

Parallel Bayesian optimization with applications to synthetic gene design

Javier González

University of Sheffield, Sheffield, UK

February 23, 2016. Oxford, UK.



The
University
Of
Sheffield.

Sheffield Institute for
Translational Neuroscience

General goal of the talk

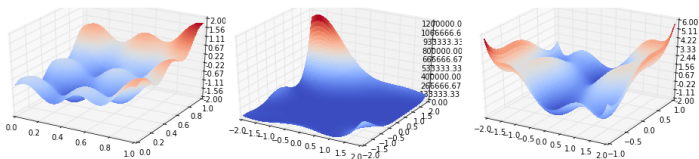
“Civilization advances by extending the number of important operations which we can perform without thinking of them.”
(Alfred North Whitehead)

- ▶ To configure statistical/ML models automatically.
- ▶ To automatically design sequential experiments to optimize physical processes.

General framework: global optimization

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

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

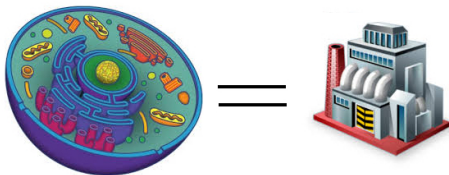


- ▶ f is **explicitly unknown** (computer model, process embodied in a physical process) and multimodal.
- ▶ Evaluations of f may be **perturbed**.
- ▶ Evaluations of f are (very) **expensive**.

Expensive functions, who doesn't have one?

[González, Lonworth, James and Lawrence, 2014, 2015]

Design of experiments: gene optimization



- ▶ Use mammalian cells to make protein products.
- ▶ Control the ability of the cell-factory to use synthetic DNA.

Optimize genes (ATTGGTUGA...) to best enable the cell-factory to operate most efficiently.

Regret minimization

Random of grid search require many function evaluations.

The goal is to make a series of x_1, \dots, x_N evaluations of f such that the *cumulative regret*

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

is minimized.

r_N is minimized if we start evaluating f at x_M as soon as possible.

Probabilistic numerics approach

<http://www.probablistic-numerics.org/>

1. Minimize the regret implies to see an *optimization* problem as a *decision* problem.
2. *Decision* problems can be seen as *inference*: need to model the *epistemic* uncertainty we have about the system we are studying.

Probability theory to model uncertainty

Bayesian Optimization

[Mockus, 1978]

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

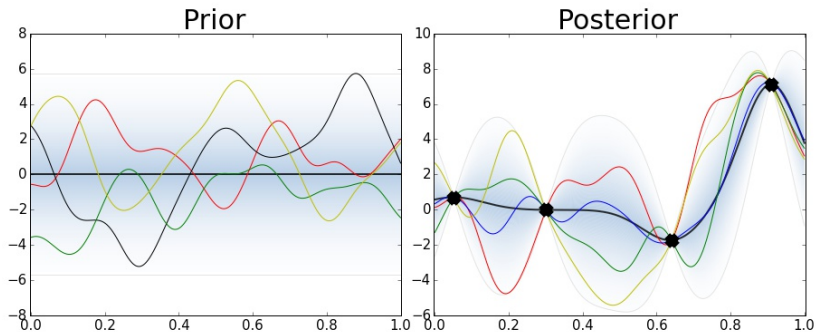
1. Choose some *prior measure* over the space of possible objectives f .
2. Combine prior and the likelihood to get a *posterior* over the objective given some observations.
3. Use the posterior to decide where to take the next evaluation according to some *acquisition/loss function*.
4. Augment the data.

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

Probability measure over functions

Gaussian processes [Rasmussen and Williams, 2006]

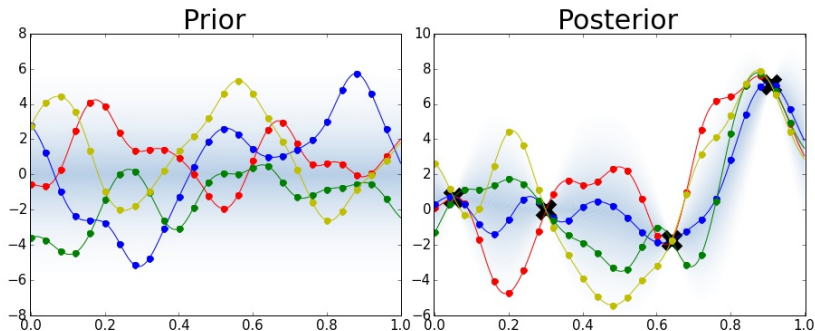
Infinite-dimensional probability density, such that each linear finite-dimensional restriction is multivariate Gaussian.



Probability measure over functions

Gaussian processes [Rasmussen and Williams, 2006]

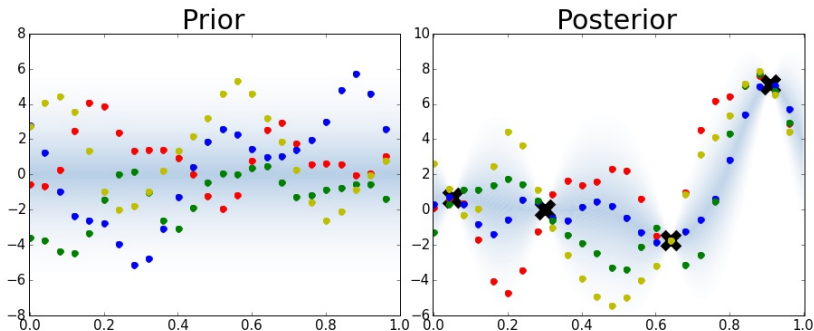
Infinite-dimensional probability density, such that each linear finite-dimensional restriction is multivariate Gaussian.



Probability measure over functions

Gaussian processes [Rasmussen and Williams, 2006]

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



Probability measure over functions

Gaussian processes [Rasmussen and Williams, 2006]

- ▶ GP is fully determined by a **covariance function** $k(\mathbf{x}, \mathbf{x}'; \theta)$ operator.
- ▶ Regression problems: $y_i = f(\mathbf{x}_i) + \epsilon_i$.
- ▶ Marginals at any \mathbf{x}_* are Gaussians with mean and variance

$$\mu(\mathbf{x}_*|\theta, \mathcal{D}) = \mathbf{k}_\theta(\mathbf{X}_*)^\top [\mathbf{k}_\theta + \sigma^2 \mathbf{I}]^{-1} \mathbf{y}$$

$$\sigma^2(\mathbf{x}_*|\theta, \mathcal{D}) = k_\theta(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_\theta(\mathbf{x}_*)^\top [\mathbf{K}_\theta + \sigma^2 \mathbf{I}]^{-1} \mathbf{k}_\theta(\mathbf{x}_*)$$

where \mathcal{D} is available dataset.

Acquisition functions

Making use of the model uncertainty

GPs has marginal closed-form for the posterior mean $\mu(\mathbf{x}_*)$ and variance $\sigma^2(\mathbf{x}_*)$.

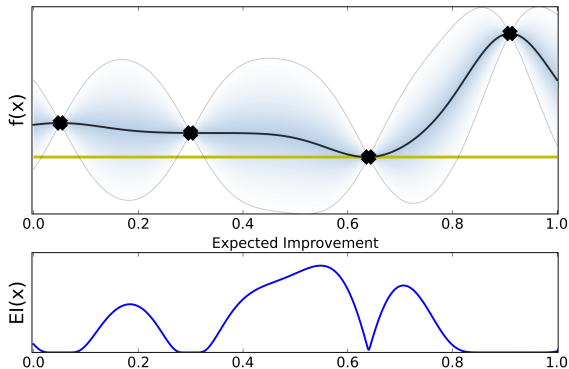
- ▶ **Exploration**: Evaluate in places where the variance is large.
- ▶ **Exploitation**: Evaluate in places where the mean is low.

Acquisition functions balance these two factors to determine where to evaluate next.

Expected improvement

[Jones et al., 1998]

$$\alpha_{EI}(\mathbf{x}; \theta, \mathcal{D}) = \int_y \max(0, y_{best} - y) p(y|\mathbf{x}; \theta, \mathcal{D}) dy$$



Bayesian Optimization

As a 'mapping' between two problems

BO is an strategy to transform the problem

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

unsolvable!

into a series of problems:

$$x_{n+1} = \arg \max_{x \in \mathcal{X}} \alpha(x; \mathcal{D}_n, \theta_n)$$

solvable!

where now:

- ▶ $\alpha(x)$ is inexpensive to evaluate.
- ▶ The gradients of $\alpha(x)$ are typically available.
- ▶ Still need to find x_{n+1} : gradient descent, DIRECT or other heuristics.

Illustration of BO

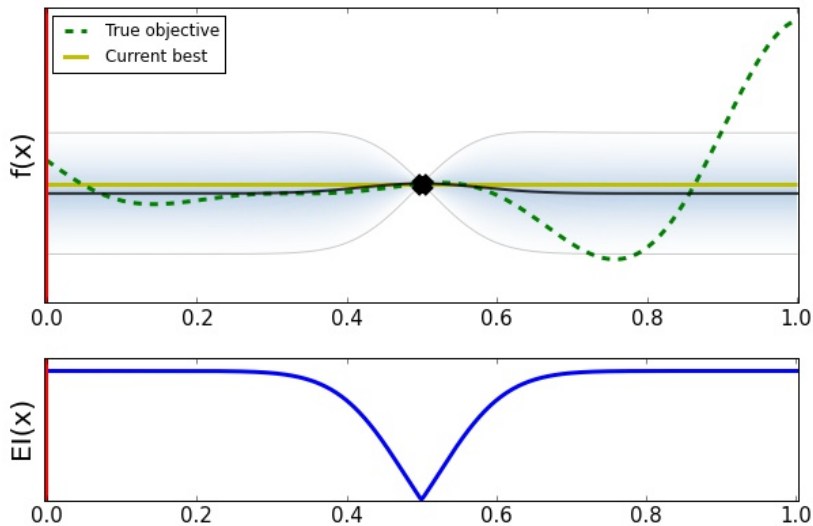


Illustration of BO

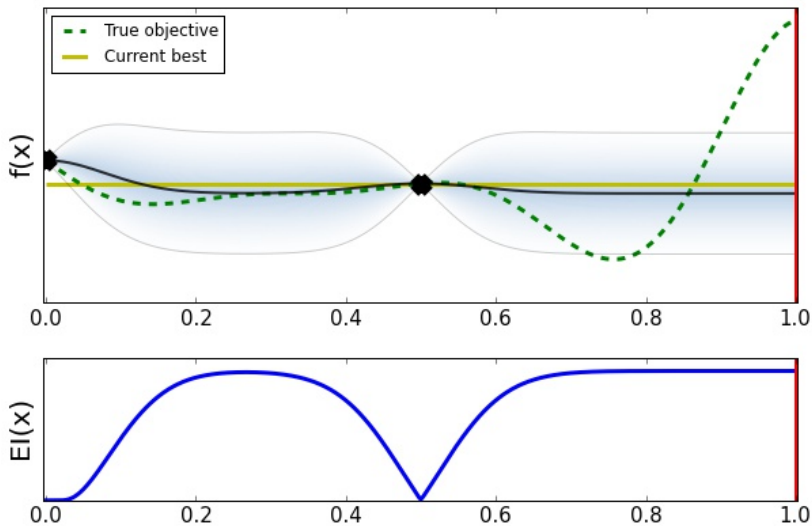


Illustration of BO

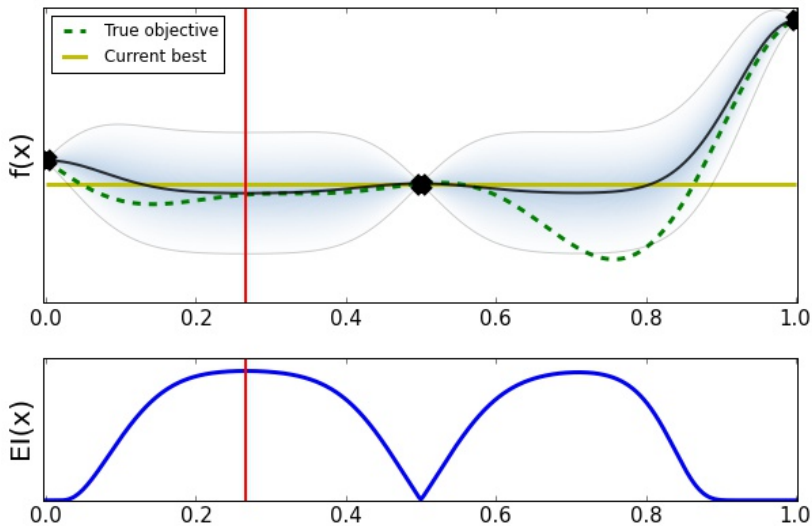


Illustration of BO

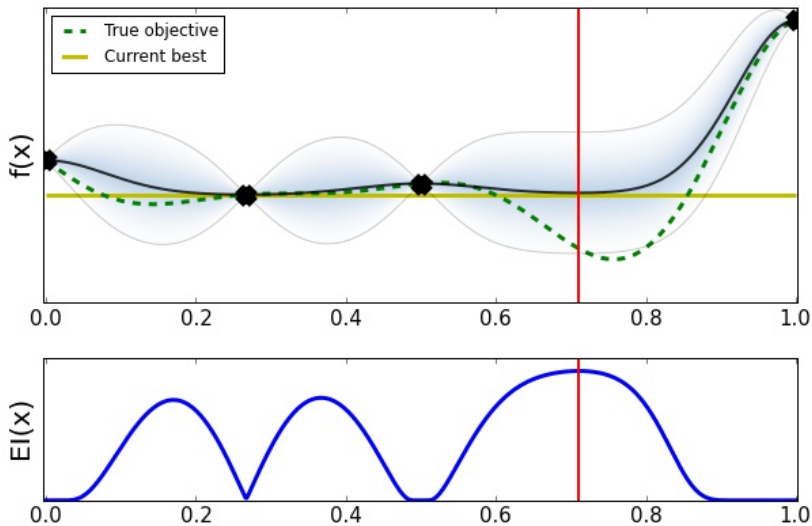


Illustration of BO

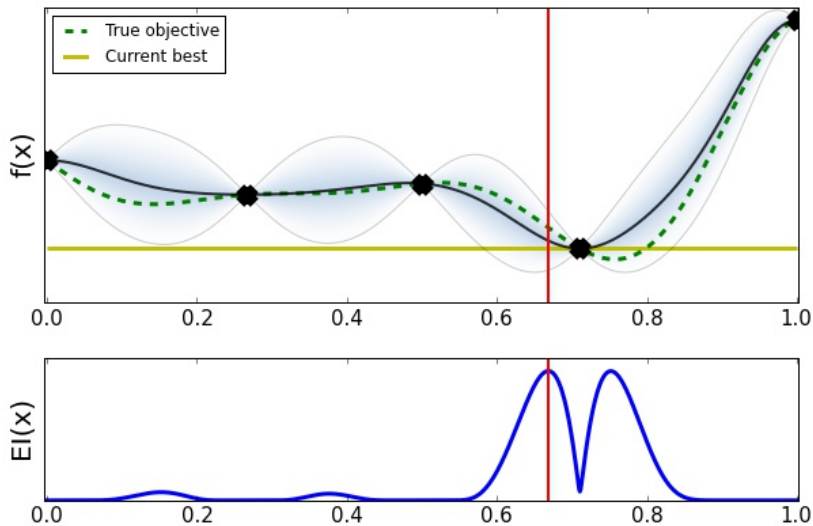


Illustration of BO

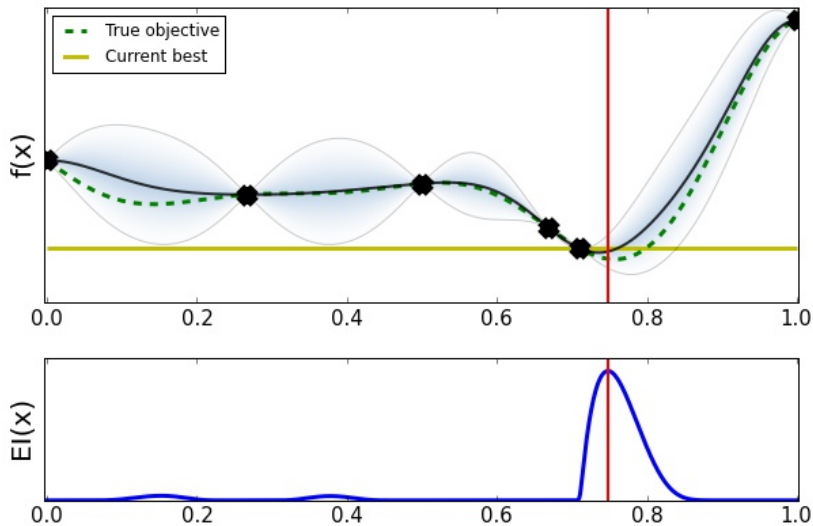


Illustration of BO

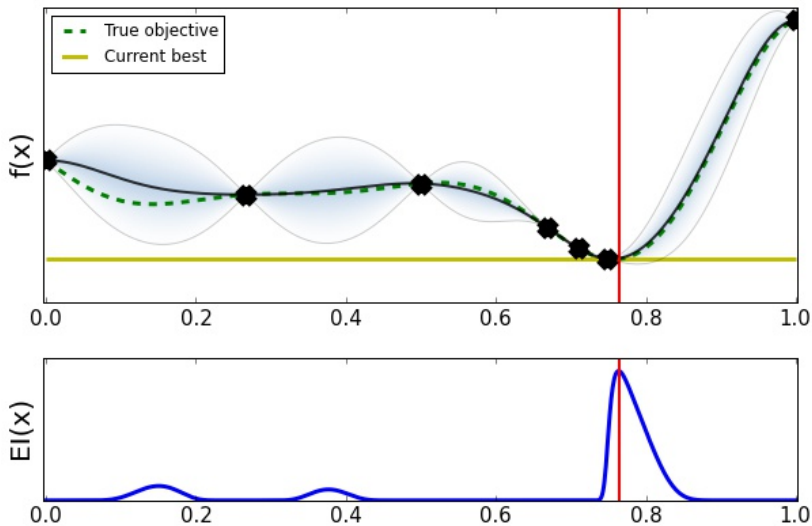
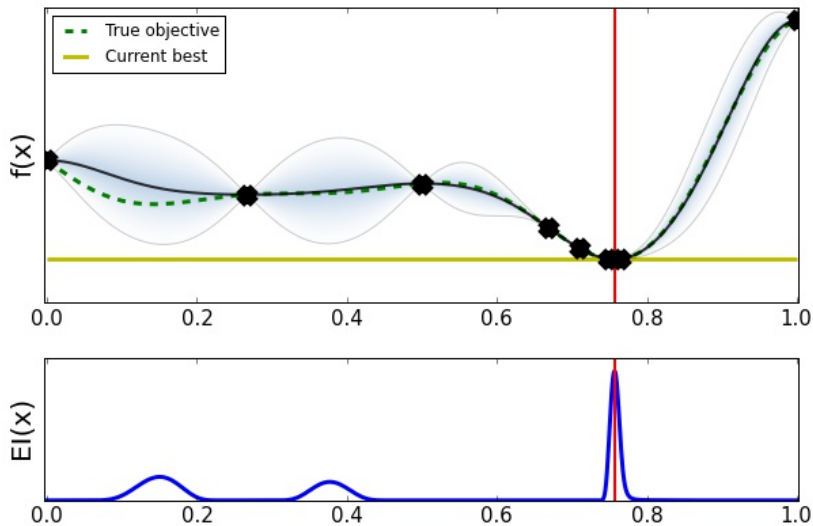


Illustration of BO



Why these ideas have been ignored for years?

- ▶ BO depends on its own (model) parameters.
- ▶ Reduced scalability in dimensions and number of evaluations.
- ▶ Lack of software to apply these methods as a black optimization boxes.

Open Software: GPyOpt

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

GPyOpt

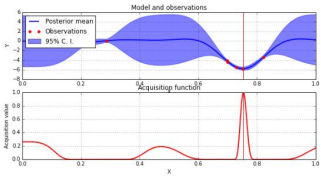
Tune your algorithms and design your wetlab experiments

Fork On GitHub

GPyOpt

- Why?
- Installation
- Documentation
- GPyOpt in...
- Releases
- Contact

GPyOpt



-- Python open-source library for Bayesian Optimization --

-- Developed by the [Machine Learning](#) group of the University of Sheffield --

-- Based on [GPy](#), python framework for Gaussian process modeling--

Why?

With GPyOpt you can:

- Solve global optimization problems with Bayesian optimization.

Open Software: GPyOpt

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

- ▶ Easy python interface (compatible with spearmin).
- ▶ Surrogate models available: GPs, sparse GPs, deep GPs, etc.
- ▶ MCMC integration of the acquisition functions.
- ▶ Parallel (synchronous batch) optimization.
- ▶ Constrain optimization.
- ▶ Handles continuous and discrete inputs.
- ▶ More to come!

Open source code. You can contribute!

Scalable BO: Parallel/batch BO

Avoiding the bottleneck of evaluating f



- ▶ Cost of $f(\mathbf{x}_n) = \text{cost of } \{f(\mathbf{x}_{n,1}), \dots, f(\mathbf{x}_{n,nb})\}$.
- ▶ 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 $\mathcal{B}_t^{n_b} = \{\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,n_b}\}$.
- ▶ Goal: design $\mathcal{B}_1^{n_b}, \dots, \mathcal{B}_m^{n_b}$.

Notation:

- ▶ \mathcal{I}_n : represents the available data set \mathcal{D}_n and the \mathcal{GP} structure when n data points are available ($\mathcal{I}_{t,k}$ in the batch context).
- ▶ $\alpha(\mathbf{x}; \mathcal{I}_n)$: generic acquisition function given \mathcal{I}_n .

Optimal greedy batch design

Sequential policy: Maximize:

$$\alpha(\mathbf{x}; \mathcal{I}_{t,0})$$

Optimal greedy batch design

Sequential policy: Maximize:

$$\alpha(\mathbf{x}; \mathcal{I}_{t,0})$$

Greedy batch policy, 1st element t -th batch: Maximize:

$$\alpha(\mathbf{x}; \mathcal{I}_{t,0})$$

Optimal greedy batch design

Sequential policy: Maximize:

$$\alpha(\mathbf{x}; \mathcal{I}_{t,0})$$

Greedy batch policy, 2nd element t-th batch: Maximize:

$$\int \alpha(\mathbf{x}; \mathcal{I}_{t,1}) p(y_{t,1} | \mathbf{x}_{t,1}, \mathcal{I}_{t,0}) p(\mathbf{x}_{t,1} | \mathcal{I}_{t,0}) d\mathbf{x}_{t,1} dy_{t,1}$$

- ▶ $p(y_{t,1} | \mathbf{x}_1, \mathcal{I}_{t,0})$: predictive distribution of the \mathcal{GP} .
- ▶ $p(\mathbf{x}_1 | \mathcal{I}_{t,0}) = \delta(\mathbf{x}_{t,1} - \arg \max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x}; \mathcal{I}_{t,0}))$.

Optimal greedy batch design

Sequential policy: Maximize:

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

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} .
- ▶ $p(\mathbf{x}_{t,j} | \mathcal{I}_{t,j-1}) = \delta(\mathbf{x}_{t,j} - \arg \max_{\mathbf{x} \in \mathcal{X}} \alpha(\mathbf{x}; \mathcal{I}_{t,j-1}))$.

Available approaches

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

- ▶ Exploratory approaches, reduction in system uncertainty.
- ▶ Generate 'fake' observations of f using $p(y_{t,j}|\mathbf{x}_j, \mathcal{I}_{t,j-1})$.
- ▶ Simultaneously optimize elements on the batch using the joint distribution of $y_{t_1}, \dots, y_{t,nb}$.

Bottleneck

All these 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: $O(n^3)$

How to design batches reducing this cost? Local penalization

Goal: eliminate the marginalization step

“To develop an heuristic approximating the ‘optimal batch design strategy’ at lower computational cost, while incorporating information about global properties of f from the \mathcal{GP} model into the batch design”

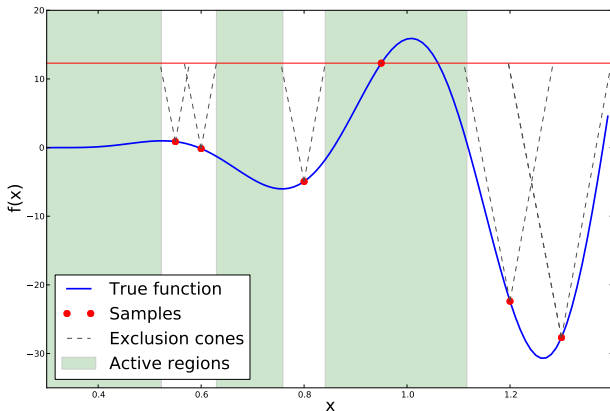
Lipschitz continuity:

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

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 $\sigma^2(r_{x_j}) = \frac{\sigma^2(\mathbf{x}_j)}{L^2}$.

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

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

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 from \mathbf{x}_j and decreases otherwise.

Idea to collect the batches

Without using explicitly the model.

Optimal batch: maximization-marginalization

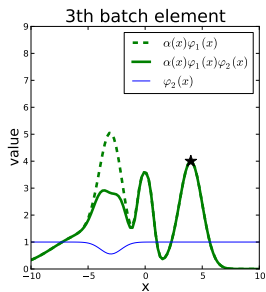
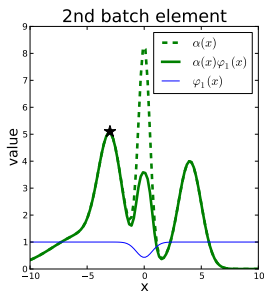
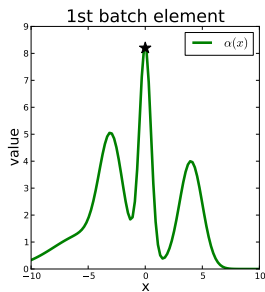
$$\int \alpha(\mathbf{x}; \mathcal{I}_{t,k-1}) \prod_{j=1}^{k-1} p(y_{t,j} | \mathbf{x}_{t,j}, \mathcal{I}_{t,j-1}) p(\mathbf{x}_{t,j} | \mathcal{I}_{t,j-1}) d\mathbf{x}_{t,j} dy_{t,j}$$

Proposal: maximization-penalization.

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

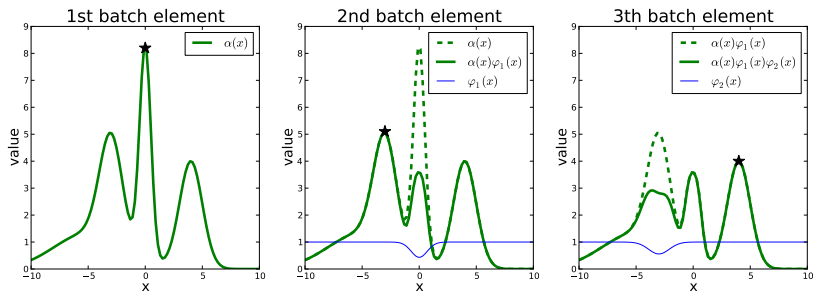
Local penalization strategy

[González, Dai, Hennig, Lawrence, 2016]



Local penalization strategy

[González, Dai, Hennig, Lawrence, 2016]

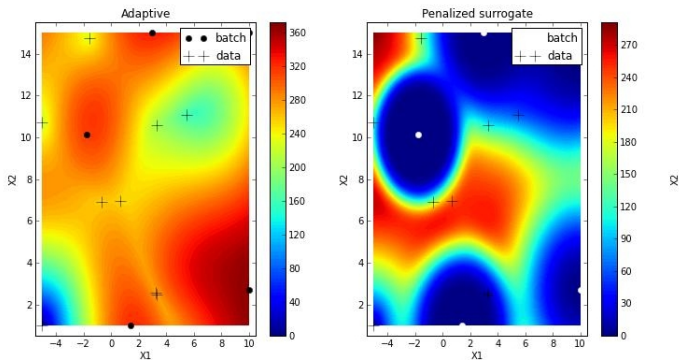


The maximization-penalization strategy selects $\mathbf{x}_{t,k}$ as

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

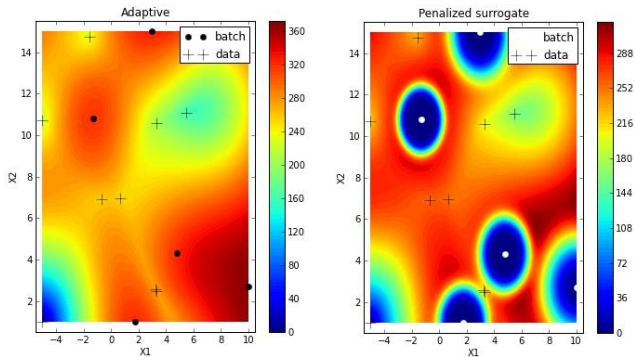
g is a transformation of $\alpha(\mathbf{x}; \mathcal{I}_{t,0})$ to make it always positive.

Example for $L = 50$



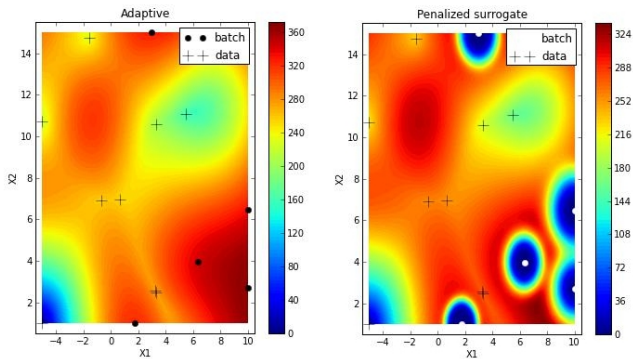
L controls the exploration-exploitation balance within the batch.

Example for $L = 100$



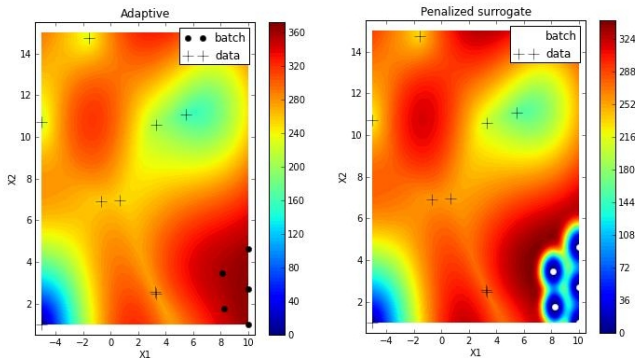
L controls the exploration-exploitation balance within the batch.

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.

Finding an unique Lipschitz constant

Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be a L -Lipschitz continuous function defined on a compact subset $\mathcal{X} \subseteq \mathbb{R}^D$. Then

$$L_p = \max_{\mathbf{x} \in \mathcal{X}} \|\nabla f(\mathbf{x})\|_p,$$

is a valid Lipschitz constant.

The gradient of f at \mathbf{x}^* is distributed as a multivariate Gaussian

$$\nabla f(\mathbf{x}^*) | \mathbf{X}, \mathbf{y}, \mathbf{x}^* \sim \mathcal{N}(\mu_{\nabla}(\mathbf{x}^*), \Sigma_{\nabla}^2(\mathbf{x}^*))$$

We choose:

$$\hat{L} = \max_{\mathbf{x}} \|\mu_{\nabla}(\mathbf{x}^*)\|$$

Experiments: Sobol function

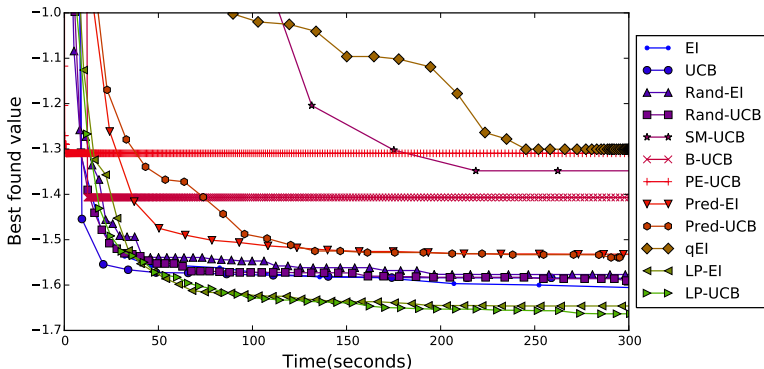
Best (average) result for some given time budget.

d	n_b	EI	UCB	Rand-EI	Rand-UCB	SM-UCB	B-UCB
2	5	0.31±0.03	0.32±0.06	0.32±0.05	0.31±0.05	1.86±1.06	0.56±0.03
	10			0.65±0.32	0.79±0.42	4.40±2.97	0.59±0.00
	20			0.67±0.31	0.75±0.32	-	0.57±0.01
5	5	8.84±3.69	11.89±9.44	9.19±5.32	10.59±5.04	137.2±113.0	6.01±0.00
	10			1.74±1.47	2.20±1.85	108.7±74.38	3.77±0.00
	20			2.18±2.30	2.76±3.06	-	2.53±0.00
10	5	559.1±1014	1463±1803	690.5±947.5	1825±2149	9e+04±7e+04	2098±0.00
	10			200.9±455.9	1149±1830	9e+04±1e+05	857.8±0.00
	20			639.4±1204	385.9±642.9	-	1656±0.00

d	n_b	PE-UCB	Pred-EI	Pred-UCB	qEI	LP-EI	LP-UCB
2	5	0.99±0.74	0.41±0.15	0.45±0.16	1.53±0.86	0.35±0.11	0.31±0.06
	10	0.66±0.29	1.16±0.70	1.26±0.81	3.82±2.09	0.66±0.48	0.69±0.51
	20	0.75±0.44	1.28±0.93	1.34±0.77	-	0.50±0.21	0.58±0.21
5	5	123.5±81.43	10.43±4.88	11.77±9.44	15.70±8.90	11.85±5.68	10.85±8.08
	10	120.8±78.56	9.58±7.85	11.66±11.48	17.69±9.04	3.88±4.15	1.88±2.46
	20	98.60±82.60	8.58±8.13	10.86±10.89	-	6.53±4.12	1.44±1.93
10	5	2e+05±2e+05	793.0±1226	1412±3032	-	1881±1176	1194±1428
	10	6e+04±8e+04	442.6±717.9	1725±3205	-	1042±1562	100.4±338.7
	20	5e+04±4e+04	1091±1724	2231±3110	-	1249±1570	20.75±50.12

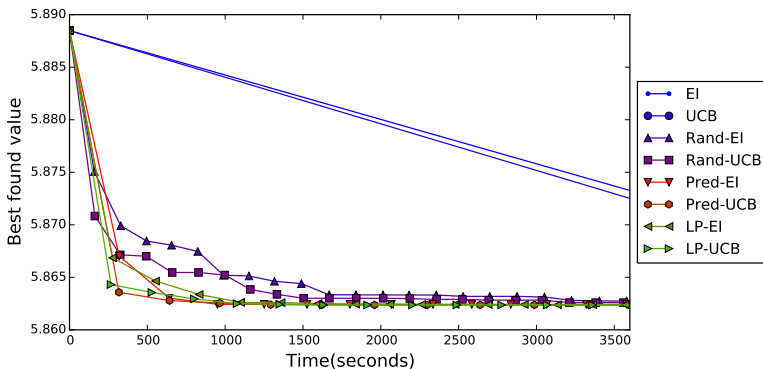
2D experiment with 'large domain'

Comparison in terms of the wall clock time



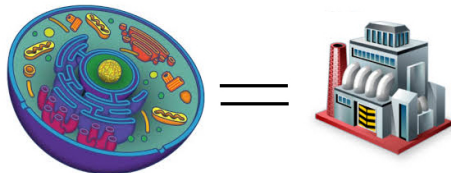
Support Vector Regression

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



Application: Synthetic gene design

[González, Lonworth, James and Lawrence, 2014, 2015]



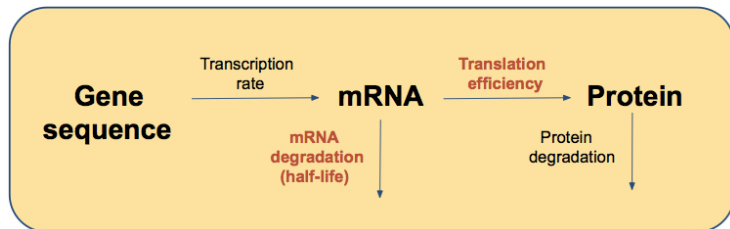
- ▶ Use mammalian cells to make protein products.
- ▶ Control the ability of the cell-factory to use synthetic DNA.

Cornerstone of modern biotechnology: Design DNA code that will best enable the cell-factory to operate most efficiently.

Synthetic gene design

Natural cells vs. cell factories

Central dogma of systems biology

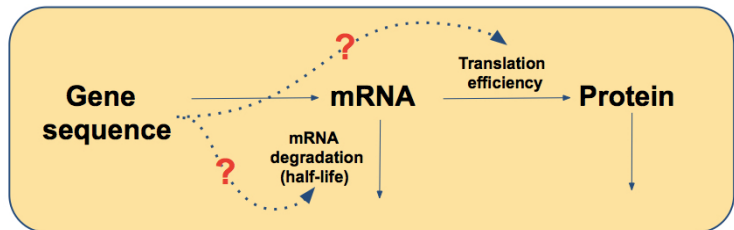


In a natural mammalian cell:

- ▶ Not all genes encode proteins of therapeutical interest.
- ▶ 'Natural' genes are not optimized to maximize protein production.

Natural cells vs. cell factories

Central dogma of systems biology



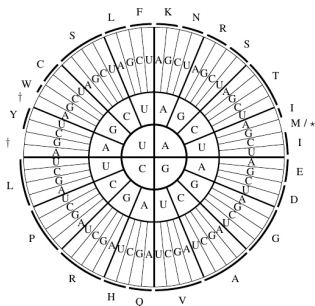
Current tools in synthetic biology allow to:

- ▶ control cell transcription...
- ▶ ...but it is unknown how to control cell translation and mRNA stability.

Develop a synthetic gene design tool to control/optimize translation

Why can we rewrite the genetic code?

- ▶ Different gene sequences may encode the same protein...
- ▶ ...but the sequence affects the synthesis efficiency.
- ▶ The codon usage is the key (codon = triplet of bases).



The genetic code is redundant:

*ATGUUG***ACA**... = *ATGUUG***ACU**...

Both genes encode the same protein.

Challenges

- ▶ Huge and structured design space: gene features extraction.
- ▶ Unknown mechanistic model of the cell behaviour: multioutput Gaussian processes.
- ▶ Expensive and time consuming experiments: Bayesian Optimization.
- ▶ Many experiments can be run in parallel.

Gene features extraction

Sequence lengths

- Coding sequence length.
- 5'UTR length, 3'UTR length.

Nucleotide frequencies and properties

- 3'UTR, 5'UTR, coding region free folding energy.
- Free folding energy in 3'UTR, 5'UTR, the end of 5'UTR, the first 40 nucleotides of the coding region.
- Best local, normalized and randomized secondary structure for 3'UTR and 5'UTR with window size 40, 60.

Codons and aminoacid frequencies and properties

- Codons usage in coding region and in the first 30-50 codons in the coding region.
- Amino-acid frequencies.
- Relative synonymous codon usage.

Specific motifs frequency

- AT content.
- GC ratio.
- GC content in coding region, whole gene, starting of coding region, codon positions 1, 2 and 3.
- AUG (atg) frequency in 5'UTR (uORF frequency in 5'UTR).
- Number of A's at the beginning of the 5'UTR.
- Number of Polyadenylation sites/3'UTR (AUAAA motif).
- Motif RYMRVAUGGC.

Codon bias indices

- Codon adaptation index in coding region.
- Codon adaptation index in first 30-50 codons.
- Codon bias index (relative codon bias sequence RCBS)



Model as an emulator of the cell behavior

-Model inputs

Gene features (\mathbf{x}_i).

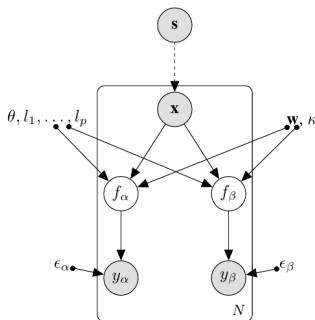
-Model outputs

Translation rates and mRNA half-life $\mathbf{f} := (f_\alpha, f_\beta)$.

-Model: Multi-output GP

$$\mathbf{f} \approx \mathcal{GP}(\mathbf{m}, \mathbf{K})$$

where $\mathbf{K} = \mathbf{B} \otimes \mathbf{K}_{in}$ with ARD.



Bayesian Optimization principles for gene design

[González, Lonworth, James and Lawrence, 2014]

do:

1. Build a GP model as an **emulator of the cell behavior**.
2. Obtain a set* of **gene design rules** (features optimization).
3. Design one/many **new gene/s** coherent with the design rules.
4. **Test genes in the lab** (get new data).

until the gene is optimized (or the budget is over...).

*Many sets or rules can be obtained for parallel designs.

Designing new genes coherent with the optimal design rules

Simulating-matching approach:

1. **Simulate** genes 'coherent' with the target.
2. **Extract features**.
3. **Rank synthetic genes** according to their similarity with the 'optimal' design rules.

Ranking criterion: $eval(\mathbf{s}|\mathbf{x}^\star) = \sum_{j=1}^p w_j |\mathbf{x}_j - \mathbf{x}_j^\star|$

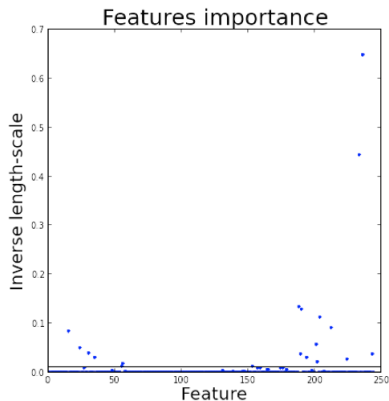
- ▶ \mathbf{x}^\star : optimal gene design rules.
- ▶ \mathbf{s}, \mathbf{x}_j generated '**synonyms sequence**' and its features.
- ▶ w_j : weights of the p features.

Experiments

- ▶ Dataset in Schwanhauser et al. (2011) for 3810 genes rates. Sequences extracted from <http://wet-labpic/www.ensembl.org>.
- ▶ 250 features involving 5'UTR, 3'UTR and coding region.
- ▶ Gaussian process with ARD and coregionalized outputs.
- ▶ Synthetic genes to produce siaP.
- ▶ 10,000 random 'synonyms sequences' generated from each gene.
- ▶ GPy and GPyOpt (<https://github.com/SheffieldML/>).

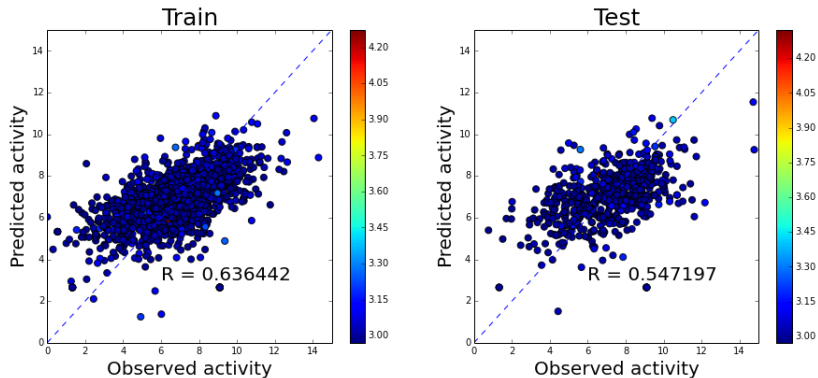
We can evaluate the gene features relevance

Feature	Score
5' UTR free fold energy	0.644
5' UTR length	0.443
number of stop codons	0.134
Cysteine	0.128
Serine	0.112
Length	0.090
Codon ATT	0.084
Proline	0.057
Codon CGA	0.050
Codon CTG	0.038
Alanine	0.037
Free folding energy (size window 60) in 5'UTR	0.036
Glycine	0.029
Codon GAT	0.029
3' UTR length	0.027



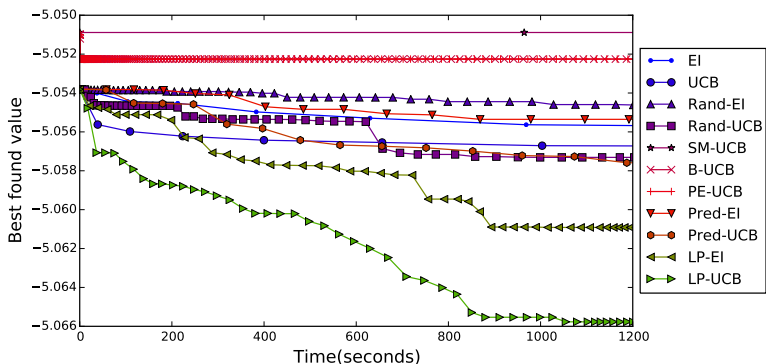
A few number of features are relevant

The model is able to predict translation rates

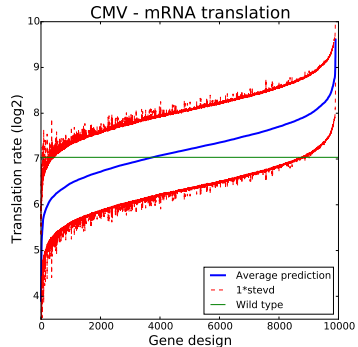
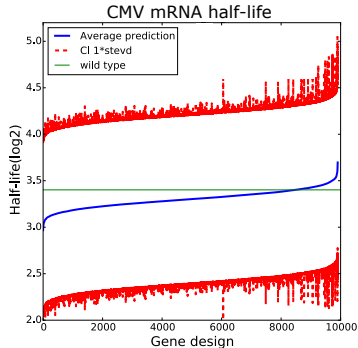


Maximizing gene translation

In-silico comparison of different batch methods for parallel gene design.



We can use the model to control translation



Ranking of 10,000 recombinant simulate sequences for the average translation rates and mRNA half-life.

Wrapping up

- ▶ BO is fantastic tool for global parameter optimization in ML and experimental design.
- ▶ To design a batch to parallelize BO requires modelling the interaction between the elements in the batch: this can be done without updating the model explicitly after each batch element is collected.
- ▶ BO is a great tool to design synthetic genes.
- ▶ Software available! Come to the tutorial today at 2pm and use GPyOpt!

Many thanks to

- ▶ Michael Osborne, University of Oxford.
- ▶ Neil Lawrence, University of Sheffield.
- ▶ Zhenwen Dai, University of Sheffield.
- ▶ Philipp Hennig, Max Planck institute.
- ▶ Andreas Damianou, University of Sheffield.
- ▶ David James, Joseph Longworth and others at CBE, University of Sheffield.