Real-World Bayesian Optimization with A/B Tests

Benjamin Letham, Qing Feng, Samuel Daulton, and Eytan Bakshy
Facebook

April 6, 2021

Abstract

Many internet systems have parameters that are tuned by running field experiments, also known as A/B tests. This setting provides both unique challenges and a diverse set of applications for optimization. We describe the use of Bayesian optimization with A/B tests through five case studies, each describing a particular experiment in which Bayesian optimization was used at Facebook to tune an online system. Through these case studies, we discuss the challenges of optimizing with A/B tests, and show how Bayesian optimization can be successfully adapted to the setting.

Keywords: field experiments, causal inference, surrogate modeling

1 Introduction

Internet-based field experiments, otherwise known as *A/B tests*, are an important part of the product development cycle at internet firms, and are used daily to evaluate product changes and drive improvements. The prototypical A/B test is of a discrete change like a new interface design or an improvement to a machine learning model, and such experiments typically compare one or two test groups to a control group. However, A/B tests are also frequently used to tune more complex software changes. Systems like recommendation system scoring policies and streaming controllers for real-time communication often have a variety of parameters that can be tuned for the systems to perform optimally. Many of the outcomes of interest are measured at the person-level over long periods of time. Examples of outcomes may include usage, interactions with others or with content, or performance metrics like stall rates or loading times. Consequently, the only way to reliably evaluate a particular system configuration is with an A/B test.

Bayesian optimization (BayesOpt) is a popular approach for optimizing black-box functions with time- or resource-consuming evaluations. It has been applied across a wide range of settings including simulation optimization (Santner et al., 2003; Roustant et al., 2012; Picheny et al., 2013), robotics (Lizotte et al., 2007; Calandra et al., 2015; Rai et al., 2018), and automated machine learning (Hutter et al., 2011; Snoek et al., 2012). BayesOpt can also be used to tune online systems that are evaluated with A/B tests (Letham et al., 2019; Letham and Bakshy, 2019), which is the focus of our work here.

This paper presents five case studies, each describing a particular experiment in which BayesOpt techniques were applied to A/B tests at Facebook. The paper does not include any simulated results or toy datasets: all results are from real tests and real online systems. The objective of each case study is to (1) demonstrate how a broad range of BayesOpt techniques are being applied to internet experiments, (2) identify challenges particular to the setting, and (3) describe "lessons learned" on getting the full benefit of BayesOpt in A/B tests.

Sec. 2 provides background on BayesOpt and A/B testing. Sec. 3 contains the case studies, beginning with a fetch tuning for the Android app that initially led us towards BayesOpt (3.1), followed by video playback policy optimization (3.2), combining heterogeneous treatment effect estimation with BayesOpt (3.3), online-offline optimization in News Feed ranking (3.4), and Pareto optimization in a video recommender system (3.5).

2 Background

2.1 Bayesian Optimization

BayesOpt is a form of sequential model-based optimization in which a surrogate model is fit to the current set of function evaluations and used to identify which parts of the parameter

space should be evaluated next. The surrogate model is typically a Gaussian process (GP) $f \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot))$ with prior mean $\mu(\cdot)$ and kernel $k(\cdot,\cdot)$, whose hyperparameters are fit to the data (Rasmussen and Williams, 2006). The GP has several properties that make it an excellent choice for BayesOpt: the posterior predictive distribution is available in closed form, and it produces well-calibrated uncertainty estimates that are useful for decision-making under uncertainty. Once the model has been fit, an acquisition function $\alpha(\mathbf{x})$ uses the model posterior to compute the value of evaluating the function at $\mathbf{x}$, while balancing exploration (high posterior predictive variance) and exploitation (good posterior predictive mean). We select the point with maximum acquisition value for evaluation, update the model, and repeat the process.

There is a large and growing literature on BayesOpt; see Shahriari et al. (2015) for a review. The vanilla BayesOpt loop typically uses an ARD RBF or Matérn 5/2 kernel in the GP, and common choices for the acquisition function include Expected Improvement (EI, Jones et al., 1998), UCB (Srinivas et al., 2010), and KG (Scott et al., 2011). An advantage of BayesOpt is its adaptability to a wide range of settings. On the modeling side, the GP offers significant flexibility in the kernel specification, which can be used to encode knowledge of a problem's structure or incorporate auxiliary data (as described in Secs. 3.2 and 3.4, respectively). Similarly, flexibility in the choice of the acquisition function allows us to tailor the optimization to the goals of each particular problem, as we show in the case study on Pareto optimization (Sec. 3.5).

2.2 A/B Tests

Online systems are by nature amenable to randomized controlled trials since it is possible to create and deploy many variants of a system with minimal overhead. In a typical A/B test, one or more test variants (treatments) are deployed into a live internet-based service. When individuals access the service, they are randomly assigned to one of the variants, or to control. Individual-level outcomes are logged along with exposure to the treatment, so that aggregate outcomes can be measured and compared across variants. The result is an unbiased estimate of the treatment effect for each measured outcome and for each treatment. The long history and literature on design and analysis of experiments applies to internet experiments (Box et al., 2005; Gerber and Green, 2012), although with many additional considerations specific to the setting (Kohavi et al., 2009; Tang et al., 2010; Kohavi et al., 2012; Bakshy et al., 2014).

The ease of deploying A/B tests belies significant challenges in using A/B tests for the system tuning problems that we highlight in the case studies here. The most significant is the duration: A/B tests usually take around 1–4 weeks to run, in part to accumulate sufficiently many samples, but also to account for changes in the exposed population (more active users are more likely to be exposed early on) and changes in treatment effects over time (Bakshy et al., 2014). Fig. 1 shows an example of the observed population treatment

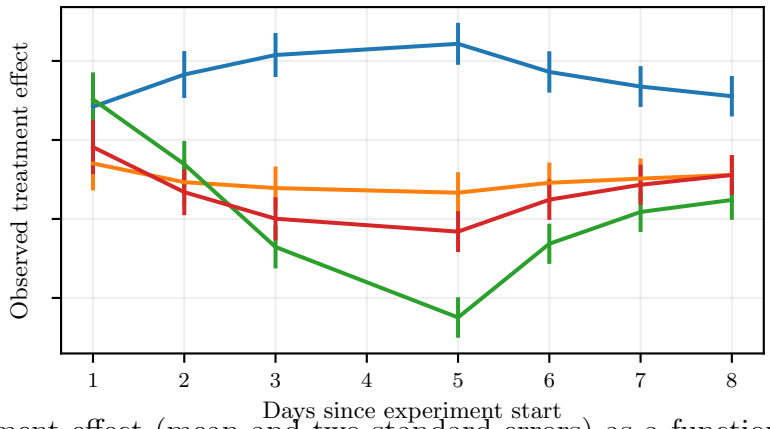

Figure 1: Treatment effect (mean and two standard errors) as a function of time for four treatment groups in an information retrieval system tuning experiment. Nonstationarities in the treatment effect necessitate running A/B tests for 1–4 weeks.

effect over time for four treatment groups that were evaluated in the same A/B test while tuning a retrieval system (see Sec. 3.3). In this experiment, the results after the first day were not even directionally correct, and it took more than a week for the treatment effects to reach a steady state. This is a truly time-consuming function evaluation that eliminates the possibility of more than 2–4 rounds of iteration in a typical optimization. We are, however, usually able to test many treatments in parallel. The pattern for sequential optimization with A/B tests is thus to evaluate configurations in 2–4 batches, each with 10–60 variants[1], a much higher level of parallelization than most BayesOpt applications.

Changes to online systems also often involve complex trade-offs between several outcomes: improvements in computational efficiency may be counterbalanced by increased memory utilization, or one may need to balance different engagement metrics like comments and video views. All of the problems described in the case studies here involved constraints on multiple (2–10) outcomes. Finally, even with millions of samples, noise levels in A/B tests are much higher than what is seen in BayesOpt applications like hyperparameter optimization. With a $\frac{1}{\sqrt{n}}$ relationship between the sample size and the standard error, there are few options for reducing the noise and we must appropriately handle it.

Despite, and in part because of, these challenges, BayesOpt is a good fit for tuning systems that are evaluated with A/B tests. These system tuning problems fall into the usual BayesOpt setting: derivatives cannot be directly measured and function evaluations are, as has been discussed, time-consuming by necessity. A/B test response surfaces are also appropriate for GP modeling since they tend to be smooth. Response surfaces are an average of outcomes over possibly millions of units, and a small change in, *e.g.*, the number of stories fetched (Sec. 3.1), or a video playback controller (Sec. 3.2), will generally produce

[1]The number of variants that can be tested in parallel generally depends on experimentation resource availability, and on the noise levels of the outcomes of interest relative to their effect sizes. The ability to run more variants in parallel permits joint optimization of more parameters.

a small change in the average outcomes. Moreover, being an average of millions of units means the measurements are in the CLT regime where the GP assumption of Gaussian noise is valid. Finally, like many real BayesOpt problems, our experience has been that the alternative to BayesOpt, and often the standard practice prior to our involvement, is expert manual tuning. The move to BayesOpt produces better results, while freeing the experts (in our case usually software engineers) to focus their attention on problems that they find more interesting than tweaking parameters.

3 Lessons Learned from Using BayesOpt with A/B Tests

3.1 From Bandits to BayesOpt

Our work in using BayesOpt for A/B tests originated in an effort to use multi-armed bandits for optimizing data fetching parameters for the Facebook Android app. When a person scrolls through the News Feed, a fetch call is made to pull additional posts as the individual approaches the end of the Feed loaded thus far. The fetch threshold, the number of unseen posts at which the fetch is triggered, is a parameter that must be chosen. A large threshold improves user experience by reducing the risk of stalling as the user approaches the bottom of their Feed, but every fetch requires pulling data to the device, and a large threshold may unnecessarily increase data transfer resources by fetching more posts than will be scrolled to. We wish to select a fetch threshold that optimally balances engagement and data usage (bytes received), among other metrics. Since the fetch threshold is a discrete parameter (an integer), it is a natural fit for a multi-armed bandit framing, and we had success optimizing it with discrete bandit methods.

Tuning a single parameter for all devices can improve the average quality of experience, but the Facebook Android app is used around the world on an incredibly diverse set of devices and network conditions. The engineering team hypothesized that the optimal trade-off between engagement and data usage would depend on network quality. With a poor internet connection, there will generally be both higher mean and variance in the amount of time each fetch takes compared to a fast, stable connection. This means that a poor connection may benefit from a higher fetch threshold in order to maintain lower perceived latency, and the balance between engagement and data usage will be fundamentally different. The team proposed setting a separate fetch threshold for each of five internet connection classes ("unknown", "poor", "moderate", "good", and "excellent") to produce the best user experience across all network types. The range of reasonable fetch threshold values was 3–11, which meant a full factorial design required an infeasibly large ($8^5 = 32,768$) number of treatment groups, each of which would be an arm in a discrete bandit framing. The engineers used their domain expertise to limit the search space to

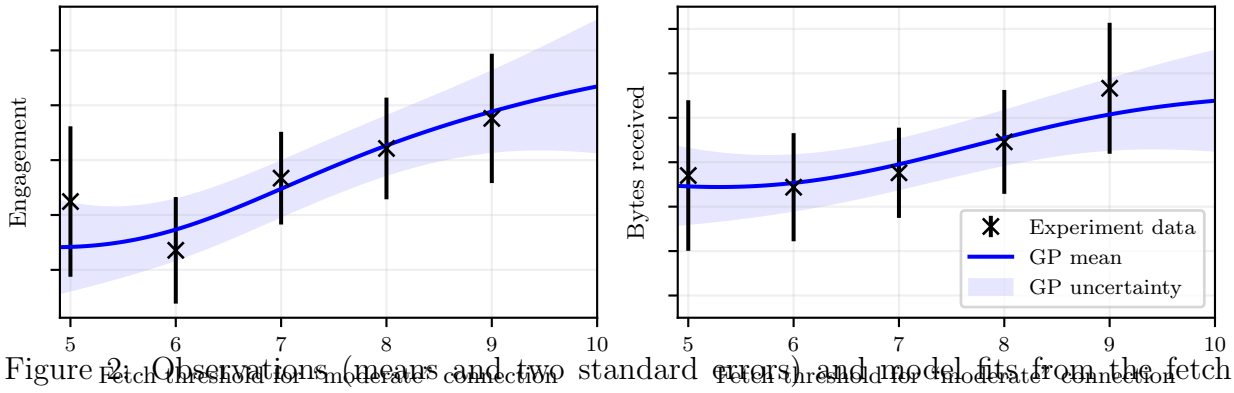

Figure 2: Observations (means and two standard errors) and model fits from the fetch threshold experiment, for a slice of the "moderate" connection quality threshold. Outcomes measured in A/B tests often vary smoothly with system parameters, so surrogate modeling significantly improves over discrete, factorial designs.

combinations they expected to be best, producing a factorial-type design of 32 arms that were deployed in an A/B test.

Fig. 2 shows the experiment outcomes for 5 of those 32 arms, in which only the threshold for the "moderate" connection class was varied[2]. The engagement and performance metrics of interest show a clear, smooth relationship with the fetch threshold. Rather than treating this as an 8^5 factorial space, we can instead consider it as a 5-d continuous space, which is well within the scope of BayesOpt. Fig. 2 shows the predictions on this slice from a GP fit in the 5-d space to all 32 arms. The model is able to effectively borrow strength across observations to make global predictions.

Initial adoption of BayesOpt was not an easy task, both from a social and technical perspective. Teams had difficulty expressing their goals in terms of specific tradeoffs (*e.g.*, as a linear combination of two or more objectives), and so we settled on framing the task as a constrained optimization problem. Acquisition functions like EI also exhibited pathologies that made them less suited for A/B testing, where the low signal-to-noise ratio caused batch EI to oversample particular regions. To address these practical considerations, we used Noisy Expected Improvement (NEI, Letham et al., 2019), a constrained, batch BayesOpt acquisition function well-suited to noisy settings. Rather than search for a design point that exceeds the best observed value f^* (a concept that only really makes sense in the noiseless case), NEI integrates EI over the posterior distribution of the observed noisy function values.

We have found that GP surrogates provide good predictive accuracy across a large number of experimental settings, including those described in the subsequent case studies, and that BayesOpt has been highly effective for exploring and optimizing A/B test response surfaces. In practice, we begin with an initial space-filling design of at least $2d$ datapoints, where d is the number of parameters being tuned. Experiments are ran for the standard

[2]In this slice, the threshold was 10 for "unknown", 9 for "poor", 4 for "good", and 3 for "excellent".

number of days, after which a GP with known noise is fit to the data, and engineers and data scientists may analyze the results and launch subsequent batches of BayesOpt-optimized configurations[3].

3.2 Personalization from Aggregate Measurements

The data fetching experiment just described was an example of a *contextual policy*, in which the parameter value depends on the context in which it is used (in that case, network connection quality). Contextualization allows mobile applications like Facebook to create adaptive systems that maintain a high quality of experience across heterogeneous environments. In this case study, we describe the tuning of a contextual policy for video playback in the Facebook iOS app. During video playback, the video bitrate is determined by an adaptive bitrate (ABR) controller that adjusts the video quality based on playback performance. The controller itself has many continuous parameters that determine how quickly the controller adjusts to changes in playback performance, and how it estimates instantaneous bandwidth. The goal in tuning the ABR controller is to create the best user experience that balances between high video quality and uninterrupted playback.

Similar to the data fetching experiment in Sec 3.1, the optimal ABR controller policy depends on the device's internet connection quality. With a poor connection that has high variance in throughput, the controller may need to adapt differently to fluctuations in bandwidth than with a fast, stable connection, and as before the trade-off between video quality and uninterrupted playback will be different for different connection qualities. With an excellent connection, the ABR controller can push to higher video resolutions while still being less likely to stall.

Our goal was thus to tune a contextual policy in which the values of six parameters were set for each of the same five connection classes in the data fetching experiment. The contextual policy is a map from the five discrete context values to the six continuous parameters. Tuning such a policy is referred to as contextual BayesOpt. In the typical contextual BayesOpt problem, each function evaluation includes an observation of the context, and so multi-task models are used to borrow strength across the discrete context values (Krause and Ong, 2011; Char et al., 2019). A considerable challenge for using BayesOpt for this type of personalization is that when policies are evaluated via A/B tests, it is often difficult, and sometimes impossible, to reliably instrument and log context along with each outcome. In the case of the ABR controller, the contextual policy is sent to the device, and context (internet connection quality) is a device property that may vary throughout the day, or even throughout a particular playback session, and cannot be reliably tied to playback metrics. Instead, only the aggregate outcomes for the entire contextual policy can be measured across a large population that spans the entire distribution of context values. Policy tuning is thus a high-dimensional optimization problem where a map from C context

[3]Our software framework is freely available via our open-source software, `anonymized url`

values to d parameters requires optimizing a black-box function of $C \times d$ parameters; for the ABR controller tuning, this means a dimensionality of $5 \times 6 = 30$.

Standard BayesOpt is known to struggle in high-dimensional settings with more than about 20 parameters. BayesOpt can be adapted to high dimensions by taking advantage of particular problem structure, such as low-dimensional linear structure (Wang et al., 2016; Nayebi et al., 2019; Letham et al., 2020) or additivity (Kandasamy et al., 2015; Wang et al., 2017). In this case, we can take advantage of the fact that each observation is an aggregate over the context distribution to construct an effective surrogate model. Let $\bar{\mathbf{x}} \in \mathbb{R}^{C \times d}$ be the full policy and $\mathbf{x}_c \in \mathbb{R}^d$ be the parameters for context value c. Assuming no interference across context values, the aggregate outcome can be modeled as the weighted average of unobserved outcomes for each context value f_c, *i.e.* $f(\bar{\mathbf{x}}) = \sum_{c=1}^{C} w_c f_c(\mathbf{x}_c)$, where w_c is the population frequency of c.

Feng et al. (2020) study this problem structure and give each latent $f_c(\cdot)$ a GP prior, under which the aggregate $f(\cdot)$ is also a GP for which they propose the LCE-A kernel:

$$k(\bar{\mathbf{x}}, \bar{\mathbf{x}}') = \sum_{c=1}^{C} \sum_{c'=1}^{C} w_c w_{c'} k^z(E(c), E(c')) k^x(\mathbf{x}_c, \mathbf{x}'_c).$$

Here $E(\cdot)$ is an embedding that maps the discrete context values into continuous space, $k^z(\cdot, \cdot)$ is a kernel over the embedding, and $k^x(\cdot, \cdot)$ is the usual kernel over the parameter space. The embedding, the weights w_c, and the hyperparameters for both subkernels are jointly fit to the data.

With BayesOpt, one can flexibly change the GP kernel to the LCE-A and the rest of the optimization procedure remains unchanged. For tuning the iOS ABR controller policy, we began by running a quasi-random initial batch of 54 arms, each being a policy in the 30-dimensional space. We then fit a GP with an LCE-A kernel to each outcome of interest and used cross-validation to confirm that it was an effective surrogate model for the response surfaces (Fig. 3). We used the NEI acquisition function to generate a new batch of 24 optimized points to maximize stall-free watch time, with constraints to maintain low stall rates and high average video quality. We subsequently identified and validated a contextual policy that improved over the existing production policy.

By taking advantage of the unique problem structure in the surrogate model, BayesOpt enables the remarkable feat of tuning personalized policies from purely aggregate outcomes. This is essential in many real-world scenarios where implementing additional logging and metric computation may incur additional engineering and infrastructure costs. In many cases, outcomes at the granularity we would ideally want simply cannot be measured, yet with BayesOpt we can still capture the benefits of contextualization. An additional advantage of using BayesOpt for contextual policy tuning is that the results are highly interpretable, and we have found that our partner teams are very interested in understanding how the policy varies across context values. Fig. 4 shows the surrogate model predictions for the objective metric, stall-free watch time, for two connection qualities ("excellent" and

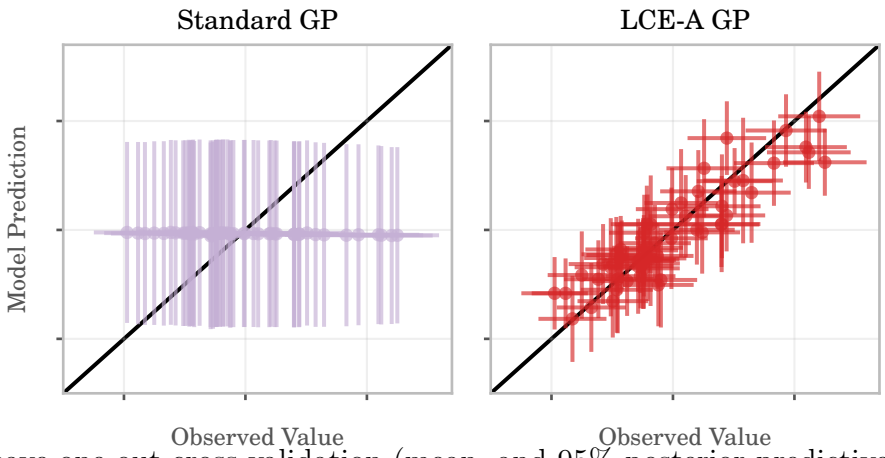

Figure 3: Leave-one-out cross validation (mean, and 95% posterior predictive interval) for standard and contextual GPs fit to the results of the video playback controller experiment with a 30-dimensional policy. The standard GP is unable to learn the high-dimensional response surface, while the contextual model, LCE-A, makes accurate predictions.

"poor"), across a slice of two parameters that control the controller's internal bandwidth estimation[4]. We see that the two connection types have very different surfaces, and different optimal policies. The optimal policy for the "excellent" connection uses a higher cap on the bandwidth estimate that enables it to be more optimistic in increasing the video quality when instantaneous throughput is good.

3.3 Policy Optimization for Heterogeneous Treatment Effects

Personalization of the sort described for the ABR controller requires knowledge of the salient factors affecting the optimal policy. For video playback, domain experts identified network connection quality as a natural context variable, but it is not always clear *a priori* what the most appropriate features for personalization are. The A/B test setting provides a unique opportunity to combine BayesOpt with techniques from causal inference such as heterogeneous treatment effect (HTE) models. HTE models are a powerful tool for understanding and predicting how treatment effects differ across subpopulations and vary with features of each unit of experimentation (Wager and Athey, 2018; Künzel et al., 2019). In this case study, we describe how HTE models can be combined with BayesOpt to personalize the retrieval component of an Instagram feed recommender system.

One of the first stages of a recommender system is the retrieval system, responsible for assembling the collection of items that are subsequently sent to predictive models for scoring and possible inclusion in the ranked list. Content is retrieved by queries which are configured by many parameters, including *sourcing limits* that specify how many items

[4]Note that we are unable to directly observe the response surface for this outcome separately for each connection class; only with the surrogate model can we explore these latent surfaces.

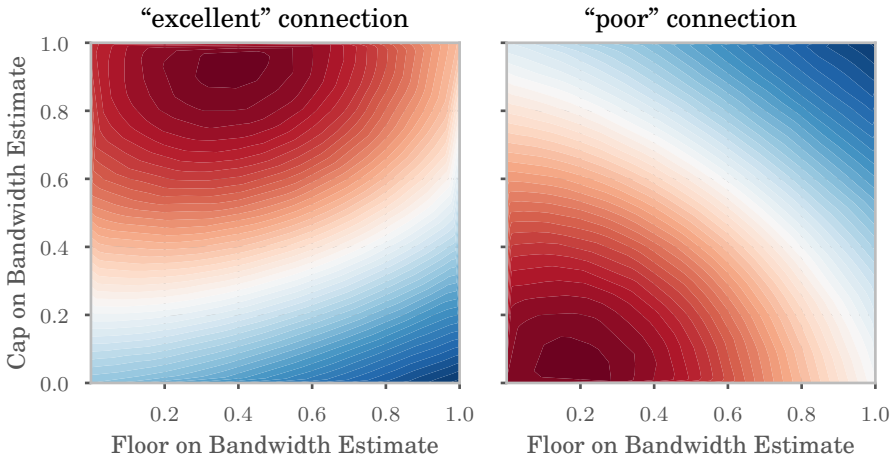

Figure 4: A slice of the GP predictions for two context values and two ABR controller parameters shows different behavior and optimal values for each connection type.

to retrieve from each one of a number of content sources. Retrieving more items incurs infrastructure load, and may lead to unnecessarily ranking items that are unlikely to be relevant. However, some individuals may benefit from the retrieval of more items from a particular source (for instance, because they may simply like the types of content generated by the source). Our goal was to improve the trade-off between infrastructure load and engagement by personalizing a particular sourcing limit parameter.

An initial A/B test that applied a small change to the sourcing limit was used to learn a single-learner HTE model, a binary tree-based model with splits chosen to maximize the sensitivity difference between child nodes (Rzepakowski and Jaroszewicz, 2012; Sołtys et al., 2015), to identify heterogeneity in the treatment effect on engagement. We then used the HTE model as the basis for the personalization: We created 10 clusters corresponding to deciles of the model-predicted treatment effect, and used BayesOpt to tune the sourcing limit value for each. Logging outcomes at the level of each cluster was infeasible, so tuning had to be done using aggregate measurements only. That makes this a 10-dimensional tuning problem, but it has the same contextual structure as in Sec. 3.2, so we used the LCE-A kernel and were able to construct an accurate surrogate model from an initial batch of 33 evaluated configurations (see cross-validation results in Figs. S1 and S2 in the supplement). BayesOpt proceeded with two optimized batches of a total of 26 configurations that were able to improve the objective (engagement), while satisfying constraints on infra load and other outcomes.

Fig. 5 shows the latent GP-predicted response surfaces for four of the context buckets, ordered by sensitivity score from the HTE model. The clusters with higher sensitivity show a faster and larger increase in engagement with sourcing limit than the lower sensitivity clusters, showing that the HTE model was able to effectively identify subpopulations that benefit from personalization. The infra load metric was not part of the HTE model but still

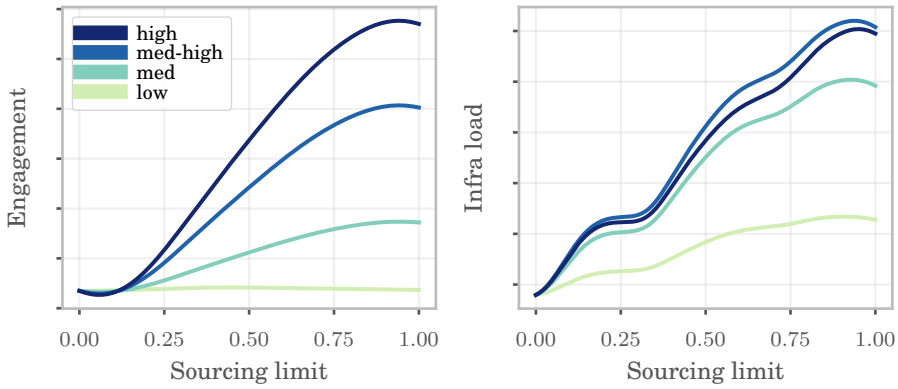

Figure 5: Model-inferred response surfaces for four of the clusters used for sourcing limit personalization, arranged by sensitivity according to the HTE model. Different trade-offs between engagement and infra load for each cluster provide a benefit for personalization.

shows heterogeneity across clusters. The results of this tuning show that BayesOpt nicely complements techniques from causal inference. We can use causal inference to identify sources of heterogeneity and opportunities for personalization, and then apply BayesOpt on top of the causal model to construct the actual personalized policy.

3.4 Simple Offline Estimates Are Useful

The greatest challenge in using A/B tests for tuning online systems is their long duration. Offline simulators, such as those that leverage counterfactual policy evaluation (Bottou et al., 2013), are a much higher-throughput way of evaluating configurations. While off-policy evaluation in reinforcement learning remains a challenging task (Thomas and Brunskill, 2016; Munos et al., 2016), it is possible to construct simple offline simulators for recommender systems that can greatly accelerate the tuning. In this case study we describe combining online (A/B tests) and offline (simulator) experiments for tuning the Facebook News Feed recommendation system.

The News Feed recommendation system is responsible for generating the ordered lists of posts referred to as the Feed. For any given post, machine learning models predict the probability of various engagement events, such as commenting or sharing, and these predictions are used along with attributes of the post (like the strength of the connection between viewer and producer) to order the posts. The score is computed by a component of the system called the *value model*, a collection of rules and weights that encode business logic and effectively parameterize the ranking policy to balance trade-offs between various key metrics. Tuning Feed value model parameters with BayesOpt is challenging because the long duration of the tests, a limited available population, and a large number of parameters make it difficult to test enough configurations to model the response surface.

A simple approach to offline policy estimation for a recommender system is to replay

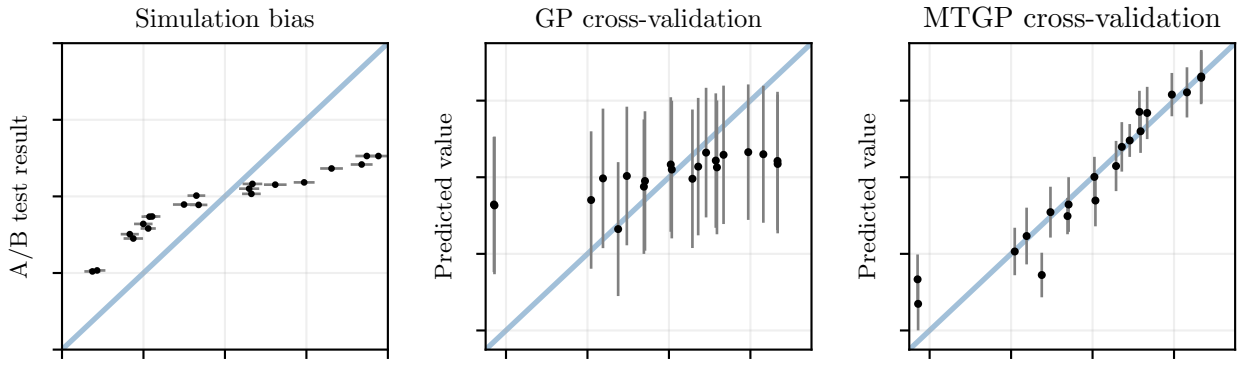

Figure 6: (Left) Observations from 20 configurations that were tested both with A/B tests and on the offline simulator. (Center) Leave-one-out cross-validation for a GP fit to the 20 online observations only. (Right) Cross-validation for an MTGP fit to the same 20 online observations plus 100 offline observations. With multi-task modeling, the biased simulator substantially improved predictions for online outcomes.

sessions to an offline recommender system that has the configuration we wish to test. We obtain a new ranked list for each session, and then apply the event prediction models to each item in the counterfactual ranking. These predicted event probabilities can be aggregated across many replay sessions to produce offline estimates of some of the same outcomes that are measured in an A/B test. This approach to offline simulation is appealing because it relies only on the existing event prediction models, but it also fails to capture many important dynamics of how humans interact with a recommendation system. For instance, an individual's behavior on Facebook can be influenced by complex patterns of interactions with others in past sessions (Eckles et al., 2016). A faithful simulation of human interactions in such a system is not a reasonable target, and so we are left with some amount of irreducible bias.

Here we describe a particular value model tuning in which 10 parameters (weights associated with various story attributes) were tuned to maximize a measure of ranking quality, subject to three constraints on different measures of engagement. We ran an A/B test with 20 value model configurations in the 10-d space, from a space-filling design. We ran these same 20 configurations on the offline simulator, and Fig. 6(Left) compares the A/B test and simulator measurements for one of the constraint metrics. The simulator results were correlated with the online measurements, but too biased to be used as a substitute for A/B tests[5]. However, the online data was also not sufficient for BayesOpt: Fig. 6(Center) shows the GP cross-validation, and the model fails to make good out-of-sample predictions. This is unsurprising, considering it is a 10-d space with only 20 observations.

This problem required taking advantage of the flexibility in GP modeling to enable

[5]If the goal were just to maximize the objective, then the monotonic relationship between online and offline estimates would be sufficient to optimize fully offline. However, there are also constraints that must be satisfied at particular value.

optimization where neither the online nor offline measurements alone were sufficient. Swersky et al. (2013b) show how multi-task GPs (MTGPs) can be used to combine data from different sources for BayesOpt. An MTGP generalizes the kernel function to be across both tasks and the parameter space (Álvarez et al., 2012). In the simplest formulation, the intrinsic coregionalization model, the covariance for parameter-task pairs $(\mathbf{x}, t)$ and $(\mathbf{x}', t')$ is decomposed as $k((\mathbf{x}, t), (\mathbf{x}', t')) = \mathbf{B}_{t,t'} k^x(\mathbf{x}, \mathbf{x}')$, where $k^x(\cdot, \cdot)$ is the usual kernel over the parameter space, and $\mathbf{B}$ is a task covariance matrix that is fit to the data. The MTGP is able to borrow strength across tasks, in our case the "online" and "offline" tasks. Fig. 6(Right) shows that when the online and offline data were combined into a multi-task model, the model made accurate out-of-sample predictions for online outcomes.

With an accurate surrogate model in the MTGP, we were able to proceed with BayesOpt as usual. Fig. S3 in the supplement shows the outcomes at each iteration of optimization. We ran four batches of online experiments, in which a total of 46 configurations were tested. Augmenting these were 300 offline simulations that were incorporated into the MTGP. In the quasi-random initial batch, most of the configurations violated at least one constraint and none of them improved on the existing, manually-tuned production configuration. The subsequent three batches of BayesOpt were able to find configurations that satisfied the constraints and significantly improved over the manually-tuned status quo.

Creating a perfect simulator in this setting is impossible, but these results show how with surrogate modeling and BayesOpt we do not need the simulator to be perfect in order for it to be useful. Letham and Bakshy (2019) show that across a large number of value model tuning experiments, incorporating simple, offline estimates into the model via a multi-task kernel significantly reduces prediction error and enables more efficient optimization.

3.5 Considering Trade-offs Between Multiple Competing Metrics

In each of the experiments reviewed above, there were multiple competing metrics of interest. In Sec. 3.1, fetching parameters were tuned to find an optimal balance between engagement and data usage; in Sec. 3.2, the goal was to balance video quality and stall time; and in Sec. 3.3 we sought to balance engagement and infrastructure load. Often, outcome constraints are placed on all but one objective metric (Schonlau et al., 1998), but constraints do not account for the benefit of *improving* the value of the constrained outcome. For example, in the constrained optimization problem "maximize engagement without increasing infrastructure load," if two solutions yield the same increase in engagement, the solution with the lower load should be preferred. A natural alternative is to scalarize the metrics into a single objective, but specifying the scalarization weights requires knowing exactly the preferred relative trade-offs, and even then can lead to extreme trade-offs depending on the topography of the response surface.

To address the limitation of these approaches, it is often of interest to simultaneously optimize multiple competing metrics. This multi-objective optimization problem involves

optimizing a vector-valued objective function. Typically, there is no single best solution, but rather a *Pareto* set of optimal metric trade-offs such that improving one metric means degrading another. The Pareto frontier enables decision-makers to fully understand the trade-offs between the metrics and make more informed launch decisions.

A common measure of the quality of the Pareto frontier is the hypervolume that is dominated by the Pareto frontier and bounded from below by a reference point. Maximizing the hypervolume has been shown to produce high-quality Pareto sets (Zitzler et al., 2003; Couckuyt et al., 2014; Yang et al., 2019). Rather than the goal of the optimization being to maximize a particular function, the goal in this multi-objective framing is to maximize the hypervolume of the Pareto frontier. This can be accomplished with BayesOpt by taking advantage of the flexibility in the acquisition function. We use the usual GP surrogate model to estimate the response surface for each outcome, and then use an acquisition function that selects points specifically to increase the hypervolume.

The hypervolume improvement (HVI) from a new point is defined as the hypervolume dominated exclusively by that new point. A natural acquisition function for maximizing the hypervolume is the *expected hypervolume improvement* (EHVI) (Emmerich et al., 2006), where the expectation is over the modeled posterior distribution at that point. However, EHVI only supports selecting configurations one at a time, whereas in the A/B test setting, as we have discussed, it is necessary to test large batches in parallel.

Recently, Daulton et al. (2020) proposed a variant of EHVI called qEHVI which assigns an acquisition value to a set of q candidate points $\mathcal{X}_{\text{cand}} = \{\mathbf{x}_i\}_{i=1}^{q}$ equal to their expected *joint* hypervolume improvement, which is the hypervolume dominated by at least one of the q new candidates and not dominated by the current Pareto frontier. The expectation over the GP posterior can be efficiently computed using Monte-Carlo integration with samples $\{\boldsymbol{f}_t\}_{t=1}^{N}$ drawn from the GP posterior:

$$\alpha_{q\text{EHVI}}(\mathcal{X}_{\text{cand}}) = \mathbb{E}\Big[\text{HVI}(\boldsymbol{f}(\mathcal{X}_{\text{cand}}))\Big] \approx \frac{1}{N}\sum_{t=1}^{N} \text{HVI}(\boldsymbol{f}_t(\mathcal{X}_{\text{cand}})). \tag{1}$$

In this case study we describe tuning a retrieval system similar to that described in Sec. 3.3, except for a different, video recommender system. The team was interested in improving the trade-off between engagement and infrastructure load, but did not know for this particular system what trade-offs could be achieved, and so were unable to specify their desired trade-off in a single-objective formulation. We approached the problem from a multi-objective optimization lens with the goal of identifying the Pareto frontier between engagement and infra load resulting from tuning nine sourcing parameters. We first tested a batch with 29 space-filling configurations and fit GPs for each outcome. Measuring hypervolume requires specifying a reference point, for which the team specified an upper bound on infra load and a lower bound on engagement outside which configurations would not be considered useful. We then used qEHVI to generate a second batch of 19 configurations that were evaluated in A/B tests. Fig. 7 shows the Pareto fronts after the initial batch and

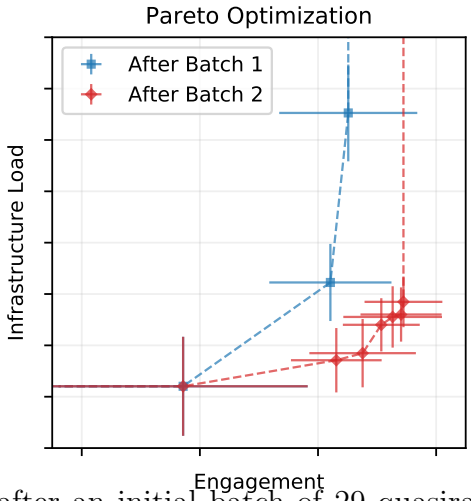

Figure 7: Pareto frontiers after an initial batch of 29 quasirandom configurations, and a follow-up batch of 19 configurations selected via multi-objective BayesOpt with qEHVI. The goal was lower infrastructure load and higher engagement (bottom right direction). BayesOpt enables understanding and improving trade-offs between competing metrics.

after the qEHVI-optimized batch. BayesOpt was able to significantly improve the Pareto frontier and find trade-offs that were better for both metrics.

Every A/B test involves trade-offs between competing metrics, and in many cases teams do not know what trade-off they want until they understand what trade-offs are achievable. Multi-objective BayesOpt is able to find the best outcome for every trade-off, and can thus improve that important understanding at the same time as improving the experiment outcomes.

4 Discussion

The case studies in this paper illustrate the breadth of the application of BayesOpt with A/B tests, both in the systems (core app parameters, a video playback controller, and various aspects of recommendation systems) and in methodology (additive kernels, multi-task models, and multi-objective acquisition functions). We highlighted some of the useful tasks that BayesOpt makes possible: efficiently exploring factorial spaces, personalizing with only aggregate measurements, constructing policies to make use of heterogeneous treatment effect models, getting value from simple and biased simulators, and understanding complex metric trade-offs. These tasks are prevalent in internet systems, and advances in BayesOpt in these areas can drive improvements in system performance.

Beyond what was described here, there are other active areas of work in BayesOpt that we expect will be particularly useful in the A/B test setting. Unbounded BayesOpt (Shahriari et al., 2016; Ha et al., 2019) can be helpful for settings where the domain

experts do not know in advance what appropriate bounds are for the parameter values, and need to avoid the possibility of large negative changes. Some systems have hierarchical and conditional search spaces (Swersky et al., 2013a; Jenatton et al., 2017) and advances in surrogate modeling and optimization in those settings will be valuable. Finally, non-myopic optimization (González et al., 2016; Jiang et al., 2020) could be especially useful with the large batch sizes and high parallelization of A/B tests.

A/B tests and internet systems tuning are a rich application area for driving advances in methodology. It is also a driver of high-quality software: robustness, performance, and modularity are critical, which needs motivated the development of our [anonymized for review] package for BayesOpt. Building methodological advances into open-source tooling will enable the techniques and insights gained from using BayesOpt with A/B tests to have impact across the many disciplines in which BayesOpt is useful.

SUPPLEMENTARY MATERIAL

Supplemental results figures: Additional results to support the case studies. (pdf)

References

Álvarez, M. A., Rosasco, L., and Lawrence, N. D. (2012), "Kernels for Vector-Valued Functions: A Review," *Foundations and Trends in Machine Learning*, 4, 195–266.

Bakshy, E., Eckles, D., and Bernstein, M. S. (2014), "Designing and Deploying Online Field Experiments," in *Proceedings of the 23rd International Conference on World Wide Web*, WWW.

Bottou, L., Peters, J., Quiñonero-Candela, J., Charles, D. X., Chickering, D. M., Portugaly, E., Ray, D., Simard, P., and Snelson, E. (2013), "Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising," *Journal of Machine Learning Research*, 14, 3207–3260.

Box, G. E. P., Hunter, J. S., and Hunter, W. G. (2005), *Statistics for Experimenters: Design, Innovation, and Discovery*, Wiley.

Calandra, R., Seyfarth, A., Peters, J., and Deisenroth, M. P. (2015), "Bayesian Optimization for Learning Gaits under Uncertainty," *Annals of Mathematics and Artificial Intelligence*, 76, 5–23.

Char, I., Chung, Y., Neiswanger, W., Kandasamy, K., Nelson, A. O., Boyer, M., Kolemen, E., and Schneider, J. (2019), "Offline Contextual Bayesian Optimization," in *Advances in Neural Information Processing Systems 32*, NeurIPS.

Couckuyt, I., Deschrijver, D., and Dhaene, T. (2014), "Fast Calculation of Multiobjective Probability of Improvement and Expected Improvement Criteria for Pareto Optimization," *Journal of Global Optimization*, 60, 575–594.

Daulton, S., Balandat, M., and Bakshy, E. (2020), "Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, NeurIPS.

Eckles, D., Kizilcec, R. F., and Bakshy, E. (2016), "Estimating Peer Effects in Networks with Peer Encouragement Designs," *Proceedings of the National Academy of Sciences*, 113, 7316–7322.

Emmerich, M. T. M., Giannakoglou, K. C., and Naujoks, B. (2006), "Single- and Multiobjective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels," *IEEE Transactions on Evolutionary Computation*, 10, 421–439.

Feng, Q., Letham, B., Mao, H., and Bakshy, E. (2020), "High-Dimensional Contextual Policy Search with Unknown Context Rewards using Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, NeurIPS.

Gerber, A. S. and Green, D. P. (2012), *Field Experiments: Design, Analysis, and Interpretation*, WW Norton.

González, J., Osborne, M., and Lawrence, N. D. (2016), "GLASSES: Relieving the Myopia of Bayesian Optimisation," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, AISTATS.

Ha, H., Rana, S., Gupta, S., Nguyen, T., Tran-The, H., and Venkatesh, S. (2019), "Bayesian Optimization with Unknown Search Space," in *Advances in Neural Information Processing Systems 32*, NeurIPS.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011), "Sequential Model-Based Optimization for General Algorithm Configuration," in *International Conference on Learning and Intelligent Optimization*, LION.

Jenatton, R., Archambeau, C., González, J., and Seeger, M. (2017), "Bayesian Optimization with Tree-Structured Dependencies," in *Proceedings of the 34th International Conference on Machine Learning*, ICML.

Jiang, S., Jiang, D. R., Balandat, M., Karrer, B., Gardner, J., and Garnett, R. (2020), "Efficient Nonmyopic Bayesian Optimization via One-Shot Multi-Step trees," in *Advances in Neural Information Processing Systems 33*, NeurIPS.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998), "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, 13, 455–492.

Kandasamy, K., Schneider, J., and Póczos, B. (2015), "High Dimensional Bayesian Optimisation and Bandits via Additive Models," in *Proceedings of the 32nd International Conference on Machine Learning*, ICML.

Kohavi, R., Deng, A., Frasca, B., Longbotham, R., Walker, T., and Xu, Y. (2012), "Trustworthy Online Controlled Experiments: Five Puzzling Outcomes Explained," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD.

Kohavi, R., Longbotham, R., Sommerfield, D., and Henne, R. M. (2009), "Controlled Experiments on the Web: Survey and Practical Guide," *Data Mining and Knowledge Discovery*, 18, 140–181.

Krause, A. and Ong, C. S. (2011), "Contextual Gaussian Process Bandit Optimization," in *Advances in Neural Information Processing Systems 24*, NIPS.

Künzel, S. R., Sekhon, J. S., Bickel, P. J., and Yu, B. (2019), "Metalearners for Estimating Heterogeneous Treatment Effects using Machine Learning," *Proceedings of the National Academy of Sciences*, 116, 4156–4165.

Letham, B. and Bakshy, E. (2019), "Bayesian Optimization for Policy Search via Online-Offline Experimentation," *Journal of Machine Learning Research*, 20, 1–30.

Letham, B., Calandra, R., Rai, A., and Bakshy, E. (2020), "Re-Examining Linear Embeddings for High-Dimensional Bayesian Optimization," in *Advances in Neural Information Processing Systems 33*, NeurIPS.

Letham, B., Karrer, B., Ottoni, G., and Bakshy, E. (2019), "Constrained Bayesian Optimization with Noisy Experiments," *Bayesian Analysis*, 14, 495–519.

Lizotte, D. J., Wang, T., Bowling, M., and Schuurmans, D. (2007), "Automatic Gait Optimization with Gaussian Process Regression," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. (2016), "Safe and Efficient Off-Policy Reinforcement Learning," *Advances in Neural Information Processing Systems 29*, 1054–1062.

Nayebi, A., Munteanu, A., and Poloczek, M. (2019), "A Framework for Bayesian Optimization in Embedded Subspaces," in *Proceedings of the 36th International Conference on Machine Learning*, ICML.

Picheny, V., Ginsbourger, D., Richet, Y., and Caplin, G. (2013), "Quantile-Based Optimization of Noisy Computer Experiments with Tunable Precision," *Technometrics*, 55, 2–13.

Rai, A., Antonova, R., Song, S., Martin, W., Geyer, H., and Atkeson, C. G. (2018), "Bayesian Optimization Using Domain Knowledge on the ATRIAS Biped," in *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA.

Rasmussen, C. E. and Williams, C. K. I. (2006), *Gaussian Processes for Machine Learning*, Cambridge, Massachusetts: The MIT Press.

Roustant, O., Ginsbourger, D., and Deville, Y. (2012), "DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization," *Journal of Statistical Software*, 51, 1–55.

Rzepakowski, P. and Jaroszewicz, S. (2012), "Decision Trees for Uplift Modeling with Single and Multiple treatments," *Knowledge and Information Systems*, 32, 303–327.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003), *The Design and Analysis of Computer Experiments*, Springer.

Schonlau, M., Welch, W. J., and Jones, D. R. (1998), "Global versus Local Search in Constrained Optimization of Computer Models," *Lecture Notes—Monograph Series*, 34, 11–25.

Scott, W., Frazier, P., and Powell, W. (2011), "The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression," *SIAM Journal of Optimization*, 21, 996–1026.

Shahriari, B., Bouchard-Côté, A., and de Freitas, N. (2016), "Unbounded Bayesian Optimization via Regularization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, AISTATS.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. (2015), "Taking the Human out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, 104, 148–175.

Snoek, J., Larochelle, H., and Adams, R. P. (2012), "Practical Bayesian Optimization of Machine Learning Algorithms," in *Advances in Neural Information Processing Systems 25*, NIPS.

Sołtys, M., Jaroszewicz, S., and Rzepakowski, P. (2015), "Ensemble Methods for Uplift Modeling," *Data Mining and Knowledge Discovery*, 29, 1531–1559.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. (2010), "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design," in *Proceedings of the 27th International Conference on Machine Learning*, ICML.

Swersky, K., Duvenaud, D., Snoek, J., Hutter, F., and Osborne, M. A. (2013a), "Raiders of the Lost Architecture: Kernels for Bayesian Optimization in Conditional Parameter Spaces," in *NIPS 2013 workshop on Bayesian Optimization*.

Swersky, K., Snoek, J., and Adams, R. P. (2013b), "Multi-Task Bayesian Optimization," in *Advances in Neural Information Processing Systems 26*, NIPS.

Tang, D., Agarwal, A., O'Brien, D., and Meyer, M. (2010), "Overlapping Experiment Infrastructure: More, Better, Faster Experimentation," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD.

Thomas, P. and Brunskill, E. (2016), "Data-Efficient Off-Policy Policy Evaluation for Reinforcement Learning," in *Proceedings of the 33rd International Conference on Machine Learning*, ICML.

Wager, S. and Athey, S. (2018), "Estimation and Inference of Heterogeneous Treatment Effects using Random Forests," *Journal of the American Statistical Association*, 113, 1228–1242.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and de Freitas, N. (2016), "Bayesian Optimization in a Billion Dimensions via Random Embeddings," *Journal of Artificial Intelligence Research*, 55, 361–387.

Wang, Z., Li, C., Jegelka, S., and Kohli, P. (2017), "Batched High-Dimensional Bayesian Optimization via Structural Kernel Learning," in *Proceedings of the 34th International Conference on Machine Learning*, ICML.

Yang, K., Emmerich, M., Deutz, A., and Bäck, T. (2019), "Multi-Objective Bayesian Global Optimization Using Expected Hypervolume Improvement Gradient," *Swarm and Evolutionary Computation*, 44, 945–956.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., and da Fonseca, V. G. (2003), "Performance Assessment of Multiobjective Optimizers: an Analysis and Review," *IEEE Transactions on Evolutionary Computation*, 7, 117–132.

Supplemental Materials: Real-World Bayesian Optimization with A/B Tests

This supplement contains several extra figures to enhance the case studies in the main text.

S1 Sourcing Personalization

Here we provide the model cross-validation for the sourcing personalization experiment of Sec. 3.3 in the main text. Fig. S1 shows leave-one-out cross-validation results for the infrastructure load outcome, and Fig. S2 has the corresponding results for the engagement outcome.

Both figures include cross-validation for both a standard GP (ARD Matérn 5/2 kernel) and a GP with the LCE-A kernel that takes advantage of the known contextual structure. A standard GP had high predictive uncertainty and generally failed to make useful predictions. The LCE-A GP was able to make good out-of-sample predictions for both outcomes, and so could be used for BayesOpt.

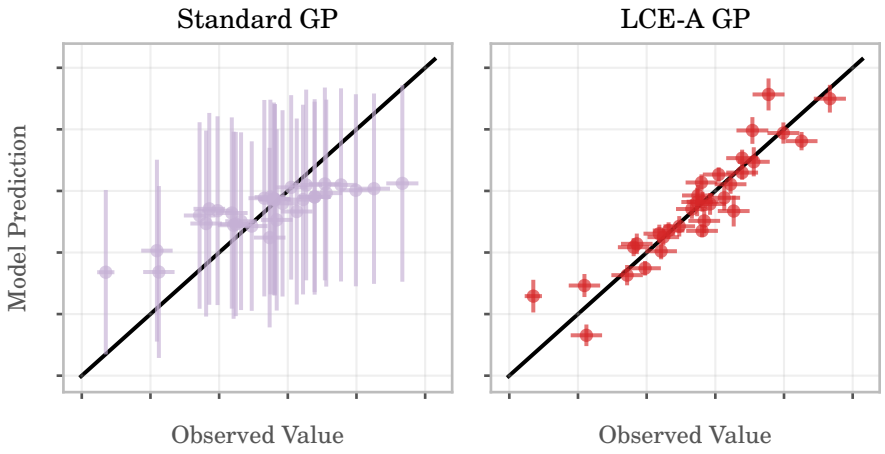

Figure S1: Leave-one-out cross validation for the infra load outcome of the experiment in Sec. 3.3.

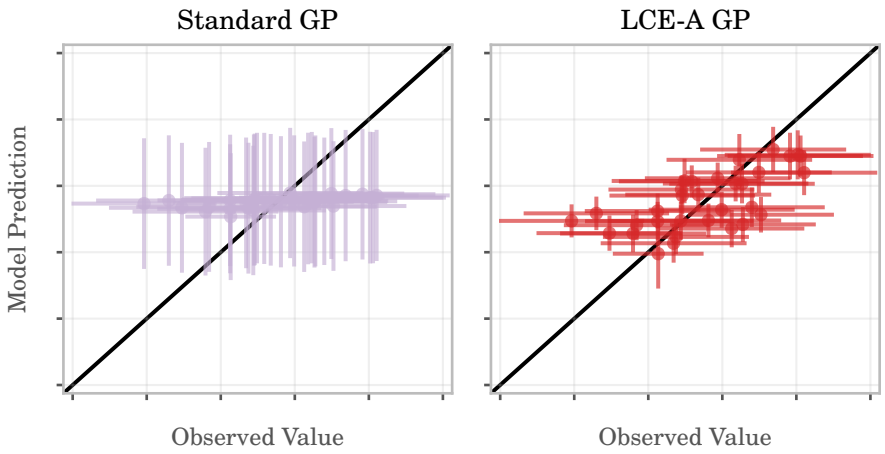

Figure S2: Leave-one-out cross validation for the engagement outcome of the experiment in Sec. 3.3.

S2 Online-Offline Optimization

Here we show the outcomes for each online iteration of Bayesian Optimization for the online-offline value model tuning of Sec. 3.4. There were four batches tested online: an initial quasirandom batch of 20 configurations followed by three optimized batches of 10, 8, and 8 configurations respectively. Prior to generating each optimized batch, the model was updated with the results of 100 offline simulator points, which were chosen as a mix of quasirandom points and points with high acquisition value.

Fig. S3 shows the outcome for each configuration for the objective and for each of the three constraints. The constraints were difficult to satisfy in the quasirandom batch, and most of the feasible points had very poor objective values. Subsequent optimized batches were able to take advantage of the surrogate model to focus on parts of the space that were much more likely to be feasible, while simultaneously improving the objective.

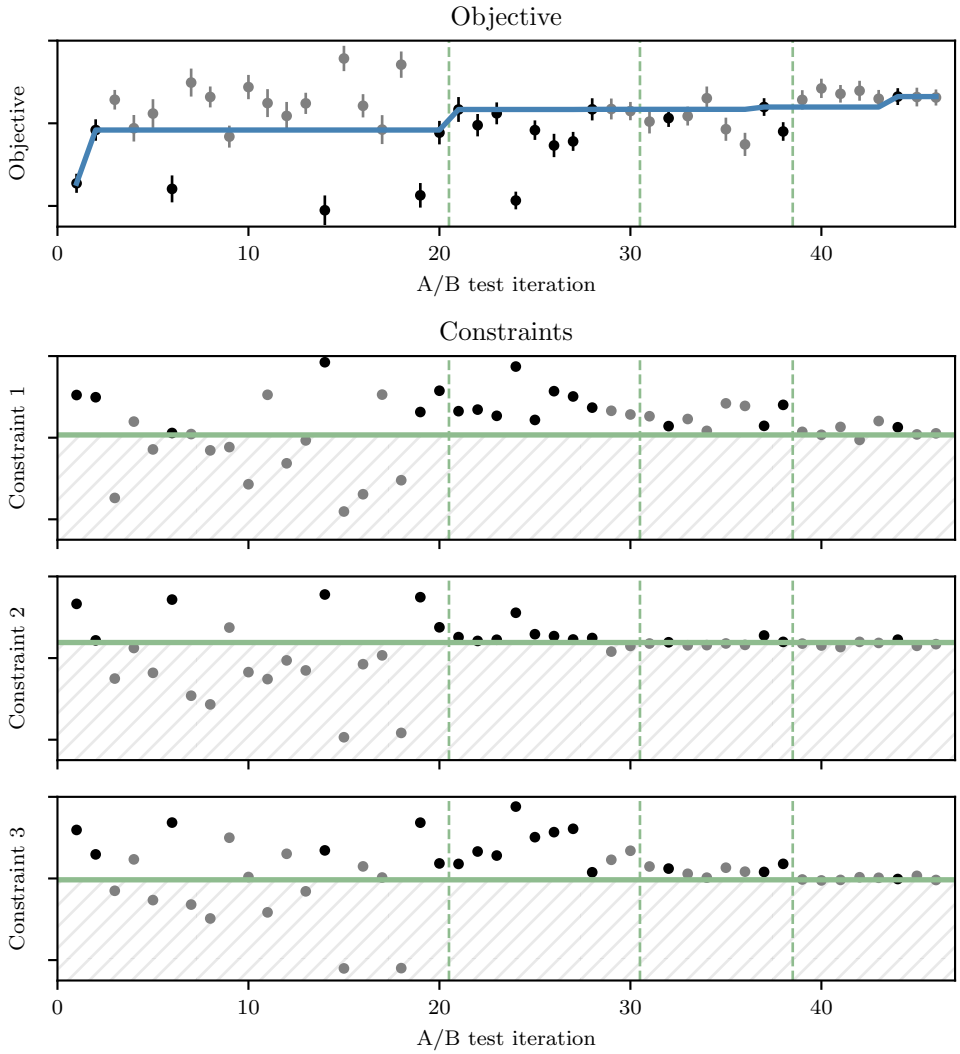

Figure S3: Optimization results for the experiment of Sec. 3.4. Vertical lines denote the four batches of points. Horizontal lines for constraints indicate the lower bound, with the shaded region below being infeasible. Grayed out points were infeasible due to violating at least one constraint. Line in the objective plot shows the best-feasible point by each iteration. BayesOpt significantly improved the feasibility of the points, and was able to improve the best-feasible objective across each of the three optimized batches.