

# Bayesian optimization

Mauricio A. Álvarez, PhD

Curso de entrenamiento ArcelorMittal

# Contents

Introduction

Applications

Acquisition functions

Practical considerations

Exotic “Bayes Opt”

# Contents

Introduction

Applications

Acquisition functions

Practical considerations

Exotic “Bayes Opt”

# Bayes opt

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

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

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  $\{\mathbf{x} \in \mathbb{R}^d : a_i \leq x_i \leq b_i\}$ .
  - the  $d$ -dimensional simplex  $\{\mathbf{x} \in \mathbb{R}^d : \sum_i x_i = 1\}$ .
- 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 problem

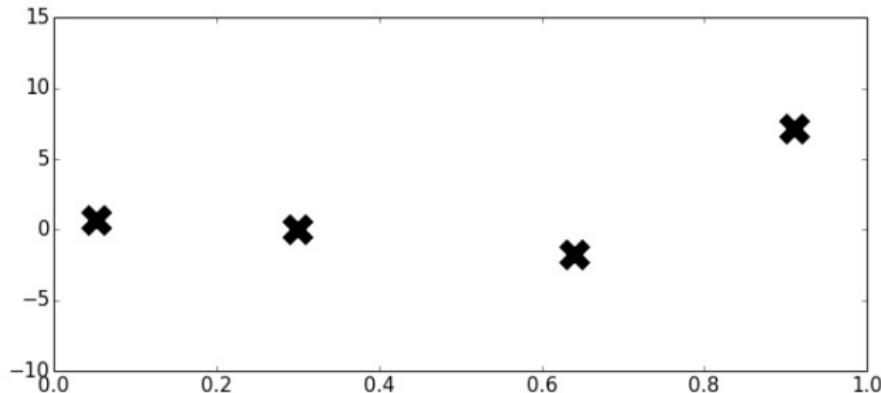
$$\mathbf{x}^* = \arg \max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x}).$$

- ❑ 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)

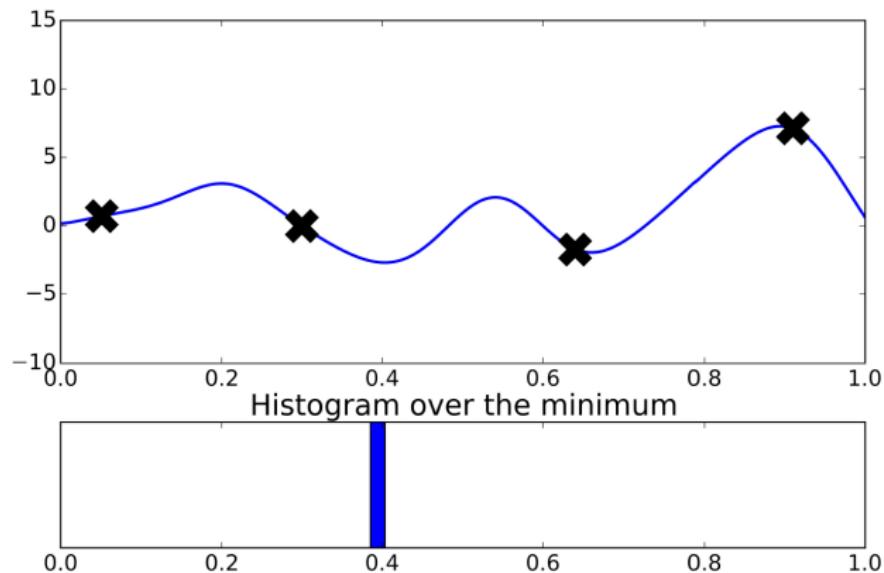
- The Bayesian optimization framework has two key ingredients.
- The first ingredient is a probabilistic surrogate model
  - a prior distribution that captures our beliefs about the behavior of the unknown objective function.
  - an observation model that describes the data generation mechanism.
- We 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?

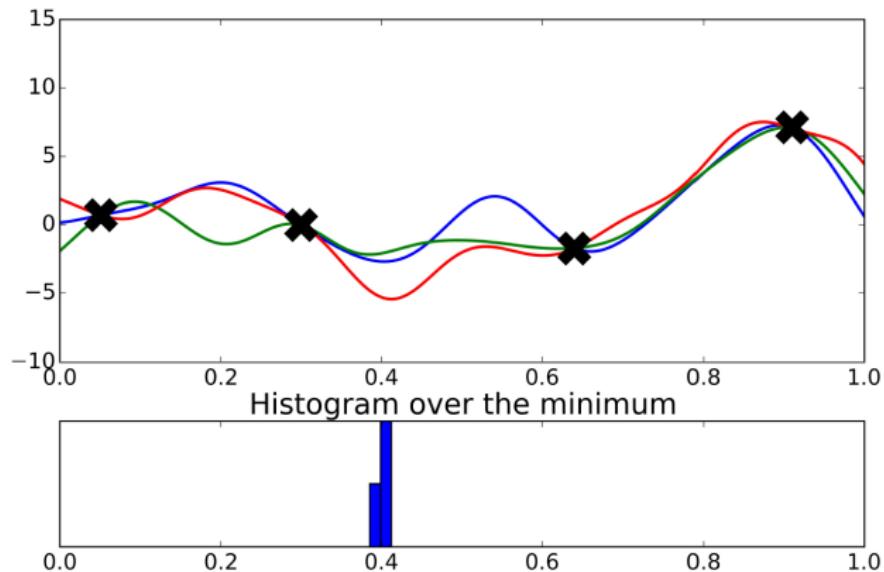


Where is the minimum of  $f$ ?  
Where should the take the next evaluation?

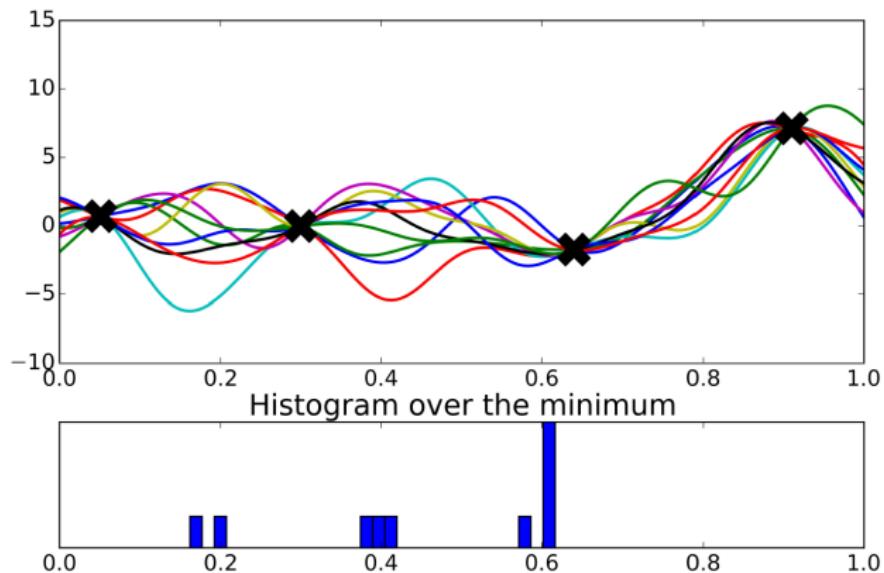
# How can we use the uncertainty in the GP?



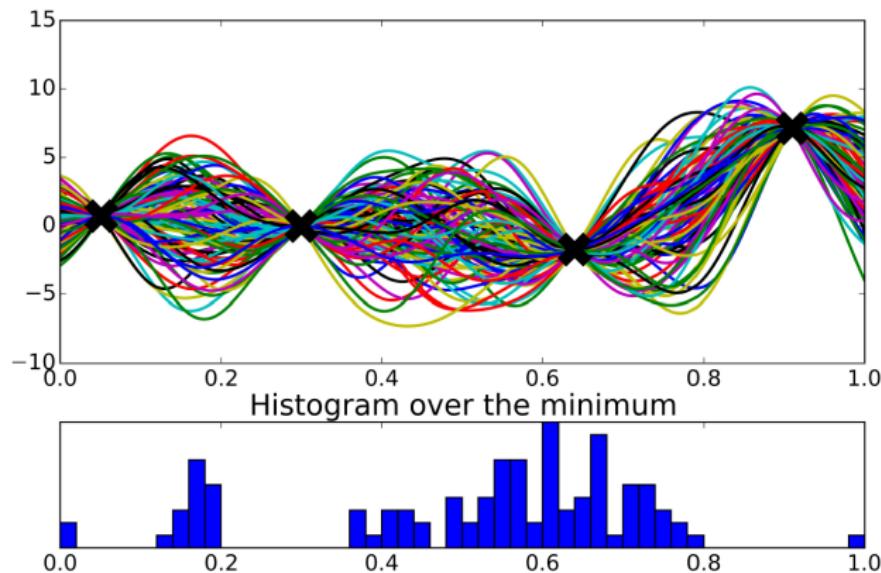
# How can we use the uncertainty in the GP?



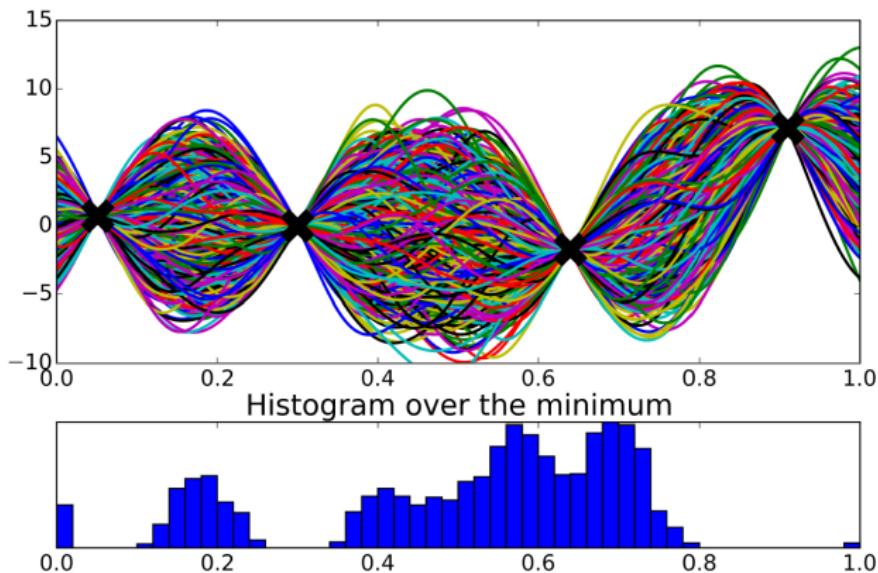
# How can we use the uncertainty in the GP?



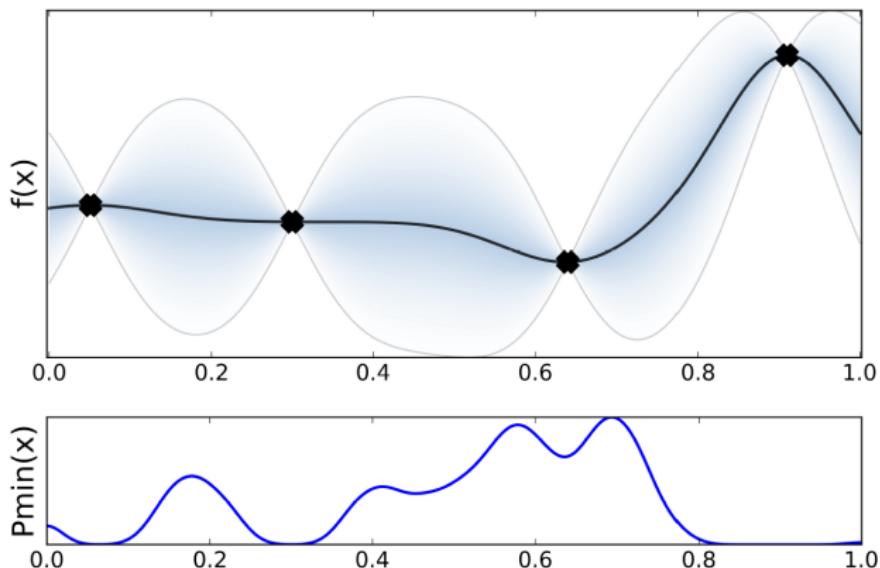
# How can we use the uncertainty in the GP?



# How can we use the uncertainty in the GP?



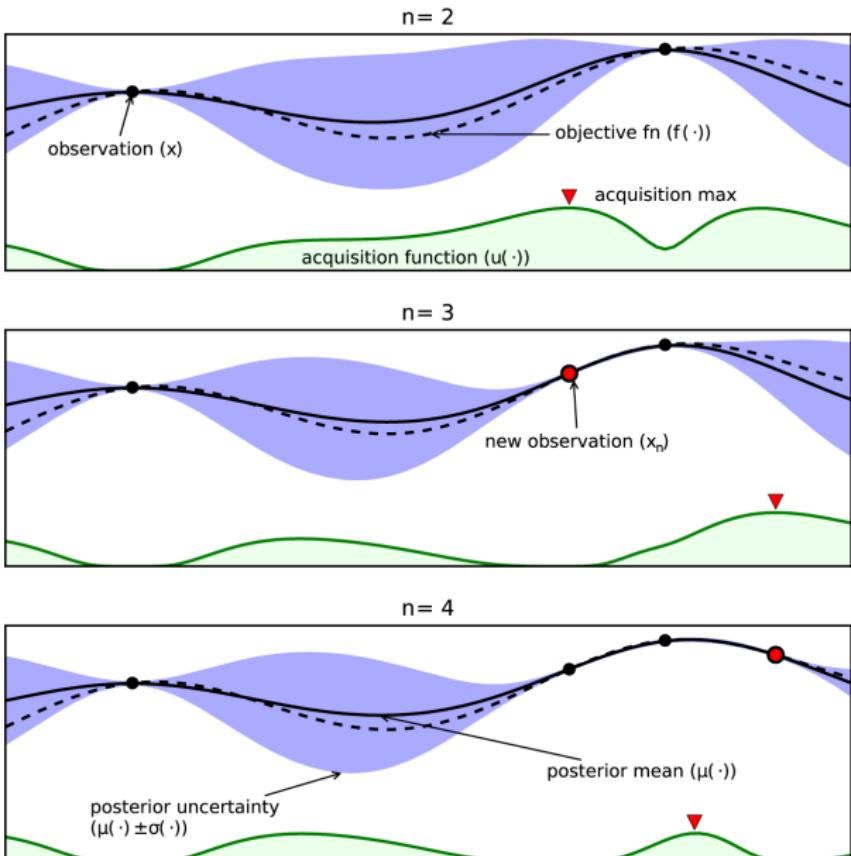
# How can we use the uncertainty in the GP?



# 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 over  $\mathbf{x}$ , 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



# Contents

Introduction

Applications

Acquisition functions

Practical considerations

Exotic “Bayes Opt”

## 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](#)[Example](#)[Manual](#)[License](#)[Citing auto-sklearn](#)[Contributing](#)

## auto-sklearn

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

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

*auto-sklearn* frees a machine learning user from algorithm selection and hyperparameter tuning. It leverages recent advantages in *Bayesian optimization*, *meta-learning* and *ensemble construction*. Learn more about the technology behind *auto-sklearn* by reading our paper published at [NIPS 2015](#).

### NEW: Auto-sklearn 2.0

Auto-sklearn 2.0 includes latest research on automatically configuring the AutoML system itself and contains a multitude of improvements which speed up the fitting the AutoML system.

*auto-sklearn 2.0* works the same way as regular *auto-sklearn* and you can use it via

```
>>> from autosklearn.experimental.askl2 import AutoSklearn2Classifier
```

A paper describing our advances is available on [arXiv](#).

# Contents

Introduction

Applications

Acquisition functions

Practical considerations

Exotic “Bayes Opt”

# Exploration vs exploitation

**Welcome to El Jalapeño Mexican Grill**

**Burrito**

HOUSE BURRITO  
Wrapped in 12" Tortilla  
Favorite Fillings + Veggie, Chicken or Beef  
Favorite Fillings + Steak . . . . .

BURRITO PLATE  
Includes 12" Tortilla  
Favorite Fillings + Veggie, Chicken or Beef  
Favorite Fillings + Steak . . . . .

**Quesadilla**

HOUSE QUESADILLA  
12" flour tortilla w/ El Jalapeño's special sauce  
Cheese Only . . . . .  
Beef or Chicken . . . . .  
Steak . . . . .

**Tacos**

HOUSE TACOS  
3 crispy corn or soft flour tortillas  
Favorite Fillings + Veggie, Chicken or Beef  
Favorite Fillings + Steak . . . . .

CRISPY TACO SALAD  
(crunchy tortilla bowl)  
Favorite Fillings + Veggie, Chicken or Beef  
Favorite Fillings + Steak . . . . .

**Choose Your Favorite Fillings!**  
Lightly seasoned white or brown rice, organic black or pinto beans, green peppers, onions, mild / hot / fire / Verde Salsa, crunchy lettuce, shredded cheese, sour cream, Add fresh chips, jalapeño cheese, salsa + fountain drink TO ANY main item for \$ . . . . .

**Los Niños**  
Each meal includes juice + Los Niños size chips + salsa.

QUESADILLA (10" flour tortilla)  
Los Niños choice of cheese, chicken or steak . . . . .

TACOS  
Favorite Fillings + Los Niños choice of beef, chicken or steak, lettuce + cheese in 2 crispy corn or soft tacos . . . . .

**House Specials**

TRIPLE BEEFSTEAK  
FAJITA BURRITO  
Favorite Fillings + 3 scoops of beef steak, fire-Rja sauce . . . . .

CHALUPA  
Fried flatbread + 1 crispy or soft taco  
Veggie, chicken, or beef . . . . .  
Steak . . . . .

JALAPEÑO NACHOS  
Veggie, chicken or Beef . . . . .  
Steak . . . . .

**Extras**  
Add a taco (crispy or soft) . . . . .  
Fresh chips, jalapeño  
Cheese, + Salsa . . . . .  
Add guacamole . . . . .  
Extra salsa . . . . .  
Extra jalapeño cheese . . . . .  
Chips only . . . . .

## 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

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

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

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

- There are theoretically motivated guidelines for setting and scheduling the hyperparameter  $\beta_n$ .

# Expected improvement (I)

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

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

- ❑ What this means is that  $\mathbf{x}^*$  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

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

- It can be evaluated in closed form, leading to

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

where  $\Phi(\cdot)$  is the cumulative Gaussian distribution and  $\phi$  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:

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

- 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

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

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

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

where

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

with  $m$  the number of random features,  $\alpha$  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  $\boldsymbol{\theta} \sim p(\boldsymbol{\theta} | \mathcal{D}_n)$ ,

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

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

## 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 acquisition function for ES can be expressed formally as

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

where

- $H(\mathbf{x}^* \mid \mathcal{D}_n)$  denotes the differential entropy of the posterior distribution  $p_*(\mathbf{x} \mid \mathcal{D}_n)$ .
- 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)

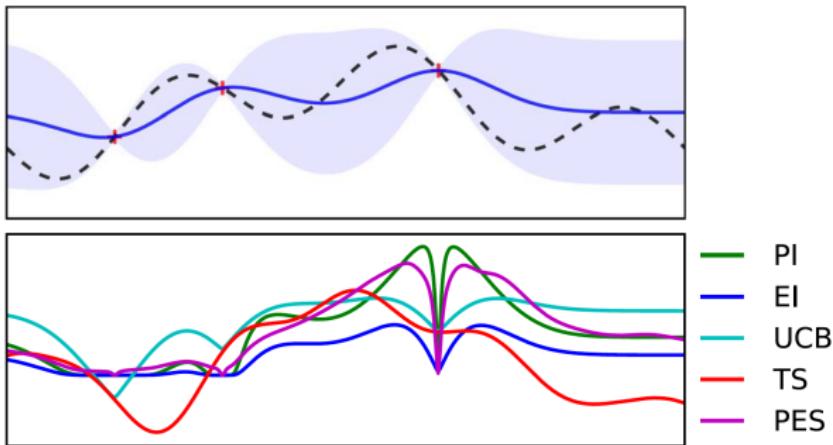
- Predictive entropy search (PES) (Hernández-Lobato et al., 2014) removes the need for a discretization and approximates the acquisition function in  $\mathcal{O}((n + d)^3)$ .
- For  $d < n$  is of the same order as EI.
- This is achieved by using the symmetric property of mutual information to write

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

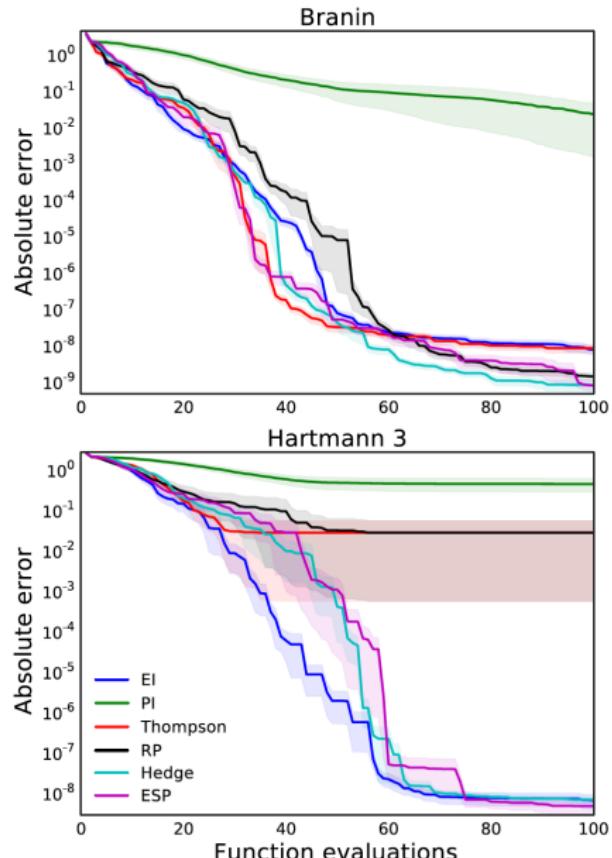
## Entropy search (V)

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

## Example different acquisition functions



# Results



# Contents

Introduction

Applications

Acquisition functions

Practical considerations

Exotic “Bayes Opt”

## 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  $\theta$  of the GP,  $\alpha(\mathbf{x}, \theta)$ .
- We would like to marginalize out our uncertainty about  $\theta$  with the following expression:

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

where

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

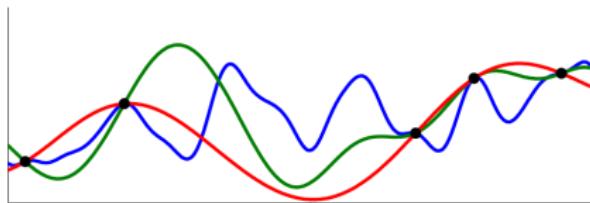
## Handling hyperparameters (II)

- Points estimates for  $\theta$ ,  $\hat{\alpha}_n(\mathbf{x}) = \alpha(\mathbf{x}; \hat{\theta}_n)$ , where  $\hat{\theta}_n$  can either be  $\hat{\theta}_n^{\text{ML}}$  or  $\hat{\theta}_n^{\text{MAP}}$ .
- Marginalizing out the hyperparameters using either quadrature or Monte Carlo, for example, approximating the integral before using

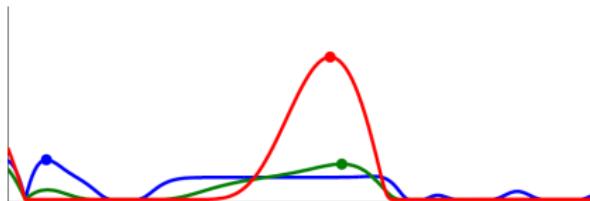
$$\mathbb{E}_{\theta|\mathcal{D}_n}[\alpha(\mathbf{x}; \theta)] \approx \frac{1}{M} \sum_{i=1}^M \alpha(\mathbf{x}; \theta_n^{(i)}),$$

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

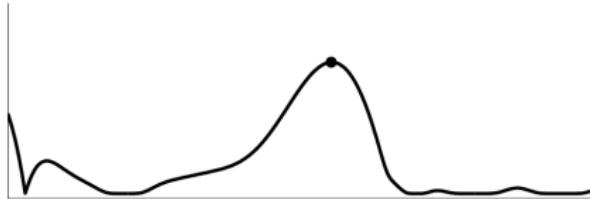
# Effect of the hyperparameters



(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).

# 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.

# REMBO: Bayes Opt with Random Embeddings

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

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

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



# Contents

Introduction

Applications

Acquisition functions

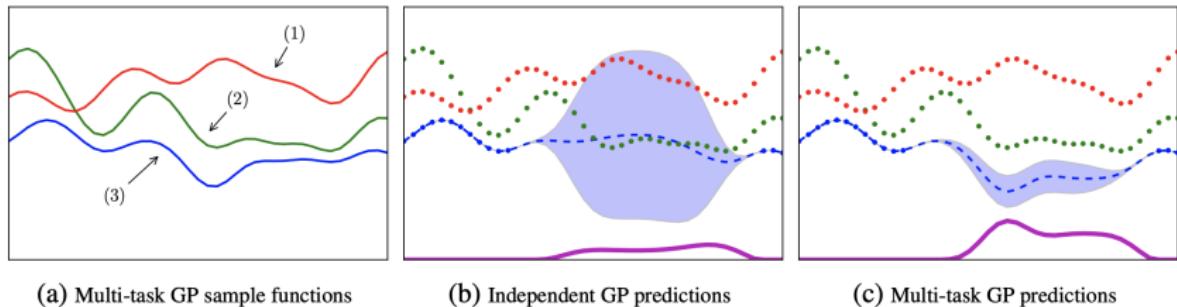
Practical considerations

Exotic “Bayes Opt”

# 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.
  - We want to optimize several tasks at the same time.
- Multi-output Gaussian processes can help here.

# Multi-task Bayesian optimization (II)



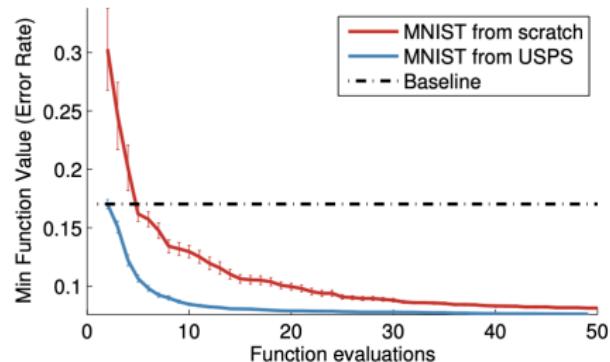
(a) Multi-task GP sample functions

(b) Independent GP predictions

(c) Multi-task GP predictions

# Applications multi-task Bayes Opt

- ❑ Cold start problem



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

## Parallelization (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 approaches arise as the natural solution to circumvent the computational bottleneck around these evaluations of  $f(\mathbf{x})$ .

## Parallelization (II)

- In this context, there are current observations  $\mathcal{D}_n = \{(\mathbf{x}_n, y_n)\}$  and pending experiments  $\mathcal{D}_p = \{\mathbf{x}_p\}$ .
- There are different ways in which it is possible to impute a set of experimental outcomes  $\tilde{\mathcal{D}}_p = \{(\mathbf{x}_p, \tilde{y}_p)\}$
- We then perform a step of Bayesian optimization using the augmented data set  $\mathcal{D}_t \cup \tilde{\mathcal{D}}_p$ .
- 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.

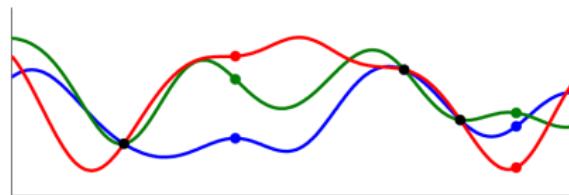
## Parallelization (III)

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

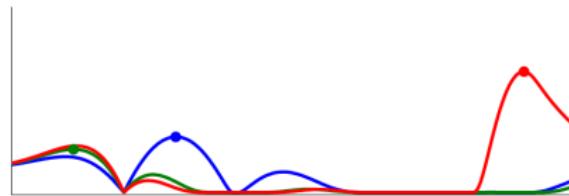
$$\begin{aligned}\alpha(\mathbf{x}; \mathcal{D}_n, \mathcal{D}_p) &= \int_{\mathbb{R}^J} \alpha(\mathbf{x}; \mathcal{D}_n \cup \tilde{\mathcal{D}}_p) p(y_{1:J} | \mathcal{D}_n) dy_{1:J} \\ &\approx \frac{1}{S} \sum_{s=1}^S \alpha(\mathbf{x}; \mathcal{D}_n \cup \tilde{\mathcal{D}}_p^{(s)}),\end{aligned}$$

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

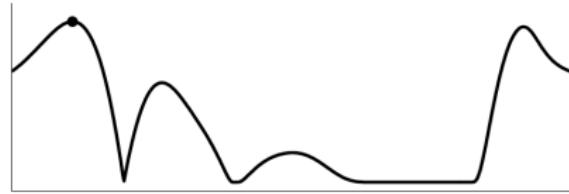
# Parallelization (IV)



(a) Posterior samples after three data



(b) Expected improvement under three fantasies



(c) Expected improvement across fantasies

# 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.

# References I

- 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.
- 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>.
- 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>.