Bayesian algorithms for practical accelerator control

Ryan Roussel

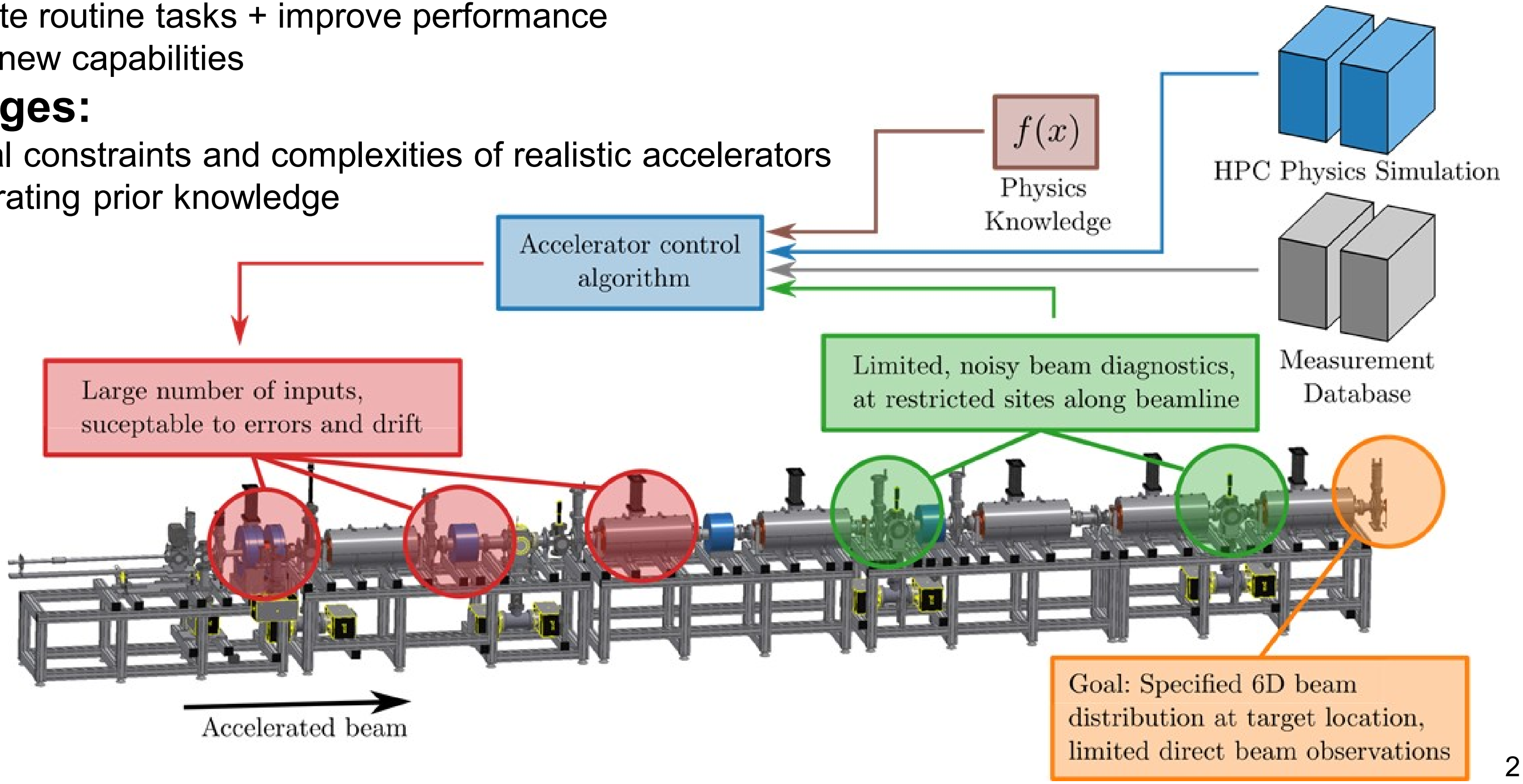
11/1/2022

rroussel@slac.stanford.edu

Machine Learning Based Accelerator Control

Goals:
- Automate routine tasks + improve performance
- Enable new capabilities

Challenges:
- Practical constraints and complexities of realistic accelerators
- Incorporating prior knowledge
- Scaling



Global Optimization Strategies

Can you calculate gradients easily?

- Analytical models

Yes

No

- Simulations
- Experiments

Gradient descent (SGD, Adam etc.)

Black box optimization algorithms

Scales to >10k parameters (see ML training)

Scales poorly with input dimension if not unimodal

Go ahead, try it with simplex…

Model Based Optimization of Accelerators

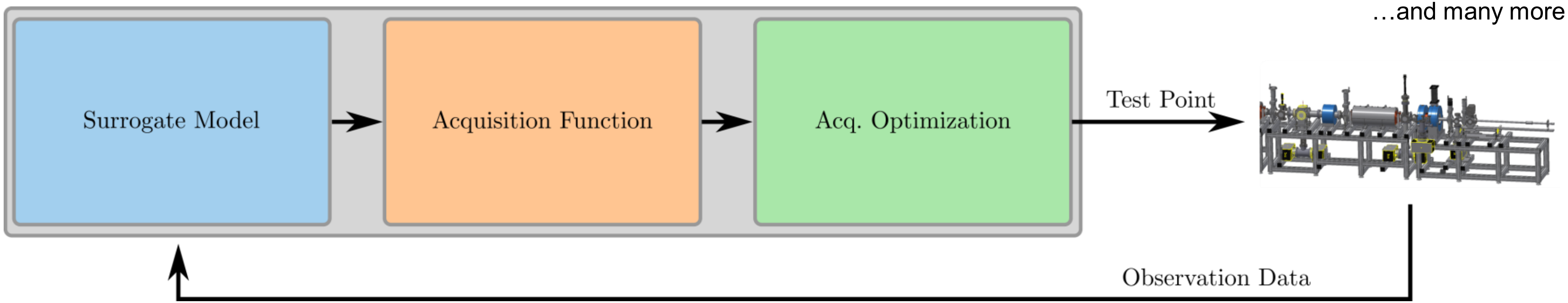
Can we improve black box optimization with surrogate models? **YES**

Create a computational model of the system

Determine the value of potential future measurements

Pick the point that maximizes value

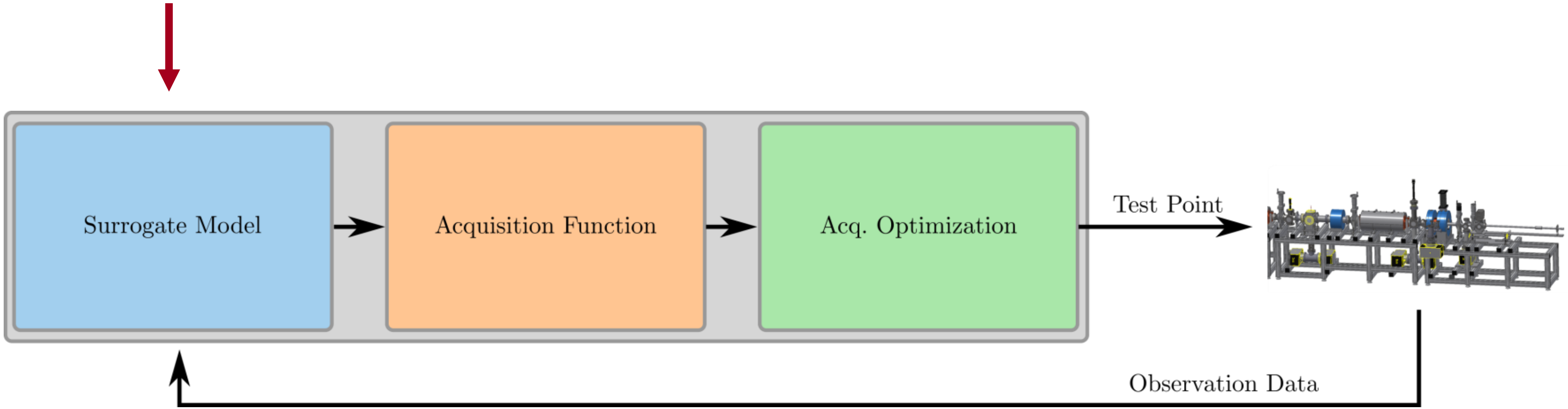
Duris et al. PRL 2020
Roussel et al. PRAB 2021
Hanuka et al. PRAB 2021
Miskovich et al. PRAB 2021
…and many more



… **but** we must consider the cost of creating models with enough information

Model Based Optimization of Accelerators

Gaussian processes (GPs)



Why?

- Extracts a lot of information from a small number of data points → efficient
- Produces explicit uncertainty estimations -> advantageous for global optimization

Bayesian Statistics: A Practical Example

likelihood prior

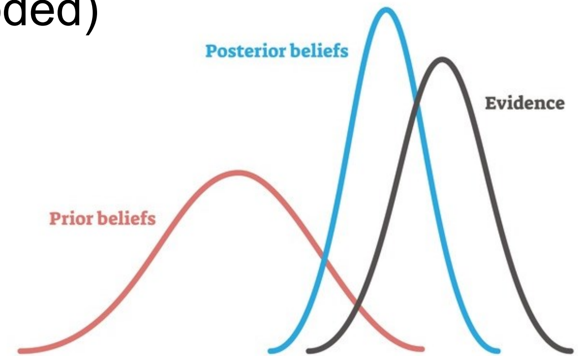
posterior

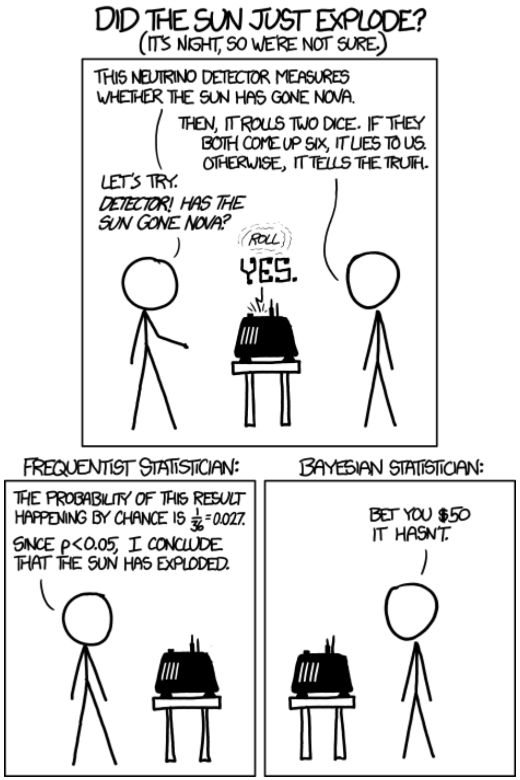
$$p(B|A) = \frac{p(A|B)p(B)}{p(A)}$$

marginal likelihood

A: our measurement
(we measure that the sun has/has not exploded)

B: our prediction
(the sun has/has not exploded)





https://xkcd.com/1132/

Bayesian Statistics: A (More) Practical Example

Model:

Gaussian noise

$$y = w_0 + w_1 x + \epsilon$$

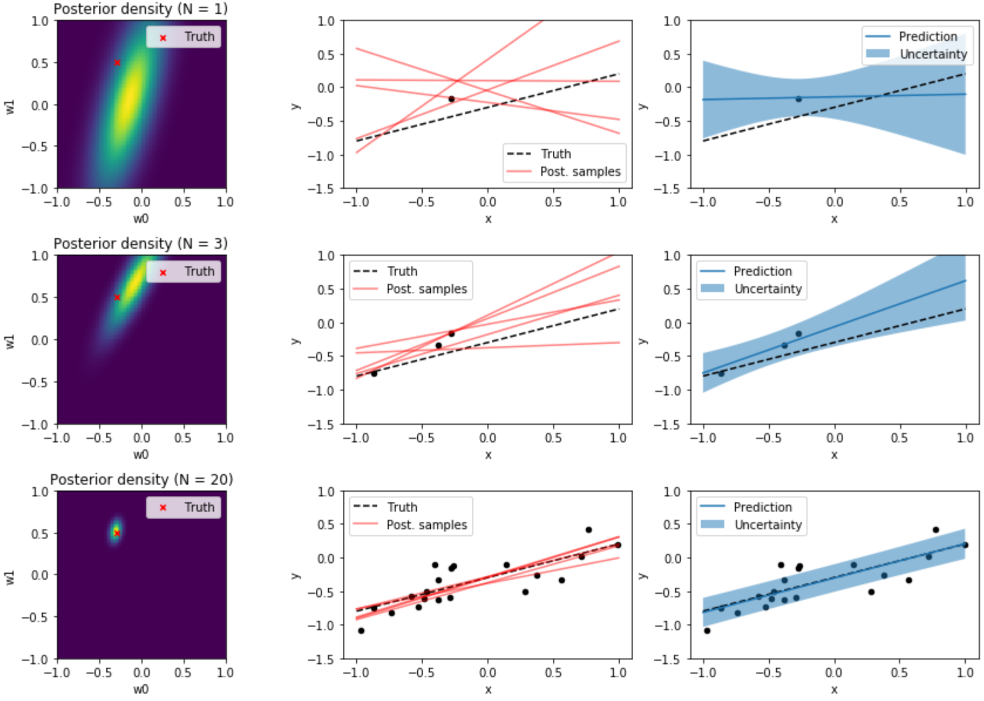
Bayesian regression (determining the weights):

$$p(w_0, w_1 | \boldsymbol{y}_N, \boldsymbol{x}_N) \propto p(\boldsymbol{y}_N | w_0, w_1, \boldsymbol{x}_N) p(w_0, w_1)$$

Making predictions:

$$p(\boldsymbol{y}_M | w_0, w_1, \boldsymbol{x}_M)$$

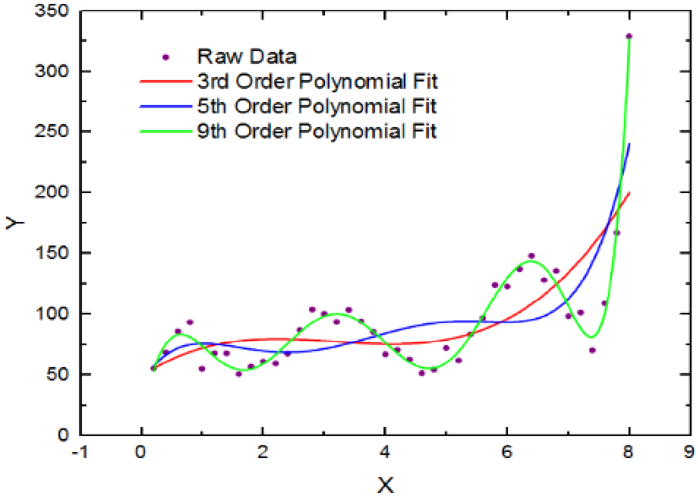
Note: least squares fitting is equivalent to using a uniform prior and Gaussian likelihood



M. Krasser. https://nbviewer.org/github/krasserm/bayesian-machine-learning/blob/dev/bayesian-linear-regression/bayesian_linear_regression.ipynb

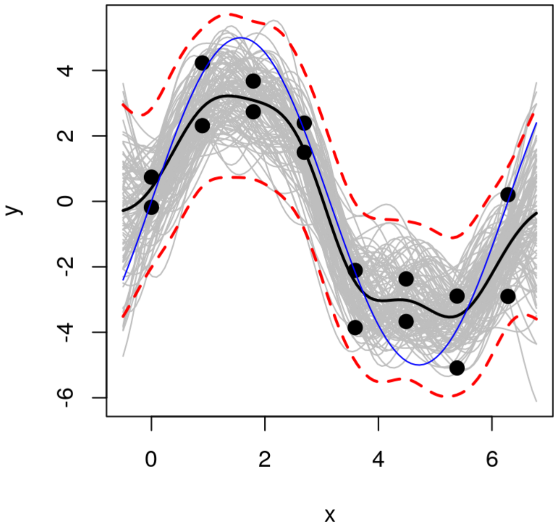
Parametric vs. Non-Parametric Modeling

Parametric modeling



$$f(x) = f(x;\theta)$$

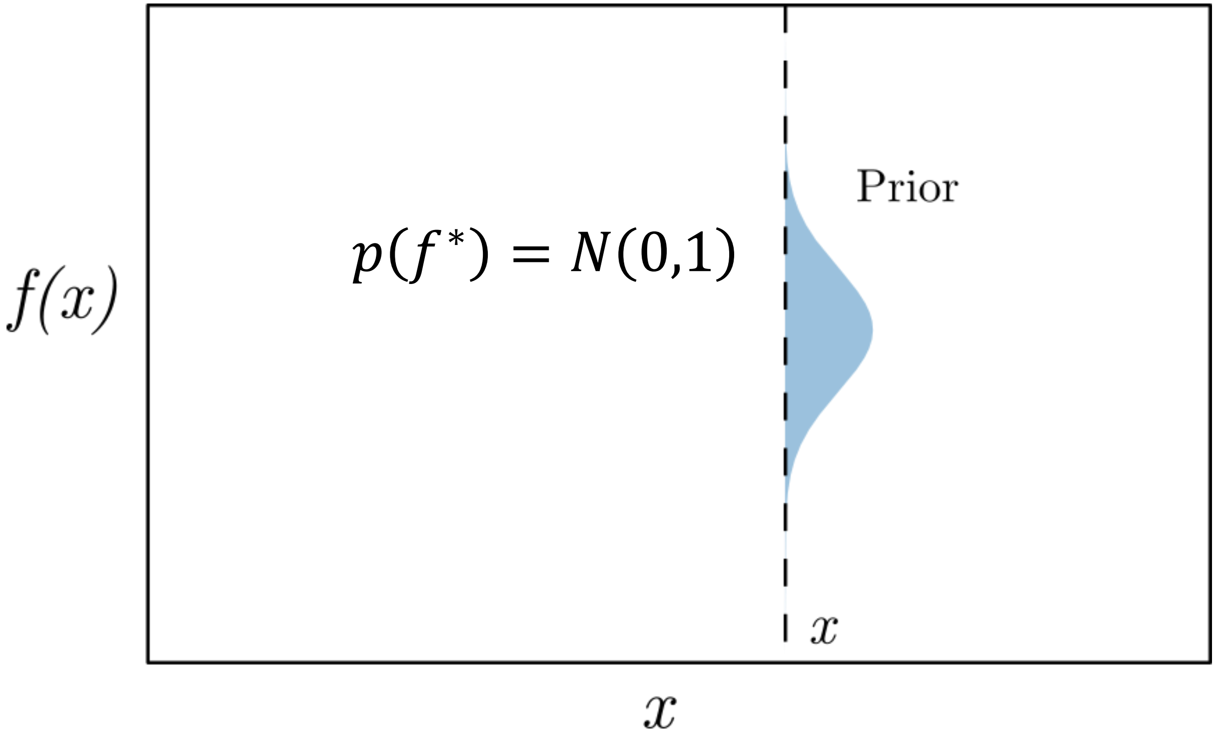
Non-Parametric modeling



$$f = \{\theta_0(x_0), \theta_1(x_1), \dots, \theta_N(x_N)\}$$

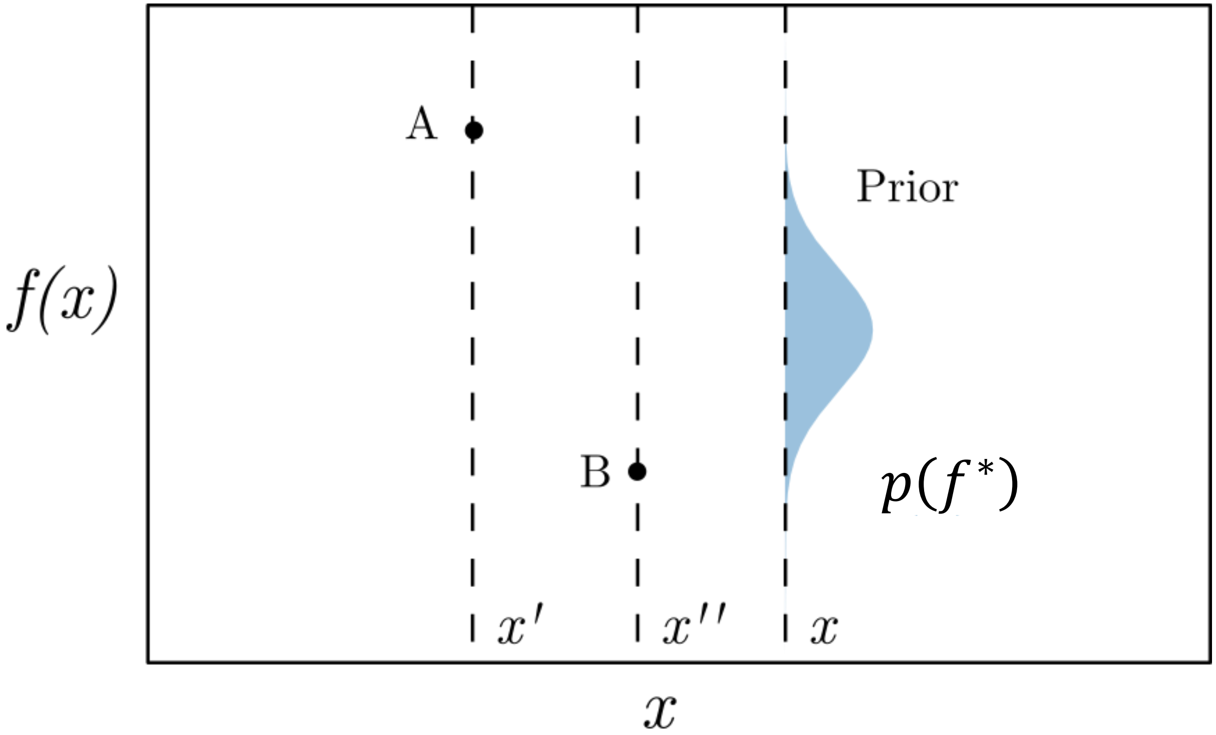
Some intuition…

Let's predict the function value f^* at the point x



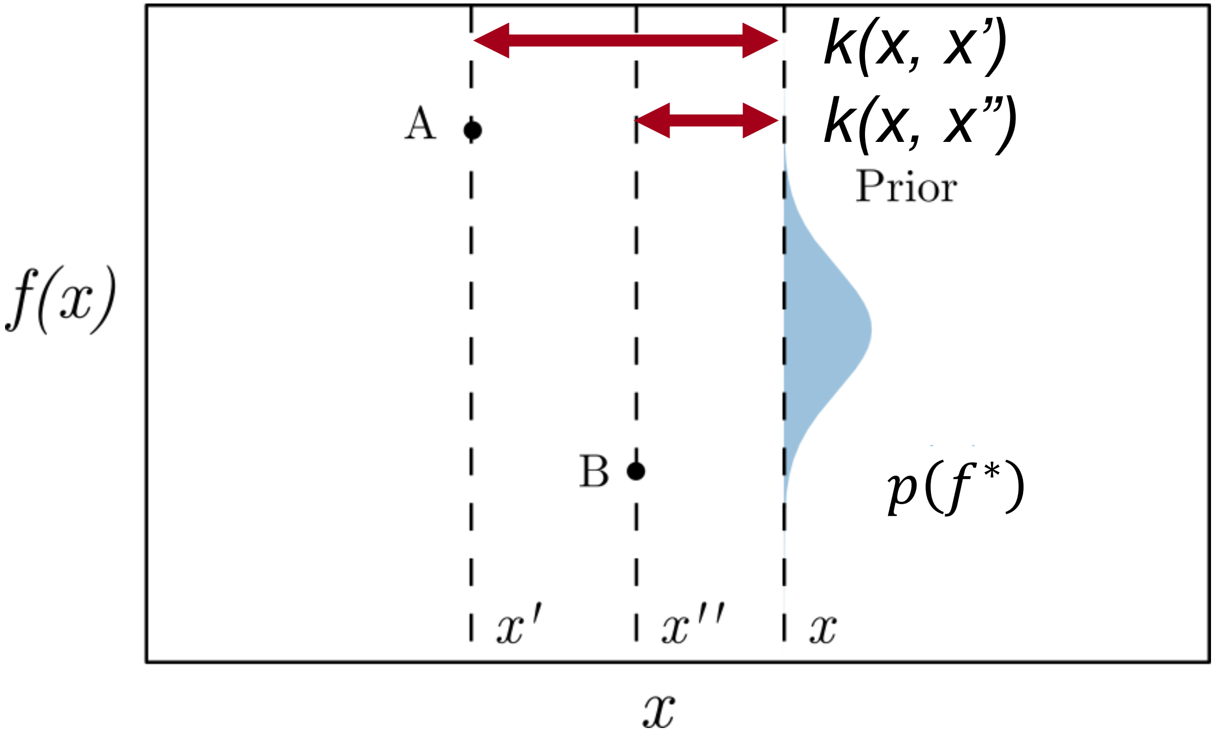
Some intuition…

Which observation will have a larger impact on changing *p(f)*?



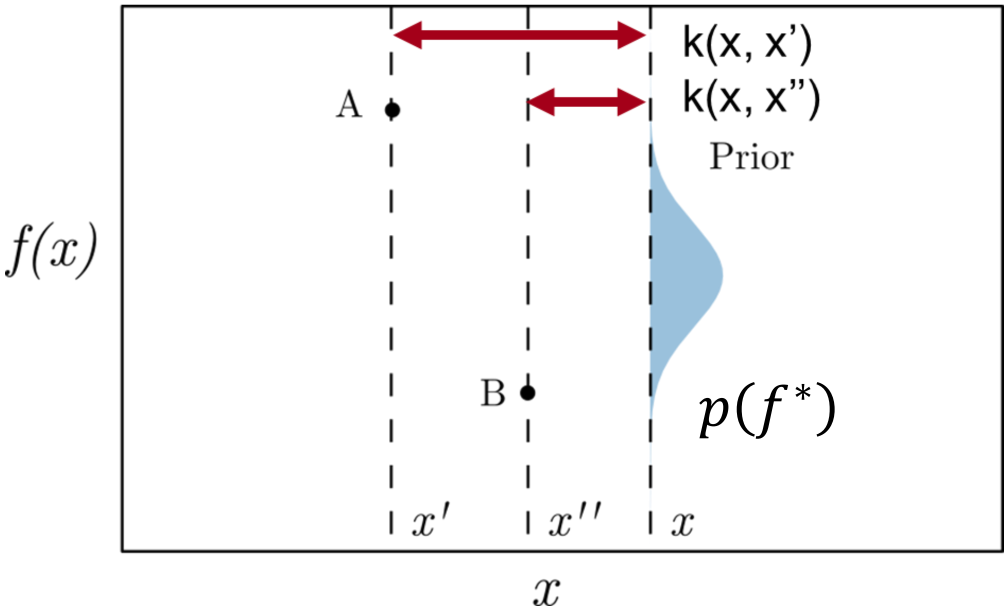
Adding some math

Which observation will have a larger impact on changing *p(f)*?



$k(x, x') < k(x, x'')$

Adding some math



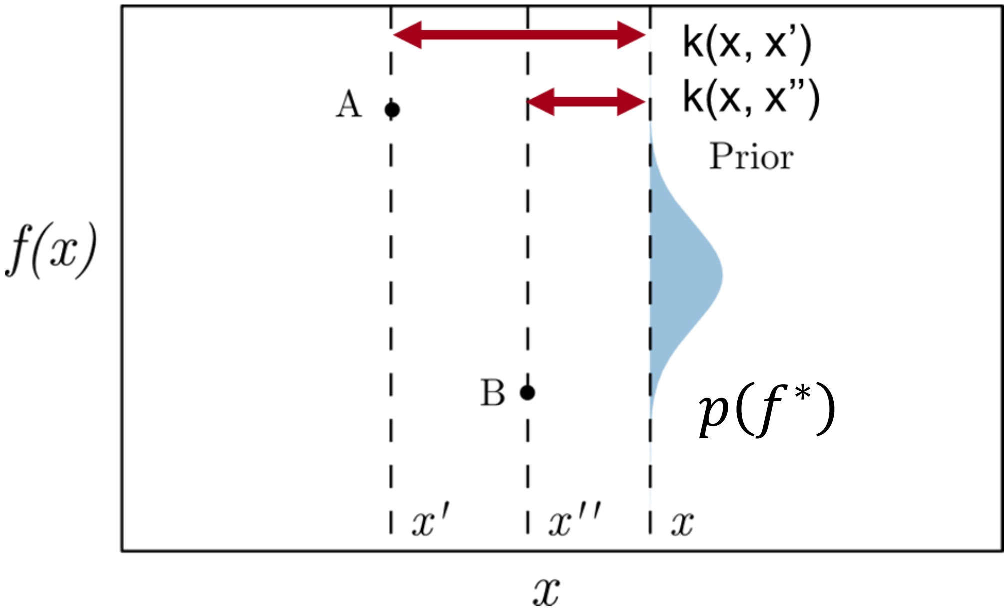
$$p(f_A, f_B, f^*) = N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma} = \begin{pmatrix} k(x',x') & k(x',x'') & k(x',x) \\ k(x',x'') & k(x'',x'') & k(x'',x) \\ k(x',x) & k(x'',x) & k(x,x) \end{pmatrix}$$

$$p(f^* | f_A, f_B) = \frac{p(f_A, f_B | f^*) p(f^*)}{p(f_A, f_B)} = \frac{p(f_A, f_B, f^*)}{p(f_A, f_B)}$$

Bayes rule

Adding some math



$$p(f^* | f_A, f_B) = N(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)$$

$$\boldsymbol{\mu}^* = \boldsymbol{\mu} + K^* K^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$$
$$\boldsymbol{\sigma}^* = K^{**} - K^{*T} K^{-1} K^*$$

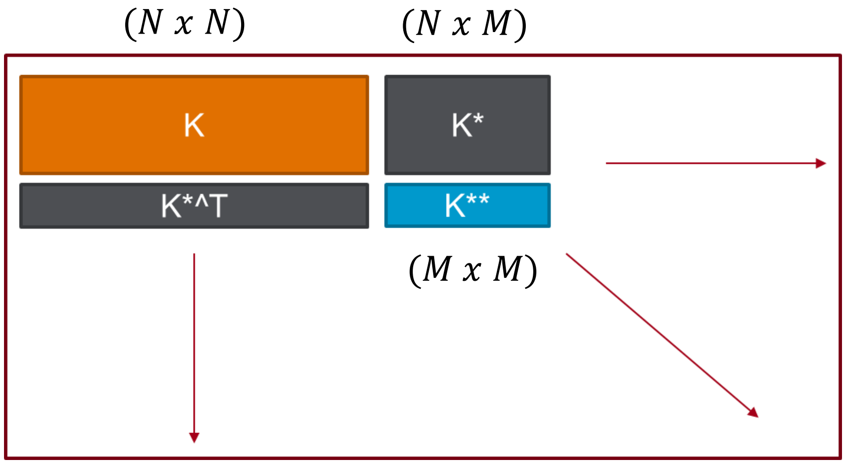
$$p(f^* | f_A, f_B) = \frac{p(f_A, f_B | f^*) p(f^*)}{p(f_A, f_B)} = \frac{p(f_A, f_B, f^*)}{p(f_A, f_B)}$$

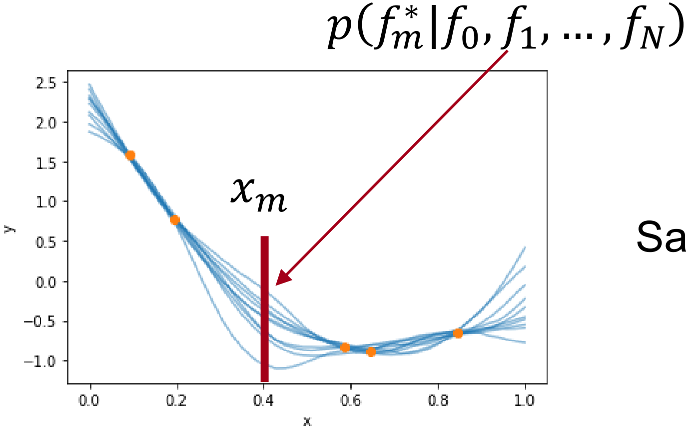
$K^{-1} \sim \mathcal{O}(N^3)$!

Making predictions with GP's

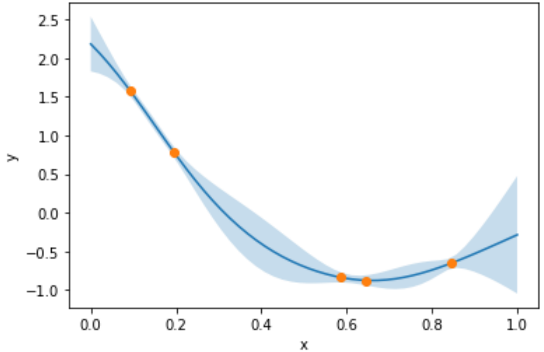
What about multiple predictions?

$$p(f_0^*, f_1^*, \dots, f_M^* | f_0, f_1, \dots, f_N) = N(\boldsymbol{\mu}^*, \boldsymbol{\sigma}^*)$$





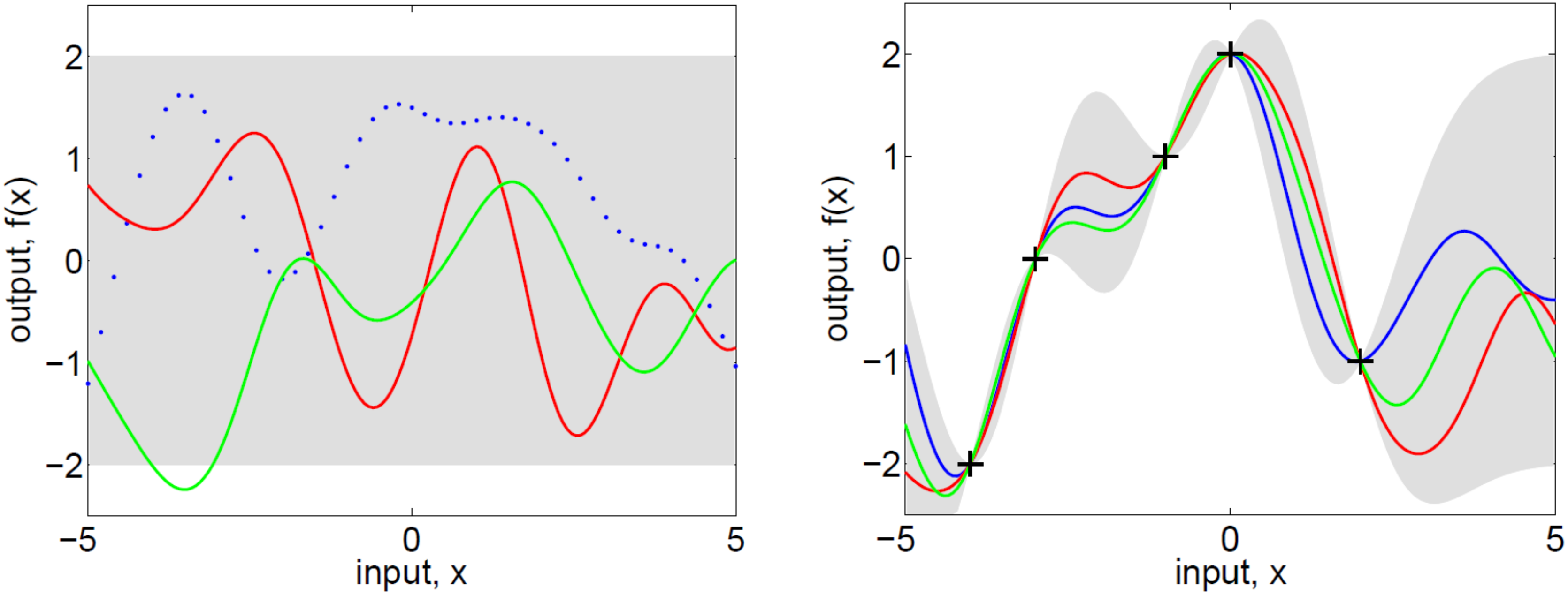
Samples



Distribution

Draw function samples? Sample from the joint posterior distribution at requested points

Another perspective



(a), prior (b), posterior

Rasmussen and Williams. 2006

Fitting Gaussian Processes to Data

Need to determine the form and hyperparameters of the kernel

Each kernel has hyperparameters that control the overall function behavior.

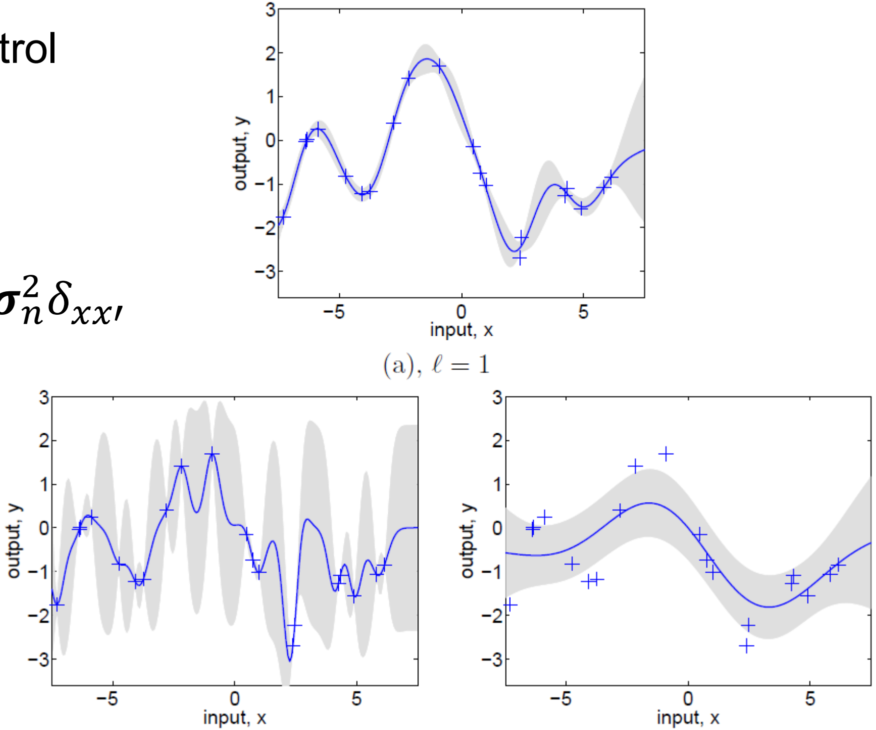
Radial Basis Function:

$$k(x, x') = \boldsymbol{\sigma_f^2} \exp\left(-\frac{1}{2\boldsymbol{l}^2}(x - x')^2\right) + \boldsymbol{\sigma_n^2}\delta_{xx'}$$

Kernel amplitude

Kernel length scale

Noise



(a), $\ell = 1$

(b), $\ell = 0.3$

(c), $\ell = 3$

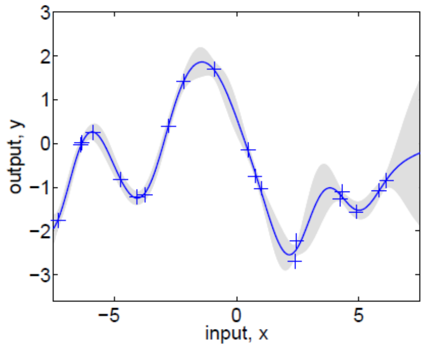
R&W

Fitting Gaussian Processes to Data

Need to determine the form and hyperparameters of the kernel

We fit kernel parameters to data by using Maximum Likelihood Estimation (MLE or MLE-II).

$$\boldsymbol{\theta}^* = \mathrm{argmax}_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{X})$$



(a), $\ell = 1$

With a Gaussian likelihood the log likelihood can be calculated analytically:

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

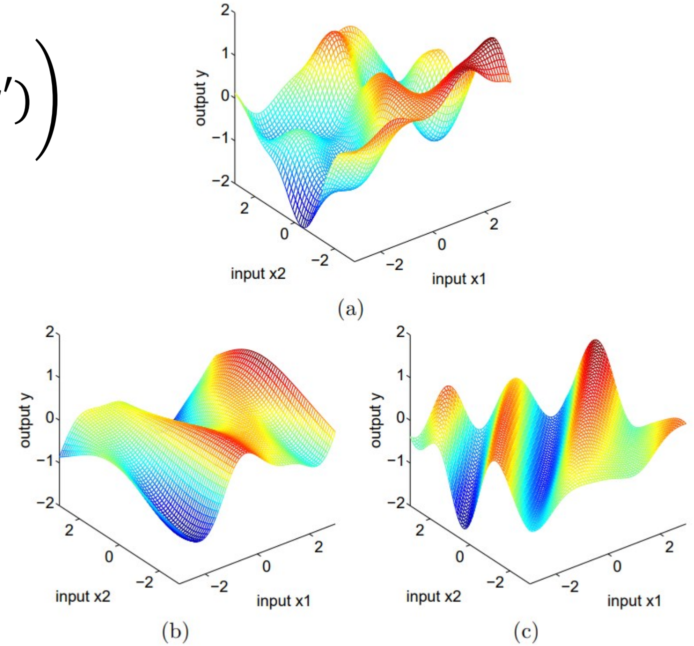
Predictive accuracy

Model complexity

More Expressive Kernels

We can also encode low dimensional structure into the kernel.

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^T \boldsymbol{\Sigma} (\boldsymbol{x} - \boldsymbol{x}')\right)$$



Automatic relevance determination

$$\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{l})^{-2}$$

Factor analysis distance

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \mathrm{diag}(\boldsymbol{l})^{-2}$$

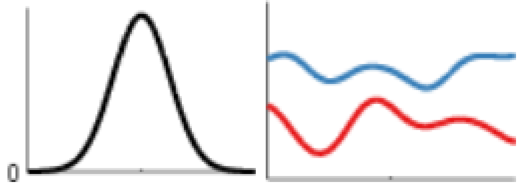
Rasmussen and Williams. 2006

Fitting Gaussian Processes to Data

Need to determine the form and hyperparameters of the kernel

Radial Basis Function:

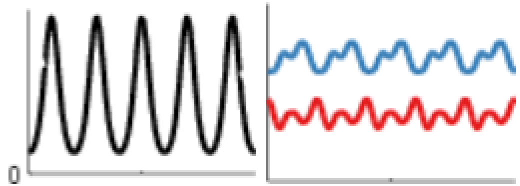
$$k_{\mathrm{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$$



Stationary

Periodic:

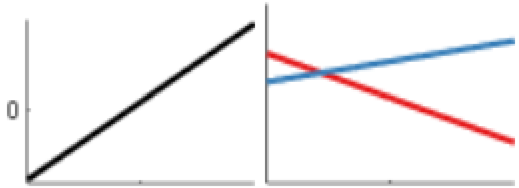
$$k_{\mathrm{Per}}(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x-x'|/p)}{\ell^2}\right)$$



Stationary

Linear:

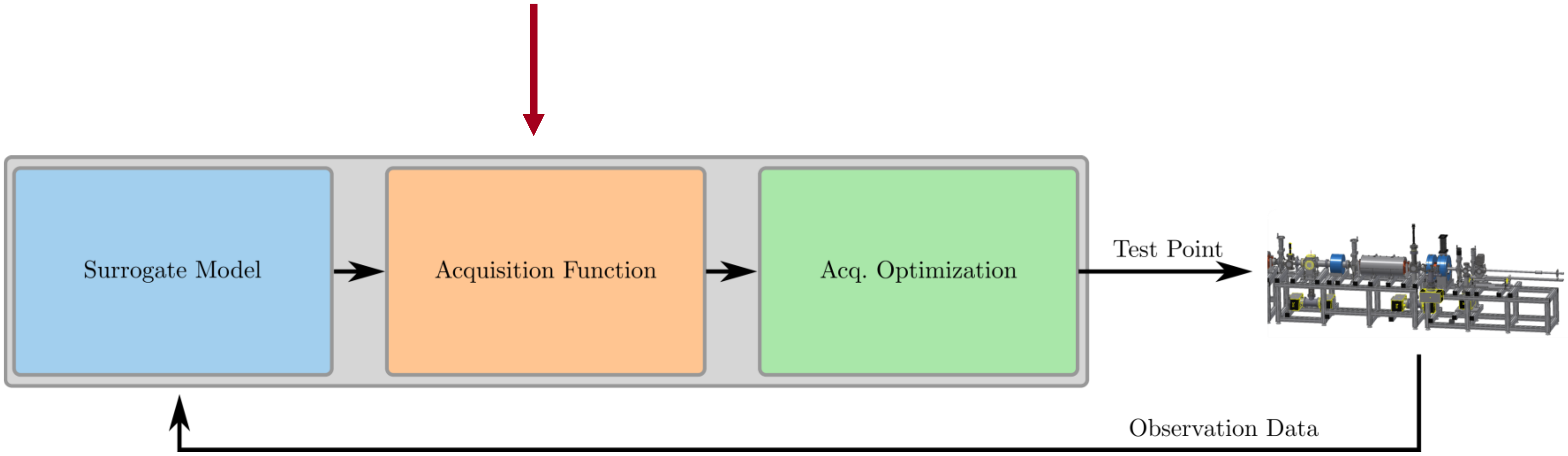
$$k_{\mathrm{Lin}}(x, x') = \sigma_b^2 + \sigma_v^2(x-c)(x'-c)$$



Non-stationary

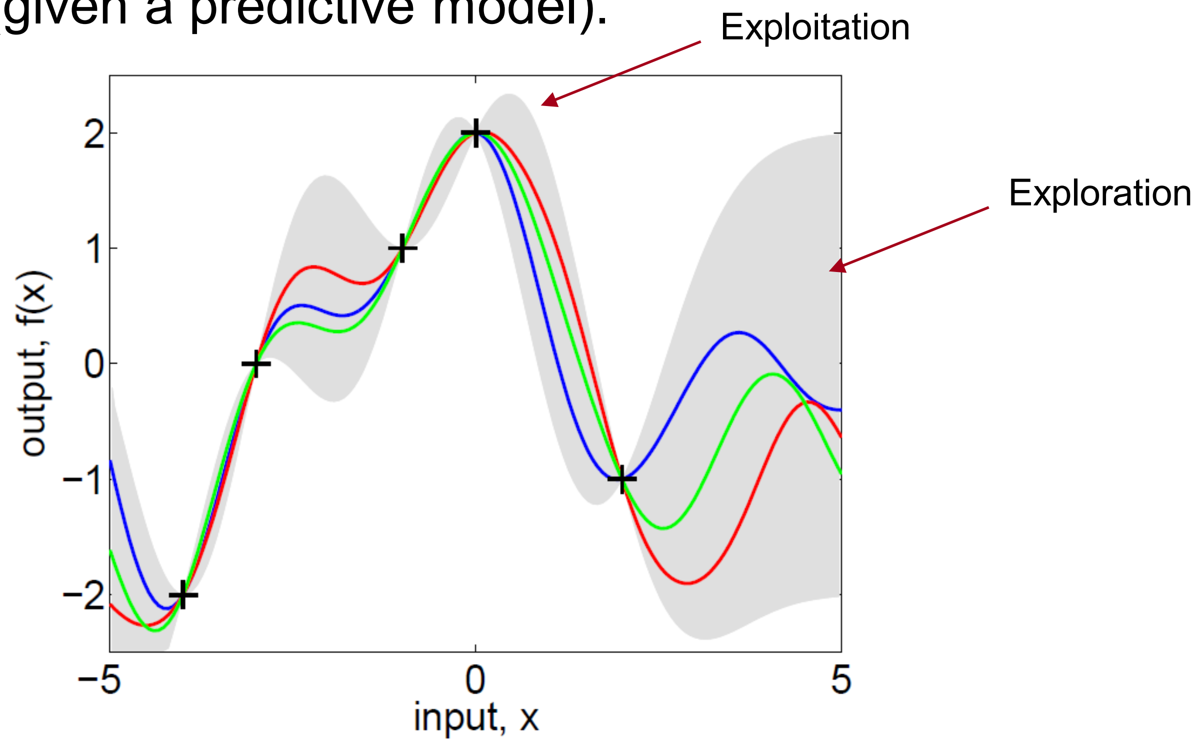
D. Duvenaud, https://www.cs.toronto.edu/~duvenaud/cookbook/

Model Based Optimization of Accelerators



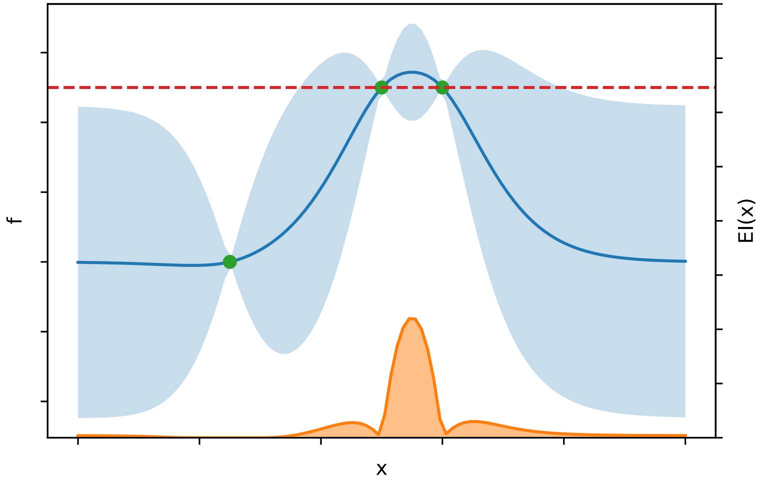
The Acquisition Function

Define a function that characterizes the value of making a potential measurement (given a predictive model).



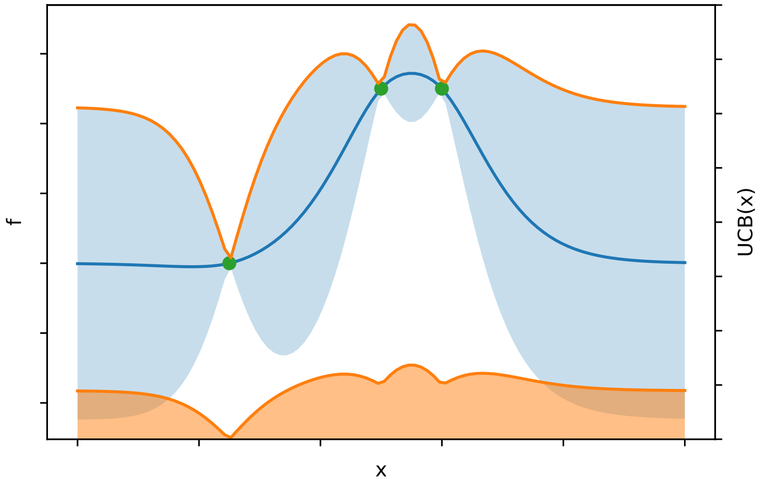
Single Objective Optimization

Expected Improvement



$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}[\max(f(\boldsymbol{x}) - f^*)]$$

Upper Confidence Bound



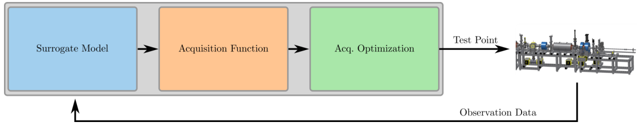
$$\mathrm{UCB}(\boldsymbol{x}) = \mu(\boldsymbol{x}) + \sqrt{\beta}\sigma(\boldsymbol{x})$$

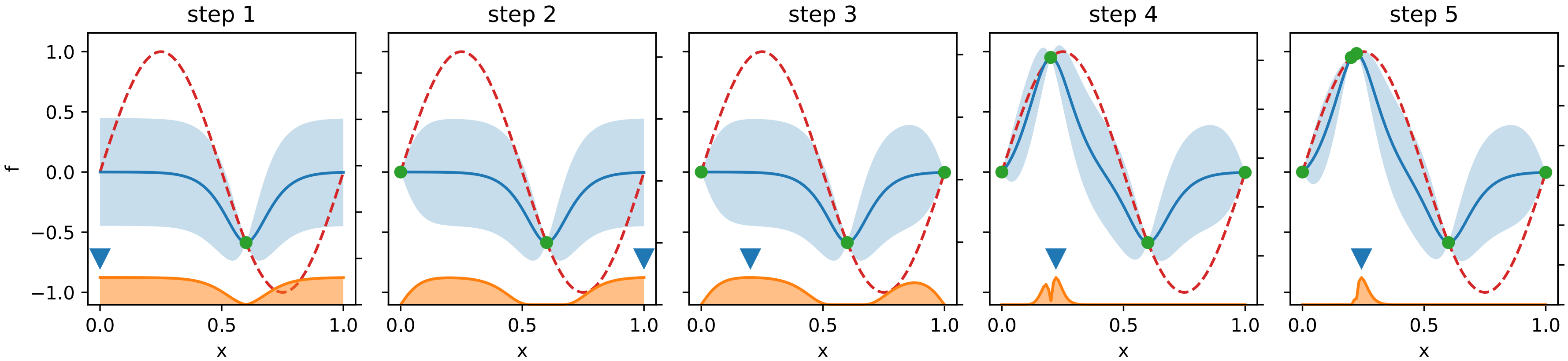
Note: most implementations of BO assume maximization

Single Objective Optimization

$$\mathrm{EI}(\boldsymbol{x}) = \mathbb{E}[\max(f(\boldsymbol{x}) - f^*)]$$

$$\boldsymbol{x}_{t+1} = \mathrm{argmax}_{\boldsymbol{x}} \mathrm{EI}(\boldsymbol{x})$$



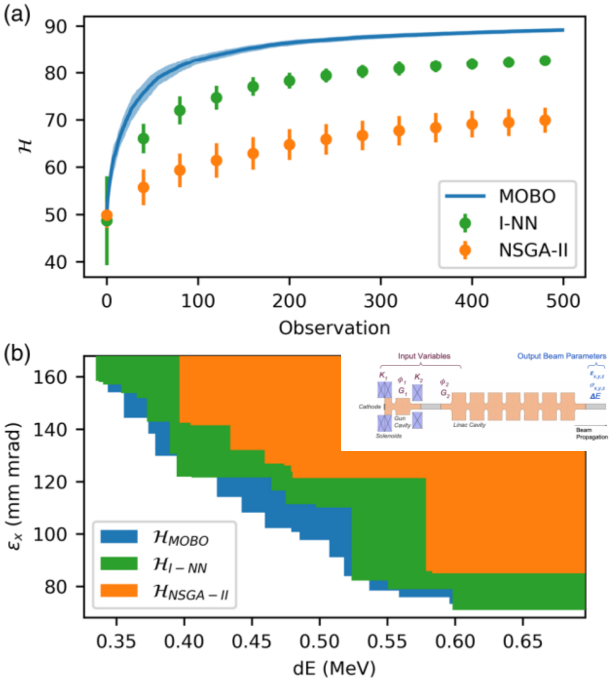


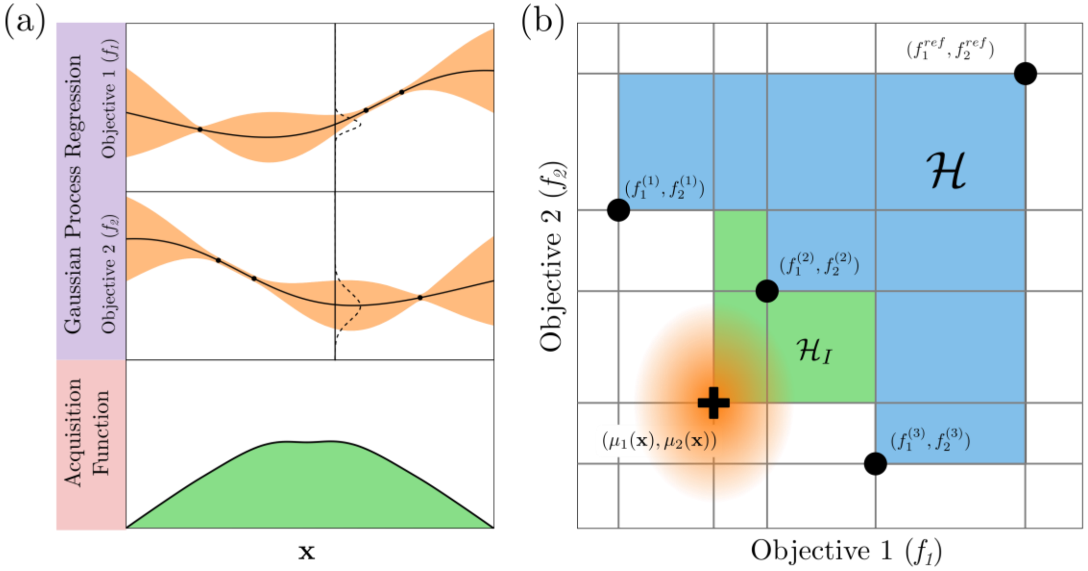
Some notes:

- The model accuracy improves in the region of interest!
- Initially the model uncertainty is maximized at the domain boundaries, BO likes to sample those
- Helpful if the acquisition function is differentiable → use gradient descent to optimize

Multi-Objective Optimization

Determine the optimal trade-off between objectives -> the Pareto front





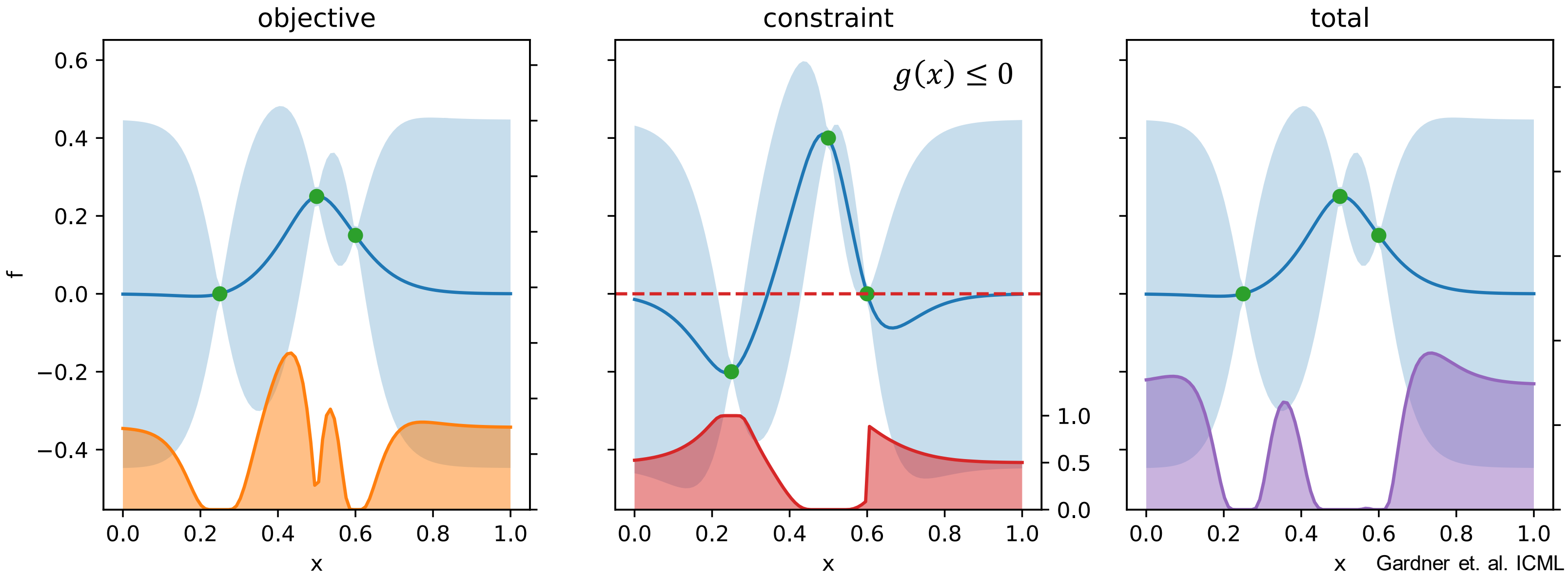
$$\alpha_{EHVI}(\boldsymbol{\mu},\boldsymbol{\sigma},\mathcal{P},\boldsymbol{r}) := \int_{\mathbb{R}^P} \mathrm{HVI}(\mathcal{P},\boldsymbol{y},\boldsymbol{r}) \cdot \boldsymbol{\xi}_{\mu,\boldsymbol{\sigma}}(\boldsymbol{y}) d\boldsymbol{y}$$

Roussel et. al. PRAB **2021**

Incorporating Constraints

Weight the acquisition function by the probability that constraints are satisfied

$$\hat{\alpha}(x) \rightarrow \alpha(x) \prod_i p[g_i(x) \leq h_i]$$ Warning: Requires $\alpha(x) \geq 0$

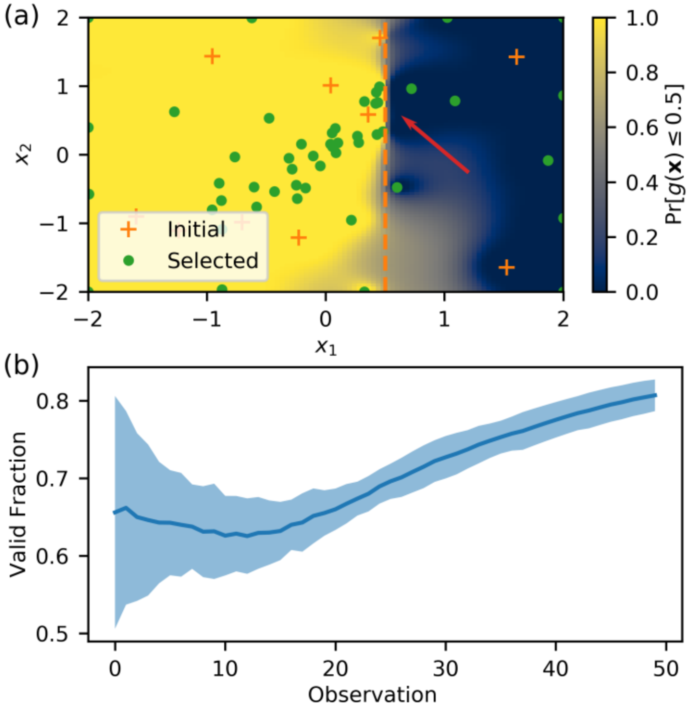


Gardner et. al. ICML 2014

Incorporating Constraints

Weight the acquisition function by the probability that constraints are satisfied

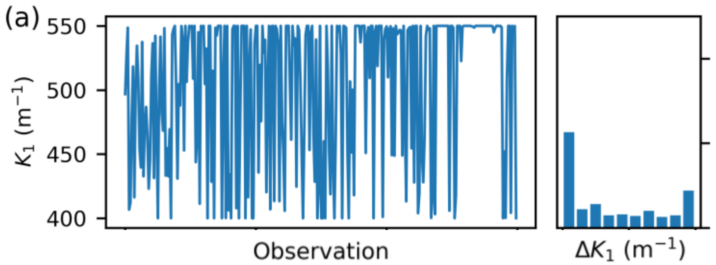
$$\alpha(x) \to \alpha(x) \prod_i p[g_i(x) \leq h_i]$$



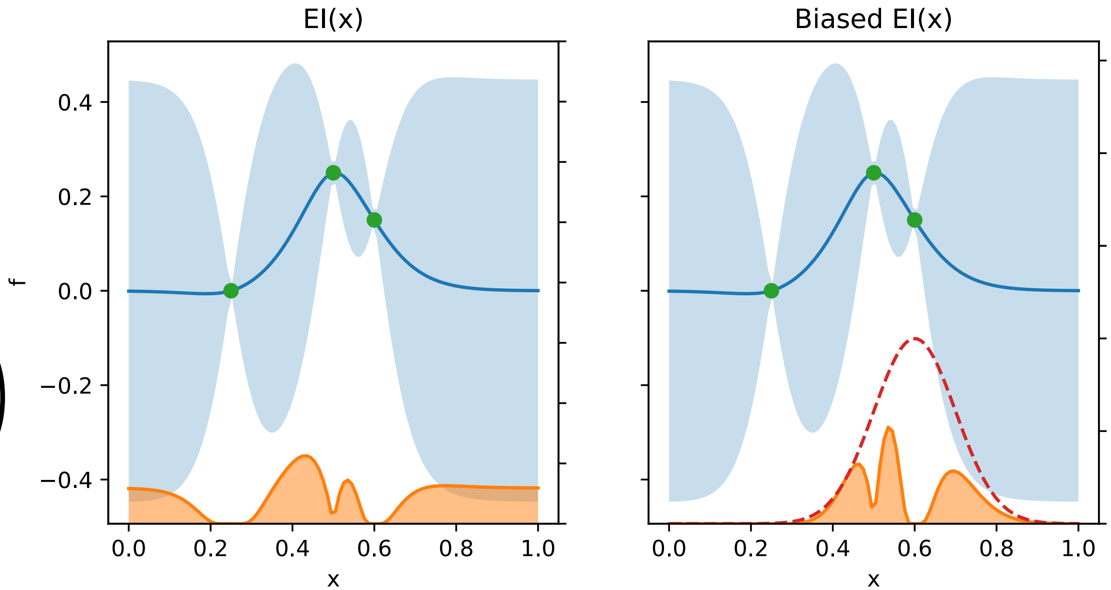
Roussel et. al. PRAB 2021

Proximal Biasing

Poor optimization behavior for experimental beamlines



Weight the acquisition function by travel distance → better than hard limits



$$\hat{\alpha}(x) \to \alpha(x) \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right)$$

Warning: Requires $\alpha(x) \geq 0$

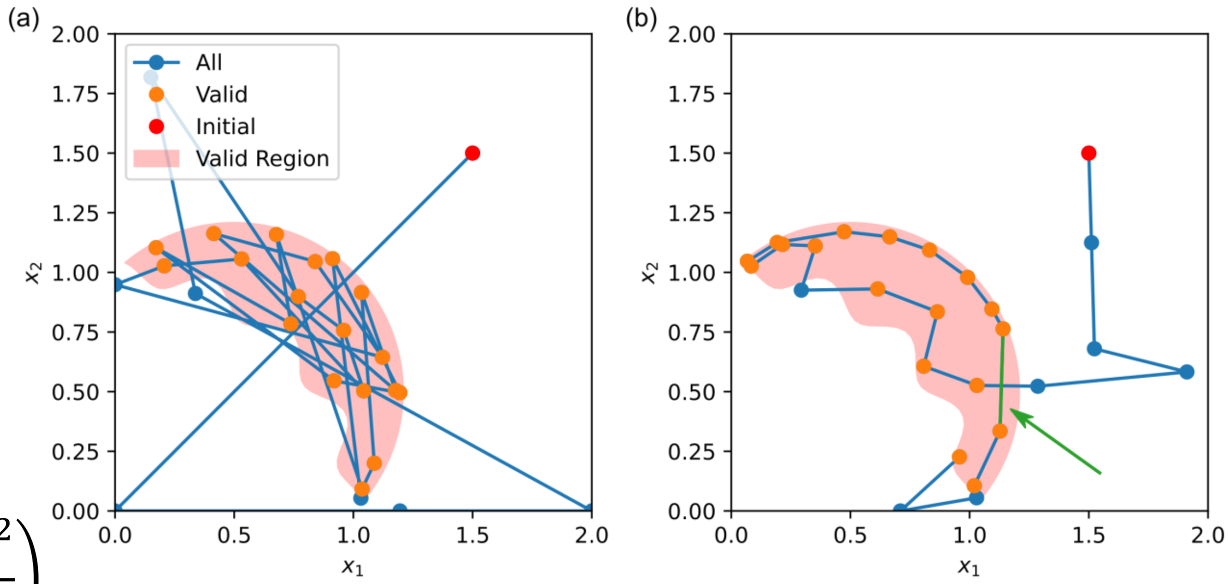
Proximal Biasing

Poor optimization behavior for experimental beamlines

Weight the acquisition function by travel distance → better than hard limits

$$\hat{\alpha}(x) \rightarrow \alpha(x) \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right)$$

Warning: Requires $\alpha(x) \geq 0$



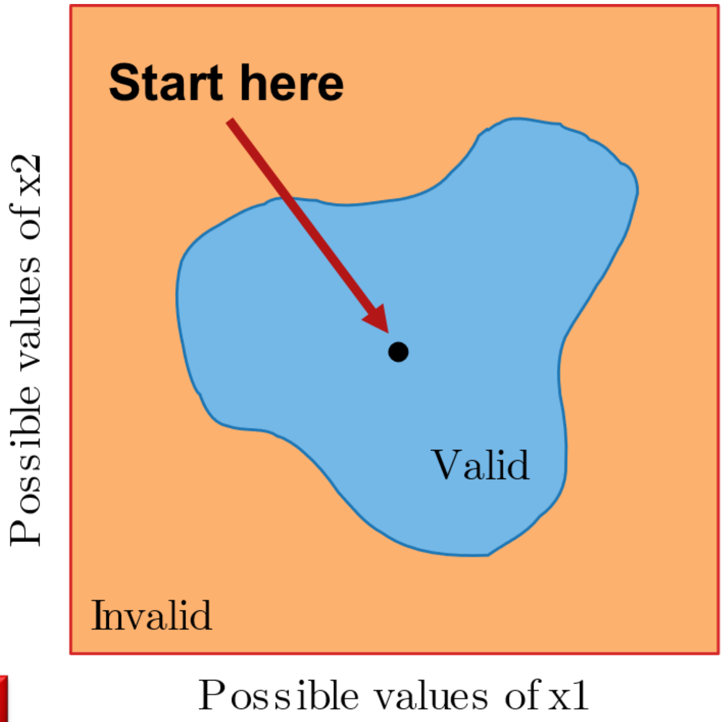
Roussel et. Al. *Nat. Comm.* **2021**

Case study: Bayesian Exploration

Autonomous characterization of novel systems.

- Starts with a single valid observation AND **no prior information except for hardware limitations**
- Samples points in a quasi-uniform grid where the grid spacing is learned automatically based on the beam response
- Learns where the valid region is w/limited invalid observations
- Considers costs associated with changing parameters
- Natively applicable to multi-dimensional exploration
- General purpose that works in almost any case

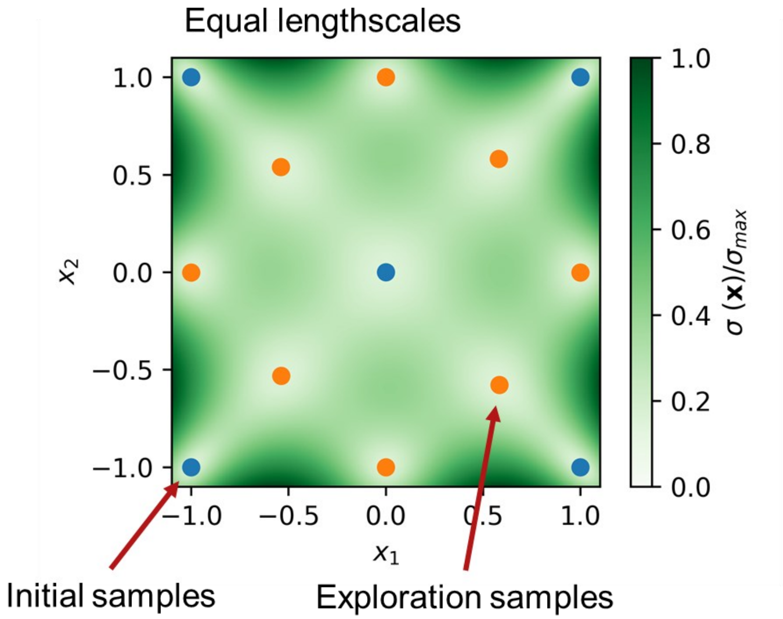
Click button → **GO!**

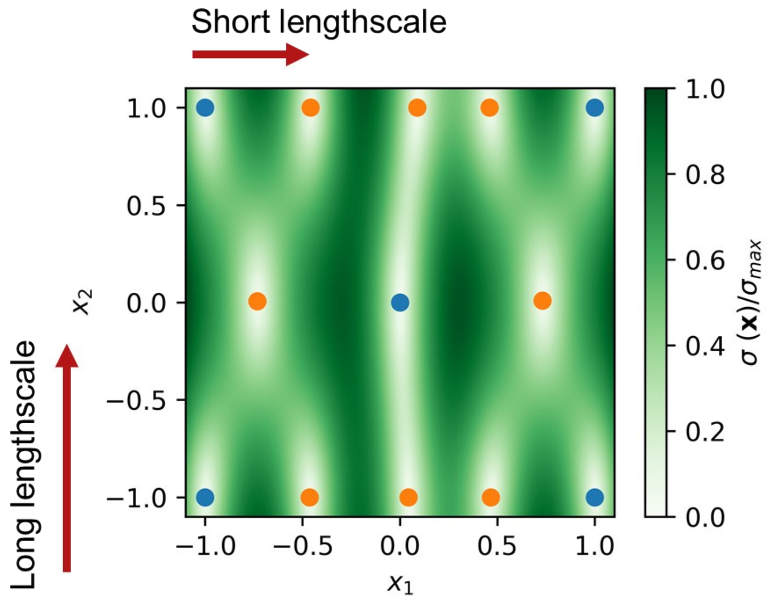


Uncertainty Sampling

If the function changes more rapidly along one axis, sample more points along that axis!

$$\alpha(\boldsymbol{x}) = \sigma(\boldsymbol{x})$$

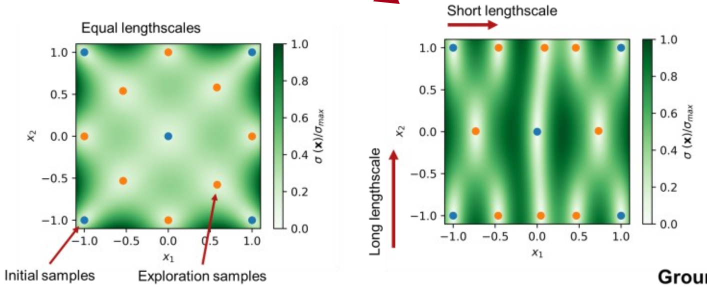




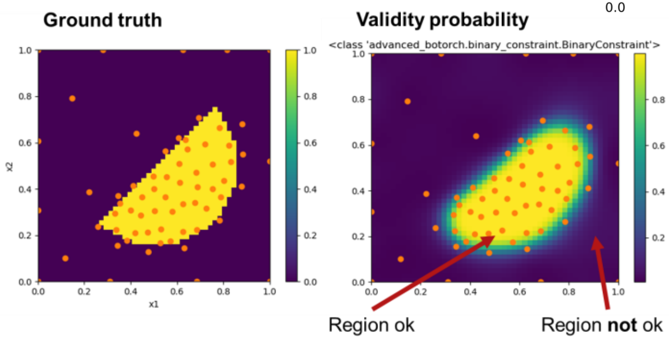
Bayesian Exploration

$$\alpha(\boldsymbol{x}) = \sigma(\boldsymbol{x}) \prod_{i=1}^{N} p(g_i(\boldsymbol{x}) \geq h_i)\, \Psi(\boldsymbol{x}, \boldsymbol{x_0})$$

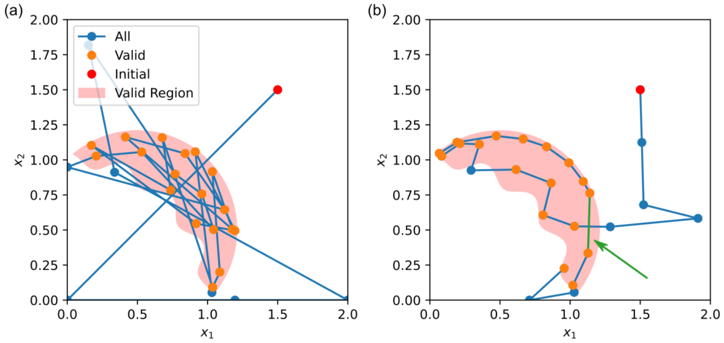
Adaptive sampling



Unknown constraints

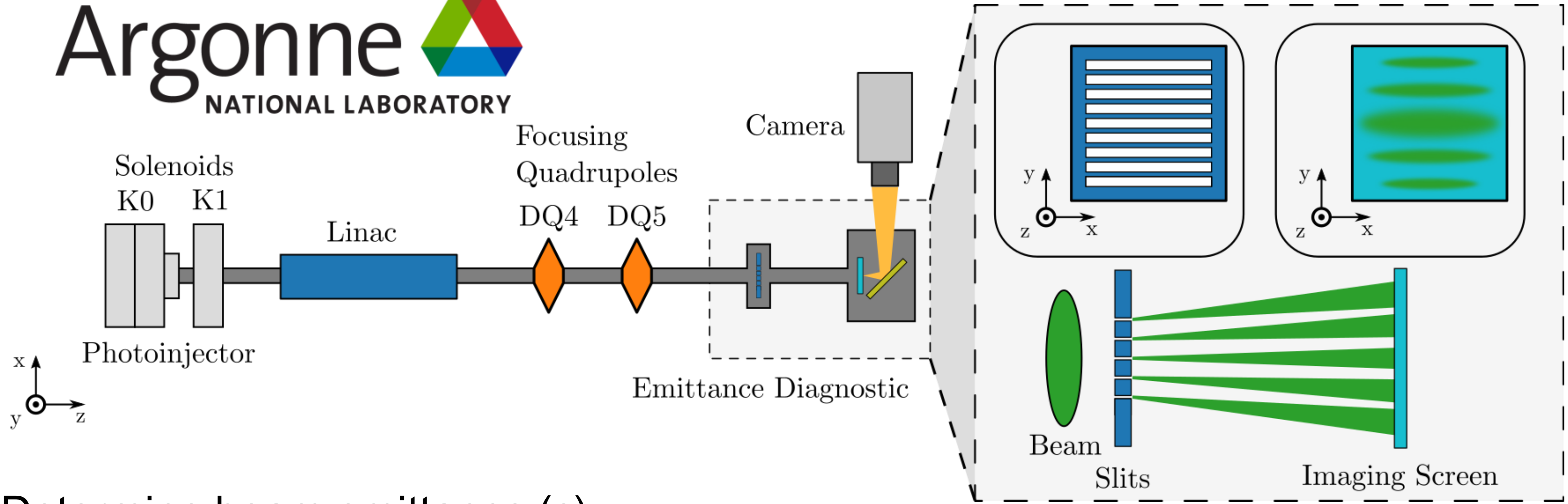


Proximal biasing



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

Characterizing Photoinjector Emittance at AWA

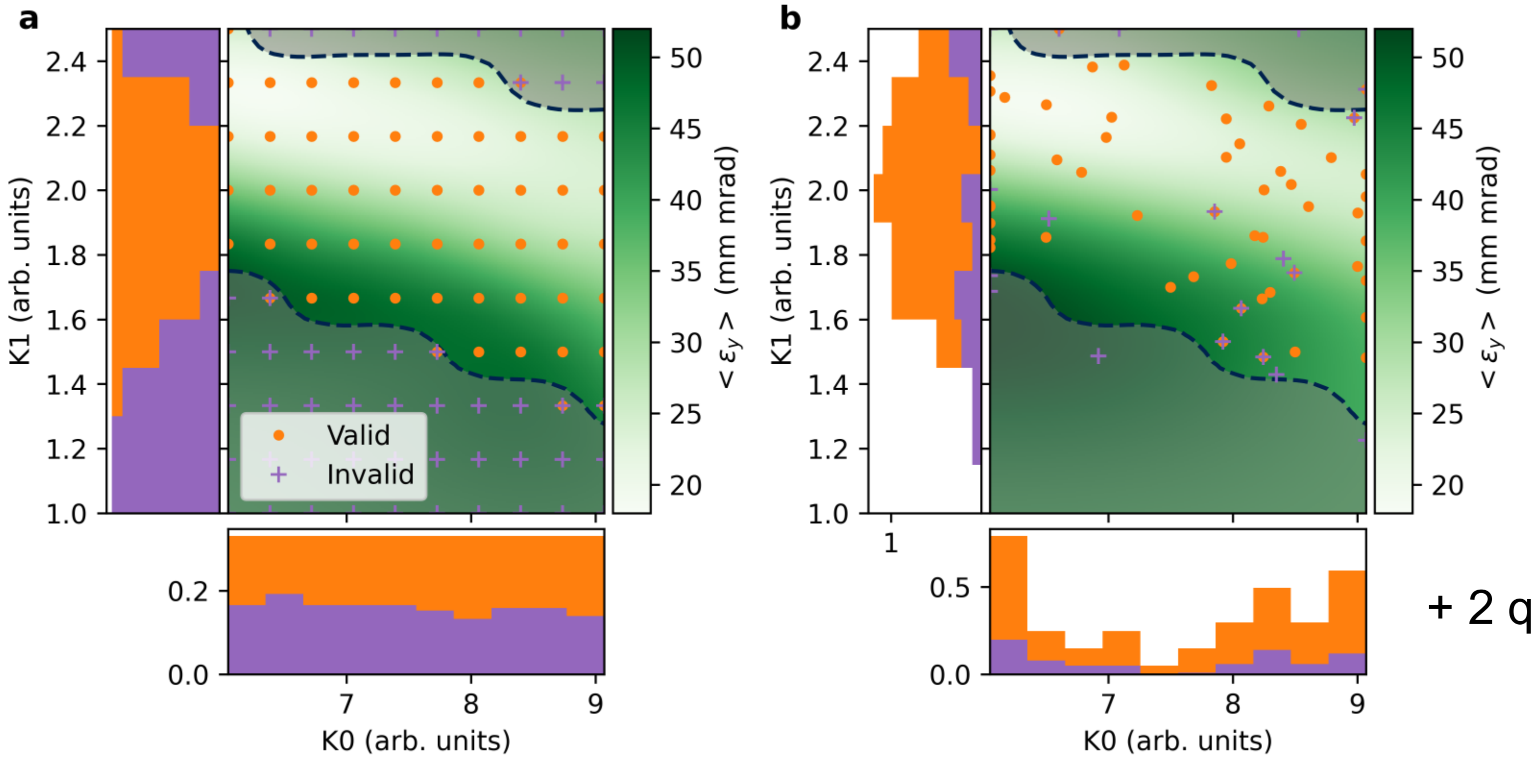


Determine beam emittance (ε) as a function of:

- 2 solenoids
- 2 quadrupoles

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

Characterizing Photoinjector Emittance at AWA

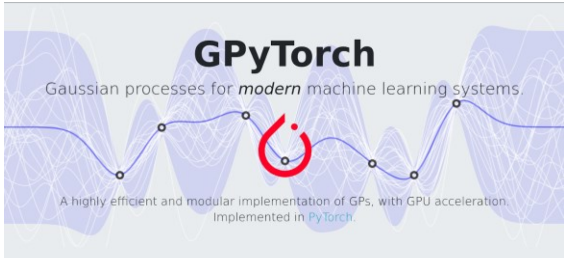


+ 2 quads!

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

Implementing Bayesian Optimization

PyTorch → GPyTorch → BoTorch



BoTorch

BAYESIAN OPTIMIZATION IN PYTORCH

Use Botorch

- Single/multi-objective Bayesian optimization (serial and parallel)
- Constraints
- Proximal biasing
- MC Acquisition function evaluation
- Flexible incorporation of pyTorch Modules
- GPU support

See Examples: https://github.com/slaclab/bo_tutorial

Xopt

See afternoon talk/poster!

Final thoughts

- Bayesian optimization is at its best when evaluating objective functions is **expensive** (incl. simulations)!
- Need to **balance** the costs of creating/evaluating the model vs. your application
- Take advantage of **prior information** about the objective function to speed up optimization (see NN prior poster, Nikita talk)
- Be **creative** with your acquisition functions to suit your optimization needs

Resources

SLAC

- Gaussian Processes for Machine Learning (R & W) - https://gaussianprocess.org/gpml/
- Greenhill, Stewart, et al. "Bayesian optimization for adaptive experimental design: a review." IEEE access 8 (2020): 13937-13948.
- Balandat, Maximilian, et al. "BoTorch: a framework for efficient Monte-Carlo Bayesian optimization." Advances in neural information processing systems 33 (2020): 21524-21538.
- Tutorial Examples: https://github.com/slaclab/bo_tutorial