Suppl. Materials: 'Batch Bayesian Optimization via Local Penalization'

Javier González
Dept. of Computer Science
University of Sheffield

Zhenwen Dai
Dept. of Computer Science
University of Sheffield

Philipp Hennig
Max Planck Institute
for Intelligent Systems

Neil Lawrence
Dept. of Computer Science
University of Sheffield

S1 Proof of Proposition 1

We compute the explicit form of the penalization functions $\varphi(\mathbf{x};\mathbf{x}_j)$. The distribution of r_j is Gaussian with mean $(M-\mu_n(\mathbf{x}_j))/L$ and variance $\sigma_n^2(\mathbf{x}_j)/L^2$ by the properties of $f(\mathbf{x}_j)$. Then we obtain that

$$\begin{aligned}
\varphi(\mathbf{x};\mathbf{x}_j) &= 1-p(\mathbf{x}\in B_{r_j}(\mathbf{x}_j))\\
&= 1-p(r_j \geq \|\mathbf{x}_j-\mathbf{x}\|_p)\\
&= p(r_j \leq \|\mathbf{x}_j-\mathbf{x}\|_p)\\
&= p\left(\mathcal{N}(0,1)\leq \frac{L\|\mathbf{x}_j-\mathbf{x}\|_p - M+\mu_n(\mathbf{x}_j)}{\sigma_n(\mathbf{x}_j)}\right)\\
&= \Phi\left(\frac{L\|\mathbf{x}_j-\mathbf{x}\|_p - M+\mu_n(\mathbf{x}_j)}{\sigma_n(\mathbf{x}_j)}\right)\\
&= \frac{1}{2}\operatorname{erfc}(-z)
\end{aligned}$$

for

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

S2 Optimization of the penalized acquisition function

Under the proposed local penalization method, to select the k-th element of the t-th batch requires the optimization of the function

$$\tilde{\alpha}_{t,k}(\mathbf{x};\mathcal{I}_{t,0}) = g(\alpha(\mathbf{x};\mathcal{I}_{t,0}))\prod_{j=1}^{k-1}\varphi(\mathbf{x};\mathbf{x}_{t,j}),$$

which can be done by any gradient descend method as follows. We fist map the problem into the natural log space by observing that

$$\arg\max_{x\in\mathcal{X}}\left\{\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0})\right\} = \arg\max_{x\in\mathcal{X}}\left\{\ln\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0})\right\}.$$

Applying the properties of the logarithms we transform the problem into the maximization of

$$\ln\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0}) = \ln\left[g(\alpha(\mathbf{x};\mathcal{I}_{t,0}))\right] + \sum_{j=1}^{k-1}\ln\left[\varphi(\mathbf{x};\mathbf{x}_{t,j})\right].$$

The gradient with respect to $\mathbf{x}$ is now easy to calculate since the problem is in additive form. First, note that the gradients of the local penalizers $\nabla\varphi(\mathbf{x};\mathbf{x}_{t,j})$ are

$$\nabla\varphi(\mathbf{x};\mathbf{x}_{t,j}) = \frac{e^{-z^2}}{\sqrt{2\pi\sigma_n^2(\mathbf{x}_j)}}\frac{2L}{\|\mathbf{x}_j-\mathbf{x}\|}(\mathbf{x}_j-\mathbf{x}), \tag{S.1}$$

with

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

Then, it holds that

$$\begin{aligned}
\nabla\ln\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0}) &= [g(\alpha(\mathbf{x};\mathcal{I}_{t,0}))^{-1}\\
&\quad \frac{d}{d\alpha(\mathbf{x};\mathcal{I}_{t,0})}g(\alpha(\mathbf{x};\mathcal{I}_{t,0}))]\nabla\alpha(\mathbf{x};\mathcal{I}_{t,0})\\
&+ \sum_{j=1}^{k-1}\varphi(\mathbf{x};\mathbf{x}_{t,j})^{-1}\nabla\varphi(\mathbf{x};\mathbf{x}_{t,j})
\end{aligned}$$

where $\nabla\alpha(\mathbf{x};\mathcal{I}_{t,0})$ is the (assumed known) gradient of the original acquisition function. In cases in which the acquisition is already positive it is natural to choose $g(z) = z$ and the gradient of $\ln\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0})$ reduces to

$$\begin{aligned}\nabla \ln\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0}) &= \alpha(\mathbf{x};\mathcal{I}_{t,0})^{-1}\nabla\alpha(\mathbf{x};\mathcal{I}_{t,0}) + \\ &= \sum_{j=1}^{k-1}\varphi(\mathbf{x};\mathbf{x}_{t,j})^{-1}\nabla\varphi(\mathbf{x};\mathbf{x}_{t,j}).\end{aligned}$$

When $\alpha(\mathbf{x};\mathcal{I}_{t,0})$ is not necessarily positive one can take $g(z) = \exp(z)$ and the gradient simplifies to

$$\nabla \ln\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0}) = \nabla\alpha(\mathbf{x};\mathcal{I}_{t,0}) + \sum_{j=1}^{k-1}\varphi(\mathbf{x};\mathbf{x}_{t,j})^{-1}\nabla\varphi(\mathbf{x};\mathbf{x}_{t,j}).$$

When $g(z) = \ln(1+e^z)$ the gradient is

$$\begin{aligned}\nabla \ln\tilde{\alpha}_{t,k}(\mathbf{x},\mathcal{I}_{t,0}) &= \frac{1}{\ln(1+e^{\alpha(\mathbf{x};\mathcal{I}_{t,0})})}\frac{e^{\alpha(\mathbf{x};\mathcal{I}_{t,0})}}{1+e^{\alpha(\mathbf{x};\mathcal{I}_{t,0})}}\cdot \\ &\quad \nabla\alpha(\mathbf{x};\mathcal{I}_{t,0}) + \sum_{j=1}^{k-1}\varphi(\mathbf{x};\mathbf{x}_{t,j})^{-1}\cdot \\ &\quad \nabla\varphi(\mathbf{x};\mathbf{x}_{t,j}).\end{aligned}$$

S3 Lipschitz constant approximation

In this section we elaborate in the approximation of the Lipschitz constant. First, we include the following proposition that allows to uniquely identify a valid value of L.

Proposition 1 *Let $f : \mathcal{X} \to \mathrm{I\!R}$ be a L-Lipschitz continuous function defined on a compact subset $\mathcal{X} \subseteq \mathrm{I\!R}^d$. Take*

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

where $\nabla f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial \boldsymbol{x}_1}, \cdots, \frac{\partial f}{\partial \boldsymbol{x}_p}\right)^\top$. Then, L_p is a valid Lipschitz constant such that the Lipschitz condition

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq L_p\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_q,$$

where $\frac{1}{s} + \frac{1}{l} = 1$, holds.

Proof 1 *Using the mean value theorem for every $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}$ there exist a $\boldsymbol{w} = \boldsymbol{x}_1 + \beta\boldsymbol{x}_2$, with $\beta \in (0,1)$ such that,*

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| = |\nabla f(\boldsymbol{w})(\boldsymbol{x}_1 - \boldsymbol{x}_2)|.$$

By the Holder's inequality we have that

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq \|\nabla f(\boldsymbol{w})\|_p\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_q.$$

Since $\boldsymbol{w} \in \mathcal{X}$ by definition, we have that

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \leq L_p\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_q,$$

for $L_p = \max_{\boldsymbol{x}\in\mathcal{X}} \|\nabla f(\boldsymbol{x})\|_p$.

In order to test the empirical approximation of the Lipschitz constant detailed in Section 2.2 we use the Cosines function described in the experimental section of this work. The true L_∇ for this function is 8.808636, that was calculated by maximizing the norm of gradient of f in a very fine grid. We check the quality of our approximation to L_∇ for increasing sample size up to 50 observations, where the locations of the points are randomly selected along the domain of f using a bivariate uniform distribution. The evaluations of f at the selected locations were perturbed with Gaussian noise with standard deviations $\sigma = 0, 0.1, 0.25$. In Figure 1 we show the results for 30 replicates of the experiment. The average approximation of L converges to the true L_∇, being this convergence slower when the evaluation errors increase.

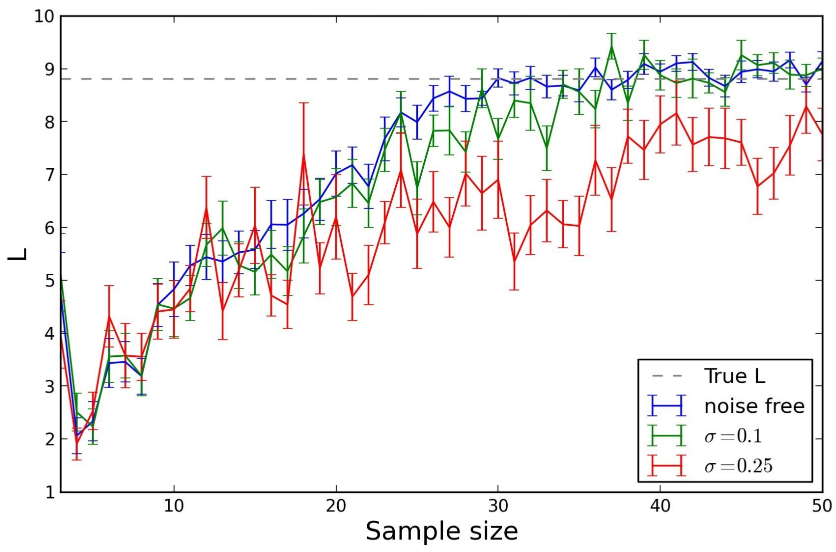

Figure 1: Approximation of the Lipschitz constant in the cosines function using the GP-LCA method. We compare the convergence in the approximation for different noise levels and increasing sample size. For each sample size show we show the average of 30 replications. Vertical bars represent the 95% confidence interval for the average estimate.

Name	Function	$\mathcal{X}$
gSobol	$f(\mathbf{x}) = \prod_{i=1}^{d} \frac{\lvert 4x_i - 2\rvert + a_1}{1+a_i}$	$[-5,5]^d$
Cosines	$f(\mathbf{x}) = 1 - \sum_{i=1}^{2}(g(x_i) - r(x_i))$	$[0,5]^2$
	with $g(x_i) = (1.6x_i - 0.5)^2$,	
	$r(x_i) = 0.3\cos(3\pi(1.6x_i - 0.5))$.	

Table 1: Functions used in the experimental section. All the parameters a_i of the gSobol function are set to $a_i = 1$ in the experiments.

S4 Detailed description of the experiments

S4.1 Synthetic functions

Table 1 contains the the details of the functions used in the experiments of this work.

S4.2 Gene design experiment

There is an increasing interest in the pharmacological industry in the the design of synthetic genes capable of transforming cells into ‘factories’ able to produce drugs of interest. In this experiment we emulate a gene design process.

The function to maximize is the production of cell proteins, that it is known depends on certain features of the gene sequences. We built a GP to link gene features and protein production efficiency based the model described in González et al. [2014]. A total of 71 gene features are considered, which correspond to the dimension of the final design space. We validated the model with the remaining 2908 genes of the dataset and we used its posterior mean as the function to optimize. We can understand this model as a mathematical surrogate of the cell behavior in which the mean evaluations play the role of physical wet-lab gene design tests, many of which can be run in parallel for the same price of one.

S4.3 SVR parameter tuning experiment

Support Vector Machines (SVR) for regression Drucker et al. [1997] with an EQ kernel, depend on three parameters: the kernel lengthscale (γ), the soft margin parameter (C) and the band size (ϵ). A proper choice of the parameters is crucial to guarantee a good performance of the SVR, which is typically done by minimizing the mean square error (RMSE) in a test dataset. This task can be expensive, specially for large datasets. We use BO to optimize the parameters of the SVR using the 'Physiochemical' properties of protein tertiary structure' dataset available in the UCI Machine Learning repository Bache and Lichman [2013]. This dataset has 45,730 instances and 9 continuous attributes that are used to predict the coordinate root mean square distance (RMSD), a measure that describes the distance per residue between to optimally aligned protein sequences. We trained the SVR using a randomly selected subset of 22,000 proteins and we tested the results of using the rest. Every iteration takes around 300 seconds.

References

Kevin Bache and Moshe Lichman. UCI machine learning repository, 2013.

Harris Drucker, Chris, Burges L. Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, pages 155–161, 1997.

Javier González, Joseph Longworth, David James, and Neil Lawrence. Bayesian optimisation for synthetic gene design. *NIPS Workshop on Bayesian Optimization in Academia and Industry*, 2014.