École chercheurs MEXICO, La Rochelle, Mars 2018

Model Based Optimization

Nicolas Durrande, nicolas@prowler.io

PROWLER.io, Cambridge (UK) – Mines St-Étienne (France)

Kernel Design

Making new from old: Many operations can be applied to psd functions while retaining this property

Kernels can be:

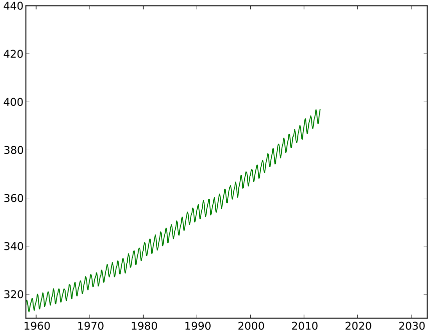
- Summed together
  - On the same space $k(x,y) = k_1(x,y) + k_2(x,y)$
  - On the tensor space $k(x,y) = k_1(x_1,y_1) + k_2(x_2,y_2)$
- Multiplied together
  - On the same space $k(x,y) = k_1(x,y) \times k_2(x,y)$
  - On the tensor space $k(x,y) = k_1(x_1,y_1) \times k_2(x_2,y_2)$
- Composed with a function
  - $k(x,y) = k_1(f(x), f(y))$

How can this be useful?

Sum of kernels over the same space

Example (The Mauna Loa observatory dataset)

This famous dataset compiles the monthly CO_2 concentration in Hawaii since 1958.

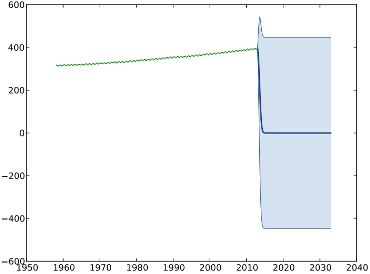


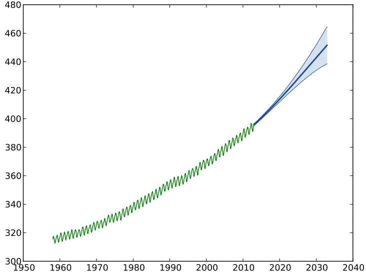
Let's try to predict the concentration for the next 20 years.

Sum of kernels over the same space

We first consider a squared-exponential kernel:

$$k_{se}(x, y) = \sigma^2 \exp\left(-\frac{(x-y)^2}{\theta^2}\right)$$





The results are terrible!

Sum of kernels over the same space

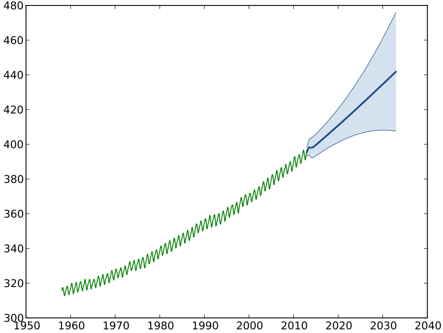
What happen if we sum both kernels?

$$k(x,y) = k_{se1}(x,y) + k_{se2}(x,y)$$

Sum of kernels over the same space

What happen if we sum both kernels?

$$k(x,y) = k_{se1}(x,y) + k_{se2}(x,y)$$



The model is drastically improved!

Sum of kernels over the same space

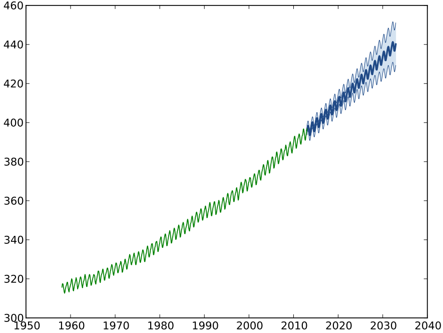
We can try the following kernel:

$$k(x, y) = \sigma_0^2 x^2 y^2 + k_{se1}(x, y) + k_{se2}(x, y) + k_{per}(x, y)$$

Sum of kernels over the same space

We can try the following kernel:

$$k(x, y) = \sigma_0^2 x^2 y^2 + k_{se1}(x, y) + k_{se2}(x, y) + k_{per}(x, y)$$



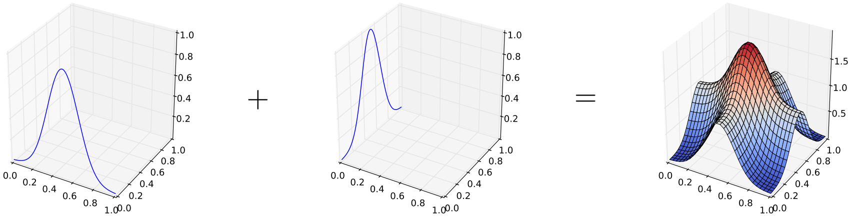
Once again, the model is significantly improved.

Sum of kernels over tensor space

Property

$$k(x,y) = k_1(x_1,y_1) + k_2(x_2,y_2)$$

is valid covariance structure.

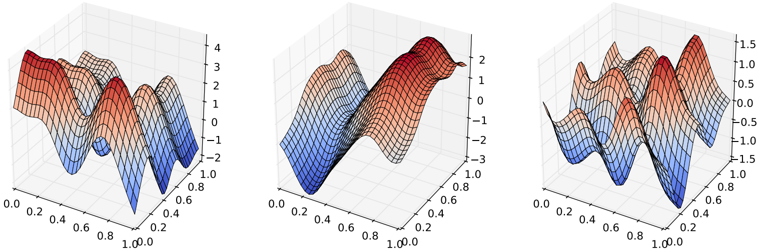


Remark: From a GP point of view, k is the kernel of

$$Z(x) = Z_1(x_1) + Z_2(x_2)$$

Sum of kernels over tensor space

We can have a look at a few sample paths from Z:



⇒ They are additive (up to a modification)

Tensor Additive kernels are very useful for

- Approximating additive functions
- Building models over high dimensional inputs spaces

Product over the same space

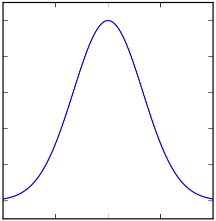
Property

$$k(x,y) = k_1(x,y) \times k_2(x,y)$$

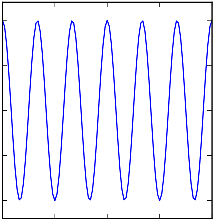
is valid covariance structure.

Example

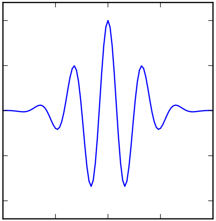
We consider the product of a squared exponential with a cosine:



×



=



Product over the tensor space

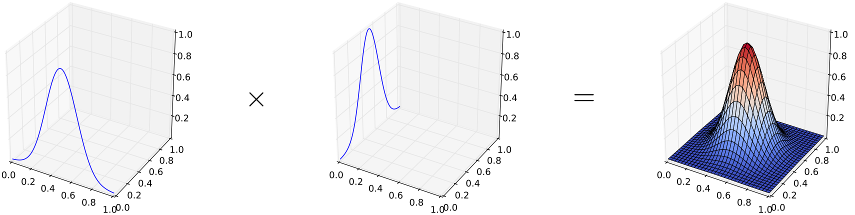
Property

$$k(x, y) = k_1(x_1, y_1) \times k_2(x_2, y_2)$$

is valid covariance structure.

Example

We multiply 2 squared exponential kernel



Calculation shows this is the usual 2D squared exponential kernel.

Composition with a function

Property

Let k_1 be a kernel over $D_1 \times D_1$ and f be an arbitrary function $D \to D_1$, then

$$k(x, y) = k_1(f(x), f(y))$$

is a kernel over $D \times D$.

proof

$$\sum \sum a_i a_j k(x_i, x_j) = \sum \sum a_i a_j k_1(\underbrace{f(x_i)}_{y_i}, \underbrace{f(x_j)}_{y_j}) \geq 0$$

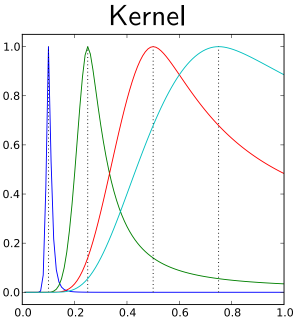
Remarks:

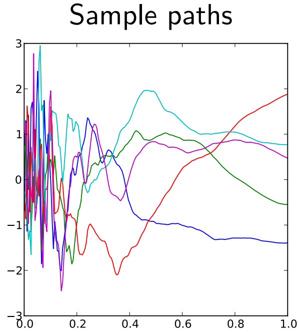
- k corresponds to the covariance of $Z(x) = Z_1(f(x))$
- This can be seen as a (non-linear) rescaling of the input space

Example

We consider $f(x) = \frac{1}{x}$ and a Matérn 3/2 kernel $k_1(x, y) = (1 + |x - y|)e^{-|x-y|}$.

We obtain:





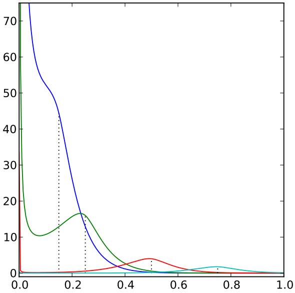
All these transformations can be combined!

Example

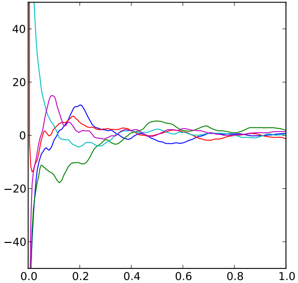
$k(x,y) = f(x)f(y)k_1(x,y)$ is a valid kernel.

This can be illustrated with $f(x) = \frac{1}{x}$ and $k_1(x,y) = (1+|x-y|)e^{-|x-y|}$:

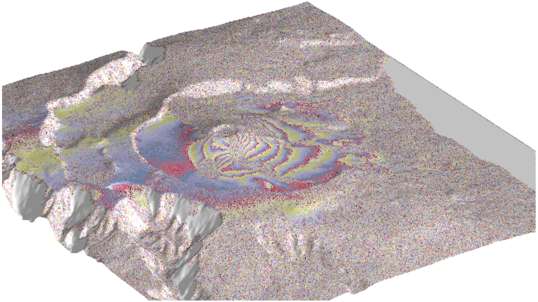
Kernel



Sample paths



Example



⇒ R demo

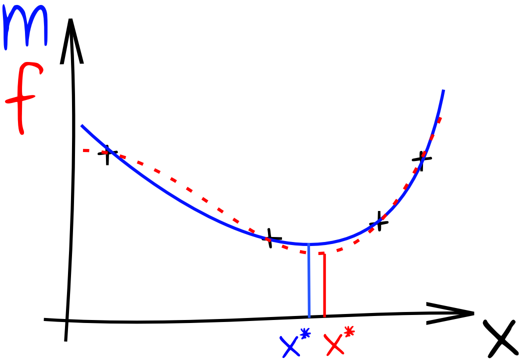
Other kernel design methods

There are two other popular methods for kernel design:

- Bochner Theorem
  There is an equivalence between positive measures and stationnary positive definite functions.
- Linear operators
  If the function to approximate has particular properties that can be obtained via a linear transform, it is possible to build a GP with the wanted properties. For example, one can build symmetric GPs or GPs with integral equal to zero.

Model based optimisation

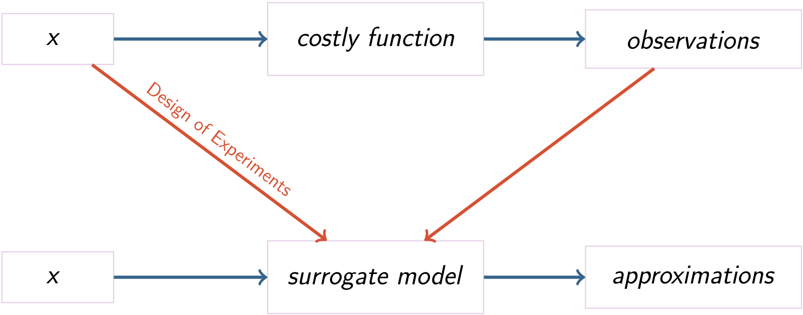
If the number of function evaluations is limited, we can run the optimization on the model instead of running it directly on the function



In the end, we hope that:

$$\text{argmin}(m) \approx \text{argmin}(f)$$
$$\min(m) \approx \min(f)$$

Overall framework



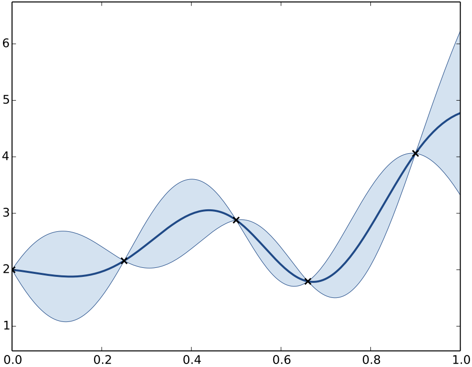
In practice, it is risky to take decisions based only on the model...

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

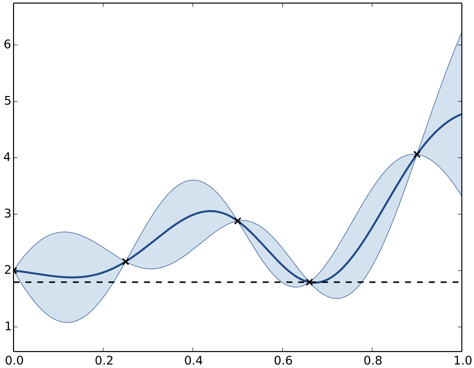
Global optimization methods are a trade-off between

- Exploitation of past good results
- Exploration of the space

How can GPR models be helpful?



In our example, the best observed value is 1.79

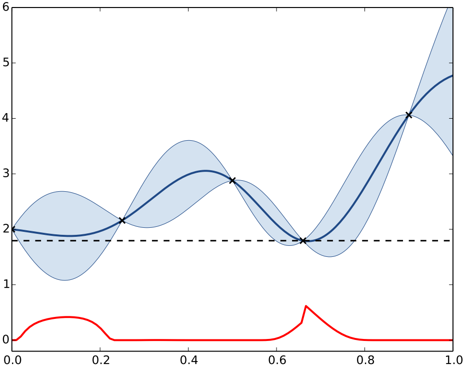


Various criteria can be studied

- 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 $PI(x) \geq 0.5$.

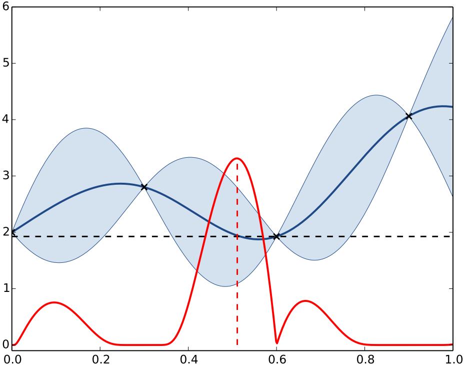
For such points, the improvement cannot be large...

Can we find another criterion?

Expected Improvement:

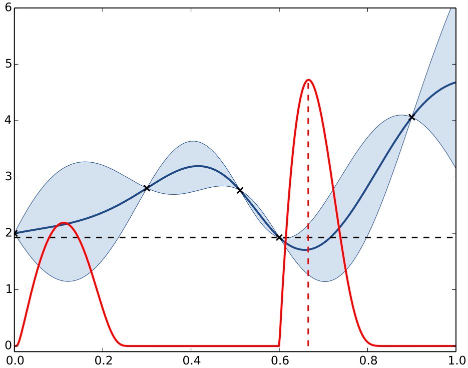
$$EI(x) = \int_{-\infty}^{\min(F)} \max\left(0, Y(x)\right) \; dy(x) = \cdots =$$

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



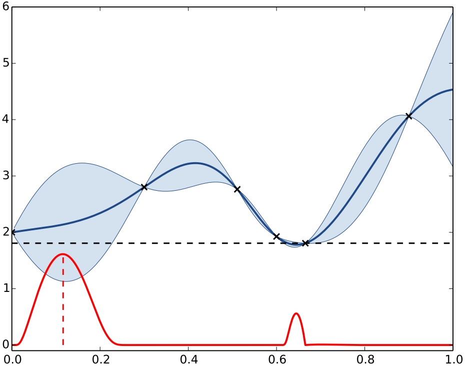
Expected Improvement

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



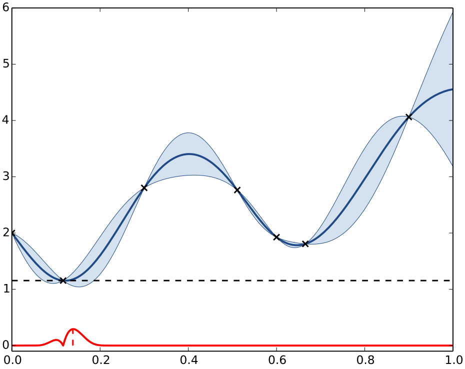
Expected Improvement

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



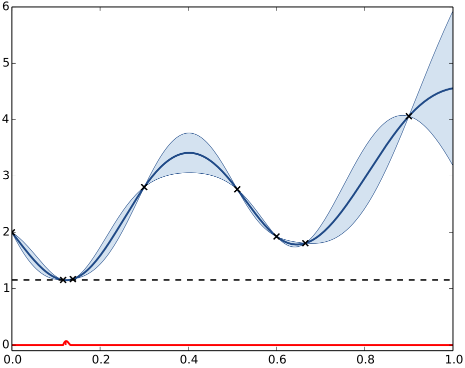
Expected Improvement

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



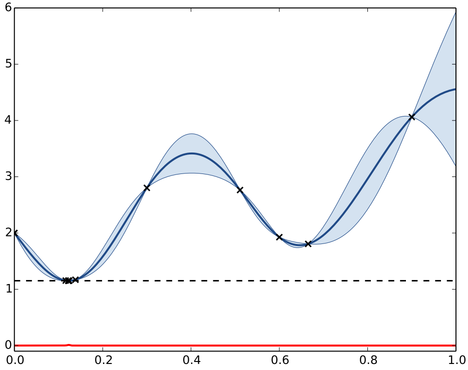
Expected Improvement

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



Expected Improvement

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



Expected Improvement

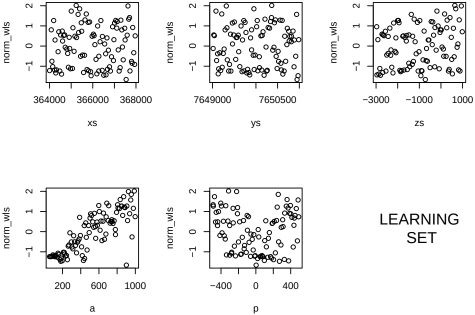
This algorithm is called **Efficient Global Optimization** (EGO, Jones et al., 1998):

1. make an initial design of experiments X and calculate the associated F, $t = \texttt{length}(F)$
2. built a GP from (X, F) (max. log-likelihood on σ and θ_i's)
3. $X_{t+1} = \arg\max_x EI(x)$
4. calculate $F_{t+1} = f(X_{t+1})$, increment t
5. stop ($t > t^{\text{max}}$) or go to 2.

+ EGO provides a good trade-off between exploitation and exploration without arbitrary parameters.
+ It requires few function observations (10 in the example) to get close to optimal regions.

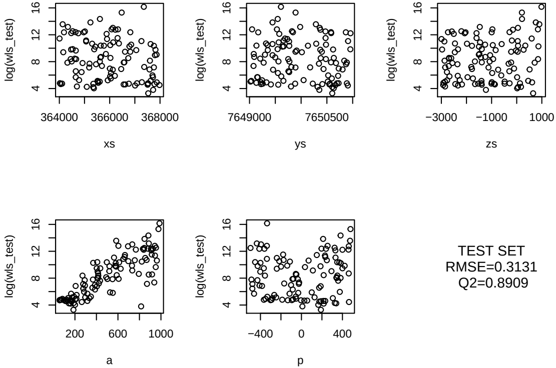
Example in 5d: surface displacements misfit minimization

$\Rightarrow$ demo with `mainInversionPunctualDisplSource.R`
!!! normalize the data: WLS has a few very large values, it is always > 0: make it more gaussian, $\texttt{wls}\_norm = \log(1 + \texttt{wls})$ and all x's and $\texttt{wls}\_norm$ between 0 and 1.



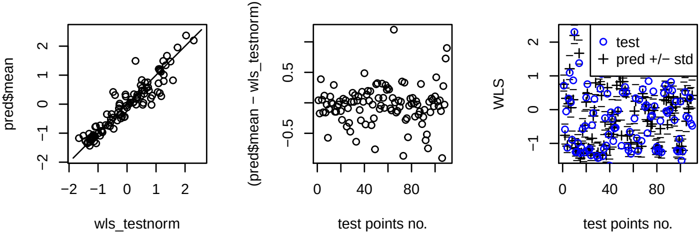
100 $\{xs, ys, zs, a, p\}$ points chosen through an optimized Latin Hypercube Sampling (`R` libraries `DiceDesign` or `lhs`).

(demo with `mainInversionPunctualDisplSource.R`, cont.)



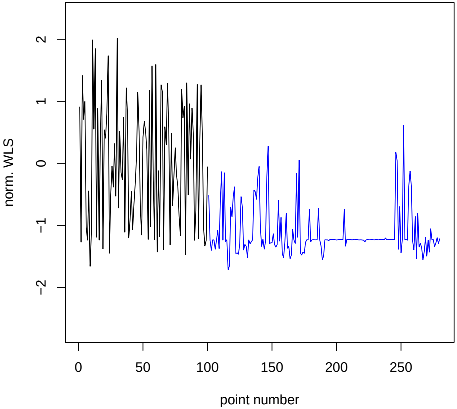
110 random $\{xs, ys, zs, a, p\}$ test points.

(demo with `mainInversionPunctualDisplSource.R`, cont.)



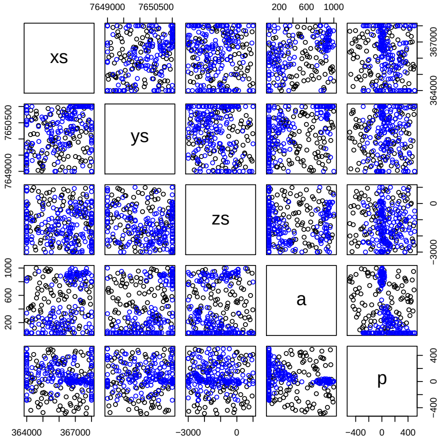
110 test points.

(demo with `mainInversionPunctualDisplSource.R`, cont.)
EGO parameters: anisotropic Matèrn 5/2 kernel, GP updated (log-likelihood maximized) every 5 added points, BFGS with bounded variables (from `optim()` function) restarted from random initial points for maximizing log-likelihood and EI.



Preferential sampling of good regions of S, but global therefore sometimes increasing WLS. Lower bound on θ_i's increased from 0.08 to 0.1 at $t = 250$ (x_i's and θ_i's normed between 0 and 1).

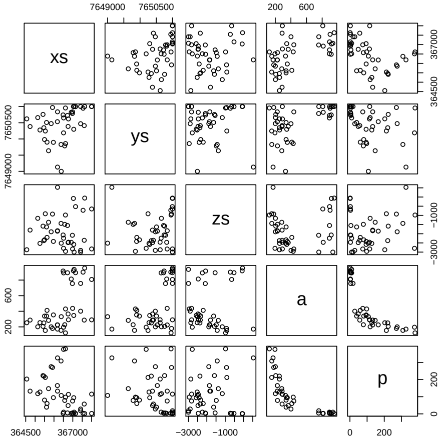
(demo with `mainInversionPunctualDisplSource.R`, cont.)



Black: LHS initial points.
Blue: EGO points.

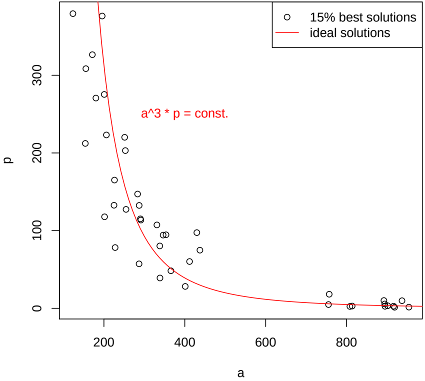
Note the patterns in new points. Accumulation at lower bound of a and mid interval of p before $t = 250$.

(demo with `mainInversionPunctualDisplSource.R`, cont.)



15% best sampled points. Note the "function" for the (a, p) pair, i.e., $a^{\star}(p^{\star})$.

(demo with `mainInversionPunctualDisplSource.R`, cont.)



Mogi model only dependency in a and p is through $a^3 \times p$: it is not identifiable.

EGO tells it by preferential sampling in the valley

$$a^3 \times p = \text{const.} = a^{\star 3} \times p^\star$$

Other EGO output: a statistical model of WLS.
The last length scales are an indication of the sensitivity of WLS to each variable: a, p and zs are very sensitive (θ_i's small, in $[0.08, 0.1]$), xs a little sensitive (θ in $[0.1, 2.5]$) and ys insensitive ($\theta \approx 3$).

Difficulties and challenges with EGO

- Standard GPs are limited to $n \approx 1000$ points (covariance matrix inversion).
- EGO clusters points in good regions, the covariance matrix may become ill-conditionned if length scales θ_i are too large w.r.t. X.
- Although the method perfectly applies to large dimensional spaces ($d > 100$), larger d may require larger n, back 2 lines above.
- EGO does not converge in the traditional sense: it creates dense samples in the volume of S. The efficiency comes from the order in which points are sampled.

$\Rightarrow$ these are the topics of current research. Let's mention a few extensions next.

EGO continuations

- Parallelized EGO: estimate the *EI* of groups of points, cf. Ginsbourger et al.
- Finite budget: *EI* of a single x is only optimal at the last iteration. Theory of dynamic *EI*, cf. Ginsbourger et al.
- EGO and bad covariance matrix conditioning: replace points that are close-by by one point and the associated derivatives (cf. M. Osborn, L. Laurent), regularizations (cf. Le Riche et al.)
- SUR strategies: (Step-wise Uncertainty Reduction), reduce the entropy of the optimum (cf. Vasquez et al.), or the average probability of incursions below $min(F)$ (cf. Picheny).

Related problems addressed with GPs

- EGO with constraints: $\min_x f(x)$ s.t. $g(x) \leq 0$, multiply the *EI* by the probability of constraints satisfaction.
- GP for target attainment: find the set of x s.t. $f(x) = T$, change the *EI* into $c(x,x) \times \mathrm{pdf}\,((T - m(x))/sqrt(c(x,x)))$, cf. Picheny et al.
- GP for probability estimation: find $\mathbb{P}(f(x,U) \leq T)$ where U is a random vector.
- GP for multi-objective optimization: $\min_x \{f_1(x), \ldots f_m(x)\}$, cf. Binois et al.

Robust optimization

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

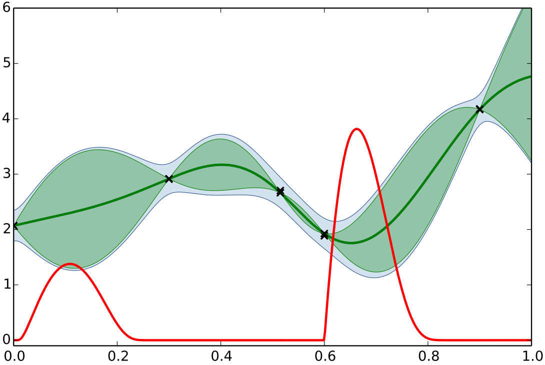
S1 Build a new model that interpolates $m(X)$ at X where $m(X)$ accounts for the noise (non interpolating GP, e.g. with a white noise part in the kernel).

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 (V. Picheny).

Solution 1

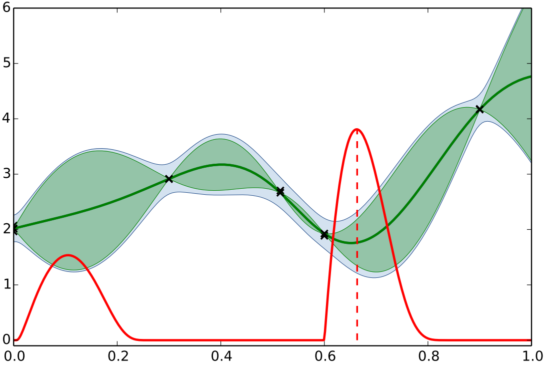
iteration 0



(noisy observations and their denoised versions are both shown as black crosses)

Solution 1

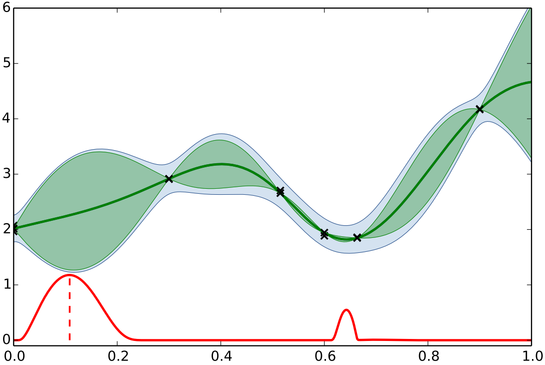
iteration 1



(noisy observations and their denoised versions are both shown as black crosses)

Solution 1

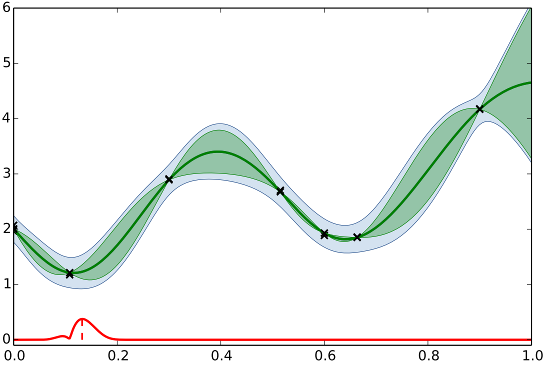
iteration 2



(noisy observations and their denoised versions are both shown as black crosses)

Solution 1

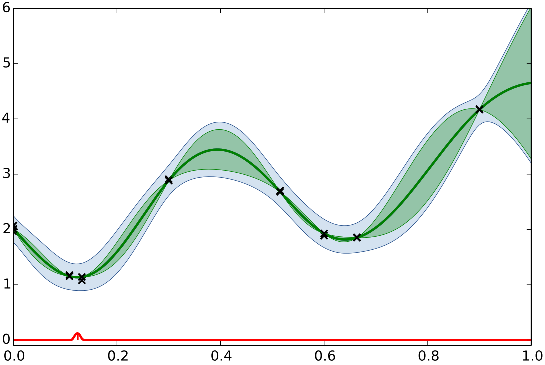
iteration 3



(noisy observations and their denoised versions are both shown as black crosses)

Solution 1

iteration 4



(noisy observations and their denoised versions are both shown as black crosses)