Scalable (and usable!) Bayesian Optimisation

Javier González
(with Zhenwen Dai, Philipp Hennig and Neil Lawrence)

University of Sheffield, Sheffield, UK

April 7th, 2016. SIAM-UQ, Lausanne, Switzerland.

The University Of Sheffield.

Sheffield Institute for Translational Neuroscience

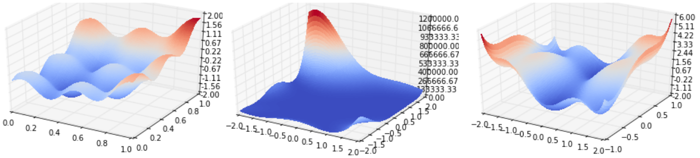
General goal of the talk

"Civilisation advances by extending the number of important operations which we can perform without thinking of them."
(Alfred North Whitehead)

- Scalable BO: models + parallelisation.
- Usable BO: new users + expert users.

Consider a *well behaved* function $f : \mathcal{X} \to \mathbb{R}$ where $\mathcal{X} \subseteq \mathbb{R}^D$ is (in principle) a bounded domain.

$$x_M = \arg\min_{x \in \mathcal{X}} f(x).$$

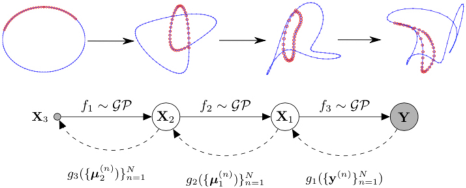


- f is explicitly unknown (computer model, process embodied in a physical process) and multimodal.
- Evaluations of f may be perturbed.
- Evaluations of f are (very) expensive.

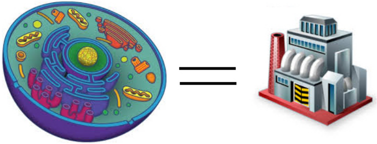
Expensive functions, who doesn't have one?

[Dai, Damianou, González and Lawrence, ICLR'2016]
[González et al. NIPS-ComBio 2014, 2015]

Model configuration: find learning rates, number of layers, etc



Design of experiments: Design synthetic genes that best enable cells to scale up the production of proteins of interest.



Probabilistic numerics approach?

http://www.probabilistic-numerics.org/, Michael Osborne, Philipp Hennig

Make a series of $x_1, \ldots, x_N$ evaluations of f to minimise *cumulative regret*

$$r_N = \sum_{n=1}^{N} f(x_n) - N f(x_M)$$

1. *Optimisation* as *decision*: Minimise the regret.
2. *Decision* as *inference*: need to model the *epistemic* uncertainty we have about f.

Probability theory to model uncertainty

Bayesian Optimisation

[Mockus, 1978]

Methodology to perform global optimisation of multimodal black-box functions.

1. Choose some *prior measure* over the space of possible objectives f.

2. Combine prior and the likelihood to get a *posterior measure* over the objective given some observations.

3. Use the posterior to decide where to take the next evaluation according to some *acquisition/loss function*.

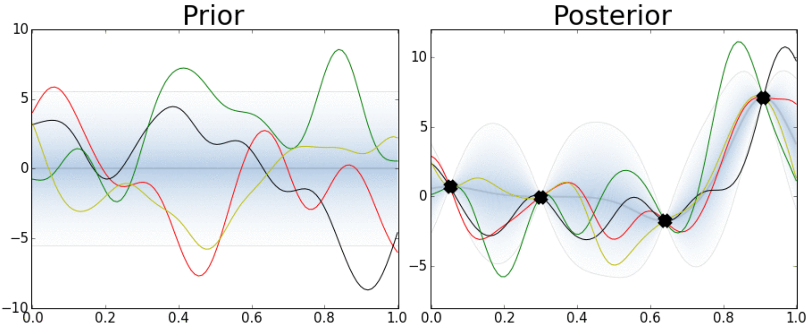
4. Augment the data.

Iterate between 2 and 4 until the evaluation budget is over.

Probability measure over functions

Gaussian processes [Rasmunsen and Williams, 2006]

Infinite-dimensional probability density, such that each linear finite-dimensional restriction is multivariate Gaussian.

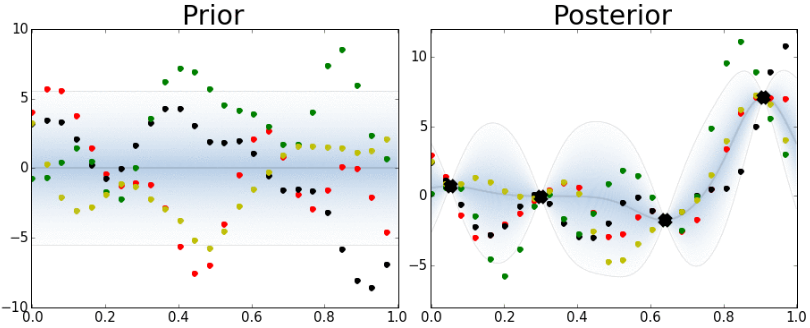


- Fully determined by a covariance function $k(\mathbf{x}, \mathbf{x}'; \theta)$ operator.
- Marginals are Gaussians with known mean and variance.

Probability measure over functions

Gaussian processes [Rasmunsen and Williams, 2006]

Infinite-dimensional probability density, such that each linear finite-dimensional restriction is multivariate Gaussian.

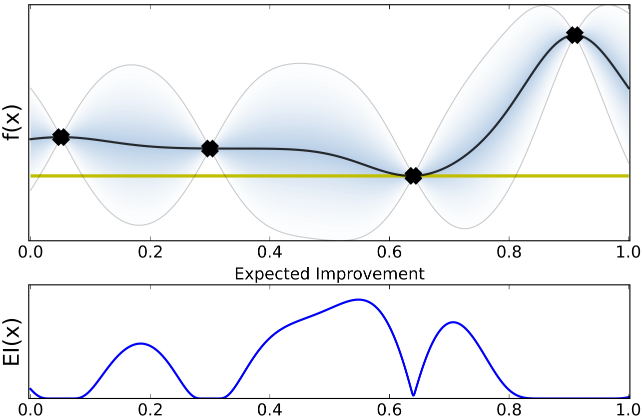


- Fully determined by a covariance function $k(\mathbf{x}, \mathbf{x}'; \theta)$ operator.
- Marginals are Gaussians with known mean and variance.

Expected Improvement

[Jones et al, 1998]

$$\alpha_{EI}(\mathbf{x};\theta,\mathcal{D}) \triangleq \mathbb{E}[\max(0, y_{best} - y)]$$



Exploration vs. exploitation to determine the next evaluation.

Illustration of BO

Iteration 1

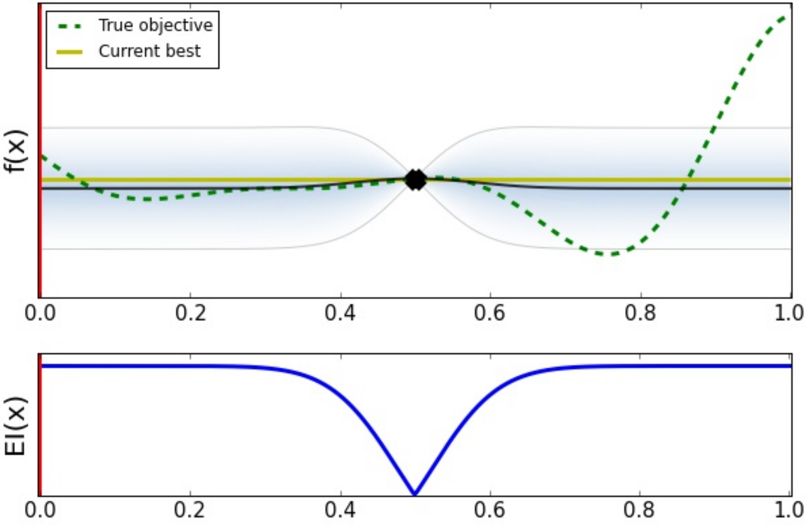


Illustration of BO

Iteration 2

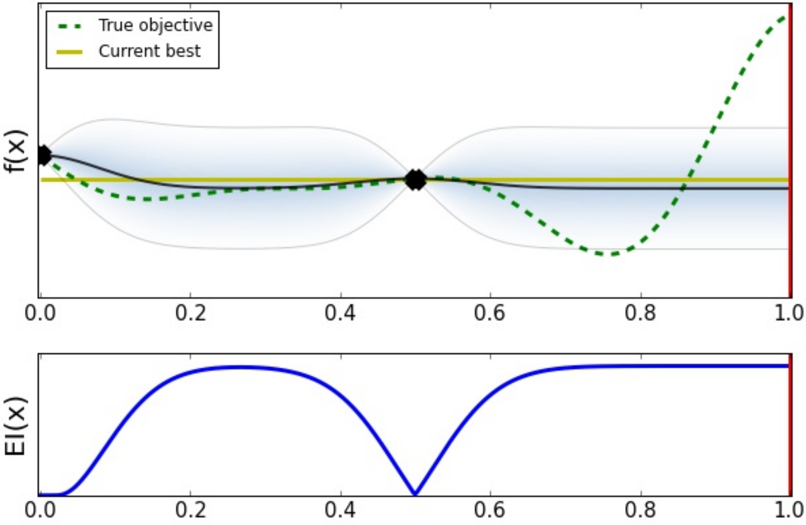


Illustration of BO

Iteration 3

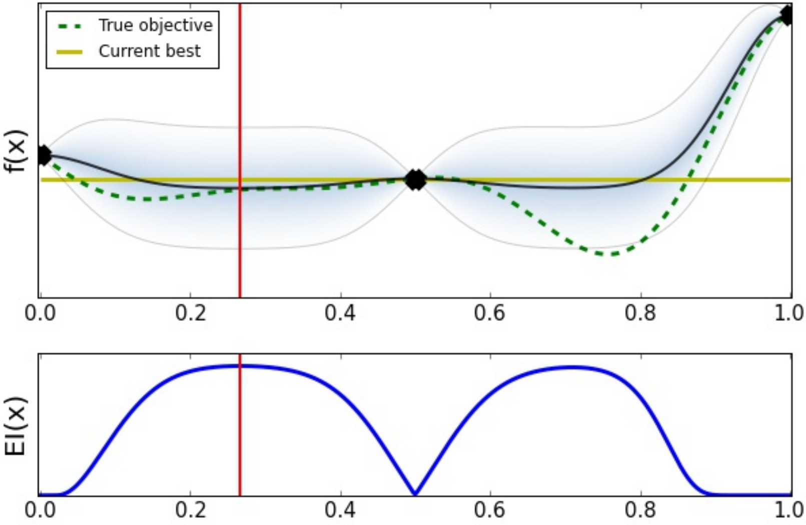


Illustration of BO

Iteration 4

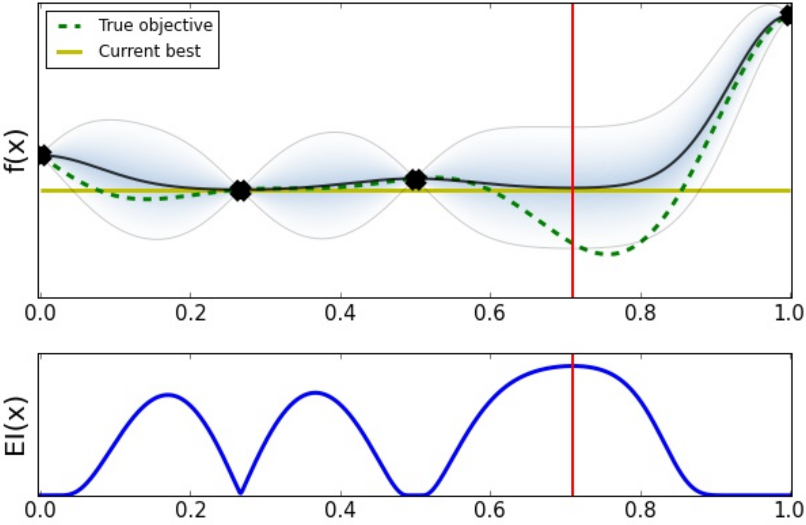


Illustration of BO

Iteration 5

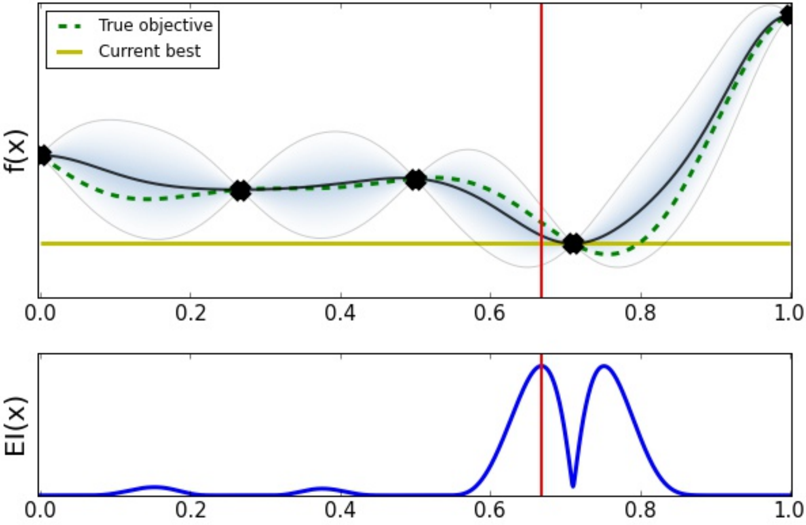


Illustration of BO

Iteration 6

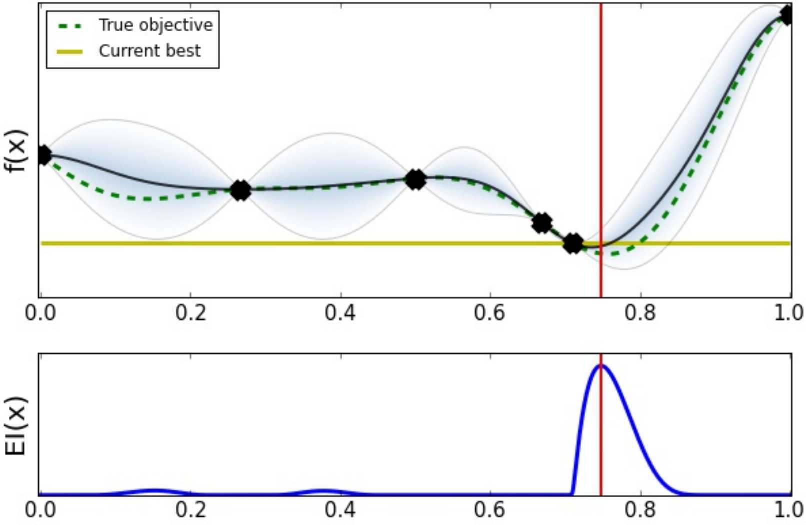


Illustration of BO

Iteration 8

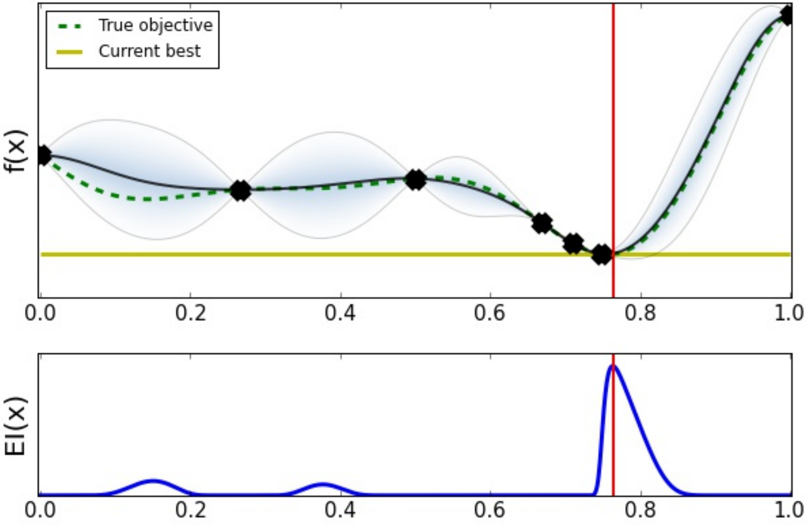
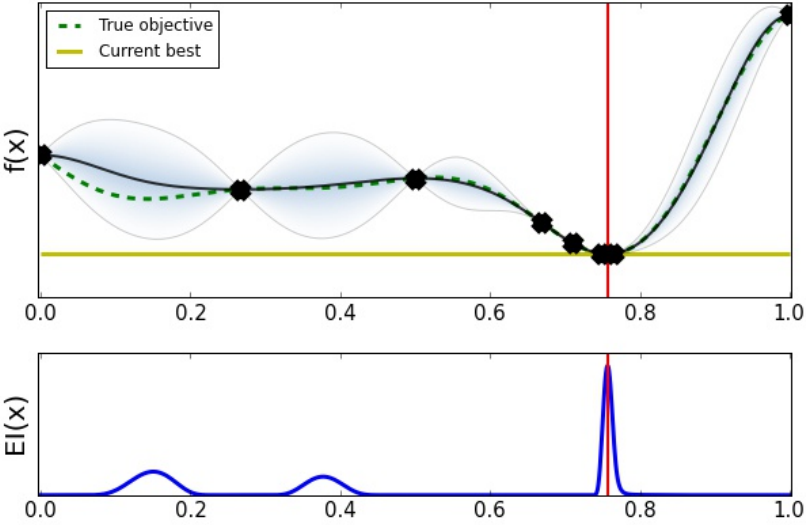


Illustration of BO

Iteration 8



Why these ideas have been ignored for years?

- Lack of general software to apply these methods as a black optimisation boxes of for experimental design.
- Reduced scalability in dimensions, number of evaluations (parallelisation) and available models.

Usable BO: GPyOpt

http://sheffieldml.github.io/GPyOpt/

- Easy python interface (compatible with spearmint).
- Based on GPy: GPs, Sparse GPs, Warped GPs, Deep GPs, etc.
- MCMC integration of the acquisition functions.
- Parallel (synchronous batch) optimisation.
- Constrain optimisation.
- Armed bandits optimisation.
- Handles continous and discrete inputs.
- Several acquisition optimisers.
- More to come!

Open source code (BSD-3 license). You can contribute!

Scalable BO: Parallel/batch BO

Avoiding the bottleneck of evaluating f

- Cost of $f(\mathbf{x}_n)$ = cost of $\{f(\mathbf{x}_{n,1}), \ldots, f(\mathbf{x}_{n,nb})\}$.
- Many cores available, simultaneous lab experiments, etc.

Considerations when designing a batch

- Available pairs $\{(\mathbf{x}_j, y_i)\}_{i=1}^{n}$ are augmented with the evaluations of f on $\mathcal{B}_t^{n_b} = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,nb}\}$.
- Goal: design $\mathcal{B}_1^{n_b}, \ldots, \mathcal{B}_m^{n_b}$.

Notation:

- $\mathcal{I}_n$: data set $\mathcal{D}_n$ + $\mathcal{GP}$ structure ($\mathcal{I}_{t,k}$ in the batch context).
- $\alpha(\mathbf{x}; \mathcal{I}_n)$: generic acquisition function given $\mathcal{I}_n$.

Optimal greedy batch design

Design a batch optimally is intractable

Sequential policy: Maximise:

$$\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$$

Greedy batch policy, k-th element t-th batch: Maximize:

$$\int \alpha(\mathbf{x}; \mathcal{I}_{t,k-1}) \prod_{j=1}^{k-1} p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1}) p(\mathbf{x}_{t,j}|\mathcal{I}_{t,j-1}) d\mathbf{x}_{t,j} dy_{t,j}$$

- $p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1})$: predictive distribution of the $\mathcal{GP}$.
- $p(\mathbf{x}_j|\mathcal{I}_{t,j-1}) = \delta(\mathbf{x}_{t,j} - \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha(\mathbf{x}; \mathcal{I}_{t,j-1}))$.

Available approaches

[Azimi et al., 2010; Desautels et al., 2012; Chevalier et al., 2013; Contal et al. 2013]

Bottleneck

Available methods require to iteratively update $p(y_{t,j}|\mathbf{x}_j, \mathcal{I}_{t,j-1})$ to model the iteration between the elements in the batch: $\mathcal{O}(n^3)$

How to design batches reducing this cost? Local penalisation

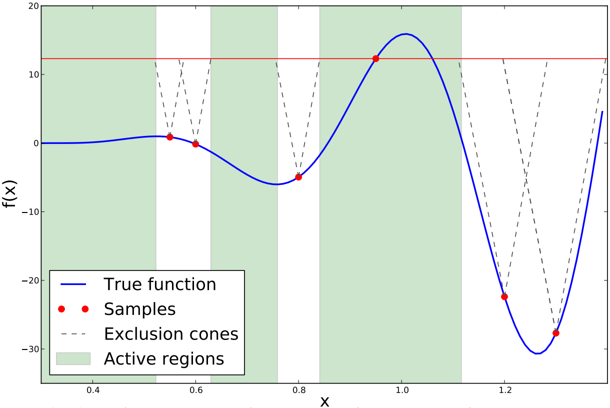
Lipschitz continuity

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|_p.$$

Interpretation of the Lipschitz continuity of f

$M = \max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$ and $B_{r_{x_j}}(\mathbf{x}_j) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{x}_j\| \leq r_{x_j}\}$ where

$$r_{x_j} = \frac{M - f(\mathbf{x}_j)}{L}$$



$x_M \notin B_{r_{x_j}}(\mathbf{x}_j)$ otherwise, the Lipschitz condition is violated.

Probabilistic version of $B_{r_x}(\mathbf{x})$

We can do this because $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$

- r_{x_j} is Gaussian with $\mu(r_{x_j}) = \frac{M-\mu(\mathbf{x}_j)}{L}$ and $\sigma^2(r_{x_j}) = \frac{\sigma^2(\mathbf{x}_j)}{L^2}$.

Local penalisers: $\varphi(\mathbf{x}; \mathbf{x}_j) = p(\mathbf{x} \notin B_{r_{\mathbf{x}_j}}(\mathbf{x}_j))$

$$\begin{aligned}\varphi(\mathbf{x}; \mathbf{x}_j) &= p(r_{\mathbf{x}_j} < \|\mathbf{x} - \mathbf{x}_j\|) \\ &= 0.5\mathrm{erfc}(-z)\end{aligned}$$

where $z = \frac{1}{\sqrt{2\sigma_n^2(\mathbf{x}_j)}}(L\|\mathbf{x}_j - \mathbf{x}\| - M + \mu_n(\mathbf{x}_j))$.

- Reflects the size of the ‘Lipschitz’ exclusion areas.
- Approaches to 1 when $\mathbf{x}$ is far form $\mathbf{x}_j$ and decreases otherwise.

Idea to collect the batches

Without using explicitly the model.

Optimal batch: maximisation-marginalisation

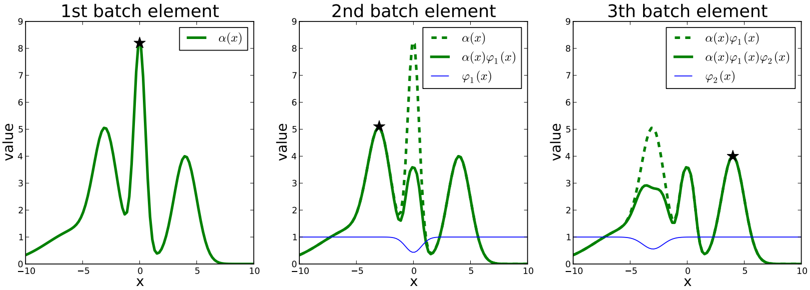
$$\int \alpha(\mathbf{x}; \mathcal{I}_{t,k-1}) \prod_{j=1}^{k-1} p(y_{t,j}|\mathbf{x}_{t,j}, \mathcal{I}_{t,j-1}) p(\mathbf{x}_{t,j}|\mathcal{I}_{t,j-1}) d\mathbf{x}_{t,j} dy_{t,j}$$

Proposal: maximisation-penalisation.

Use the $\varphi(\boldsymbol{x}; \boldsymbol{x}_j)$ to penalise the acquisition and predict the expected change in $\alpha(\boldsymbol{x}; \mathcal{I}_{t,k-1})$.

Local penalisation strategy

[González, Dai, Hennig, Lawrence, 2016]

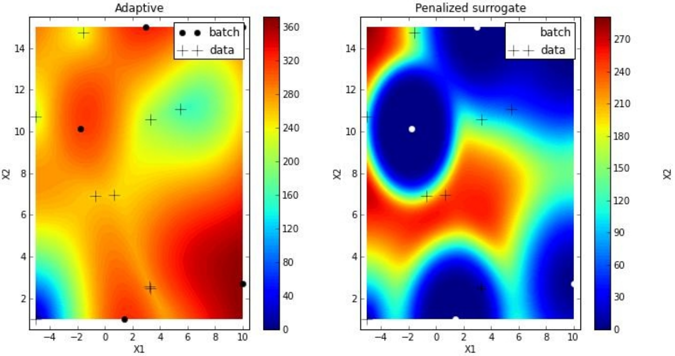


The maximization-penalisation strategy selects $\mathbf{x}_{t,k}$ as

$$\mathbf{x}_{t,k} = \arg\max_{x \in \mathcal{X}} \left\{ g(\alpha(\mathbf{x}; \mathcal{I}_{t,0})) \prod_{j=1}^{k-1} \varphi(\mathbf{x}; \mathbf{x}_{t,j}) \right\},$$

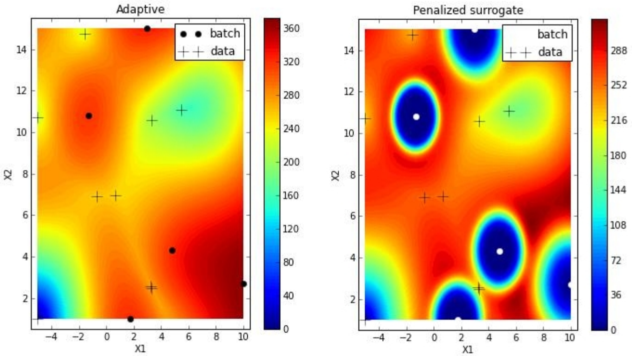
g is a transformation of $\alpha(\mathbf{x}; \mathcal{I}_{t,0})$ to make it always positive.

Example for $L = 50$



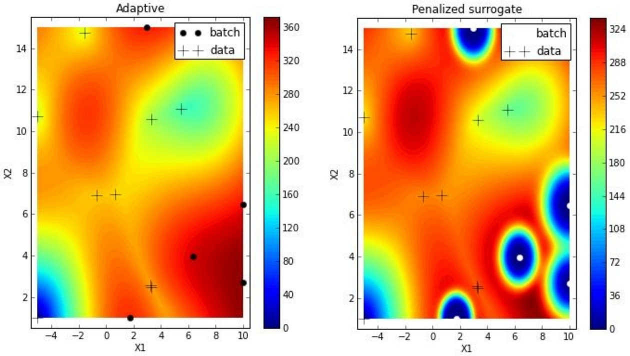
L controls the exploration-exploitation balance within the batch.

Example for $L = 100$



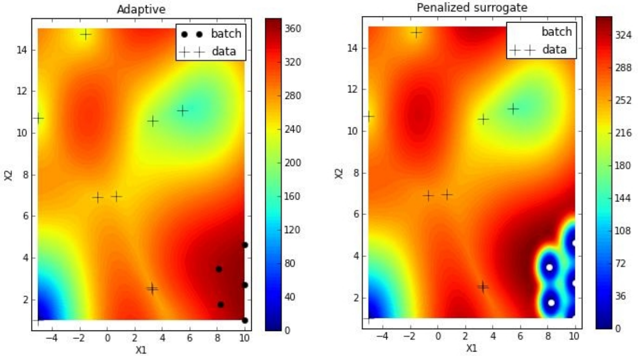
L controls the exploration-exploitation balance within the batch.

Example for $L = 150$



L controls the exploration-exploitation balance within the batch.

Example for $L = 250$

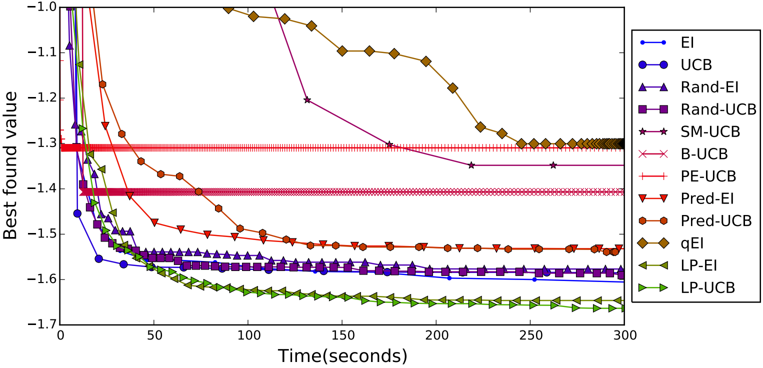


L controls the exploration-exploitation balance within the batch.

We choose $\hat{L} = \max_{\mathcal{X}} \|\mu_{\nabla}(\mathbf{x}^*)\|$.

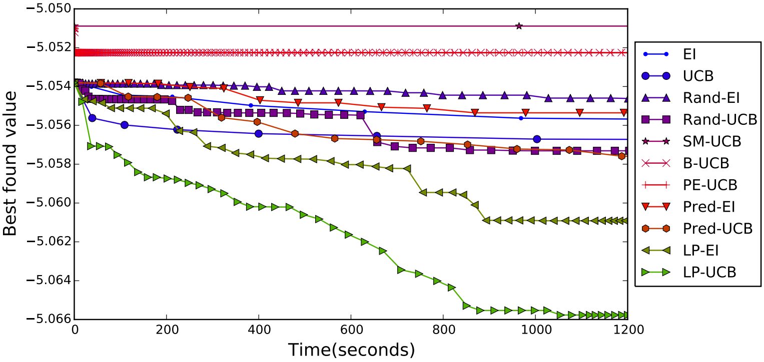
2D experiment with 'large domain'

Comparison in terms of the wall clock time



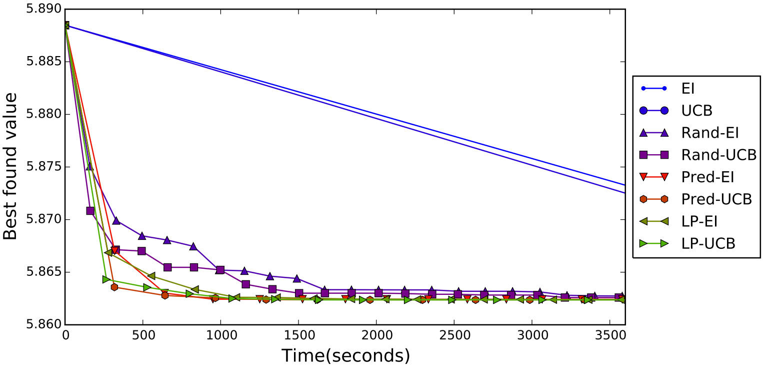
Optimisation of a fitted model for gene design

70 dimensions (gene features), emulator of the protein production by cells.



Support Vector Regression

- Minimisation of the RMSE on a test set over 3 parameters.
- 'Physiochemical' properties of protein tertiary structure?
- 45730 instances and 9 continuous attributes.



Wrapping up

- BO is fantastic tool for global parameter optimisation in ML and experimental design.
- To parallelise BO requires modelling the interaction between the elements in the batches to design. This can be done without updating the model explicitly after each batch element is collected.
- Software available! Use GPyOpt!