Bayesian Optimization With Censored Response Data

Frank Hutter, Holger Hoos, and Kevin Leyton-Brown
Department of Computer Science
University of British Columbia
{hutter, hoos, kevinlb}@cs.ubc.ca

Abstract

Bayesian optimization (BO) aims to minimize a given blackbox function using a model that is updated whenever new evidence about the function becomes available. Here, we address the problem of BO under partially *right-censored* response data, where in some evaluations we only obtain a lower bound on the function value. The ability to handle such response data allows us to adaptively censor costly function evaluations in minimization problems where the cost of a function evaluation corresponds to the function value. One important application giving rise to such censored data is the runtime-minimizing variant of the *algorithm configuration* problem: finding settings of a given parametric algorithm that minimize the runtime required for solving problem instances from a given distribution. We demonstrate that terminating slow algorithm runs prematurely and handling the resulting right-censored observations can substantially improve the state of the art in model-based algorithm configuration.

1 Introduction

Right-censored data—data for which only a lower bound on a measurement is available—occurs in several applications. For example, if a patient drops out of a clinical study (for a reason other than death), we know only a lower bound on her survival time. In some cases, one can actively decide to censor certain data points in order to save time or other resources; for example, if a drug variant V_1 is unsuccessful in curing a disease by the time a known drug is successful, one may decide to stop the trial with variant V_1 and instead invest the resources to test a new variant V_2. Here, we describe how to integrate such censored observations into Bayesian optimization (BO). BO aims to find the minimum of a blackbox function $f : \mathbf{\Theta} \rightarrow \mathbb{R}$—a potentially noisy function that is not available in closed form, but can be queried at arbitrary input values. BO proceeds in two phases: (1) constructing a model of f using the observed function values; and (2) using the model to select the input for the next query.

We extend the standard formulation of blackbox function minimization to include a *cost function* $c : \mathbf{\Theta} \rightarrow \mathbb{R}$ that measures the cost of obtaining the function value for a given input. The budget for minimizing f is now given as a limit on the *cumulative cost* of function evaluations (in contrast to the traditional number of allowed function evaluations). We call the resulting blackbox function minimization variant (f, c) *cost-varying*. In this paper, we focus on problems (f, c) with the following *cost monotonicity* property.

Definition 1 *A cost-varying blackbox function minimization problem* (f, c) *is* cost monotonic *if*

$$\forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbf{\Theta}. \left(f(\boldsymbol{\theta}_1) < f(\boldsymbol{\theta}_2) \Leftrightarrow c(\boldsymbol{\theta}_1) < c(\boldsymbol{\theta}_2)\right).$$

For example, a function f may describe how quickly different drug variants cure a disease, or how quickly plants reach a desired size given different fertilizer variants; in these examples, it takes exactly time $f(\boldsymbol{\theta})$ to determine $f(\boldsymbol{\theta})$, *i.e.*, $c = f$. When terminating the function evaluation prematurely after a *censoring threshold* $\kappa < f(\boldsymbol{\theta})$, the cost is only κ, but the resulting censored data point is also less informative: we only obtain a lower bound $\kappa < f(\boldsymbol{\theta})$. Cost monotonicity also applies to minimization objectives other than time, such as energy consumption, communication overhead, or strictly monotonic functions of these.

The application domain motivating our research is the following *algorithm configuration* (AC) problem. We are given a parameterized algorithm A, a distribution D of problem instances $\pi \in \mathcal{I}$, and a performance metric $m(\boldsymbol{\theta}, \pi)$ capturing A's performance with parameter settings $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ on instances $\pi \in \mathcal{I}$. Let $f(\boldsymbol{\theta}) = \mathbb{E}_{\pi \sim D}[m(\boldsymbol{\theta}, \pi)]$ denote the expected performance of A with setting $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ (where the expectation is over instances π drawn from D; in the case of randomized algorithms, it would also be over random seeds). The problem is then to find a parameter setting $\boldsymbol{\theta}$ of A that solves $\arg\min_\theta f(\boldsymbol{\theta})$. Automated procedures (*i.e.*, algorithms) for solving AC have recently led to substantial improvements of the state of the art in a wide variety of problem domains, including SAT-based formal verification [12], mixed integer programming [13], and automated planning [8]. Traditional AC methods are based on heuristic search [18, 9, 1, 16, 3] and racing algorithms [4, 5], but recently the BO method SMAC [14] has been shown to compare favourably to these approaches.

A particularly important performance metric $m(\boldsymbol{\theta}, \pi)$ in the AC domain is algorithm A's *runtime* for solving problem instance π given settings $\boldsymbol{\theta}$. Minimizing this metric within a given time budget is a cost monotonic problem (since it yields $c = f$). Algorithm runs can also be terminated prematurely, yielding a cheaper lower bound on f. As shown in the following, by exploiting this cost monotonicity, we can substantially improve the state of the art in model-based algorithm configuration.

2 Regression Models Under Censoring

Our training data is $(\boldsymbol{\theta}_i, y_i, c_i)_{i=1}^n$, where $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}$ is a parameter setting, $y_i \in \mathbb{R}$ is an observation, and $c_i \in \{0, 1\}$ is a censoring indicator such that $f(\boldsymbol{\theta}_i) = y_i$ if $c_i = 0$ and $f(\boldsymbol{\theta}_i) \geq y_i$ if $c_i = 1$. Various types of models can handle such censored data. In Gaussian processes (GPs)—the most widely used tool for Bayesian optimization—one could use approximations to handle the resulting non-Gaussian observation likelihoods; for example, [7] described a Laplace approximation for handling right-censored data. Here, we use random forests (RFs) [6], which have been shown to yield better predictive performance for the high-dimensional and predominantly discrete inputs typical of algorithm configuration [2]. Following [14], we define the predictive distribution of a RF model F for input $\boldsymbol{\theta}$ as $\mathcal{N}(\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}})$, where $\mu_{\boldsymbol{\theta}}$ and $\sigma^2_{\boldsymbol{\theta}}$ are the empirical mean and variance of predictions of $f(\boldsymbol{\theta})$ across the trees in F. RFs have previously been adapted to handle censored data [20, 11], but the classical methods yield non-parametric Kaplan-Meier estimators that do not lend themselves to Bayesian optimization since they are undefined beyond the largest uncensored data point.

Here, we introduce a simple EM-type algorithm for filling in censored values. We denote the probability density function and the cumulative density function of a standard Normal distribution by ϕ and Φ, respectively. Let $\boldsymbol{\theta}$ be an input for which we observed a censored value $\kappa < f(\boldsymbol{\theta})$. Given a Gaussian predictive distribution $\mathcal{N}(\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}})$ of $f(\boldsymbol{\theta})$, the truncated Gaussian distribution $\mathcal{N}(\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}})_{\geq \kappa}$ is defined by the probability density function

$$p(x) = \begin{cases} 0 & x < \kappa \\ \frac{1}{\sigma_{\boldsymbol{\theta}}} \varphi(\frac{x - \mu_{\boldsymbol{\theta}}}{\sigma_{\boldsymbol{\theta}}}) / (1 - \Phi(\frac{\mu_{\boldsymbol{\theta}} - \kappa}{\sigma_{\boldsymbol{\theta}}})) & x \geq \kappa. \end{cases}$$

Our algorithm is inspired by the EM algorithm of Schmee and Hahn [19]. When applied with an RF model as its base model, that algorithm first fits an initial RF using only uncensored data and then iterates between the following E and M steps:

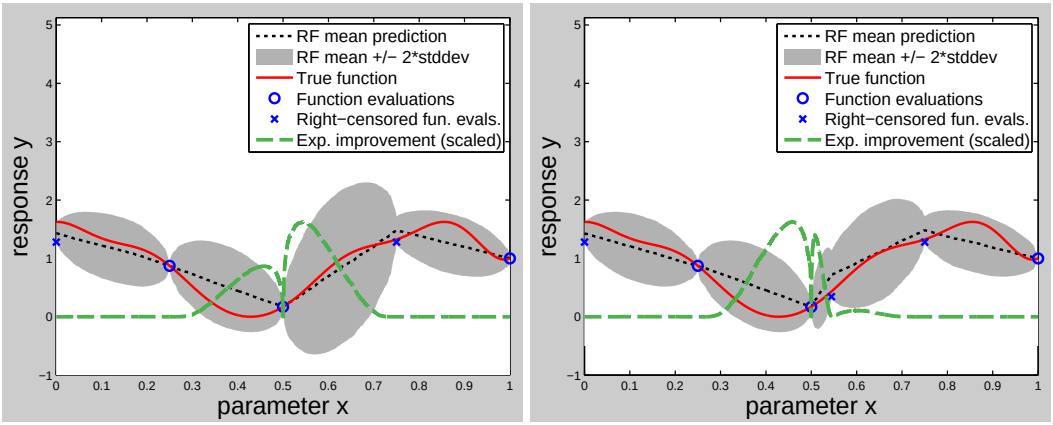
E. For each tree T in the RF and each i s.t. $c_i = 1$: $\hat{y}_i^{(T)} \leftarrow$ mean of $\mathcal{N}(\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}})_{\geq y_i}$;
M. Refit the RF using $(\boldsymbol{\theta}_i, \hat{y_i}^{(T)}, c_i)_{i=1}^n$ as the basis for tree T.

While the mean of $\mathcal{N}(\mu_{\boldsymbol{\theta}}, \sigma^2_{\boldsymbol{\theta}})_{\geq \kappa}$ is the best single value to impute, this algorithm yields overly confident predictions. To preserve our uncertainty about the true value of $f(\boldsymbol{\theta})$, we change Step 1 to:

E. For each tree T in the RF and each i s.t. $c_i = 1$: $\hat{y}_i^{(T)} \leftarrow$ sample from $\mathcal{N}(\mu_{\boldsymbol{\theta}_i}, \sigma^2_{\boldsymbol{\theta}_i})_{\geq y_i}$.

Predictive distributions from this sample-based method for a simple function are visualized in Figure 1(a); note that the predictive variance at censored data points does not collapse to zero (as it would when using Schmee & Hahn's original procedure with a random forest model).[1] As shown in Figure 2, both Schmee & Hahn's procedure and our sampling version yield substantially lower error than either dropping censored data points or treating them as uncensored. By preserving predictive uncertainty for the censored data points, our sampling method yields the highest log likelihoods.

[1] Note that our RF's predictive mean converges to a linear interpolation between data points with a sufficient number of trees, and that its variance grows with the distance from observed data points. (Like the classical approach for building regression trees, at each node we select an interval $[a, b]$ from which to select a split point to greedily minimize the weighted within-node variance of the node's children. Instead of selecting this point as $(a + b)/2$, we sample it uniformly at random from $[a, b]$. This yields linear interpolation in the limit.)



(a) BO with censoring, initialized with an LHD (b) BO with censoring, step 2

Figure 1: Two steps of Bayesian optimization for minimizing a simple 1-D blackbox functions under censoring, starting from a Latin hypercube design (LHD). Circles and x-symbols denote uncensored and right-censored function evaluations, respectively. The dotted line denotes the mean prediction of our random forest model with 1000 trees, and the grey area denotes its uncertainty. The true function is shown as a solid line and expected improvement (scaled for visualization) as a dashed line.

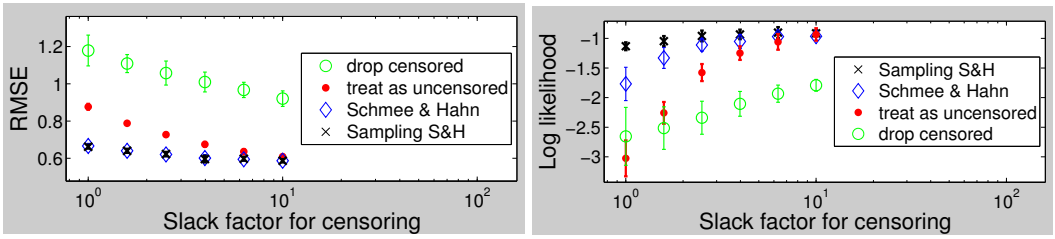


Figure 2: RMSE and log likelihood of various ways of handling censored data with random forests, for various strategies of setting the censoring threshold (larger slack factors mean less censoring; see Section 3).

3 Bayesian Optimization Under Censoring

One standard method for trading off exploration and exploitation in Bayesian optimization is to select the next query point $\boldsymbol{\theta}$ to maximize the expected positive improvement $\mathbb{E}[I(\boldsymbol{\theta})] = \mathbb{E}[\max\{0, f_{\min} - f(\boldsymbol{\theta})\}]$ over the minimal function value f_{min} seen so far. Let $\mu_{\boldsymbol{\theta}}$ and $\sigma^2_{\boldsymbol{\theta}}$ denote the mean and variance predicted by our model for input $\boldsymbol{\theta}$, and define $u = \frac{f_{\min} - \mu_{\boldsymbol{\theta}}}{\sigma_{\boldsymbol{\theta}}}$. Then, one can obtain (see, *e.g.*, [17]) the closed-form expression

$$\mathbb{E}[I(\boldsymbol{\theta})] = \sigma_{\boldsymbol{\theta}} \cdot [u \cdot \Phi(u) + \varphi(u)],$$

where φ and Φ denote the probability density function and cumulative distribution function of a standard normal distribution, respectively. As usual, we maximize this criterion across the input space $\boldsymbol{\Theta}$ to select the next setting $\boldsymbol{\theta}$ to evaluate.

In our problem formulation, we can also pick the censoring threshold κ for this new sample. There is no obvious best choice: increasing κ yields more informative but also more costly data. Here, we heuristically set κ to a multiplicative factor (the "slack factor") times f_{min}. [2] Figure 1 visualizes the first two steps of this procedure for optimizing a blackbox function under censoring.

We now return to algorithm configuration (AC), the problem motivating our research. AC differs from standard problems attacked by Bayesian Optimization (BO) in some important ways: most importantly, categorical input dimensions are common (due to algorithm parameters with finite, non-ordered domains); inputs tend to be high dimensional; the optimization objective is a *marginal* over instances (in the BO literature, this particular problem has, *e.g.*, been addressed by [21, 10]); the objective varies exponentially (good settings perform orders of magnitude better than poor ones), and the overhead of fitting and using models has to be taken into account since it is part of the time budget available for AC. The model-based AC method SMAC addresses these issues, including several

[2] Our approach here is inspired by the "adaptive capping" method used by the algorithm configuration procedure PARAMILS [16]; indeed, we recover that method when the slack factor is 1. We allow for slack factors greater than 1 because they can improve model fits, albeit at the expense of more costly data acquisition.

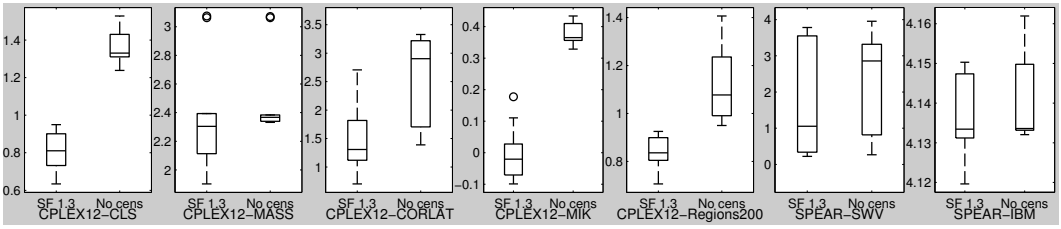


Figure 3: Visual comparison of SMAC's performance with and without censoring; note that the y-axis is runtime on a log_{10} scale (lower is better). To avoid clutter, we only show capping with slack factor 1.3 ("SF 1.3"). We performed 10 independent runs of each configuration procedure C and show boxplots of the test performances for the resulting 10 final configurations (mean runtimes of CPLEX/SPEAR across the test instances).

Scenario	Unit	Median of mean runtimes on test set					
		SF 1	SF 1.1	SF 1.3	SF 1.5	SF 2	No censoring
CPLEX12-CLS	$[\cdot 10^0 s]$	**5.27**	**6.21**	**6.47**	8.3	6.66	21.4
CPLEX12-MASS	$[\cdot 10^2 s]$	**6.39**	**1.94**	**2.02**	**1.94**	**1.97**	**2.33**
CPLEX12-CORLAT	$[\cdot 10^0 s]$	**17.6**	**9.52**	**20.5**	**15.4**	**16.9**	826
CPLEX12-MIK	$[\cdot 10^{-1} s]$	**8.88**	**9.3**	**9.54**	**9.45**	**9.86**	23.9
CPLEX12-Regions200	$[\cdot 10^0 s]$	**6.93**	**6.65**	**6.85**	**7.21**	**8.07**	12
SPEAR-SWV	$[\cdot 10^0 s]$	**67.2**	**521**	**8.15**	**7.78**	**290**	**1030**
SPEAR-IBM	$[\cdot 10^4 s]$	**1.36**	**1.36**	**1.36**	**1.36**	**1.36**	**1.36**

Table 1: Comparison of SMAC without and with censoring (using several slack factors, "SF"). For each configurator and scenario, we report median test performance (defined as in Fig. 3; lower is better). We bold-faced entries for configurators that are not significantly worse than the best configurator for the respective scenario, based on a Mann-Whitney U test (note that the bold-facing of "No censoring" for SPEAR-SWV is not a typo: due to the very large variation for SPEAR-SWV visible in Figure 3 the Null hypothesis was not rejected).

modifications of standard BO methods to achieve state-of-the-art performance for AC (for details, see [15, 14]). Here, we improve SMAC further by setting censoring thresholds as described above and handling the resulting censored data as described in Section 2.

We compared our modified version of SMAC to the original version on a range of challenging real-world configuration scenarios: optimizing the 76 parameters of the commercial mixed integer solver CPLEX on five different sets of problem instances (obtained from [13]), and the 26 parameters of the industrial SAT solver SPEAR on two sets of problem instances from formal verification (obtained from [12]). Each SMAC run was allowed 2 days and the maximum censoring time for each CPLEX/SPEAR run was 10 000 seconds. Algorithm configuration scenarios with such high maximum runtimes have been identified as a challenge for SMAC [14], and we demonstrate here that our adaptive censoring technique substantially improved its performance for these scenarios.

We performed 10 configuration runs for each problem domain and each version of SMAC (no censoring, and censoring with various slack factors), for a total runtime of 840 CPU days. At the end of each configuration run, we recorded SMAC's best found configuration and computed the run's test performance as that configuration's mean runtime on a test set of instances disjoint from the training set, but sampled from the same distribution.

Figure 3 and Table 1 show that our modified version of SMAC with censoring substantially outperformed the original SMAC version without capping. Our modified version gave better results in all 7 cases (with statistical significance achieved in 4 of these). The improvements in median test performance reached up to a factor of 126 (SPEAR-SWV).

4 Conclusion

We have demonstrated that censored data can be integrated effectively into Bayesian optimization (BO). We proposed a simple EM algorithm for handling censored data in random forests and adaptively selected censoring thresholds for new data points at small multiples above the best seen function values. In an application to the problem of algorithm configuration, we achieved substantial speedups of the state-of-the-art procedure SMAC. In future work, we would like to apply censoring in BO with Gaussian processes, actively select the censoring threshold to yield the most information per time spent, and evaluate the effectiveness of BO with censoring in further domains.

Acknowledgments

We would like to thank our summer Co-Op student Jonathan Shen for many useful discussions and for assistance in cleaning up the source code of SMAC.

References

[1] B. Adenso-Diaz and M. Laguna. Fine-tuning of algorithms using fractional experimental design and local search. *Operations Research*, 54(1):99–114, Jan–Feb 2006.

[2] Anonymous. Anonymous. 2011.

[3] C. Ansotegui, M. Sellmann, and K. Tierney. A gender-based genetic algorithm for the automatic configuration of solvers. In *Proc. of CP-09*, pages 142–157, 2009.

[4] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proc. of GECCO-02*, pages 11–18, 2002.

[5] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. *Empirical Methods for the Analysis of Optimization Algorithms*, chapter F-race and iterated F-race: An overview. Springer, Berlin, Germany, 2010.

[6] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[7] E. Ertin. Gaussian process models for censored sensor readings. In *Proceedings of the IEEE Statistical Signal Processing Workshop 2007 (SSP'07)*, pages 665–669, 2007.

[8] C. Fawcett, M. Helmert, H. H. Hoos, E. Karpas, G. Röger, and J. Seipp. FD-Autotune: Domain-specific configuration using fast-downward. In *Proc. of ICAPS-PAL 2011*, 2011. To appear.

[9] J. Gratch and G. Dejong. Composer: A probabilistic solution to the utility problem in speed-up learning. In P. Rosenbloom and P. Szolovits, editors, *Proc. of AAAI-92*, pages 235–240. AAAI Press / The MIT Press, Menlo Park, CA, USA, 1992.

[10] P. Groot, A. Birlutiu, and T. Heskes. Bayesian Monte Carlo for the global optimization of expensive functions. In *Proc. of ECAI 2010*, pages 249–254, 2010.

[11] T. Hothorn, B. Lausen, A. Benner, and M Radespiel-Tröger. Bagging survival trees. *Statistics in Medicine*, 23:7791, 2004.

[12] F. Hutter, D. Babić, H. H. Hoos, and A. J. Hu. Boosting verification by automatic tuning of decision procedures. In *Proc. of FMCAD'07*, pages 27–34, Washington, DC, USA, 2007. IEEE Computer Society.

[13] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In *Proc. of CPAIOR-10*, pages 186–202, 2010.

[14] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, 2011. To appear.

[15] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy. Time-bounded sequential parameter optimization. In *Proc. of LION-4*, LNCS. Springer Verlag, 2010.

[16] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, October 2009.

[17] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[18] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Minimizing conflicts: A heuristic repair method for constraint-satisfaction and scheduling problems. *AIJ*, 58(1):161–205, 1992.

[19] J. Schmee and G. J. Hahn. A simple method for regression analysis with censored data. *Technometrics*, 21(4):417–432, 1979.

[20] M. R. Segal. Regression trees for censored data. *Biometrics*, 44(1):35–47, March 1988.

[21] B. J. Williams, T. J. Santner, and W. I. Notz. Sequential design of computer experiments to minimize integrated response functions. *Statistica Sinica*, 10:1133–1152, 2000.