GLASSES: Relieving The Myopia Of Bayesian Optimisation

Anonymous Author 1
Unknown Institution 1

Anonymous Author 2
Unknown Institution 2

Anonymous Author 3
Unknown Institution 3

Abstract

We present GLASSES: Global optimisation with Look-Ahead through Stochastic Simulation and Expected-loss Search. The majority of global optimisation approaches in use are myopic, in only considering the impact of the next function value; the non-myopic approaches that do exist are able to consider only a handful of future evaluations. Our novel algorithm, GLASSES, permits the consideration of dozens of evaluations into the future. This is done by approximating the ideal *look-ahead* loss function, which is expensive to evaluate, by a cheaper alternative in which the future steps of the algorithm are simulated beforehand. An Expectation Propagation algorithm is used to compute and optimise the the expected loss. We show that the far-horizon planning thus enabled leads to substantive performance gains in empirical tests.

1 Introduction

Global optimisation is core to any complex problem where design and choice play a role. Within Machine Learning, such problems are found in the tuning of hyperparameters [Snoek et al., 2012], sensor selection [Garnett et al., 2010] or experimental design [González et al., 2014, Martinez-Cantin et al., 2009]. Most global optimisation techniques are myopic, in considering no more than a single step into the future. Relieving this myopia requires solving the *multi-step lookahead* problem: the global optimisation of an function by considering the significance of the next function evaluation on function evaluations (steps) further into the future. It is clear that a solution to the problem would offer performance gains. For example, consider the case in which we have a budget of two evaluations with which to optimise a function $f(x)$ over the domain $\mathcal{X} = [0, 1] \subset \mathbb{R}$. If we are strictly myopic, our first evaluation will likely be at $x = 1/2$, and our second then at only one of $x = 1/4$ and $x = 3/4$. This myopic strategy will thereby result in ignoring half of the domain $\mathcal{X}$, regardless of the second choice. If we adopt a two-step lookahead approach, we will select function evaluations that will be more evenly distributed across the domain by the time the budget is exhausted. We will consequently be better informed about f and its optimum.

There is a limited literature on the multi-step lookahead problem. Osborne et al. [2009] perform multi-step lookahead by optimising future evaluation locations, and sampling over future function values. This approach scales poorly with the number of future evaluations considered, and the authors present results for no more than two-step lookahead. [Marchant et al., 2014] reframe the multi-step lookahead problem as a partially observed Markov decision process, and adopt a Monte Carlo tree search approach in solving it. Again, the scaling of the approach permits the authors to consider no more than six steps into the future. In the past, the multi-step look ahead problem was studied by Streltsov and Vakili [1999] proposing a utility function that is provably a globally optimal in cases where the model of the function values remains unchanged.

Interestingly, there exists a link between the multi-step lookahead problem and *batch* Bayesian optimisation [Ginsbourger et al., 2009, Azimi et al., 2011, 2012]. In this later case, batches of locations rather than individual observations are selected in each iteration of the algorithm and evaluated in parallel. When such locations are selected *greedily*, that is, one after the other, the key to selecting good batches relies on the ability of the batch criterion of predicting future steps of the algorithm. In this work we will exploit this parallelism to compute a non-myopic loss for Bayesian optimisation.

This paper is organised as follows. In Section 2 we formalise the problem and describe the contributions

Preliminary work. Under review by AISTATS 2016. Do not distribute.

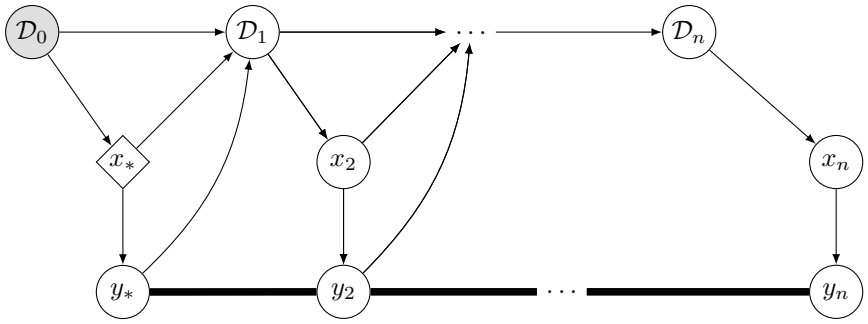

Figure 1: A Bayesian network describing the n-step lookahead problem. The shaded node ($\mathcal{D}_0$) is known, and the diamond node (x_*) is the current decision variable. All y nodes are correlated with one another under the GP model.

of this work. Section 3 describe the details of the proposed algorithm. Section 4 illustrate the superior performance of GLASSES in a variety of test functions and we conclude in Section 5 with a discussion about the most interesting results observed in this work.

2 Background and challenge

2.1 Bayesian Optimisation with one step look-ahead

Let $f : \mathcal{X} \to \mathbb{R}$ be well behaved function defined on a compact subset $\mathcal{X} \subseteq \mathbb{R}^q$. We are interested in solving the global optimization problem of finding

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

We assume that f is a *black-box* from which only perturbed evaluations of the type $y_i = f(\mathbf{x}_i) + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, are available. Bayesian Optimization (BO) is an heuristic strategy to make a series of evaluations $\mathbf{x}_1, \ldots, \mathbf{x}_n$ of f, typically very limited in number, such that the the minimum of f is evaluated as soon as possible [Lizotte, 2008, Jones, 2001, Snoek et al., 2012, Brochu et al., 2010].

Assume that N points have been gathered so far, having a dataset $\mathcal{D}_0 = \{(\mathbf{x}_i, y_i)\}_{i=1}^N = (\mathbf{X}_0, \mathbf{y}_0)$. Before collecting any new point, a surrogate probabilistic model for f is calculated. This is typically a Gaussian Process (GP) $p(f) = \mathcal{GP}(\mu; k)$ with mean function μ and a covariance function k, and whose parameters will be denoted by $\boldsymbol{\theta}$. Let $\mathcal{I}_0$ be the current available information: the conjunction of $\mathcal{D}_0$, the model parameters and the model likelihood type. Under Gaussian likelihoods, the predictive distribution for y_* at $\mathbf{x}_*$ is also Gaussian with mean posterior mean and variance

$$\mu(\mathbf{x}_*|\mathcal{I}_0) = \mathbf{k}_{\boldsymbol{\theta}}(\mathbf{X}_*)^\top [\mathbf{k}_{\boldsymbol{\theta}} + \sigma^2\mathbf{I}]^{-1}\mathbf{y} \text{ and}$$

$$\sigma^2(\mathbf{x}_*|\mathcal{I}_0) = k_{\boldsymbol{\theta}}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_*)^\top [\mathbf{K}_{\boldsymbol{\theta}} + \sigma^2\mathbf{I}]^{-1}\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_*),$$

where $\mathbf{K}_{\boldsymbol{\theta}}$ is the matrix such that $(\mathbf{K}_{\boldsymbol{\theta}})_{ij} = k_{\boldsymbol{\theta}}(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k}_{\boldsymbol{\theta}}(\mathbf{x}_*) = [k_{\boldsymbol{\theta}}(\mathbf{x}_1, \mathbf{x}_*), \ldots, k_{\boldsymbol{\theta}}(\mathbf{x}_n, \mathbf{x}_*)]^\top$ [Rasmussen and Williams, 2005].

Given the GP model, we now need to determine the best location to sample. Imagine that we only have one remaining evaluation ($n = 1$) before we need to report our inferred location about the minimum of f. Denote by $\eta = \min\{\mathbf{y}_0\}$, the current best found value. We can define the loss of evaluating f this last time at $\mathbf{x}_*$ assuming it is returning y_* as

$$\lambda(y_*) \triangleq \begin{cases} y_*; & \text{if} \quad y_* \leq \eta \\ \eta; & \text{if} \quad y_* > \eta. \end{cases}$$

Therefore the loss corresponds is the new observed minimum, $\min(\eta, y_*)$. Its expectation is

$$\Lambda_1(\mathbf{x}_*|\mathcal{I}_0) \triangleq \mathbb{E}[\min(y_*, \eta)] = \int \lambda(y_*) p(y_*|\mathbf{x}_*, \mathcal{I}_0) \mathrm{d}y_*$$

where the subscript in Λ refers to the fact that we are considering one future evaluations. Giving the properties of the GP, $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$ can be computed in closed form for any $\mathbf{x}_* \in \mathcal{X}$. In particular, for Φ the usual Gaussian cumulative distribution function, we have that

$$\begin{aligned} \Lambda_1(\mathbf{x}_*|\mathcal{I}_0) &\triangleq \eta \int_\eta^\infty \mathcal{N}(y_*; \mu, \sigma^2) \mathrm{d}y_* \\ &+ \int_{-\infty}^\eta y_* \mathcal{N}(y_*; \mu, \sigma^2) \mathrm{d}y_* \\ &= \eta + (\mu - \eta)\Phi(\eta; \mu, \sigma^2) - \sigma^2 \mathcal{N}(\eta, \mu, \sigma^2), \end{aligned} \tag{1}$$

where we have abbreviated $\sigma^2(y_*|\mathcal{I}_0)$ as σ^2 and $\mu(y_*|\mathcal{I}_0)$ as μ. Finally, the next evaluation is located where $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$ gives the minimum value. This point can be obtained by any gradient descent algorithm since analytical expressions for the gradient and Hessian of $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$ exist [Osborne, 2010].

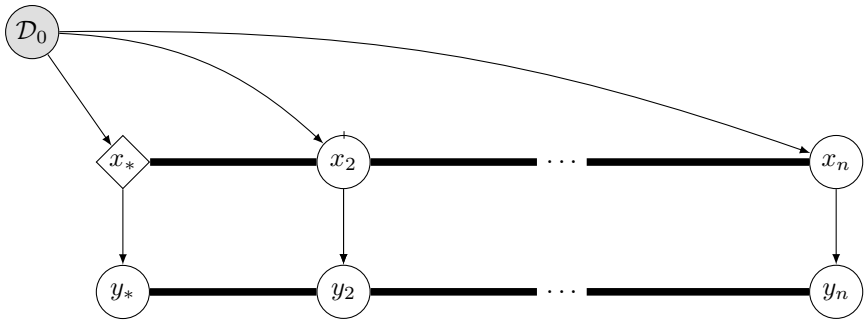

Figure 2: A Bayesian network describing our approximation to the n-step lookahead problem. The shaded node ($\mathcal{D}_0$) is known, and the diamond node (x_*) is the current decision variable, which is now directly connected with all future steps of the algorithm.

2.2 Looking many steps ahead

Expression (1) can also be used as a myopic approximation to the optimal decision when n evaluations of f remain available. Indeed, most BO methods are myopic and ignore the future decisions that will be made by the algorithm in the future steps.

Denote by $\{(\mathbf{x}_j, y_j)\}$ for $j = 1, \ldots, n$ the remaining n available evaluations and by $\mathcal{I}_j$ the available information after the data set $\mathcal{D}_0$ has been augmented with $(\mathbf{x}_j, y_j), \ldots, (\mathbf{x}_j, y_j)$ and the parameters $\boldsymbol{\theta}$ of the model updated. We use $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$ to denote the expected loss of selecting $\mathbf{x}_*$ given $\mathcal{I}_0$ and considering n future evaluations. A Proper Bayesian formulation allows us to define this *long-sight* loss [Osborne, 2010] as[1]

$$\Lambda_n(\mathbf{x}_*|\mathcal{I}_0) = \int \lambda(y_n) \prod_{j=1}^{n} p(y_j|\mathbf{x}_j, \mathcal{I}_{j-1}) p(\mathbf{x}_j|\mathcal{I}_{j-1}) \, \mathrm{d}y_* \ldots \mathrm{d}y_n \mathrm{d}\mathbf{x}_2 \ldots \mathrm{d}\mathbf{x}_n \quad (2)$$

where

$$p(y_j|\mathbf{x}_j, \mathcal{I}_{j-1}) = \mathcal{N}\left(y_j; \mu(\mathbf{x}_j; \mathcal{I}_{j-1}), \sigma^2(\mathbf{x}_j|\mathcal{I}_{j-1})\right)$$

is the predictive distribution of the GP at $\mathbf{x}_j$ and

$$p(\mathbf{x}_j|\mathcal{I}_{j-1}) = \delta(\mathbf{x}_j - \arg\min_{\mathbf{x}_* \in \mathcal{X}} \Lambda_{n-j+1}(\mathbf{x}_*|\mathcal{I}_{j-1}))$$

reflects the optimization step required to obtain $\mathbf{x}_j$ after all previous the evaluations f have been iteratively optimized and marginalized. The graphical probabilistic model underlying (2) is illustrated in Figure 1.

To evaluate Eq. (2) we can successively sample from y_1 to y_{j-1} and optimize for the appropriate $\Lambda_{n-j+1}(\mathbf{x}_*|\mathcal{I}_{j-1})$. This is in done in [Osborne, 2010] for only two steps look ahead. The reason why further horizons remain unexplored is the computational burden required to compute this loss for many steps ahead. Note that analytical expression are only available in the myopic case $\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)$.

[1]We assume that $p(\mathbf{x}_*|\mathcal{I}_0) = 1$

2.3 Contributions of this work

The goal of this work is *to propose a computationally efficient approximation to Eq. (2) for many steps ahead able to relieve the myopia of classical Bayesian optimization.* The precise contributions of this paper are:

- A new algorithm, GLASSES, to relieve the myopia of Bayesian optimisation that is able to efficiently take into account dozens of steps ahead. The method is based on the prediction of the future steps of the myopic algorithm to efficiently integrate out a long-side loss.
- The key aspect of our approach is to split the recursive optimization marginalization loop in Eq. (2) into two independent optimisation-marginalization steps that jointly act on all the future steps. We propose an Expectation-Propagation formulation for the joint marginalisation and we discuss different strategies to carry out the optimisation step.
- Together with this work, we deliver a *open source Python code framework*[2] containing a fully functional implementation of the method useful to reproduce the results of this work and applicable in general global optimisation problems. As we mentioned in the introduction of this work, there exist a limited literature in BO non-myopic methods and, to our knowledge, none available generic BO package can be used with myopic loss functions.
- Simulations: New practical experiments and insights that show that non-myopic methods outperform myopic approaches in a benchmark of optimisation problems.

[2]Link removed for blind review

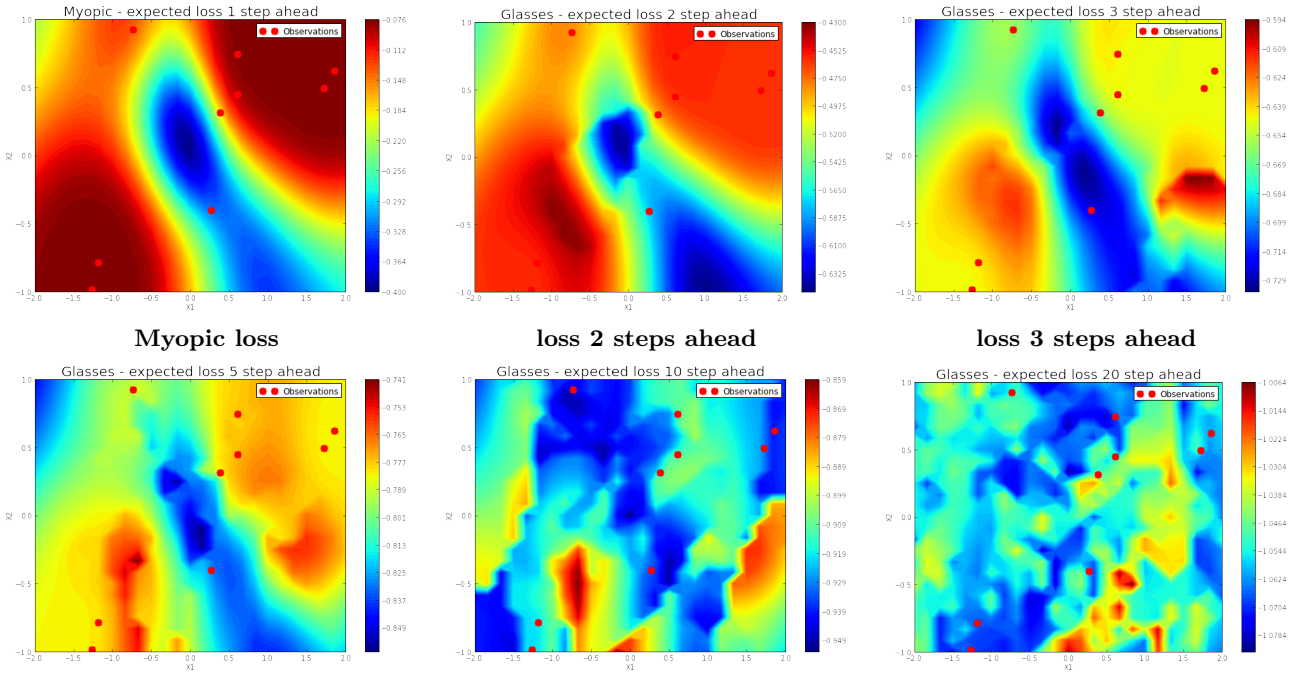

Table 1: Expected loss for different number of steps ahead in an example with 10 data points and the Six-hump Camel function. Increasing the number of steps ahead flatten down the loss since it is likely for the algorithm to hit a good location irrespective of the initial point (all candidate points look better because of the future chances of the algorithm to be in a good location).

3 The GLASSES Algorithm

As detailed in the previous section, a proper multi-step look ahead loss function requires the iterative optimization-marginalization of the future steps, which is computationally intractable. A possible way of dealing with this issue in to jointly modelling our epistemic uncertainty over the future locations $\mathbf{x}_2, \dots, \mathbf{x}_n$ with a joint probability distribution $p(\mathbf{x}_2, \dots, \mathbf{x}_n | \mathcal{I}_0, \mathbf{x}_*)$ and to consider the expected loss

$$\Gamma_n(\mathbf{x}_*|\mathcal{I}_0) = \int \lambda(y_n) p(\mathbf{y}|\mathbf{x}, \mathcal{I}_0, \mathbf{x}_*) p(\mathbf{x}|\mathcal{I}_0, \mathbf{x}_*) \mathrm{d}\mathbf{y}\mathrm{d}\mathbf{x} \tag{3}$$

for $\mathbf{y} = \{y_*, \dots, \dots, y_n\}$ the vector of future evaluations of f and $\mathbf{X}$ the $(n-1) \times d$ dimensional matrix whose rows are the future evaluatios $\mathbf{x}_2, \dots, \mathbf{x}_n$. $p(\mathbf{y}|\mathbf{x}, \mathcal{I}_0, \mathbf{x}_*)$ is multivariate Gaussian, since it corresponds to the predictive distribution of the GP at $\mathbf{X}$. The graphical probabilistic model underlying (2) is illustrated in Figure 2. $\Gamma_n(\mathbf{x}_*|\mathcal{I}_0)$ differs from $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$ in the fact that all future evaluations are modelled jointly rather then sequentially. A proper choice of $p(\mathbf{X}|\mathcal{I}_0, \mathbf{x}_*)$ is crucial here. An interesting option would be to choose $p(\mathbf{X}|\mathcal{I}_0, \mathbf{x}_*)$ to be some determinantal point process (DPP)[3] defined on $\mathcal{X}$ [Affandi et al., 2014] and integrate Eq. (3) with respect to $\mathbf{x}_2, \dots, \mathbf{x}_n$ by averaging over multiple samples [Kulesza and Taskar, 2012, Kulesza and Taskar]. However, although DPPs have nice computational properties in discrete sets, here we would need to take samples from the DPP by conditioning on $\mathbf{x}_*$ and the number of steps ahead. Although this is possible in theory, the computational burden of generating this samples will make this strategy impractical.

An alternative, and more efficient approach that we explore here is to work with a fixed $\mathbf{X}$, which we assume it is computed beforehand. As we show in this section, although this approach does not make use of our epistemic uncertainty on the future steps, it drastically reduces the computational burden of approximating $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$.

3.1 Oracle multiple steps look-ahead expected loss

Suppose that we had access to an oracle function $\mathcal{F}_n : \mathcal{X} \to \mathcal{X} \cdots^n \mathcal{X}$ able to predict the n future lo-

[3] A determinantal point process is probability measure over sets that is entirely characterised by the determinant of some (kernel) function.

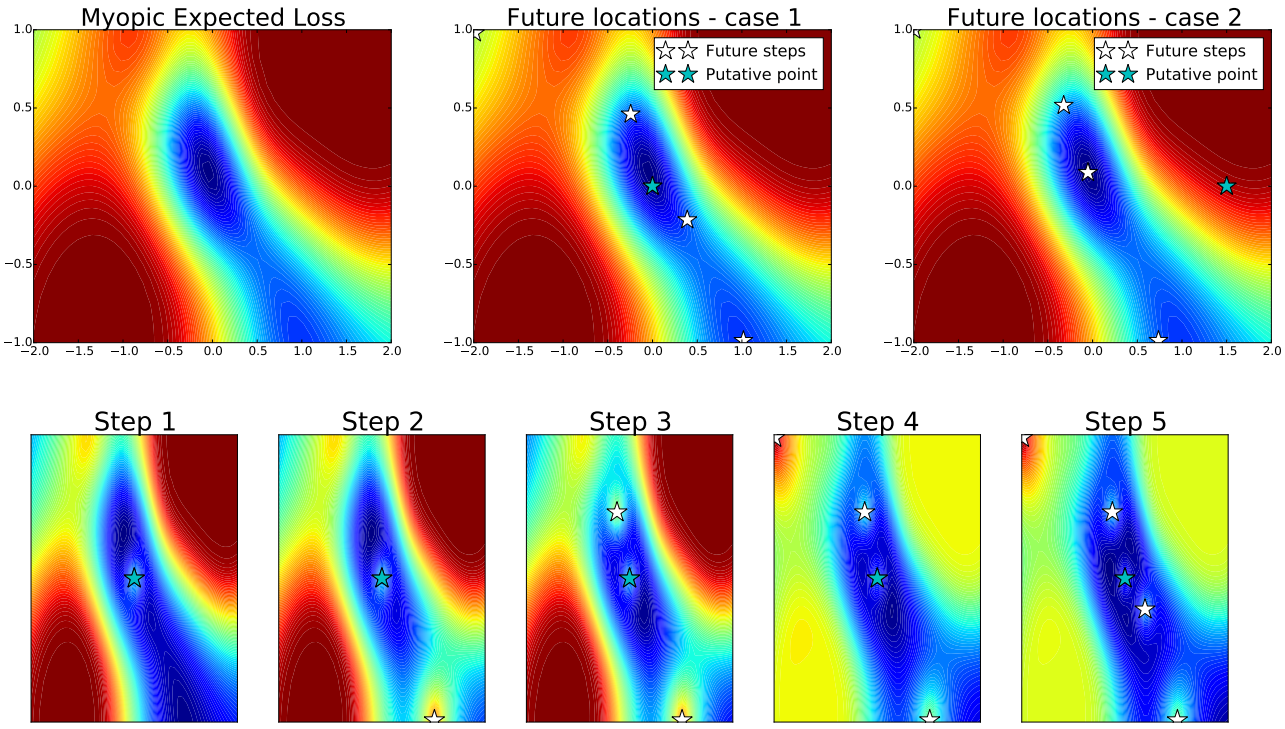

Figure 3: *Top row*: (left) Myopic expected loss computed after 10 observations of the Six-Hump Camel function; (center, case 1) predicted steps ahead when the putative point, grey star, is close to the global optimum of the acquisition; (right, case 2) predicted steps ahead when the putative input is located far from the optimum of the acquisition. *Bottom row*: iterative decision process that predicts that 5 steps look ahead of case 2. Every time a point is selected, the loss is penalised in a neighbourhood, encouraging the next location to be selected far from any previous location point but still in a region where the value of the loss is low.

cations that the loss $\Lambda_n(\cdot)$ would suggest if we started evaluating f at $\mathbf{x}_*$. In other words, $\mathcal{F}_n$ takes the putative location $\mathbf{x}_*$ as input and it returns $\mathbf{x}_*$ and the predicted future locations $\mathbf{x}_2, \ldots, \mathbf{x}_n$. We work here under the assumption that the oracle has perfect information about the future locations, in the same way we have have perfect information about the locations that the algorithm already visited. This is an unrealistic assumption in practice, but it will help us to set-up our algorithm. We leave for the next section the details of how to marginalise over $\mathcal{F}_n$.

Assume, for now, that $\mathcal{F}_n$ exists and that we have access to it. We it and denote by $\mathbf{y} = (y_*, \ldots, \ldots, y_n)^T$ the vector of future locations evaluations of f at $\mathcal{F}_n(\mathbf{x}_*)$. Assuming that $\mathbf{y}$ is known, it is possible to rewrite the expected loss in Eq. (2) as

$$\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) = \mathbb{E}[\min(\mathbf{y}, \eta)], \tag{4}$$

where the expectation is taken over the multivariate Gaussian distribution, with mean vector μ and covariance matrix Σ, that gives rise after marginalizing the posterior distribution of the GP at $\mathcal{F}_n(\mathbf{x}_*)$. Note that under a fixed $\mathcal{F}_n(\mathbf{x}_*)$, it also holds that $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) = \Gamma_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*))$. See supplementary materials for details.

The intuition behind Eq. (4) is as follows: the expected loss at $\mathbf{x}_*$ is the best possible function value that we expect to find in the next n steps, conditional on the first evaluation being made at $\mathbf{x}_*$. Rather than merely quantify the benefit provided by the next evaluation, this loss function accounts for the expected gain in the whole future sequence of evaluations of f. As it is illustrated in Figure 2, the effect of this is an adaptive loss that tends to be more explorative the more remaining evaluations are available and more explorative as soon as we approach to the final evaluations of f.

To compute Eq. (4) we use Expectation Propagation (EP) [Minka, 2001]. This turns out to be a natural operation by observing that

$$\begin{aligned} \mathbb{E}[\min(\mathbf{y}, \eta)] &= \eta \int_{\mathbb{R}^n} \prod_{i=1}^{n} h_i(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y} \\ &+ \sum_{j=1}^{n} \int_{\mathbb{R}^n} y_j \prod_{i=1}^{n} t_{j,i}(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y} \end{aligned} \tag{5}$$

Algorithm 1 Decision process of the full GLASSES algorithm.

Input: dataset $\mathcal{D}_0 = \{(\mathbf{x}_0, y_0)\}$, number of remaining evaluations (n), look-ahead predictor $\mathcal{F}$.
for $j = 0$ **to** n **do**
1. Fit a GP with kernel k to $\mathcal{D}_j$.
2. Build a predictor of the future $n - l$ evaluations: $\hat{\mathcal{F}}_{n-j}$.
3. Select next location $\mathbf{x}_j$ by taking $\mathbf{x}_j = \arg\min_{\mathbf{x}\in\mathcal{X}} \Lambda_{n-j}(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_{n-j}(\mathbf{x}_*))$.
4. Evaluate f at $\mathbf{x}_j$ and obtain y_j.
5. Augment the dataset $\mathcal{D}_{j+1} = \{\mathcal{D}_j \cup (\mathbf{x}_j, y_j)\}$.

end for
Fit a GP with kernel k to $\mathcal{D}_n$
Returns: Propose final location at $\arg\min_{\mathbf{x}\in\mathcal{X}} \{\mu(\mathbf{x}; \mathcal{D}_n)\}$.

where $h_i(\mathbf{y}) = \mathbb{I}\{y_i > \eta\}$ and

$$t_{j,i}(\mathbf{y}) = \begin{cases} \mathbb{I}\{y_j \leq \eta\} & \text{if i=j} \\ \\ \mathbb{I}\{0 \leq y_i - y_j\} & \text{otherwise.} \end{cases}$$

See supplementary materials for details. The first term in Eq. (5) is a Gaussian probability on an unbounded polyhedron in which the limits are aligned with the axis. The second term is the sum of the Gaussian expectations on different non-axis-aligned different polyhedra defined by the indicator functions. Both terms can be computed with EP using the approach proposed in [Cunningham et al., 2011]. In a nutshell, to compute the integrals one need to replace the indicator functions with univariate Gaussian that play the role of *soft-indicators* in the EP iterations. This method is computationally efficient and scales well for high dimensions. Note that when $n = 1$, Eq. (4) reduces to Eq. (1).

Under the hypothesis of this section, the next evaluation is located where $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*))$ gives the minimum value. We still need, however, to propose a way to approximate the oracle $\hat{\mathcal{F}}_n(\mathbf{x}_*)$, what we do in next section.

3.2 Local Penalisation to Predicting the steps ahead

This section proposes an empirical surrogate $\hat{\mathcal{F}}_n(\mathbf{x}_*)$ for $\mathcal{F}_n(\mathbf{x}_*)$. A sensible option would be to use the *maximum a posteriori probability*, MAP, of the abovementioned DPP. However, as it is the generation of DPP samples, to compute the MAP of a DPP is an expensive operation [Gillenwater et al., 2012]. Alternatively, here we use some ideas that have been recently developed in the batch Bayesian optimisation literature. In a nutshell, batch BO methods aim to define sets of points in $\mathcal{X}$ where f should be evaluated in parallel, rather than sequentially. In essence, key aspect to building a 'good' batch is the same as to computing a good approximation for $\Lambda_n(\mathbf{x}_*|\mathcal{I}_0)$: to predict properly the steps ahead of the algorithm.

In this work we adapt to our context the batch BO idea proposed by González et al. [2015]. Inspired by the repulsion properties of DPP, González et al. [2015] propose to build each batch by iteratively optimising and penalising the acquisition function in a neighbourhood of the already collected points by means of some local penalisers $\varphi(\mathbf{x}; \mathbf{x}_j)$. Note that any other batch method could be used here, but we consider this approach since it is computationally tractable and scales well with the size of the batches (steps ahead in our context) and the dimensionality of the problem.

More formally, assume that the objective function f is L-Lipschitz continuous, that is, it satisfies that

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|, \qquad \forall\, \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X} \quad (6)$$

Take $M = \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ and valid Lipschitz constant L. It is possible to show that the ball

$$B_{r_j}(\mathbf{x}_j) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x}_j - \mathbf{x}\| \leq r_j\} \quad (7)$$

where

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

doesn't contain the minimum of f. Probabilist versions of these balls are used to define the above mentioned penalisers by noting that $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. In particular, $\varphi(\mathbf{x}; \mathbf{x}_j)$ is chosen as the probability that $\mathbf{x}$, any point in $\mathcal{X}$ that is a potential candidate to be a minimum, does not belong to $B_{r_j}(\mathbf{x}_j)$:

$$\varphi(\mathbf{x}; \mathbf{x}_j) = 1 - p(\mathbf{x} \in B_{r_j}(\mathbf{x}_j)). \quad (8)$$

As detailed in González et al. [2015], these functions have close form and create an exclusion zone around the point $\mathbf{x}_j$. The predicted k-th location when looking at n step ahead and using $\mathbf{x}_*$ as putative point is defined as

$$[\hat{\mathcal{F}}_n(\mathbf{x}_*)]_k = \arg\min_{x\in\mathcal{X}} \left\{ g(\Lambda_1(\mathbf{x}_*|\mathcal{I}_0)) \prod_{j=1}^{k-1} \varphi(\mathbf{x}; \hat{\mathbf{x}}_j) \right\}, \quad (9)$$

for $k = 2, \ldots, n$ and where $\varphi(\mathbf{x}; \mathbf{x}_j)$ are local local penalizers centered at $\mathbf{x}_j$ and $g : \mathbb{R} \to \mathbb{R}^+$ is the *softplus* transformation $g(z) = \ln(1 + e^z)$.

To illustrate how $\hat{\mathcal{F}}_n$ computes the steps ahead we include Figure (3). We show the myopic loss in a BO experiment together with predicted locations by $\hat{\mathcal{F}}_n$ for two different putative points. In case 1, the putative point $\mathbf{x}_*$ (grey star) is close to the location of the minimum of the myopic loss (blue region). In case 2, $\mathbf{x}_*$ is located in an uninteresting region. In the bottom row of the figure we show how the points are selected in case 1. The fist putative point creates an exclusion zone that shifts the minimum of the acquisition, where the next location is selected. Iterative, new locations are found by balancing diversity (due to the effect of the exclusion areas) and quality (exploring locations where the loss is low).

3.3 Algorithm and computational details

The full GLASSES algorithm is detailed in ... say something here about the computational complexity

4 Experiments

4.1 Testing the effect of considering multiple steps ahead

To study the validity of our approximation we choose a variety of functions with a range of dimensions and domain sizes. See Table 4.1 for details. We test the GLASSES algorithm using 2, 3, 5, and 10 steps look-ahead to compute the loss function. Also we included the full GLASSES approach in which the number of remaining iterations is used as the number of steps-ahead that we use to compute the expected loss. Each problem is solved using 5 different initialisation, with a number of data points equal to 5. This points are generated uniformly random in the domains of the function are kept the same across the replicates for the different method to ensure a fair comparison. This allows us to compare the average performance of each method on each problem. As baseline we used the myopic expected loss. for comparison purposes, all expected losses were optimised using the dividing rectangles algorithm DIRECT [Jones et al., 1993].

As surrogate model for the functions we used a GP with a squared exponential kernel plus a bias kernel [Rasmussen and Williams, 2005]. The hyper-parameters of the model were optimized by the standard method of maximising the marginal likelihood, using L-BFGS [Nocedal, 1980] for 1,000 iterations and selected the best of 5 random restarts. To compare the methods we used the 'gap' measure of performance [Huang et al., 2006], which is defined as

$$G \triangleq \frac{y(\mathbf{x}_{first}) - y(\mathbf{x}_{best})}{y(\mathbf{x}_{first}) - y(\mathbf{x}_{opt})},$$

where $y(\cdot)$ represents the evaluation of the objective function, $y(\mathbf{x}_{opt})$ is the global minimum, and $\mathbf{x}_{first}$ and $\mathbf{x}_{best}$ are the first and best evaluated point, respectively. For every method tested, the initial point $\mathbf{x}_{first}$ was chosen to be the the best of the generated points used to initialise the model.

Name	Function domain	q
Cosines	$[0, 1] \times [0, 1]$	2
Branin	$[-5, 10] \times [-5, 10]$	2
Sixhumpcamel	$[-2, 2] \times [-1, 1]$	2
McCormick	$[-1.5, 4] \times [-3, 4]$	2
Egg-holder	$[-512, 512] \times [-512, 512]$	2
Powers	$[-1, 1] \times [-1, 1]$	2
Alpine2-2		2
Alpine2-5	$[-10, 10]^q$	5
Alpine2-10		10
Ackely-2		2
Ackely-5	$[-5, 5]^q$	5
Ackely-10		10

Table 2: Details of the functions used in the experiments

Table 4.1 shows the comparative across different functions and methods. None of the methods is universally optimal and the full GLASSES algorithm is the best in XX of the XX cases. The myopic shows the best performance in some of the experiments, but when this happen the full GLASSES method typically shows a very similar performance.

4.2 GLASSES vs. other myopic acquisition functions

We use the Cosines function to compare the performance of GLASSES with respect to the most popular acquisition functions. We use the Expected Improvement (EI), the Maximum probability of Improvement (MPI) and the GP Upper confidence bound (GP-UCB) in which we set to 2 the parameter to balance between exploration and exploitation. See [] for details.

5 Conclusions

sdev.=0.1	EL	GL-2	GL-3	GL-5	GL-10	GL-full
Cosines	0.8161	**0.8423**	0.8118	0.7946	0.7477	**0.8722**
Branin	**0.9973**	0.9928	0.9743	0.9896	**0.9959**	0.9883
Sixhumpcamel	**0.9721**	0.9529	0.9479	0.9378	0.9538	0.9284
McCormick	0.9055	0.9139	0.9189	0.9283	0.9389	**0.9424**
Egg-holder						
Powers	0.2238	0.2449	0.2393	0.2411	0.2360	**0.3688**
Alpine2-2	**0.8010**	0.7570	0.6070	0.6791	0.6480	0.7914
Alpine2-5						
Alpine2-10						
Ackley-2	**0.7333**	0.6382	0.5864	0.6864	0.6293	0.7001
Ackley-5						
Ackley-10						

Table 3: Results for the average 'gap' measure of the 5 replicates. For each function the best result is bolded.

References

R.H. Affandi, E.B. Fox, and B. Taskar. Approximate inference in continuous determinantal processes. In *Neural Information Processing Systems 26*. MIT Press, 2014.

Javad Azimi, Ali Jalali, and Xiaoli Fern. Dynamic batch Bayesian optimization. *CoRR*, abs/1110.3347, 2011.

Javad Azimi, Ali Jalali, and Xiaoli Zhang Fern. Hybrid batch Bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Eric Brochu, Vlad M. Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

John P. Cunningham, Philipp Hennig, and Simon Lacoste-Julien. Gaussian probabilities and expectation propagation. *arXiv:1111.6832 [stat]*, Nov 2011. arXiv: 1111.6832.

R. Garnett, M. A. Osborne, and S. J. Roberts. *Bayesian optimization for sensor set selection*, page 209219. ACM, 2010. ISBN 1605589888. doi: 10.1145/1791212.1791238.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. Near-opt imal map inference for determinantal point processes. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2735–2743. Curran Associates, Inc., 2012.

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A multi-points criterion for deterministic parallel global optimization based on Gaussian processes. *HAL: hal-00260579*, 2009.

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.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil D Lawrence. Batch bayesian optimization via local penalization. *arXiv preprint arXiv:1505.08052*, 2015.

D. Huang, T. T. Allen, W. I. Notz, and N. Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *J. of Global Optimization*, 34(3):441–466, March 2006. ISSN 0925-5001.

D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *J. Optim. Theory Appl.*, 79(1):157–181, October 1993. ISSN 0022-3239.

Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345383, 2001.

Alex Kulesza and Ben Taskar. In Lise Getoor and Scheffe, editors, *ICML*.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(23):123–286, 2012. ISSN 1935-8237. doi: 10.1561/2200000044. URL `http://dx.doi.org/10.1561/2200000044`.

Daniel James Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, 2008. AAINR46365.

Roman Marchant, Fabio Ramos, and Scott Sanner. Sequential Bayesian optimisation for spatial-temporal monitoring. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 2014.

Ruben Martinez-Cantin, Nando de Freitas, Eric Brochu, Jos Castellanos, and Arnaud Doucet. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2): 93–103, August 2009. ISSN 0929-5593, 1573-7527. doi: 10.1007/s10514-009-9130-2.

Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 362–369, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1.

Jorge Nocedal. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation*, 35 (151):773–782, 1980. URL `http://www.jstor.org/stable/2006193`.

Michael Osborne. *Bayesian Gaussian Processes for Sequential Prediction, Optimisation and Quadrature*. PhD thesis, PhD thesis, University of Oxford, 2010.

Michael A. Osborne, Roman Garnett, and Stephen J. Roberts. Gaussian processes for global optimization. In *3rd international conference on learning and intelligent optimization (LION3)*, pages 1–15, 2009.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. *Practical Bayesian optimization of machine learning algorithms*, page 29512959. 2012.

Simon Streltsov and Pirooz Vakili. A non-myopic utility function for statistical global optimization algorithms. *J. Global Optim.*, 14(3):283–298, 1999.

Supplementary materials for: 'GLASSES: Relieving The Myopia Of Bayesian Optimisation'

Authors here

S1 Oracle Multiple Steps loook-ahead Expected Loss

Denote by $\eta_n = \min\{\mathbf{y}_0, y_*, y_2 \dots, y_{n-1}\}$ the value of the best visited location when looking at n evaluations in the future. Note that η_n reduces to the current best lost η in the one step-ahead case. It is straightforward to see that

$$\min(y_n, \eta_n) = \min(\mathbf{y}, \eta).$$

It holds hat

$$\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) \quad = \quad \int \min(\mathbf{y}, \eta) \prod_{j=1}^{n} p(y_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*)) \mathrm{d}y_* \dots \mathrm{d}y_n$$

where the integrals with respect to $\mathbf{x}_2 \dots \mathrm{d}\mathbf{x}_n$ are $p(\mathbf{x}_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*)) = 1$, $j = 2, \dots, n$ since we don't need to optimize for any location and $p(y_j|\mathbf{x}_j, \mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*)) = p(y_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*))$. Notice that

$$\begin{aligned}
\prod_{j=1}^{n} p(y_j|\mathcal{I}_{j-1}, \mathcal{F}_n(\mathbf{x}_*)) &= p(y_n|\mathcal{I}_{n-1}, \mathcal{F}_n(\mathbf{x}_*)) \prod_{j=1}^{n-1} p(y_j|\mathcal{I}_{j-1}\mathcal{F}_n(\mathbf{x}_*)) \\
&= p(y_n, y_{n-1}|\mathcal{I}_{n-2}, \mathcal{F}_n(\mathbf{x}_*)) \prod_{j=1}^{n-2} p(y_j|\mathcal{I}_{j-1}\mathcal{F}_n(\mathbf{x}_*)) \\
&\dots \\
&= p(y_n, y_{n-1}, \dots, y_2|\mathcal{I}_1, \mathcal{F}_n(\mathbf{x}_*)) \prod_{j=1}^{2} p(y_j|\mathcal{I}_{j-1}\mathcal{F}_n(\mathbf{x}_*)) \\
&= p(\mathbf{y}|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*))
\end{aligned}$$

and therefore

$$\Lambda_n(\mathbf{x}_*|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) = \mathbb{E}[\min(\mathbf{y}, \eta)] = \int \min(\mathbf{y}, \eta) p(\mathbf{y}|\mathcal{I}_0, \mathcal{F}_n(\mathbf{x}_*)) d\mathbf{y}$$

S2 Formulation of the Oracle Multiple Steps loook-ahead Expected Loss to be computed using Expectation Propagation

Assume that $\mathbf{y} \sim \mathcal{N}(\mathbf{y}; \mu, \Sigma)$. Then we have that

$$\begin{aligned}
\mathbb{E}[\min(\mathbf{y}, \eta)] &= \int_{\mathbb{R}^n} \min(\mathbf{y}, \eta) \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} \\
&= \int_{\mathbb{R}^n - (\eta, \infty)^n} \min(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} + \int_{(\eta, \infty)^n} \eta \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y}.
\end{aligned}$$

The first term can be written as follows:

$$\int_{\mathbb{R}^n - (\eta, \infty)^n} \min(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} = \sum_{j=1}^{n} \int_{P_j} y_j \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y}$$

where $P_j := \{\mathbf{y} \in \mathbb{R}^n - (\eta, \infty)^n : y_j \leq y_i,\ \forall i \neq j\}$. We can do this because the regions P_j are disjoint and it holds that $\cup_{j=1}^n P_j = \mathbb{R}^n - (\eta, \infty)^n$. Also, note that the $\min(\mathbf{y})$ can be replaced within the integrals since within each P_j it holds that $\min(\mathbf{y}) = y_j$. Rewriting the integral in terms of indicator functions we have that

$$\sum_{j=1}^{n} \int_{P_j} y_j \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y} = \sum_{j=1}^{n} \int_{\mathbb{R}^n} y_j \prod_{i=1}^{n} t_{j,i}(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) \mathrm{d}\mathbf{y} \tag{S.1}$$

where $t_{j,i}(y) = \mathbb{I}\{y_i \leq \eta\}$ if $j = i$ and $t_{j,i}(y) = \mathbb{I}\{y_j \leq y_i\}$ otherwise.

The second term can be written as

$$\int_{(\eta,\infty)^n} \eta \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} = \eta \int_{\mathbb{R}^n} \prod_{i=1}^{n} h_i(\mathbf{y}) \mathcal{N}(\mathbf{y}; \mu, \Sigma) d\mathbf{y} \tag{S.1}$$

where $h_i(\mathbf{y}) = \mathbb{I}\{y_i > \eta\}$. Merge (S.1) and (S2) to obtain Eq. (5).