BayesOpt: Extensions and applications

Javier González

Masterclass, 7-February, 2107 @Lancaster University

Agenda of the day

- **9:00-11:00, Introduction to Bayesian Optimization**:
  - What is BayesOpt and why it works?
  - Relevant things to know.

- **11:30-13:00, Connections, extensions and applications**:
  - Extensions to multi-task problems, constrained domains, early-stopping, high dimensions.
  - Connections to Armed bandits and ABC.
  - An applications in genetics.

- **14:00-16:00, GPyOpt LAB!**: Bring your own problem!

- **16:30-15:30, Hot topics current challenges**:
  - Parallelization.
  - Non-myopic methods
  - Interactive Bayesian Optimization.

Section II: Connections, extensions and applications

- Extensions to multi-task problems, constrained domains, early-stopping, high dimensions.
- Connections to Armed bandits and ABC.
- An applications in genetics.

Multi-task Bayesian Optimization

[Wersky et al., 2013]

Two types of problems:

1. Multiple, and conflicting objectives: design an engine more powerful but more efficient.

2. The objective is very expensive, but we have access to another cheaper and correlated one.

Multi-task Bayesian Optimization

[Wersky et al., 2013]

- We want to optimise an objective that it is very expensive to evaluate but we have access to another function, correlated with objective, that is cheaper to evaluate.

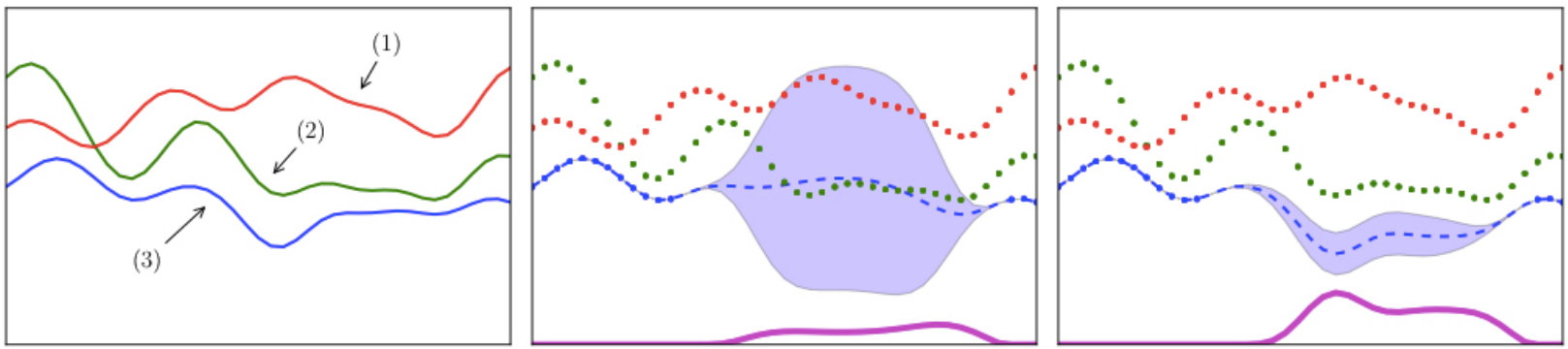
- The idea is to use the correlation among the function to improve the optimization.

Multi-output Gaussian process

$$\tilde{k}(x, x') = \mathbf{B} \otimes k(x, x')$$

Multi-task Bayesian Optimization

[Wersky et al., 2013]



(a) Multi-task GP sample functions (b) Independent GP predictions (c) Multi-task GP predictions

- Correlation among tasks reduces global uncertainty.
- The choice (acquisition) changes.

Multi-task Bayesian Optimization

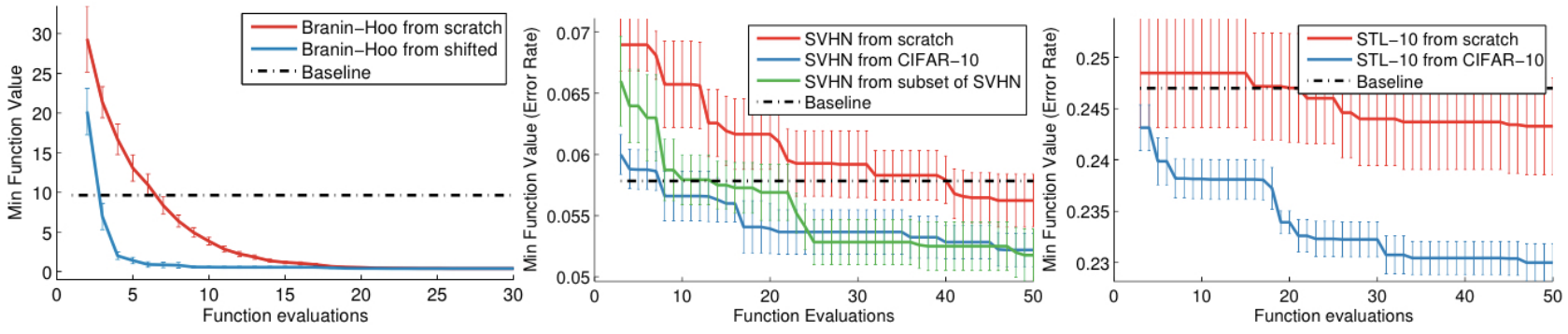
[Wersky et al., 2013]

- In other cases we want to optimize several tasks at the same time.
- We need to use a combination of them (the mean, for instance) or have a look to the Pareto frontiers of the problem.

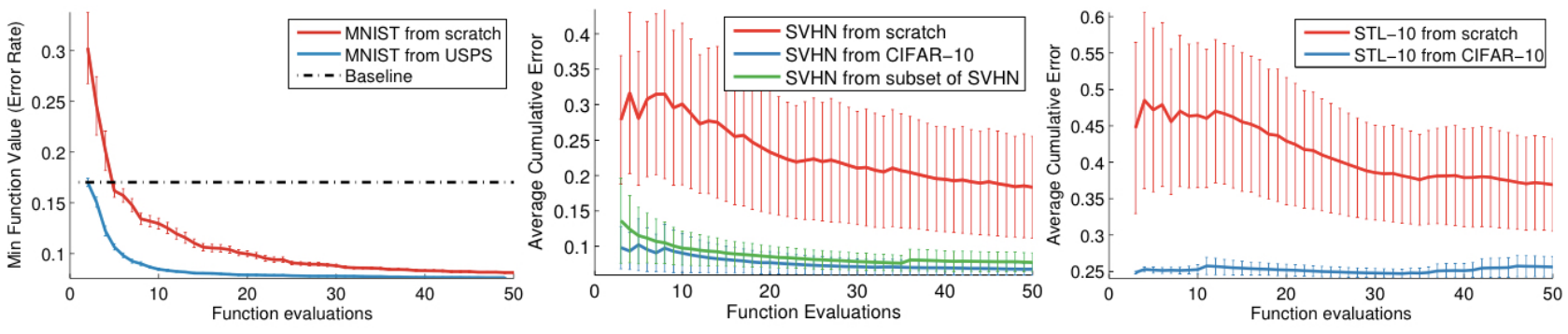
Averaged expected improvement.

Multi-task Bayesian Optimization

[Wersky et al., 2013]



(a) Shifted Branin-Hoo (b) CNN on SVHN (c) CNN on STL-10



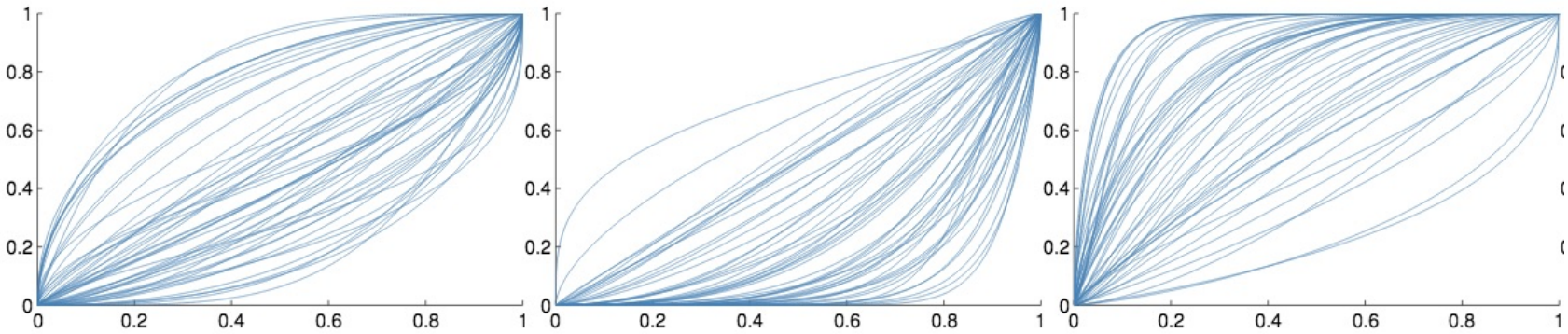
(d) LR on MNIST (e) SVHN ACE (f) STL-10 ACE

Non-stationary Bayesian Optimization

[Snoek et al., 2014]

The beta distributions allows for a rich family of transformations.

$$
\begin{aligned}
w_d(\mathbf{x}_d) &= \text{BetaCDF}(\mathbf{x}_d; \alpha_d, \beta_d)\,, \\
&= \int_0^{\mathbf{x}_d} \frac{u^{\alpha_d - 1}(1-u)^{\beta_d - 1}}{B(\alpha_d, \beta_d)} \mathrm{d}u\,,
\end{aligned}
$$

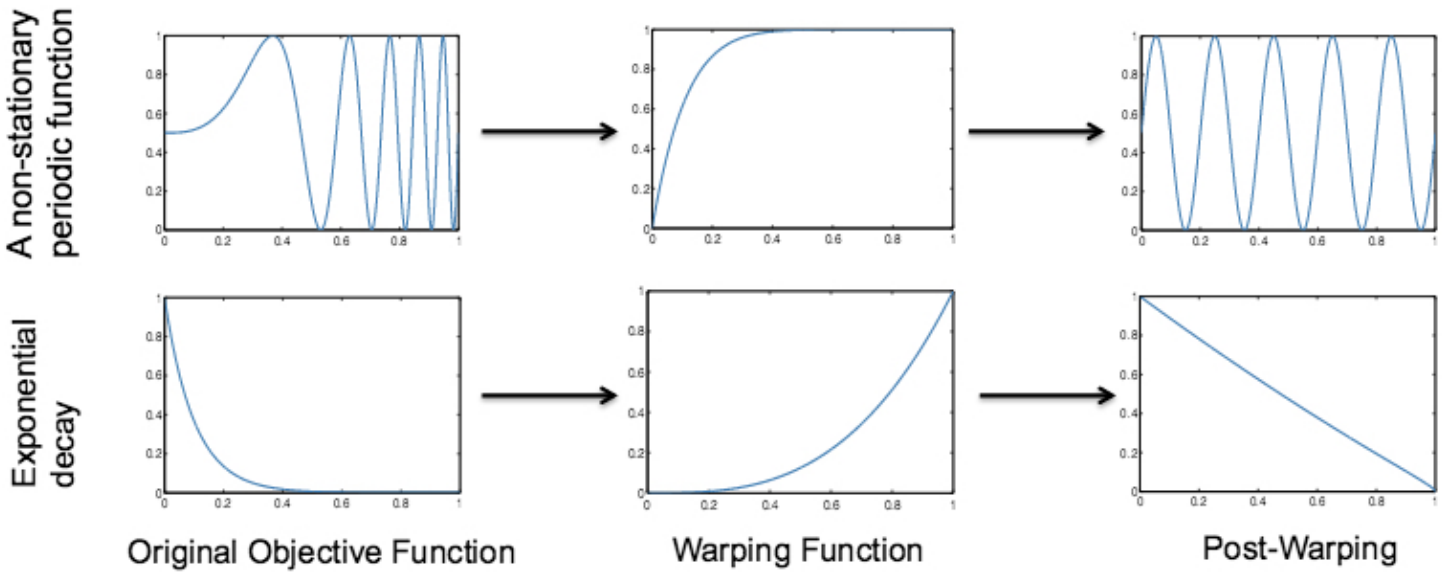


(a) Linear (b) Exponential (c) Logarithmic

Non-stationary Bayesian Optimization

[Snoek et al., 2014]

Idea: transform the function to make it stationary.

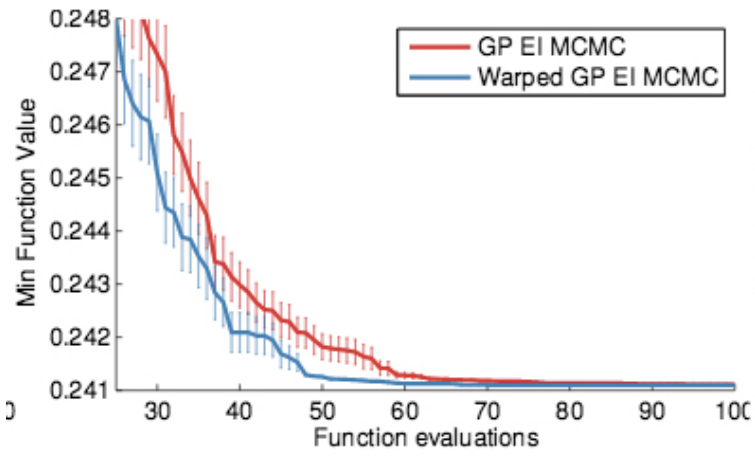


Non-stationary Bayesian Optimization

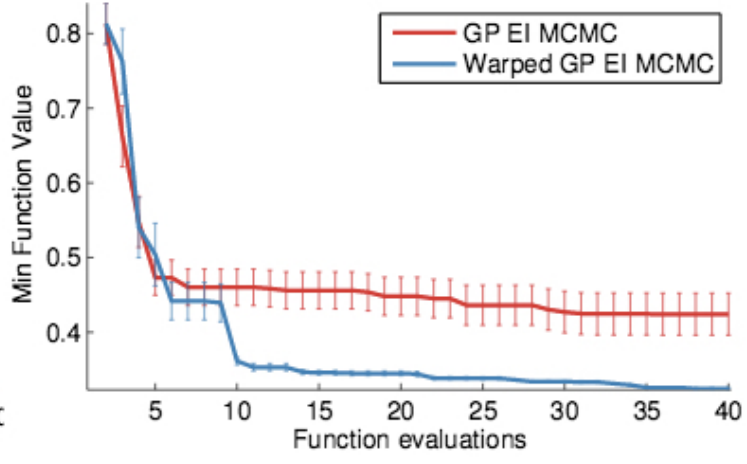
[Snoek et al., 2014]

Results improve in many experiments by warping the inputs.

Extensions to multi-task warping.



(c) Structured SVM



(d) Cifar 10 Subset

Inequality Constraints

[Gardner et al., 2014]

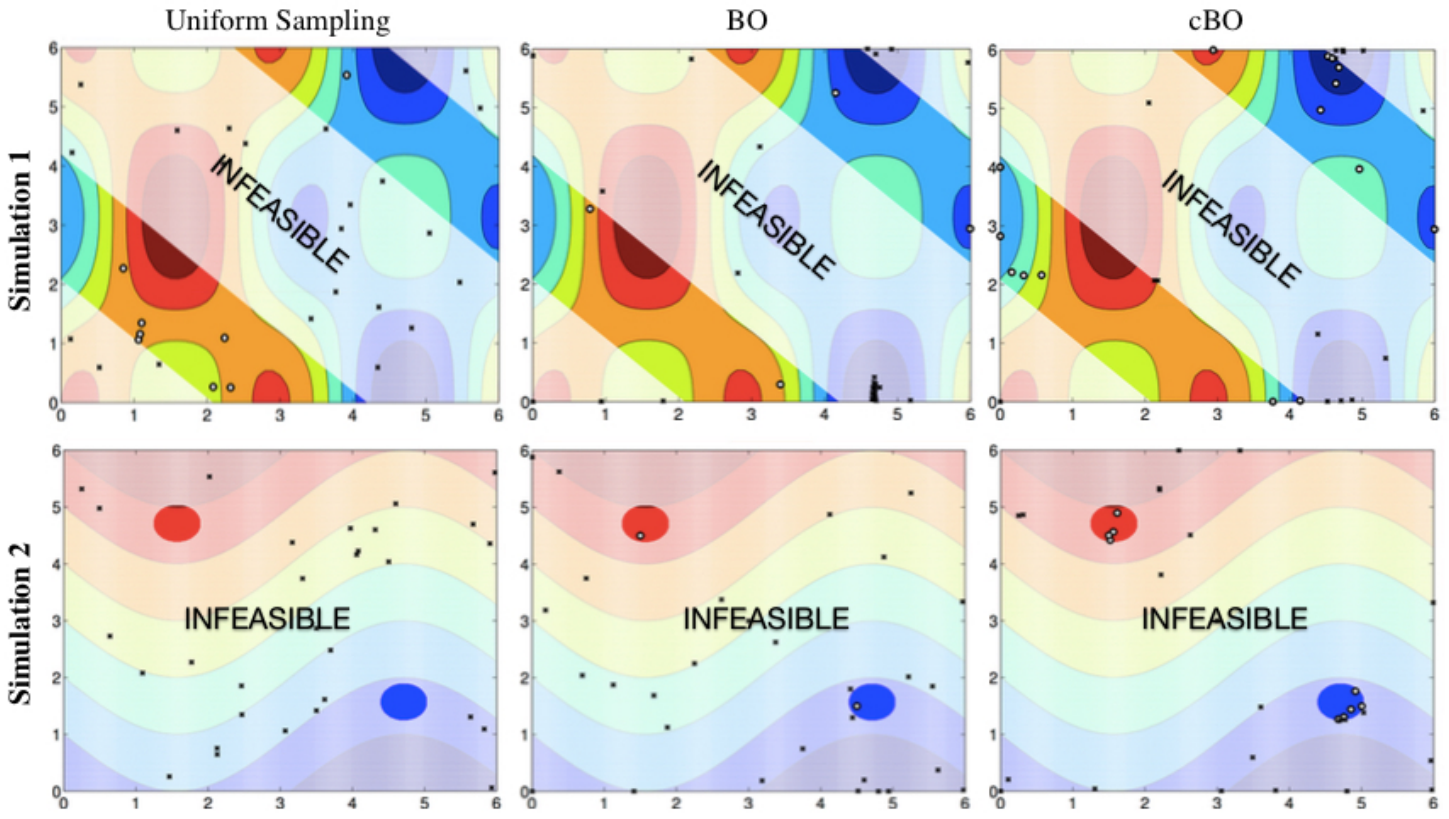
An option is to penalize the EI with an indicator function that vanishes the acquisition out the domain of interest.

$$I_C(\hat{\mathbf{x}}) = \Delta(\hat{\mathbf{x}}) \max\left\{0, \ell(\mathbf{x}^+) - \ell(\hat{\mathbf{x}})\right\} = \Delta(\hat{\mathbf{x}}) I(\hat{\mathbf{x}})$$

Inequality Constraints

[Gardner et al., 2014]

Much more efficient than standard approaches.



High-dimensional BO: REMBO

[Wang et al., 2013]

Bayesian Optimization in a Billion Dimensions via Random Embeddings

---

Ziyu Wang ZIYUW@CS.UBC.CA
University of British Columbia

Masrour Zoghi M.ZOGHI@UVA.NL
University of Amsterdam

David Matheson DAVIDM@CS.UBC.CA
University of British Columbia

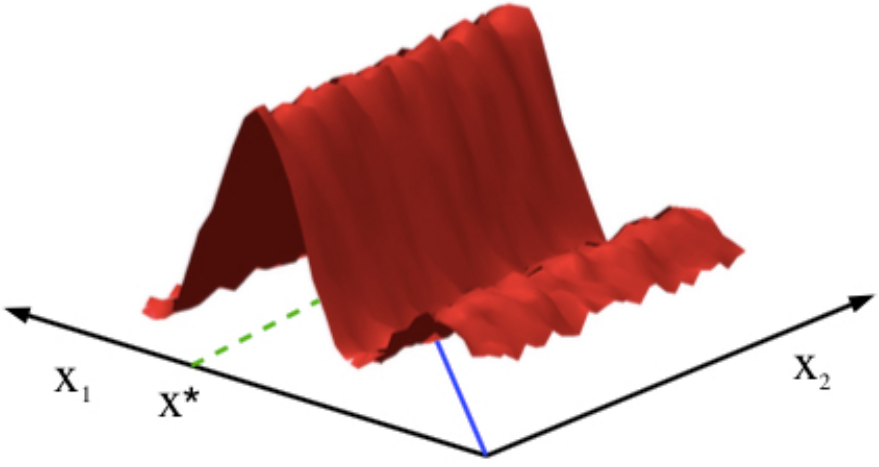
Frank Hutter HUTTER@CS.UBC.CA
University of British Columbia

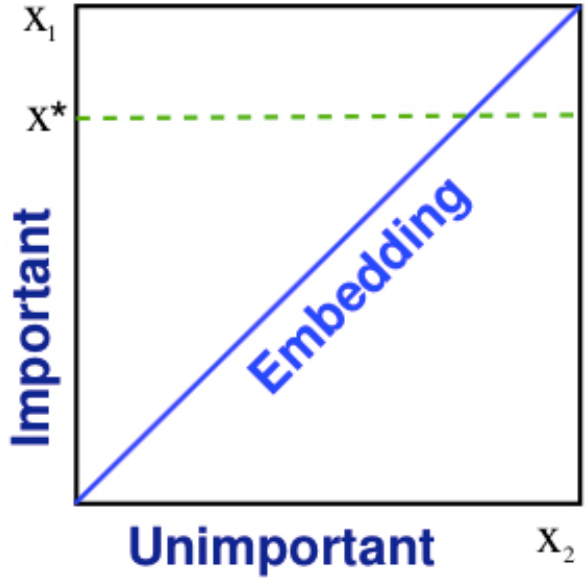
Nando de Freitas NANDO@CS.UBC.CA
University of British Columbia

High-dimensional BO: REMBO

[Wang et al., 2013]

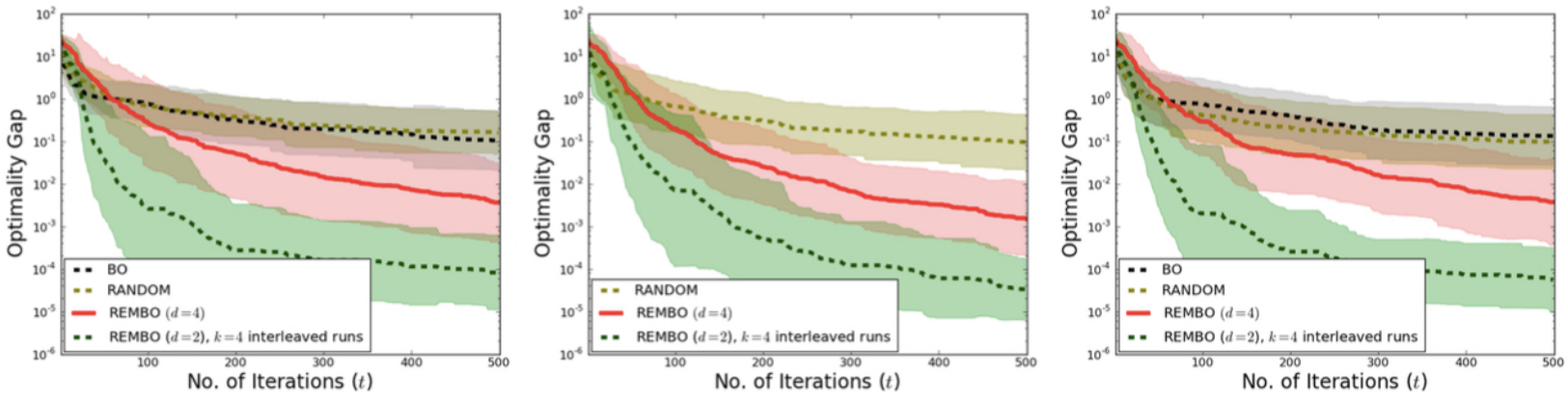
A function $f : \mathcal{X} \to \Re$ is called to have effective dimensionality d with $d \leq D$ if there exist a linear subspace $\mathcal{T}$ of dimension d such that for all $x_\perp \subset \mathcal{T}$ and $x_\top \subset \mathrm{T}^\top \subset \mathcal{T}$ we have $f(x_\perp) = f(x_\perp + x_\top)$ where $\mathcal{T}^\top$ is the orthogonal complement of $\mathcal{T}$.





High-dimensional BO: REMBO

[Wang et al., 2013]



- Better in cases in the which the intrinsic dimensionality of the function is low.
- Hard to implement (need to define the bounds of the optimization after the embedding).

High-dimensional BO: Additive models

Use the Sobol-Hoeffding decompostion

$$f(x) = f_0 + \sum_{i=1}^{D} f_i(x_i) + \sum_{i<j} f_{ij}(x_i, x_j) + \cdots + f_{1,\ldots,D}(x)$$

where

- $f_0 = \int_{\mathcal{X}} f(x)dx$
- $f_i(x_i) = \int_{\mathcal{X}_{-i}} f(x)dx_{-i}$ - f_0
- etc...

and assume that the effects of high order than q are null.

High-dimensional BO: Additive models

High Dimensional Bayesian Optimisation and Bandits via Additive Models

Kirthevasan Kandasamy KANDASAMY@CS.CMU.EDU
Jeff Schneider SCHNEIDE@CS.CMU.EDU
Barnabás Póczos BAPOCZOS@CS.CMU.EDU
Carnegie Mellon University, Pittsburgh, PA, USA

Abstract

Bayesian Optimisation (BO) is a technique used in optimising a D-dimensional function which is typically expensive to evaluate. While there have been many successes for BO in low dimensions, scaling it to high dimensions has been notoriously difficult. Existing literature on the topic are under very restrictive settings. In this paper, we identify two key challenges in this endeavour. We tackle these challenges by assuming an addi-

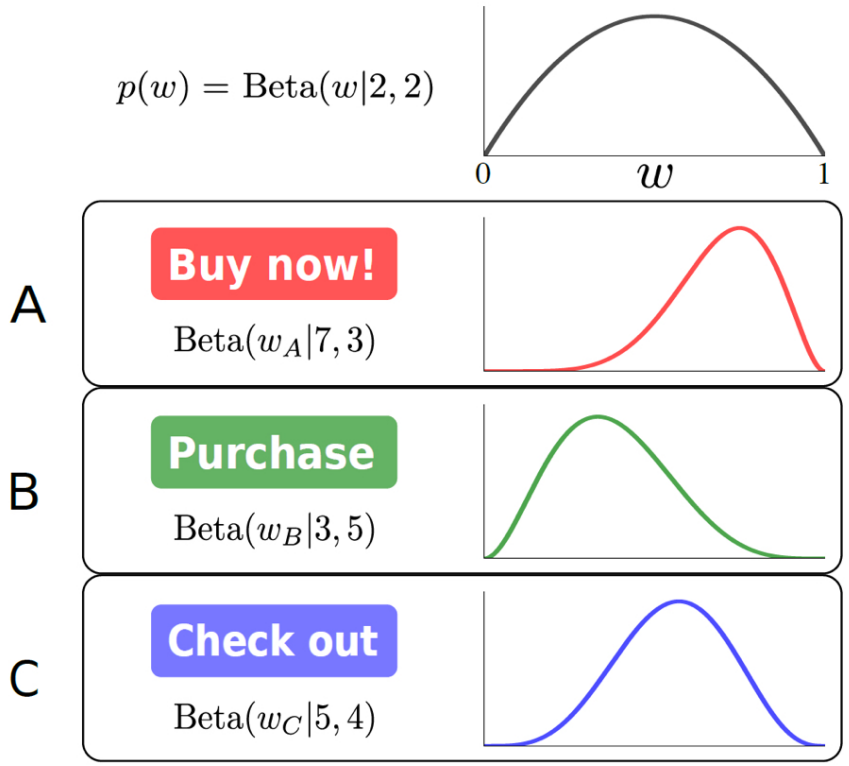
Bayesian Optimisation (Mockus & Mockus, 1991) refers to a suite of methods that tackle this problem by modeling f as a Gaussian Process (GP). In such methods the challenge is two fold. At time step t, first estimate the unknown f from the query value-pairs. Then use it to intelligently query at $\mathbf{x}_t$ where the function is likely to be high. For this, we first we use the posterior GP to construct an acquisition function φ_t which captures the value of the experiment. Then we maximise φ_t to determine $\mathbf{x}_t$.

Gaussian process bandits and Bayesian optimisation (GPB/

Armed bandits - Bayesian Optimization

Shahriari et al, [2016]

Beta-Bernoulli Bayesian optimization: Beta prior on each arm.



$$p(\mathbf{w} \mid \alpha, \beta) = \prod_{a=1}^{K} \text{Beta}(w_a \mid \alpha, \beta)$$

$$n_{a,0} = \sum_{i=1}^{n} \mathbb{I}(y_i = 0, a_i = a)$$

$$n_{a,1} = \sum_{i=1}^{n} \mathbb{I}(y_i = 1, a_i = a)$$

Armed bandits - Bayesian Optimization

Shahriari et al, [2016]

Beta posterior:

$$p(\mathbf{w} \mid \mathcal{D}) = \prod_{a=1}^{K} \text{Beta}(w_a \mid \alpha + n_{a,1}, \beta + n_{a,0})$$

Thompson sampling:

$$a_{n+1} = \arg\max_a f_{\tilde{\mathbf{w}}}(a) \quad \text{where } \tilde{\mathbf{w}} \sim p(\mathbf{w} \mid \mathcal{D}_n)$$

Armed bandits - Bayesian Optimization

Shahriari et al, [2016]

Beta-Bernoulli Bayesian optimization:

Algorithm 2 Thompson sampling for Beta-Bernoulli bandit

Require: α, β: hyperparameters of the beta prior

Initialize $n_{a,0} = n_{a,1} = i = 0$ for all a
repeat
  for $a = 1, \ldots, K$ **do**
    $\tilde{w}_a \sim \text{Beta}(\alpha + n_{a,1}, \beta + n_{a,0})$
  end for
  $a_i = \arg\max_a \tilde{w}_a$
  Observe y_i by pulling arm a_i
  if $y_i = 0$ **then**
    $n_{a_i,0} = n_{a_i,0} + 1$
  else
    $n_{a_i,1} = n_{a_i,1} + 1$
  end if
  $i = i + 1$
until stopping criterion reached

Armed bandits - Bayesian Optimization

Shahriari et al, [2016]

Linear bandits:

We introduce correlations among the arms.

$$f_{\mathbf{w}}(a) = \mathbf{x}_a^T \mathbf{w}$$

Normal-inverse Gamma prior.

$$\begin{aligned}\mathrm{NIG}(\mathbf{w}, \sigma^2 \mid \mathbf{w}_0, \mathbf{V}_0, \alpha_0, \beta_0) = \\ |2\pi\sigma^2 \mathbf{V}_0|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2\sigma^2}(\mathbf{w}-\mathbf{w}_0)^T \mathbf{V}_0^{-1}(\mathbf{w}-\mathbf{w}_0)\right\} \\ \times \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)(\sigma^2)^{\alpha_0+1}} \exp\left\{-\frac{\beta_0}{\sigma^2}\right\}.\end{aligned}$$

Armed bandits - Bayesian Optimization

Shahriari et al, [2016]

Linear bandits:

Now we can extract analytically the posterior mean and variance:

$$\mathbf{w}_n = \mathbf{V}_n(\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{X}^T\mathbf{y})$$
$$\mathbf{V}_n = (\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1}$$
$$\alpha_n = \alpha_0 + n/2$$
$$\beta_n = \beta_0 + \frac{1}{2}\left(\mathbf{w}_0^T\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{y}^T\mathbf{y} - \mathbf{w}_n^T\mathbf{V}_n^{-1}\mathbf{w}_n\right)$$

And do Thompsom sampling again:

$$a_{n+1} = \arg\max_a \mathbf{x}_a^T\tilde{\mathbf{w}} \text{ where } \tilde{\mathbf{w}} \sim p(\mathbf{w} \mid \mathcal{D}_n)$$

Armed bandits - Bayesian Optimization

Shahriari et al, [2016]

From linear bandits to Bayesian optimization:

- Replace $\mathbf{X}$ by a basis of functions Φ.
- Bayesian optimization generalizes Linear bandits as Gaussian processes generalizes Bayesian linear regresion.
- Infinitely many + linear + correlated Bandits = Bayesian optimization.

Early-stopping Bayesian optimization

Swersky et al. [2014]

Considerations:

- When looking for a good parameters set for a model, in many cases each evaluation requires of a inner loop optimization.
- Learning curves have a similar (monotonically decreasing) shape.
- Fit a meta-model to the learning curves to predict the expected performance of sets of parameters

Main benefit: allows for early-stopping

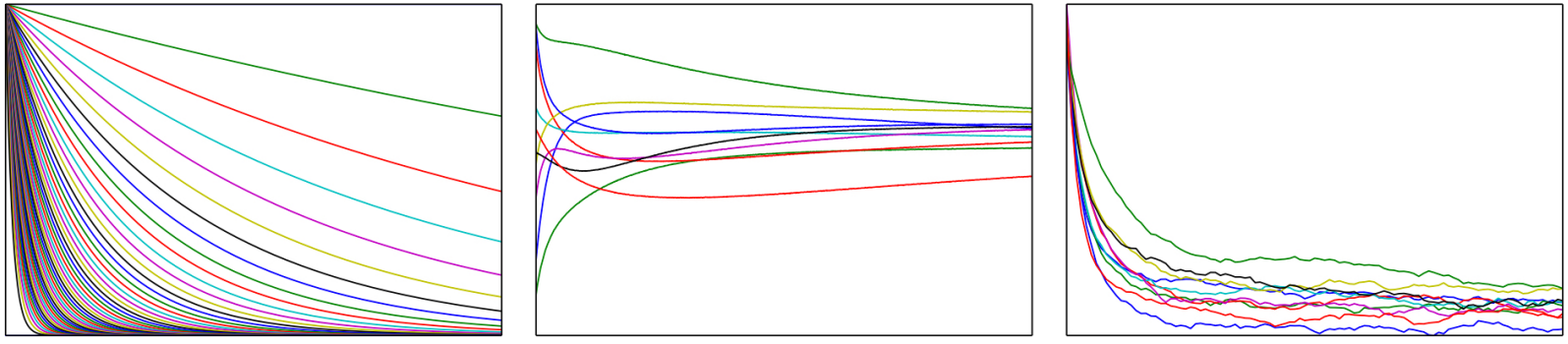
Early-stopping Bayesian optimization

Swersky et al. [2014]

Kernel for learning curves

$$k(t, t') = \int_0^\infty e^{-\lambda t} e^{-\lambda t} \varphi(d\lambda)$$

where φ is a Gamma distribution.

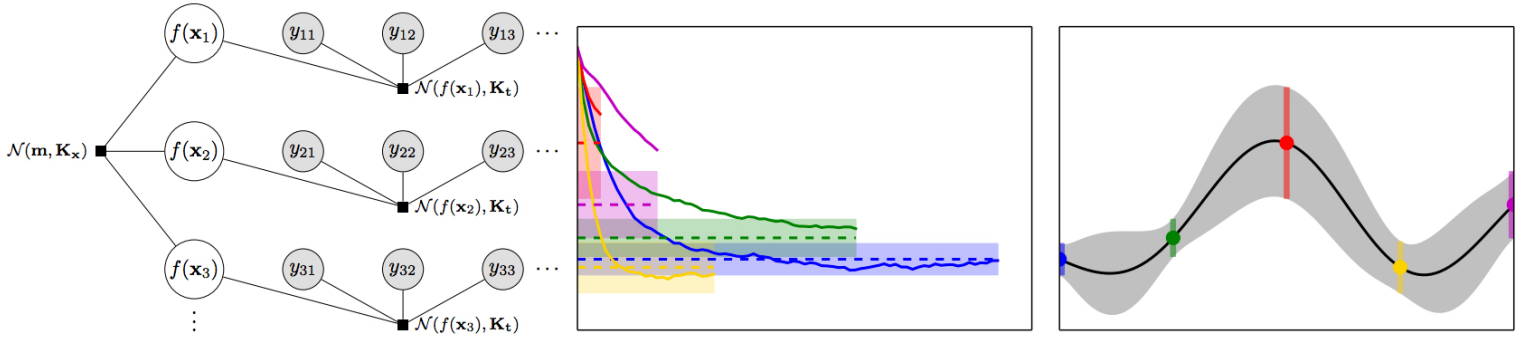


(a) Exponential Decay Basis (b) Samples (c) Training Curve Samples

Early-stopping Bayesian optimization

Swersky et al. [2014]

- Non-stationary kernel as an infinite mixture of exponentially decaying basis function.
- A hierarchical model is used to model the learning curves.
- Early-stopping is possible for bad parameter sets.

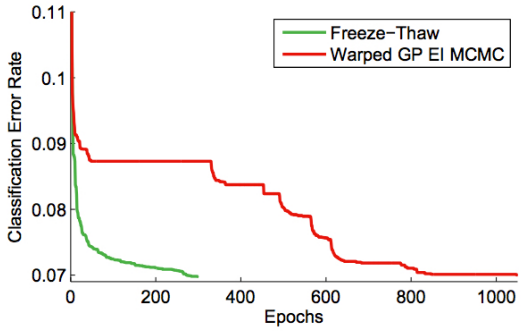


(a) Graphical Model (b) Training curve predictions (c) Asymptotic GP

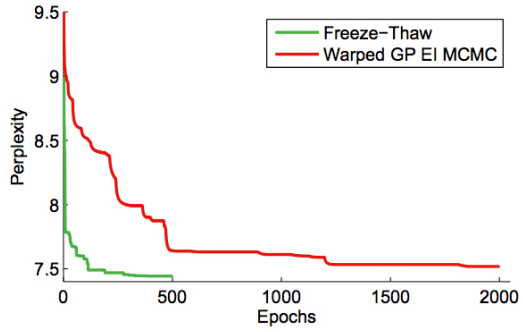
Early-stopping Bayesian optimization

Swersky et al. [2014]

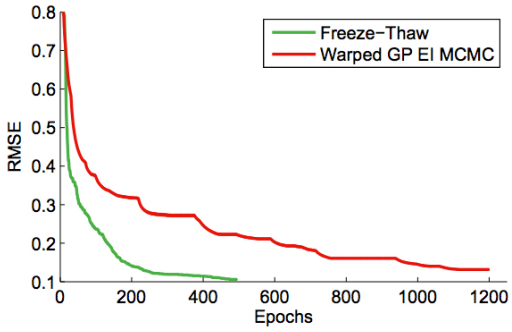
- Good results compared to standard approaches.
- What to do if exponential decay assumption does not hold?



(a) Logistic Regression



(b) Online LDA

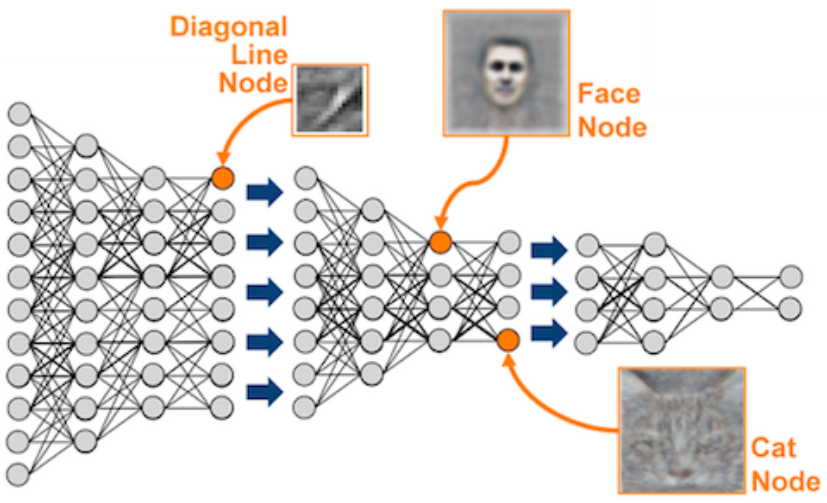


(c) PMF

Conditional dependencies

Swersky et al. [2014]

- Often, we search over structures with differing numbers of parameters: find the best neural network architecture
- The input space has a conditional dependency structure.
- Input space $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_d$. The value of $x_j \in \mathcal{X}_j$ depends on the value of $x_i \in \mathcal{X}_i$.



Conditional dependencies

Swersky et al. [2014]

Raiders of the Lost Architecture: Kernels for Bayesian Optimization in Conditional Parameter Spaces

Kevin Swersky
University of Toronto
kswersky@cs.toronto.edu

David Duvenaud
University of Cambridge
dkd23@cam.ac.uk

Jasper Snoek
Harvard University
jsnoek@seas.harvard.edu

Frank Hutter
Freiburg University
fh@informatik.uni-freiburg.de

Michael A. Osborne
University of Oxford
mosb@robots.ox.ac.uk

Abstract

In practical Bayesian optimization, we must often search over structures with differing numbers of parameters. For instance, we may wish to search over neural network architectures with an unknown number of layers. To relate performance data gathered for different architectures, we define a new kernel for conditional parameter spaces that explicitly includes information about which parameters are relevant in a given structure. We show that this kernel improves model quality and Bayesian optimization results over several simpler baseline kernels.

Robotics Video

Approximate Bayesian Computation - BayesOpt

Gutmann et al. [2015]

Bayesian inference:

$$p(\theta|y) \propto L(\theta|theta)p(\theta)$$

Focus on cases where:

- The likelihood function $L(\theta|theta)$ is too costly to compute.
- It is still possible to simulate from the model.

Approximate Bayesian Computation - BayesOpt

Gutmann et al. [2015]

ABC idea: Identify the values of θ for which simulated data resemble the observed data y_0

1. Sample θ from the prior $p(\theta)$.
2. Sample $y|\theta$ from the model.
3. Compute some distance $d(y, y_0)$ between the observed and simulated data (using sufficient statistics).
4. Retain θ if $d(y, y_0) \leq \epsilon$

Approximate Bayesian Computation - BayesOpt

Gutmann et al. [2015]

- Produce samples from the approximate posterior $p(\theta|y)$.
- Small ϵ: accurate samples but very inefficient (a lot of rejection).
- Small ϵ: less rejection but inaccurate samples.

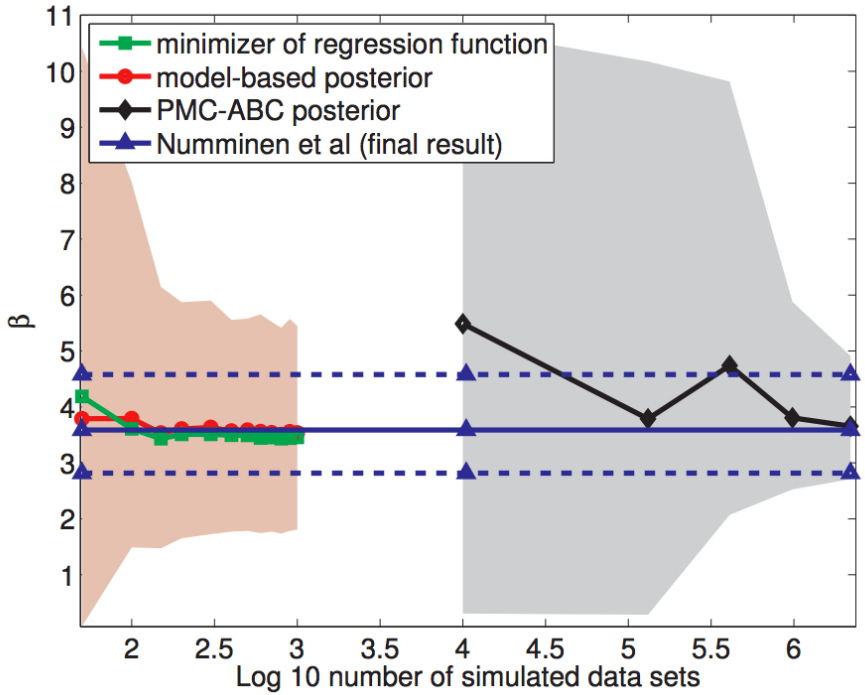
Idea: Model the discrepancy $d(y, y_0)$ with a (log) Gaussian process and use Bayesian optimization to find regions of the parameters space it is small.

$$\text{Meta-model for } (\theta_i, d_i) \text{ where } d_i = d(y_\theta^{(i)}, y_0)$$

Approximate Bayesian Computation - BayesOpt

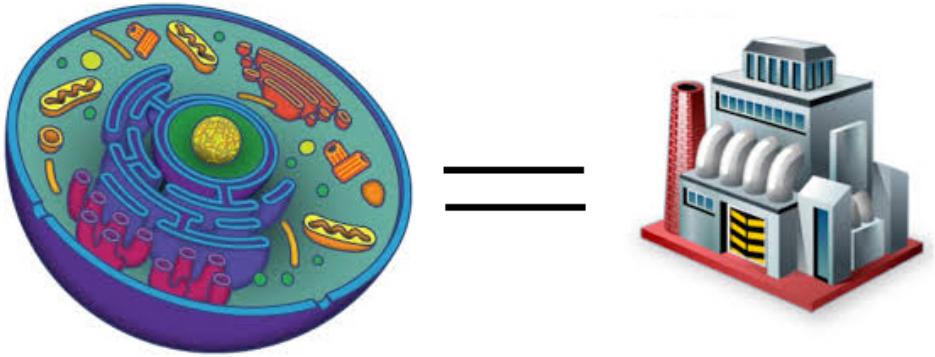
Gutmann et al. [2015]

- BayesOpt applied to minimize the discrepancy.
- Stochastic acquisition to encourage diversity in the points (GP-UCB + jitter term).



ABC-BO vs. Monte Carlo (PMC) ABC approach: Roughly equal results using 1000 times fewer simulations.

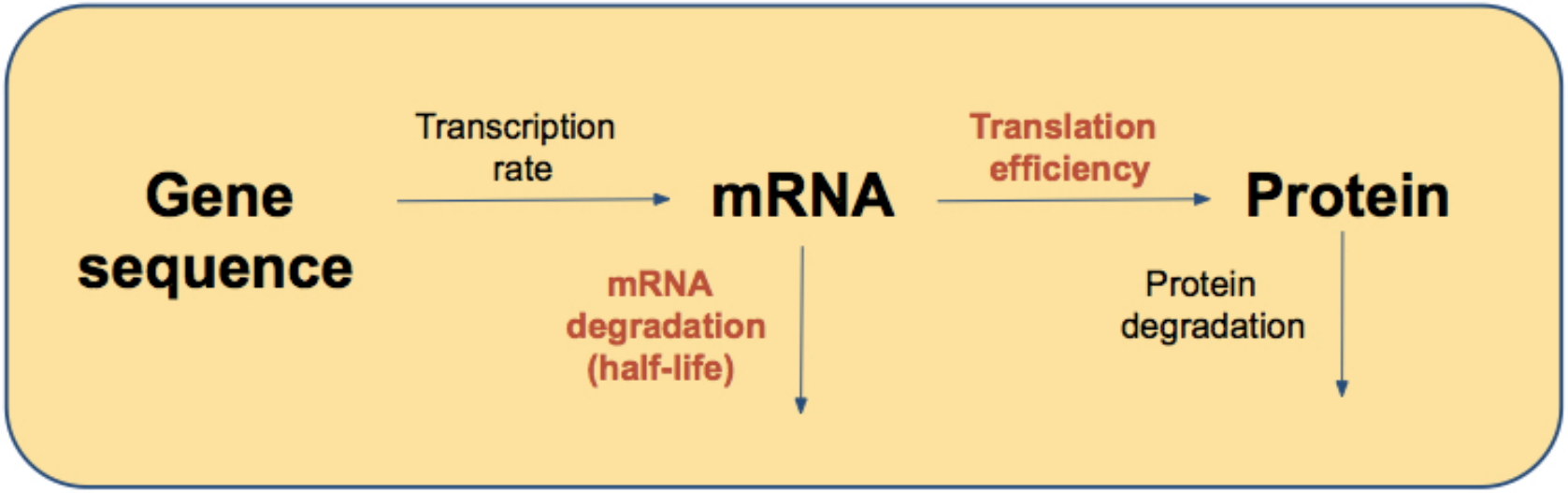
Synthetic gene design with Bayesian optimization



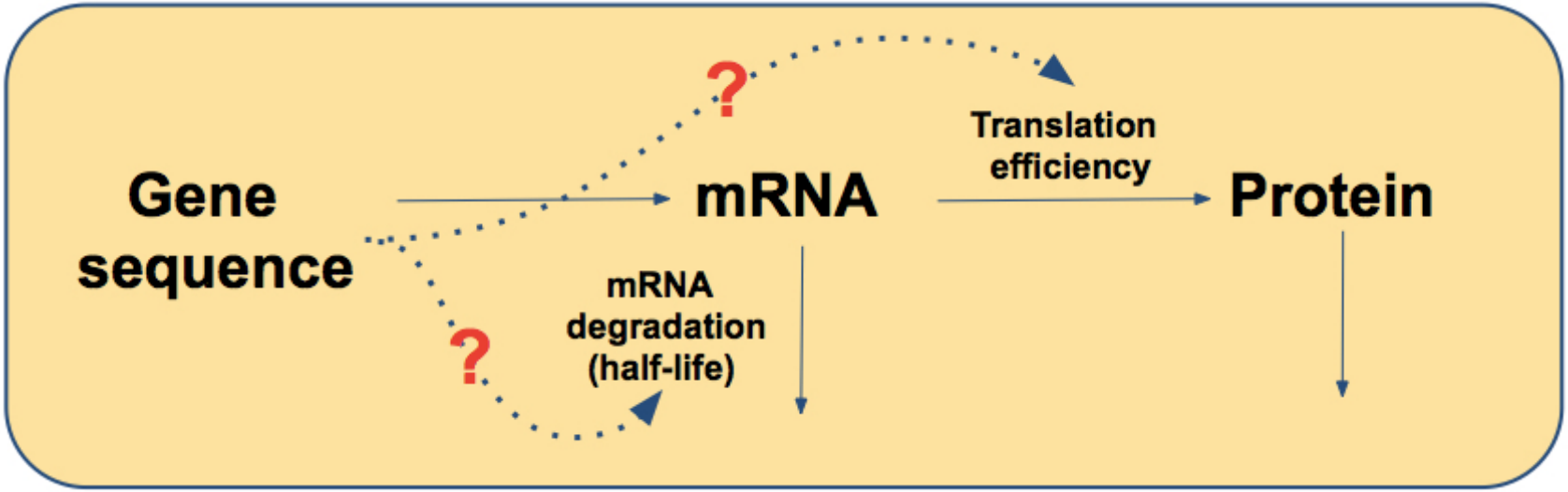
- Use mammalian cells to make protein products.
- Control the ability of the cell-factory to use synthetic DNA.

Optimize genes (ATTGGTUGA...) to best enable the cell-factory to operate most efficiently [González et al. 2014].

Central dogma of molecular biology



Central dogma of molecular biology



Big question

Remark: 'Natural' gene sequences are not necessarily optimized to maximize protein production.

ATGCTGCAGATGTGGGGGTTTGTTCTCTATCTCTTCCTGAC
TTTGTTCTCTATCTCTTCCTGACTTTGTTCTCTATCTCTTC...

Considerations

- Different gene sequences $\rightarrow$ same protein.
- The sequence affects the synthesis efficiency.

Which is the most efficient sequence to produce a protein?

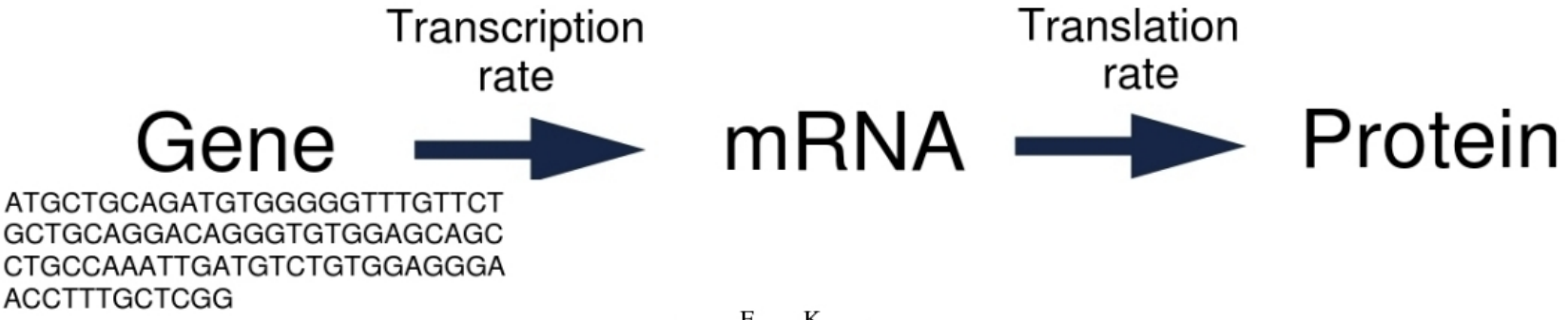
Redundancy of the genetic code

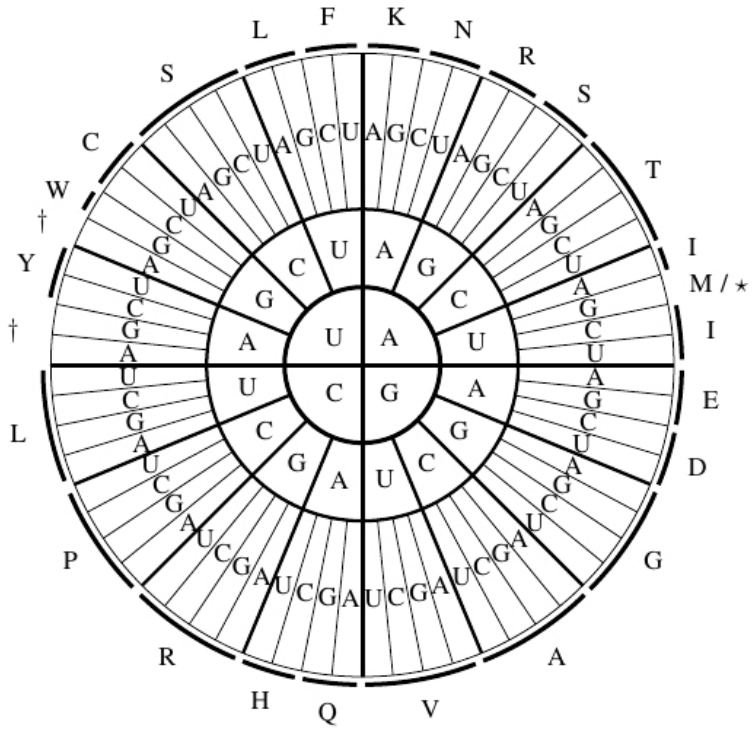
- Codon: Three consecutive bases: AAT, ACG, etc.
- Protein: sequence of amino acids.
- Different codons may encode the same aminoacid.
- ACA=ACU encodes for Threonine.

$$ATUUUGACA = ATUUUGACU$$

synonyms sequences $\rightarrow$ same protein but different efficiency

Redundancy of the genetic code





How to design a synthetic gene?

A good model is crucial: Gene sequence features → protein production efficiency.

Bayesian Optimization principles for gene design

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...).

Model as an emulator of the cell behavior

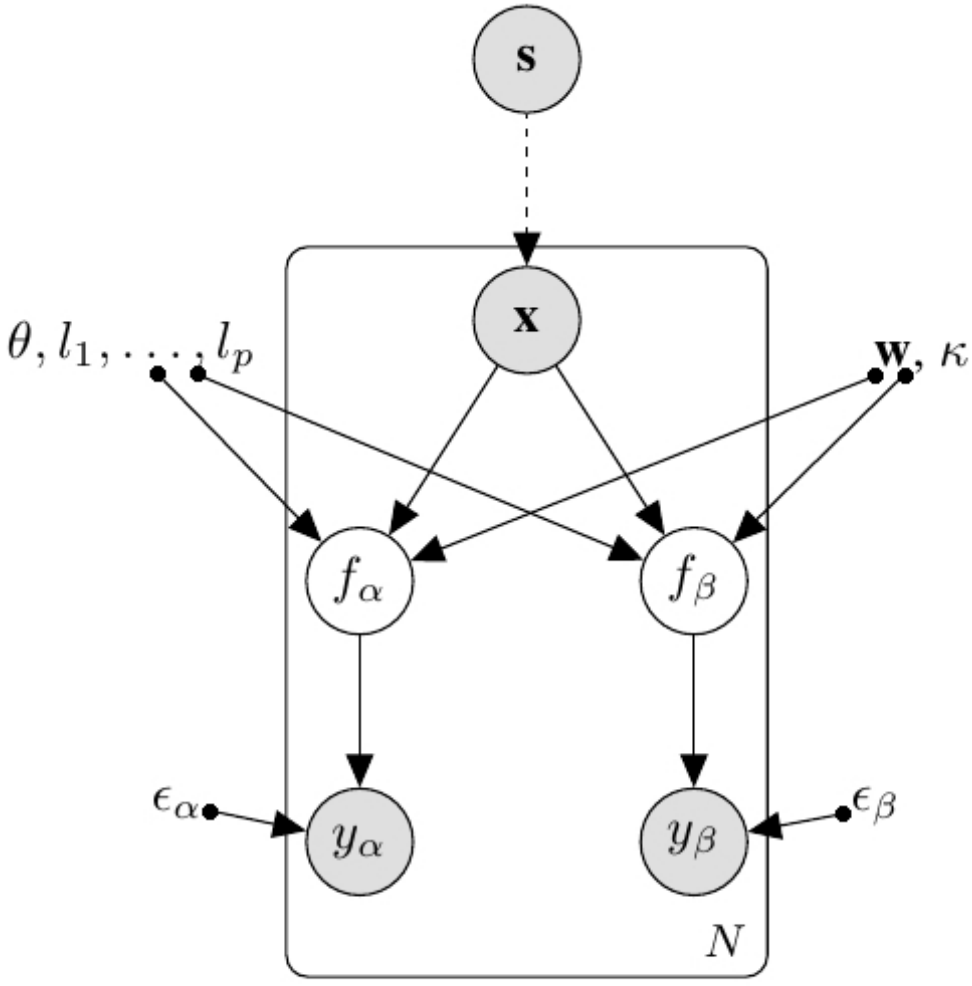
Model inputs
Features ($\mathbf{x}_i$) extracted gene sequences ($\mathbf{s}_i$): codon frequency, cai, gene length, folding energy, etc.

Model outputs
Transcription and translation rates $\mathbf{f} := (f_\alpha, f_\beta)$.

Model type
Multi-output Gaussian process $\mathbf{f} \approx \mathcal{GP}(\mathbf{m}, \mathbf{K})$ where $\mathbf{K}$ is a corregionalization covariance for the two-output model (+ SE with ARD).

The correlation in the outputs help!

Model as an emulator of the cell behavior



Obtaining optimal gene design rules

Maximize the averaged EI [Swersky et al. 2013]

$$\alpha(\mathbf{x}) = \bar{\sigma}(\mathbf{x})(-u\Phi(-u) + \phi(u))$$

where $u = (y_{max} - \bar{m}(\mathbf{x}))/\bar{\sigma}(x)$ and

$$\bar{m}(\mathbf{x}) = \frac{1}{2}\sum_{l=\alpha,\beta} \mathbf{f}_*(\mathbf{x}),\ \bar{\sigma}^2(\mathbf{x}) = \frac{1}{2^2}\sum_{l,l'=\alpha,\beta} (\mathbf{K}_*(\mathbf{x},\mathbf{x}))_{l,l'}.$$

A batch method is used when several experiments can be run in parallel

Designing new genes

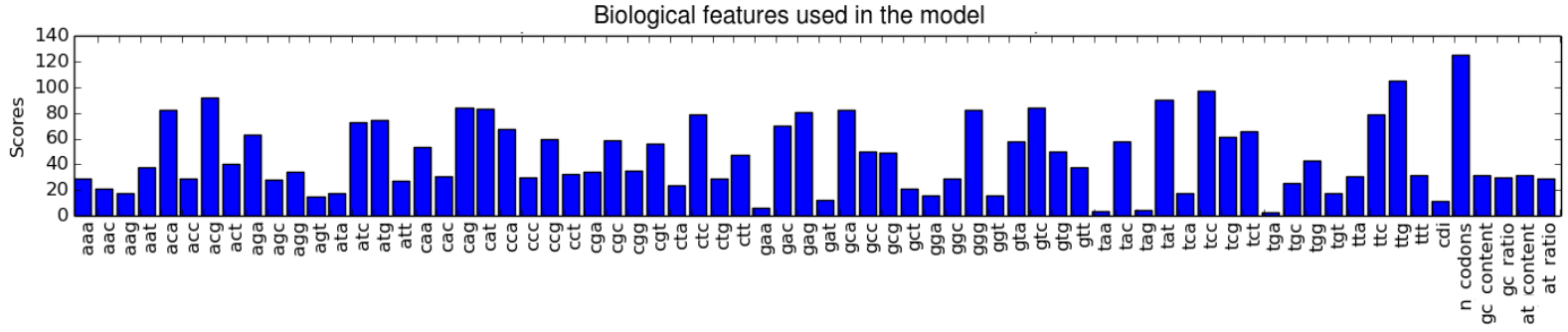
Simulating-matching approach:

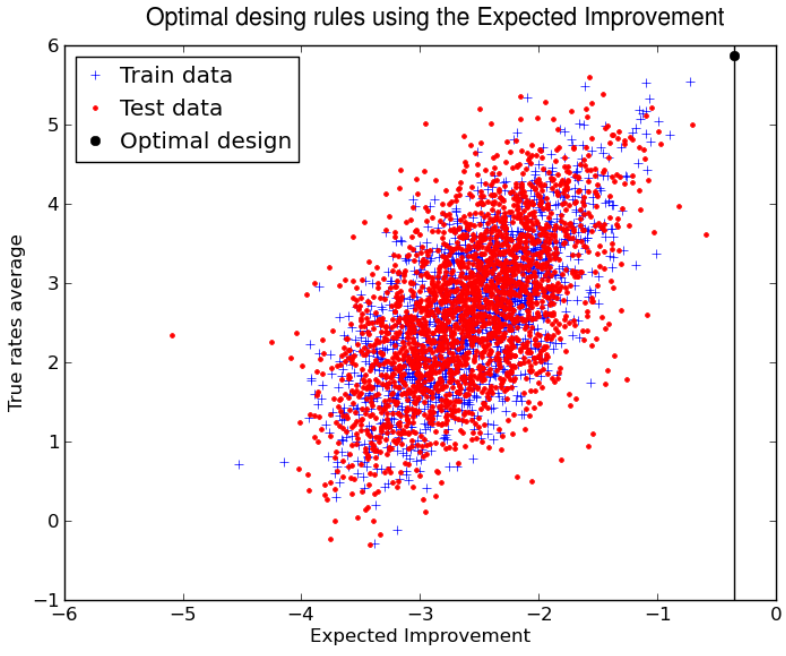
1. Simulate genes 'coherent' with the target (same amino-acids).
2. Extract features.
3. Rank synthetic genes according to their similarity with the 'optimal' design rules.

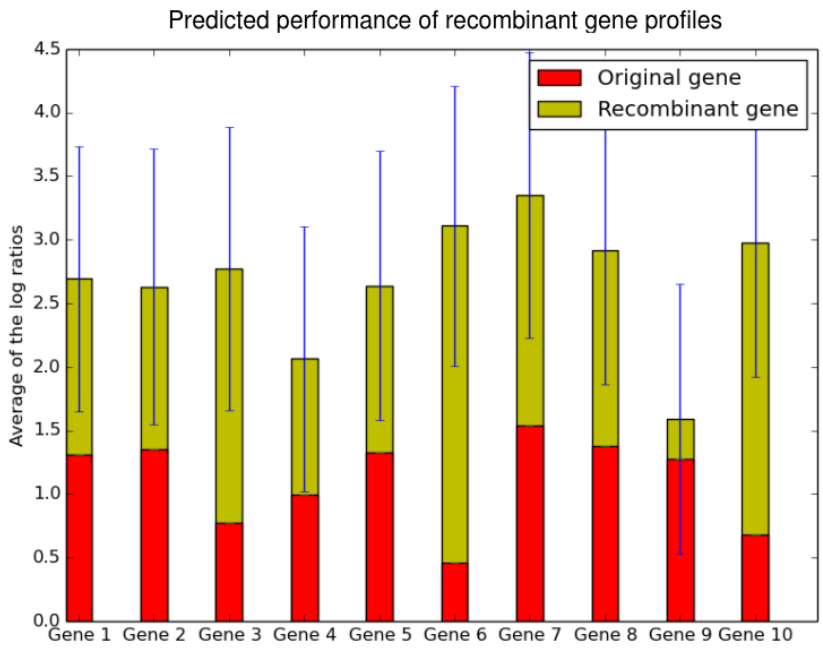
Ranking criterion: $eval(\mathbf{s}|\mathbf{x}^\star) = \sum_{j=1}^{p} w_j |\mathbf{x}_j - \mathbf{x}_j^\star|$

- $\mathbf{x}^\star$: optimal gene design rules.
- $\mathbf{s}$, $\mathbf{x}_j$ generated 'synonyms sequence' and its features.
- w_j: weights of the p features (inverse length-scales of the model covariance).

Results


Biological features used in the model
Scores
0
20
40
60
80
100
120
140
aaa aac aag aat aca acc acg act aga agc agg agt ata atc atg att caa cac cag cat cca ccc ccg cct cga cgc cgg cgt cta ctc ctg ctt gaa gac gag gat gca gcc gcg gct gga ggc ggg ggt gta gtc gtg gtt taa tac tag tat tca tcc tcg tct tga tgc tgg tgt tta ttc ttg ttt cdi
n codons
gc content
gc ratio
at content
at ratio


Optimal desing rules using the Expected Improvement
Train data
Test data
Optimal design
True rates average
Expected Improvement
−6
−5
−4
−3
−2
−1
0
−1
0
1
2
3
4
5
6


Predicted performance of recombinant gene profiles
Original gene
Recombinant gene
Average of the log ratios
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
Gene 1
Gene 2
Gene 3
Gene 4
Gene 5
Gene 6
Gene 7
Gene 8
Gene 9
Gene 10

Available software

- Spearmint (https://github.com/HIPS/Spearmint).
- BayesOpt (http://rmcantin.bitbucket.org/html/).
- pybo (https://github.com/mwhoffman/pybo).
- robo (https://github.com/automl/RoBO).

GPyOpt

- Python code framework for Bayesian Optimization.
- Developed by the group with other contributions.
- Builds on top of GPy, framework for Gaussian process modelling (any model in GPy can be imported as a surrogate model to do optimization in GPyOpt).
- We started to develop it on Jun 2014.

Main features

Feature	**Availability**
- GPs, warped-GP, RF, etc.	✓
- EI, MPI, GP-UCB	✓
- Internal optimizers: BFGS, DIRECT, CMA-ES	✓
- Model hyperpatemeters integration	✓
- Discrete/continuous/categorical variables	✓
- Bandits optimization	✓
- Parallel/batch optimization	✓
- Arbitrary constrains	✓
- Spearmint compatibility	✓
- Cost functions (including evaluation time)	✓
- Modular optimization	✓
- Structured inputs (conditional dependencies)	✓
- Context variables	✓

Code sample

example.py

```python
import GPyOpt


# --- Objective function
objective = GPyOpt.objective_examples.experiments2d.sixhumpcamel().f


# --- Domain
bounds =[{'name': 'var_1', 'type': 'continuous', 'domain': (-3,3)},
         {'name': 'var_2', 'type': 'continuous', 'domain': (-3,3)]

# --- Constrains
constrains = [{'name': 'constr_1', 'constrain': '-x[:,1] -.5 + abs(x[:,0]) - np.sqrt(1-x[:,0]**2)'},
              {'name': 'constr_2', 'constrain': 'x[:,1] +.5 - abs(x[:,0]) - np.sqrt(1-x[:,0]**2)'}]


# --- Create Optimization object
optim = GPyOpt.methods.BayesianOptimization(objective.f,
                                            domain = bounds,
                                            constrains = constrains,
                                            model_type = 'GP_MCMC',
                                            acquisition_type='EI_MCMC',
                                            batch_size = 4,
                                            num_cores = 2,
                                            normalize_Y = False,
                                            acquisition_jitten = 0.1)


# Run optmization
optim.run_optimization(max_iter = 40, max_time=60)

```