Short course on Statistical Modelling for Optimization – lecture 4/4

Global optimization with GPs

June 2017 – Universidad Tecnológica de Pereira – Colombia

Nicolas Durrande (durrande@emse.fr)
Jean-Charles Croix (jean-charles.croix@emse.fr)
Mines St-Étienne – France

Today will be the last lecture

- Motivations
- Usual optimization methods
- Efficient Global Optimization
- Solving inverse problems

Introduction

Optimization is one of the most common problem in engineering.

One can distinguish methodologies:

- Local optimization
- Global optimization
- Robust optimization

and different function input spaces:

- Discrete
- Continuous

We usually consider as a reference the **minimization** problem, and this course is about continuous variables

According to the context of this short course, we will focus on optimization methods that do not require a lot of function evaluations.
$\rightarrow$ model based optimization.

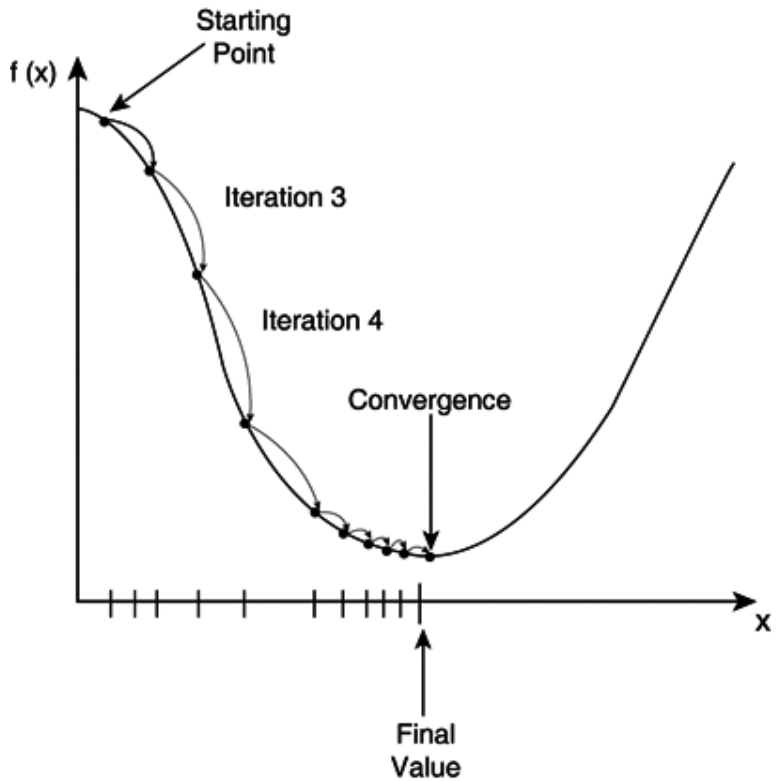
We will also pay a particular attention to optimization when

- observations are noisy
- parameter values are noisy

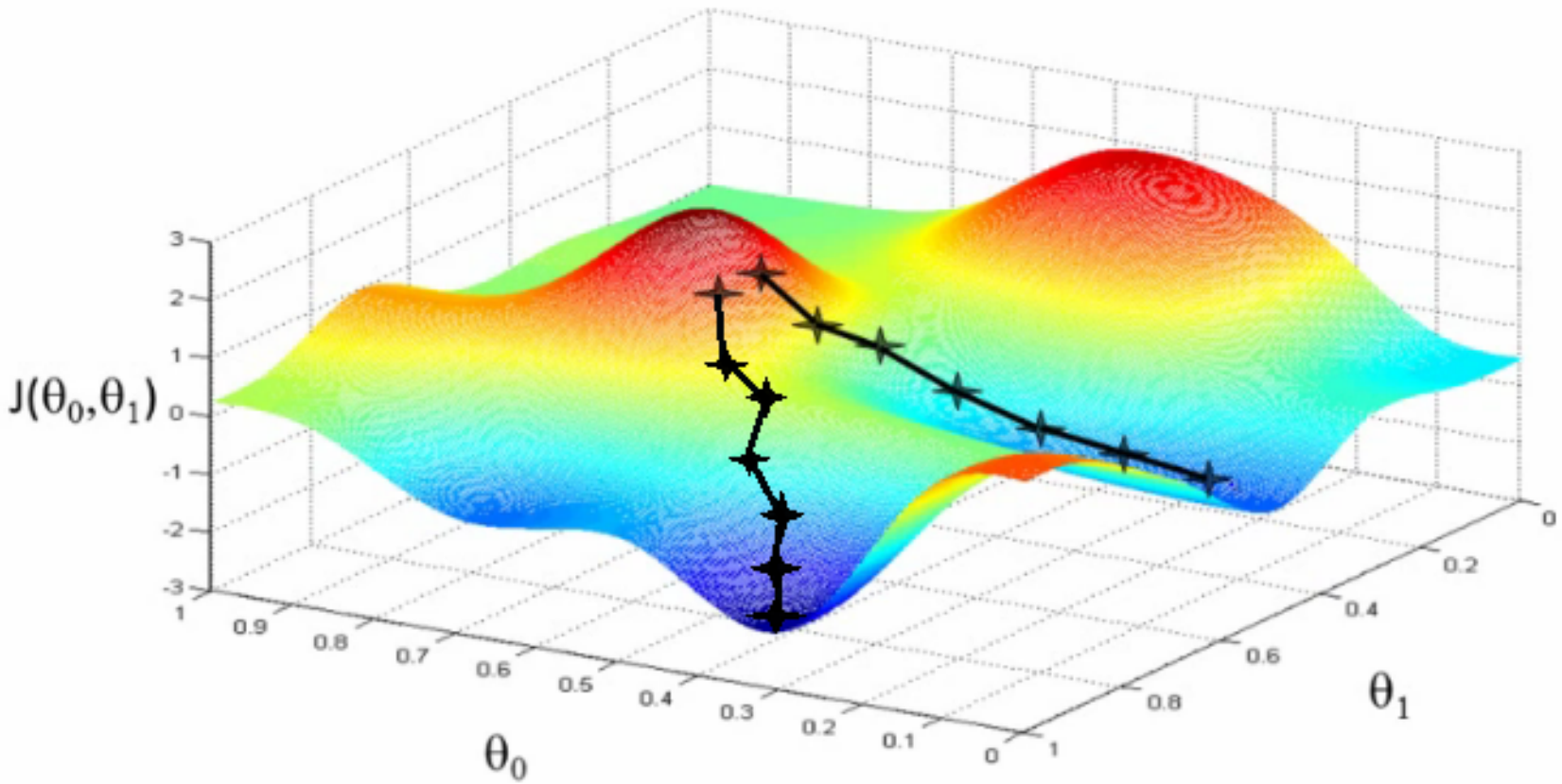
Local optimization

Local optimization methods are looking for the minimum value that can be found in the "neighbourhood" of the starting point.

Typical method is **gradient descent**.



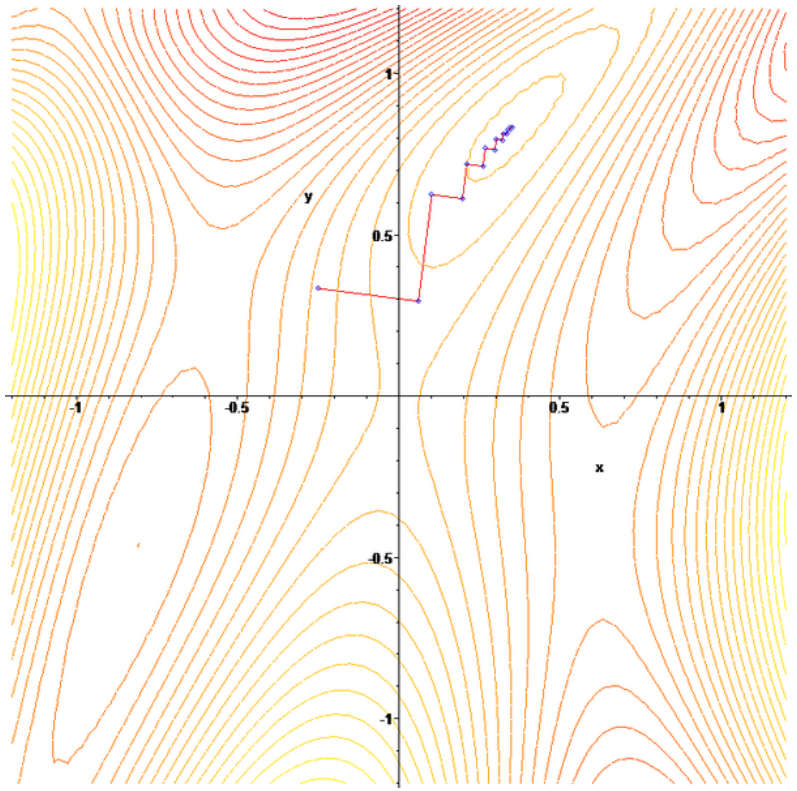
This can also be done in higher dimension:



The convergence points depends on the starting region.

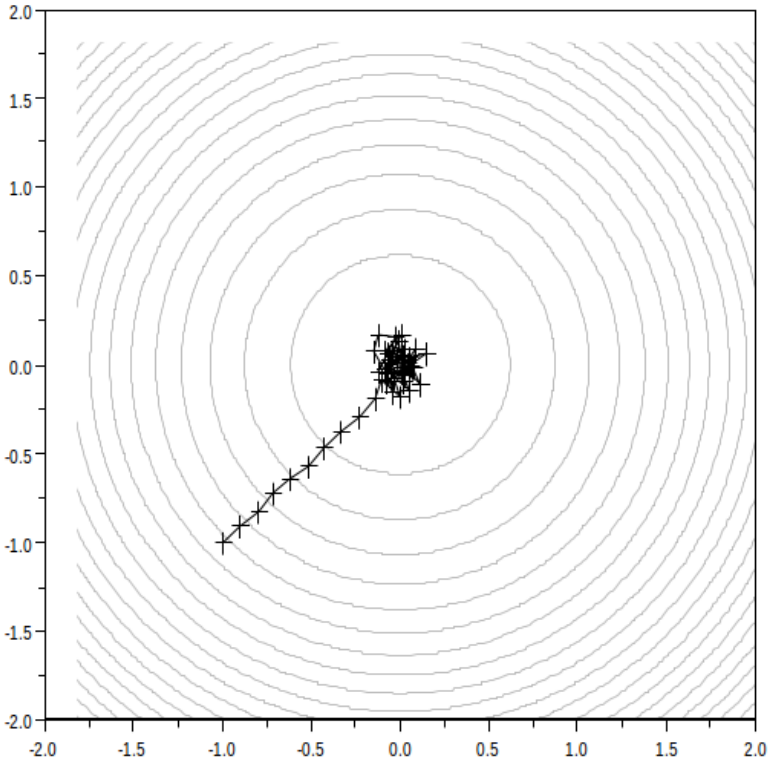
Cons of gradient descent:

- cannot handle noise
- in high dimension, computing the gradient is costly
- it does not follow the shortest path

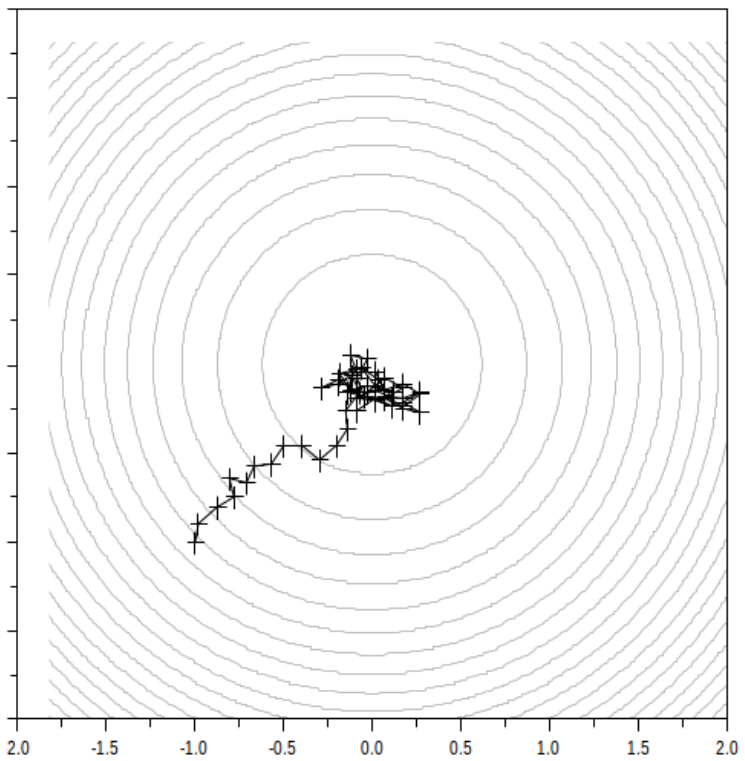


The convergence points depends on the starting region.

Observation noise is always an issue for gradient based methods:



Deterministic



Noisy observations

Source: Talk from R. Le Riche at the *Modeling and Numerical Methods for Uncertainty Quantification*, Porquerolles, 2014

One improvement is to consider not only the first derivative but also higher orders

Newton methods The sequence of visited points is given by:

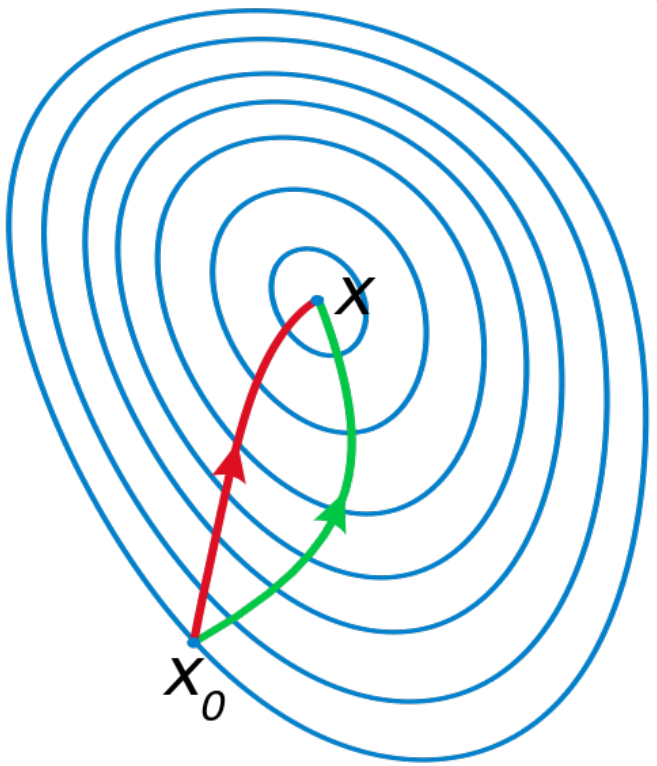
$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

This can be interpreted as fitting a quadratic function at each step and to define the next point as the maximum/minimum of the quadratic function.

In higher dimension, the first and second derivatives are replaced by the gradient and the Hessian matrix:

$$x_{n+1} = x_n - (H_f(x_n))^{-1} \nabla f(x_n)$$

- \+ The convergence is faster than gradient descent
- – Does not handle noise
- – Computationally expensive (Hessian matrix)

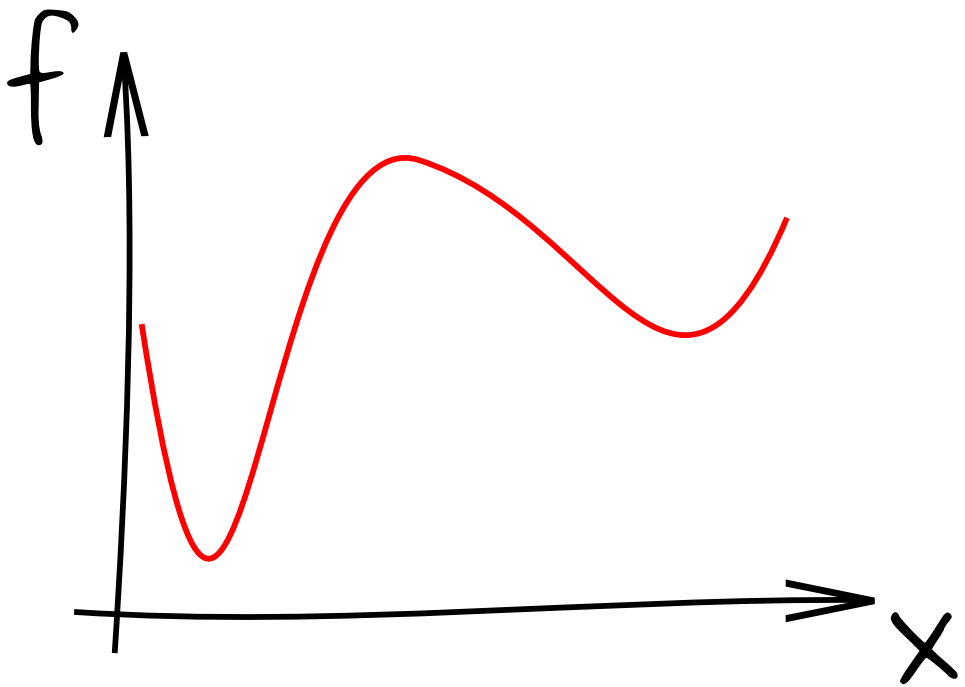


In order to overcome the expensive computation of the Hessian matrix, **Quasi Newton methods** use an approximation of it based on low rank updates.

BFGS is probably the most famous example.

- \+ They are fast algorithm
- \+ Already implemented in all languages
- – Requires a large number of evaluations
- – Cannot handle noise

In practice, we are often interested in the **global minimum** and not just the local ones:



Cheap improvements are:

- **multi-start**.
- Monte-Carlo + BFGS

Global optim.

A wide variety of global optimization methods have been developed:

- Evolutionary algorithms
- Simulated annealing
- Model based optimization

Evolutionary algorithms

The principle of **evolutionary algorithm** is to consider a population of points that will change at each time step. The evolution of the population is typically done by:

- selecting the best points (Parents)
- combining points together (Breeding)
- mutations of points

Evolutionary algorithms

A state of the art evolutionary algorithm for optimization is **CMA-ES** (Covariance Matrix Adaptation Evolution Strategy):

Starting from an initial set of normally distributed points, the algorithm loops over

1. select the best subset of points
2. update the distribution mean
3. update the covariance matrix
4. generate a new set of points

Remarks: no gradient is computed. The algorithm can handle noise.

Example: steps of the CMA-ES algorithm

1. Sampling (λ individuals, g generation) and evaluating:

$$x_k^{g+1} = m^g + \sigma^g \mathcal{N}(0, C^g),\ k = 1..\lambda \Rightarrow f(x_k^{g+1})$$

2. Selection and recombination:

$$m^{g+1} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{g+1}, \sum_{i=1}^{\mu} w_i = 1,\ w_1 \geq ... \geq w_\mu \geq 0$$

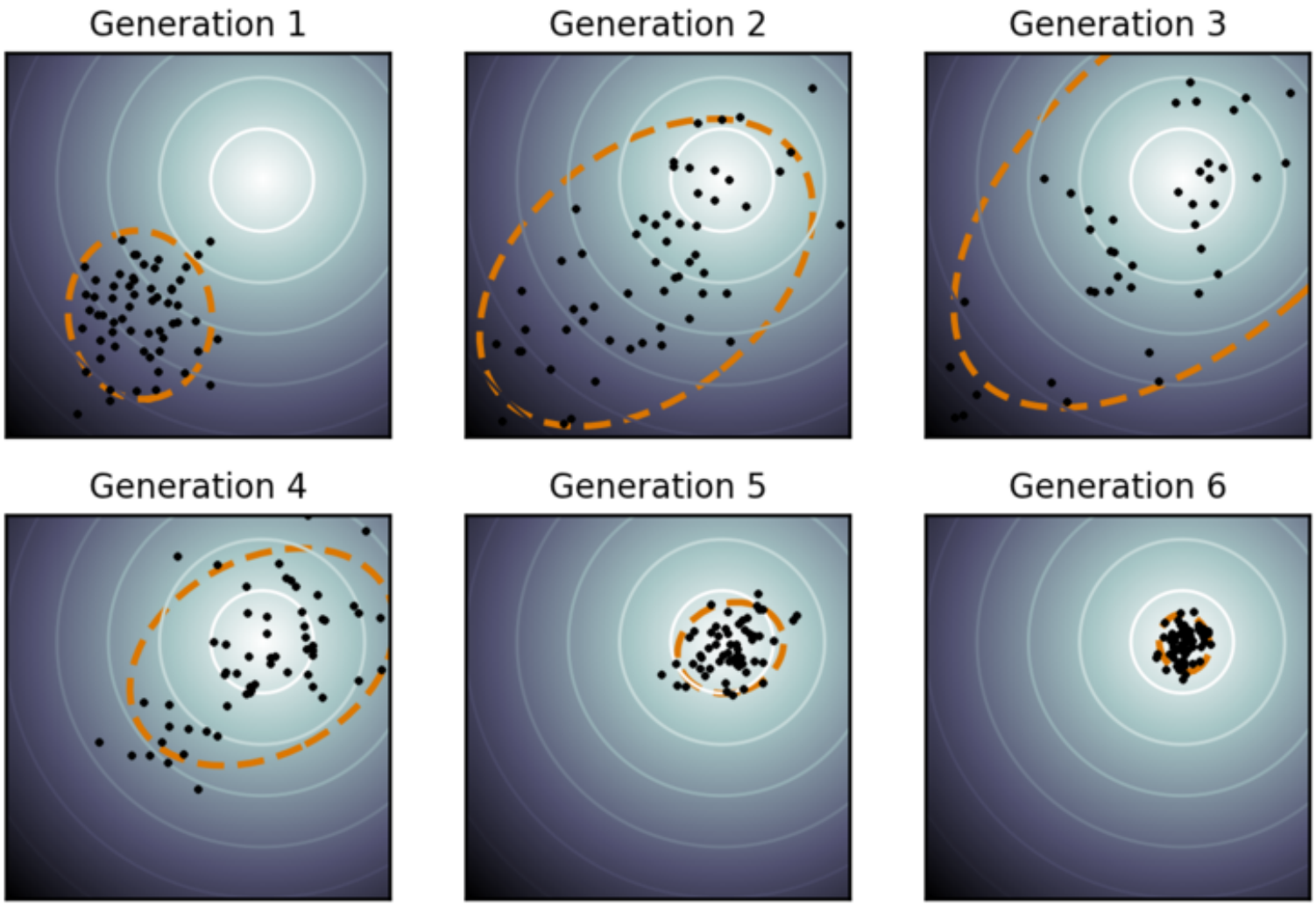
3. Covariance update (multiple methods...):

$$C_\mu^{g+1} = \sum_{i=1}^{\mu} w_i (x_{i:\lambda}^{g+1} - m^g)(x_{i:\lambda}^{g+1} - m^g)^T$$

Source: The CMA Evolution Strategy: A Tutorial, N. Hansen, 2016

Evolutionary algorithms

Example



Simulated annealing

The principle of simulated annealing is to consider a random walk in the input space and to accept the move from x_n to x_{n+1} if either

- $f(x_{n+1}) < f(x_n)$ (the moves improve the criteria)
- $\exp(-(f(x_{n+1}) - f(x_n))/T) < U$ where U is a uniform(0,1) random variable and T is the temperature (we accept "small" degradations).

Simulated annealing is inspired from a metallurgy technique involving heating and controlled cooling of materials to improve the structure. Allowing moves that degrade the function allows to escape from local minima.

T must tends toward zero with time to obtain convergence

Simulated annealing

Pros:

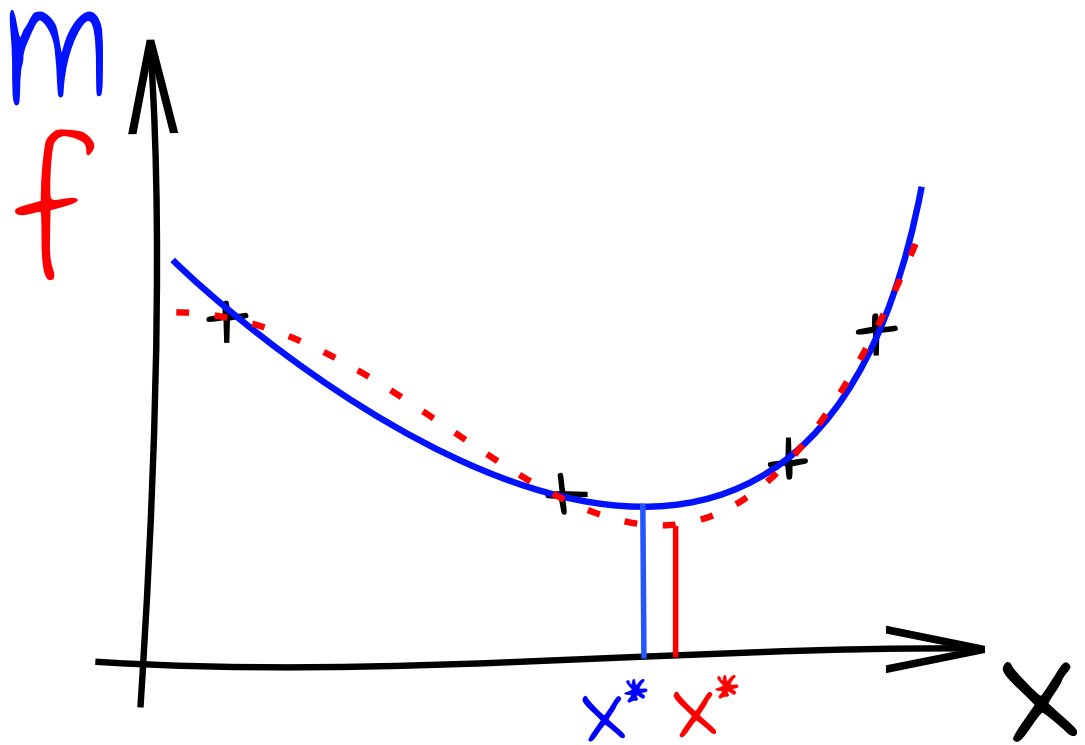
- No need for gradients
- Can be used in high dimension
- Provides a trade-off exploitation/optimization

Cons:

- The temperature is difficult to tune
- Requires a large number of observations

Model based optimization methods

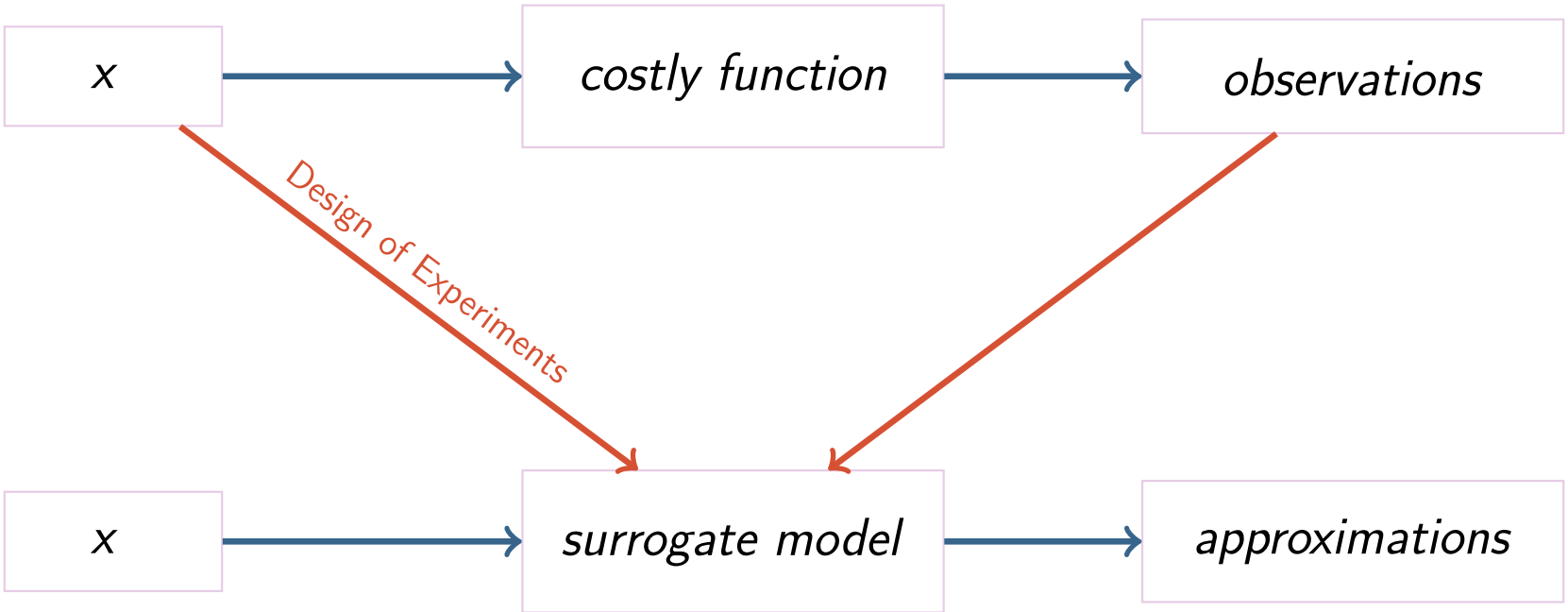
If the number of function evaluations are limited, we can run the optimization on the model instead of running it directly on the function



In the end, we hope that:

$$\begin{aligned} \text{argmin}(m) &\approx \text{argmin}(f) \\ \min(m) &\approx \min(f) \end{aligned} \tag{1}$$

Overall framework



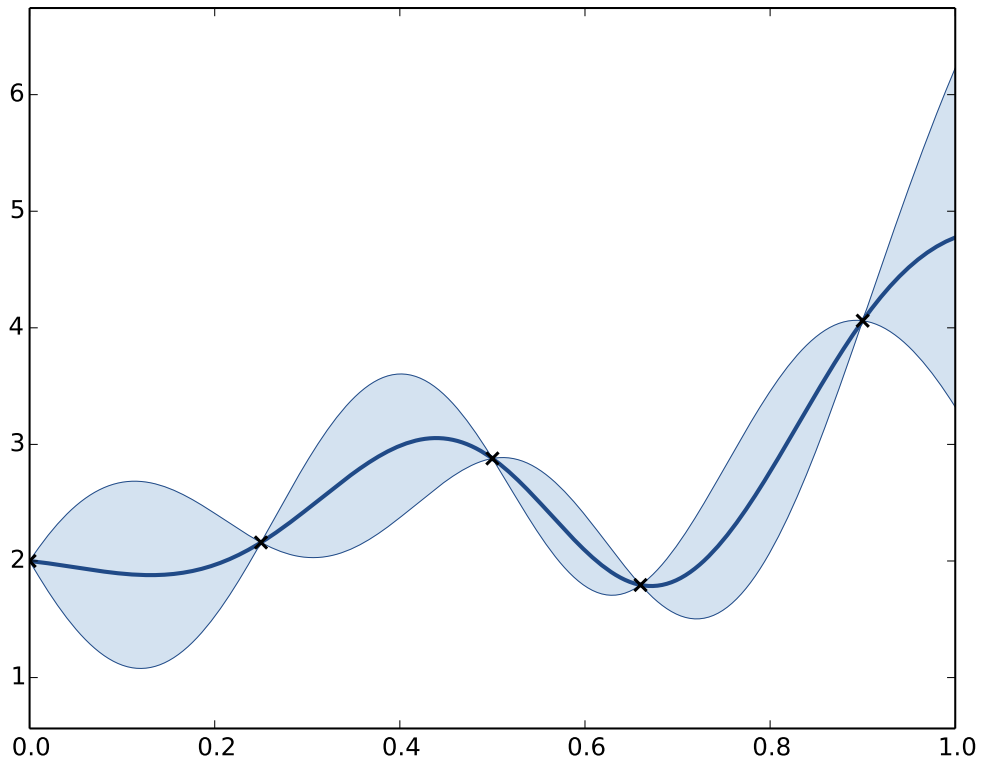
In practice, it is risky to take decisions based only on the model...

On the other hand, the model can be used to guide us in the search for the optimum.

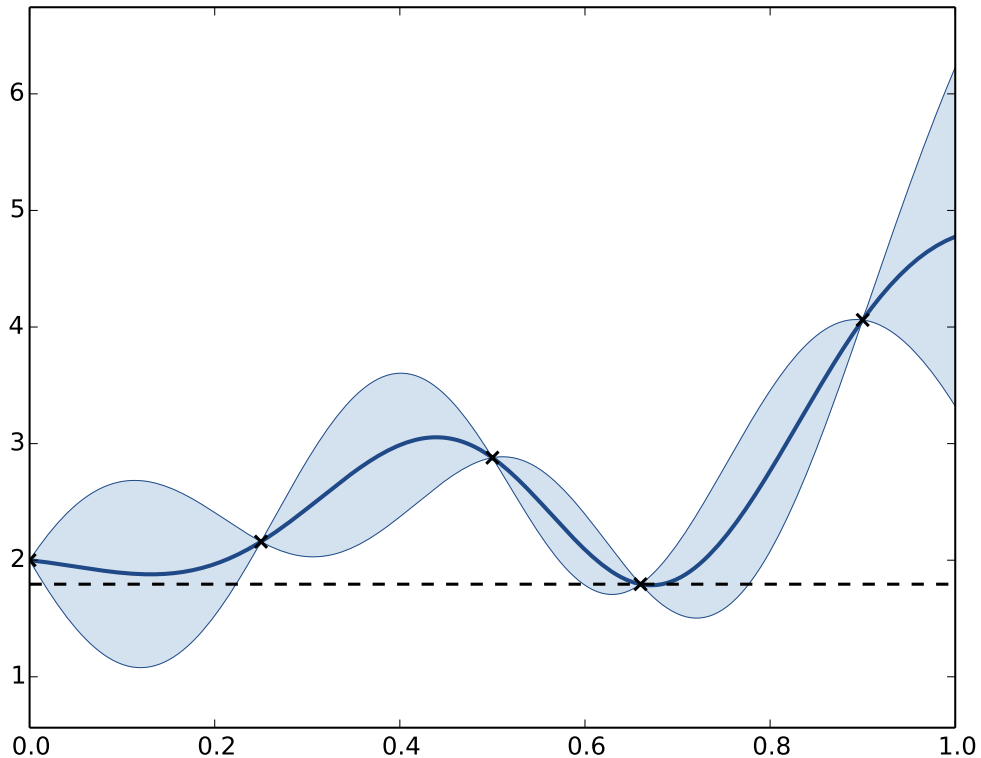
Global optimization methods are a trade-off between

- Exploitations of good results
- Exploration of the space

How can GPR models be helpful?



In our example, the best observed value is 1.79

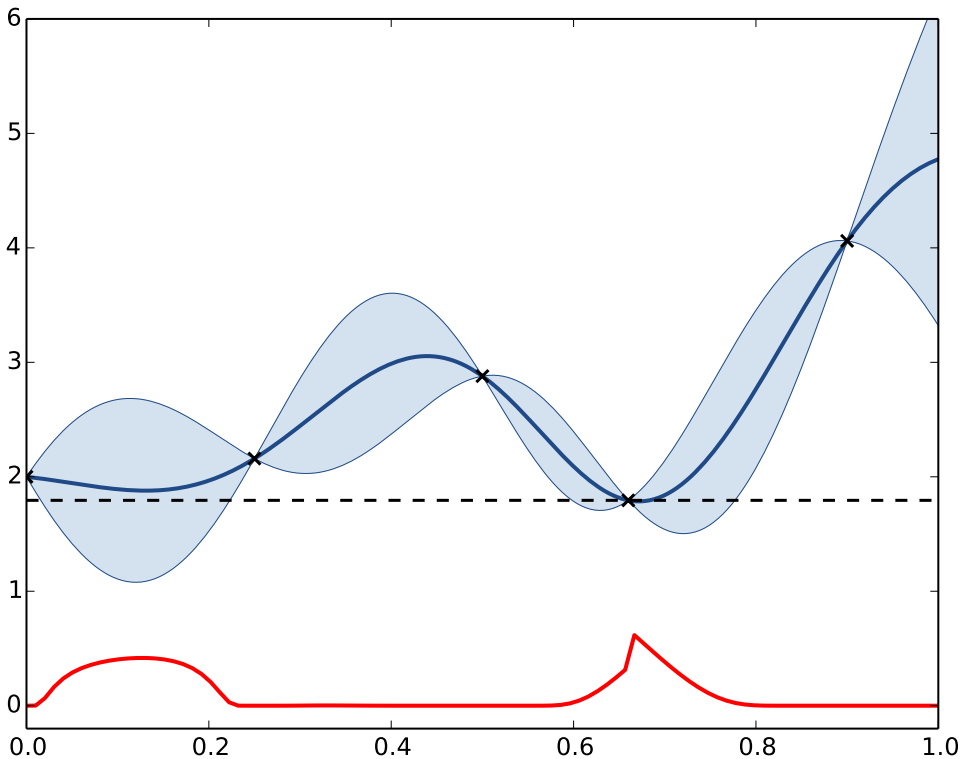


Various criteria can be studied

- probability of improvement
- Expected improvement

Probability of Improvement:

$$PI(x) = cdf\left(\frac{\min(F) - m(x)}{\sqrt{(c(x,x))}}\right)$$



The point with the highest PI is often very close to the best observed value. We can show that there is a x in the neighbourhood of x^* such that $PI(x) \geq 0.5$.

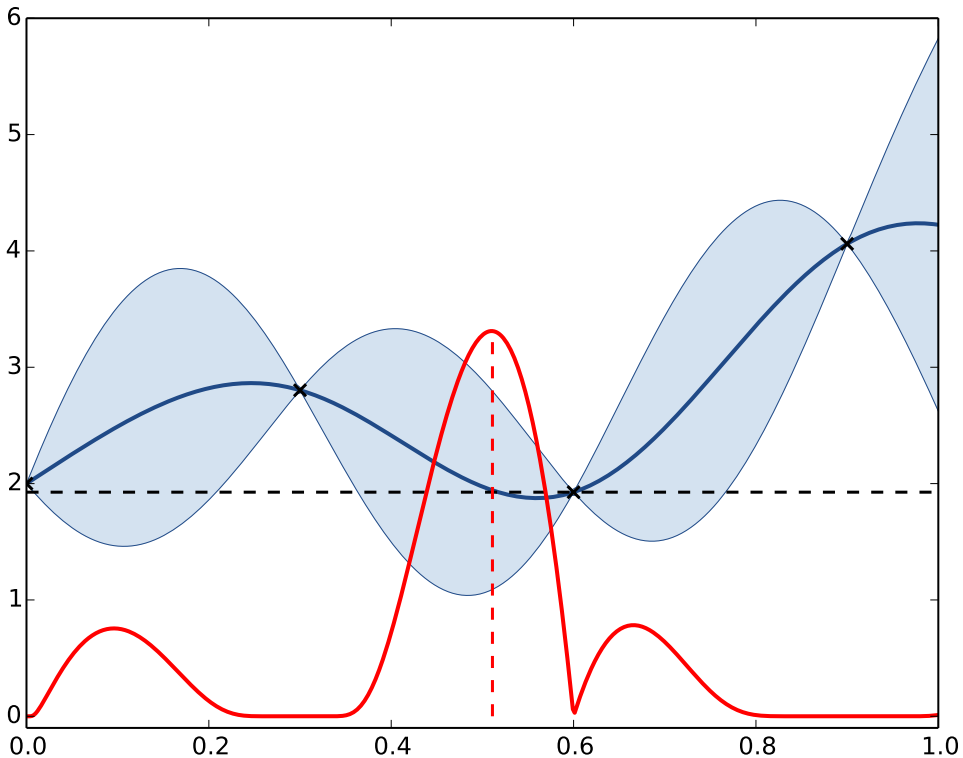
For such points, the improvement cannot be large...

Can we find another criterion?

Expected Improvement:

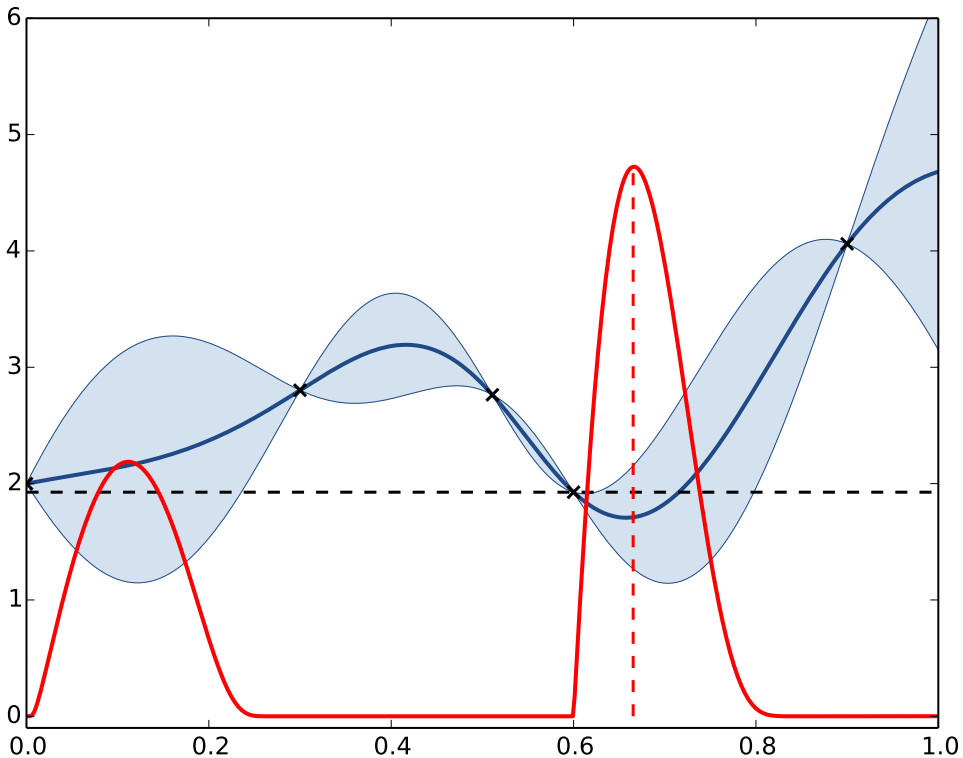
$$EI(x) = \sqrt{c(x,x)}(u(x)cdf(u(x)) + pdf(u(x)))$$

$$\text{with } u(x) = \frac{\min(F) - m(x)}{\sqrt{(c(x,x))}}$$



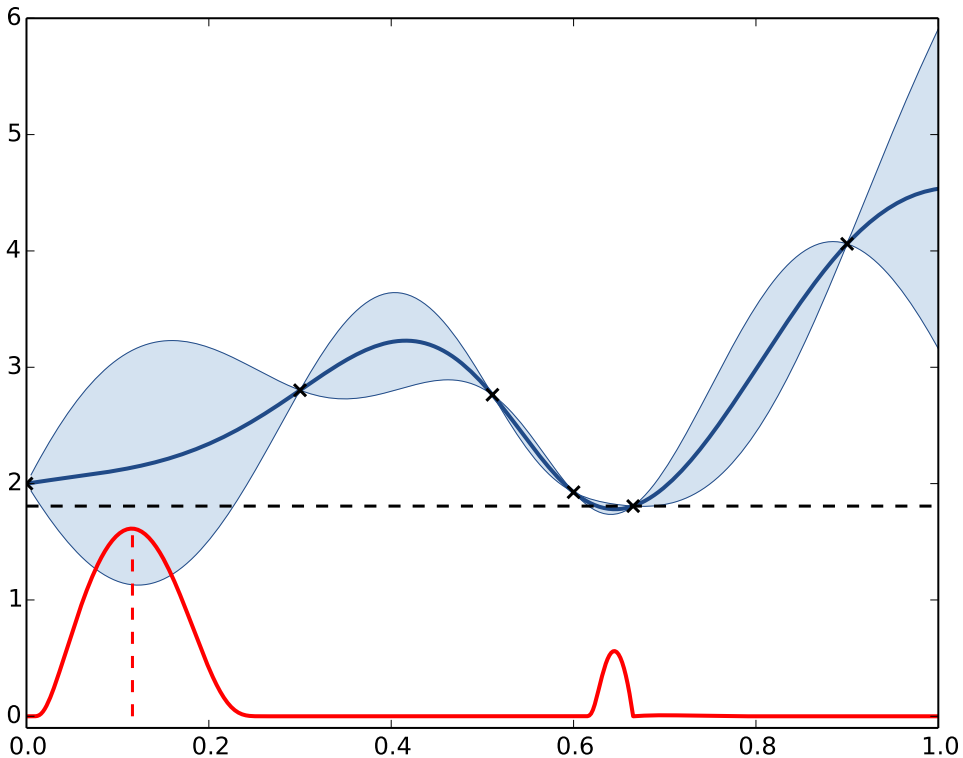
Expected Improvement

Let's see how it works... iteration 1



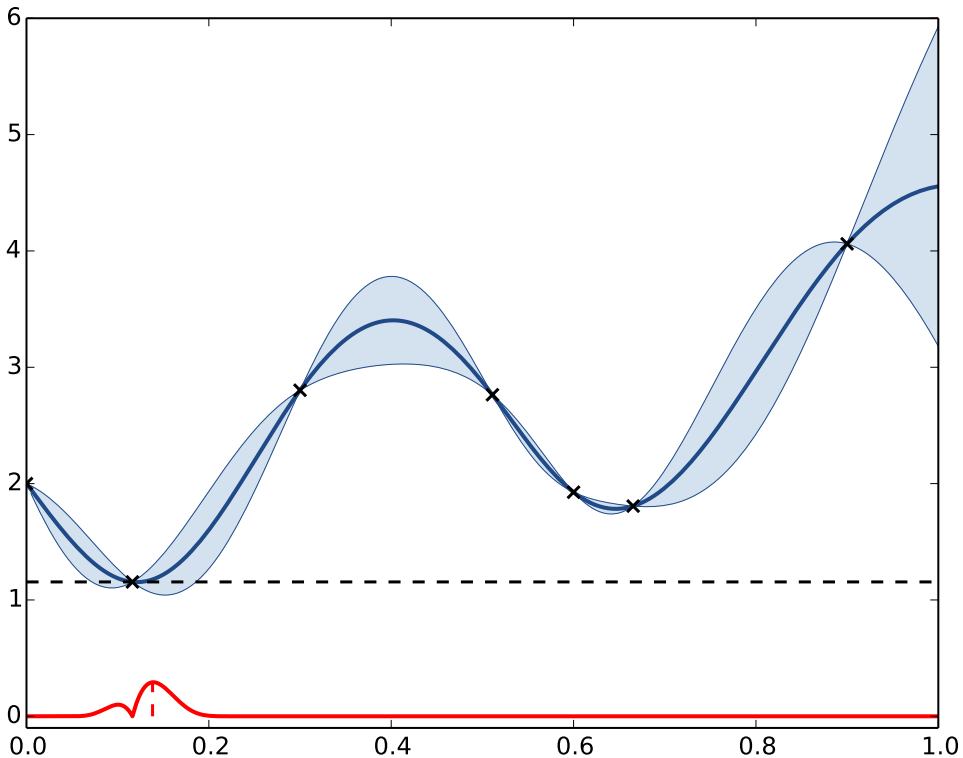
Expected Improvement

Let's see how it works... iteration 2



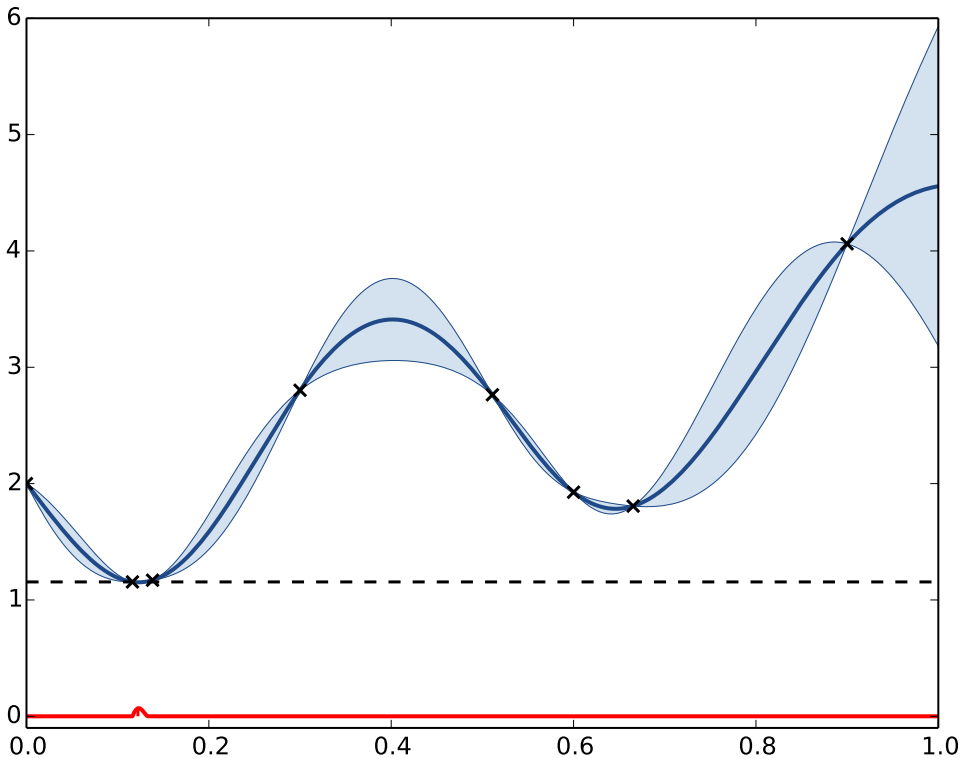
Expected Improvement

Let's see how it works... iteration 3



Expected Improvement

Let's see how it works... iteration 4



Expected Improvement

Let's see how it works... iteration 5

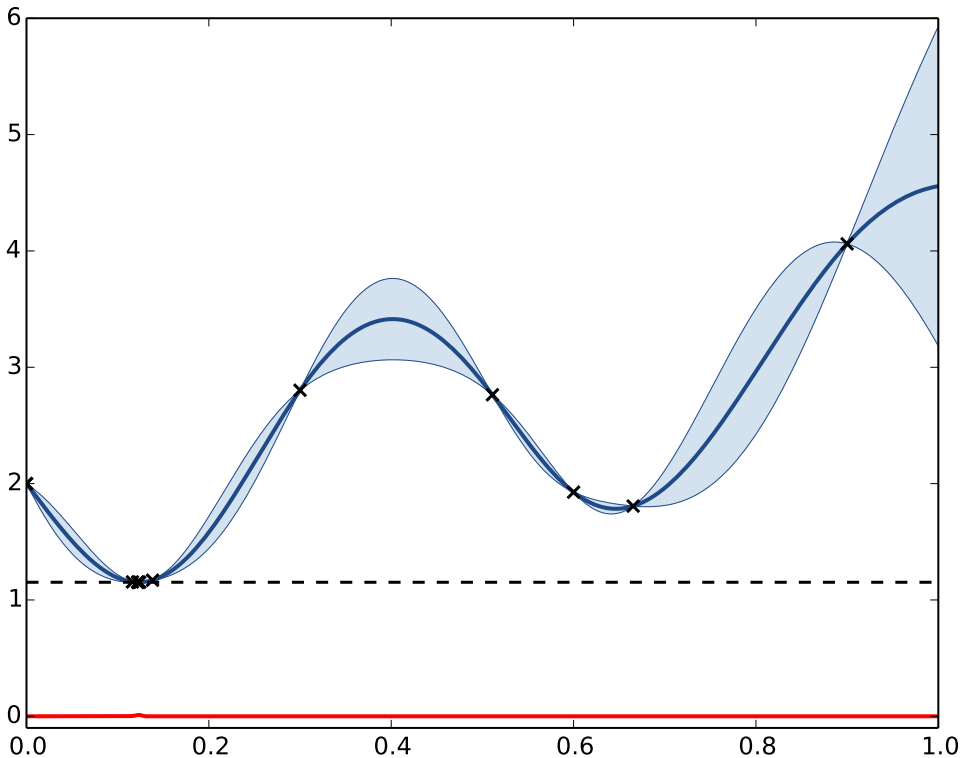
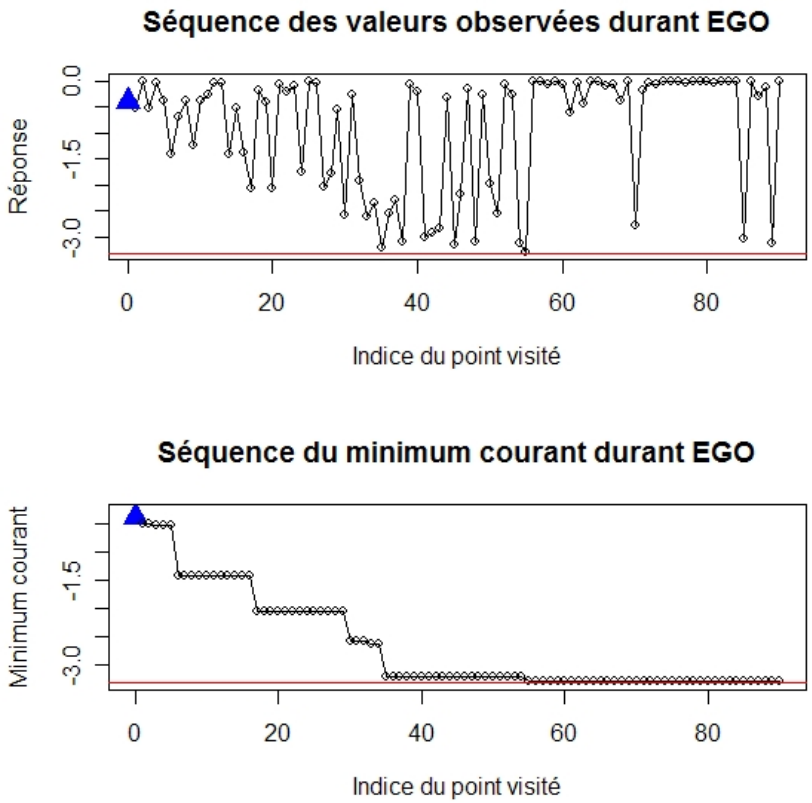
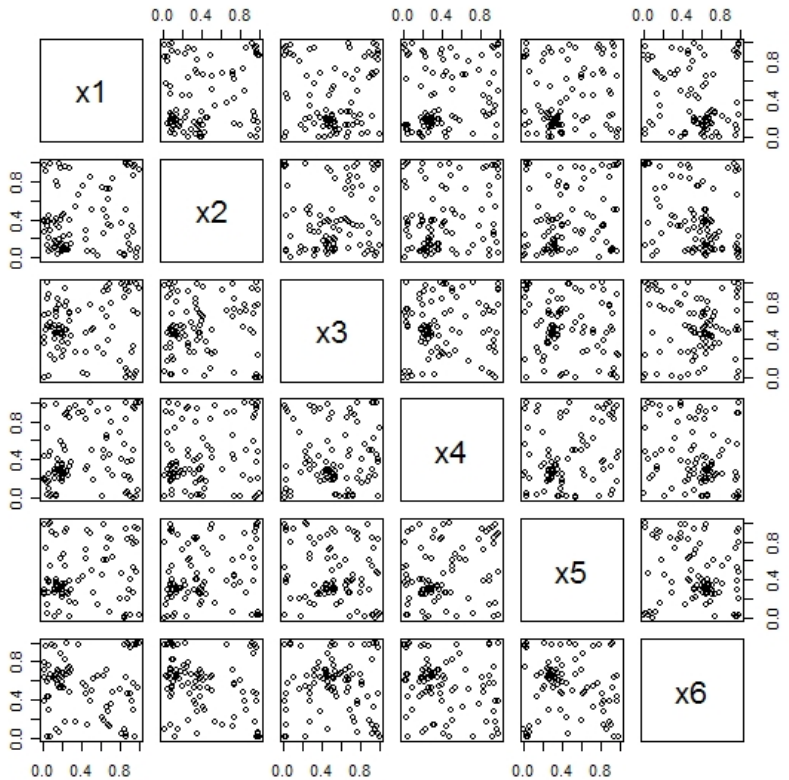


Illustration for $d = 6$ (Hartman)

Illustration in higher dimension





Source: DiceOptim, D. Ginsbourger, 2009.

Expected Improvement

This algorithm is called **Efficient Global Optimization** (EGO). It is famous since a paper of Jones et Al in 1998.

+ EGO provides a good trade-off between exploitation and exploration.
+ It only requires a few function observations (10 in the example)

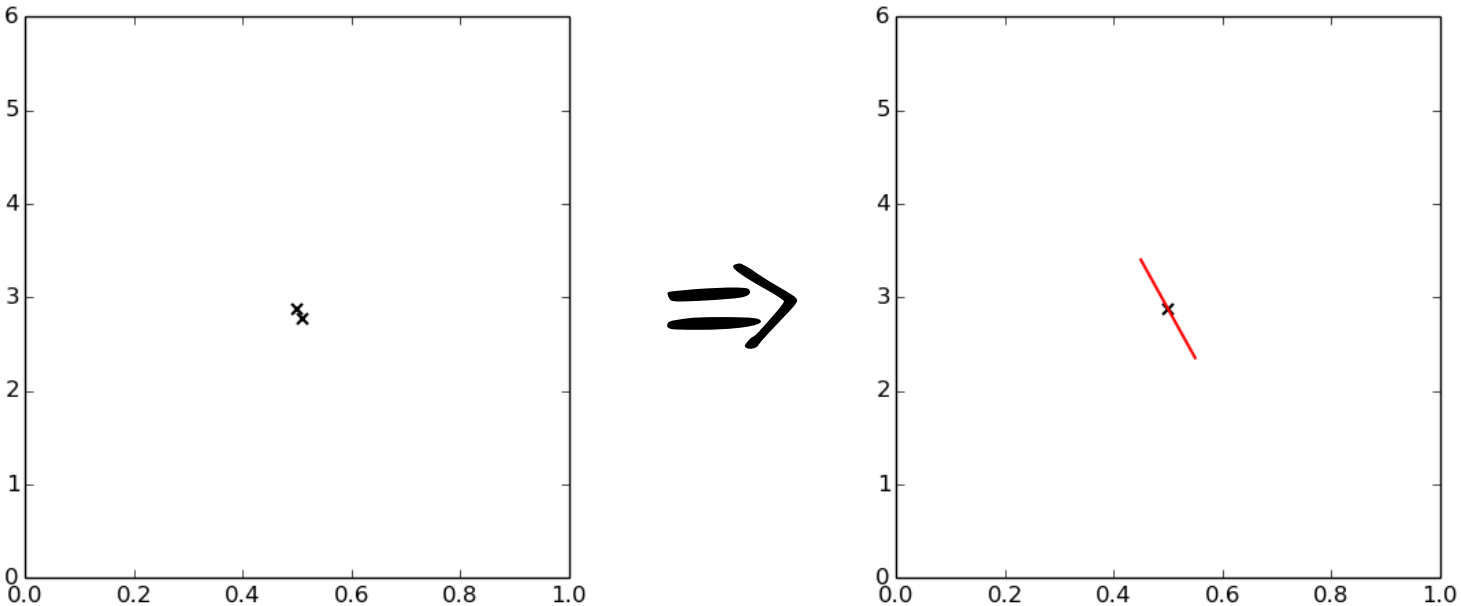
One issue is that we may have a model with observations very close one from each other

Example

From the previous 5 iterations, we obtain $1.44e39$ for the conditioning of the covariance matrix. Eigenvalues are

$$(67.70,\ 24.86,\ 5.13,\ 1.68,\ 0.45,\ 0.16,\ 0.01,\ 0.00,\ 0.00,\ 0.00)$$

One way to improve the conditioning of the covariance matrix is to replace two values that are close-by by one function value and one derivative:



Cond. $= 3842$ Cond. $= 10$

This can be generalised to higher orders $\rightarrow$ Taylor expansion
see articles from M. Osborn

If we now the computational budget in advance, adding new points at the **best one step ahead location** is not optimal.

Some improvements have been made toward this

- Batch EGO
- Parallelization of the algorithm

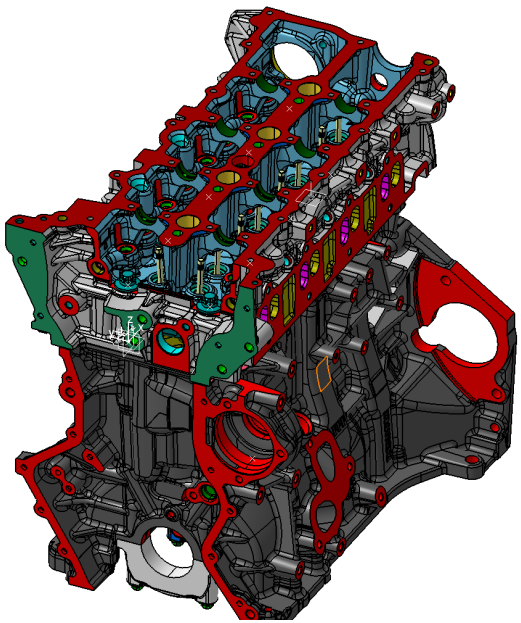
see works from D. Ginsbourger

Robust optimization

Robust optimization may mean various things:

- There is observation noise on the output
- Some input variables are uncertain
- Model is uncertain

Example

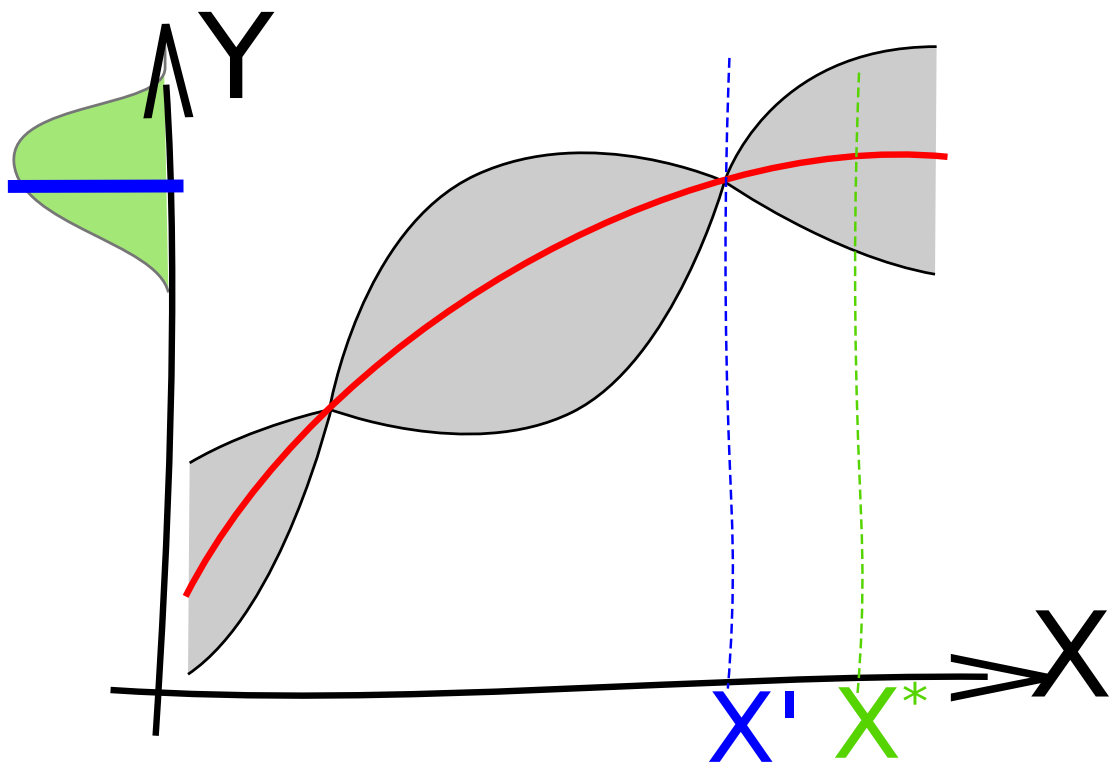


a +/- 1mm dispersion in the manufacturing of a car cylinder head can degrade its performance (g CO2/km) by -20% (worst case)

Source: Talk from R. Le Riche at the Porquerolles Summer School, 2014

Example

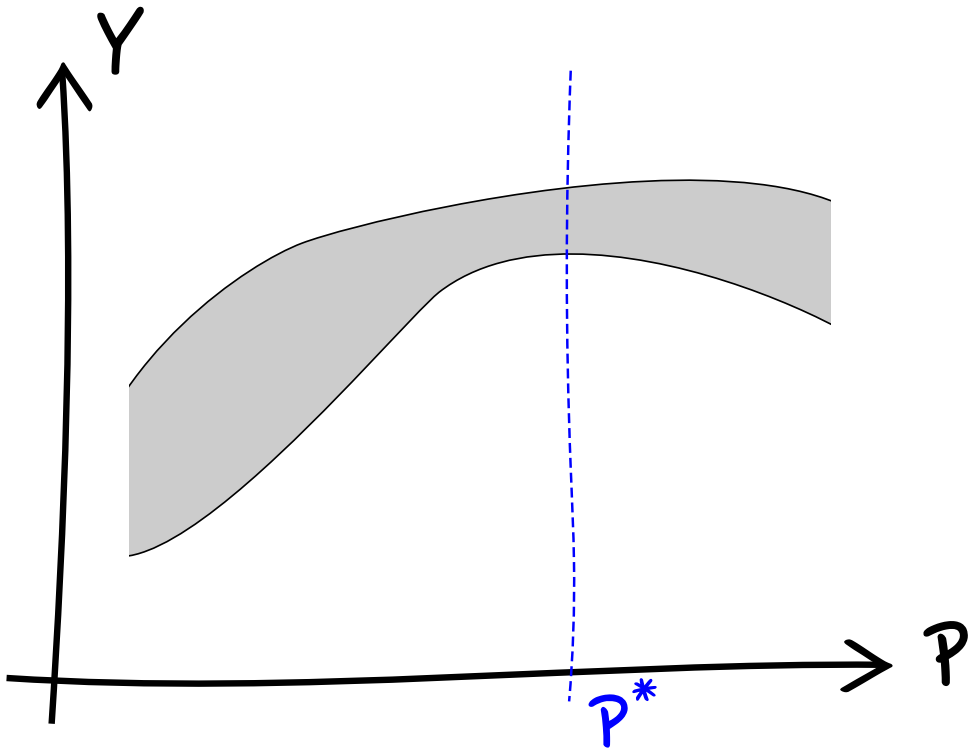
Here is a basic example:



Which input is the best?

Example

A non Gaussian example



In some cases, we may want to optimize the worst case scenario.

Can EGO be adapted when observations are noisy?

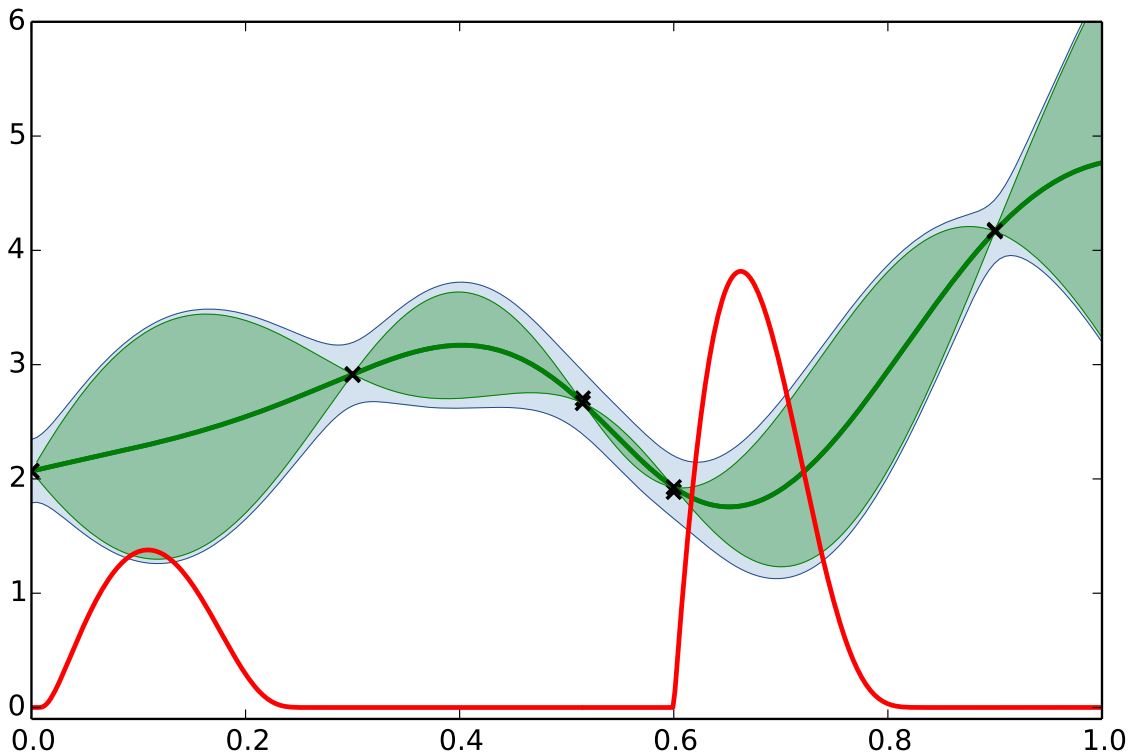
First of all, using the current best observation as a minimum does not make much sense...

Some solutions are

- S1 Build a new model that interpolates $m(X)$ at X.
- S2 Include observation noise and replace $\min(F)$ by $\min(m(X))$ in the EI expression
- S3 Similar to 2 but consider an Expected Mean Improvement.

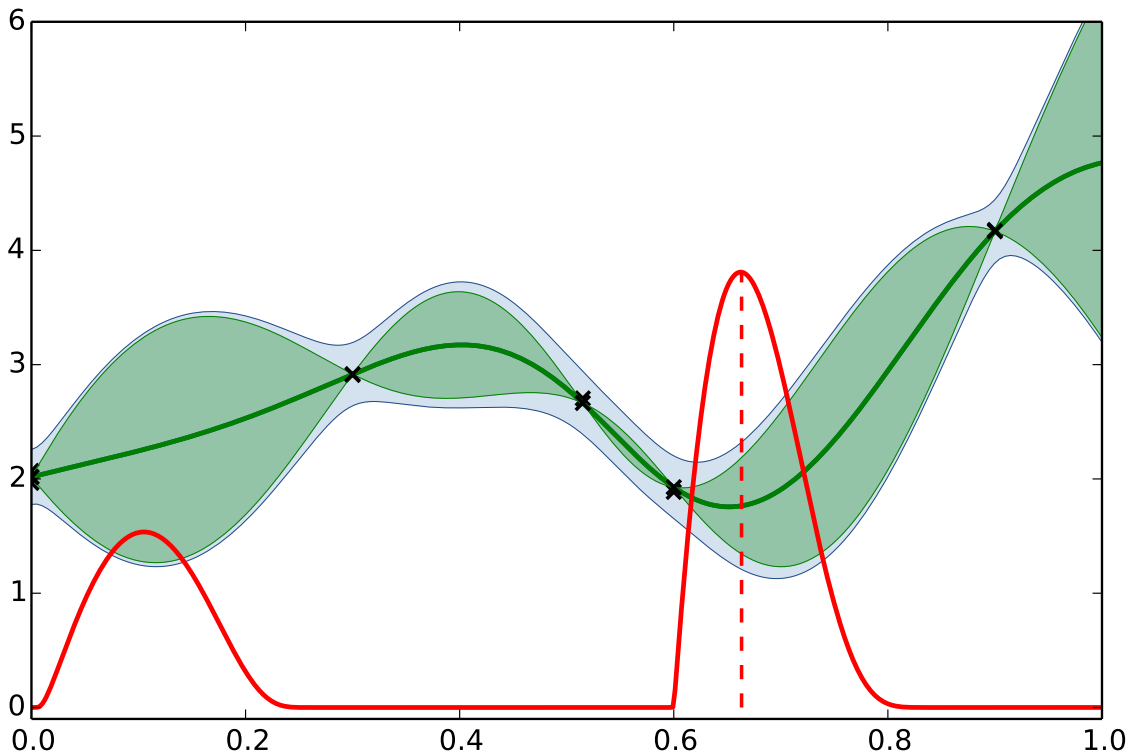
Solution 1

iteration 0



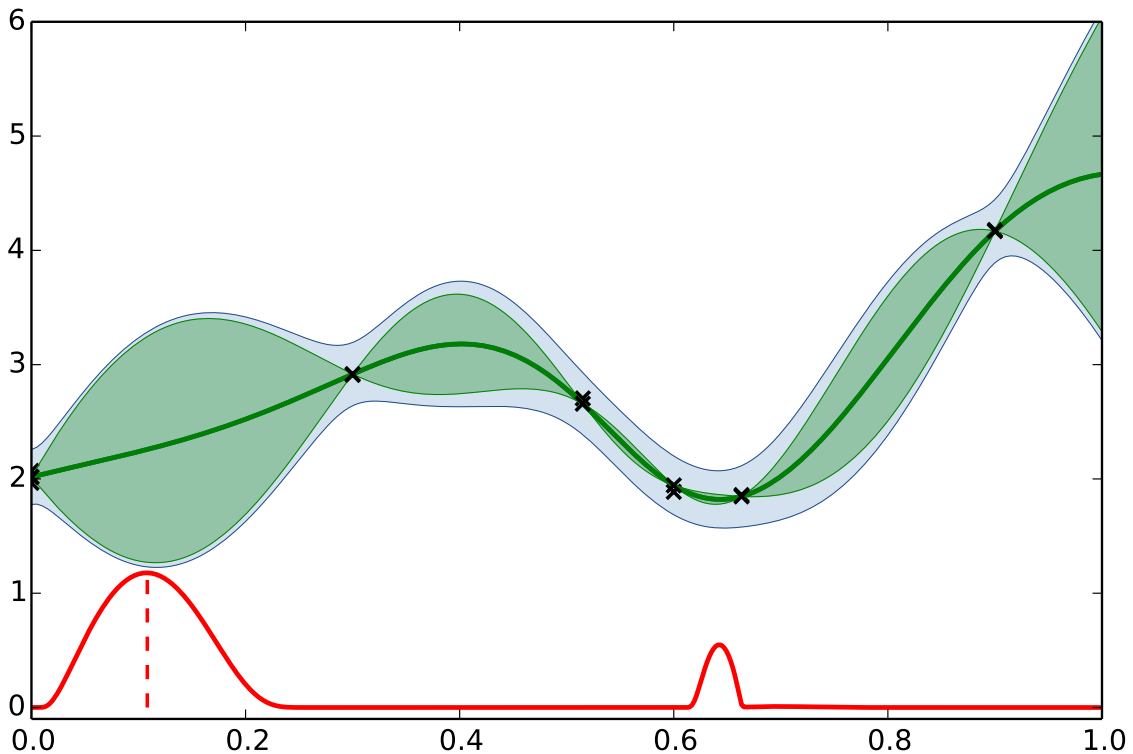
Solution 1

iteration 1



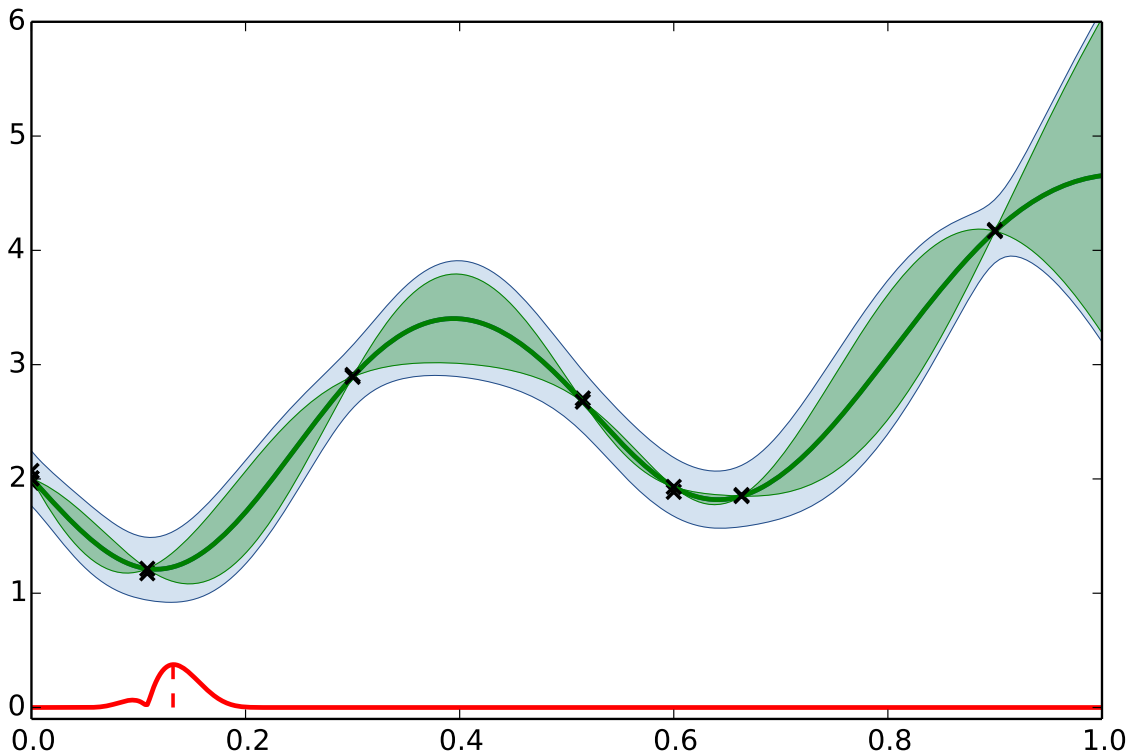
Solution 1

iteration 2



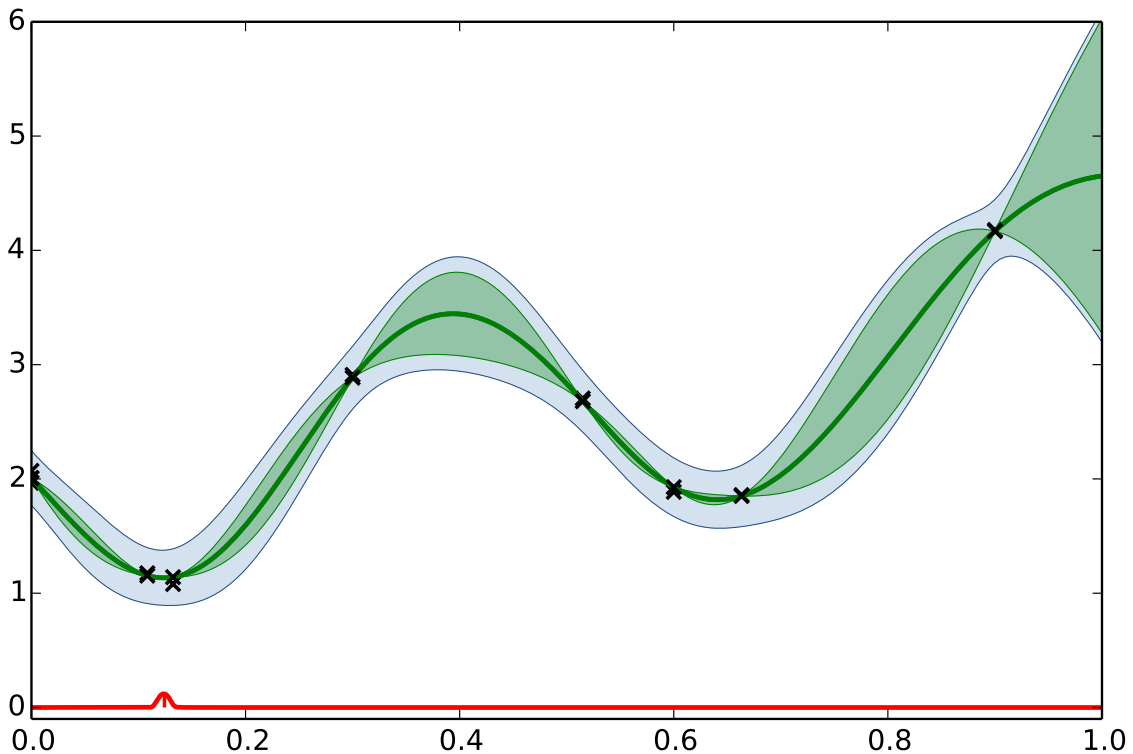
Solution 1

iteration 3



Solution 1

iteration 4



Related problems

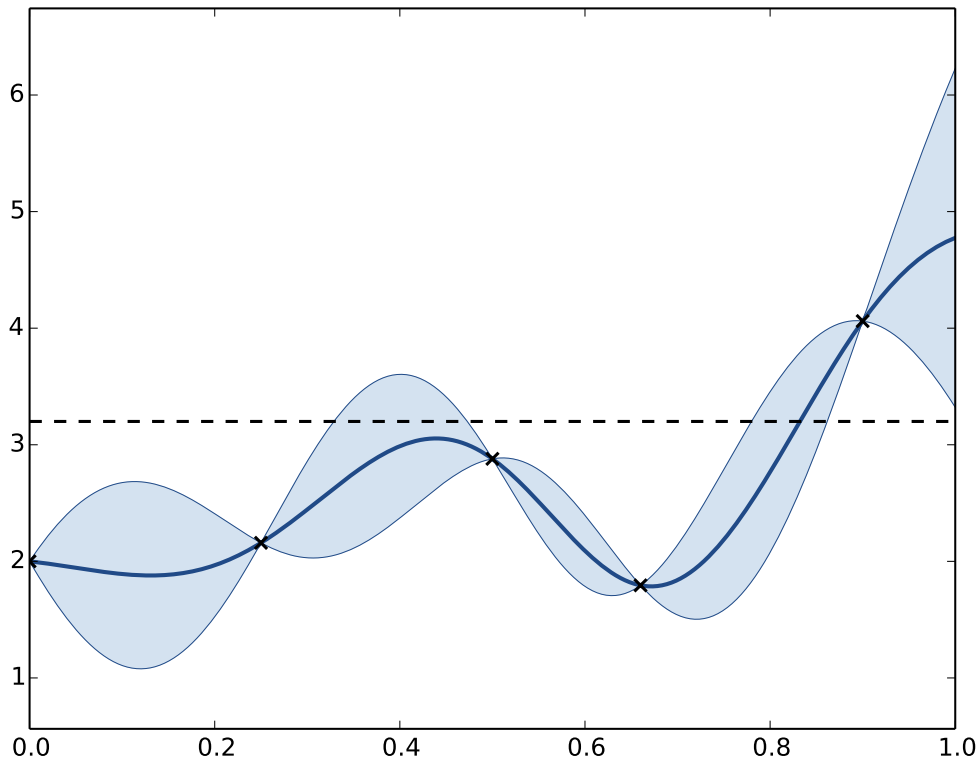
Some related optimization problems are:

- calibration problems
- probability computations

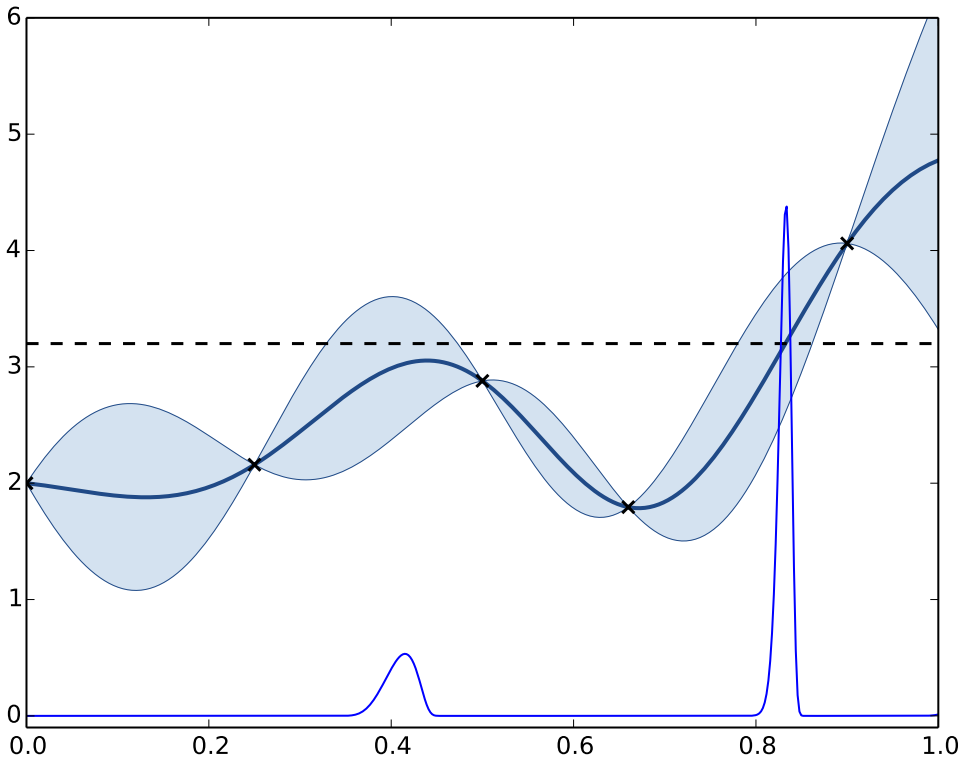
Some algorithms with an EGO spirit can be applied:

- SUR methods

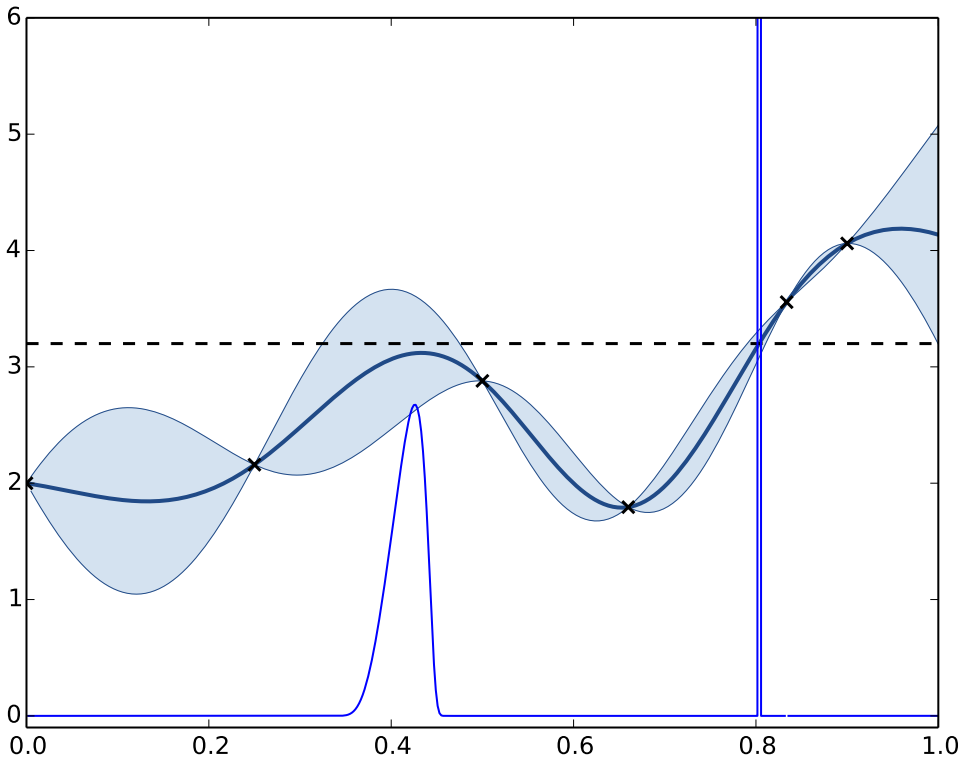
We want to find the input(s) such that f(x) = 3.2



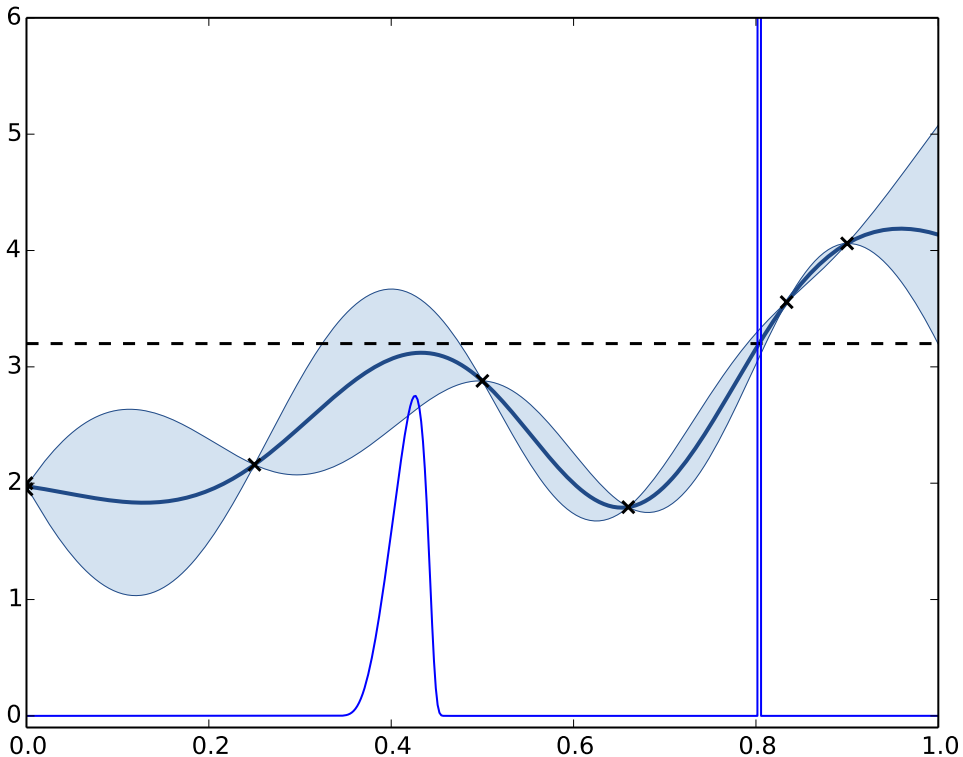
iteration 0:



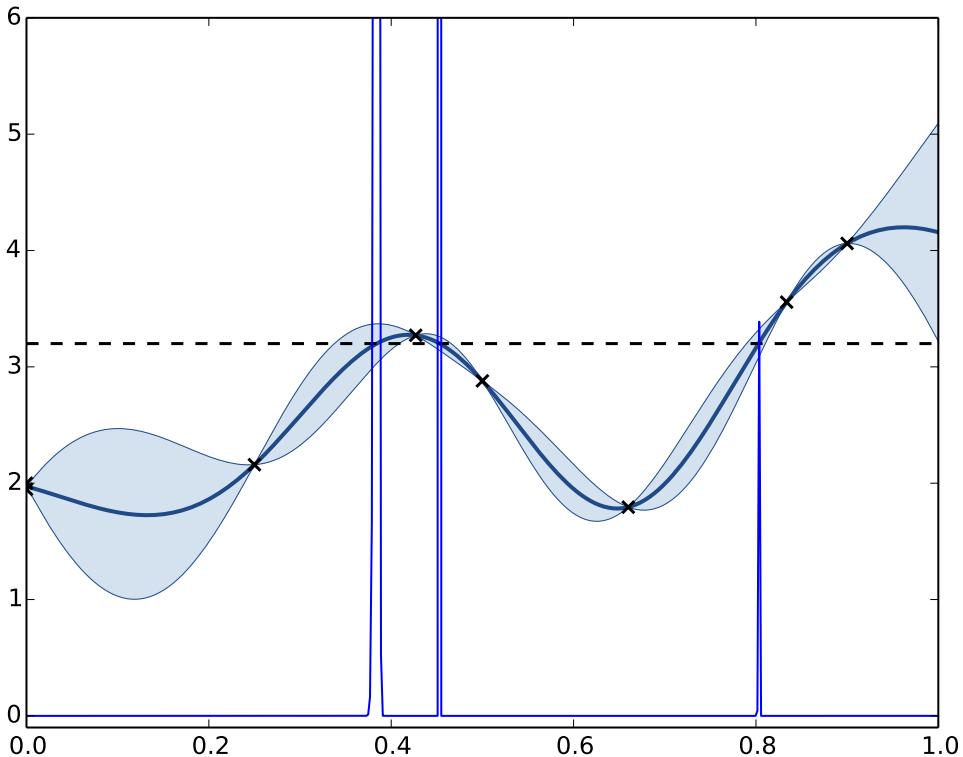
iteration 1:



iteration 2:



iteration 3:



Conclusion

Usual optimization methods such as

- gradient based methods
- evolutionary/genetic algorithms

are not relevant for costly to evaluate functions.

Once again, statistical models can be of great help...

- few number of function evaluations
- good noise filtering
- trade-off exploitation/exploration available