QBUS6850 Week 12

Bayesian Hyperparameter Optimization with GPR

Dr Stephen Tierney

The University of Sydney Business School

Reading

Gaussian Process Regression (choose one)

- Gaussian Processes for Machine Learning - Chapter 2: Regression (book)
- Cornell CS4780 - Lecture 15: Gaussian Processes (lecture and notes)
- Machine Learning Summer School 2012 - Gaussian Processes for Machine Learning (lecture)

Bayesian Hyperparameter Optimization

- Python Tutorial (blog)

Hyper-Parameter Tuning

Hyper-parameter tuning and is usually achieved with a **Grid Search** in combination with Hold-out or CV.

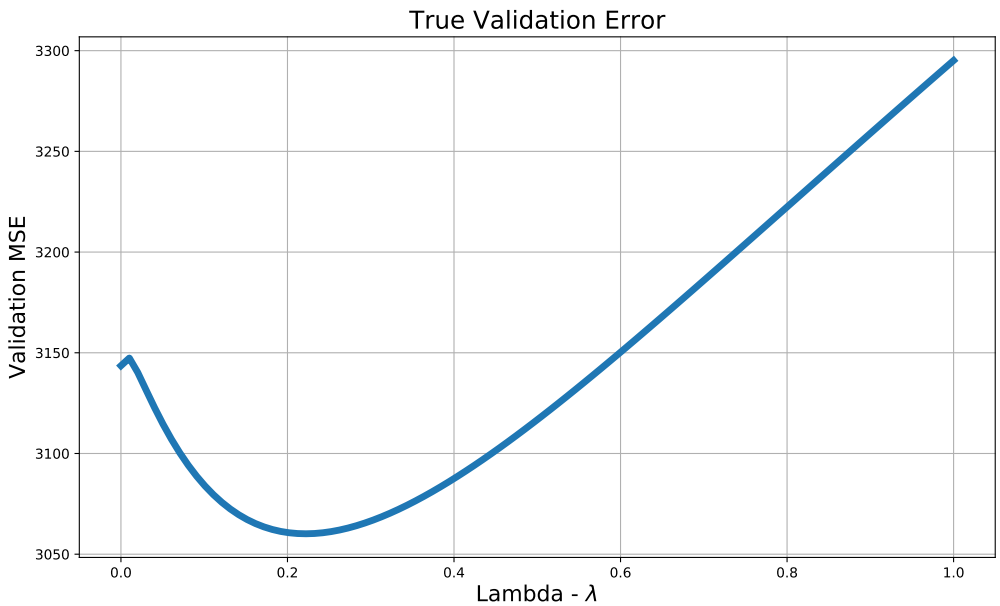
For example a LASSO model has parameter λ to control sparsity.

λ	β	Validation Loss
0.001	Est set 1	
0.01	Est set 2	
0.02	Est set 3	
0.04	Est set 4	
0.08	Est set 5	Smallest
...		
100	Est set 10000	

Test a large number of different λ value, e.g., 10000 values between 0.0001 and 100, denoted by λ_j, $j = 1, 2, 3, \ldots, 10000$.

Grid Search

Plotting these values reveals the minimum point and thus optimal hyper-parameter.



Grid Search - Computational Issues and Shortcuts

Grid Search can be burdensome, particularly if you have a large number of hyper parameters, no prior knowledge about them, huge volumes of data and many models to compare.

Simple tricks you can use:

- Breadth first search
- Randomised search
- Parallelisation

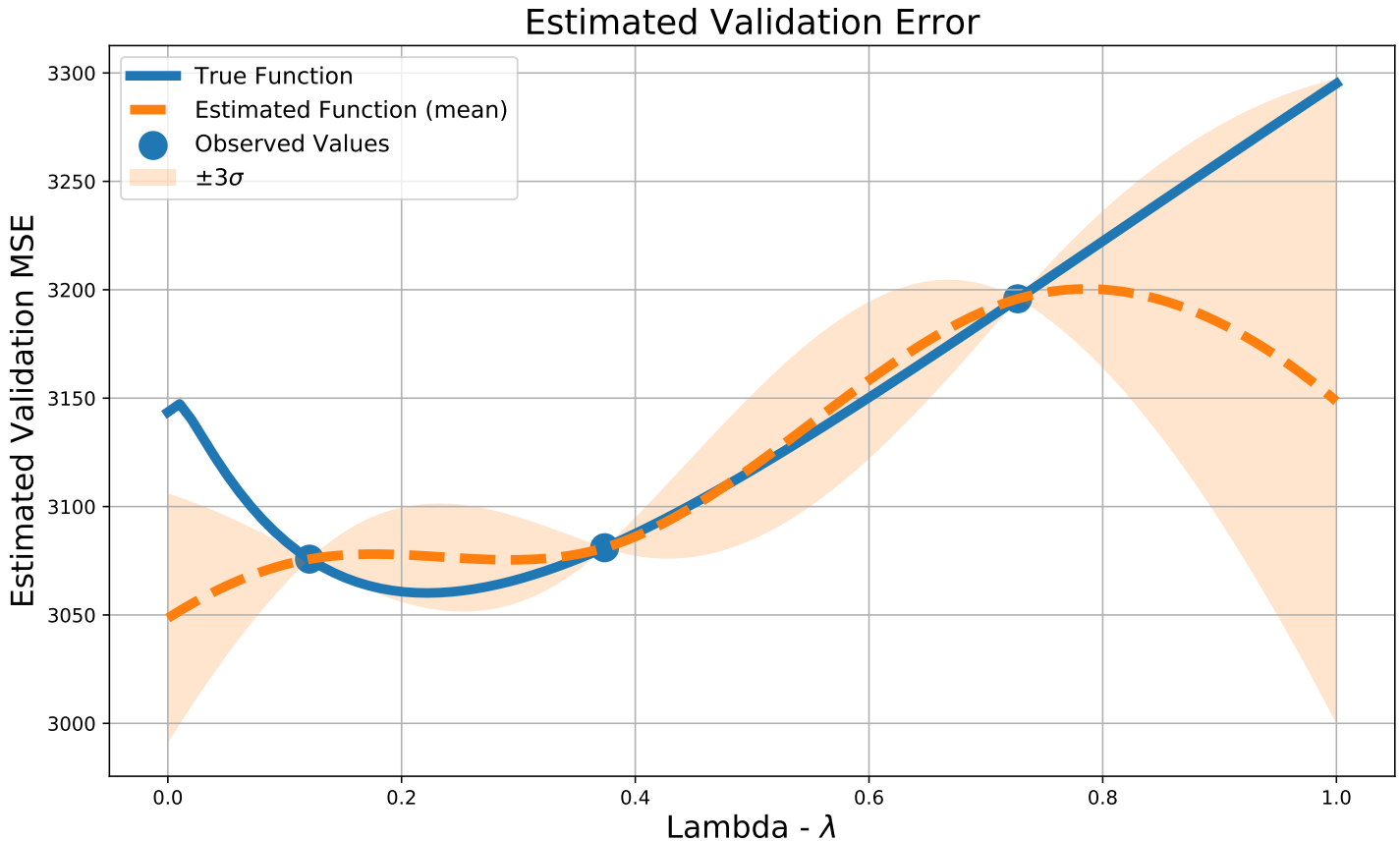
Can we do better? **YES!**

Bayesian Hyperparameter Optimization (BHO)

How? **Bayesian Hyperparameter Optimization!**

If we can create a model of the hyper-parameter function that **interpolates** (fills the gaps) between a few observations, then we do not need to evaluate the entire space.

Bayesian Hyperparameter Optimization (BHO)



Bayesian Hyperparameter Optimization (BHO)

There are two key pieces to be successful:

- a probabilistic regression model
- a policy for selecting sample points

We need a probabilistic model since we should sample points we are uncertain about and the strategy should be smart enough to minimise the number of sampled points.

Reinforcement Learning

Bayesian Hyperparameter Optimisation casts the problem as one of "reinforcement learning".

Reinforcement learning is the situation in which an agent has to explore an environment to learn. Each action the agent takes generates some reward. The goal is to maximise the cumulative reward.

Reinforcement Learning

In BHO the reward is the the negative validation error.

The "agent" is just an algorithm or policy that selects a hyperparameter value, observes the validation error and then adjusts its model of the environment.

Thompson Sampling

Thompson Sampling

Thompson Sampling is a policy used for many reinforcement learning problems.

In each round of play (iteration) a random sample is drawn from the reward distribution of each action. We pick the action that has the highest reward.

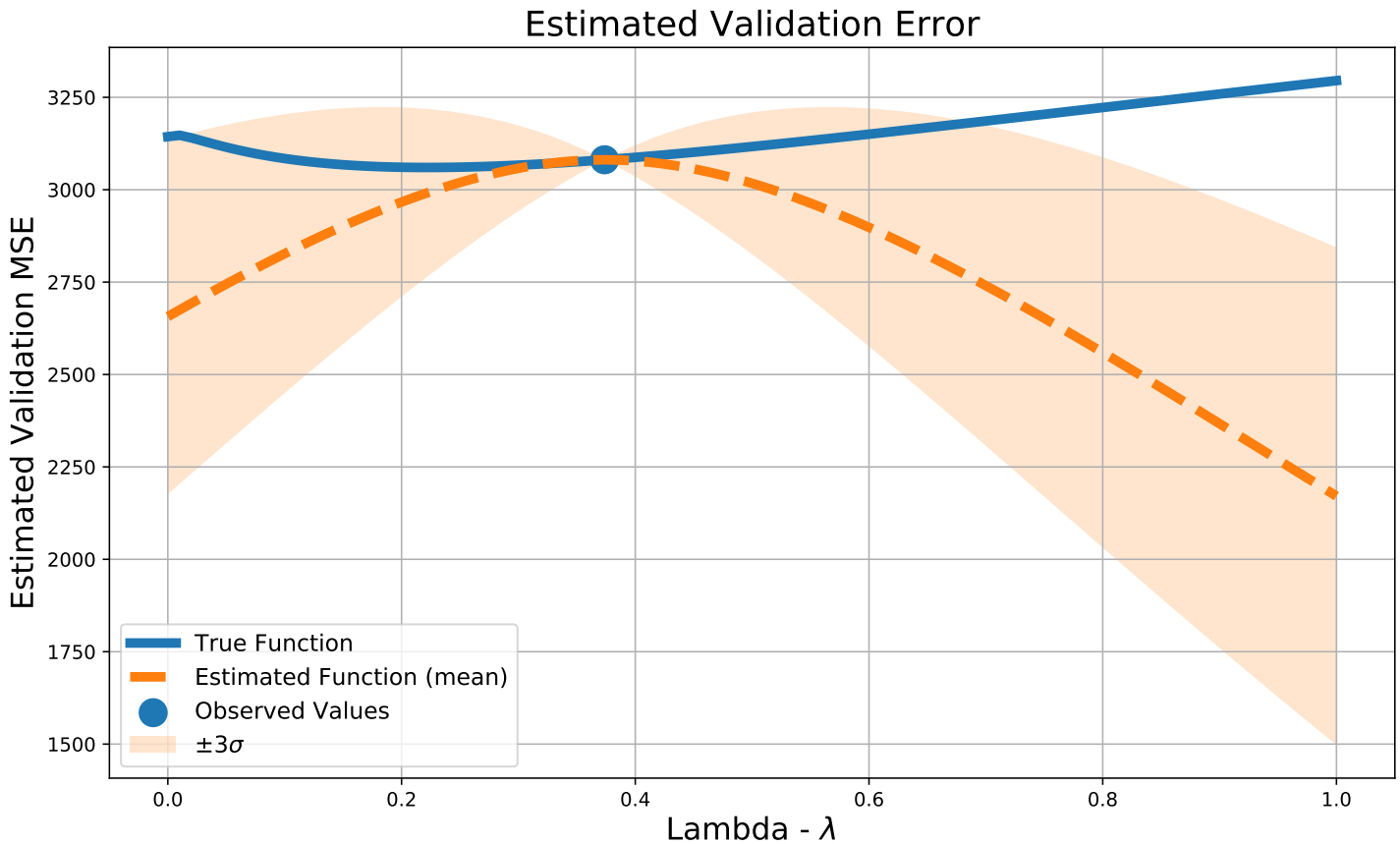
Thompson Sampling for BHO

Thompson Sampling:

1. generate sample from model
2. greedily select the outcome that maximises the reward
3. update the model
4. repeat

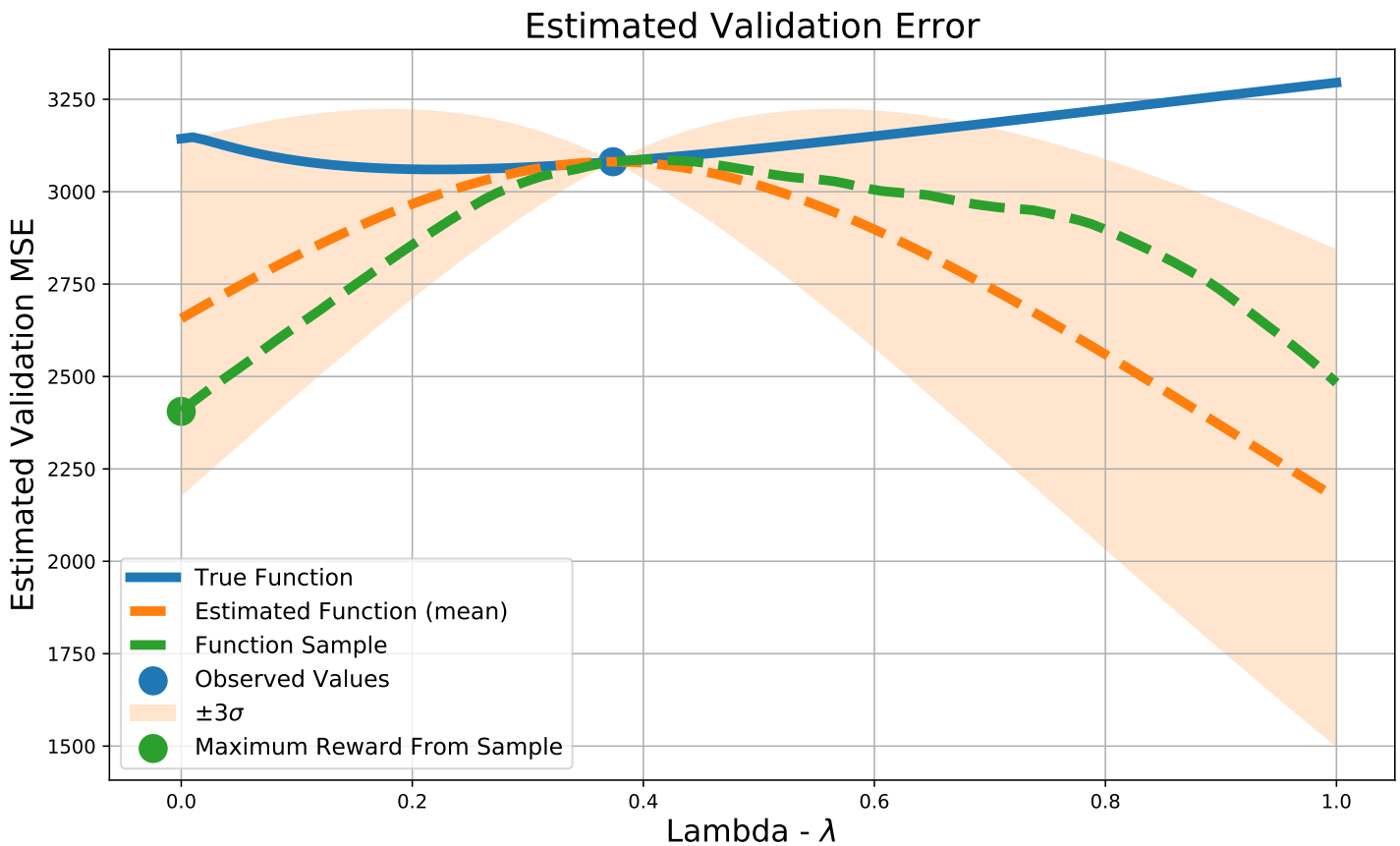
Thompson Sampling for BHO

Initialise - pick a random point in the space and update the model.



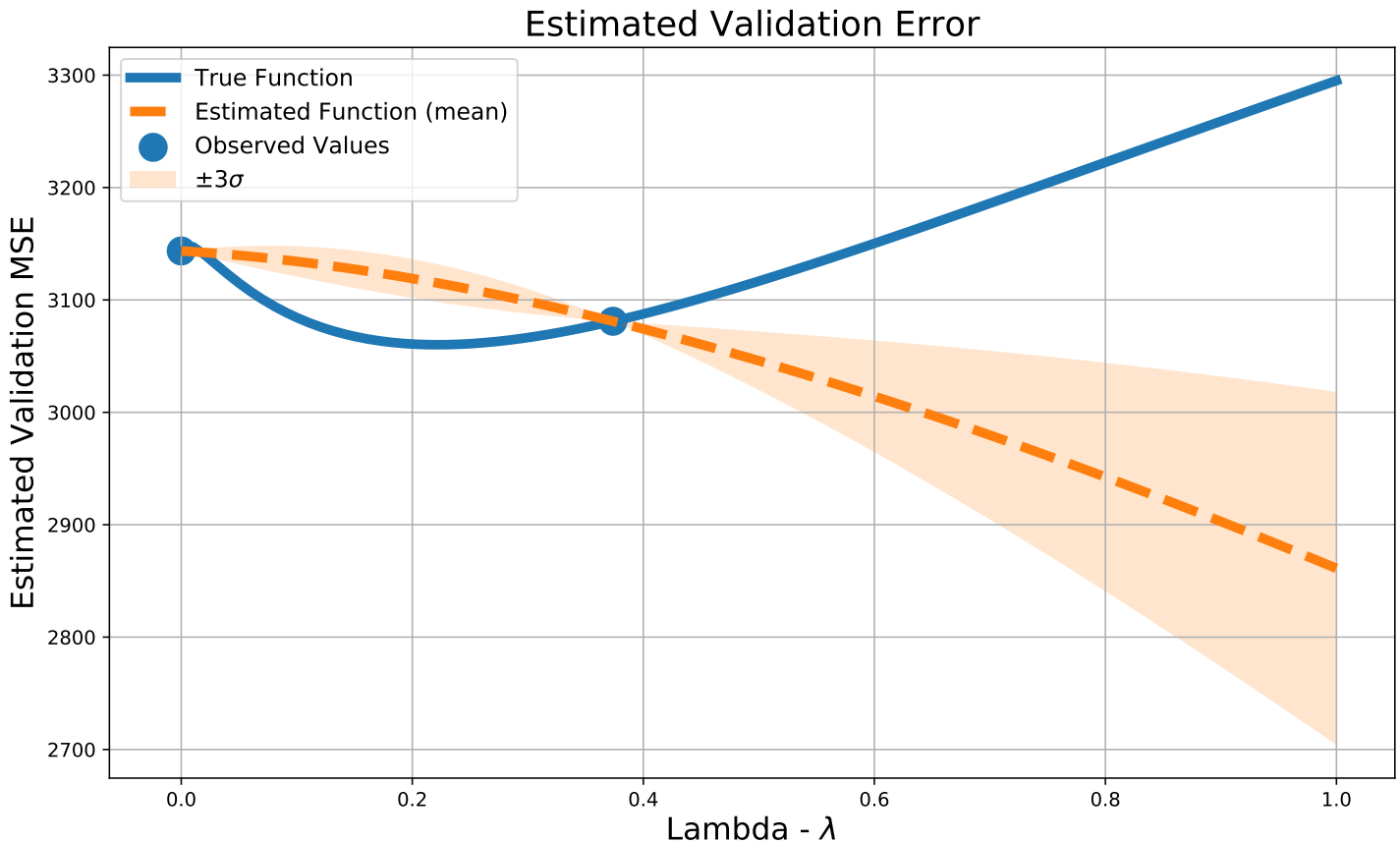
Thompson Sampling for BHO

Generate a sample and find maximum reward point



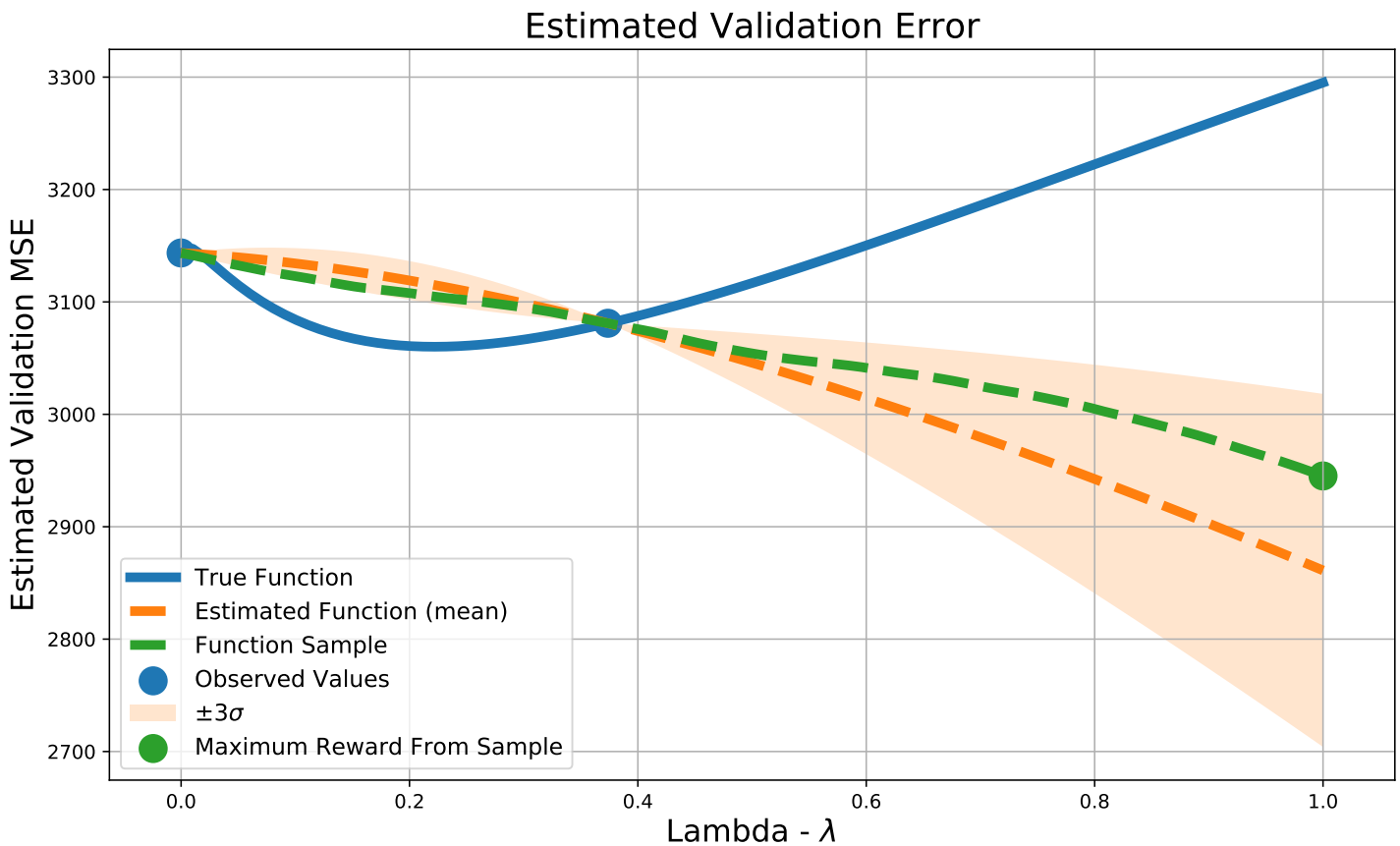
Thompson Sampling for BHO

Use the maximum reward point to update the model



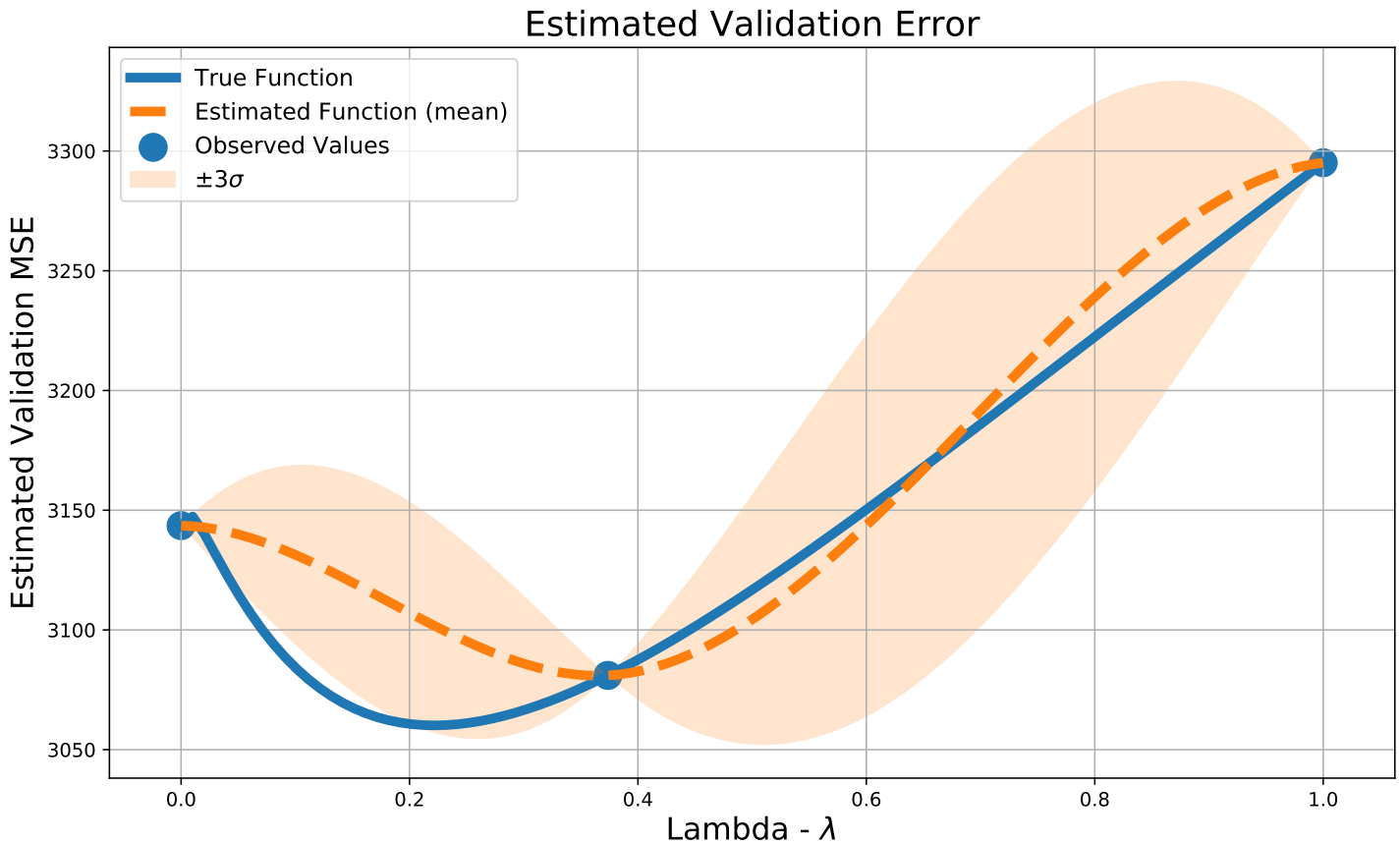
Thompson Sampling for BHO

Generate a sample and find maximum reward point



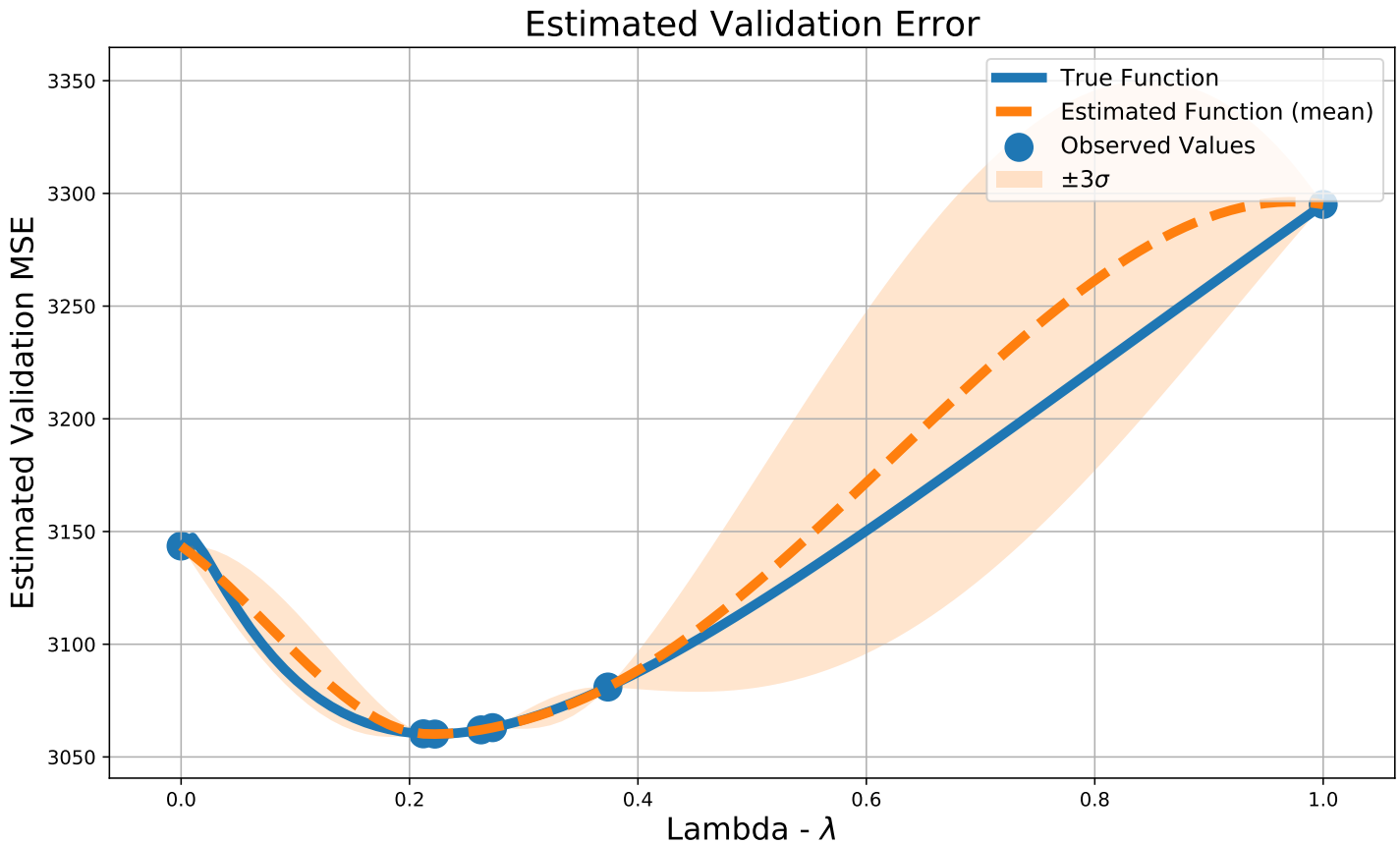
Thompson Sampling for BHO

Use the maximum reward point to update the model



Thompson Sampling for BHO

Repeating we will reach a better approximation of the function.



Stopping Criteria

We can stop the process when:

- the distance between sampled points is less than a threshold
- a specific number of iterations has been reached

Why does Thompson Sampling Work?

How does this find the minimum point of our hyper-parameter function?

When we draw a sample there are two possibilities:

1. the model is very accurate and the minimum of the sample is close to the true minimum
2. the model is inaccurate and the minimum occurs by low chance in an area we are uncertain about

In case 1, we continue to refine our estimate and after case 2 our uncertainty about that area will dramatically decrease.

Gaussian Process Regression

Modelling the Hyper-Parameter Function

A **Gaussian Process** is typically used to model the hyper-parameter function.

A GP is a random process, which creates a sequence of values i.e. a **function**.

Gaussian Processes

Properties of a GP:

- a sample from a GP is an entire sequence/function
- each value in the sequence is Gaussian distributed i.e. each value is a sample from a random variable
- subsets of values in the sequence are jointly Gaussian
- values within a GP are linked through their covariance

Gaussian Processes

Suppose I walk along the ridge line of some mountains. Each step I take is somewhat random and changes my altitude by some amount because the terrain is rocky.

If I regularly walk this path I could calculate an average altitude based on all my walks and the variance at each point.

Each walk is a sample from the Gaussian Process of walking the ridge line.

Gaussian Process - Specification

A GP is fully parameterised by a mean vector, $\boldsymbol{\mu}$, and covariance matrix Σ, i.e. $f(x) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where

- $\boldsymbol{\mu}$ specifies the central location of the function at each position x_i
- the diagonal elements of Σ specify the variance at each value x_i
- the off-diagonal elements of Σ specify the covariance between points x_i and x_j for all i, j

For example positive covariance between x_i and x_j means that f_i and f_j will be similar, while negative covariance between x_i and x_j means that f_i and f_j will be dissimilar.

A covariance of 0 means that there's no relationship between x_i and x_j, they are independent.

Gaussian Process - Specification

For GPs the covariance matrix is empirically estimated via a kernel function e.g. $\Sigma_{ij} = k(\mathrm{x}_i, \mathrm{x}_j)$.

A typical example is the squared exponential (AKA RBF),

$$K(\mathrm{x}_i, \mathrm{x}_j) = e^{-\frac{1}{2}\|\mathrm{x}_i - \mathrm{x}_j\|^2},$$

although we can choose any kernel that we like.

We can then say

$$f(x) \sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{K})$$

where K is the matrix of our kernel distances.

Gaussian Process Regression

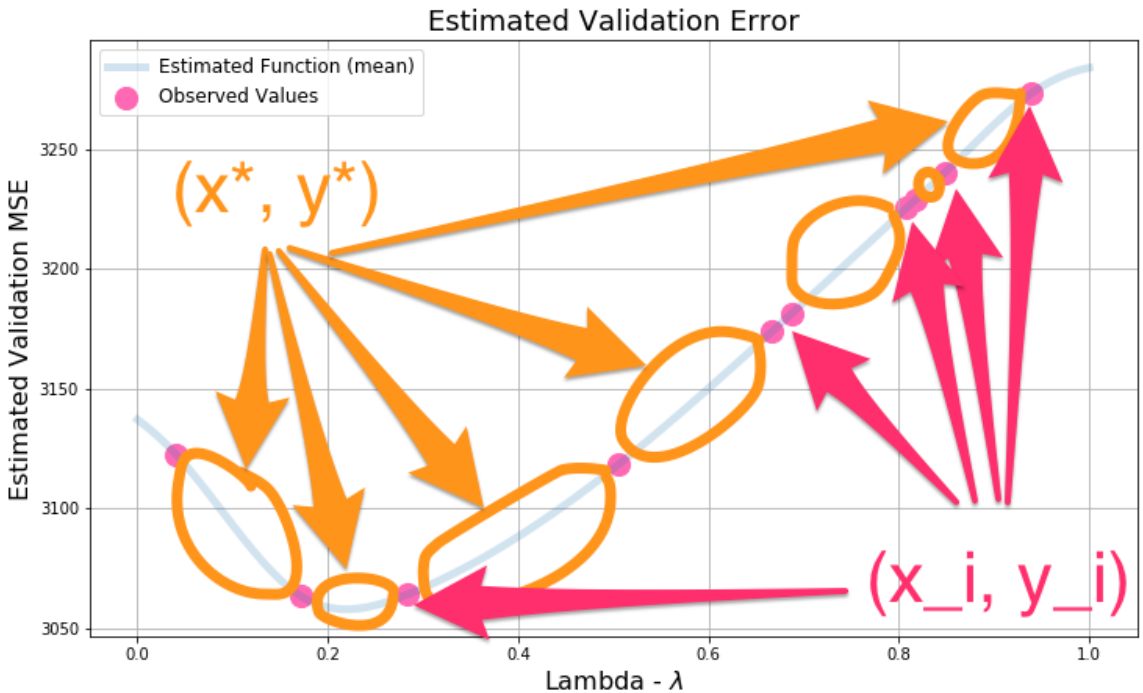
The Gaussian Process model allows regression (inference) from a small number of points.

This works because we assume a Kernel matrix which describes the similarity (covariance) between each of the points based on their distance.
Sometimes Gaussian Process Regression is called **Kriging**.

Gaussian Process Regression

Suppose we have some training data $D = \{(x_i, y_i) | i = 1, \ldots, n\}$ and that y_i is noise free.

We wish to estimate the values y_* (so called "test values"), which we have not observed.



Gaussian Process Regression

To make predictions we construct the joint distribution of the training and test values

$$\begin{bmatrix} \mathrm{y} \\ \mathrm{y}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{bmatrix}, \begin{bmatrix} \mathrm{K} & \mathrm{K}_* \\ \mathrm{K}_*^T & \mathrm{K}_{**} \end{bmatrix}\right)$$

where $\mathrm{y} = [y_1, y_2, \ldots, y_n]$ and K, K_* and K_{**} are the kernel matrices for training points, training and test points, and for the test points respectively.

By the conditioning property of Gaussian distributions we get the conditional probability of y_* given $\mathrm{y}, \mathrm{X}, \mathrm{X}_*$

$$p(\mathrm{y}_* | \mathrm{y}, \mathrm{X}, \mathrm{X}_*) \sim \mathcal{N}(\mathrm{K}_*^T \mathrm{K}^{-1} \mathrm{y}, \mathrm{K}_{**} - \mathrm{K}_*^T \mathrm{K}^{-1} \mathrm{K}_*)$$

GPR in BHO

In BHO the policy will select a new point to evaluate.

The point and reward (value) are then added to our training data.

Afterwards we can then re-evaluate $p(\mathrm{y}_* | \mathrm{y}, \mathrm{X}, \mathrm{X}_*)$ to obtain the updated model.

Bayesian Hyper-Paramater Optimisation

Wrapping up:

- using a Gaussian Process model for hyper-parameter optimisation is generally known as “Bayesian Optimisation”
- this approach can significantly reduce the time it takes to tune a model
- the benefits are largest when the training time is long, rather than a large number of hyper-parameters
- we can parallelise the approach either during training or by drawing multiple samples and updating the model using many test points