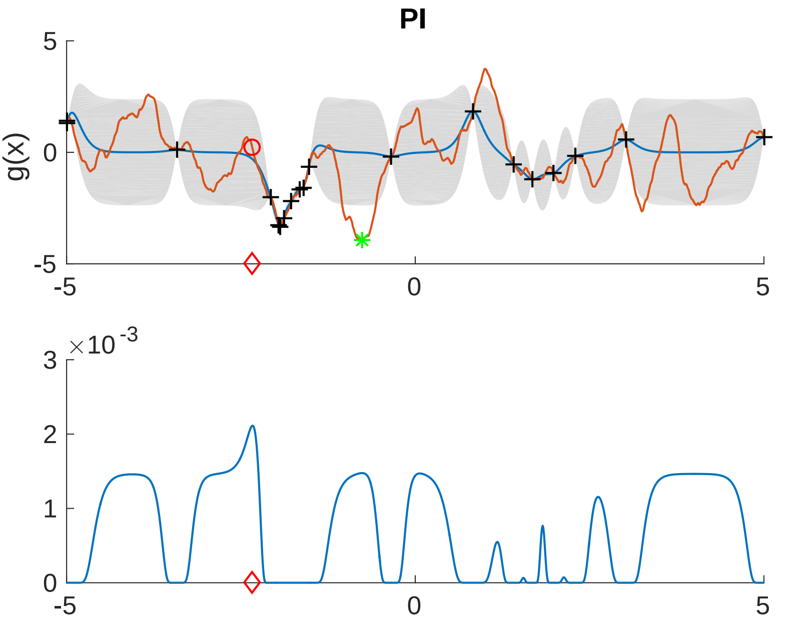

PI
g(x)
5
0
-5
-5
0
5
×10 -3
6
4
2
0
-5
0
5

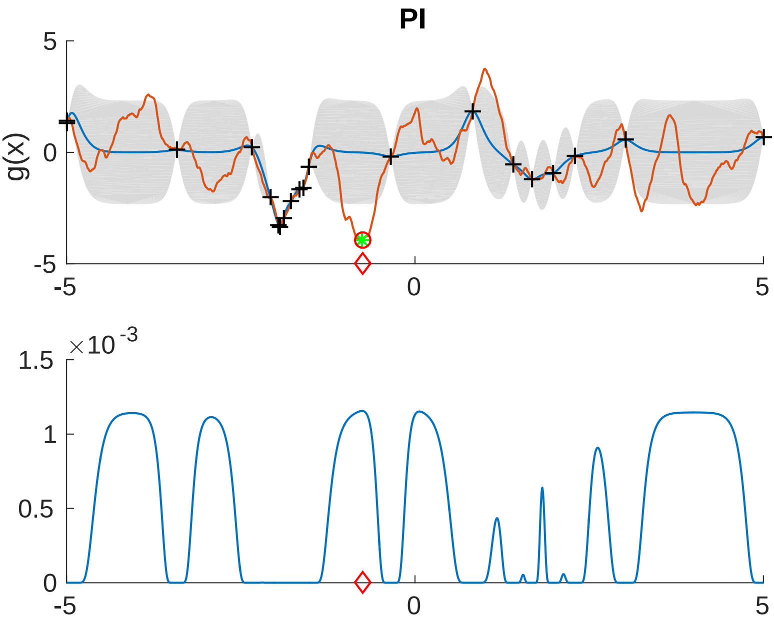

PI
g(x)
5
0
-5
-5
0
5
×10 -3
6
4
2
0
-5
0
5


PI
5
0
-5
g(x)
-5
0
5
×10 -3
4
3
2
1
0
-5
0
5

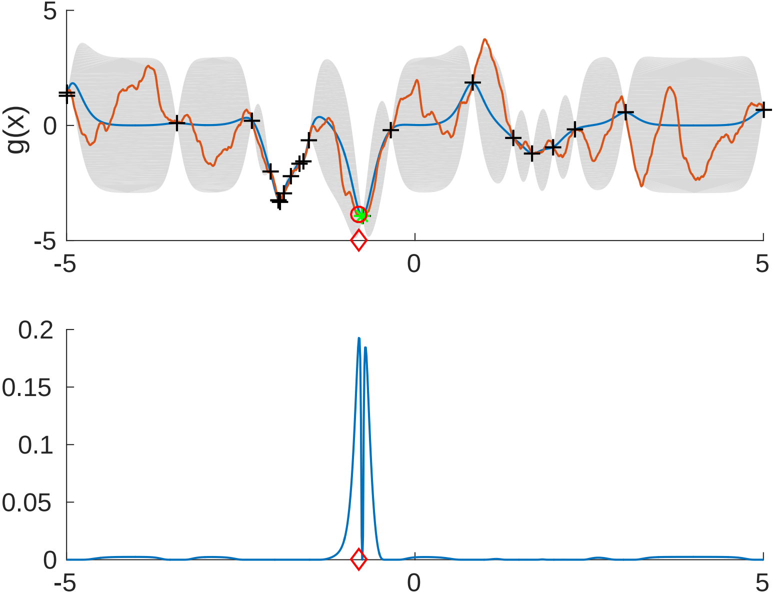

PI
5
0
-5
g(x)
-5
0
5
×10 -3
3
2
1
0
-5
0
5

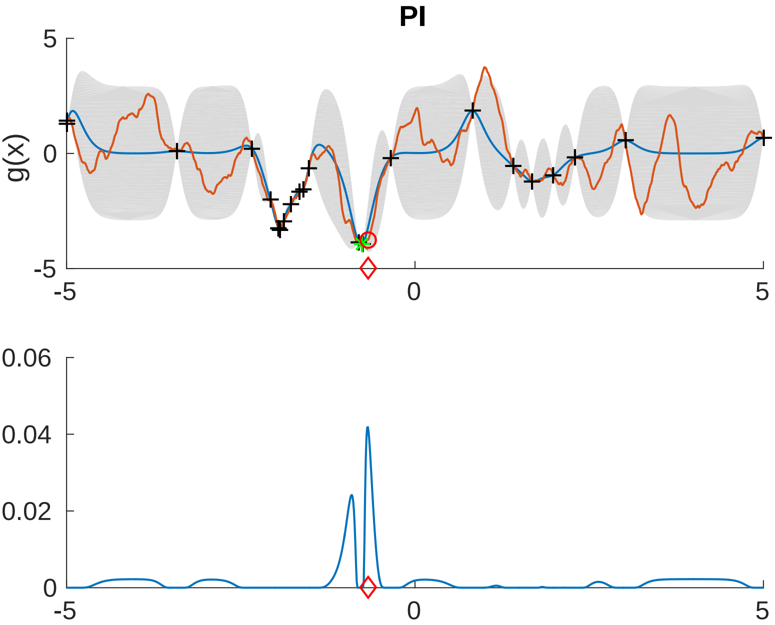

PI
g(x)
5
0
-5
-5
0
5
×10^-3
3
2
1
0
-5
0
5


PI
5
0
-5
g(x)
-5
0
5
×10 -3
3
2
1
0
-5
0
5

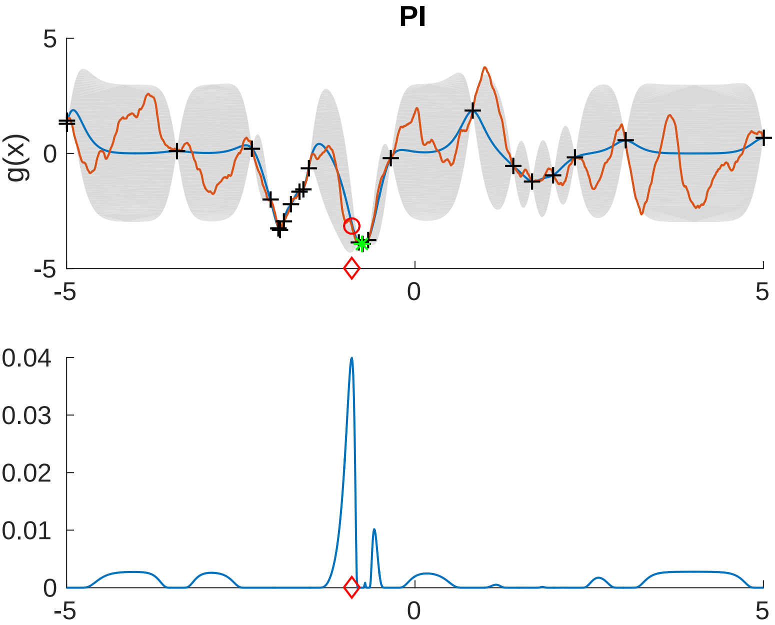

PI
g(x)
5
0
-5
-5
0
5
×10
-3
1.5
1
0.5
0
-5
0
5

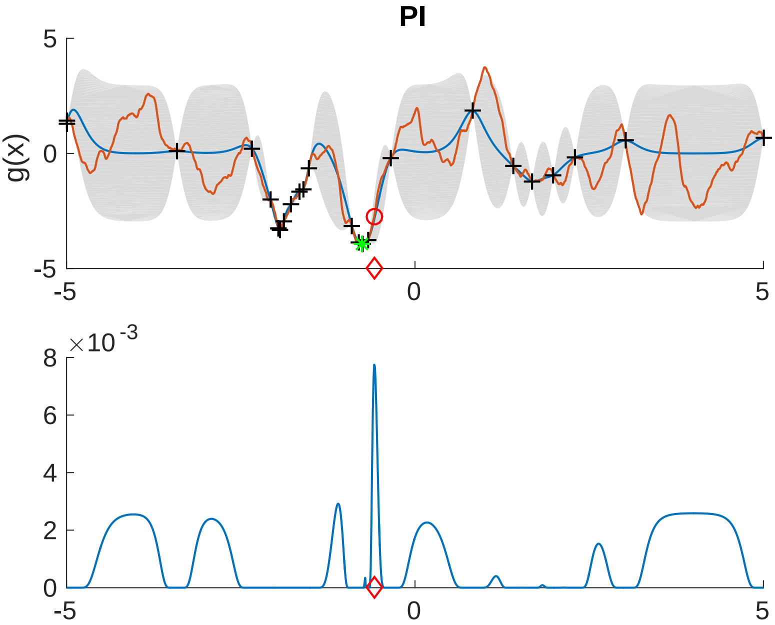
PI

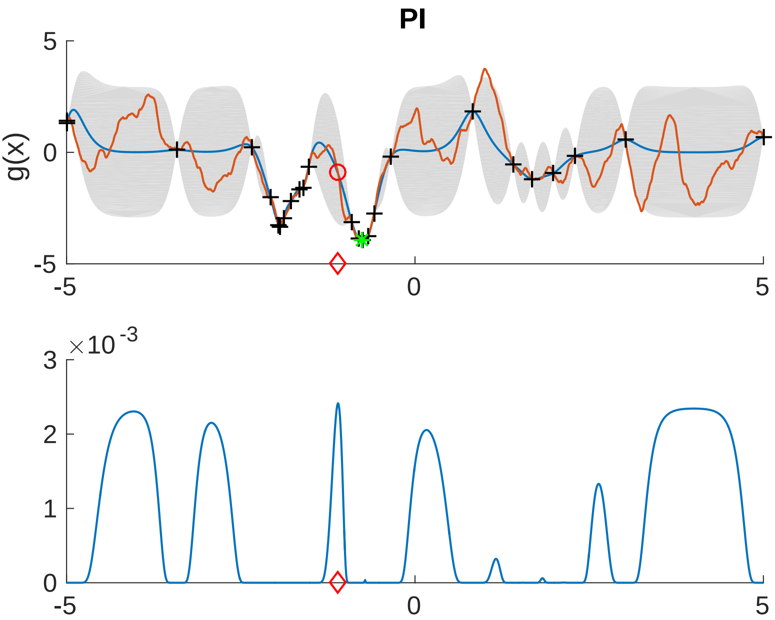


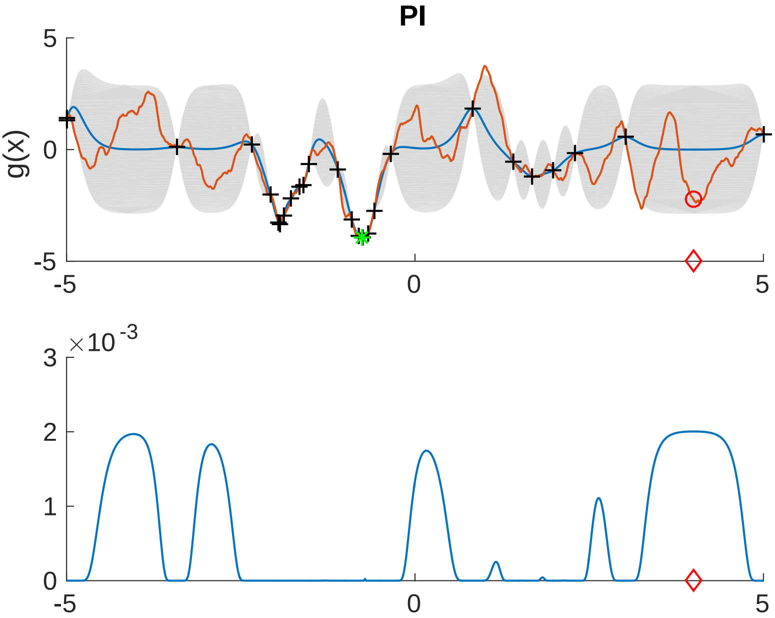


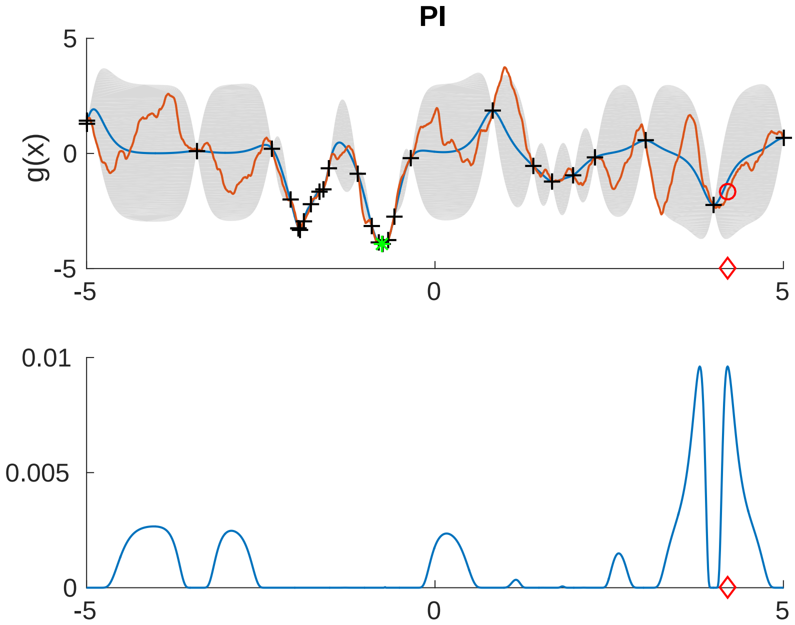
PI
5
0
-5
g(x)
-5
0
5
0.06
0.04
0.02
0
-5
0
5

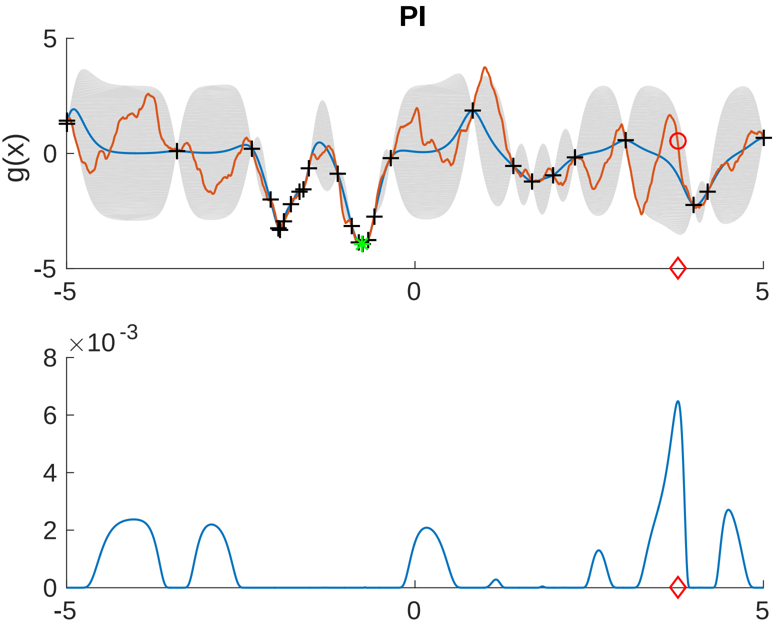

PI
5
0
-5
g(x)
-5
0
5
0.04
0.03
0.02
0.01
0
-5
0
5


PI
g(x)
5
0
-5
-5
0
5
×10 -3
8
6
4
2
0
-5
0
5

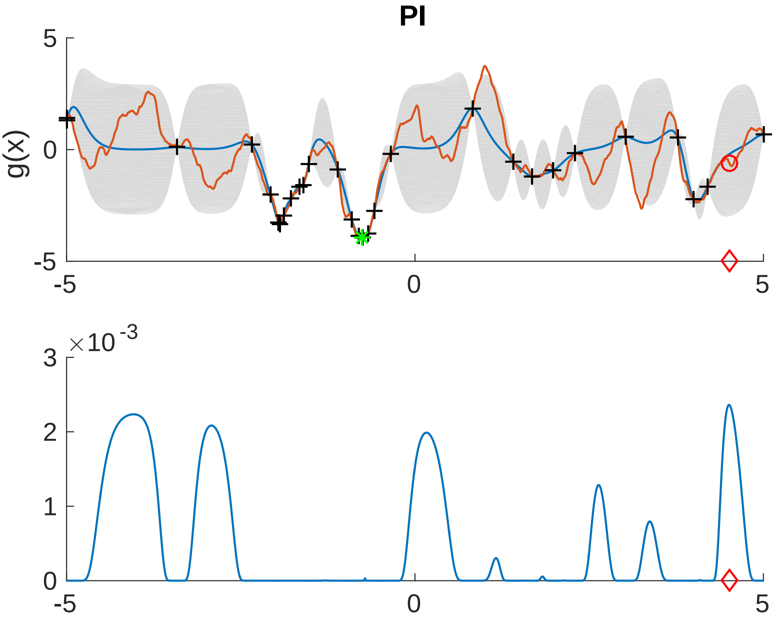

PI
g(x)
5
0
-5
-5
0
5
×10 -3
3
2
1
0
-5
0
5

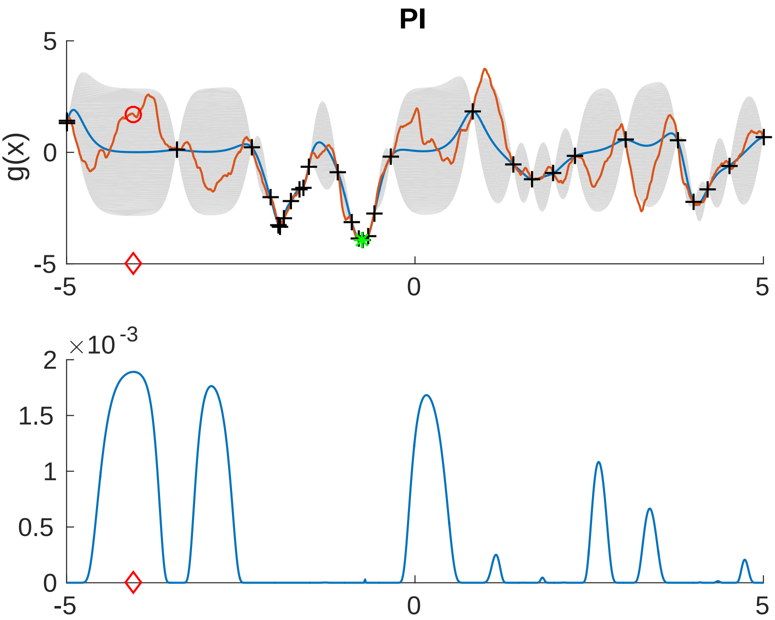

PI
g(x)
5
0
-5
-5
0
5
×10^-3
3
2
1
0
-5
0
5


PI
5
0
-5
g(x)
-5
0
5
0.01
0.005
0
-5
0
5



PI
5
0
-5
g(x)
-5
0
5
×10 -3
8
6
4
2
0
-5
0
5

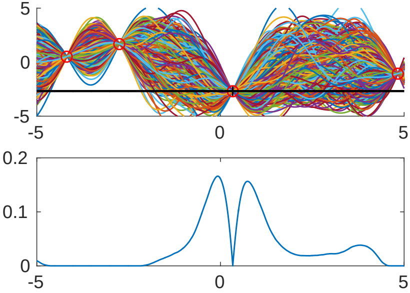

PI
g(x)
5
0
-5
-5
0
5
×10 -3
3
2
1
0
-5
0
5


PI
g(x)
5
0
-5
-5
0
5
×10 -3
2
1.5
1
0.5
0
-5
0
5

Expected Improvement

- Idea: Quantify the amount of improvement
- Sampling-based scenario, where $g_i \sim p(f)$:

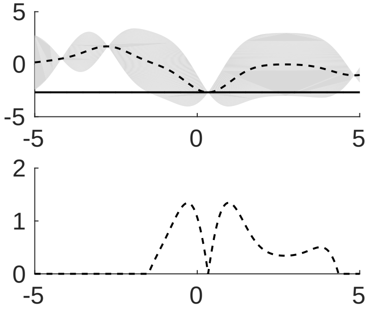
$$\alpha_{\mathrm{EI}}(\boldsymbol{x}) = \mathbb{E}[\max\{0, g(\boldsymbol{x}_{\mathrm{best}}) - g(\boldsymbol{x})\}]$$
$$\approx \frac{1}{N}\sum_{i=1}^{N}\max\{0, g(\boldsymbol{x}_{\mathrm{best}}) - g_i(\boldsymbol{x})\}$$



- If $f \sim GP$, we have a closed-form expression:

$$\alpha_{\mathrm{EI}}(\boldsymbol{x}) = \sigma(\boldsymbol{x})\big(\gamma(\boldsymbol{x})\Phi(\gamma(\boldsymbol{x})) + \mathcal{N}\big(\gamma(\boldsymbol{x})\,|\,0,\,1\big)\big)$$

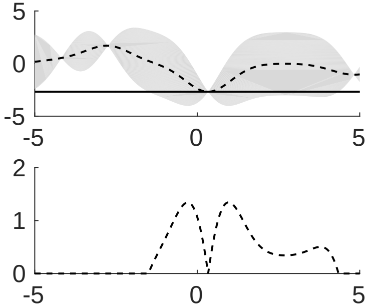
GP-Lower Confidence Bound (1)



- Use the predictive mean $\mu(\boldsymbol{x})$ and variance $\sigma^2(\boldsymbol{x})$ of the GP prediction directly for targeted exploration by means of the acquisition function

$$\alpha_{\text{LCB}}(\boldsymbol{x}_t) = -\big(\mu(\boldsymbol{x}_t) - \sqrt{\kappa}\sigma(\boldsymbol{x}_t)\big)$$

GP-Lower Confidence Bound (2)



- More generally, we can get regret bounds for iteration-dependent κ (Srinivas et al., 2010)

$$\alpha_{\text{LCB}}(\boldsymbol{x}_t) = -\left(\mu(\boldsymbol{x}_t) - \sqrt{\kappa_t}\sigma(\boldsymbol{x}_t)\right)$$

where $\kappa_t \in \mathcal{O}(\log t)$ grows with the iteration t

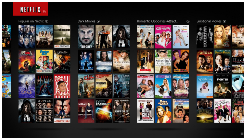
▶▶ Continue exploration

Optimizing the Acquisition Function

- Optimizing the acquisition function requires us to run a global optimizer inside Bayesian optimization
- What have we gained?

Optimizing the Acquisition Function

- Optimizing the acquisition function requires us to run a global optimizer inside Bayesian optimization
- What have we gained?
- Evaluating the acquisition function is cheap compared to evaluating the true objective
  - ⏩ We can afford evaluating it many times

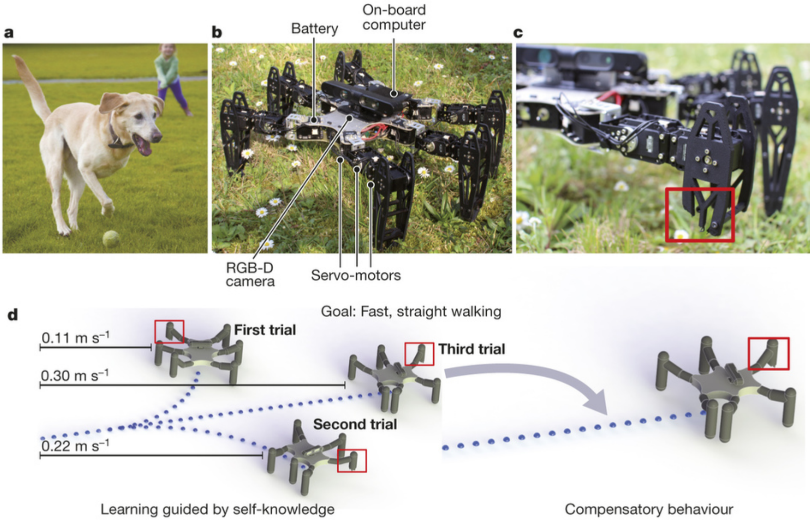



Limitations

- Getting the function model (e.g., covariance function) wrong can be catastrophic
- Limited scalability in the number of dimensions and/or evaluations of the true objective function
  Why?

Overview

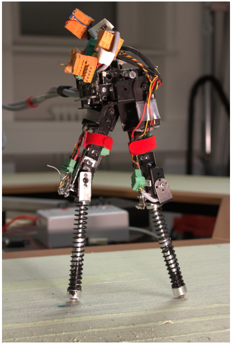
Robots That Learn to Recover from Damage



Cully et al. (2015)

Application Example: Controller Learning in Robotics (Calandra et al., 2015)

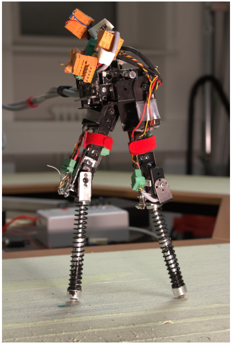
- Fragile bipedal robot
  ⏩ Only few experiments feasible
- Maximize robustness and walking speed
- 4 motors:
  2 actuated hips + 2 actuated knees
- Controller implemented as a finite-state-machine (8 parameters)



Calandra et al. (2015)

Application Example: Controller Learning in Robotics (Calandra et al., 2015)

- Fragile bipedal robot
  ⏩ Only few experiments feasible
- Maximize robustness and walking speed
- 4 motors:
  2 actuated hips + 2 actuated knees
- Controller implemented as a finite-state-machine (8 parameters)
- Good parameters found after 80–100 experiments
- Substantial speed-up compared to manual parameter search



Calandra et al. (2015)

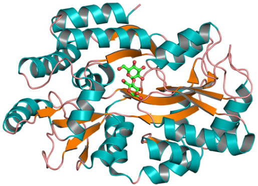
Further Topics in BO

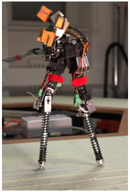
- Entropy-based acquisition functions: Directly describe the distribution over the best input location (Hennig & Schuler, 2012; Hernández-Lobato et al., 2014)
- Non-myopic BO (e.g., Osborne et al., 2009)
- High-dimensional optimization (e.g., Wang et al., 2016; Moriconi et al., 2019)
- Large-scale BO (Hutter et al., 2014)
- Efficient optimization of acquisition functions (Wilson et al., 2018)
- Non-GP BO (Hutter et al., 2014; Snoek et al., 2015)
- Constraints (e.g., Gelbart et al., 2014)
- Automated machine learning (e.g., Feurer et al., 2015)
- Multi-tasking, parallelizing, resource allocation, ... (e.g., Swersky et al., 2014; Snoek et al., 2012; Wilson et al., 2018)

Software

- BayesOpt `https://bitbucket.org/rmcantin/bayesopt/` (Martinez-Cantin, 2014)
- Spearmint `https://github.com/HIPS/Spearmint`
- Pybo `https://github.com/mwhoffman/pybo` (Hoffman & Shariari)
- GPyOpt `https://github.com/SheffieldML/GPyOpt` (Gonzalez et al.)
- botorch `https://github.com/pytorch/botorch` (Facebook)
- Matlab toolbox (bayesopt)

Summary

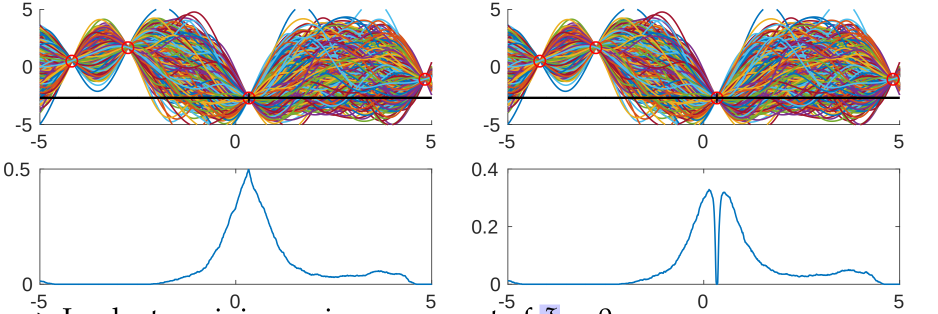




- Global optimization of black-box functions, which are expensive to evaluate ⏩ Meta-challenges in machine learning, Auto-ML
- Use a probabilistic proxy model that is cheap to evaluate and use this to suggest next experiments
- Acquisition function trades of exploration and exploitation

Appendix

Probability of Improvement (2)



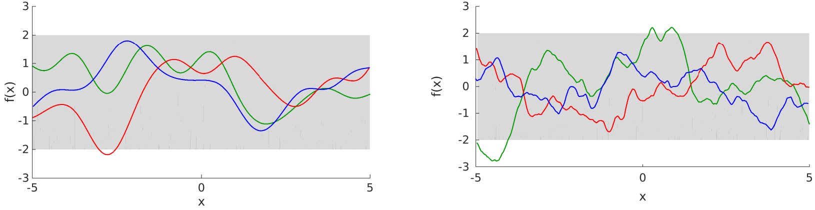
- Look at a minimum improvement of $\xi > 0$:

$$\alpha_{\text{PI}}(\boldsymbol{x}) = p(g(\boldsymbol{x}) < g(\boldsymbol{x}_{\text{best}}) - \xi) \approx \frac{1}{N} \sum_{i=1}^{N} \delta\big(g_i(\boldsymbol{x}) < g(\boldsymbol{x}_{\text{best}}) - \xi\big)$$

- If $f \sim GP$ and $p(g(\boldsymbol{x})) = \mathcal{N}\big(\mu(\boldsymbol{x}), \sigma(\boldsymbol{x})\big)$:

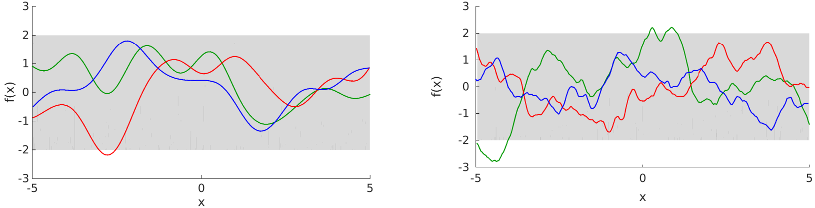
$$\alpha_{\text{PI}}(\boldsymbol{x}) = \Phi(\gamma(\boldsymbol{x}, \xi)), \qquad \gamma(\boldsymbol{x}, \xi) = \frac{g(\boldsymbol{x}_{\text{best}}) - \xi - \mu(\boldsymbol{x})}{\sigma(\boldsymbol{x})}$$

Poor Model Choice



- Covariance function selection is crucial for good performance
  ▶▶ Choose a sufficiently flexible and adaptive kernel, e.g., Matérn (but not the squared exponential (Gaussian))

Poor Model Choice



- Covariance function selection is crucial for good performance
  ▶▶ Choose a sufficiently flexible and adaptive kernel, e.g., Matérn (but not the squared exponential (Gaussian))
- Nice side-effect of Matérn: Exploration is more encouraged than with the Gaussian kernel

Choosing Covariance Functions

- Structured SVM for Protein Motif Finding (Miller et al., 2012)
- Optimize hyper-parameters of SSVM using BO (Snoek et al., 2012)

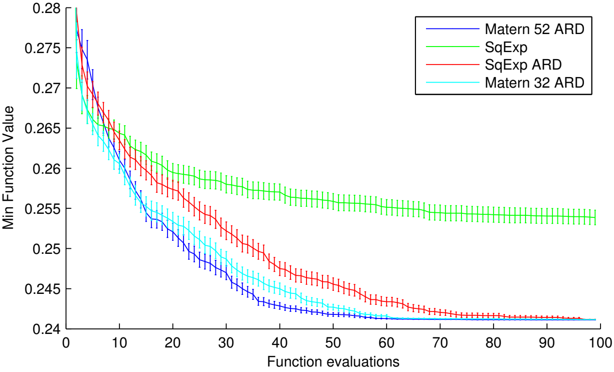


Figure: Figure from Snoek et al. (2012)

Gaussian Process Hyper-Parameters

- Empirical Bayes (maximize the marginal likelihood) can fail horribly, especially in the early stages of Bayesian optimization when we have only a few data points

Gaussian Process Hyper-Parameters

- Empirical Bayes (maximize the marginal likelihood) can fail horribly, especially in the early stages of Bayesian optimization when we have only a few data points
- Solution: Integrate out the GP hyper-parameters $\boldsymbol{\theta}$ by Markov Chain Monte Carlo (MCMC) sampling (e.g., slice sampling)

Gaussian Process Hyper-Parameters

- Empirical Bayes (maximize the marginal likelihood) can fail horribly, especially in the early stages of Bayesian optimization when we have only a few data points
- Solution: Integrate out the GP hyper-parameters $\boldsymbol{\theta}$ by Markov Chain Monte Carlo (MCMC) sampling (e.g., slice sampling)
- Look at integrated acquisition function

$$\begin{aligned}\alpha(\boldsymbol{x}) &= \mathbb{E}_{\boldsymbol{\theta}}[\alpha(\boldsymbol{x},\boldsymbol{\theta})] = \int \alpha(\boldsymbol{x},\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \\ &\approx \frac{1}{K}\sum_{k=1}^{K}\alpha(\boldsymbol{x},\boldsymbol{\theta}^{(k)}), \quad \boldsymbol{\theta}^{(k)} \sim \underbrace{p(\boldsymbol{\theta}|\boldsymbol{X}_n,\boldsymbol{y}_n)}_{\text{hyper-parameter posterior}}\end{aligned}$$

Integrating out GP Hyper-parameters

- Online LDA (Hoffman et al., 2010) for topic modeling
- Two critical hyper-parameters that control the learning rate learned by BO (Snoek et al., 2012)

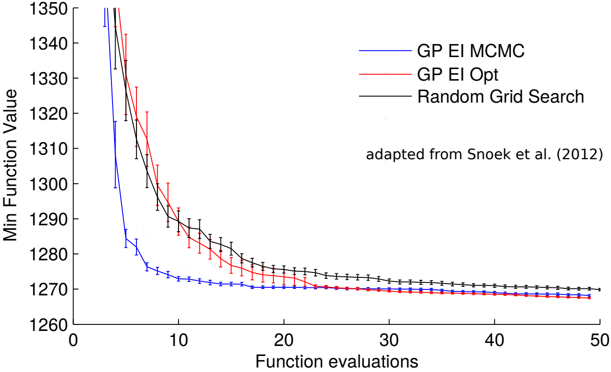


Figure: Figure from Snoek et al. (2012)

References I

[1] E. Brochu, V. M. Cora, and N. de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. Technical Report TR-2009-023, Department of Computer Science, University of British Columbia, 2009.

[2] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth. Bayesian Optimization for Learning Gaits under Uncertainty. *Annals in Mathematics and Artificial Intelligence*, pages 1–19, 2015.

[3] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas. Bayesian Optimization in AlphaGo. *arXiv:1812.06855*, 2018.

[4] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots That Can Adapt Like Animals. *Nature*, 521:503–507, 2015.

[5] M. P. Deisenroth, A. A. Faisal, and C. S. Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.

[6] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and Robust Automated Machine Learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2962–2970. Curran Associates, Inc., 2015.

[7] M. Gelbart, J. Snoek, and R. P. Adams. Bayesian Optimization with Unknown Constraints. In *International Conference on Uncertainty in Artificial Intelligence*, pages 1–14, 2014.

[8] P. Hennig and C. J. Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[9] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *Advances in Neural Information Processing Systems*, pages 1–9, 2014.

[10] M. D. Hoffman, D. M. Blei, and F. Bach. Online Learning for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, 23:1–9, 2010.

[11] F. Hutter, H. Hoos, and K. Leyton-Brown. An Efficient Approach for Assessing Hyperparameter Importance. In *Proceedings of International Conference on Machine Learning*, pages 754–762, June 2014.

[12] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, Dec. 1998.

References II

[13] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86:97, 1964.

[14] D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Edmonton, Alberta, 2008.

[15] R. Martinez-Cantin. BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design and Bandits. *Journal of Machine Learning Research*, 15:3915–3919, 2014.

[16] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty. In *Proceedings of Robotics: Science and Systems III*, Atlanta, GA, USA, June 2007.

[17] K. Miller, M. P. Kumar, B. Packer, D. Goodman, and D. Koller. Max-Margin Min-Entropy Models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 22, pages 779–787, 2012.

[18] J. Mockus, V. Tiesis, and A. Zilinska. The Application of Bayesian Methods for Seeking the Extremum. *Towards Global Optimization*, 2:117–129, 1978.

[19] R. Moriconi, K. S. S. Kumar, and M. P. Deisenroth. High-Dimensional Bayesian Optimization with Projections Using Quantile Gaussian Processes. *Optimization Letters*, 2019.

[20] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimization. In *Proceedings of the International Conference on Learning and Intelligent Optimization*, 2009.

[21] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the Human out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[22] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems*, 2012.

[23] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams. Scalable Bayesian Optimization Using Deep Neural Networks. In *Proceedings of the International Conference on Machine Learning*, 2015.

[24] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian Optimization with Robust Bayesian Neural Networks. In *Advances in Neural Information Processing Systems 29*, December 2016.

References III

[25] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the International Conference on Machine Learning*, 2010.

[26] K. Swersky, J. Snoek, and R. P. Adams. Freeze-Thaw Bayesian Optimization. Technical report, 2014.

[27] D. Ulmasov, C. Baroukh, B. Chachuat, M. P. Deisenroth, and R. Misener. Bayesian Optimization with Dimension Scheduling: Application to Biological Systems. In *Proceedings of the European Symposium on Computer Aided Process Engineering*, 2016.

[28] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. In *Journal of Artificial Intelligence Research*, volume 55, pages 361–367, 2016.

[29] J. T. Wilson, F. Hutter, and M. P. Deisenroth. Maximizing Acquisition Functions for Bayesian Optimization. In *arXiv:18-05.10196*, 2018.