Bayesian Optimisation

Basics

The generic optimisation problem:

$$x^* = \text{argmin } f(x) \qquad \text{subject to} \quad \begin{matrix} c_i(x) = 0 \\ c_i(x) \geq 0 \end{matrix}$$

where $f(x)$ is the (scalar) ***objective function*** to be minimised or maximised, x is the vector of ***unknowns*** or ***parameters*** and c_i are ***constraint functions***.

Basics

The generic optimisation problem:

$$x^* = \text{argmin } f(x) \qquad \text{subject to} \quad \begin{matrix} c_i(x) = 0 \\ c_i(x) \geq 0 \end{matrix}$$

where $f(x)$ is the (scalar) ***objective function*** to be minimised or maximised, x is the vector of ***unknowns*** or ***parameters*** and c_i are ***constraint functions***.

Convexity

Many algorithms work best if $f(x)$ is ***convex***:

local minimum = global minimum

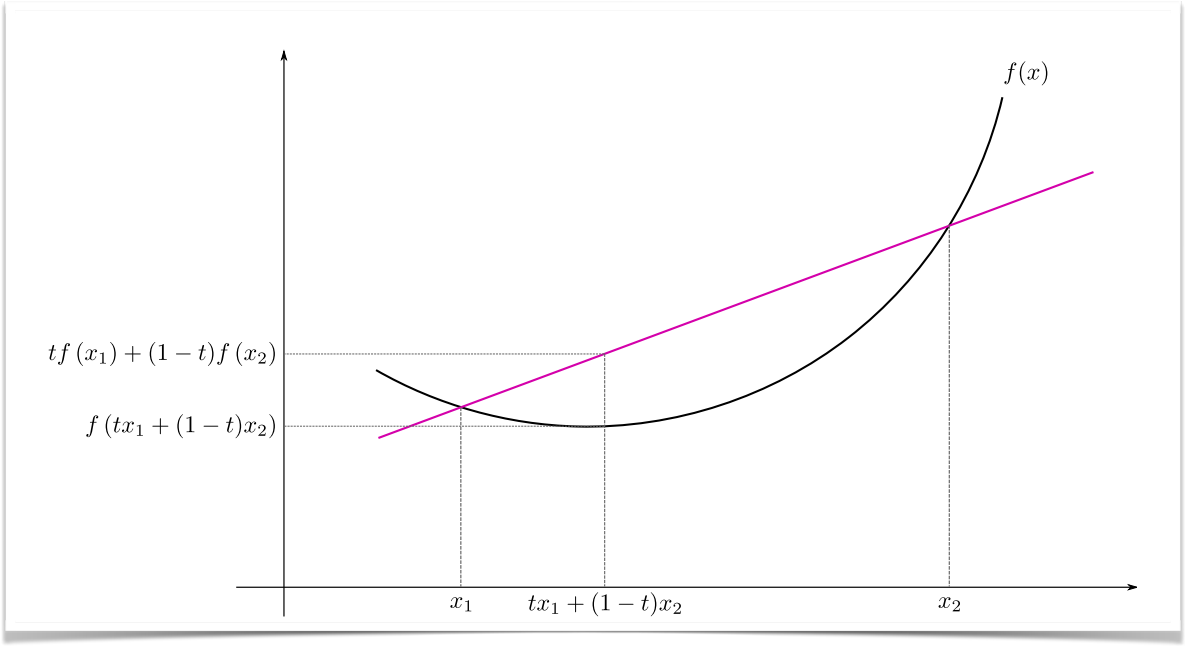
Mathematical definition for $x_1, x_2 \in X$, a convex subset:

for all $0 \leq t \leq 1$

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

Convexity

$f(x)$ is convex: if the line segment between any two points $f(x_1), f(x_2)$ is either equal or above $f(x)$ for $x = tx_1 + (1-t)x_2$

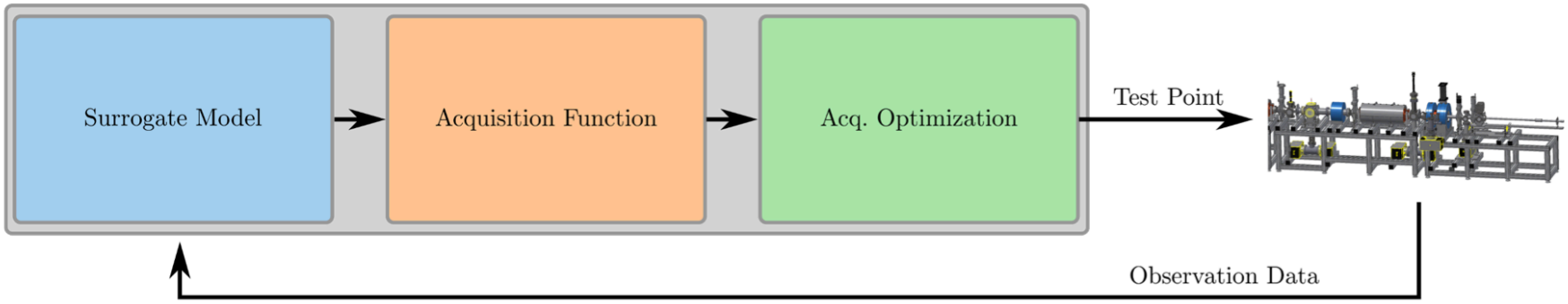


By Eli Osherovich - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=10764763

Why Bayesian Optimisation (BO)?

Non-parametric, global black-box optimisation for non-convex problems.

Key elements: **probabilistic surrogate model of objective function, Bayesian statistics,** optimisation of **Acquisition Function**



Optimisation is a sequence of decisions: at each iteration we must choose where to make the next observation and then terminate depending on the outcome.

Sample-efficiency: number of iterations to be kept low → cost of interaction with accelerator high

Bayesian statistics for sample-efficient optimisation

Use probabilistic model as surrogate of objective function → learn probability function given observations

Bayesian optimisation relies on *Bayesian inference.*

Assume that observation y at x is distributed according to some observation model ϕ depending on the underlying objective function $f(x)$.

$$p(y \mid x, \phi)$$

Bayesian inference approach: **all unknowns are treated as random variables** → use beliefs about quantities (probability distributions) → inference is then inductive process: beliefs are iteratively refined with observed data

Bayesian statistics for sample-efficient optimisation

Start with so-called prior distribution $p(\phi \mid x)$…plausible values for ϕ before observing any data. (The real measurements are y, ϕ is one or more values $\phi = f(x)$)

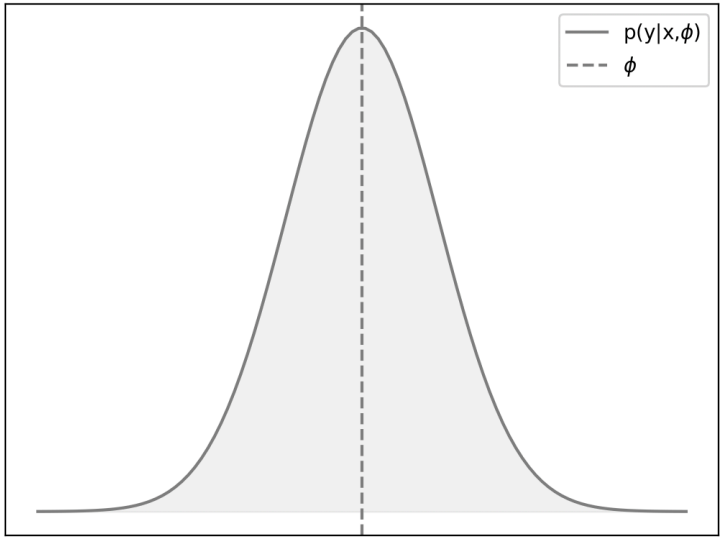
We update our prior through observation to get the posterior distribution using **Bayes' rule:**

prior **likelihood or observation model**

$$p(\phi \mid \mathcal{D}) = p(\phi \mid x, y) = \frac{p(\phi \mid x) p(y \mid x, \phi)}{p(y \mid x)}$$

marginal likelihood, normalisation constant

Example for $p(y \mid x, \phi)$ with additive noise: $y = \phi + \varepsilon$



Example for additive Gaussian noise: $y = \phi + \varepsilon$

Example: Bayes' rule for linear regression

Bayesian analysis for standard linear regression with Gaussian noise

$$f(\mathbf{x}) = \mathbf{x}^{\mathbf{T}}\mathbf{w} \quad \text{and } y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0,\sigma_n^2)$$

The likelihood:

$$p(y\,|\,\mathbf{x},\mathbf{w}) = \prod_{i=1}^{n} p(y_i\,|\,\mathbf{x}_i,\mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp(-\frac{(y_i - \mathbf{x}_i^T\mathbf{w})^2}{2\sigma_n^2}) = \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp(-\frac{1}{2\sigma_n^2}\,|\,\mathbf{y} - X^T\mathbf{w}\,|^2)$$

$$= \mathcal{N}(X^T\mathbf{w}, \sigma_n^2 I)$$

Example: Bayes' rule for linear regression

Bayesian analysis for standard linear regression with Gaussian noise

$$f(\mathbf{x}) = \mathbf{x}^{\mathbf{T}}\mathbf{w} \quad \text{and } y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0,\sigma_n^2)$$

The likelihood:

$$p(y\,|\,\mathbf{x},\mathbf{w}) = \prod_{i=1}^{n} p(y_i\,|\,\mathbf{x}_i,\mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp(-\frac{(y_i - \mathbf{x}_i^T\mathbf{w})^2}{2\sigma_n^2}) = \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp(-\frac{1}{2\sigma_n^2}\,|\,\mathbf{y} - X^T\mathbf{w}\,|^2)$$

$$= \mathcal{N}(X^T\mathbf{w}, \sigma_n^2 I)$$

Use zero mean Gaussian **prior** with covariance matrix Σ_p for weights $\mathbf{w}$: $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$

Example: Bayes' rule for linear regression

Bayesian analysis for standard linear regression with Gaussian noise

$$f(\mathbf{x}) = \mathbf{x}^{\mathbf{T}}\mathbf{w} \quad \text{and } y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0,\sigma_n^2)$$

The likelihood:

$$p(y\,|\,\mathbf{x},\mathbf{w}) = \prod_{i=1}^{n} p(y_i\,|\,\mathbf{x}_i,\mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \mathbf{x}_i^T\mathbf{w})^2}{2\sigma_n^2}\right) = \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}\,|\,\mathbf{y} - X^T\mathbf{w}\,|^2\right)$$

$$= \mathcal{N}(X^T\mathbf{w}, \sigma_n^2 I)$$

Use zero mean Gaussian **prior** with covariance matrix Σ_p for weights $\mathbf{w}$: $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$

Posterior with Bayes' rule: $p(\mathbf{w}\,|\,\mathbf{y}, X) \propto p(\mathbf{y}\,|\,X,\mathbf{w})p(\mathbf{w})$

$$p(\mathbf{w}\,|\,X,\mathbf{y}) \propto \exp\left(-\frac{1}{2\sigma_n^2}(\mathbf{y} - X^T\mathbf{w})^T(\mathbf{y} - X^T\mathbf{w})\right) \exp\left(-\frac{1}{2}\mathbf{w}^T\Sigma_p^{-1}\mathbf{w}\right)$$

$$\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \bar{\mathbf{w}})^T\Sigma_w^{-1}(\mathbf{w} - \bar{\mathbf{w}})\right) \ldots \text{Gaussian! } \mathcal{N}(\bar{\mathbf{w}}, \Sigma_w)$$

Example: Bayes' rule for linear regression

$$\bar{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}XX^T + \Sigma_p^{-1})^{-1}X\mathbf{y}$$

$$\Sigma_w = \left(\sigma_n^{-2}XX^T + \Sigma_p^{-1}\right)^{-1}$$

We need the posterior for inference through the ***predictive distribution*** for $f(\mathbf{x}^*)$:

average over all linear models

$$p(f^* \,|\, \mathbf{x}^*, X, y) = \int p(f^* \,|\, \mathbf{x}^*, w)p(\mathbf{w} \,|\, X, \mathbf{y})d\mathbf{w} = \mathcal{N}(\mathbf{x}^{*\mathbf{T}}\bar{\mathbf{w}}, \mathbf{x}^{*\mathbf{T}}\Sigma_w\mathbf{x}^*)$$…and this again

Gaussian!

Gaussian processes

The surrogate model:

- Key is "sample efficiency": extract a lot of information from a small number of data points
- Produces explicit uncertainty estimation to allow "global optimisation"

Gaussian processes = a class of nonparametric regression models; extension of the multivariate normal distribution for modelling functions on the infinite domains.

- Instead of a discrete mean μ and covariance matrix K, will have mean function $\mu(\mathbf{x})$ and covariance function or *kernel* function $k(x, x')$

- With data at $x, \phi = f(x)$ of objective function $f(x)$, the distribution of ϕ is also multivariate

$$p(\phi \,|\, x) = \mathcal{N}(\phi; \mu, \Sigma) \text{ where } \mu = \mathbb{E}[\phi \,|\, x] = \mu(x) \text{ and matrix } \Sigma = \mathrm{cov}[\phi \,|\, x] = K(x, x)$$

Gaussian processes

Use **Gaussian process** to describe **distribution of functions.**

A Gaussian process is completely defined by its mean function $\mu(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$

$$\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - \mu(\mathbf{x}))(f(\mathbf{x}') - \mu(\mathbf{x}'))]$$

$$f(\mathbf{x}) \sim GP(\mu(\mathbf{x}, k(\mathbf{x}, \mathbf{x}')))$$

To predict test point f^* at x^* from posterior with given data X, Y:

$p(f^* \,|\, x^*, X, Y) = \mathcal{N}(\mu^*, \sigma^{*2})$, assume joint prior distribution of $\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,x^*) \\ K(x^*,X) & K(x^*,x^*) \end{bmatrix}\right)$

$\rightarrow \mu^* = K(x^*, X)K(X,X)^{-1}Y$

$\rightarrow \sigma^{*2} = K(x^*,x^*) - K(x^*,X)K(X,X)^{-1}K(X,x^*)$

Beware!!: $K^{-1} \sim \mathcal{O}(N^3)$

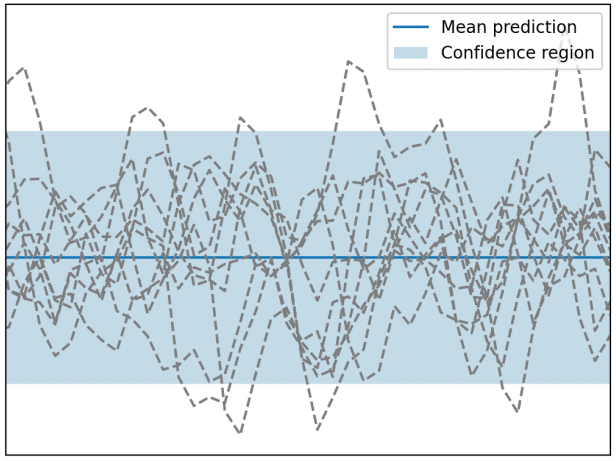
Gaussian processes

In case prior $\mu \neq \mathbf{0}$

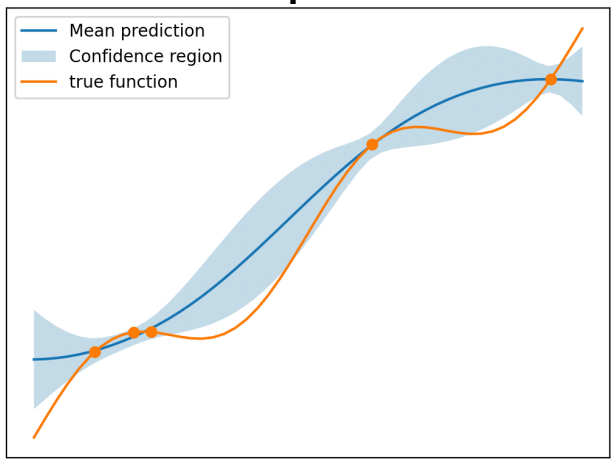
$$\mu^*(x) = \mu(x^*) + K(x^*, X)K(X, X)^{-1}(Y - \mu(X))$$

Example:

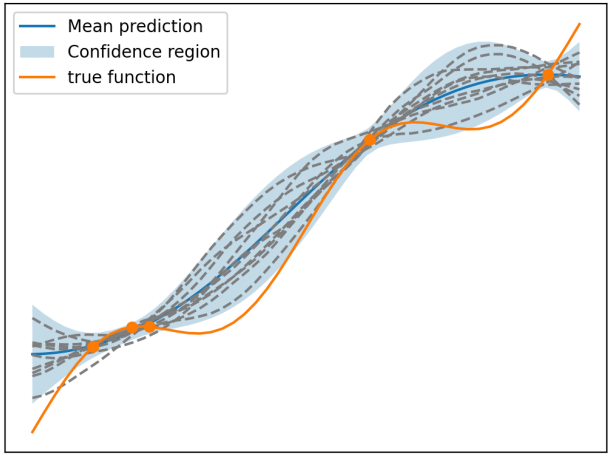
Prior



Posterior mean and confidence with 5 data pointsPrior



Some samples of posterior distribution



Modeling with Gaussian Processes

The fidelity of model of the objective function is key for optimisation performance.

It depends on the choice of prior $\rightarrow \mu(x), k(x, x')$ and as it turns out mainly about the $k(x, x')\ldots$ prior mean only enters as a shift.

Covariance function also called **kernel**, or similarity function: measures how similar pairs of input data points are to each other.

i.e. measures correlation between two function values ϕ and ϕ':

The art is in choosing a kernel the best represents the correlation in the data.

Picking a kernel also sets the characteristics of the function.

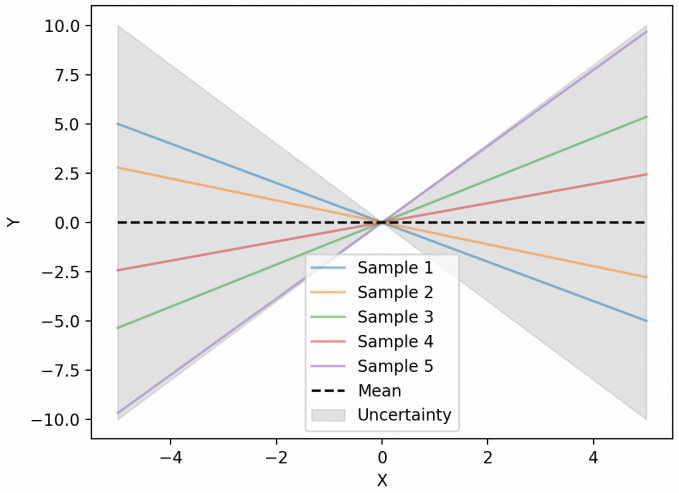
Different kernels have their own different hyper parameters.

Linear and polynomial kernels

Linear kernel: computationally efficient and interpretable models; but limited expressiveness.

$$k_{linear}(x, x') = \theta \cdot x^T x' + c$$

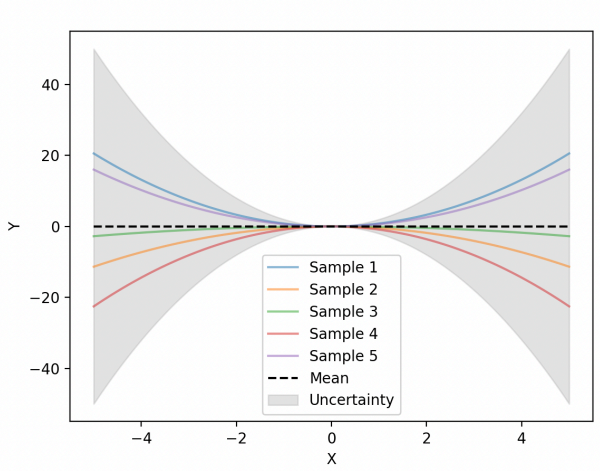
Prior of linear kernel with $c = 0$ and 5 samples



One step further: Polynomial kernel; offers more flexibility

choosing appropriate degree d can be difficult

$$k_{poly}(x, x') = (x^T x' + c)^d$$



Prior of poly kernel with $d = 2$ and 5 samples

Matern kernel

Characterised by two parameters the length scale l and the smoothness parameter ν.

In the most general form:

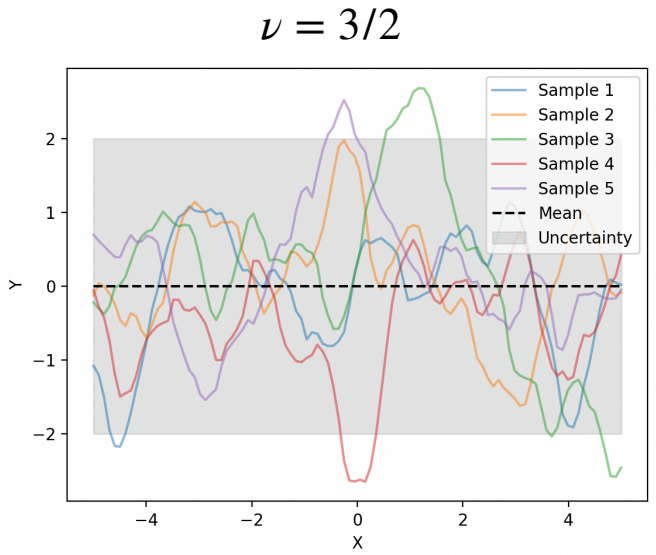
$$k_{Matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{l} \right)$$

r is the distance between two points, Γ is the gamma function and K_{ν} the modified Bessel function.

There are closed-form solutions for $\nu = 1/2, 3/2, 5/2$ and $7/2$.

$$k_{1/2}(r) = \exp\left(-\frac{r}{l}\right), \; k_{3/2}(r) = \left(1 + \frac{\sqrt{3} r}{l}\right) \exp\left(-\frac{\sqrt{3} r}{l}\right),$$

$$k_{5/2}(r) = \left(1 + \frac{\sqrt{5} r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5} r}{l}\right)$$



Special Matern kernel: Radial Basis Function kernel

Matern kernel for $\nu \to \infty$ becomes $k_{RBF} = \exp\left(-\frac{1}{2}\frac{r^2}{l^2}\right)$

For $\nu > 5/2$ becomes very smooth, equivalent to the Radial Basis Function (RBF) kernel anyway

Other name *squared exponential kernel*

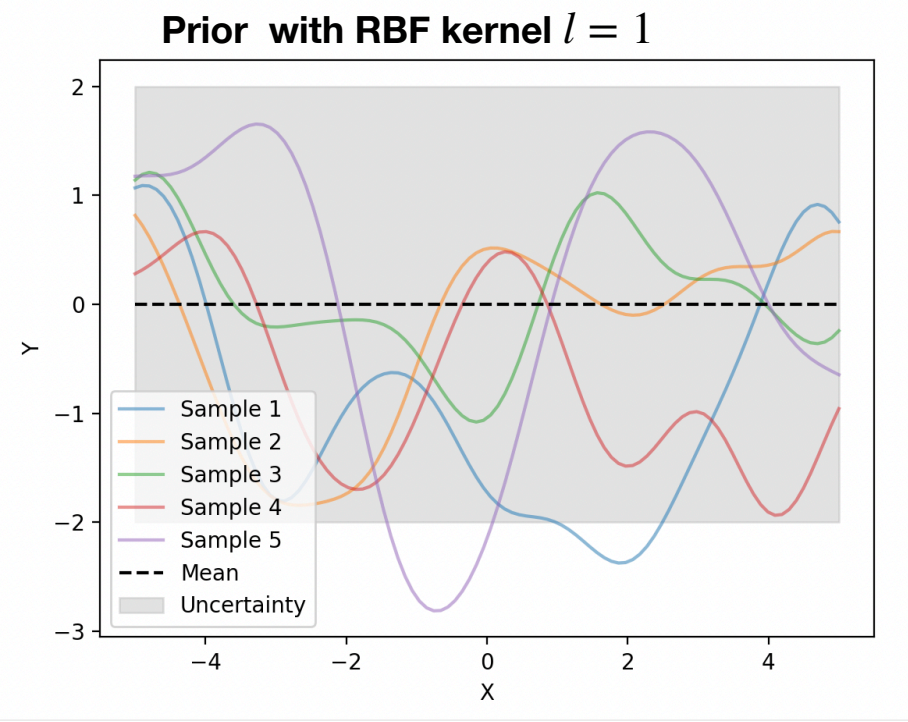
Probably the most frequently used kernel

Captures smooth and continuous functions

For non-linear relationships between inputs and outputs.

Captures complex patterns.

Struggles with discontinuous or highly oscillatory functions.

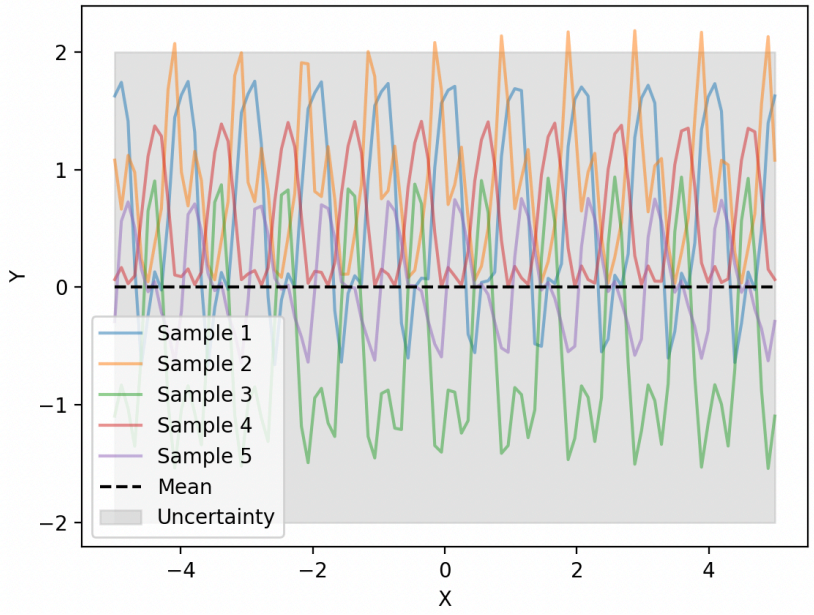


Periodic kernel

Kernel designed to model periodic patterns in data

$$k_{periodic}(x, x') = \exp\left(-\frac{2\sin^2(\pi|x - x'|/p)}{l^2}\right)$$

where p represents the period of the periodic pattern and l is again the length scale parameter.



Spectral mixture kernel

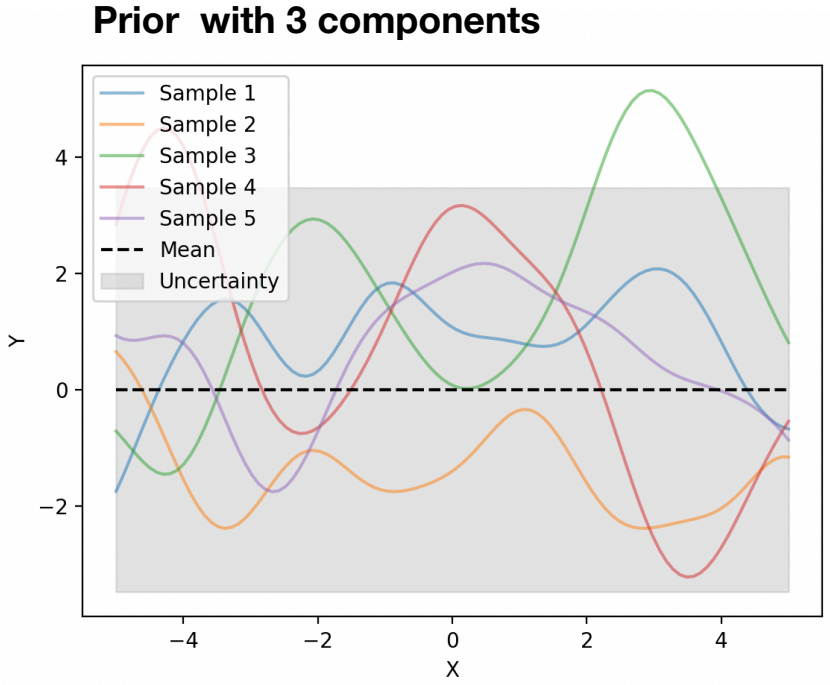
Simple and powerful idea: parameterise a space of "stationary" covariance functions by some suitable family of mixture distributions (e.g. Gaussian Mixture) in the Fourier domain representing their spectral density.

The kernel is then the Fourier transform of this. In case of Gaussian Mixture, convenient Fourier transform:

$$k_{SM}(x, x') = \sum_{i=1}^{n} w_i \exp\left(\frac{-2\pi^2 |x - x'|^2}{\lambda_i^2}\right), \text{ where}$$

w_i is the weight of the ith component,

λ_i is the length scale and n is the number of mixture components. (Useful number is $n = 5$)



Combining kernels

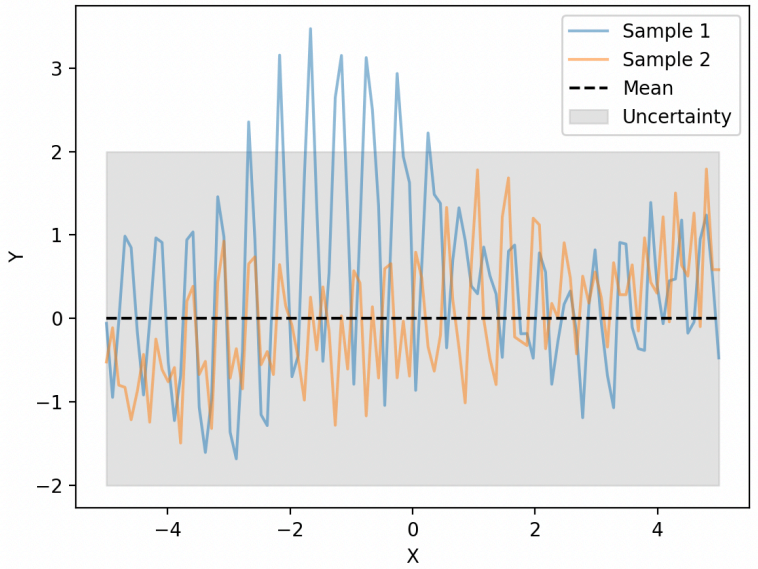
Kernels can be multiplied

e.g. multidimensional products: $k_{product}(x, y, x', y') = k_x(x, x') \cdot k_y(y, y')$

Example: locally periodic

$$k_{periodic}(x, x') = \exp\left(-\frac{2\sin^2(\pi\|x - x'\|/p)}{l^2}\right) \cdot \exp\left(-\frac{1}{2}\frac{\|x - x\|^2}{l^2}\right)$$

and summed and scaled!

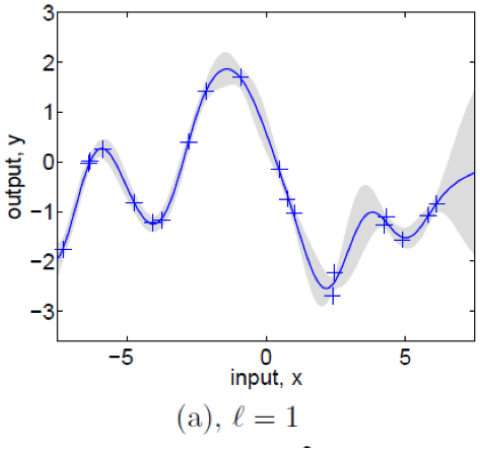


Effect of hyperparameters: e.g. length-scale

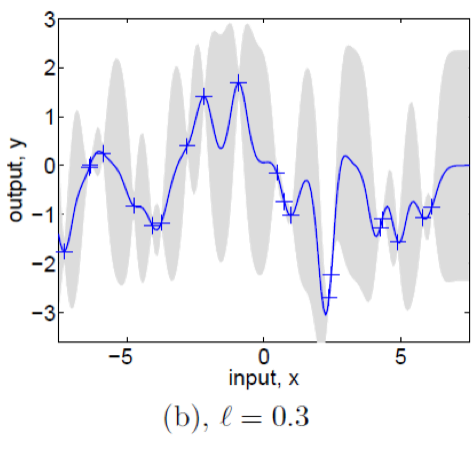
Each kernel has hyperparameters that control overall function behaviour: e.g. $\mathbf{l}$…length scale

As example…"squared exponential covariance": $k(x, x') = \exp(-\frac{1}{2\mathbf{l}^2}|x - x'|^2)$

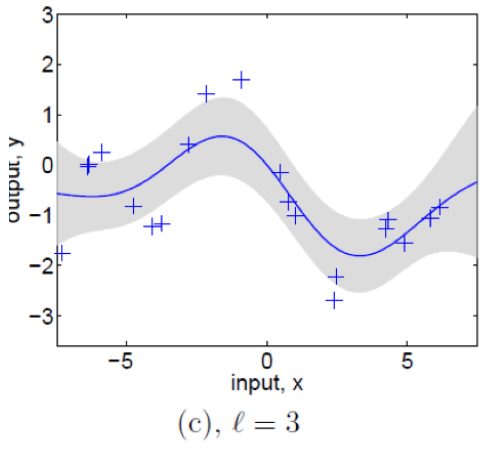
Perfect correlation locally, and decays with a length-scale.



(a), $\ell = 1$



(b), $\ell = 0.3$



(c), $\ell = 3$

from R&W 2006

Fitting a Gaussian Process to data

How to determine the hyperparameters θ of the kernel? → Fitting

As usual: Maximum Likelihood Estimation

$$\theta^* = \arg\max \log p(\mathbf{y} \mid \theta, X)$$

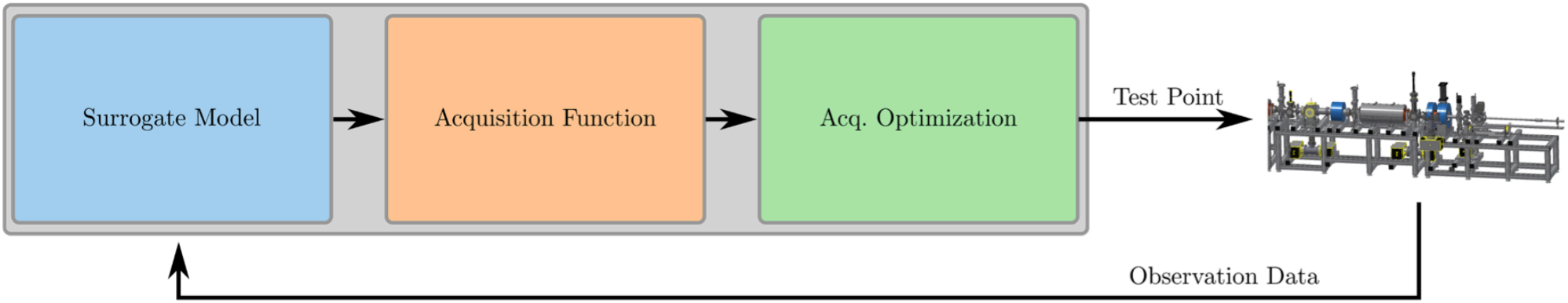
As likelihood is Gaussian log likelihood can be calculated analytically:

$$\log p(\mathbf{y} \mid \theta, X) = -\frac{1}{2} y^T K^{-1} y - \frac{1}{2} \log |K| - \frac{n}{2} \log 2\pi$$

Next step: Acquisition Functions

Optimisation is a sequence of decisions: at each iteration we must choose where to make the next observation and then terminate depending on the outcome.

→ Building and appropriate acquisition function! Again hyperparameters…



Acquisition Functions

The acquisition function “looks" at the surrogate model of the objective function and determines what areas in the domain $f(x)$ are worth exploring.

Note mostly BO is formulated as *maximisation problem*. → Acquisition Function should be constructed such that it assumes high values where either $f(x)$ is optimal or that we have not looked at yet.

→ in BO, it is not the surrogate model of $f(x)$ that is maximised directly, but AF.

The **Acquisition Function** will have to deal with the **trade-off between *exploration and exploitation*.**

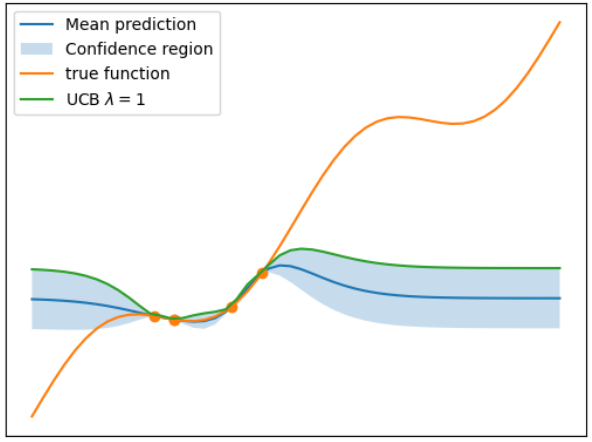
Upper Confidence Bound (UCB)

Probably the simplest acquisition function: contains explicit exploitation ($\mu(x)$) and exploration terms ($\sigma(x)$, standard deviation).

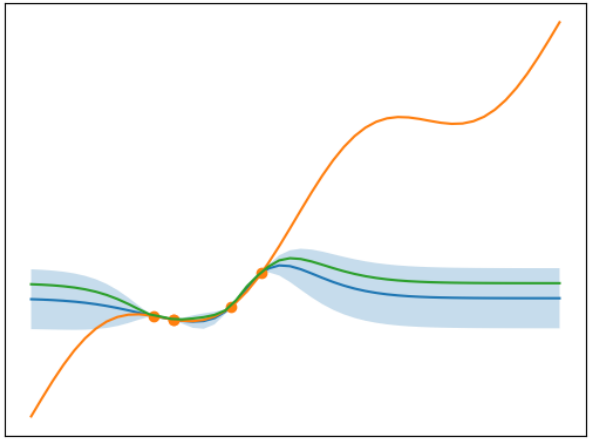
Hyperparameter λ, often also called β guides exploration vs. exploitation. $\lambda = \sqrt{\beta}$

$$a(x;\lambda) = \mu(x) + \lambda\sigma(x)$$

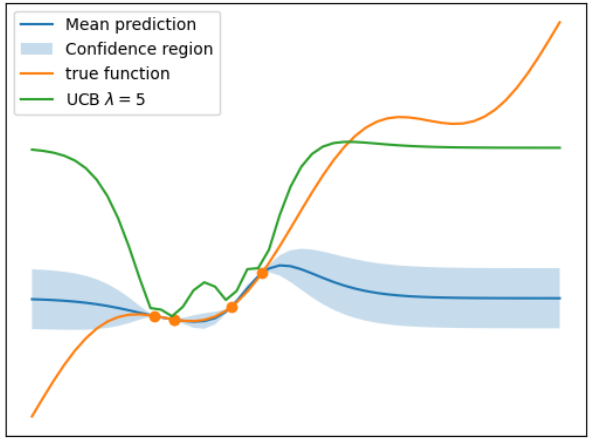
$\lambda = 1$



$\lambda = 0.5$



$\lambda = 5$



Acquisition Function: Expected Improvement EI (1)

First need to introduce *Probability of Improvement* PI

"Improvement" I: find $\max f(x)$ and so far best solution is x^*:

$$I(x) = max(f(x) - f(x^*), 0) \quad \ldots.\text{the maximum between } f(x) - f(x^*) \text{ and } 0$$

$PI(x)$ assigns the probability of $I(x) > 0$.

Acquisition Function: Expected Improvement EI (1)

First need to introduce *Probability of Improvement* PI

"Improvement" I: find $\max f(x)$ and so far best solution is x^*:

$$I(x) = max(f(x) - f(x^*), 0) \quad \ldots.\text{the maximum between } f(x) - f(x^*) \text{ and } 0$$

$PI(x)$ assigns the probability of $I(x) > 0$.

Can be calculated analytically in case of surrogate models with *GPs:* $f(x) \sim \mathcal{N}(\mu(x), \sigma^2(x))$

*Rep*arameterisation trick: $z \sim \mathcal{N}(0,1) \rightarrow f(x) = \mu(x) + \sigma(x)z \quad z = \dfrac{f(x) - \mu(x)}{\sigma(x)}$

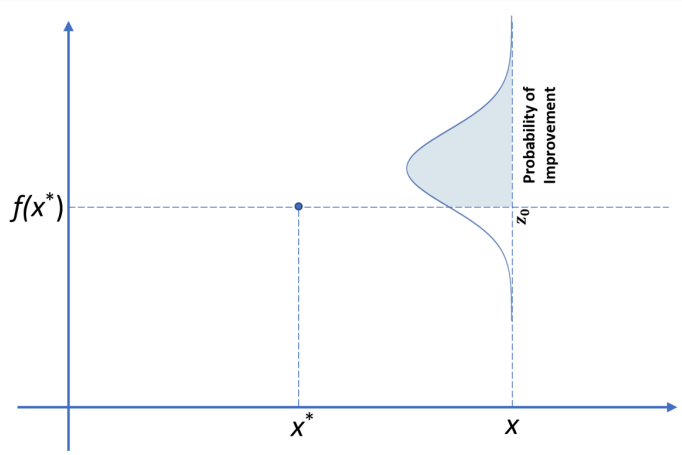
$PI(x) = P(I(x) > 0)$ equivalent with probability $P(f(x) > f(x^*))$.

Say value of $f(x^*)$ corresponds to z_0 at location x^*, then need to simply find probability of all $z > z_0$.

Acquisition Function: Expected Improvement EI (2)

Thus:

$$PI(x) = 1 - CDF(z_0) = CDF(-z_0) = CDF\left(\frac{\mu(x) - f(x^*)}{\sigma(x)}\right)$$



from https://ekamperi.github.io/

Expected Improvement $EI(x)$: with $\phi(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-z^2/2\right)$

$$EI(x) \equiv \mathbb{E}[I(x)] = \int_{-\infty}^{\infty} I(x)\phi(z)dz$$

$$EI(x) = \underbrace{\int_{-\infty}^{z_0} I(x)\phi(z)dz}_{=0} + \int_{z0}^{\infty} I(x)\phi(z)dz$$ …first integral zero because $I(x) = max(f(x) - f(x^*),0)$

Acquisition Function: Expected Improvement EI (3)

$$EI(x) = \int_{z0}^{\infty} I(x)\phi(z)dz = \int_{z0}^{\infty} \left(\mu + \sigma z - f(x^*)\right) \phi(z)dz$$

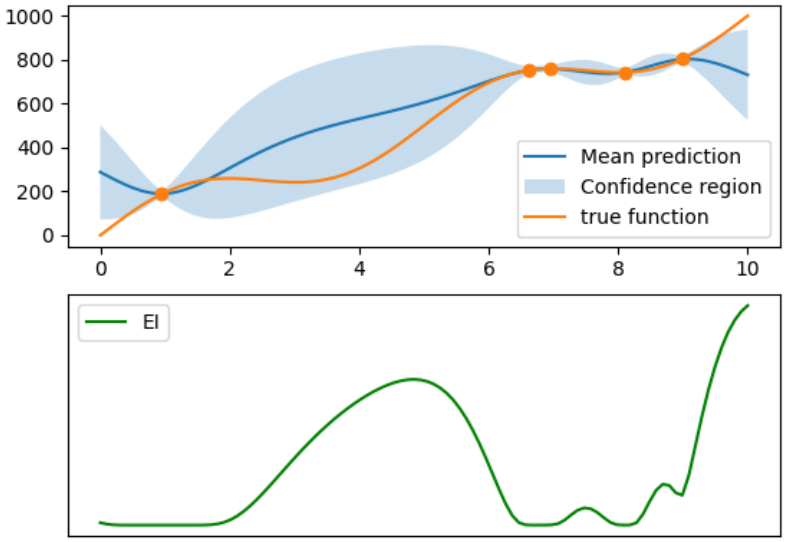
$$= \left(\mu - f(x^*)\right) \int_{z_0}^{\infty} \phi(z)dz + \frac{\sigma}{\sqrt{2\pi}} \int_{z_0}^{\infty} z e^{-z^2/2} dz$$

$$= \left(\mu - f(x^*)\right) CDF\left(\frac{\mu - f(x^*)}{\sigma}\right) + \frac{\sigma}{\sqrt{2\pi}} \int_{z_0}^{\infty} \frac{d\left(e^{-z^2/2}\right)}{dz} dz$$

$$= \left(\mu - f(x^*)\right) CDF\left(\frac{\mu - f(x^*)}{\sigma}\right) + \sigma\phi\left(\frac{\mu - f(x^*)}{\sigma}\right)$$

Acquisition Function: Expected Improvement EI (4)

$$EI(x) = \left(\mu - f(x^*)\right) CDF\left(\frac{\mu - f(x^*)}{\sigma}\right) + \sigma\phi\left(\frac{\mu - f(x^*)}{\sigma}\right)$$



Can also add extra hyperparameter to tune exploitation vs exploration.

Constraints?

$EI(x)$ can be easily extended to include inequality constraints.

$$\max_{c(x)\leq\lambda} f(x)$$

Idea: learn $f(x)$ as GP as well as $c(x)$ (i.e. $\tilde{c}(x)$), and then weigh $EI(x)$ with probability of constraint satisfied.

$$PF(x) := Pr[\tilde{c}(x) \leq \lambda]$$

$$EI_C(x) = PF(x) \cdot EI(x)$$

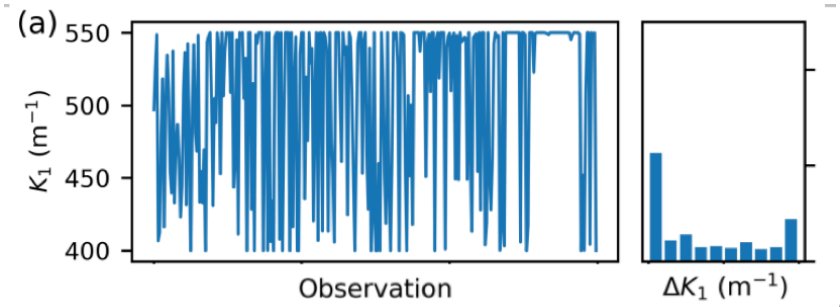
In case of multiple inequality constraints and with the assumption that the constraints are conditionally independent given x

$$PF(x) = \Pi_{i=1}^{k} p(\tilde{c}_i(x) \leq \lambda_i)$$

Proximal biasing

Particle accelerator optimisation often requires incremental traversal of parameter space to maintain accelerator stability.

Ensure that exploration not in too large steps, could be problematic for equipment,…



R. Roussel

→ weight acquisition function by distance to previous set point.

$$\hat{a}(x) = a(x) \cdot \exp\left(-\frac{(x - x_0)^2}{2\beta^2}\right) \quad \text{(only valid for } a(x) \geq 0\text{)}$$

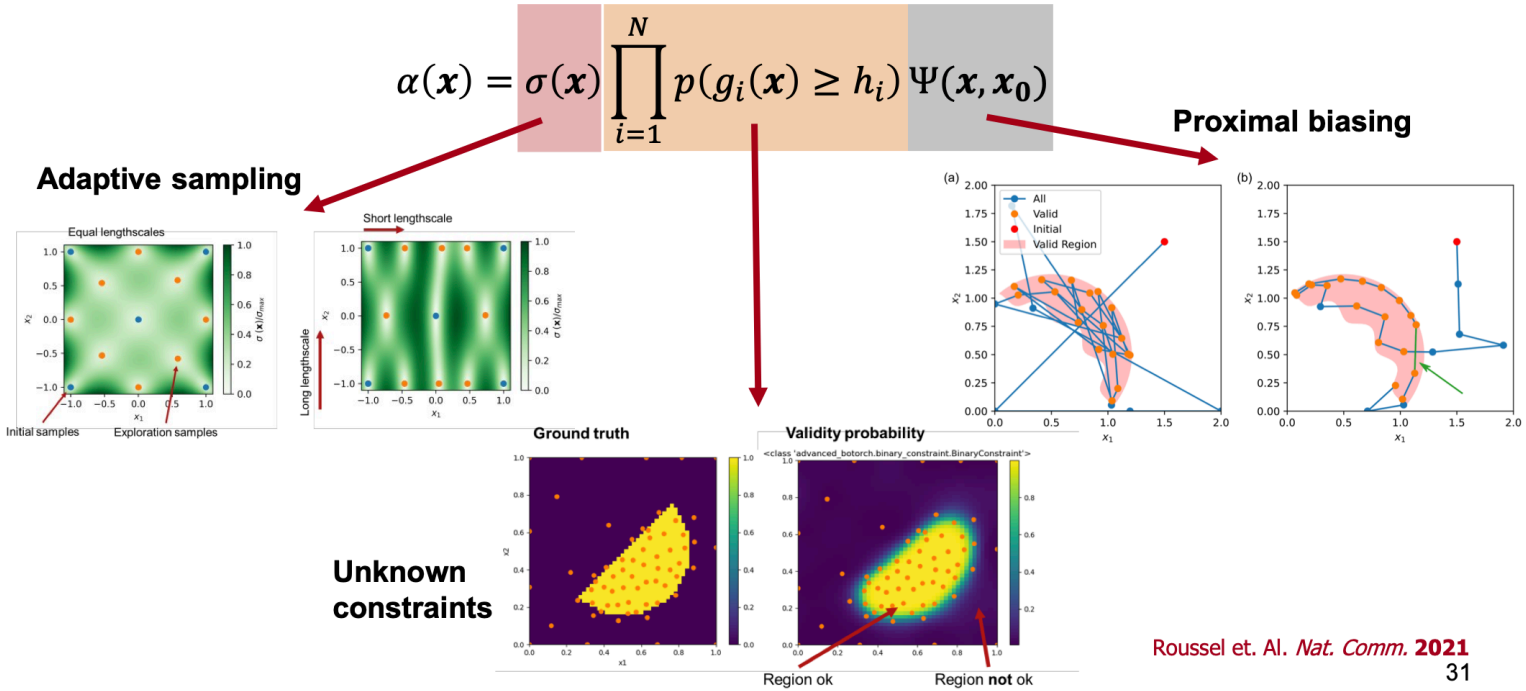
R. Roussel et al , NeurIPS 2021

Uncertainty sampling and Bayesian Exploration

Acquisition function for system identification

$a(x) = \sigma(x)$ ….adaptive sampling

→ Bayesian exploration:

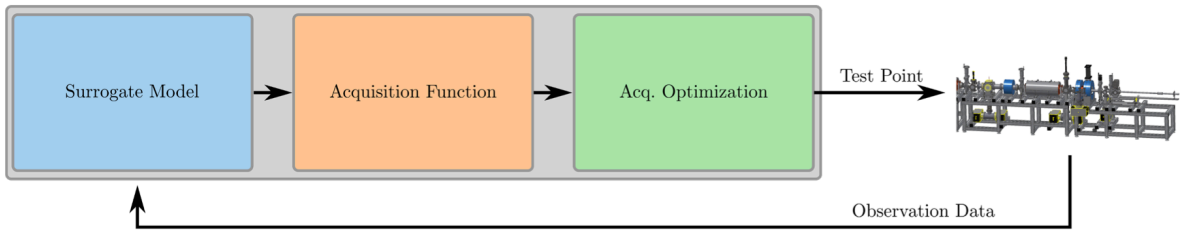


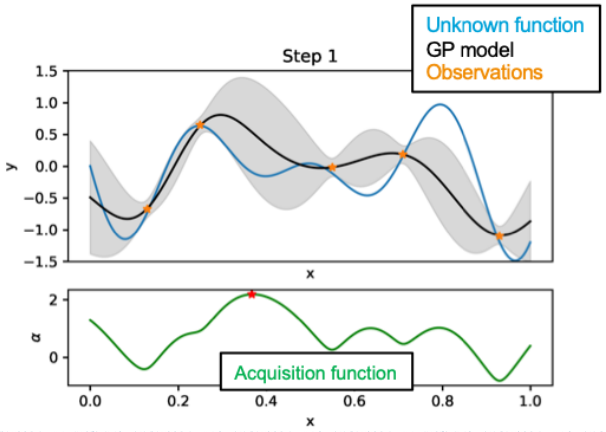
Roussel et. Al. *Nat. Comm.* **2021**

https://ml4physicalsciences.github.io/2021/files/NeurIPS_ML4PS_2021_82.pdf

Bayesian Optimisation (BO)

Majority of talks in "optimisation and control" session about BO





Recent publication of review paper:
Bayesian optimisation algorithms for accelerator physics
Phys. Rev. Accel. Beams 27, 084801

State-of-the-art BO:

- Model-based priors for various applications
- Multi-fidelity BO for laser plasma accelerators
- SafeOpt to include safety constraints - faster convergence with ModSafeOpt
- Information-based Bayesian Optimisation with virtual objectives
- …

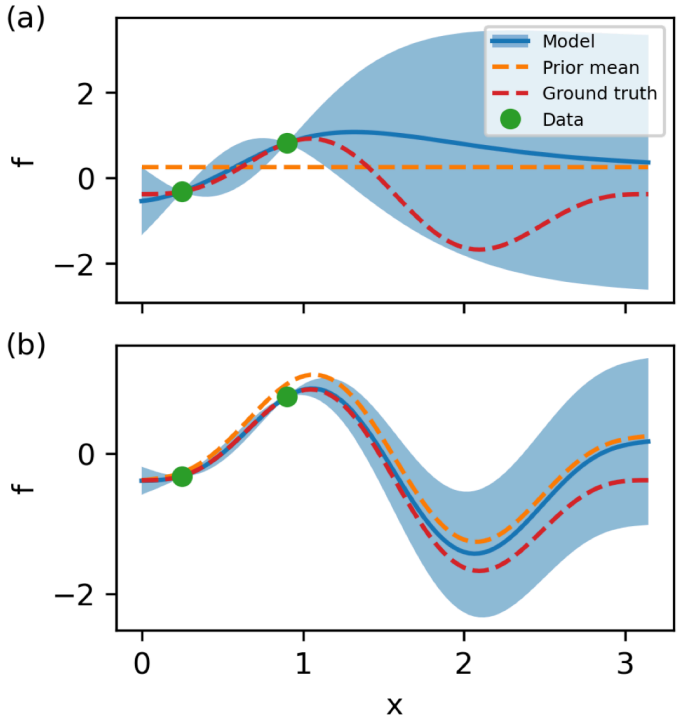
Example: BO with neural network mean priors

Including prior information through historical data not efficient with GPs.

→ Modify $p(A)$ in $p(A \mid B) \propto p(B \mid A)p(A)$

Instead of constant prior in GP → GP becomes model of model → much more sample-efficient

Model of non-constant mean $\mu(\mathbf{x})$ from simulation, ANN of historical data, other GP,…

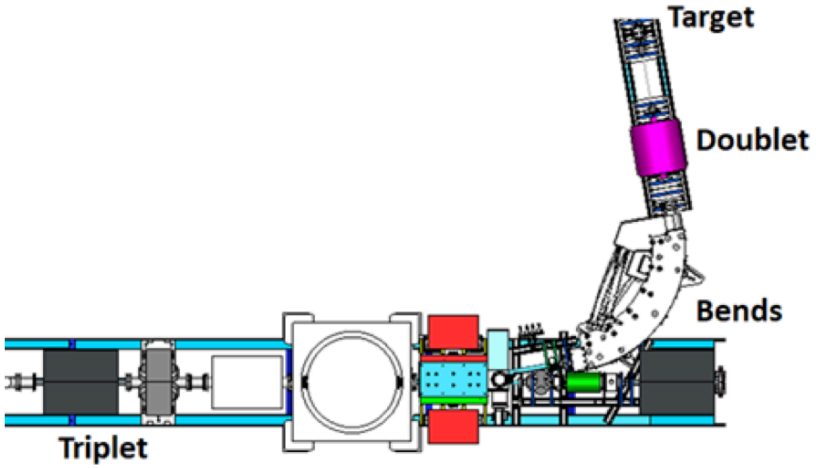


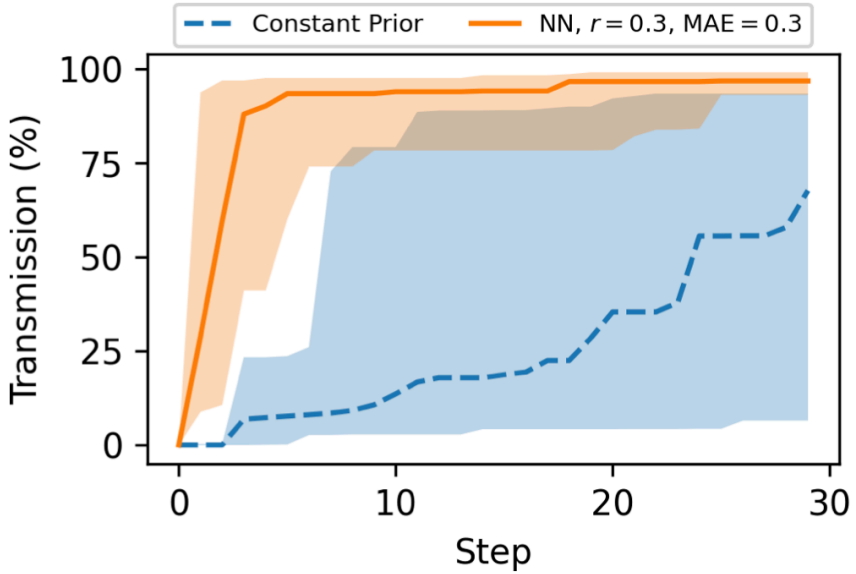
Example: BO with neural network mean priors

Example from ATLAS

- **Argonne Tandem Linear Accelerator System** for study of low-energy nuclear physics with heavy ions
- Optimise transmission to target: 5 DOF
  - Trained ANN from previous 3k dataset of ^{14}N run and used it for optimising ^{16}O run

Argonne Tandem Linear Accelerator System is a US Department of Energy User Facility dedicated to the study of low-energy nuclear physics with heavy ions

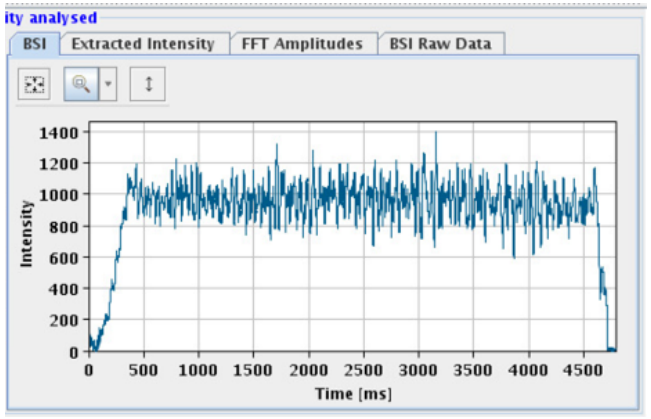




Courtesy T. Boltz et al, arXiv:2403.03225

Example: Continuous control with BO → ABO

Controlling the $n \times 50$ Hz noise in the slow extracted spill to the North Area Experimental Hall at CERN SPS.



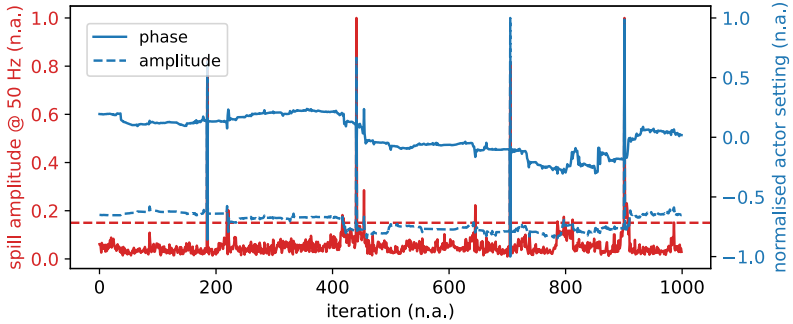
Modulate voltage of main quadrupoles at $n \times 50$ Hz to compensate.

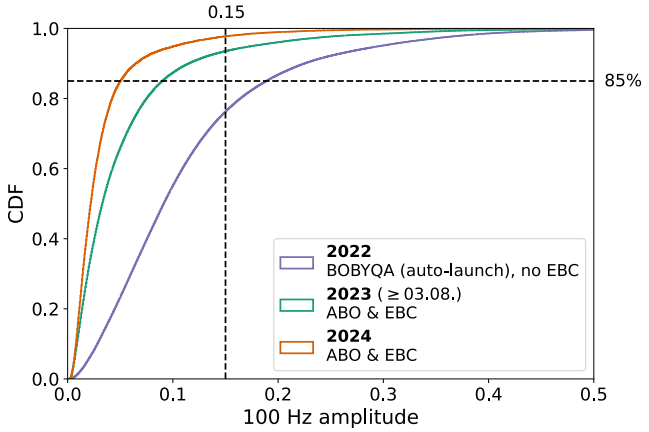
Spill noise changes over time following the European grid.

→ adaptive continuous control, **A**daptive **B**ayesian **O**ptimisation

Model objective function f as $f(\mathbf{x}, t)$. Spectral mixture kernel S for t and Matern for control parameters

Kernel: $k([t_1, \mathbf{x_1}], [t_2, \mathbf{x_2}]) = \theta_k \times S(t_1, t_2) \times M(\mathbf{x_1}, \mathbf{x_2})$





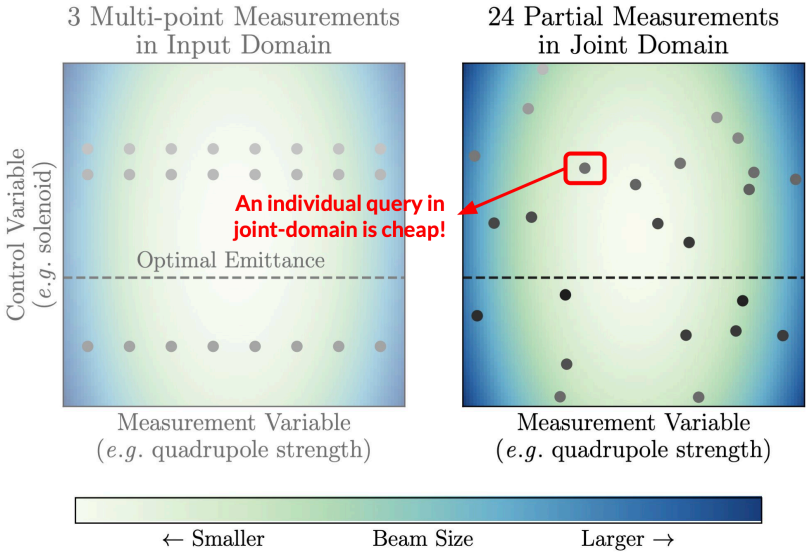
Will be key for:

SHiP
Search for Hidden Particles

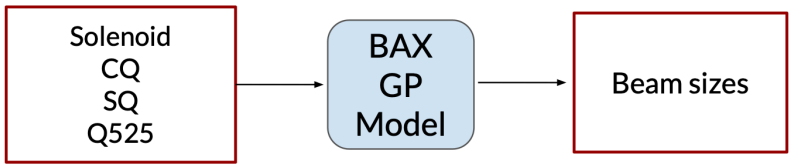
Example: info-based BAX - virtual objectives

Example for emittance tuning

- BAX (**B**ayesian **A**lgorithm e**X**ecution): https://willieneis.github.io/bax-website/
- Minimum emittance important for many applications: e.g. determines brightness of X-rays in FELs
- Classical methods slow due to multi-point queries: quadrupole scans for emittance evaluation



Courtesy S. Miskovich et al

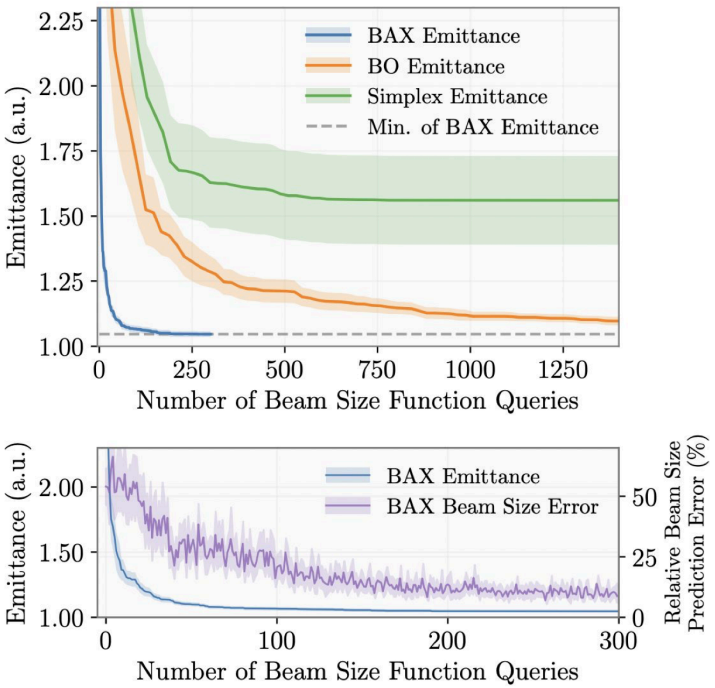


→ Virtual emittance scans on posterior samples of GP as input to acquisition function optimisation

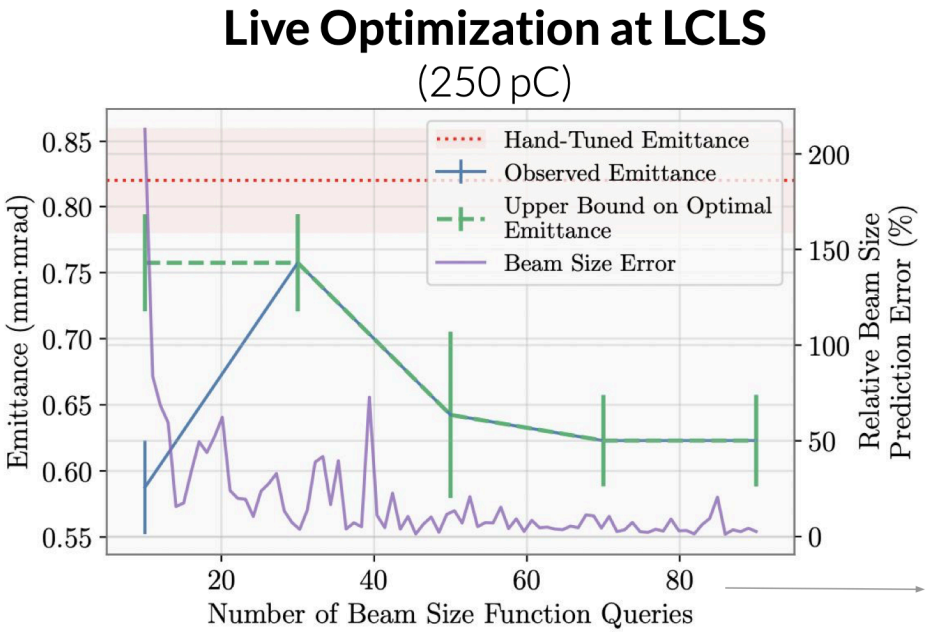
→ choose queries (settings) to maximise information and minimise emittance

Example: info-based BAX - virtual objectives

Example from noisy LCLS simulation @ SCLAC

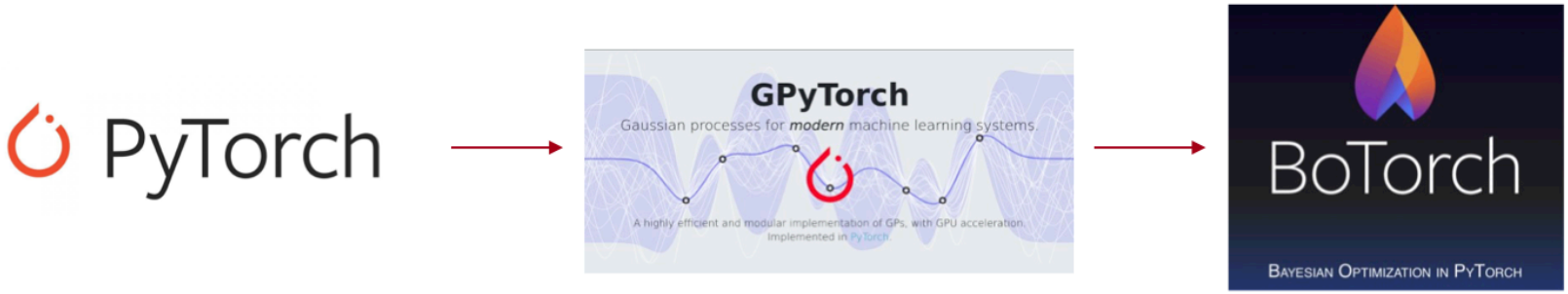


Proxy for convergence



Courtesy S. Miskovich et al

BoTorch - basic, basic intro



Use BoTorch

- Single/multi-objective Bayesian optimisation (serial and parallel)
- Constraints
- Proximal biasing
- Can incorporate pyTorch Modules
- GPU accelerated

BoTorch - basic, basic intro

Algorithm 1 Basic pseudo-code for Bayesian optimization

Place a Gaussian process prior on f
Observe f at n_0 points according to an initial space-filling experimental design. Set $n = n_0$.
while $n \leq N$ **do**
  Update the posterior probability distribution on f using all available data
  Let x_n be a maximizer of the acquisition function over x, where the acquisition function is computed using the current posterior distribution.
  Observe $y_n = f(x_n)$.
  Increment n
end while
Return a solution: either the point evaluated with the largest $f(x)$, or the point with the largest posterior mean.

BoTorch - basic, basic intro

```python
from botorch.models import SingleTaskGP


def ground_truth(x):
    return torch.sin(2 * 3.14 * x / 1.0)
```

Define model: fix the hyperparameters for example

```python
def create_model(train_x, train_y):
    model = SingleTaskGP(train_x, train_y)
    model.likelihood.noise = torch.tensor(1e-4)
    model.covar_module.base_kernel.lengthscale = torch.tensor(0.1)
    model.covar_module.outputscale = torch.tensor(0.05)

    return model
```

The optimisation loop: here 5 steps

```python
from botorch.acquisition import ExpectedImprovement
from botorch.optim import optimize_acqf

for i in range(5):
    # create model
    model = create_model(train_x, train_y)

    # create the acquisition function using the model
    acq = ExpectedImprovement(model, train_y.max())

    # optimize the acquisition function
    candidate,_ = optimize_acqf(
        acq,
        bounds=torch.tensor([0.0,1.0]).reshape(2,1),
        q=1,
        num_restarts=5,
        raw_samples=20
    )

    train_x = torch.cat([train_x, candidate])
    train_y = ground_truth(train_x)
```

References

Gaussian Processes for Machine Learning, C.E. Rasmussen and C.K.I. Williams, MIT press, 2006.

Bayesian Optimization, R. Garnett, Cambridge University Press, 2023.

Bayesian Optimisation with Inequality Constraints, J. R. Gardener et al, ICML 2014.

Proximal Biasing for Bayesian Optimisation and Characterisation of Physical Systems, R. Roussel and A. Edelen, NeurIPS 2021.