Constrained Bayesian Optimization with Max-Value Entropy Search

Valerio Perrone, Iaroslav Shcherbatyi, Rodolphe Jenatton[*], Cédric Archambeau, Matthias Seeger
Amazon
Berlin, Germany
{vperrone, siarosla, cedrica, matthis}@amazon.com

Abstract

Bayesian optimization (BO) is a model-based approach to minimize expensive black-boxes, and has been widely used to tune the hyperparameters of complex models such as deep neural networks. For many real-world black-boxes, however, the optimization is further subject to a priori unknown constraints. For example, model training may fail for certain configurations due to divergence or out of memory errors. To handle these failures, we focus on a general formulation of BO with binary feedback constraints. We propose constrained Max-Value Entropy Search (cMES), a novel information theoretic-based acquisition function implementing this formulation. We demonstrate that tuning the probability of constraint violation plays an important role, outperforming other commonly used heuristics. On an extensive set of real-world constrained hyperparameter optimization problems, we demonstrate that cMES compares favourably to prior work, while being simpler to implement and faster than other constrained extensions of entropy search.

1 Introduction

Let $y(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ represent a black-box function over a bounded set $\mathcal{X} \subset \mathbb{R}^p$. For instance, $y(\mathbf{x})$ is the validation error of a deep neural network as a function of its hyperparameters $\mathbf{x}$ (e.g., learning rate, number of layers, layer width, dropout rates). Each evaluation of $y(\mathbf{x})$ requires training the network, which can be expensive. Our aim is to minimize $y(\mathbf{x})$ with as few queries as possible. Bayesian optimization (BO) is an efficient approach to find a minimum of the black-box function $y(\mathbf{x})$, where $\mathbf{x} \in \mathcal{X}$ [1, 2, 3]. The idea is to replace $y(\mathbf{x})$ by a surrogate model, and update this model sequentially by querying the black-box at new points. Query points are found by optimizing an acquisition function, which trades off exploration and exploitation. A variety of surrogate models have been used to describe the black-box function $y(\mathbf{x})$, with the Gaussian Process (GP) being a common choice [4, 3], and a variety of acquisition functions have also been developed, with Expected Improvement (EI) [5], Upper Confidence Bound (UCB) [6], Predictive Entropy Search (PES) [7], and more recently Max-value Entropy Search (MES) [8] being popular choices. However, conventional models and acquisitions are not designed to handle constraints.

Our goal is to minimize the target black-box $y(\mathbf{x})$, subject to a constraint $c(\boldsymbol{x}) \leq \delta$. Both $y(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ and $c(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ are unknown and need to be queried sequentially. The constrained optimization problem we would like to solve is defined as follows:

$$y_\star = \min_{\boldsymbol{x} \in \mathcal{X}} \{y(\boldsymbol{x}) \parallel c(\boldsymbol{x}) \leq \delta\}, \tag{1}$$

[*]Work done while affiliated with Amazon; now at Google Brain, Berlin, rjenatton@google.com

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

where $\delta \in \mathbb{R}$ is a confidence parameter. $y(\mathbf{x})$ and $c(\mathbf{x})$ are conditionally independent in our surrogate model, with different Gaussian process (GP) priors placed on them.[2] In previous work, an evaluation returns real-valued feedback on $c(\mathbf{x})$ [9, 10, 11], which does not cover use cases we are interested in, such as model tuning in the presence of training failures, e.g., out-of-memory (OOM) errors.[3] In these settings, an evaluation returns $z_y \sim N(z_y|y(\boldsymbol{x}), \alpha_y^{-1})$ and $z_c \in \{-1, +1\}$, where $z_c = -1$ for a feasible, $z_c = +1$ for an unfeasible point. In other words, we never observe the latent constraint function $c(\mathbf{x})$ directly, but obtain binary feedback only. We assume $z_c \sim \sigma(z_c c(\boldsymbol{x}))$, where $\sigma(t) = \frac{1}{1+e^{-t}}$ is the logistic sigmoid (other choices are possible). We can then rewrite the constrained optimization problem (1) as $y_\star = \min_{\boldsymbol{x}\in\mathcal{X}} \{y(\boldsymbol{x}) \parallel P(z_c = +1|\boldsymbol{x}) = \sigma(c(\boldsymbol{x})) \leq \sigma(\delta)\}$. This formulation is similar to the one proposed in [10]. The confidence parameter $\sigma(\delta) \in (0, 1)$ controls the size of the (random) feasible region for defining $y_\star$. Finally, note that in the example of OOM training failures, the criterion observation z_y is obtained only if $z_c = -1$ (feasible).

The most established technique to tackle constrained BO is constrained EI (cEI) [9, 10, 12]. However, the probability of constraint violation cannot be controlled with the CEI. This is undesirable in practice, as the cost of violations can vary widely between applications. Several other issues with cEI are detailed in [11]. Another approach was proposed in [11], where PES is extended to the constrained case. Constrained PES (cPES) can outperform cEI and does not require the workarounds mentioned above. However, it is complex to implement and expensive to evaluate. Moreover, the authors neither consider binary constraint feedback nor controlling the probability of constraint violations. A different generalization of EI to the constrained case is given in [13]. The authors represent the constrained minimum by way of Lagrange multipliers, and the resulting query selection problem is solved as a sequence of unconstrained problems. Their approach requires numerical quadrature, is not designed for the binary constraint feedback, and is evaluated only on artificial black-boxes. Moreover, our acquisition function can be optimized by a standard unconstrained optimizer, thus is simple to integrate into BO packages such as GPyOpt [14] .

2 Max-Value Entropy Search with Constraints

Recall the constrained optimization problem (1) giving rise to the constrained minimum $y_\star$. In this section, we derive cMES, a novel max-value entropy search acquisition function, scoring the value of an evaluation at some $\mathbf{x} \in \mathcal{X}$. Importantly, the output of the constraint $z_c \in \{-1, +1\}$ is binary, where $P(z_c = +1|\mathbf{x}) = \sigma(c(\mathbf{x}))$. We assume $y(\cdot)$ and $c(\cdot)$ are given independent Gaussian process priors, with mean zero and covariance functions $k_y(\boldsymbol{x}, \boldsymbol{x}')$ and $k_c(\boldsymbol{x}, \boldsymbol{x}')$ respectively. Moreover, data $\mathcal{D} = \{\mathbf{x}_i, z_{yi}, z_{ci} \| i = 1, \ldots, n\}$ has already been acquired. Since $z_{yi} \sim N(y(\mathbf{x}_i), \alpha_y^{-1})$, the posterior for $y(\cdot)$ is a GP again [4], with marginal mean and variance given by

$$\mu_y(\boldsymbol{x}) = \boldsymbol{k}_y(\boldsymbol{x})^T \boldsymbol{M}^{-1} \boldsymbol{z}_y, \quad \sigma_y^2(\boldsymbol{x}) = k_y(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_y(\boldsymbol{x})^T \boldsymbol{M}^{-1} \boldsymbol{k}_y(\boldsymbol{x}), \quad \boldsymbol{z}_y = [z_{yi}] \in \mathbb{R}^n,$$

where $\boldsymbol{M} = [k_y(\boldsymbol{x}_i, \boldsymbol{x}_j)] + \alpha_y^{-1} \boldsymbol{I} \in \mathbb{R}^{n\times n}$, $\boldsymbol{k}_y(\boldsymbol{x}) = [k_y(\boldsymbol{x}, \boldsymbol{x}_i)] \in \mathbb{R}^n$. On the other hand, $z_{ci} \in \{-1, +1\}$ are binary. In our experiments, we use expectation propagation [15] in order to approximate the posterior for $c(\cdot)$ by a GP. In the sequel, we denote the posterior marginals of these processes at input $\mathbf{x}$ by $P(y) = N(y|\mu_y, \sigma_y^2)$ and $P(c) = N(c|\mu_c, \sigma_c^2)$, dropping the conditioning on $\mathcal{D}$ and $\mathbf{x}$ for convenience. Details on $\mu_c(\boldsymbol{x})$, $\sigma_c^2(\boldsymbol{x})$ are given in [4, Sect. 3.6.1].

To get started, the unconstrained MES acquisition function is given by $\mathcal{I}(y; y_\star) = \mathrm{H}[P(y)] - \mathrm{E}_{y_\star}[\mathrm{H}[P(y|y_\star)]]$, where the expectation is over $P(y_\star|\mathcal{D})$, and $y_\star = \min_{\boldsymbol{x}\in\mathcal{X}} y(\boldsymbol{x})$ [8]. Here, $\mathrm{H}[P(y)] = \int P(y)(-\log P(y))\, dy$ denotes the differentiable entropy, and $P(y|y_\star) \propto P(y)\mathrm{I}_{\{y\geq y_\star\}}$ is a truncated Gaussian. It should be noted that this is a simplifying assumption. In PES [7], the related distribution $P(y|\boldsymbol{x}_\star)$ is approximated, where $\boldsymbol{x}_\star$ is the argmin. Several local constraints on $y(\cdot)$ at $\boldsymbol{x}_\star$ are taken into account, such as $\nabla_{\boldsymbol{x}_\star} y = \mathbf{0}$. This is not done in MES, which simplifies derivations considerably. Second, the expectation over $y_\star$ is approximated by Monte Carlo sampling.

Our extension to constraints with binary feedback is based on the mutual information term

$$\mathcal{I}((y, z_c); y_\star) = \mathrm{H}[P(y, z_c)] - \mathrm{E}_{y_\star}[\mathrm{H}[P(y, z_c|y_\star)]],$$

[2] We limit our attention to modeling the feasible region by a single function $c(\mathbf{x})$. As the latent constraints are pairwise conditionally independent, an extension to multiple constraints is straightforward.

[3] An experiment with binary feedback is provided in [10], using constrained EI as acquisition function. We compare against this setup in our experiments, using our own implementation.

where $y_\star$ is the constrained minimum from (1). Note that we use the noise-free y in place of z_y for simplicity, the same is done in [8]. A variant incorporating Gaussian noise is described in our supplemental material. We first show how to approximate the entropy difference for fixed $y_\star$, then how to sample from $P(y_\star|\mathcal{D})$ in order to approximate $\mathrm{E}_{y_\star}[\cdot]$ by Monte Carlo.

Suppose for a moment that we observed c in place of z_c. What would $P(y, c|y_\star)$ be? If $c \leq \delta$, then $y \geq y_\star$, while if $c > \delta$, our belief in y remains the same. In other words, $P(y, c|y_\star) = Z^{-1}P(y)P(c)\kappa(y, c)$, where $\kappa(y, c) = 1 - \mathrm{I}_{\{c\leq\delta\}}\mathrm{I}_{\{y\leq y_\star\}}$ is an indicator function. The entropy difference $\mathrm{H}[P(y, c)] - \mathrm{H}[P(y, c|y_\star)]$ can now be expressed in terms of

$$\gamma_c = \frac{\delta - \mu_c}{\sigma_c}, \quad \gamma_y = \frac{y_\star - \mu_y}{\sigma_y}, \quad Z_c = \mathrm{E}[\mathrm{I}_{\{c\leq\delta\}}] = \Phi(\gamma_c), \quad Z_y = \mathrm{E}[\mathrm{I}_{\{y\leq y_\star\}}] = \Phi(\gamma_y).$$

Here, $\Phi(t) = \mathrm{E}[\mathrm{I}_{\{n\leq t\}}]$, $n \sim N(0, 1)$ is the cumulative distribution function for a standard normal variate. For example, $Z = \mathrm{E}[\kappa(y, c)] = 1 - Z_c Z_y$. Details are given in our supplemental material. For a binary response $z_c \in \{\pm 1\}$, we need to take into account that less information about $y_\star$ is obtained. Since $P(z_c|c) = \sigma(z_c c)$ is not Gaussian, we approximate $Q(z_c)Q(c|z_c) \approx P(z_c|c)P(c), \quad z_c \in \{\pm 1\}$, where the $Q(c|z_c)$ are Gaussians.[4] Following the reasoning above,

$$\begin{aligned} P(y, z_c|y_\star) &= \int P(y)P(z_c|c)P(c)\kappa(y, c)\, dc \approx \int P(y)Q(z_c)Q(c|z_c)\kappa(y, c)\, dc \\ &= P(y)Q(z_c)\tilde{\kappa}(y, z_c),\ \tilde{\kappa}(y, z_c) = 1 - \mathrm{I}_{\{y\leq y_\star\}}F(z_c),\ F(z_c) = \mathrm{E}_{Q(c|z_c)}[\mathrm{I}_{\{c\leq\delta\}}]. \end{aligned}$$

While $\tilde{\kappa}(y, z_c)$ is not an indicator, it is piece-wise constant, allowing for an analytically tractable computation of the entropy difference:

$$\begin{aligned} &\mathrm{H}[P(y)] + \mathrm{H}[Q(z_c)] - \mathrm{H}[P(y, z_c|y_\star)] = -\log Z \\ &- B\left(\gamma_y h(-\gamma_y)/2 + \tilde{Z}_c^{-1}\mathrm{E}_Q\left[(1 - F(z_c))(-\log(1 - F(z_c))) + (F(z_c) - \tilde{Z}_c)\log Q(z_c)\right]\right), \\ &B = Z_y\tilde{Z}_c Z^{-1} = \left(\exp(-\log Z_y - \log\tilde{Z}_c) - 1\right)^{-1}. \end{aligned}$$

Here,

$$F(z_c) = \mathrm{E}_{Q(c|z_c)}[\mathrm{I}_{\{c\leq\delta\}}], \quad \tilde{Z}_c = \mathrm{E}_Q[F(z_c)], \quad Z = 1 - Z_y\tilde{Z}_c,$$

and $h(x) = N(x|0, 1)/\Phi(-x)$ denotes the hazard function for the standard normal distribution. Note that, as δ grows, $F(z_c)$ and $\tilde{Z}_c$ tend to 1 rapidly, and our complex expression simplifies to MES, independent of the distribution over c. For example, $e^u u \to 0, u = log(1 - F(z_c))$, as $F(z_c) \to 1$. The full derivation is given in the supplement, along with recommendations for a numerically robust implementation. In particular, observe that all terms depending on c and z_c only are independent of $y_\star$, and can therefore be precomputed.

Sampling from $P(y_\star|\mathcal{D})$

In the constrained case, we aim to sample from $P(y_\star|\mathcal{D})$, where $y_\star = \min_{\boldsymbol{x}\in\mathcal{X}}\{y(\boldsymbol{x}) \parallel c(\boldsymbol{x}) \leq \delta\}$. Here, $y(\cdot)$ and $c(\cdot)$ are posterior GPs conditioned on the current data $\mathcal{D}$. In [11], a finite-dimensional random kitchen sink (RKS) approximation is used to draw approximate sample paths, and the constrained problem is solved for these. A simpler approach is used in [8]. They target the cumulative distribution function (CDF) of $y_\star$, which can be written as expectation over $y(\cdot)$ and $c(\cdot)$ of an infinite product. This is approximated by restricting the product over a finite set $\hat{\mathcal{X}}$, and by assuming *independence* of all $y(\boldsymbol{x})$ and $c(\boldsymbol{x})$ for $\boldsymbol{x} \in \hat{\mathcal{X}}$. While this gives rise to a tractable approximation of the CDF, we found this approximation to be problematic in our experiments. As noted in [8], $y_\star$ drawn under these assumptions are underbiased. In fact, due to the independence assumption, this bias gets *worse* the larger $\hat{\mathcal{X}}$ is: $y_\star$ diverges as $|\hat{\mathcal{X}}| \to \infty$. In our experiments, we follow [8] by restricting our attention to a finite set $\hat{\mathcal{X}}$ (we use a Sobol sequence [17]), but then draw *joint* samples of $y(\hat{\mathcal{X}})$ and $c(\hat{\mathcal{X}})$ respectively, based on which $y_\star$ (restricted to $\hat{\mathcal{X}}$) is trivial to compute (details are provided in the supplemental material). While joint sampling scales cubically in the size of $\hat{\mathcal{X}}$, sampling takes less than a second for $|\hat{\mathcal{X}}| = 2000$, the size we used in our experiments.

[4] We use Laplace's approximation here, in particular the accurate approximation $Q(z_c) \approx P(z_c)$ detailed in [16, Sect. 4.5.2].

Optimizers	Unfeasible evaluations (in %)	Average ranking
$cMES, p = 0.1$	39.25	5.6
$cMES, p = 0.5$	41.41	5.57
$cMES, p = 0.9$	46.75	**5.17**
cEI	24.26	6.14
$AP, perc. = 50$	49.63	6.13
$AP, perc. = 75$	36.62	6.1
$AP, perc. = 100$	27.87	5.77
$cEI_{observe}$	35.54	6.1
$cMES_{observe}, p = 0.1$	46.73	6.7
$cMES_{observe}, p = 0.5$	48.93	6.51
$cMES_{observe}, p = 0.9$	53.23	6.21

Table 1: Comparison on 10 `scikit-learn` problems. For cMES, $p = \sigma(\delta)$. AP denotes adaptive percentile. Subscript $observe$ indicates the objective $y(\cdot)$ is observed at unfeasible points.

3 Experiments

Methods were implemented in GPyOpt [14], using a Matérn-5/2 covariance kernel with automatic relevance determination hyperparameters, optimized by type II maximum likelihood [4]. We consider *binary feedback* scenarios, where the latent constraint function $c(\cdot)$ is observed indirectly via $z_c \in \{-1, +1\}$. The model for $c(\cdot)$ uses Bernoulli likelihood, and inference is approximated by expectation propagation [15, 4]. We distinguish two variants of this scenario. In the *observed* variant, the objective $y(\cdot)$ is observed with each evaluation, feasible or not. In the *unobserved* variant, an observation z_y is obtained only if $z_c = -1$ (feasible). We compare against cEI ([10]), which can be used with binary feedback.[5] As baselines, we use random search [18], as well as a novel heuristic called *adaptive percentile* (AP). AP is a variant of the high-value heuristic introduced in [10], where a single GP $y(\cdot)$ is used. Whenever an evaluation is unfeasible, AP is plugging in the p-percentile of all previously observed objective values as target value. Here, $p \geq 50$, and $p = 100$ corresponds to plugging in the maximum observed so far. Finally, we compute cMES by sampling the constrained optimum $y_\star$ as detailed in Section 2 (joint sampling), using $|\hat{\mathcal{X}}| = 2000$ and 10 samples. We also ran the same experiments with different sample sizes. Results are included in the supplement, showing slightly worse performance with 2 and equivalent performance with 40 samples.

We considered 10 constrained HPO problems, spanning different `scikit-learn` algorithms [19], `libsvm` datasets [20], and constraint modalities. The first 6 problems are about optimizing an accuracy metric (AUC for binary classification and coefficient of determination for regression) subject to a constraint on model size, a setup motivated by applications in IOT or mobile devices. The remaining 4 problems require minimizing the error on positives, subject to a limit on the error on negatives. We tune XGBoost, decision tree, random forest, MLP, kNN, and factorization machine models. When sampling a problem, and then a hyperparameter configuration at random, we hit a feasible point with 51.5% probability. More details on the algorithms, datasets, and fraction of feasible points is given in the supplement.

We ran each method to be compared on the 10 hyperparameter optimization (HPO) problems described above, using 20 random repetitions. We start each method with evaluations at 5 randomly sampled candidates. To account for the heterogeneous scales of the 10 blackboxes and compare the relative performance of the competing methods, common practice is to aggregate results based on the *average rank* (lower, better) [21, 22, 23]. We rank[6] methods for the same problem, iteration, and seed according to the best feasible value observed so far, then average over all these.

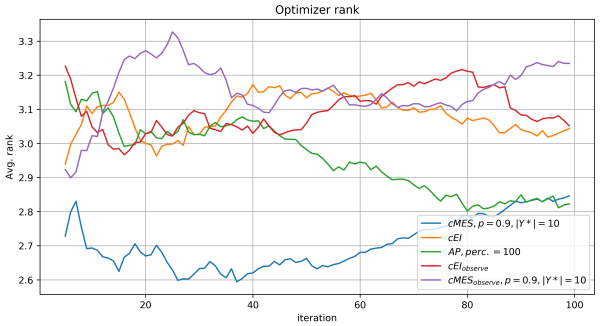

Figure 1: Average per-iteration rank for the best-performing methods in each category. The rankings are different from Table 1 as a subset of methods are compared.

The results in Table 1 and Figure 1 point to a number of conclusions. First, among methods operating in the unobserved scenario, cMES

[5] We could not compare against cPES [11], since they do not support binary constraint feedback.

[6] In initial rounds, some methods may not have made feasible observations. Say, five of ten methods have feasible evaluations. Then, the former are ranked $1, \ldots, 5$, while the latter are equally ranked $(6 + 10)/2 = 8$.

achieves the best overall average rank. While cEI uses fewer unfeasible evaluations, it is overly conservative and tends to converge to worse optima. Second, the AP baseline for $perc = 100$ is surprisingly effective, outperforming cEI. Third, using the value of $y(\cdot)$ in the unfeasible region, where the (unfeasible) global optimum resides, degrades performance for cMES. Finally, Figure 1 shows that cMES ($p = 0.9$) is particularly efficient in early iterations, outperforming all competing methods by a wide margin.

References

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

[2] K Eggensperger, F Hutter, HH Hoos, and K Leyton-brown. Efficient benchmarking of hyperparameter optimizers via surrogates background: Hyperparameter optimization. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1114–1120, 2012.

[3] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[4] Carl Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[5] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.

[6] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58:3250–3265, 2012.

[7] D. Hernandez-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams. Predictive entropy search for multi-objective Bayesian optimization. In M. Balcan and K. Weinberger, editors, *International Conference on Machine Learning 33*. JMLR.org, 2016.

[8] Z. Wang and S. Jegelka. Max-value entropy search for efficient Bayesian optimization. In D. Precup and Y. W. Teh, editors, *International Conference on Machine Learning 34*. JMLR.org, 2017.

[9] Jacob Gardner, Matt Kusner, Zhixiang Xu, Kilian Weinberger, and John Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 937–945, 2014.

[10] Michael A. Gelbart, Jasper Snoek, and Ryan P. Adams. Bayesian optimization with unknown constraints. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 250–259, 2014.

[11] José Miguel Hernández-Lobato, Michael A. Gelbart, Matthew W. Hoffman, Ryan P. Adams, and Zoubin Ghahramani. Predictive entropy search for Bayesian optimization with unknown constraints. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1699–1707, 2015.

[12] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable Bayesian optimization using deep neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2171–2180, 2015.

[13] V. Picheny, R. B. Gramacy, S. Wild, and S. Le Digabel. Bayesian optimization under mixed constraints with a slack-variable augmented Lagrangian. *Advances in Neural Information Processing Systems 29*, pages 1435–1443, 2016.

[14] GPyOpt: A Bayesian optimization framework in Python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[15] T. Minka. Expectation propagation for approximate Bayesian inference. In J. Breese and D. Koller, editors, *Uncertainty in Artificial Intelligence 17*. Morgan Kaufmann, 2001.

[16] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st edition, 2006.

[17] Ilya M Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.

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

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830, 2011.

[20] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[21] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28*, pages 2962–2970, 2015.

[22] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. Collaborative hyperparameter tuning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 199–207, 2013.

[23] A. Klein, Z. Dai, F. Hutter, N. Lawrence, and J. Gonzalez. Meta-surrogate benchmarking for hyperparameter optimization. *arXiv:1905.12982*, 2019.