Bayesian Optimization Primer

Ian Dewancker IAN@SIGOPT.COM
Michael McCourt MIKE@SIGOPT.COM
Scott Clark SCOTT@SIGOPT.COM

1. SigOpt

SigOpt employs Bayesian optimization to help experts tune machine learning models and simulations. Instead of resorting to standard techniques like grid search, random search, or manual tuning, Bayesian optimization efficiently trades off exploration and exploitation of the parameter space to quickly guide the user into the configuration that best optimizes some overall evaluation criterion (OEC) like accuracy, AUC, or likelihood. In this short introduction we introduce Bayesian optimization and several techniques that SigOpt uses to optimize users models and simulations.

For applications and examples of SigOpt using Bayesian optimization in real world problems please visit **https://sigopt.com/research**.

2. Bayesian Optimization

Bayesian optimization is a powerful tool for optimizing objective functions which are very costly or slow to evaluate (Martinez-Cantin et al., 2007; Brochu et al., 2010; Snoek et al., 2012). In particular, we consider problems where the maximum is sought for an expensive function $f : \mathcal{X} \to \mathbb{R}$,

$$\mathbf{x}_{opt} = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}),$$

within a domain $\mathcal{X} \subset \mathbb{R}^d$ which is a bounding box (tensor product of bounded and connected univariate domains).

Hyperparameter optimization for machine learning models is of particular relevance as the computational costs for evaluating model variations is high, d is typically small, and hyperparameter gradients are typically not available.

It is worth noting that Bayesian optimization techniques can be effective in practice even if the underlying function f being optimized is stochastic, non-convex, or even non-continuous.

3. Bayesian Optimization Methods

Bayesian optimization methods (summarized effectively in (Shahriari et al., 2015)) can be differentiated at a high level by their *regression models* (discussed in Section 3.2) and *acquisition functions* (discussed in Section 3.3). Several open source Bayesian optimization software packages exist and many of their methods and techniques are incorporated into SigOpt, where applicable. Spearmint (Snoek et al., 2012; 2014b;a), Hyperopt (Bergstra et al., 2013b;a), SMAC (Hutter et al., 2011b;a; Falkner, 2014) and MOE (Clark et al., 2014), are summarized in Table 1.

Table 1. Bayesian Optimization Software

SOFTWARE	REGR. MODEL	ACQ. FUNCTION
SPEARMINT	GAUSSIAN PROCESS	EXP. IMPROV
MOE	GAUSSIAN PROCESS	EXP. IMPROV
HYPEROPT	TREE PARZEN EST.	EXP. IMPROV
SMAC	RANDOM FOREST	EXP. IMPROV

In addition to these methods, non-Bayesian alternatives for performing hyperparameter optimization include grid search, random search (Bergstra & Bengio, 2012), and particle swarm optimization (Kennedy & Eberhart, 1995). These non-Bayesian techniques are often used in practice due to the administrative overhead and expertise required to get reasonable results from these, and other, open source Bayesian optimization packages. SigOpt wraps a wide swath of Bayesian Optimization research around a simple API, allowing experts to quickly and easily tune their models and leverage these powerful techniques.

3.1. Sequential Model-Based Optimization

Sequential model-based optimization (SMBO) is a succinct formalism of Bayesian optimization and useful when discussing variations (Hutter et al., 2011b; Bergstra et al., 2011; Hoffman & Shahriari, 2014). In this section we will use this formalism to contrast some of the methods employed by the open source techniques from Table 1, upon which SigOpt draws. Typically, a probabilistic regression model $\mathcal{M}$ is initialized using a small set of samples from the domain $\mathcal{X}$. Following this initialization phase, new locations within the domain are sequentially selected by optimizing an *acquisition function* S which uses the current model as a cheap surrogate for the expensive objective f.

Each suggested function evaluation produces an observed result $y_i = f(\mathbf{x}_i)$; note that the observation may be random either because f is random or because the observation process is subject to noise. This result is appended to the historical set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i)\}$, which is used to update the regression model for generating the next suggestion. In practice there is often a quota on the total time or resources available, which imposes a limit T on the total number of function evaluations. Algorithm 1 encapsulates this process.

Algorithm 1 Sequential Model-Based Optimization

```
Input: $f, \mathcal{X}, S, \mathcal{M}$
$\mathcal{D} \leftarrow$ InitSamples$(f, \mathcal{X})$
for $i \leftarrow |\mathcal{D}|$ to $T$ do
    $p(y \,|\, \mathbf{x}, \mathcal{D}) \leftarrow$ FitModel$(\mathcal{M}, \mathcal{D})$
    $\mathbf{x}_i \leftarrow \arg\max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x},\ p(y \,|\, \mathbf{x}, \mathcal{D}))$
    $y_i \leftarrow f(\mathbf{x}_i)$        $\triangleright$ Expensive step
    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_i, y_i)$
end for
```

The initialization sampling strategy is often an important consideration in the SMBO formalism. Viable approaches include random, quasi-random, and Latin hypercube sampling of the domain (Hoffman & Shahriari, 2014).

3.2. Probabilistic Regression Models ($\mathcal{M}$)

Various probabilistic regression models can be used in the Bayesian optimization context (Eggensperger et al., 2013); however, to operate in the standard SMBO algorithm the model must define a predictive distribution $p(y \,|\, \mathbf{x}, \mathcal{D})$. This distribution captures the uncertainty in the surrogate reconstruction of the objective function.

3.2.1. Gaussian Processes

Gaussian processes (GPs) (Rasmussen & Williams, 2006) have become a standard surrogate for modeling objective functions in Bayesian optimization (Snoek et al., 2012; Martinez-Cantin, 2014). In this setting, the function f is assumed to be a realization of a GP with mean function μ and covariance kernel K, $f \sim \mathcal{GP}(\mu, K)$.

The choice of kernel function K in particular can have a drastic effect on the quality of the surrogate reconstruction (Rasmussen & Williams, 2006). Automatic relevance determination (ARD) kernels, also known as anisotropic kernels, are common variants (Snoek et al., 2012; Duvenaud, 2014). By default, Spearmint and MOE use the ARD Matérn and ARD squared exponential kernels, respectively. These kernels define important quantities needed to compute the predictive distribution, including

$$
\begin{aligned}
\mathbf{k}(\mathbf{x}) &= \big(K(\mathbf{x}, \mathbf{x}_1) \ \cdots \ K(\mathbf{x}, \mathbf{x}_i)\big)^T, \\
\mathsf{K}_{j,k} &= K(\mathbf{x}_j, \mathbf{x}_k).
\end{aligned}
$$

Predictions follow a normal distribution, therefore we know $p(y \,|\, \mathbf{x}, \mathcal{D}) = \mathcal{N}(y \,|\, \hat{\mu}, \hat{\sigma}^2)$ (Fasshauer & McCourt, 2015), where, assuming $\mu(\mathbf{x}) \equiv 0$,

$$
\begin{aligned}
\mathbf{y} &= \big(y_1 \ \cdots \ y_i\big)^T, \\
\hat{\mu} &= \mathbf{k}(\mathbf{x})^T (\mathsf{K} + \sigma_n^2 \mathsf{I})^{-1} \mathbf{y}, \\
\hat{\sigma}^2 &= K(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T (\mathsf{K} + \sigma_n^2 \mathsf{I})^{-1} \mathbf{k}(\mathbf{x}).
\end{aligned}
$$

Spearmint can be flexibly configured to either assume the objective function is noiseless or attempt to estimate the process noise parameter σ_n^2, while MOE allows for only a manual specification of σ_n^2 for now.

3.2.2. Random Forests

Another choice for the probabilistic regression model is an ensemble of regression trees (Breiman et al., 1984; Breiman, 2001). Random forests are used by the Sequential Model-based Algorithm Configuration (SMAC) library (Hutter et al., 2011b). One common approach in constructing the predictive distribution is to assume a Gaussian $\mathcal{N}(y \mid \hat{\mu}, \hat{\sigma}^2)$. The parameters $\hat{\mu}$ and $\hat{\sigma}$ may be chosen as the empirical mean and variance of the regression values, $r(\mathbf{x})$, from the set of regression trees B in the forest,

$$
\begin{aligned}
\hat{\mu} &= \frac{1}{|B|} \sum_{r \in B} r(\mathbf{x}), \\
\hat{\sigma}^2 &= \frac{1}{|B| - 1} \sum_{r \in B} (r(\mathbf{x}) - \hat{\mu})^2.
\end{aligned}
$$

An attractive property of regression trees is that they naturally support domains with conditional variables (Hutter et al., 2009; Bergstra et al., 2011), that is, variables that only exist when another takes on a certain configuration or range. As an example, consider a variable that controls the selection between several machine learning algorithms. Each algorithm may have different parameters on different domains; these algorithm parameters are conditional variables, only active depending on the choice of algorithm (Eggensperger et al., 2013). SMAC supports such conditional variables, while the GP backed Spearmint and MOE currently do not.

3.2.3. Tree Parzen Estimators

Using a tree-structured Parzen estimator (TPE) (Bergstra et al., 2011; 2013b) deviates from the standard SMBO algorithm in that it does not define a predictive distribution over the objective function; instead, it creates two hierarchical processes, $\ell(\mathbf{x})$ and $g(\mathbf{x})$ acting as generative models for all domain variables. These processes model the domain variables when the objective function is below and above a specified quantile y^*,

$$p(\mathbf{x} \mid y, \mathcal{D}) = \begin{cases} \ell(\mathbf{x}), & \text{if } y < y^*, \\ g(\mathbf{x}), & \text{if } y \geq y^*. \end{cases}$$

Univariate Parzen estimators (Duda et al., 2000; Murphy, 2012) are organized in a tree structure, preserving any specified conditional dependence and resulting in a fit per variable for each process $\ell(\mathbf{x})$, $g(\mathbf{x})$. With these two distributions, one can optimize a closed form term proportional to expected improvement (Bergstra et al., 2011).

While this tree structure preserves the specified conditional relationships, other variable interdependencies may not be captured. Gaussian processes and random forests, in contrast, model the objective function as dependent on the entire joint variable configuration. One benefit to using TPE is that it naturally supports domains with specified conditional variables.

3.3. Acquisition Function (S)

Acquisition functions define a balance between exploring new areas in the objective space and exploiting areas that are already known to have favorable values. Many suitable functions have been proposed (Jones et al., 1998; Brochu, 2010; Hoffman et al., 2011), including recent work involving information theory (Hernández-Lobato et al., 2014).

The acquisition function used by each method discussed in this article is expected improvement (Jones et al., 1998). Intuitively, it defines the nonnegative expected improvement over the best previously observed objective value (denoted f_{best}) at a given location $\mathbf{x}$,

$$EI(\mathbf{x} \mid \mathcal{D}) = \int_{f_{\text{best}}}^{\infty} (y - f_{\text{best}})\, p(y \mid \mathbf{x}, \mathcal{D})\, \mathrm{d}y.$$

When the predictive distribution $p(y \mid \mathbf{x}, \mathcal{D})$ is Gaussian, $EI(\mathbf{x})$ has a convenient closed form (Jones et al., 1998). A complete Bayesian treatment of EI involves integrating over parameters of the probabilistic regression model (Snoek et al., 2012), as Spearmint does; this contrasts with MOE and SMAC, which fit only a single parameterization.

4. Conclusion

Bayesian optimization represents a powerful tool in helping experts optimize their machine learning models and simulations. Selecting the best Bayesian optimization technique for each problem of interest is often non-intuitive. SigOpt eliminates these issues by wrapping this powerful research behind a simple API, allowing users to quickly realize the promise of Bayesian optimization for their problems.

Detailed comparisons between these and other methods can be found at **https://sigopt.com/research** and in a coming ICML paper.

References

Bergstra, James and Bengio, Yoshua. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

Bergstra, James, Yamins, Dan, and Cox, David D. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. `https://github.com/hyperopt/hyperopt`, 2013a.

Bergstra, James, Yamins, Daniel, and Cox, David. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of The 30th International Conference on Machine Learning*, pp. 115–123, 2013b.

Bergstra, James S, Bardenet, Rémi, Bengio, Yoshua, and Kégl, Balázs. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pp. 2546–2554, 2011.

Breiman, Leo. Random forests. *Machine learning*, 45(1): 5–32, 2001.

Breiman, Leo, Friedman, Jerome H, Olshen, Richard A, and Stone, Charles J. Classification and regression trees. wadsworth. *Belmont, CA*, 1984.

Brochu, Eric. *Interactive Bayesian Optimization*. PhD thesis, The University of British Columbia (Vancouver, 2010.

Brochu, Eric, Brochu, Tyson, and de Freitas, Nando. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 103–112. Eurographics Association, 2010.

Clark, Scott, Liu, Eric, Frazier, Peter, Wang, JiaLei, Oktay, Deniz, and Vesdapunt, Norases. MOE: A global, black box optimization engine for real world metric optimization. `https://github.com/Yelp/MOE`, 2014.

Duda, Richard O., Hart, Peter E., and Stork, David G. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

Duvenaud, David. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

Eggensperger, Katharina, Feurer, Matthias, Hutter, Frank, Bergstra, James, Snoek, Jasper, Hoos, Holger, and Leyton-Brown, Kevin. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 2013.

Falkner, Stefan. pysmac : simple python interface to SMAC. `https://github.com/automl/pysmac`, 2014.

Fasshauer, Gregory and McCourt, Michael. *Kernel-based Approximation Methods in MATLAB*. World Scientific, 2015.

Hernández-Lobato, José Miguel, Hoffman, Matthew W, and Ghahramani, Zoubin. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems*, pp. 918–926, 2014.

Hoffman, Matthew D, Brochu, Eric, and de Freitas, Nando. Portfolio allocation for bayesian optimization. In *UAI*, pp. 327–336. Citeseer, 2011.

Hoffman, Matthew W and Shahriari, Bobak. Modular mechanisms for bayesian optimization. In *NIPS Workshop on Bayesian Optimization*, 2014.

Hutter, Frank, Hoos, Holger H, Leyton-Brown, Kevin, and Stützle, Thomas. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

Hutter, Frank, Hoos, Holger, Leyton-Brown, Kevin, Murphy, Kevin, and Ramage, Steve. SMAC: Sequential model-based algorithm configuration. `http://www.cs.ubc.ca/labs/beta/Projects/SMAC/`, 2011a.

Hutter, Frank, Hoos, Holger H, and Leyton-Brown, Kevin. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pp. 507–523. Springer, 2011b.

Jones, Donald R, Schonlau, Matthias, and Welch, William J. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13 (4):455–492, 1998.

Kennedy, James and Eberhart, Russell C. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pp. 1942–1948, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995.

Martinez-Cantin, Ruben. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *The Journal of Machine Learning Research*, 15(1):3735–3739, 2014.

Martinez-Cantin, Ruben, de Freitas, Nando, Doucet, Arnaud, and Castellanos, José A. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, pp. 321–328, 2007.

Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. the MIT Press, 2006.

Shahriari, Bobak, Swersky, Kevin, Wang, Ziyu, Adams, Ryan P., and de Freitas, Nando. Taking the human out of the loop: A review of bayesian optimization. Technical report, Universities of Harvard, Oxford, Toronto, and Google DeepMind, 2015.

Snoek, Jasper, Larochelle, Hugo, and Adams, Ryan P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Snoek, Jasper, Swersky, Kevin, Larochelle, Hugo, Gelbart, Michael, Adams, Ryan P, and Zemel, Richard S. Spearmint : Bayesian optimization software. `https://github.com/HIPS/Spearmint`, 2014a.

Snoek, Jasper, Swersky, Kevin, Zemel, Richard S., and Adams, Ryan Prescott. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014b.