Introduction to Bayesian optimization

Mauricio A. Álvarez, PhD

Centre of AI Fundamentals,
Department of Computer Science,
The University of Manchester

MLSS, Arequipa, August 5, 2025

Contents

Contents

Bayes opt

- Bayesian optimization (BayesOpt) is a class of machine-learning-based optimization methods focused on solving the problem

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$

where $\mathcal{X}$ is some design space of interest (known as the feasibility set).

Properties of the objective function $f(\mathbf{x})$ (I)

- The input $\mathbf{x}$ is in $\mathbb{R}^d$ for a value of d that is not too large. Typically $d \leq 20$ in most successful applications of BayesOpt.

- The feasible set $\mathcal{X}$ is a simple set, in which it is easy to assess membership. Typically $\mathcal{X}$ is
  - a hyper-rectangle $\left\{\mathbf{x} \in \mathbb{R}^d : a_i \leq x_i \leq b_i\right\}$.
  - the d-dimensional simplex $\left\{\mathbf{x} \in \mathbb{R}^d : \sum_i x_i = 1\right\}$.

- The objective function $f(\mathbf{x})$ is continuous.

- $f(\mathbf{x})$ is "expensive to evaluate" in the sense that the number of evaluations that may be performed is limited, typically to a few hundred,
  - each evaluation takes a substantial amount of time.
  - each evaluation bears a monetary cost.

Properties of the objective function $f(\mathbf{x})$ (II)

- $f(\mathbf{x})$ is "black box":
  - lacks known special structure like concavity or linearity that would make it easy to optimize.
- The optimization is "derivative-free":
  - we observe only $f(\mathbf{x})$ and no first- or second-order derivatives.
  - we can't apply first- and second-order methods like gradient descent, Newton's method, or quasi- Newton methods.
- The focus is on finding a global rather than local optimum.
- **Summary**: BayesOpt is designed for black-box derivative free global optimization.

Components of Bayes Opt (I)

- Fundamentally, Bayesian optimization is a sequential model-based approach to solving the problem

$$\mathbf{x}^\star = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}).$$

- We prescribe a prior belief over the possible objective functions and then sequentially refine this model as data are observed via Bayesian posterior updating.

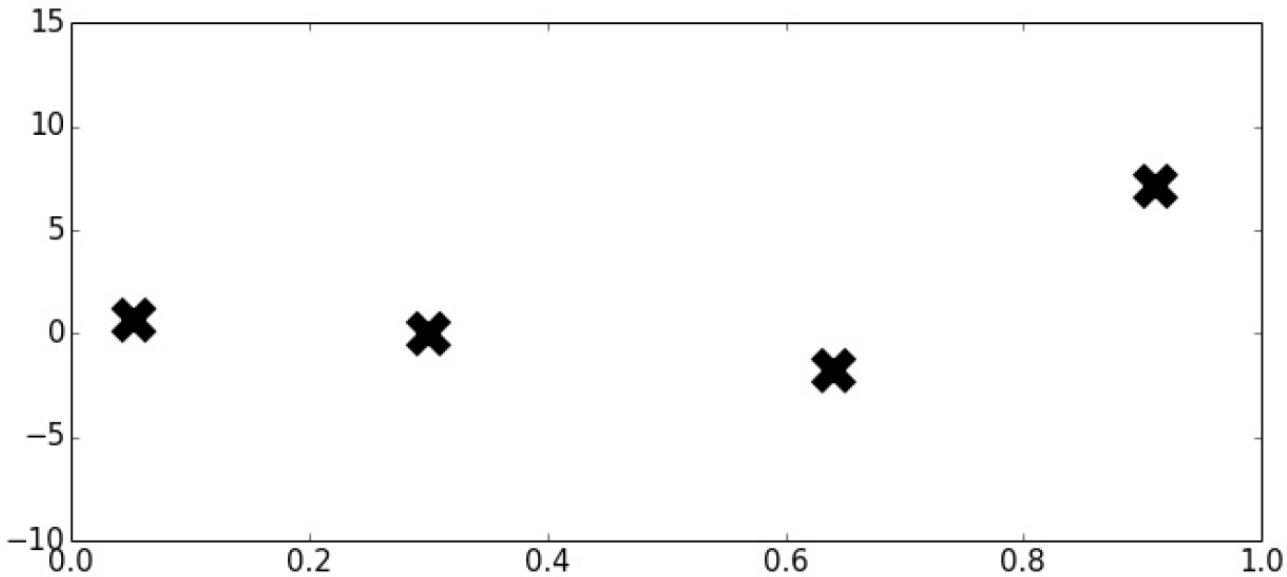
- The Bayesian posterior represents our updated beliefs, given data, on the likely objective function we are optimizing.

- Equipped with this probabilistic model, we can sequentially induce acquisition functions $\alpha_n : \mathcal{X} \mapsto \mathbb{R}$ that leverage the uncertainty in the posterior to guide exploration.

Components of Bayes Opt (II)

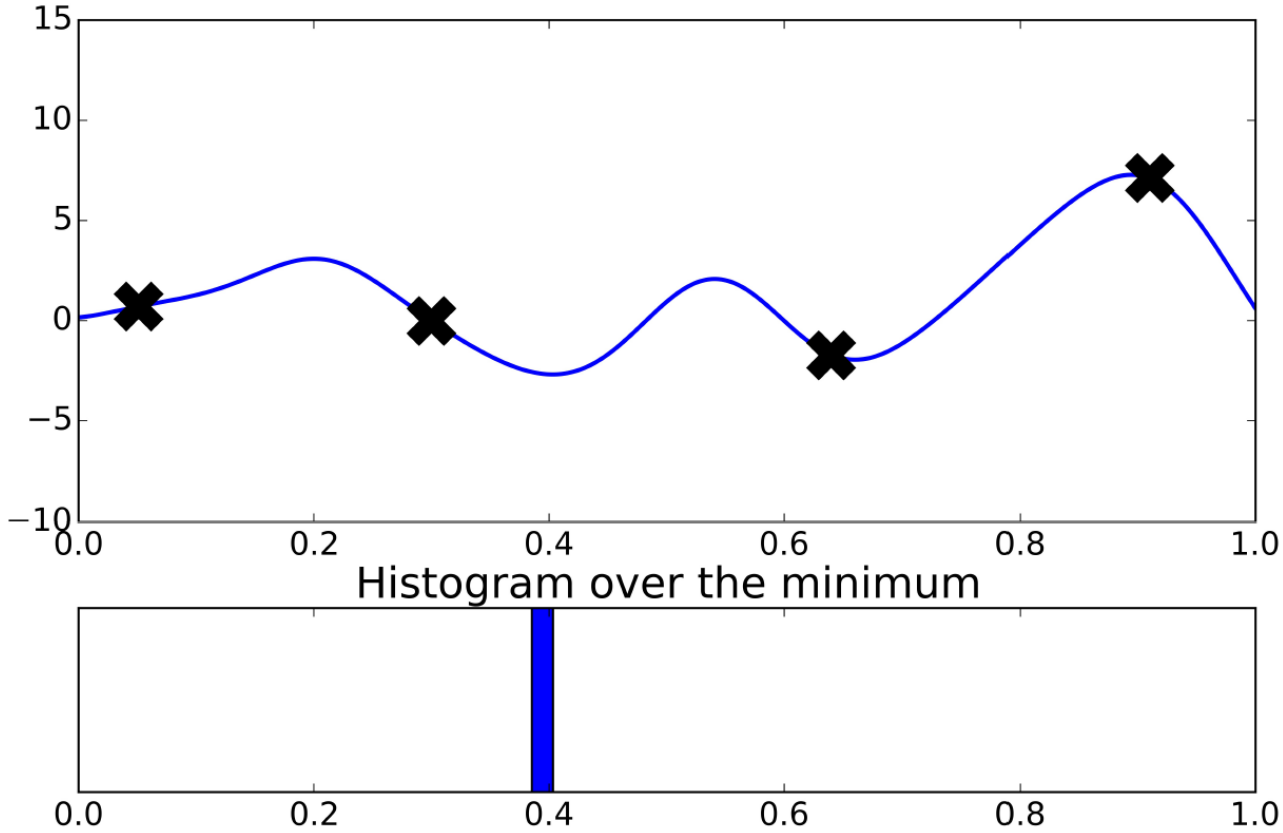
- The Bayesian optimization framework has two key ingredients.
- The first ingredient is a probabilistic surrogate model
  - a prior distribution that captures our beliefs about the behavior of the unknown objective function.
  - an observation model that describes the data generation mechanism.
- We will use a GP as the prior distribution.
- The second ingredient is the acquisition function for deciding where to sample next.
- After evaluating the objective according to an initial space-filling experimental design, these two components are used iteratively to allocate the remainder of a budget of N function evaluations.

How can we use the uncertainty in the GP?

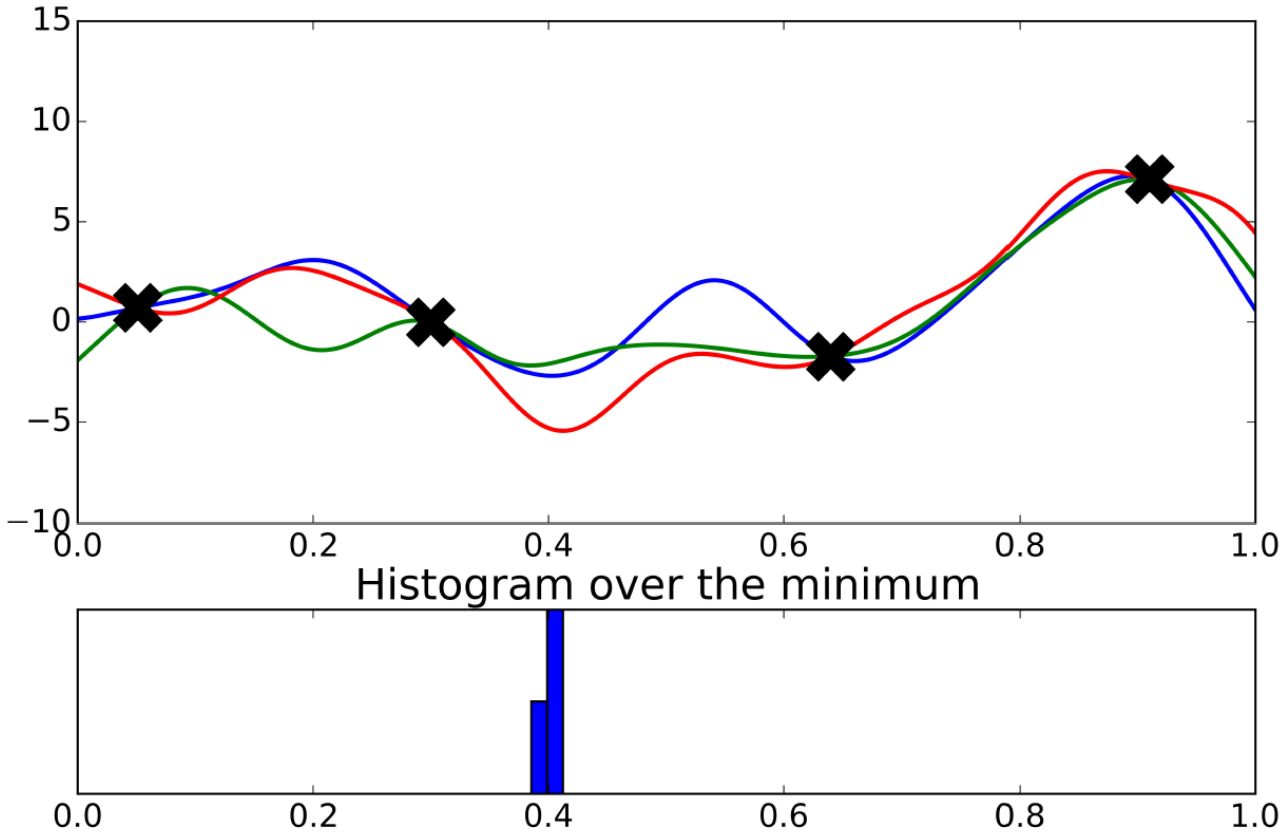


Where is the minimum of f?
Where should the take the next evaluation?

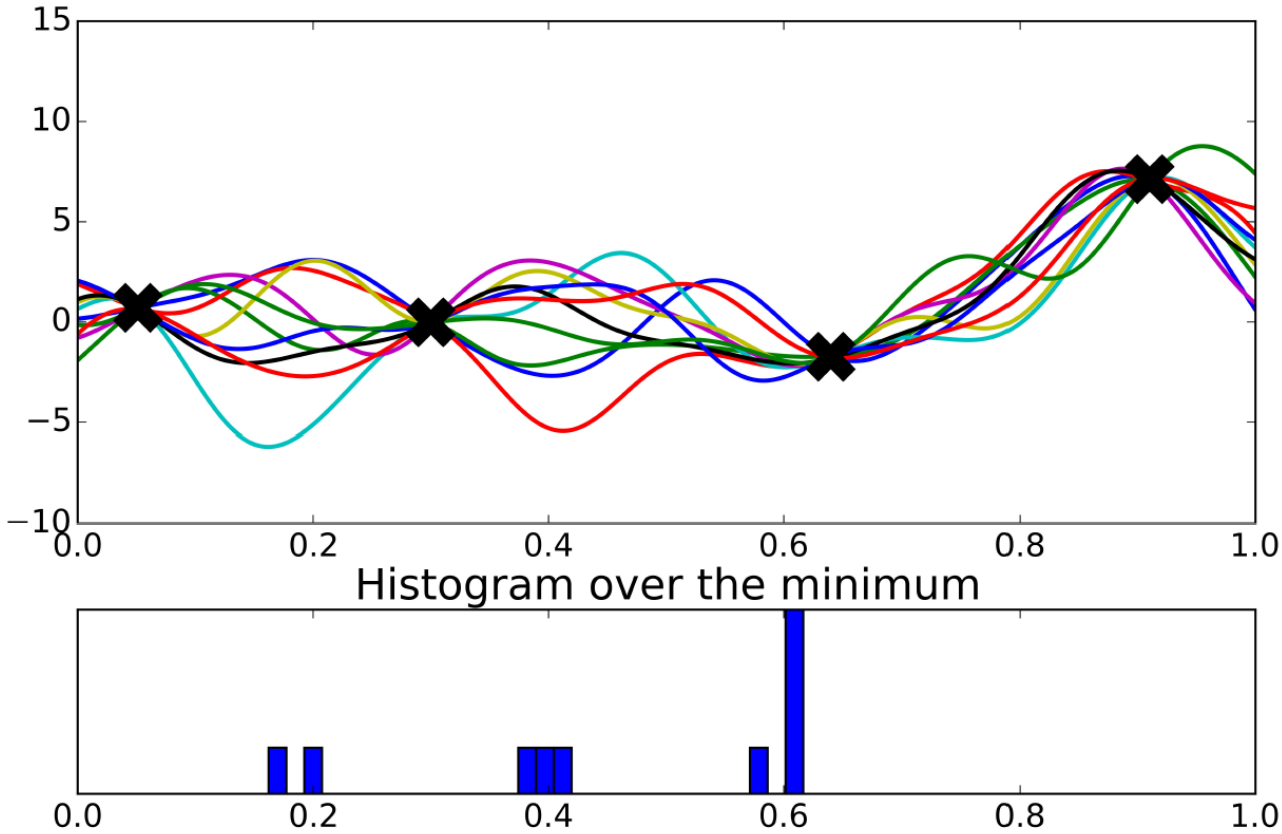
How can we use the uncertainty in the GP?



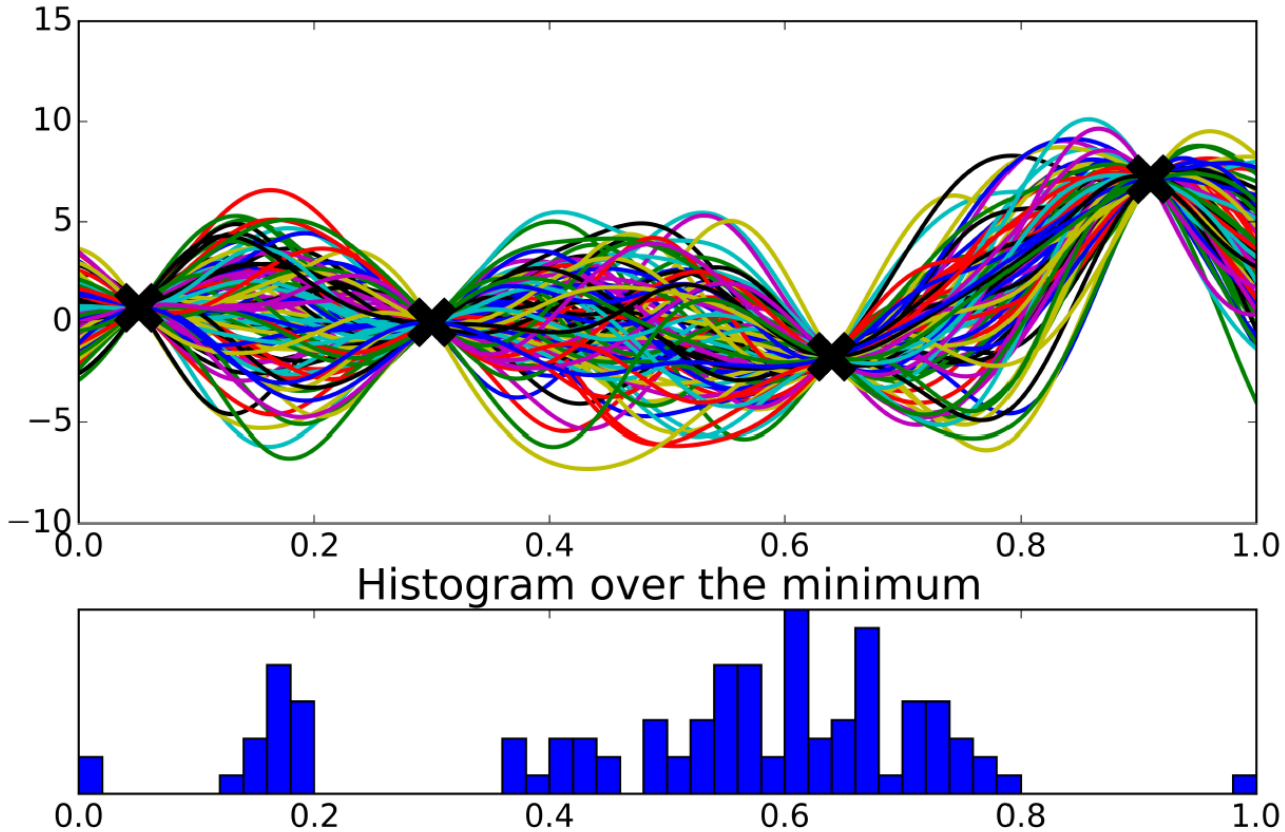
How can we use the uncertainty in the GP?



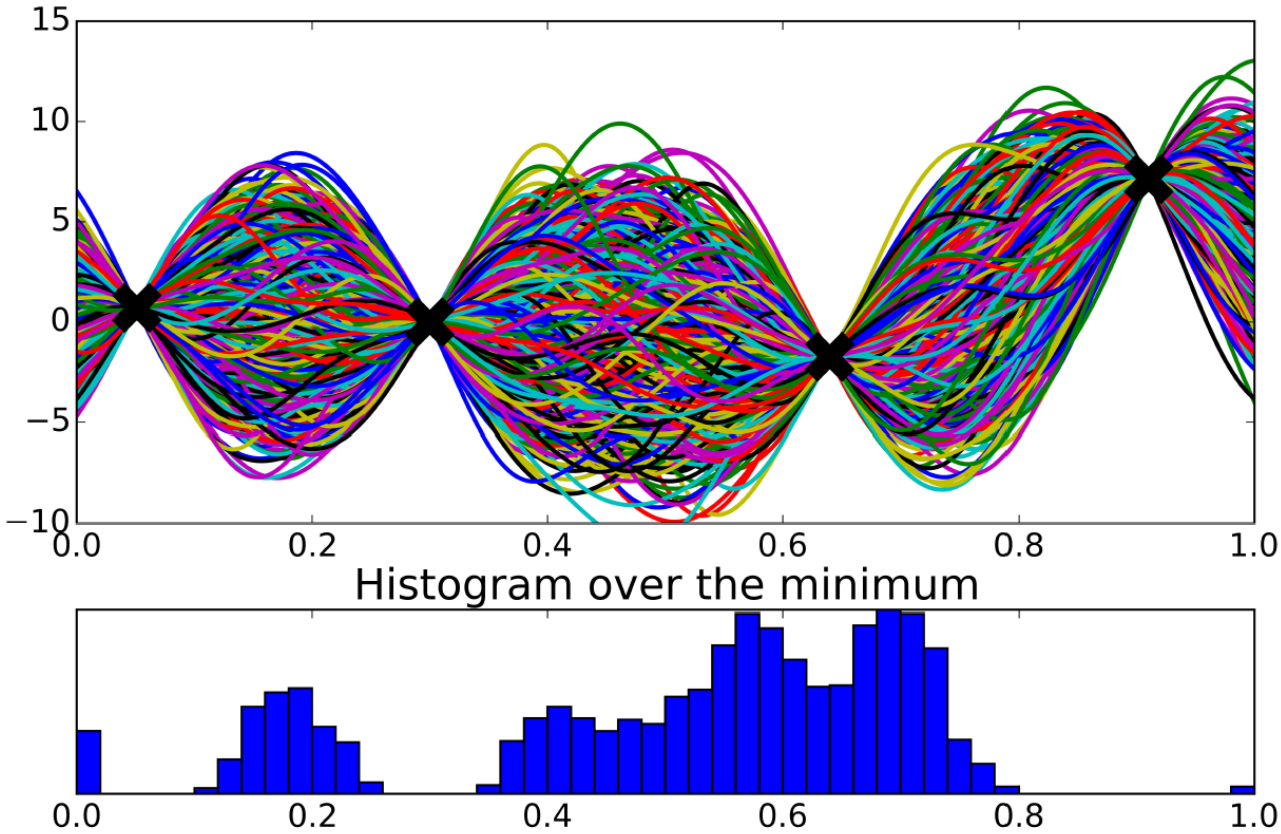
How can we use the uncertainty in the GP?



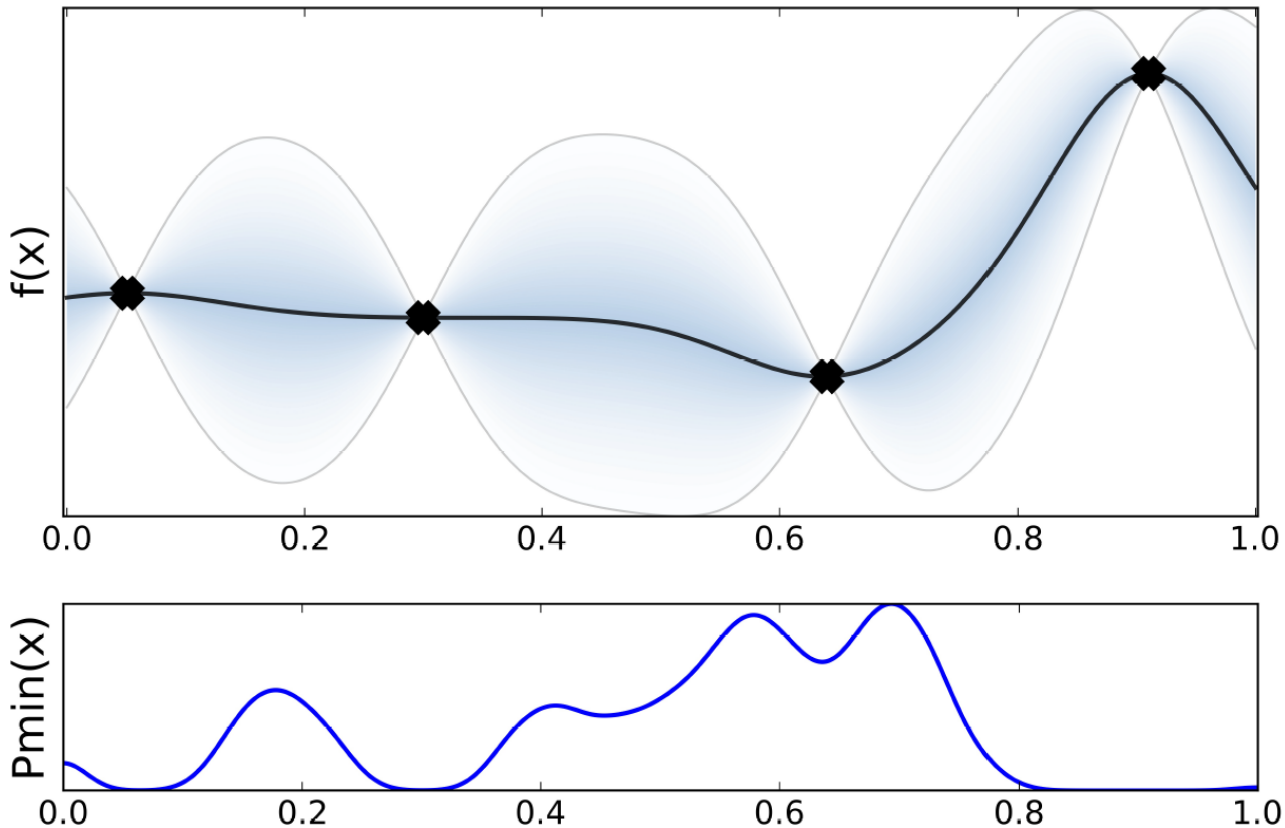
How can we use the uncertainty in the GP?



How can we use the uncertainty in the GP?



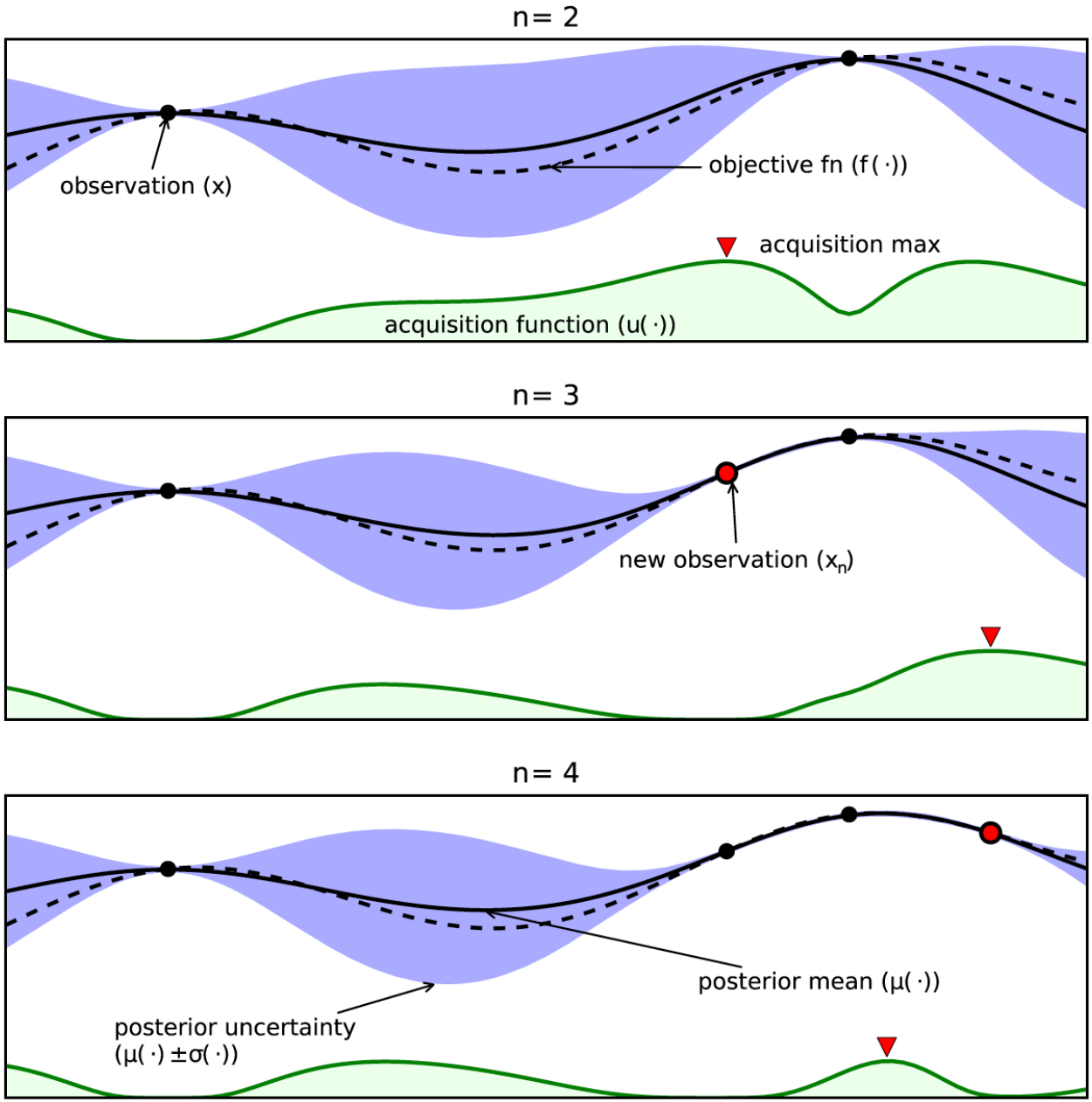
How can we use the uncertainty in the GP?



Pseudo-code for Bayesian optimization

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

Example


n= 2
observation (x)
objective fn (f(·))
acquisition max
acquisition function (u(·))
n= 3
new observation (x_n)
n= 4
posterior mean (μ(·))
posterior uncertainty
(μ(·) ±σ(·))

Contents

A/B testing

- Focus group for a product at scale.
- In its simplest form, consider two possible options denoted A and B.
- The technique consists of redirecting small fractions of user traffic to experimental designs of an ad, an app, a game, or a website.
- The developers can use noisy feedback to optimize any observable metric with respect to the product's configuration.
- The crucial problem is how to optimally query these subsets of users in order to find the best product with high probability within a predetermined query budget.

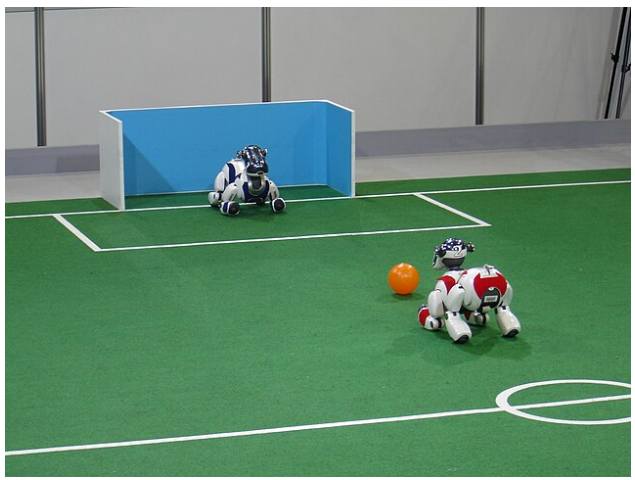
Recommender Systems

- Online content providers make product recommendations to their subscribers in order to optimize:
  - revenue in the case of e-commerce sites.
  - readership for news sites.
  - consumption for video and music streaming websites.
- The content provider can make multiple suggestions to any given subscriber.

Robotics and Reinforcement Learning

- Bayesian optimization has also been successfully applied to policy search.

- Examples include:
  - parameterizing a robot's gait it is possible to optimize it for velocity or smoothness.
  - policy parameterization and search techniques have been used to navigate a robot through landmarks, minimizing uncertainty about its own location and map estimate.



Environmental Monitoring and Sensor Networks

- Sensor networks are used to monitor environmentally relevant quantities: temperature, concentration of pollutants in the atmosphere, soil, oceans, etc
- These networks make noisy local measurements that are interpolated to produce a global model of the quantity of interest.
- These sensors are expensive to activate but one can answer important questions like what is the hottest or coldest spot in a building by activating a relatively small number of sensors.
- Bayesian optimization was used for this task and the similar one of finding the location of greatest highway traffic congestion.

Preference Learning and Interactive Interfaces

- The computer graphics and animation fields are filled with applications that require the setting of tricky parameters.
- The models are complex and the parameters unintuitive for nonexperts.
- Bayesian optimization has been used to set the parameters of several animation systems by showing the user examples of different parametrized animations and asking for feedback.

Automatic Machine Learning and Hyperparameter Tuning

- The goal is to automatically select:
  - the best model (e.g., random forests, support vector machines, neural networks, etc.)
  - its associated hyperparameters for solving a task on a given data set.

- Useful for big data sets or when considering many alternatives
  - cross validation is very expensive.
  - it is important to find the best technique within a fixed budget of cross-validation tests.

- The objective function here is the generalization performance of the models and hyperparameter settings.

- Frameworks: auto-weka and auto-sklearn.

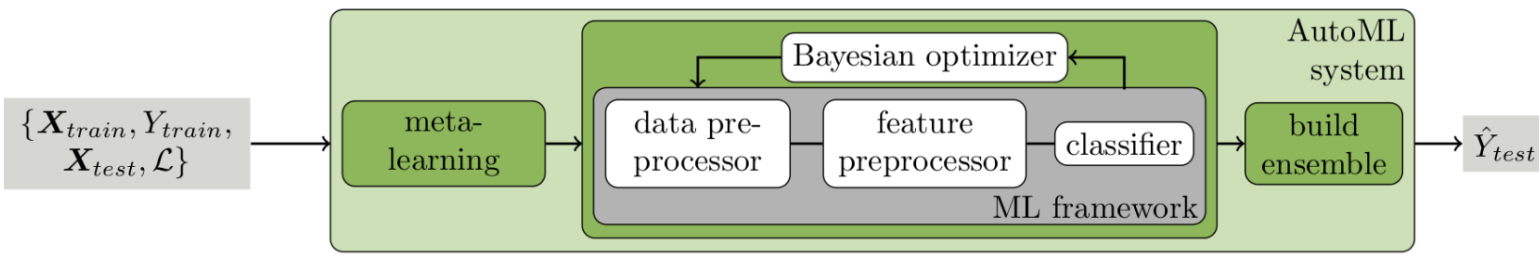
auto-sklearn

auto-sklearn is an automated machine learning toolkit and a drop-in replacement for a scikit-learn estimator.

Find the documentation **here**. Quick links:

- Installation Guide
- Releases
- Manual
- Examples
- API

auto-sklearn in one image



auto-sklearn in four lines of code

```
import autosklearn.classification
cls = autosklearn.classification.AutoSklearnClassifier()
cls.fit(X_train, y_train)
predictions = cls.predict(X_test)
```

Contents

Exploration vs exploitation

Acquisition function: exploration vs exploitation

- The acquisition function is the mechanism to implement the trade-off between exploration and exploitation in BayesOpt.

- *Exploring* means to move towards less explored regions of the search space where predictions based on the surrogate model are more uncertain, with higher variance.

- *Exploiting* means to consider the area providing more chance to improve the current solution (with respect to the current surrogate model):
  - find a lower value of the objective function if minimizing it.
  - find a higher value of the objective function if maximizing it.

GP Upper (lower) confidence band

- It provides a direct trade-off between exploitation and exploration.
- For maximization problems, we use the upper confidence bound

$$\alpha_{\mathrm{UCB}}(\mathbf{x}) := \mu_n(\mathbf{x}) + \beta_n \sigma_n(\mathbf{x}),$$

  where $\mu_n(\mathbf{x})$ and $\sigma_n(\mathbf{x})$ are the predictive mean and standard deviation after having observed $\mathcal{D}_n$.
- For minimization problems, we use the lower confidence bound

$$\alpha_{\mathrm{LCB}}(\mathbf{x}) := \mu_n(\mathbf{x}) - \beta_n \sigma_n(\mathbf{x}),$$

- There are theoretically motivated guidelines for setting and scheduling the hyperparameter β_n.

Expected improvement (I)

- Assume that we may only return a solution that we have evaluated as our final solution to

$$\mathbf{x}^\star = \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}),$$

- What this means is that $\mathbf{x}^\star$ is one of the inputs in $\mathcal{D}_N$.

- The optimal choice is the previously evaluated point with the largest observed value.

- Let $f_n^* = \max_{m \leq n} f(\mathbf{x}_m)$ be the value of this point, where n is the number of times we have evaluated f thus far.

Expected Improvement (II)

- Now suppose in fact we have one additional evaluation to perform, and we can perform it anywhere.
- After this new evaluation, the value of the best point we have observed will either be
  - $f(\mathbf{x})$ if $f(\mathbf{x}) \geq f_n^*$.
  - f_n^* if $f(\mathbf{x}) \leq f_n^*$.
- The improvement can be written as

$$[f(\mathbf{x}) - f_n^*]^+ ,$$

  where $a^+ = \max(a, 0)$ indicates the positive part.
- It can also be written as $(f(\mathbf{x}) - f_n^*)\mathbb{I}(f(\mathbf{x}) > f_n^*)$, where $\mathbb{I}(\cdot)$ is the indicator function.
- Since we don't know the value of $f(\mathbf{x})$ after we have found $\mathbf{x}$, what we can do is to use the expected value of this improvement and choose $\mathbf{x}$ to maximize it.

Expected Improvement (III)

- The expected improvement is defined as

$$\alpha_{\mathrm{EI}}(\mathbf{x}) := \mathbb{E}_n \left[[f(\mathbf{x}) - f_n^*]^+ \right].$$

- It can be evaluated in closed form, leading to

$$\alpha_{\mathrm{EI}}(\mathbf{x}) := (\mu_n(\mathbf{x}) - f_n^*)\, \Phi\left(\frac{\mu_n(\mathbf{x}) - f_n^*}{\sigma_n(\mathbf{x})} \right) + \sigma_n(\mathbf{x}) \phi\left(\frac{\mu_n(\mathbf{x}) - f_n^*}{\sigma_n(\mathbf{x})} \right),$$

where $\Phi(\cdot)$ is the cumulative Gaussian distribution and ϕ is the probability Gaussian density.

Probability of Improvement

- The probability of improvement (PI), measures the probability that a point $\mathbf{x}$ leads to an improvement upon f_n^*.

- Since the posterior distribution of $f(\mathbf{x})$ is Gaussian, we can analytically compute this probability as follows:

$$\alpha_{\mathrm{PI}}(\mathbf{x}) := \mathrm{P}[f(\mathbf{x}) > f_n^*] = \Phi\left(\frac{\mu_n(\mathbf{x}) - f_n^*}{\sigma_n(\mathbf{x})}\right).$$

- One of the drawbacks of PI is that it is biased towards exploitation.

Thompson sampling (I)

- For Thompson sampling (TS), the acquisition function is given by a sample from the GP posterior

$$\alpha_{\mathrm{TS}}(\mathbf{x}) = f^n(\mathbf{x}),$$

where $f^n(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')|\mathcal{D}_n)$.

- The optimal $\mathbf{x}$ is obtained by returning the index of the maximum element in the sampled vector.

- This process for getting the optimal $\mathbf{x}$ is impractical when the dimensionality of the input space is greater than 1.

- Furthermore, it lacks precision. Ideally we would like to represent the sample explicitly as a function of $\mathbf{x}$.

Thompson sampling (II)

- It is possible to obtain such explicit function using spectral representations for the kernel function, e.g. random Fourier features.
- The GP sample $f^n(\mathbf{x})$ can be written as

$$f^n(\mathbf{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}),$$

where

$$\boldsymbol{\phi}(\mathbf{x}) = \sqrt{2\alpha/m}\cos(\mathbf{W}\mathbf{x} + \mathbf{b}),$$

with m the number of random features, α is a constant associated to the spectral representation (sp) of the kernel, $\mathbf{W} \sim p(\mathbf{W})$, where $p(\mathbf{W})$ is also associated with the sp of the kernel and $b \sim \mathcal{U}[0, 2\pi]$.

- $\boldsymbol{\theta}$ is a vector of weights sampled from the posterior distribution $\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D}_n)$,

$$\boldsymbol{\theta} \mid \mathcal{D}_n \sim \mathcal{N}\left(\mathbf{A}^{-1}\boldsymbol{\Phi}^\top \mathbf{y}_n, \sigma^2\mathbf{A}^{-1}\right),$$

where $\mathbf{A} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \sigma^2\mathbf{I}$ and $\boldsymbol{\Phi}^\top = [\boldsymbol{\phi}(\mathbf{x}_1)\ldots\boldsymbol{\phi}(\mathbf{x}_n)]$.

Thompson sampling (III)

- A different approach for sampling from a GP uses the "Matheron's rule" (Wilson et al., 2020).
- The GP sample $f^n(\mathbf{x})$ can be written as

$$f^n(\mathbf{x}) = \boldsymbol{\eta}^\top \boldsymbol{\phi}(\mathbf{x}) + \boldsymbol{\nu}^\top \mathbf{k}(\mathbf{x}, \mathbf{Z}),$$

where $\boldsymbol{\phi}(\mathbf{x})$ are the same basis functions than before, $\boldsymbol{\eta}$ is a vector of weights sampled from the prior distribution $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$,

$$\boldsymbol{\nu} = \mathbf{K}^{-1}(\mathbf{Z}, \mathbf{Z})(\mathbf{u} - \boldsymbol{\Phi}\boldsymbol{\eta}),$$
$$\mathbf{k}(\mathbf{x}, \mathbf{Z}) = [k(\mathbf{x}, \mathbf{z}_1), \cdots, k(\mathbf{x}, \mathbf{z}_\ell)]^\top,$$

with $\mathbf{Z} = \{\mathbf{z}_j\}_{j=1}^{\ell}$ the set of so-called "inducing inputs",
$\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(\mathbf{Z}, \mathbf{Z}))$ the set of corresponding "inducing variables", and
$\mathbf{K}(\mathbf{Z}, \mathbf{Z})$ the kernel matrix with entries $k(\mathbf{z}_i, \mathbf{z}_j)$ for $i, j = 1, \cdots, \ell$.

Entropy search (I)

- Information-based policies consider the posterior distribution over the unknown minimizer $\mathbf{x}^*$, denoted $p_*(\mathbf{x}|\mathcal{D}_n)$.
- Entropy search (ES) techniques aim to reduce the uncertainty in the location $\mathbf{x}^*$ by selecting the point that is expected to cause the largest reduction in entropy of the distribution $p_*(\mathbf{x}|\mathcal{D}_n)$.
- ES measures the expected information gain from querying an arbitrary point $\mathbf{x}$ and selects the point that offers the most information about the unknown $\mathbf{x}^*$.

Entropy search (II)

- The *differential entropy* $H(\mathbf{x}^\star \mid \mathcal{D}_n)$ is given as

$$H(\mathbf{x}^\star \mid \mathcal{D}_n) = \mathbb{E}[-\log(p_\star(\mathbf{x}|\mathcal{D}_n))].$$

- The acquisition function for ES can be expressed formally as

$$\alpha_{\mathrm{ES}}(\mathbf{x}) := H(\mathbf{x}^\star \mid \mathcal{D}_n) - \mathbb{E}_{y|\mathcal{D}_n,\mathbf{x}} H(\mathbf{x}^\star \mid \mathcal{D}_n \cup \{(\mathbf{x}, y)\}),$$

  where
  - $H(\mathbf{x}^\star|\mathcal{D}_n)$ denotes the differential entropy of the posterior distribution.
  - the expectation is over the distribution of the random variable $y \sim \mathcal{N}(\mu_n(\mathbf{x}), \sigma_n^2(\mathbf{x}))$.

- This function is not tractable for continuous search spaces $\mathcal{X}$ so approximations must be made.

Entropy search (III)

- Early work discretized the space $\mathcal{X}$ and computed the conditional entropy via MC sampling (Villemonteix et al., 2008).
- Another approach is to discretize $\mathcal{X}$ to obtain a smooth approximation to p_* and its expected information gain (Hennig and Schuler, 2012).
- This method is unfortunately $\mathcal{O}(M^4)$ where M is the number of discrete so-called representer points.

Entropy search (IV)

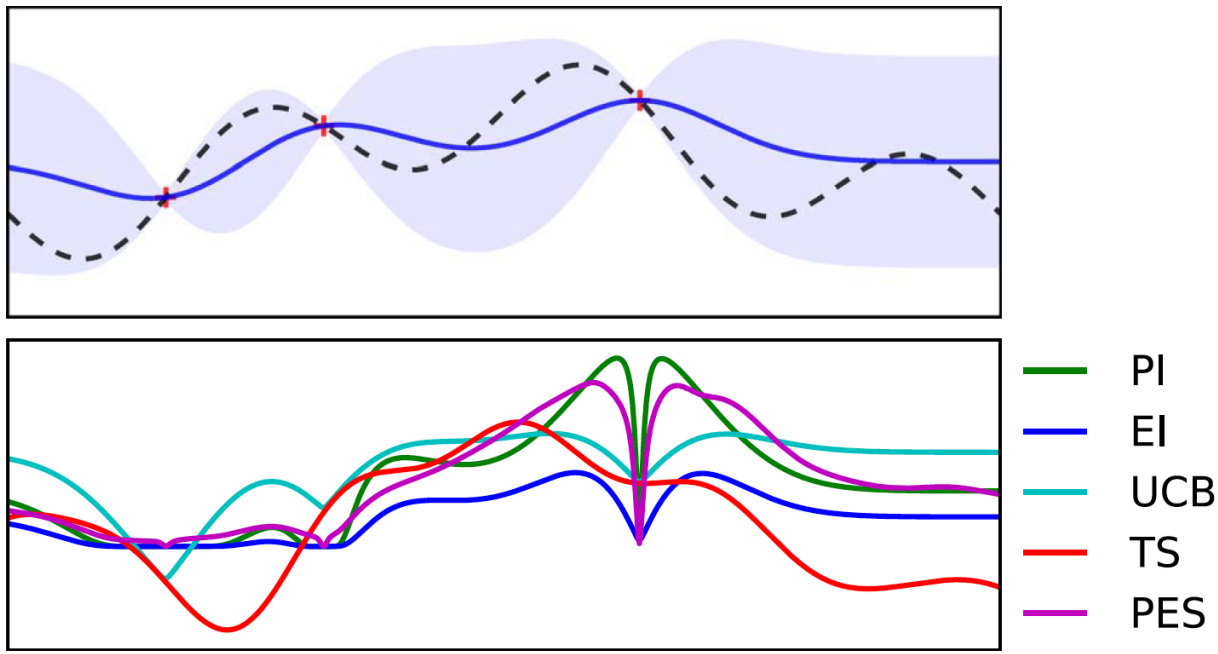
- Predictive entropy search (PES) (Hernández-Lobato et al., 2014) removes the need for a discretization and approximates the acquisition function in $\mathcal{O}((n+d)^3)$.
- For $d < n$ is of the same order as EI.
- This is achieved by using the symmetric property of mutual information to write

$$\alpha_{\text{PES}}(\mathbf{x}) := H(y|\mathcal{D}_n, \mathbf{x}) - \mathbb{E}_{\mathbf{x}^\star|\mathcal{D}_n}\left[H(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}^\star)\right].$$

Entropy search (V)

- The expectation can be approximated via MC with Thompson samples.
- Three simplifying assumptions are made to compute $H(y|\mathcal{D}_n, \mathbf{x}, \mathbf{x}^\star)$.

Example different acquisition functions



Contents

Handling hyperparameters (I)

- We have mostly ignored the kernel hyperparameters and assumed they were given.

- In reality, the acquisition function is not only a function of $\mathbf{x}$ but also a function of the hyperparameters $\boldsymbol{\theta}$ of the GP, $\alpha(\mathbf{x}, \boldsymbol{\theta})$.

- We would like to marginalize out our uncertainty about $\boldsymbol{\theta}$ with the following expression:

$$\alpha_n(\mathbf{x}) := \mathbb{E}_{\boldsymbol{\theta}|\mathcal{D}_n}[\alpha(\mathbf{x}; \boldsymbol{\theta})] = \int \alpha(\mathbf{x}; \boldsymbol{\theta}) p\left(\boldsymbol{\theta}|\mathcal{D}_n\right) \mathrm{d}\boldsymbol{\theta},$$

where

$$p\left(\boldsymbol{\theta}|\mathcal{D}_n\right) = \frac{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p\left(\mathcal{D}_n\right)}.$$

Handling hyperparameters (II)

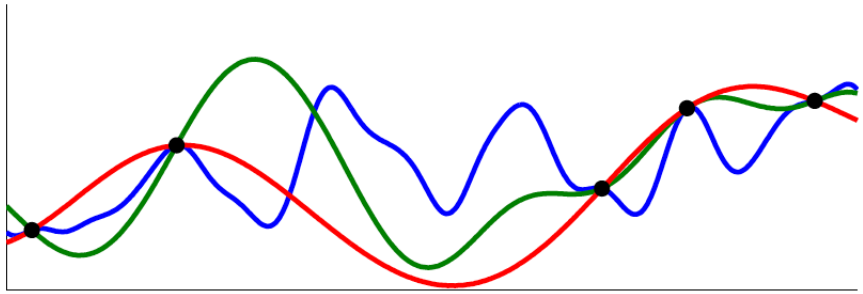
- Points estimates for $\boldsymbol{\theta}$, $\hat{\alpha}_n(\mathbf{x}) = \alpha(\mathbf{x}; \hat{\boldsymbol{\theta}}_n)$, where $\hat{\boldsymbol{\theta}}_n$ can either be $\hat{\boldsymbol{\theta}}_n^{\mathrm{ML}}$ or $\hat{\boldsymbol{\theta}}_n^{\mathrm{MAP}}$.

- Marginalizing out the hyperparameters using either quadrature or Monte Carlo, for example, approximating the integral before using

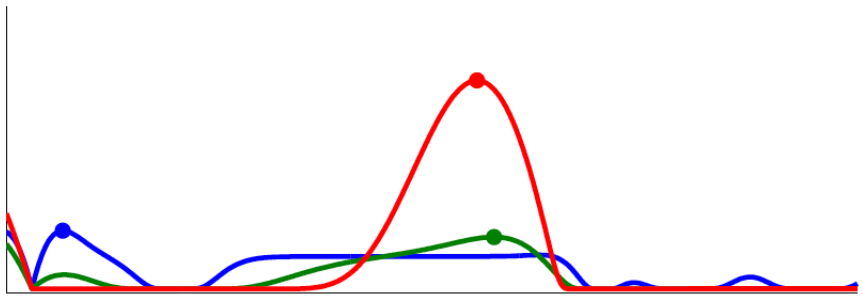
$$\mathbb{E}_{\boldsymbol{\theta}|\mathcal{D}_n}[\alpha(\mathbf{x}; \boldsymbol{\theta})] \approx \frac{1}{M} \sum_{i=1}^{M} \alpha(\mathbf{x}; \boldsymbol{\theta}_n^{(i)}),$$

  where $\boldsymbol{\theta}_n^{(i)}$ are M samples from the posterior $p(\boldsymbol{\theta}|\mathcal{D}_n)$.

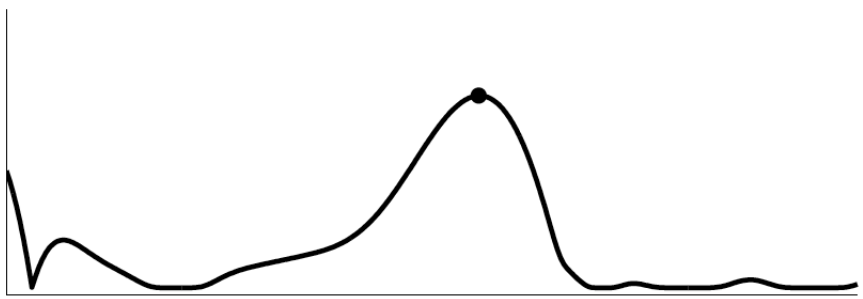
Effect of the hyperparameters



(a) Posterior samples under varying hyperparameters



(b) Expected improvement under varying hyperparameters



(c) Integrated expected improvement

Optimizing acquisition functions

- Discretization
- Gradient descent methods: Conjugate gradient, BFGS, etc.
- Evolutionary algorithms: Covariance Matrix Adaptation - Evolution Strategy (CMA-ES).

Contents

Contents

High-dimensional problems

- Bayesian optimization is restricted to problems of moderate dimension, typically 20.
- If there are more dimensions, there is a need to impose some structure in the problem.
- For certain classes of problems most dimensions do not change the objective function significantly, e.g. hyperparameter optimization for neural networks
  - find a random embedding and solve the optimization problem in a lower dimensional space (Wang et al., 2013).

REMBO: Bayes Opt with Random Embeddings

1: Generate a random matrix $\mathbf{A}$
2: Choose the set $\mathcal{Z}$
3: **for** $n = 1, 2, \ldots$ **do**
4: select $\mathbf{z}_{n+1}$ by optimizing the acquisition function α

$$\mathbf{z}_{n+1} = \arg\max_{\mathbf{z} \in \mathcal{Z}} \alpha\left(\mathbf{z} \mid \mathcal{D}_n\right)$$

5: augment the data $\mathcal{D}_{n+1} = \{\mathcal{D}_n, (\mathbf{z}_{n+1}, f\left(\mathbf{A}\mathbf{z}_{n+1}\right))\}$.
6: update the kernel hyperparameters
7: **end for**

Trust Region BO (TuRBO)

- Eriksson et al. (2019) proposed replacing the global BO approach for a collection of simultaneous local optimization runs using independent probabilistic models.
- Each GP is used to find a trust region.
- The global minimizer is the minimizer for all the trust regions.

Local GP finding a trust region

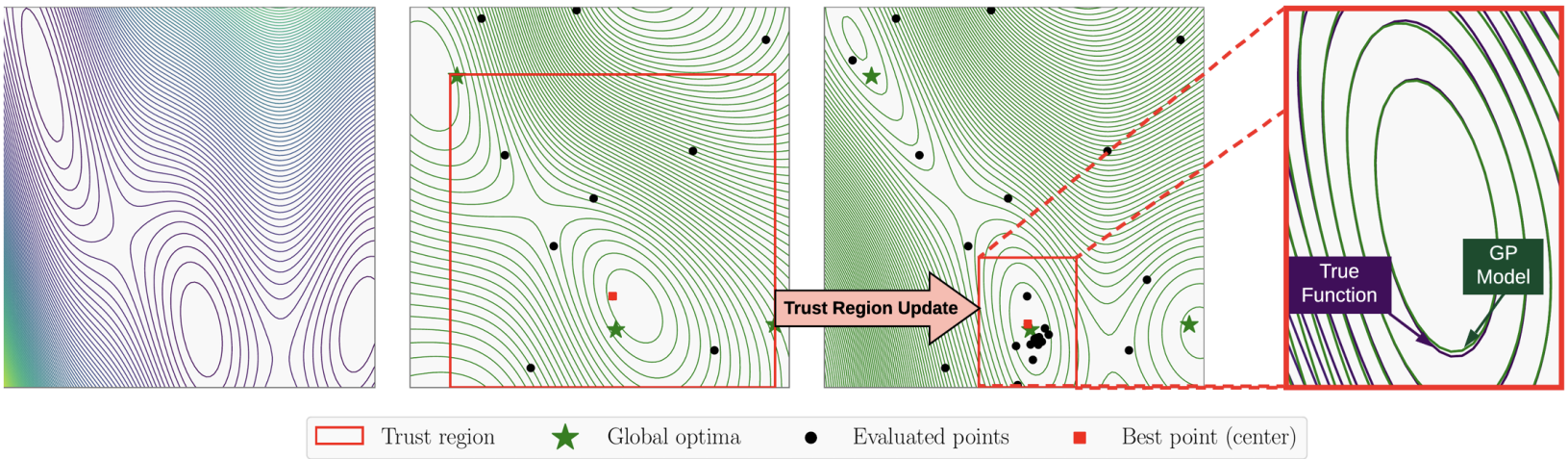


Figure 1: Illustration of the TuRBO algorithm. **(Left)** The true contours of the Branin function. **(Middle left)** The contours of the GP model fitted to the observations depicted by black dots. The current TR is shown as a red square. The global optima are indicated by the green stars. **(Middle right)** During the execution of the algorithm, the TR has moved towards the global optimum and has reduced in size. The area around the optimum has been sampled more densely in effect. **(Right)** The local GP model almost exactly fits the underlying function in the TR, despite having a poor global fit.

Vanilla BO in high dimensions

- Hvarfner et al. (2024) studies the degeneracies that make vanilla BO poorly suited to high-dimensional tasks.
- Modification: scale the Gaussian process lengthscale prior in the dimensionality.
- Standard BO works drastically better than previously thought in high dimensions.

Contents

Quite a few recent papers

- A growing literature on using LLMs at some stage of the BO loop, to incorporate expert knowledge, to replace the surrogate or to learn AFs.

- Some recent works include
  - "Large Language Models to enhance Bayesian Optimization" (ICLR, 2022). Used for zero-shot warmstarting, sampling candidates and surrogate modelling.
  - "A Sober Look at LLMs for Material Discovery: Are They Actually Good for Bayesian Optimization Over Molecules?" (ICML, 2024).

    LLMs can be useful for BO over molecules, but only if they have been pretrained or finetuned with domain-specific data

  - "FunBO: Discovering Acquisition Functions for Bayesian Optimization with FunSearch" (ICML, 2025).

    an LLM-based method that can be used to learn new AFs written in computer code by leveraging access to a number of evaluations for a limited set of objective functions

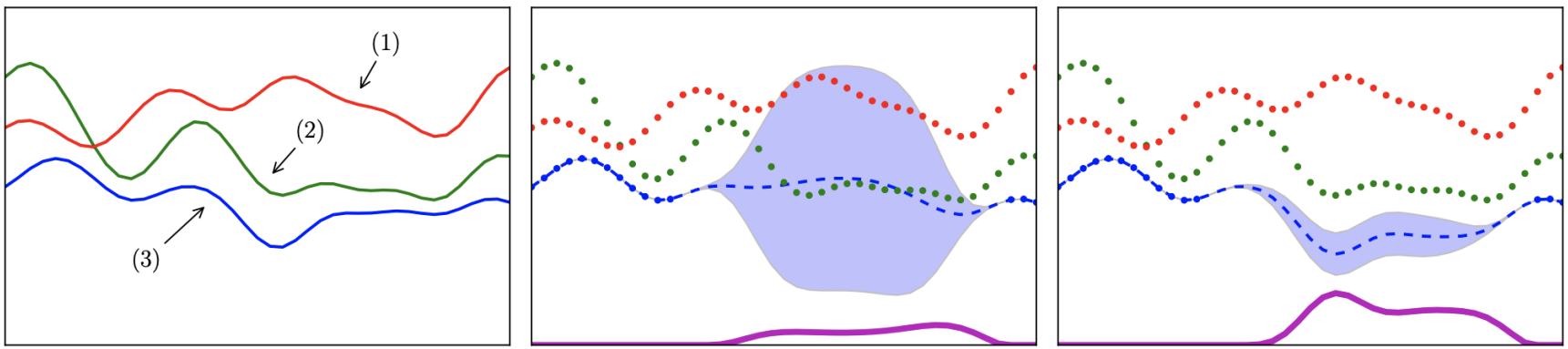
Contents

Multi-task Bayesian optimization (I)

- Two settings:
  - We want to optimise an objective that it is very expensive to evaluate but we have access to another function, correlated with the objective, that is cheaper to evaluate (multi-fidelity or transfer learning).
  - We want to optimize several tasks at the same time (multi-objective).

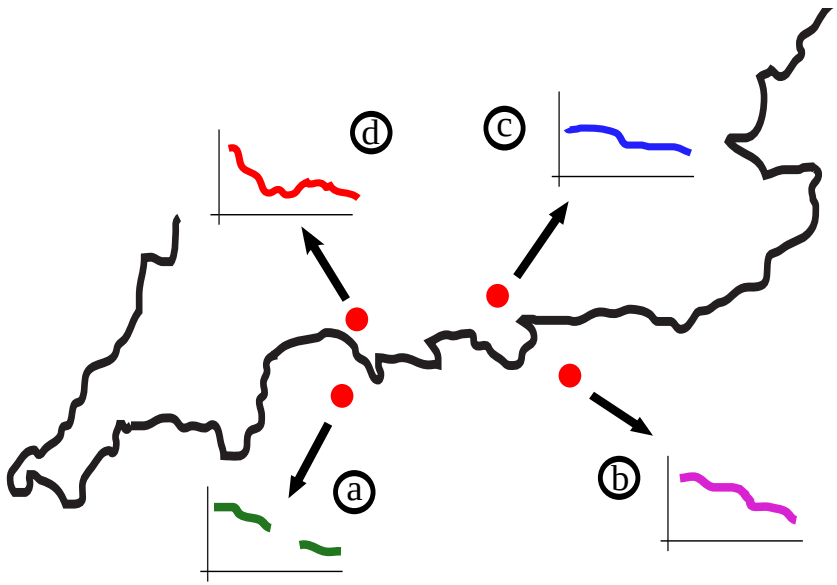
- Multi-output Gaussian processes can help here.

Multi-task Bayesian optimization (II)

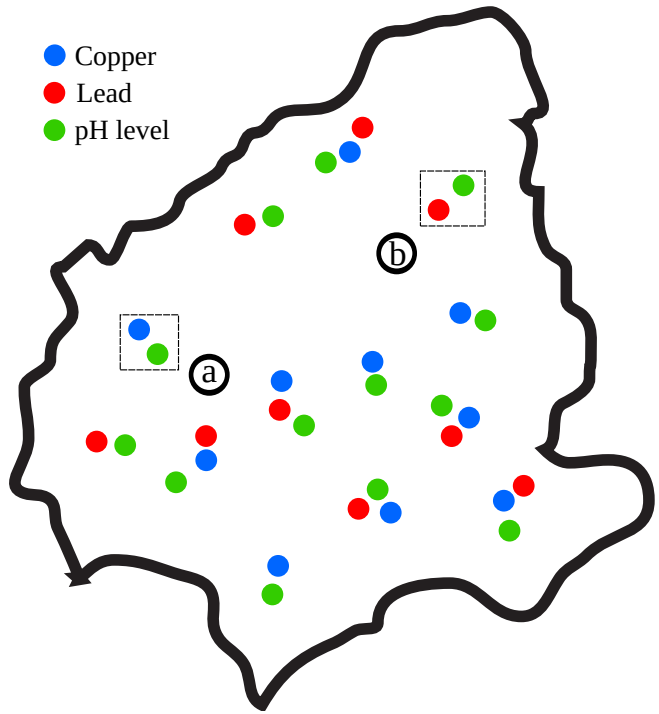


(a) Multi-task GP sample functions (b) Independent GP predictions (c) Multi-task GP predictions

Dependencies between related processes

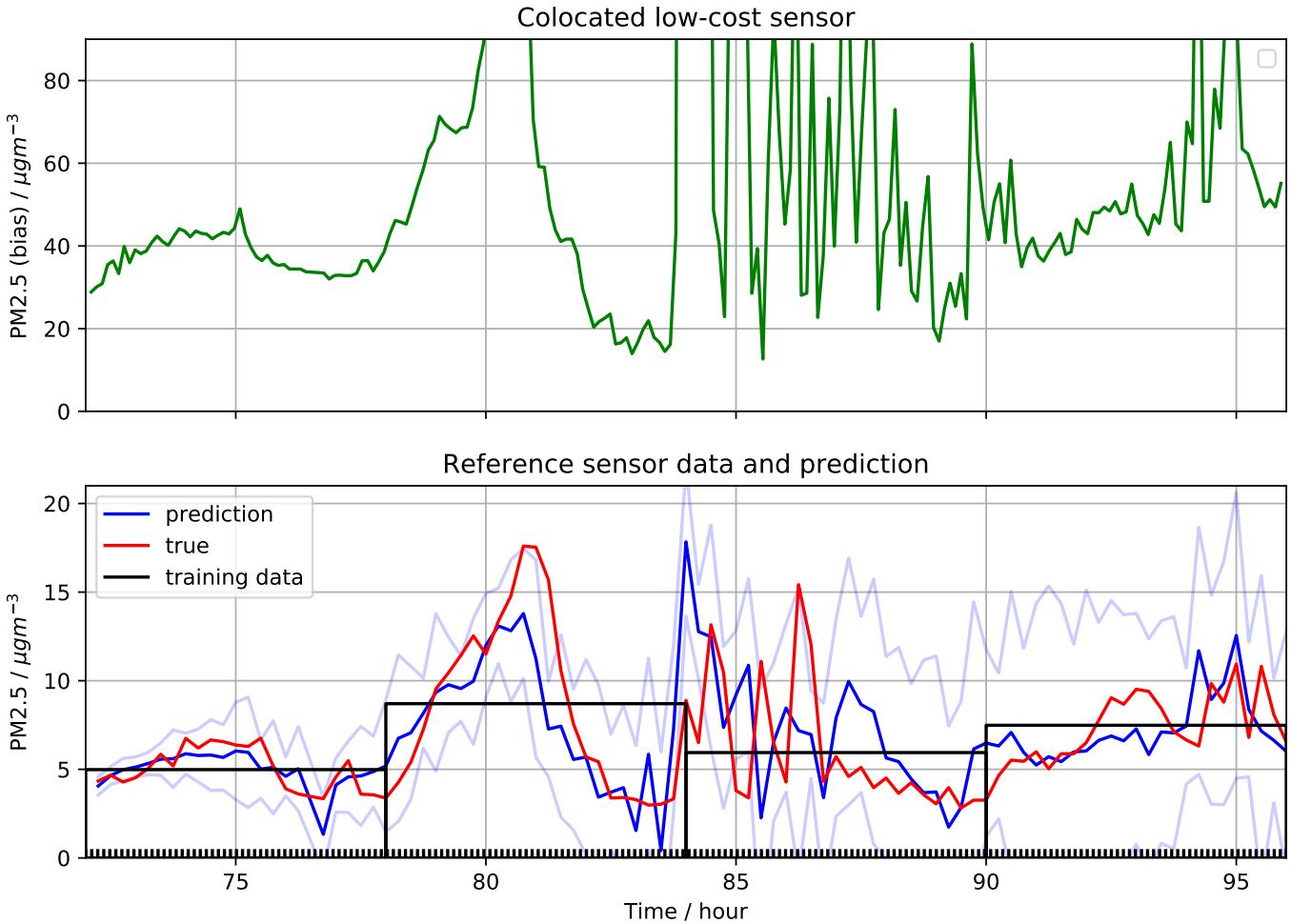


A network of sensors

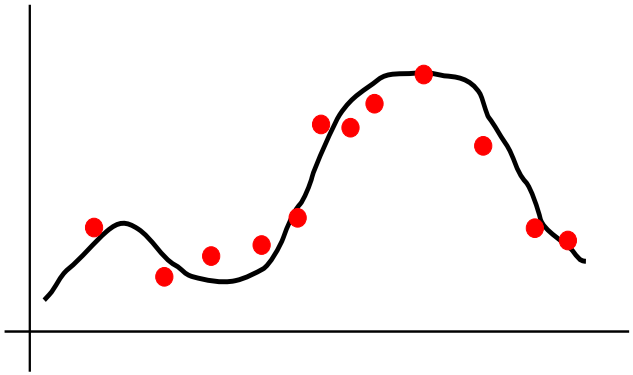


Pollutant metals concentrations

Multi-fidelity setting

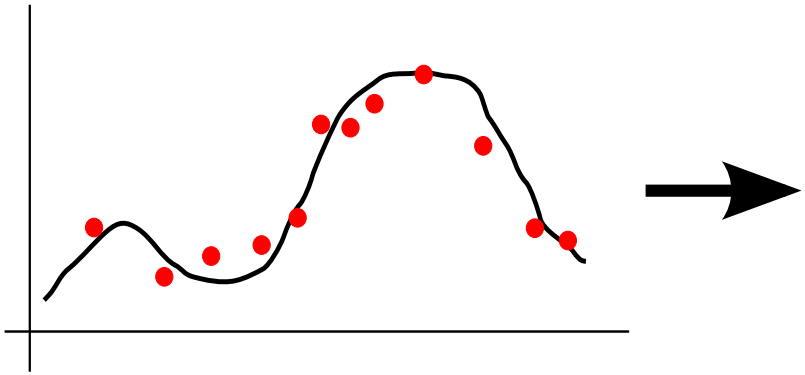


Valid covariance functions for multiple outputs (II)



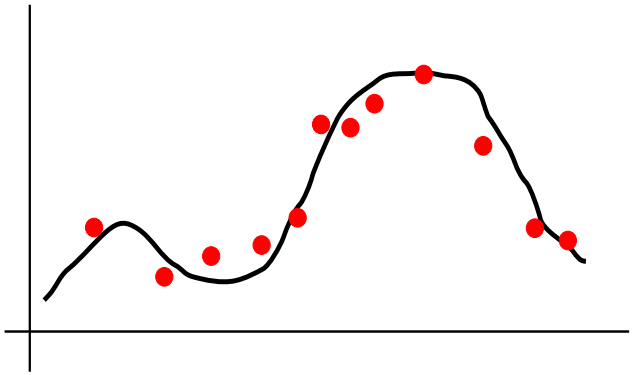
$$\mathcal{D}^1 = \{(\mathbf{x}_i^1, y_i^1) | i = 1, \ldots, N_1\}$$

Valid covariance functions for multiple outputs (II)



$$\mathcal{D}^1 = \{(\mathbf{x}_i^1, y_i^1) | i = 1, \ldots, N_1\}$$

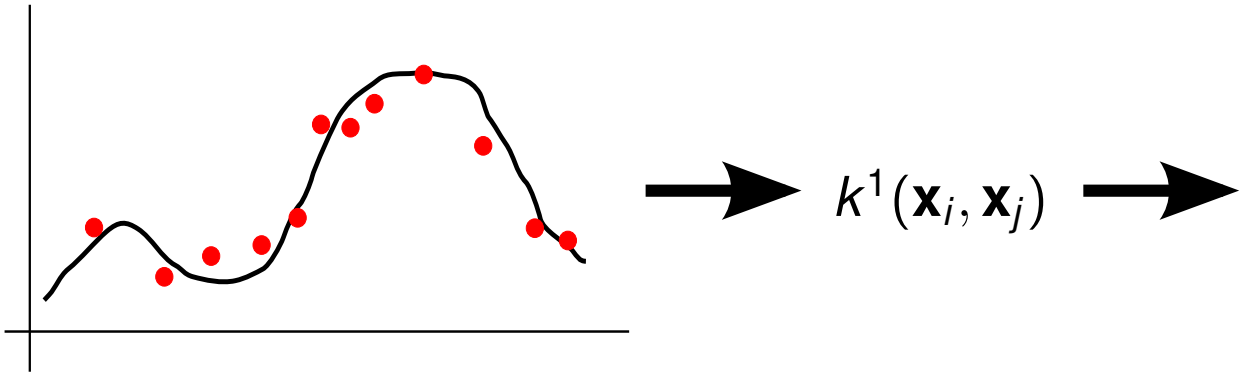
Valid covariance functions for multiple outputs (II)



$$k^1(\mathbf{x}_i, \mathbf{x}_j)$$

$$\mathcal{D}^1 = \{(\mathbf{x}_i^1, y_i^1) | i = 1, \ldots, N_1\}$$

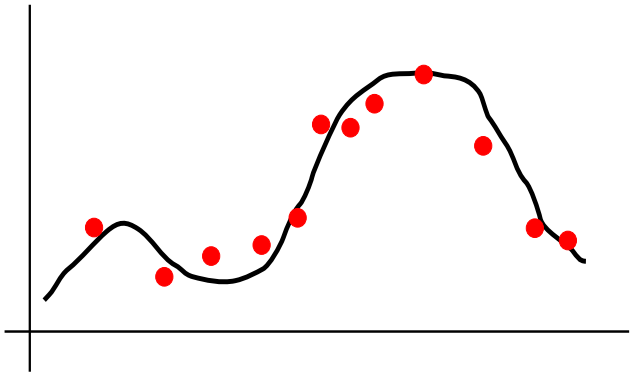
Valid covariance functions for multiple outputs (II)



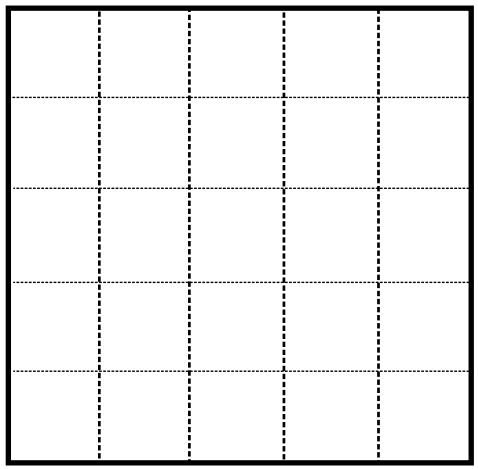
$$k^1(\mathbf{x}_i, \mathbf{x}_j)$$

$$\mathcal{D}^1 = \{(\mathbf{x}_i^1, y_i^1)|i = 1, \ldots, N_1\}$$

Valid covariance functions for multiple outputs (II)

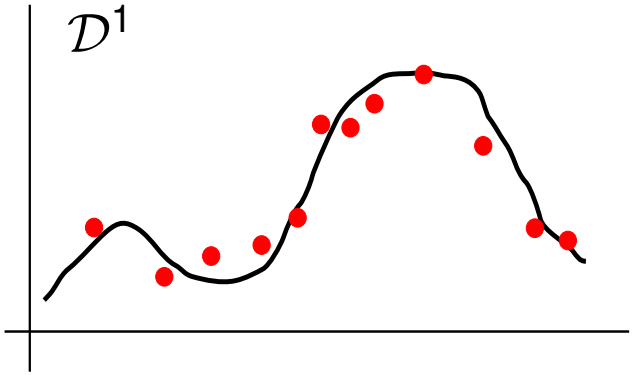


$$\rightarrow \quad k^1(\mathbf{x}_i, \mathbf{x}_j) \quad \rightarrow \quad \mathbf{K}^1 =$$

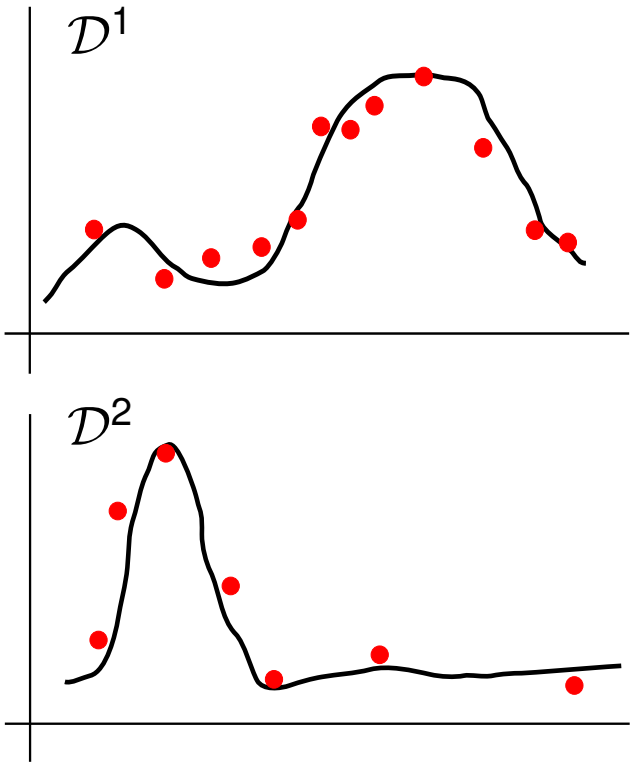


$$\mathcal{D}^1 = \{(\mathbf{x}_i^1, y_i^1) | i = 1, \ldots, N_1\}$$

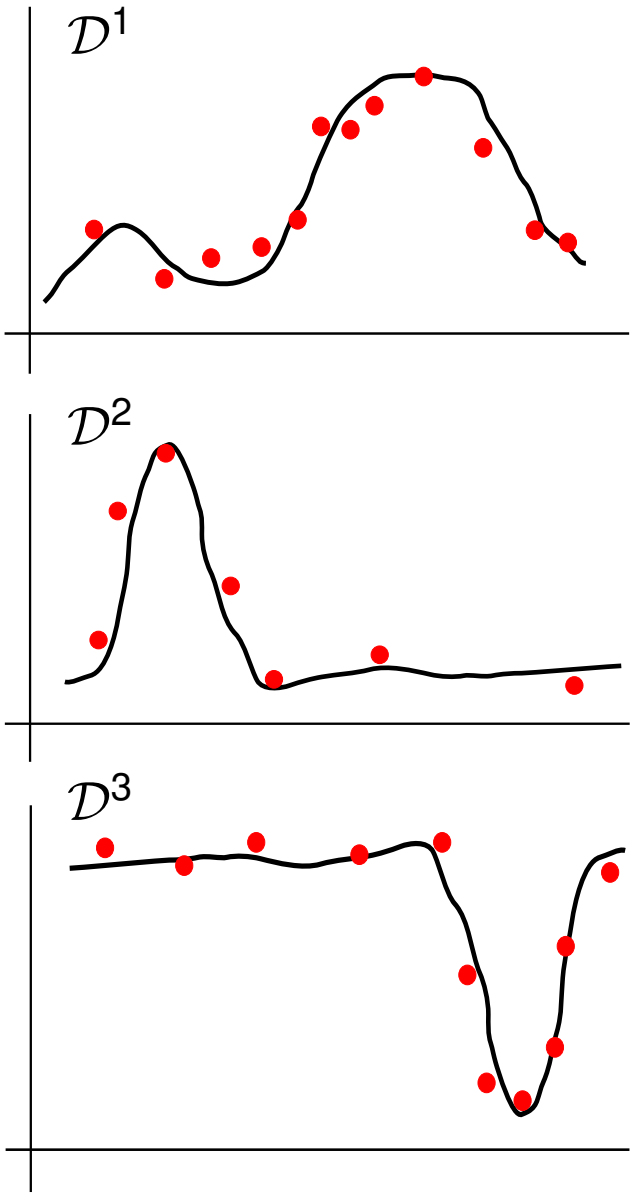
Valid covariance functions for multiple outputs (II)



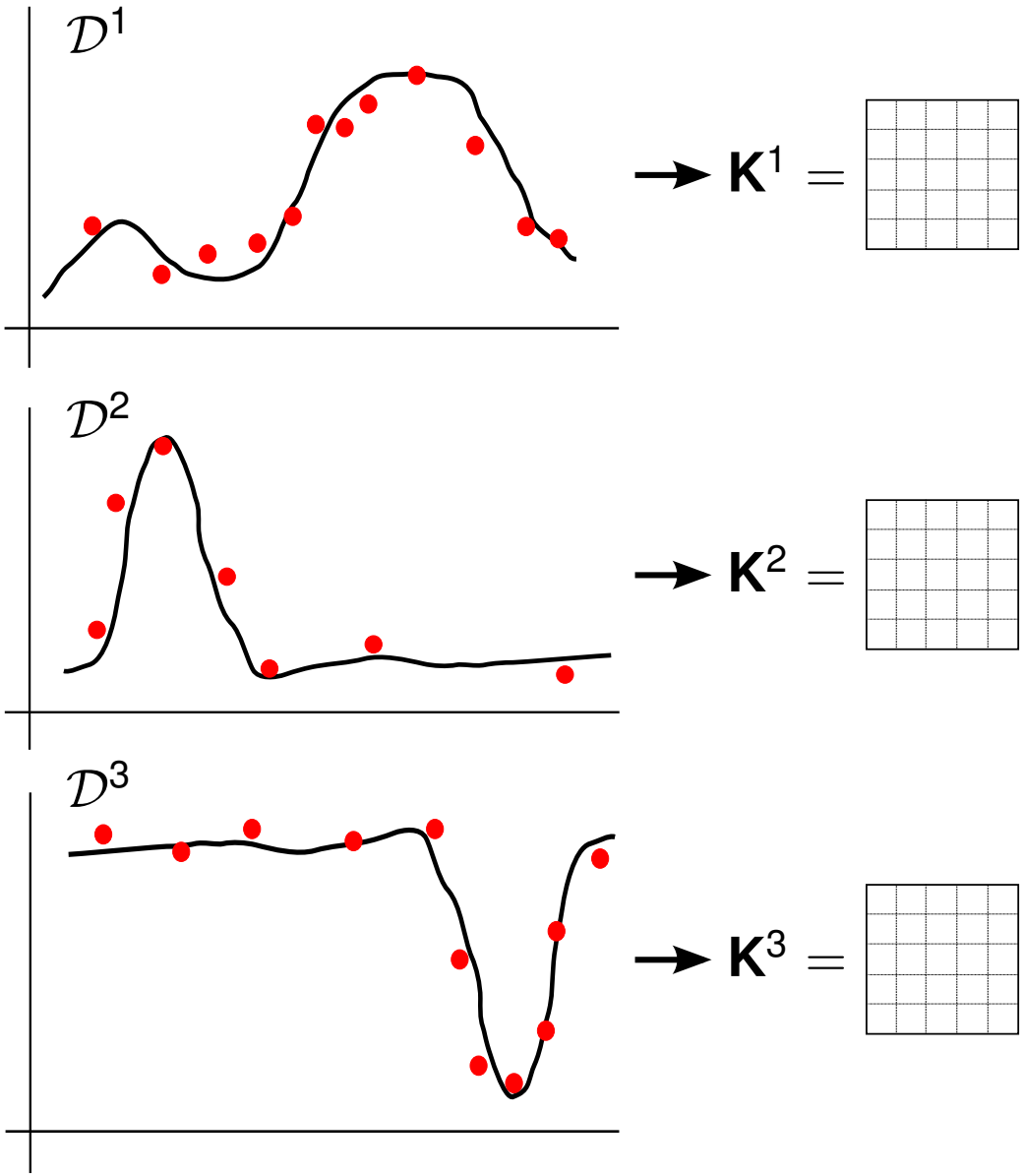
Valid covariance functions for multiple outputs (II)



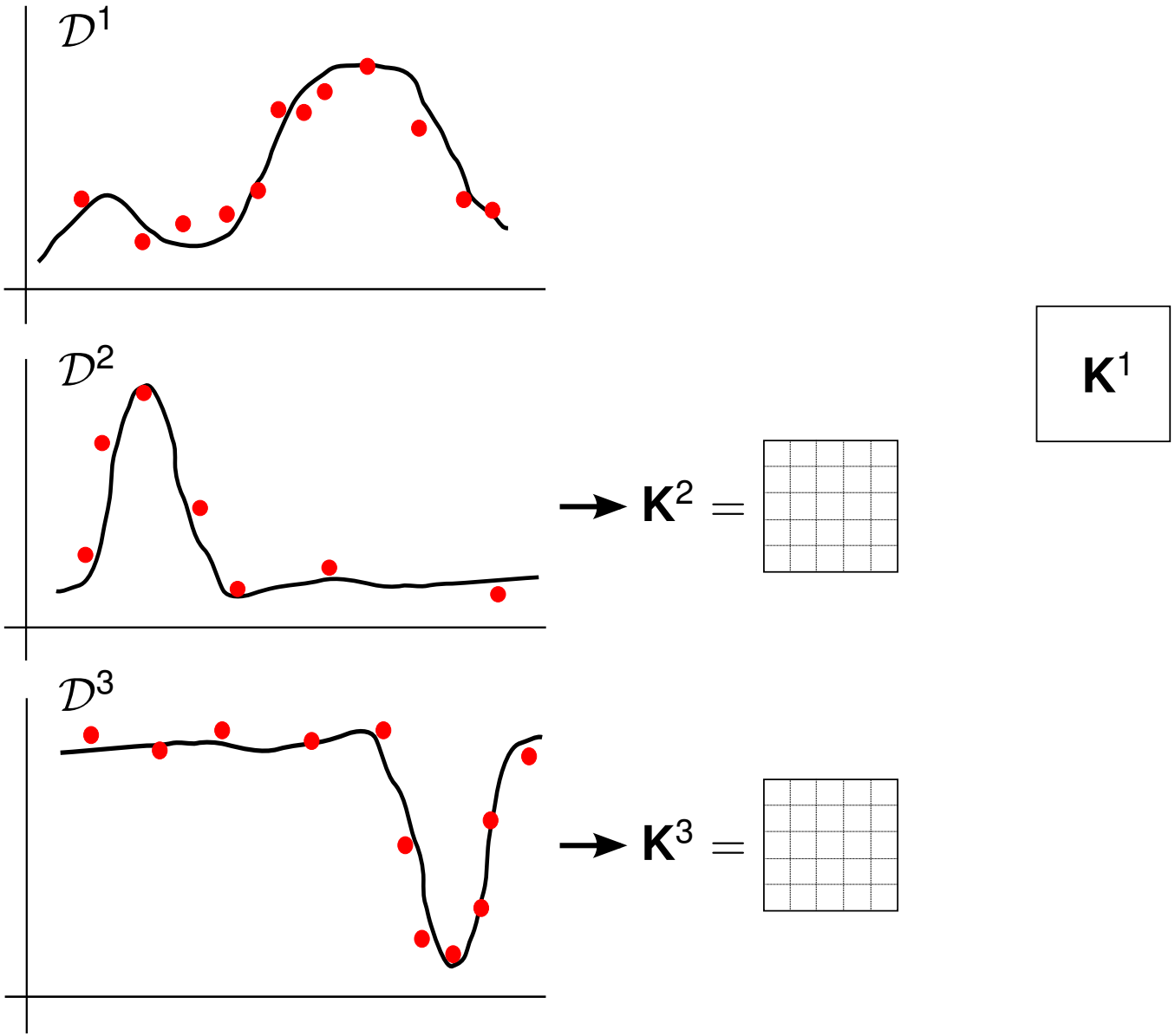
Valid covariance functions for multiple outputs (II)



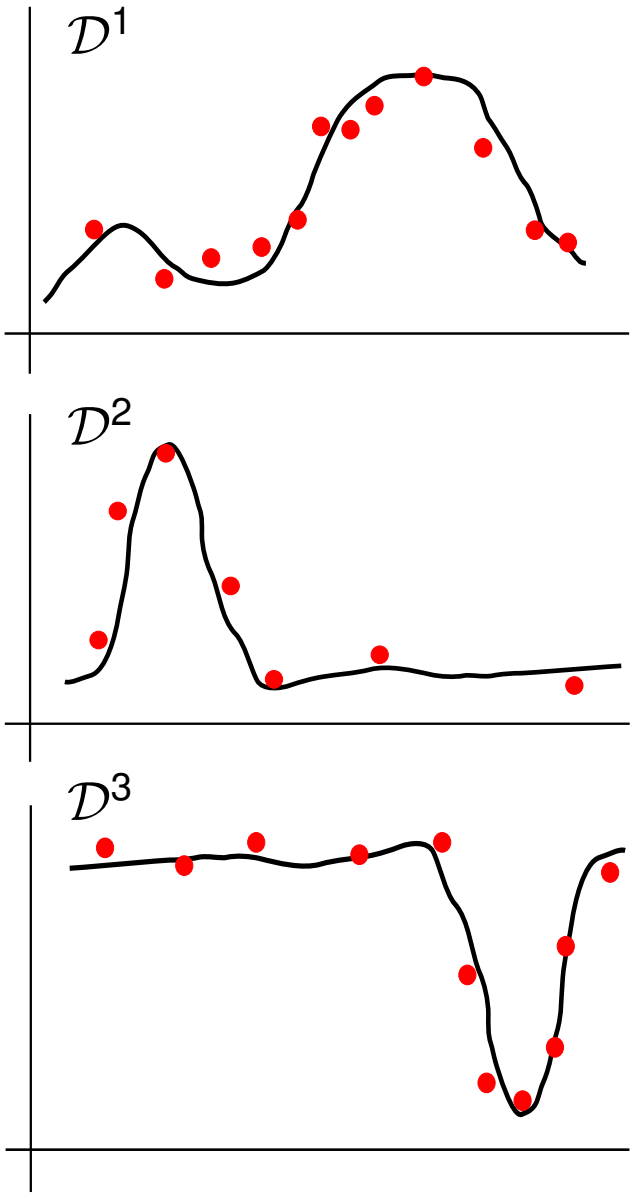
Valid covariance functions for multiple outputs (II)

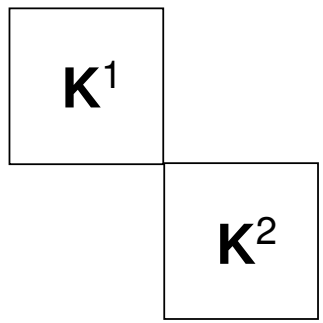


Valid covariance functions for multiple outputs (II)

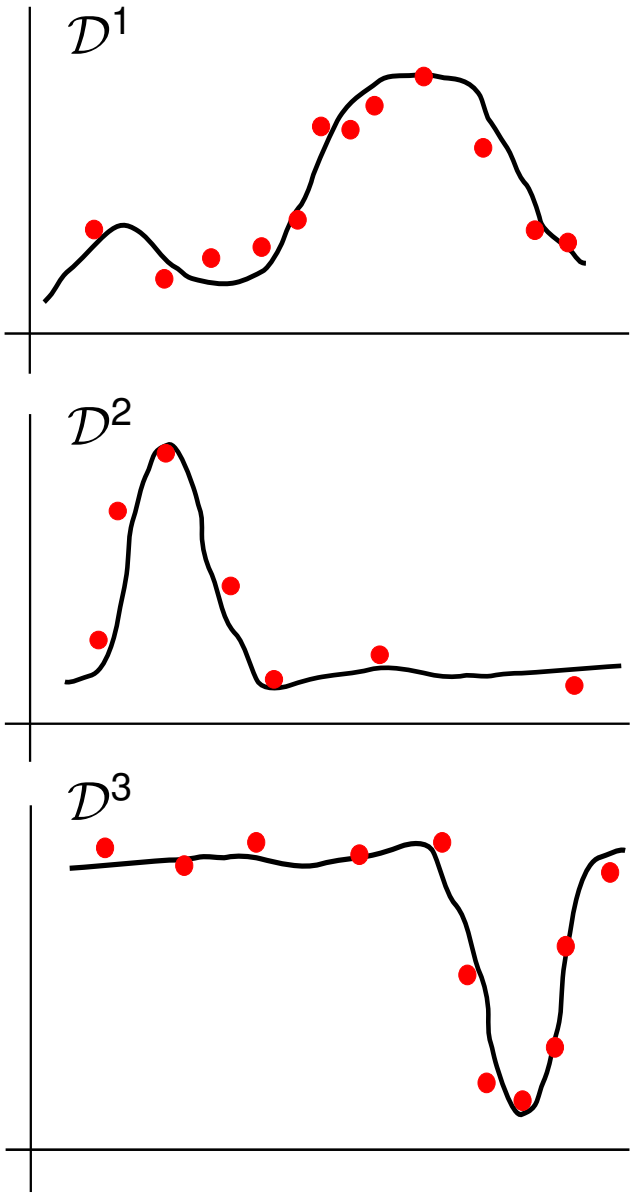


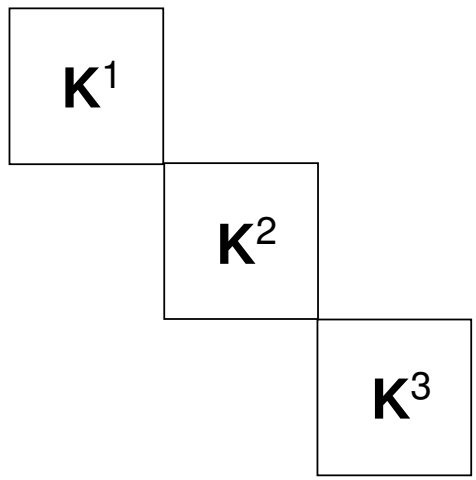
Valid covariance functions for multiple outputs (II)



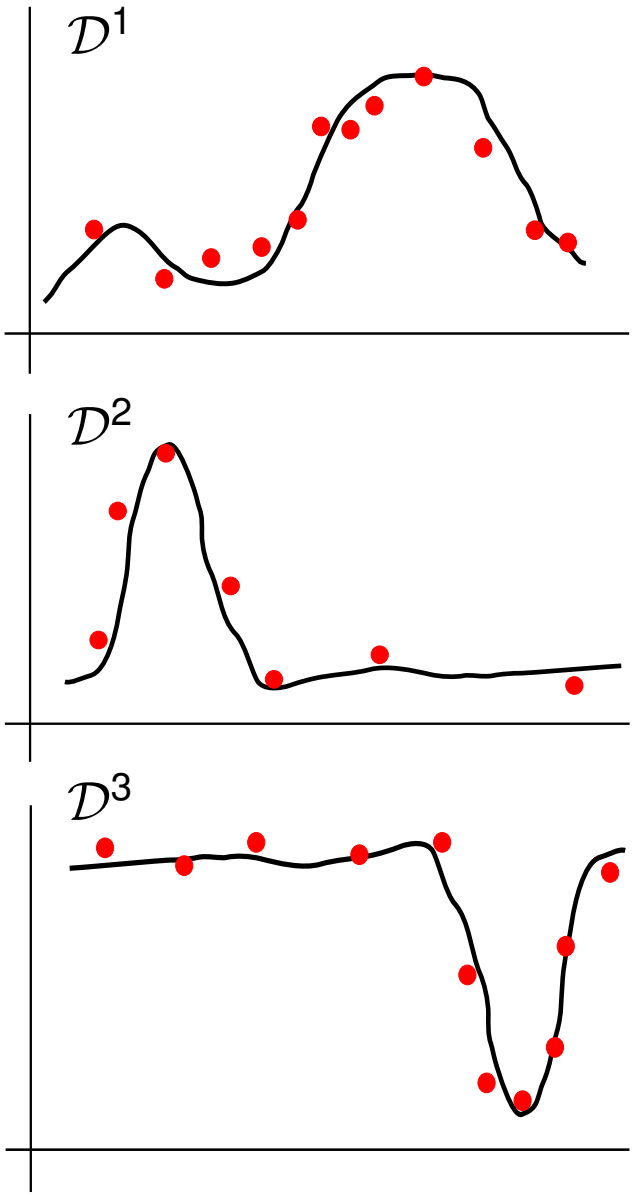


Valid covariance functions for multiple outputs (II)



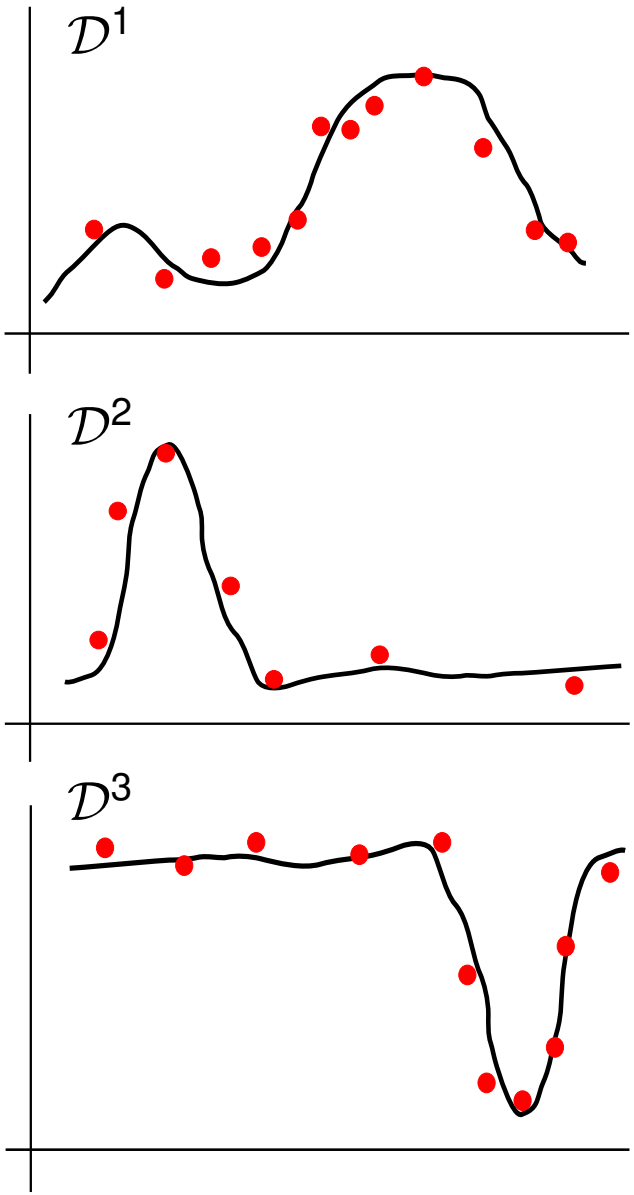


Valid covariance functions for multiple outputs (II)



$$\mathbf{K} = \begin{bmatrix} \mathbf{K}^1 & & \\ & \mathbf{K}^2 & \\ & & \mathbf{K}^3 \end{bmatrix}$$

Valid covariance functions for multiple outputs (II)



Joint covariance

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}^1 & ? & ? \\ ? & \mathbf{K}^2 & ? \\ ? & ? & \mathbf{K}^3 \end{bmatrix}$$

$\mathbf{K}$ be a valid covariance matrix

Process convolutions for multi-task learning

- Consider a set of functions $\{f_d(\mathbf{x})\}_{d=1}^{D}$ with $\mathbf{x} \in \mathbb{R}^{p\times 1}$.

- Each function can be expressed as

$$f_d(\mathbf{x}) = \int_{\mathcal{X}} G_d(\mathbf{x}-\mathbf{z})u(\mathbf{z})\mathrm{d}\mathbf{z} = G_d(\mathbf{x}) * u(\mathbf{x}).$$

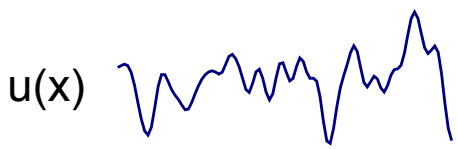
- If $u(\mathbf{x})$ is a Gaussian process (GP), $u(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, then $f_d(\mathbf{x})$ is also a GP.

- We could also include more latent processes $u_1(\mathbf{x}), u_2(\mathbf{x}), \ldots$

$$f_d(\mathbf{x}) = \sum_{q=1}^{Q} \int_{\mathcal{X}} G_{d,q}(\mathbf{x}-\mathbf{z})u_q(\mathbf{z})\mathrm{d}\mathbf{z},$$

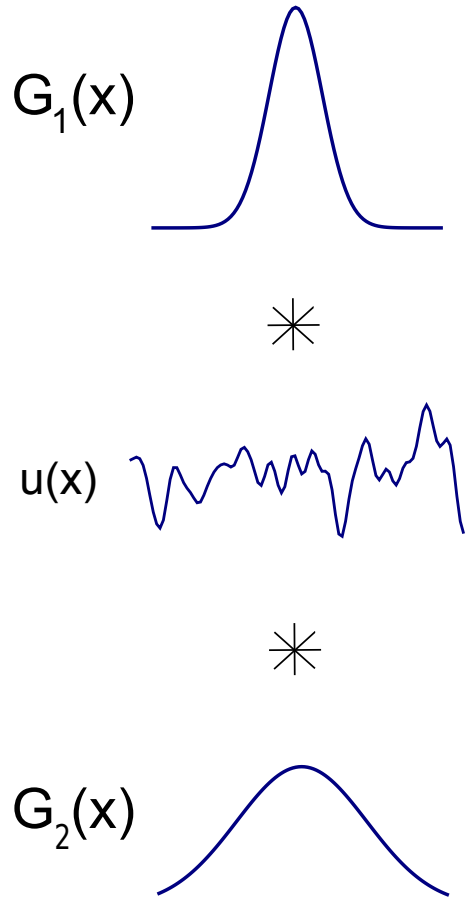
where $\mathrm{cov}[u_q(\mathbf{x}), u_{q'}(\mathbf{x}')] = k_q(\mathbf{x}, \mathbf{x}')\delta_{q,q'}$.

A pictorial representation



u(x): latent function.

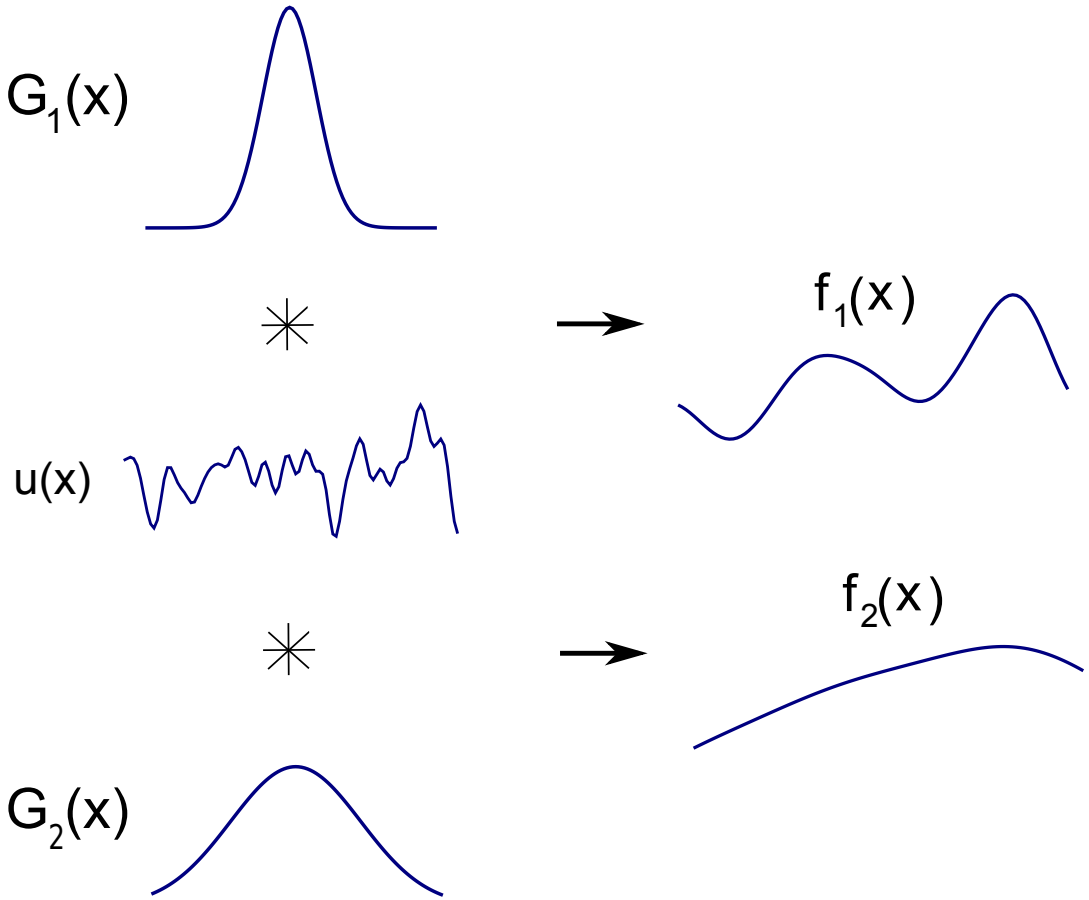
A pictorial representation



u(x): latent function.

G(x): smoothing kernel.

A pictorial representation

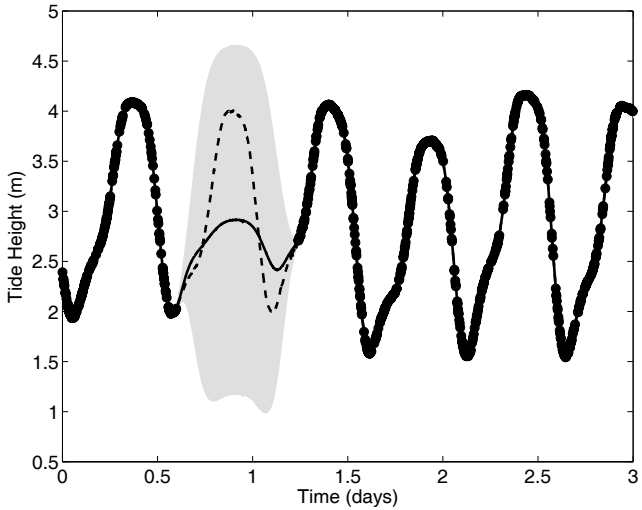


u(x): latent function.

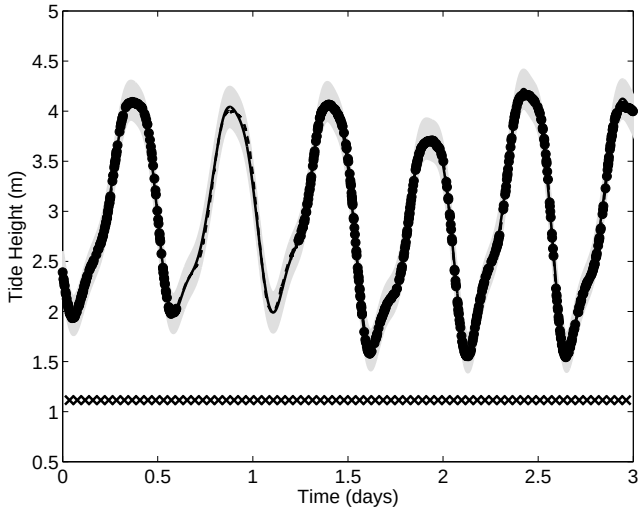
G(x): smoothing kernel.

f(x): output function.

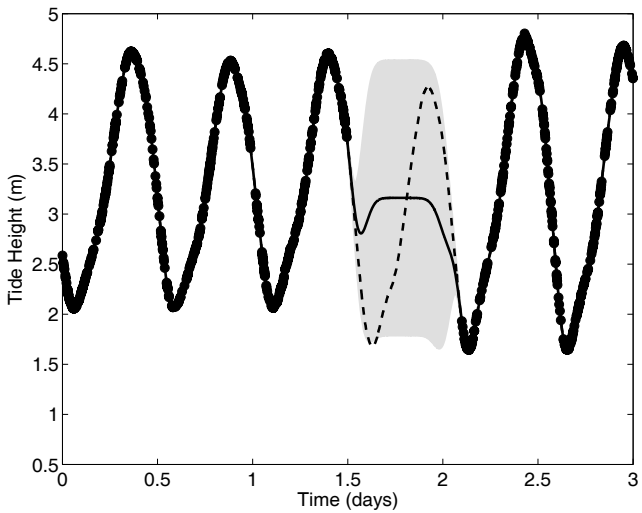
Example: Predicting tide height



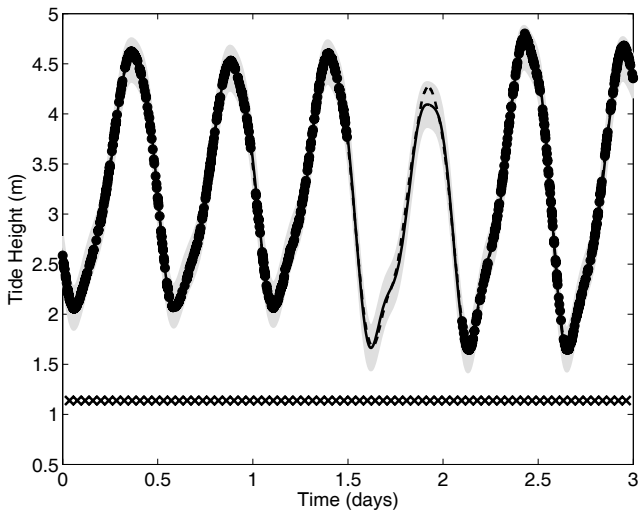
Bramblemet, independent



Bramblemet, multi-task



Cambermet, independent



Cambermet, multi-task

Human Behavior Data

Behav Ecol Sociobiol (2009) 63:1057–1066
DOI 10.1007/s00265-009-0739-0

ORIGINAL PAPER

Eigenbehaviors: identifying structure in routine

Nathan Eagle · Alex Sandy Pentland

Received: 12 September 2007 / Revised: 24 February 2009 / Accepted: 24 February 2009 / Published online: 7 April 2009

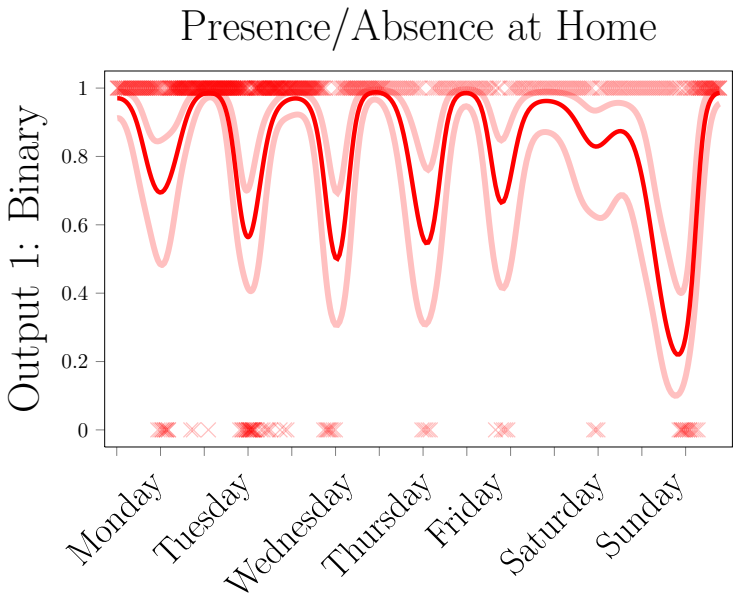
Abstract Longitudinal behavioral data generally contains a significant amount of structure. In this work, we identify the structure inherent in daily behavior with models that can accurately analyze, predict, and cluster multimodal data from individuals and communities within the social network of a population. We represent this behavioral structure by the principal components of the complete behavioral dataset, a set of characteristic vectors we have termed eigenbehaviors. In our model, an individual's behavior over a specific day can be approximated by a weighted sum of his or her primary eigenbehaviors. When these weights are calculated halfway through a day, they can be used to predict the day's remaining behaviors with 79% accuracy for our test subjects. Additionally, we demonstrate the potential for this dimensionality reduction technique to infer community affiliations within the subjects' social network by clustering individuals into a "behavior space" spanned by a set of their aggregate eigenbehaviors. These behavior spaces make it possible to determine the behavioral similarity between both individuals and groups, enabling 96% classification accuracy of community affiliations within the population-level social network. Additionally, the distance between individuals in the behavior space can be used as an estimate for relational ties such as friendship, suggesting strong behavioral homophily amongst the subjects. This approach capitalizes on the large amount of rich data previously captured during the Reality Mining study from mobile phones continuously logging location, proximate phones, and communication of 100 subjects at MIT over the course of 9 months. As wearable sensors continue to generate these types of rich, longitudinal datasets, dimensionality reduction techniques such as eigenbehaviors will play an increasingly important role in behavioral research.

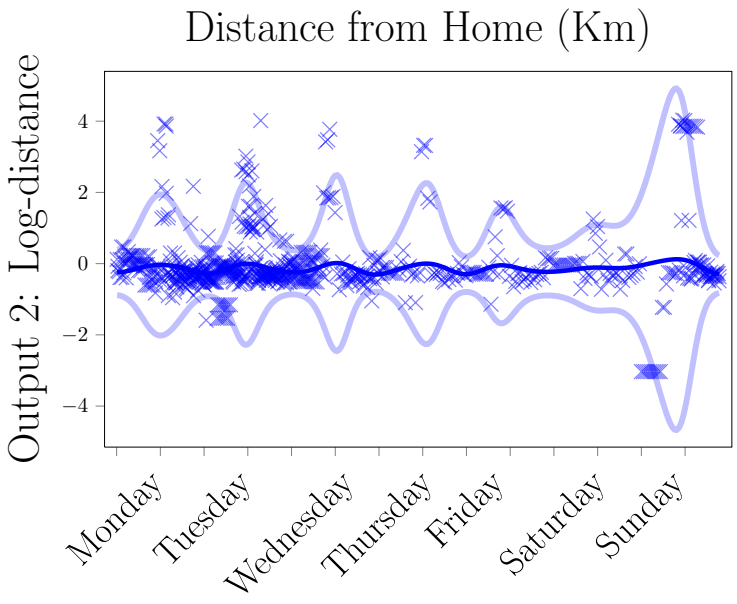
Keywords Behavioral modeling · Machine learning · Eigendecomposition

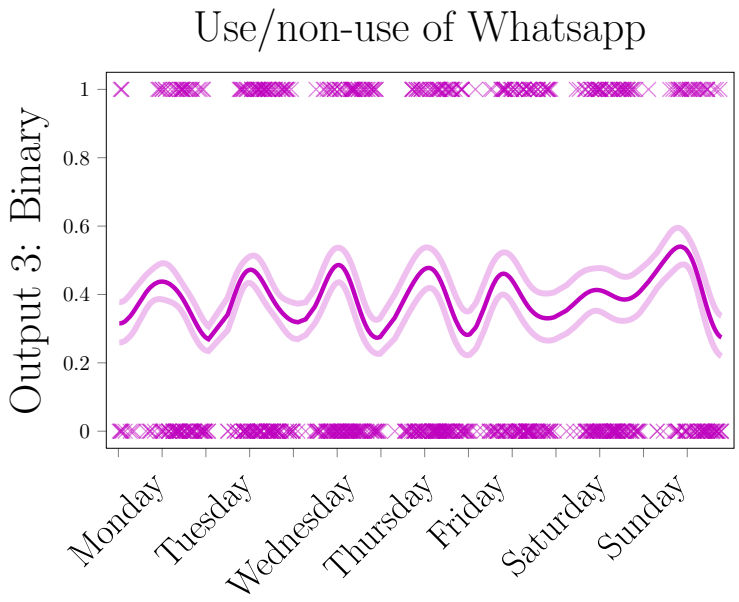
Introduction

While discrete observations of an individual's idiosyncratic behavior can appear almost random, typically there are repeating and easily identifiable routines in every person's

Human Behavior Data







Heterogeneous Multi-output Gaussian Process Prediction

Pablo Moreno-Muñoz[1] **Antonio Artés-Rodríguez**[1] **Mauricio A. Álvarez**[2]
[1]Dept. of Signal Theory and Communications, Universidad Carlos III de Madrid, Spain
[2]Dept. of Computer Science, University of Sheffield, UK
{pmoreno,antonio}@tsc.uc3m.es, mauricio.alvarez@sheffield.ac.uk

Abstract

We present a novel extension of multi-output Gaussian processes for handling heterogeneous outputs. We assume that each output has its own likelihood function and use a vector-valued Gaussian process prior to jointly model the parameters in all likelihoods as latent functions. Our multi-output Gaussian process uses a covariance function with a linear model of coregionalisation form. Assuming conditional independence across the underlying latent functions together with an inducing variable framework, we are able to obtain tractable variational bounds amenable to stochastic variational inference. We illustrate the performance of the model on synthetic data and two real datasets: a human behavioral study and a demographic high-dimensional dataset.

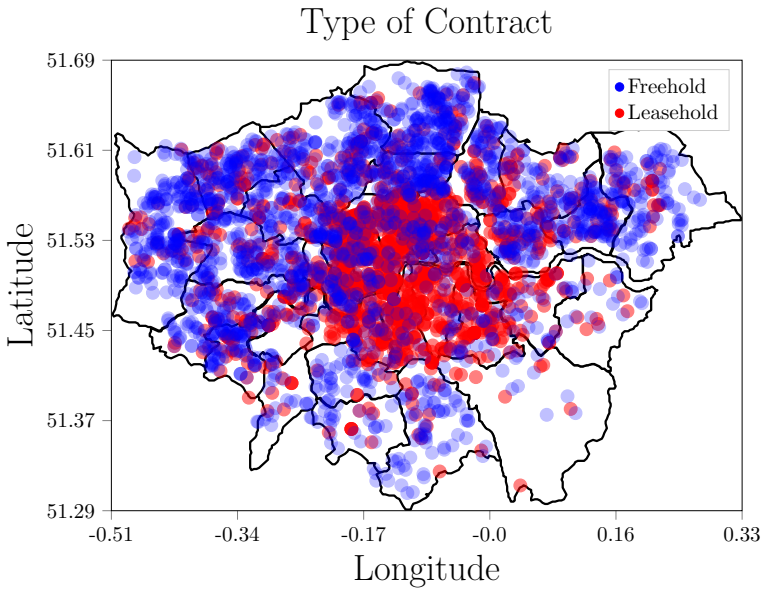
32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

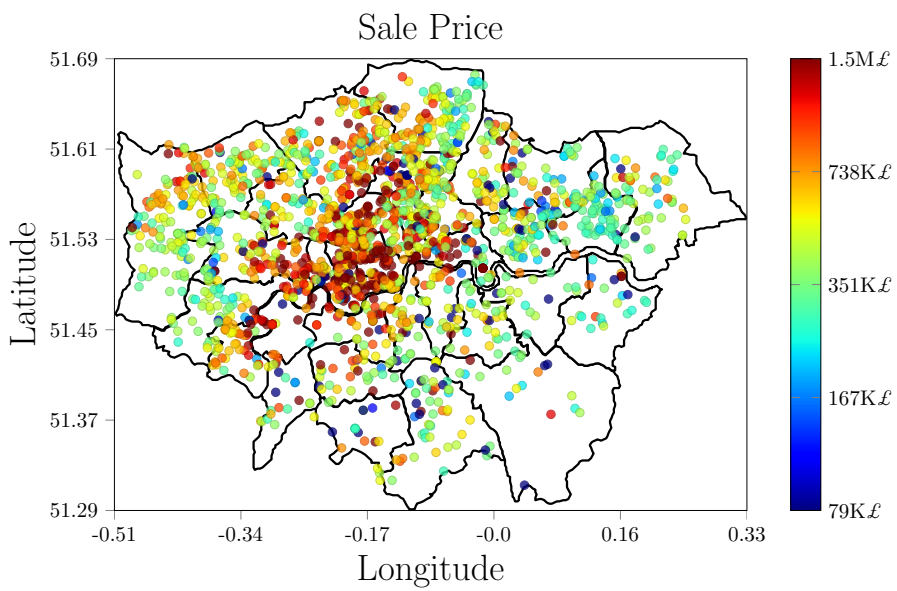
Human Behavior Data

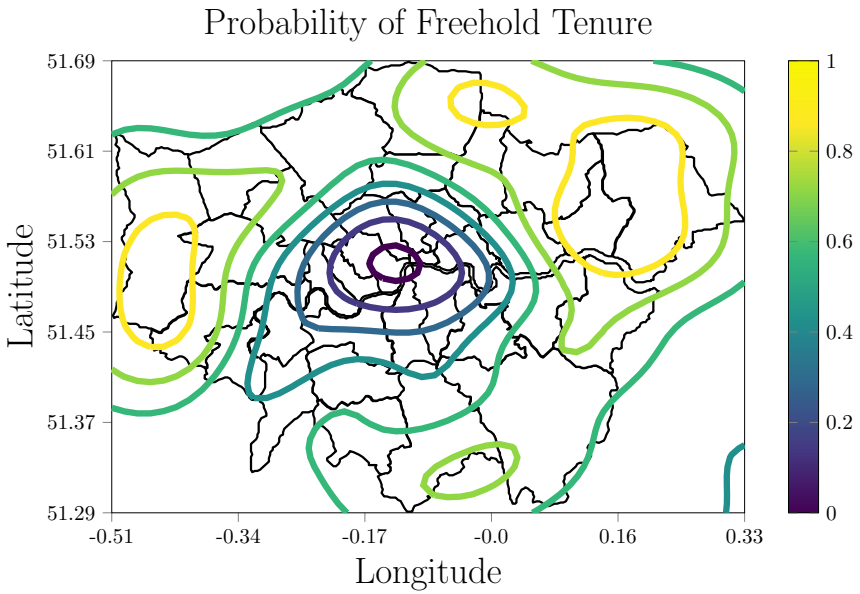
Method	Bernoulli	Heteroscedastic	Bernoulli	Average
HetMOGP	$\mathbf{2.24 \pm 0.21}$	$\mathbf{6.09 \pm 0.21}$	5.41 ± 0.05	$\mathbf{13.74 \pm 0.41}$
ChainedGP	2.43 ± 0.30	7.29 ± 0.12	$\mathbf{5.19 \pm 0.81}$	14.91 ± 1.05

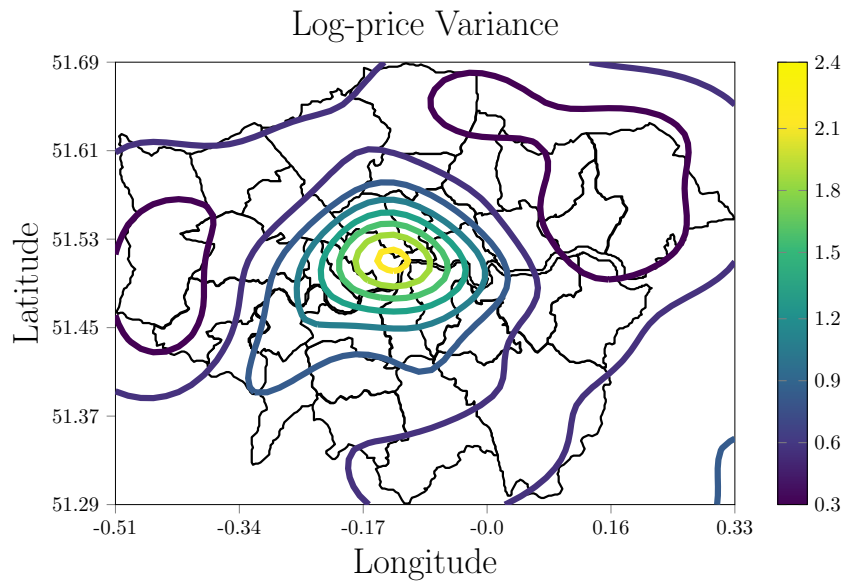
Test-NLPD ($\times 10^{-2}$)

London house prices: type of contract and sale price









London house prices: type of contract and sale price

	Average Test-NLPD
HetMOGP	$\mathbf{0.1646 \pm 0.0153}$
ChainedGP	0.2144 ± 0.0402

Multi-task GPs and change of support

- Many datasets in fields like ecology, epidemiology, remote sensing, sensor networks and demography appear naturally aggregated.
- In sensor networks, correlated variables are measured at different resolutions or scales.
- Let $\{f_d(\upsilon)\}_{d=1}^{D}$ be a set of tasks where each task is defined at a different support υ.
- Each task $f_d(\upsilon)$ is represented as

$$f_d(\upsilon) = \sum_{q=1}^{Q} \frac{a_{d,q}}{|\upsilon|} \int_{\mathbf{z} \in v} u_q(\mathbf{z}) d\mathbf{z},$$

where the coefficients $a_{d,q}$ weight the contribution of each integral term to $f_d(\upsilon)$.

Multi-task Learning for Aggregated Data using Gaussian Processes

Fariba Yousefi **Michael Thomas Smith** **Mauricio A Álvarez**

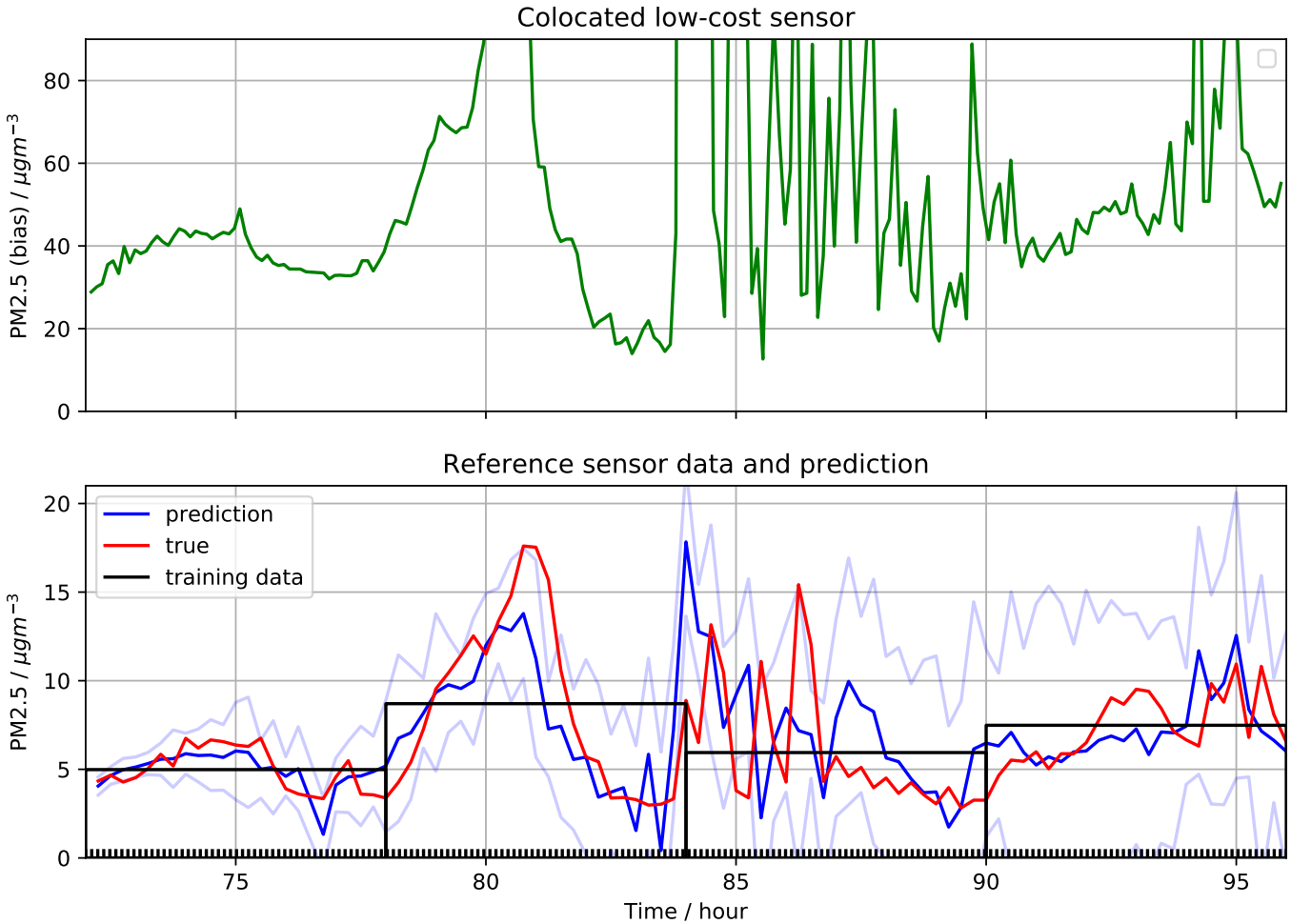
Department of Computer Science, University of Sheffield
{f.yousefi, m.t.smith, mauricio.alvarez}@sheffield.ac.uk

Abstract

Aggregated data is commonplace in areas such as epidemiology and demography. For example, census data for a population is usually given as averages defined over time periods or spatial resolutions (cities, regions or countries). In this paper, we present a novel multi-task learning model based on Gaussian processes for joint learning of variables that have been aggregated at different input scales. Our model represents each task as the linear combination of the realizations of latent processes that are integrated at a different scale per task. We are then able to compute the cross-covariance between the different tasks either analytically or numerically. We also allow each task to have a potentially different likelihood model and provide a variational lower bound that can be optimised in a stochastic fashion making our model suitable for larger datasets. We show examples of the model in a synthetic example, a fertility dataset and an air pollution prediction application.

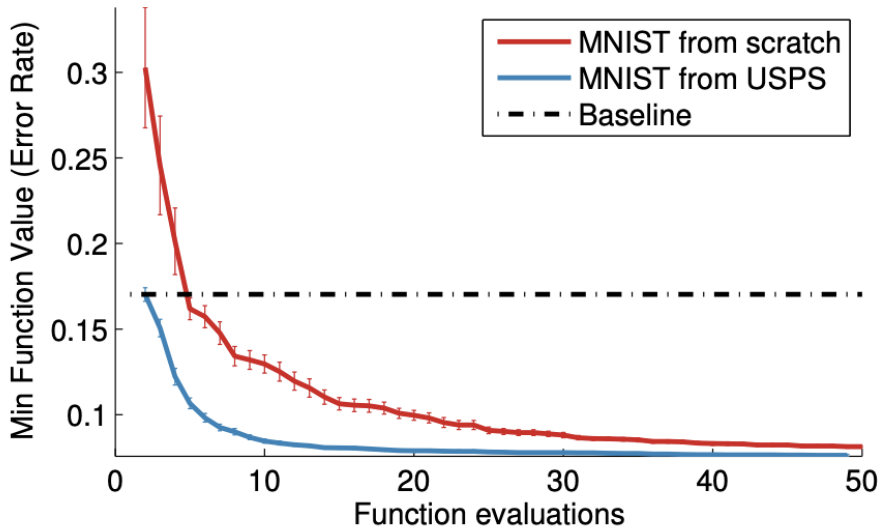
33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Air pollution monitoring network



Applications multi-task Bayes Opt

- Cold start problem (hyperparameter tuning in logistic regression)



- Exploiting correlations among the folds in k-folds cross-validation to do fast cross-validation.

Contents

Parallel opt (I)

- The goal of Bayesian optimization is to keep the number of evaluations of $f(\mathbf{x})$ as low as possible.
- In high-dimensional and or otherwise complex problems, the number of required evaluations can still be considerable.
- Parallel approaches arise as the natural solution to circumvent the computational bottleneck around these evaluations of $f(\mathbf{x})$.

Parallel opt (II)

- In this context, there are current observations $\mathcal{D}_n = \{(\mathbf{x}_n, y_n)\}$ and pending experiments $\mathcal{D}_p = \{\mathbf{x}_p\}_{p=1}^{J}$.

- There are different ways in which it is possible to impute a set of experimental outcomes $\tilde{\mathcal{D}}_p = \{(\mathbf{x}_p, \tilde{y}_p)\}_{p=1}^{J}$

- We then perform a step of Bayesian optimization using the augmented data set $\mathcal{D}' = \mathcal{D}_n \cup \tilde{\mathcal{D}}_p$.

- Sequential simulation algorithm

1: **input**: dataset $\mathcal{D}_n$, number of pending experiments J, AF α
2: $\mathcal{D}' \leftarrow \mathcal{D}_n$
3: **for** $p = 1, 2, \ldots, J$ **do**
4: $\quad \mathbf{x}_p \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x}; \mathcal{D}')$
5: $\quad \tilde{y}_p \leftarrow$ SIMULATE-OBSERVATION$(\mathbf{x}_p, \mathcal{D}')$
6: $\quad \mathcal{D}' \leftarrow \mathcal{D}' \cup \{(\mathbf{x}_p, \tilde{y}_p)\}$
7: **end for**

Parallel opt (III)

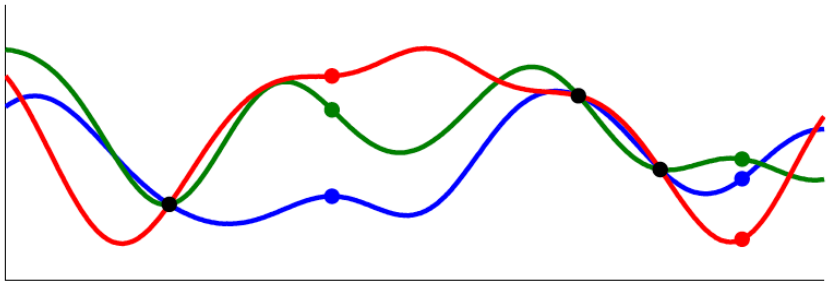
- Snoek et al. (2012) used an approach where a set of S fantasies are sampled for each unfinished experiment from the full GP posterior predictive distribution.

- These fantasies are then combined to estimate the following parallel integrated acquisition function

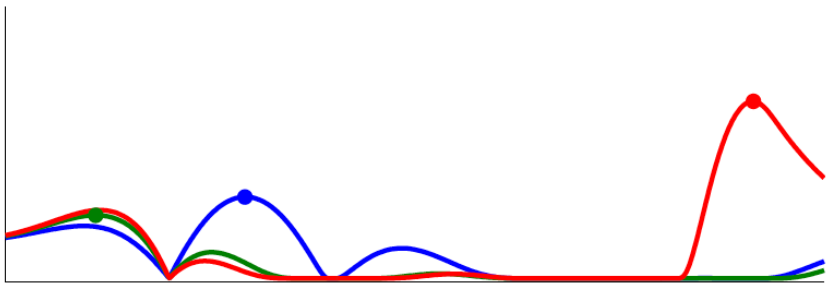
$$\begin{aligned}\alpha\left(\mathbf{x}; \mathcal{D}_n, \mathcal{D}_p\right) &= \int_{\mathbb{R}^J} \alpha\left(\mathbf{x}; \mathcal{D}_n \cup \tilde{\mathcal{D}}_p\right) p\left(y_{1:J} | \mathcal{D}_n\right) \mathrm{d}y_{1:J} \\ &\approx \frac{1}{S} \sum_{s=1}^{S} \alpha\left(\mathbf{x}; \mathcal{D}_n \cup \tilde{\mathcal{D}}_p^{(s)}\right),\end{aligned}$$

where $\tilde{\mathcal{D}}_p^{(s)} \sim p\left(y_{1:J} | \mathcal{D}_n\right)$.

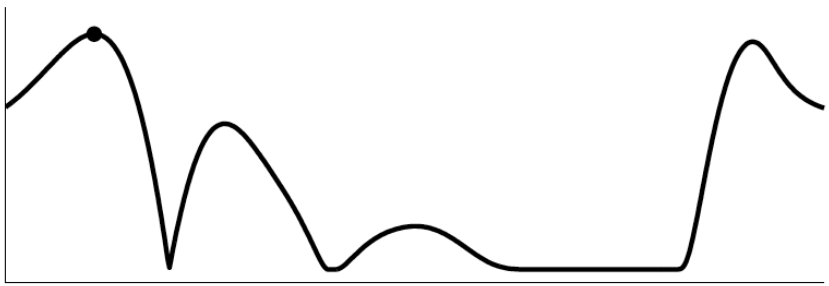
Parallel opt (IV)



(a) Posterior samples after three data



(b) Expected improvement under three fantasies



(c) Expected improvement across fantasies

Batch opt (I)

- In Batch Bayes Opt, we want to simultaneously propose a set of candidates $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_J$.
- Different acquisition functions have been extended to account for this setting, including batch EI and batch PI.
- These are rather involved for batch sizes greater than ten.

Batch opt (II)

- TS with J samples.
- UCB with J different values for β_n.
- Batch construction by local penalization (González et al., 2016)
  - select the first point by maximizing the acquisition function $\alpha(\mathbf{x}, \mathcal{D})$ to find $\mathbf{x}_1$.
  - incorporate a multiplicative penalty $\psi(\mathbf{x}; \mathbf{x}_1)$ into the acquisition function discouraging future batch members from being in a neighborhood of the initial point.
  - select $\mathbf{x}_2$ by maximizing $\alpha(\mathbf{x}, \mathcal{D})\psi(\mathbf{x}; \mathbf{x}_1)$.

Contents

Other extensions of Bayes Opt (I)

- **Conditional spaces**
  - Some variables will only influence the function being optimized when other variables take on certain values.
  - These are called conditional variables and are said to be active or inactive.
  - For example, when the function involves selecting between different algorithms as well as optimizing their hyperparameters.

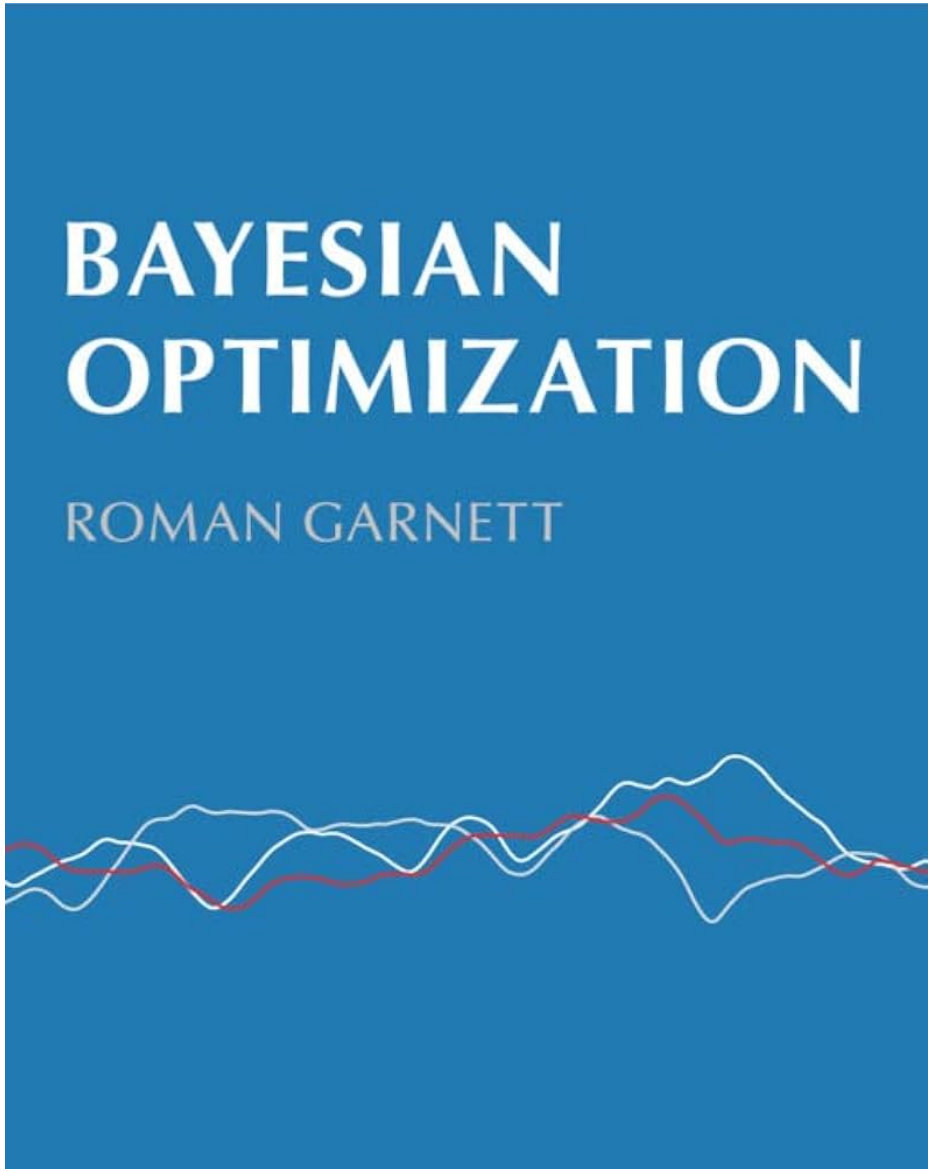
- **Constrained Bayesian optimization**
  - This is the case where certain regions of the design space $\mathcal{X}$ are invalid.
  - Several approaches deal with this problem by altering the acquisition function itself.

Other extensions of Bayes Opt (II)

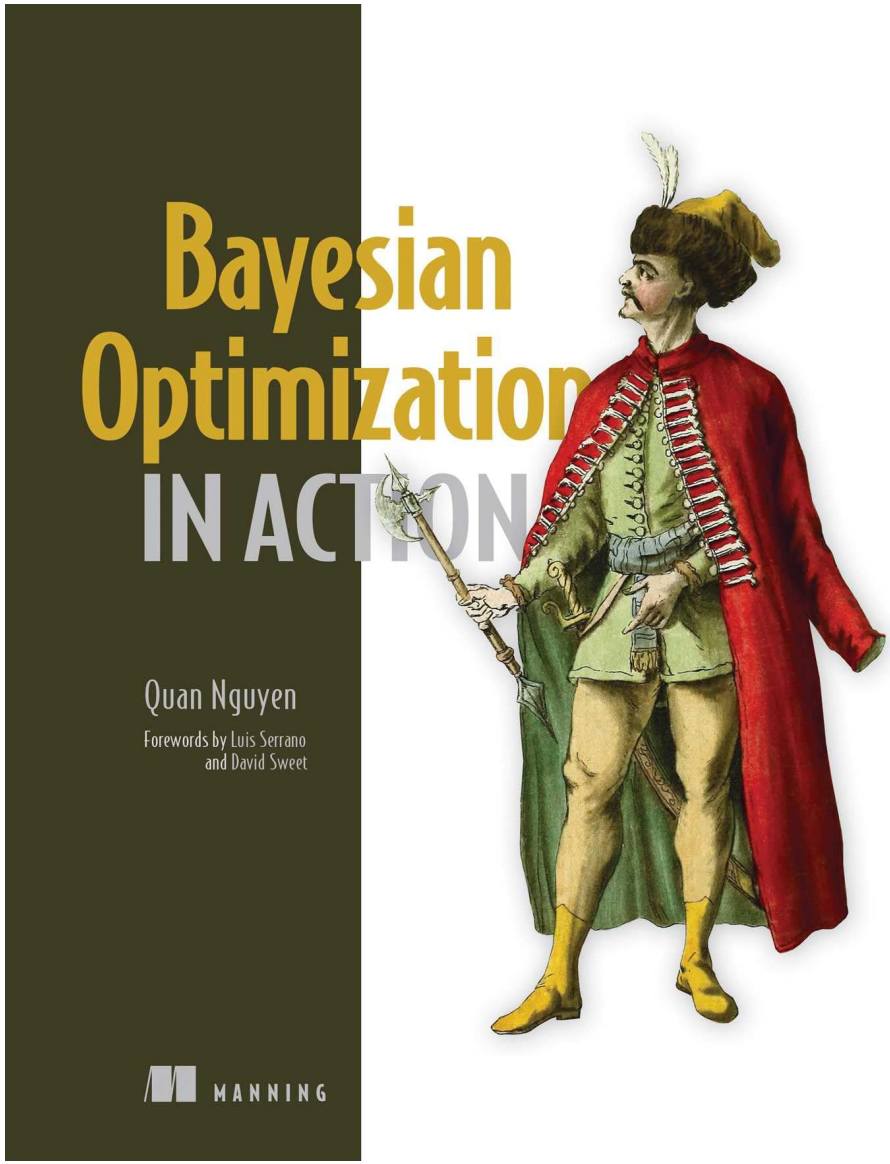
- **Cost sensitivity**
  - In some cases, each function evaluation may return both a value along with an associated cost.
  - In other words, it may be more expensive to evaluate the function in some parts of the design space than others.
  - If there is a limited budget, then the search should be biased toward low-cost areas.

Contents

Bayesian Optimization



Bayes Opt in Action



References I

David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/6c990b7aca7bc7058f5e98ea909e924b-Paper.pdf.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil D. Lawrence. Batch bayesian optimization via local penalization. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, volume 51 of *Proceedings of Machine Learning Research*, pages 648–657. PMLR, 2016. URL http://proceedings.mlr.press/v51/gonzalez16a.html.

Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(57):1809–1837, 2012. URL http://jmlr.org/papers/v13/hennig12a.html.

José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions, 2014.

Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian optimization performs great in high dimensions. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 20793–20817. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/hvarfner24a.html.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 2951–2959. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

References II

Julien Villemonteix, Emmanuel Vazquez, and Eric Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, 44(4):509, 2008. doi: 10.1007/s10898-008-9354-2. URL https://doi.org/10.1007/s10898-008-9354-2.

Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando De Freitas. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 1778–1784. AAAI Press, 2013. ISBN 9781577356332.

James Wilson, Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc Deisenroth. Efficiently sampling functions from Gaussian process posteriors. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 10292–10302. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/wilson20a.html.