OPT2019: 11th Annual Workshop on Optimization for Machine Learning

Scaling parallel Gaussian process optimization with adaptive batching and resparsification

Daniele Calandriello DANIELE.CALANDRIELLO@IIT.IT
LCSL - Istituto Italiano di Tecnologia, Genova, Italy

Luigi Carratino LUIGI.CARRATINO@DIBRIS.UNIGE.IT
LCSL - University of Genova, Genova, Italy

Alessandro Lazaric LAZARIC@FB.COM
FAIR - Facebook AI Research, Paris, France

Michal Valko VALKOM@DEEPMIND.COM
Deepmind, Paris, France

Lorenzo Rosasco LROSASCO@MIT.EDU
LCSL - University of Genova &, IIT, Genova, Italy & MIT, Cambridge, USA

Abstract

Gaussian processes (GP) are one of the most successful frameworks to model uncertainty. However, classical GP optimization (e.g., GP-UCB) suffers from major scalability limitations. Selecting each query point has a computational cost at least quadratic in the number of dimensions and iterations, which quickly becomes unfeasible. Moreover, to reduce experimental time costs queries must be selected in batches (e.g., using GP-BUCB) and evaluated in parallel.

In this paper, we introduce BBKB (Batch Budgeted Kernel Bandit), the first sparse GP-UCB approximation that provably runs in constant per-step amortized time and achieves the same regret as GP-UCB. Moreover, it is capable of batching queries, while greatly reducing the computational complexity compared to GP-BUCB. This is obtained by adaptively delaying costly updates to the sparse approximation of the GP, in combination with a novel bound for the evolution of the posterior variances that allows us to accurately choose when a resparsification is necessary. Moreover, the same bound can be used to decide how much to delay feedback, allowing us to choose larger batches compared to GP-BUCB. Finally, we show experimentally that BBKB is much faster than existing methods.

1. Introduction

Gaussian process optimization (GP-Opt) is a principled way to optimize a black-box function from noisy evaluations (i.e., sometimes referred to as *bandit* feedback). Due to the presence of noise, the optimization process is modeled as a sequential learning problem, where the goal is learning the optimum of the function with the least number of evaluations. This can be formalized using a sequential regret minimization protocol where at each step t: 1) the learner must choose an alternative $\mathbf{x}_t$ out of a finite decision set $\mathcal{A} = \{\mathbf{x}_i\}_{i=1}^{A}$ of alternatives (also called arms) in $\mathbb{R}^d$; 2) the learner receives a reward $y_t \triangleq f(\mathbf{x}_t) + \eta_t$, where f is the unknown function we wish to optimize and ε_t is an additive noise drawn i.i.d. from $\mathcal{N}(0, \xi^2)$; and 3) it must use this feedback to guide its subsequent actions. The performance of the learner is measured by its regret $R_T \triangleq \sum_{t=1}^{T} f(\mathbf{x}_\star) - f(\mathbf{x}_t)$, where $\mathbf{x}_\star \in \arg\max_{x \in \mathcal{A}} f(x)$ is the best arm. While there many GP-Opt algorithms with strong theoretical guarantees and proven empirical effectiveness, none of them simultaneously guarantee: **G1** Sublinear

© D. Calandriello, L. Carratino, A. Lazaric, M. Valko & L. Rosasco.

regret, which implies that $R_T/T \leq \mathcal{O}(1)$ and that the optimization algorithm converges; **G2** Near-linear $\widetilde{\mathcal{O}}(T)$ amortized time and space complexity, necessary to scale to complex problems; **G3** Maintaining both G1 and G2 when feedback is provided in batches and not sequentially.

There are many GP-Opt algorithms that guarantees sub-linear regret, known as no-regret algorithms [5, 14, 16], out of which the most well understood is the GP-UCB family [5, 7, 16]. However, most of them are poorly scalable due to two possible bottleneck: poor computational scalability, where most of the time is spent choosing the candidate to evaluate, and poor experimental scalability, where most of the time is spent waiting for experimental feedback. The first can be addressed using efficient approximations of GP-UCB [4, 11], while the second can be tackled by selecting query points in batches rather than sequentially [7, 8, 10], which allows us to run experiments in the batch in parallel and save time. Finally, note that we must not forget to avoid losing our regret guarantees when striving for scalability: selecting queries in batches delays feedback and can lead to sub-optimal arm choices, and GP approximations lead to similar problems.

To solve these problems we introduce the Batched Budgeted Kernelized Bandit (BBKB), the first GP-Opt algorithm that truly achieves near-linear runtime while incurring the same near-optimal regret rate of GP-UCB. Moreover, our algorithm can operate both in a sequential and batch setting, which allows it to avoid experimental bottlenecks and achieve practical applicability.

2. Batch Budgeted Kernelized Bandits

In this section we introduce BBKB and postpone a theoretical analysis to the next section.

Nyström embeddings and approximate GP posterior. Gaussian processes are usually defined in terms of a prior mean function μ, which we will assume to be zero, and a prior covariance defined by the kernel function $k : \mathcal{A} \times \mathcal{A} \to [0, \kappa^2]$. To introduce notation necessary in the rest of the paper, we will now express the GP posterior using a less common but more general formulation based on *inducing points* [9, 12], also known as sparse GP approximations. Given a so-called dictionary of inducing points $\mathcal{S} \triangleq \{\mathbf{x}_i\}_{i=1}^m$, let $\mathbf{K}_\mathcal{S} \in \mathbb{R}^{t\times t}$ be the kernel matrix constructed using the evaluations $k(\mathbf{x}_i, \mathbf{x}_j)$ for all the points in $\mathcal{S}$, and similarly let $\mathbf{k}_\mathcal{S}(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), \dots, k(\mathbf{x}_m, \mathbf{x})]^\intercal$. Then we define a Nyström embedding as $\mathbf{z}(\cdot, \mathcal{S}) \triangleq \mathbf{K}_\mathcal{S}^{+/2}\mathbf{k}_\mathcal{S}(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$, where $(\cdot)^{+/2}$ indicates the square root of the pseudo-inverse. We further denote the matrix with all actions selected so far as $\mathbf{X}_t \triangleq [\mathbf{x}_1, \dots, \mathbf{x}_t]^\intercal \in \mathbb{R}^{t\times d}$ with corresponding rewards $\mathbf{y}_t \triangleq [y_1, \dots, y_t]^\intercal$, and associated matrices $\mathbf{Z}(\mathbf{X}_t, \mathcal{S}) = [\mathbf{z}(\mathbf{x}_1, \mathcal{S}), \dots, \mathbf{z}(\mathbf{x}_t, \mathcal{S})]^\intercal \in \mathbb{R}^{t\times m}$ and $\mathbf{V}_t = \mathbf{Z}(\mathbf{X}_t, \mathcal{S})^\intercal\mathbf{Z}(\mathbf{X}_t, \mathcal{S}) + \lambda\mathbf{I} \in \mathbb{R}^{m\times m}$. Reformulating from [4, Eq. 4], the approximate posterior mean and variance[1] of an action $\mathbf{x}_i$ are

$$\widetilde{\mu}_t(\mathbf{x}_i, \mathcal{S}) = \mathbf{z}(\mathbf{x}_i, \mathcal{S})^\intercal\mathbf{V}_t^{-1}\mathbf{Z}(\mathbf{x}_i, \mathcal{S})^\intercal\mathbf{y}_t, \tag{1}$$

$$\widetilde{\sigma}_t^2(\mathbf{x}_i, \mathcal{S}) = \tfrac{1}{\lambda}(k(\mathbf{x}_i, \mathbf{x}_i) - \mathbf{z}(\mathbf{x}_i, \mathcal{S})^\intercal\mathbf{z}(\mathbf{x}_i, \mathcal{S}) + \mathbf{z}(\mathbf{x}_i, \mathcal{S})^\intercal\mathbf{V}_t^{-1}\mathbf{z}(\mathbf{x}_i, \mathcal{S}), \tag{2}$$

where λ is a parameter to be tuned. Note that the t subscript in the $\widetilde{\mu}_t$ and $\widetilde{\sigma}_t$ notation indicates what we already observed (i.e. $\mathbf{X}_t$ and $\mathbf{y}_t$), and $\mathcal{S}$ indicates the dictionary used for the embedding. For example, if we set $\mathcal{S}_{\text{exact}} = \mathbf{X}_t$, i.e. keep in the dictionary all arms selected so far, then we can write the usual posterior mean and covariance of a GP as $\mu_t(\mathbf{x}) \triangleq \widetilde{\mu}_t(\mathbf{x}, \mathcal{S}_{\text{exact}})$ and $\sigma_t(\mathbf{x}) \triangleq \widetilde{\sigma}_t(\mathbf{x}, \mathcal{S}_{\text{exact}})$.

The GP-UCB family. There are several ways to leverage a GP posterior to choose promising arms to evaluate. The GP-UCB algorithm [16] uses $u_t(\mathbf{x}) = \mu_t(\mathbf{x}) + \beta_t\sigma_t(\mathbf{x})$ as an upper confidence bounds (UCB) for $f(\mathbf{x})$ and sequentially chooses $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{A}} u_t(\mathbf{x})$. However this is

1. This is also known as the Bayesian DTC approximation [12] and as the BKB approximation [4].

computationally and experimentally slow, as evaluating $u_t(\mathbf{x})$ requires $\mathcal{O}(t^2)$ per-step and no parallel experiments are possible. To improve computations, BKB [4] replaces u_t with an approximate $\widetilde{u}_t(\mathbf{x}) = \widetilde{\mu}_t(\mathbf{x}, \mathcal{S}_t) + \widetilde{\beta}_t \widetilde{\sigma}_t(\mathbf{x}, \mathcal{S}_t)$, which they prove is sufficiently close to u_t to achieve low regret. However, maintaining accuracy requires $\mathcal{O}(t)$ per step to update the dictionary $\mathcal{S}_t$ at each iteration, and the queries are still selected sequentially. GP-BUCB [7] tries to increase GP-UCB's experimental efficiency by selecting a batch of queries that are all evaluated in parallel. To relate individual arm choices with their batch we use a vector $\texttt{fb}[] \in \mathbb{R}^T$ such that $\texttt{fb}[t]$ contains the index of the last step of the batch, i.e. at step t we have access only to feedback $\mathbf{y}_{\texttt{fb}[t]}$ up to step $\texttt{fb}[t]$. Then GP-BUCB approximate the UCB as $\widetilde{u}_t = \mu_{\texttt{fb}[t]}(\mathbf{x}) + \beta_t \sigma_t(\mathbf{x})$, where the mean is not updated until new feedback arrives, while due to its definition the variance only depends on $\mathbf{X}_t$ and can be updated in an unsupervised manner. Nonetheless, GP-BUCB is as computationally slow as GP-UCB.

The BBKB algorithm. The pseudocode of BBKB is presented in Algorithm 1. The dictionary and approximate posterior are initialized by choosing uniformly at random an initial arm $\mathbf{x}_1$, and then the optimization loop start. At the beginning of each iteration BBKB chooses the next arm to query $\mathbf{x}_{t+1}$ as the $\arg\max$ on a vector $\widetilde{\mathbf{u}}_t$ of pre-computed upper confidence bounds for each arm. The UCB $\widetilde{u}_t(\mathbf{x}) = \widetilde{\mu}_{\texttt{fb}[t]}(\mathbf{x}, \mathcal{S}_{\texttt{fb}[t]}) + \widetilde{\alpha}_{\texttt{fb}[t]} \widetilde{\sigma}_t(\mathbf{x}, \mathcal{S}_{\texttt{fb}[t]})$ is a combination of BKB and GP-BUCB's approaches, with a new element. Not only we delay feedback updates to not be forced to update the posterior mean, but compared to BKB we also delay dictionary updates, and continue to use the same dictionary $\mathcal{S}_{\texttt{fb}[t]}$ for all iterations in a batch. Fixing the dictionary has important computational consequences, which we will soon discuss. However, delaying feedback and dictionary updates can result in poor UCB approximation, and therefore after selecting $\mathbf{x}_{t+1}$ we run a check (L5) to decide whether to continue the batch, which we call a normal step, or not, which we call a resparsification step. While this approach of adaptive batch termination has been introduced in GP-UCB, the specific formulation of our condition in L5 is crucial to guarantee near-linear runtime, and improves over the terminating condition of GP-BUCB. If we continue with a normal step, the embedding $\mathbf{z}(\cdot, \mathcal{S}_{\texttt{fb}[t]})$ does not change so we can pre-embed arms for computational efficiency, and use efficient rank-one update formulas to update the posterior variance. Most importantly, for a fixed $\mathcal{S}_{\texttt{fb}[t]}$ $\widetilde{\sigma}_{t+1}(\mathbf{x}, \mathcal{S}_{\texttt{fb}[t]}) \leq \widetilde{\sigma}_t(\mathbf{x}, \mathcal{S}_{\texttt{fb}[t]})$, and since $\widetilde{\mu}_{\texttt{fb}[t]}(\mathbf{x}, \mathcal{S}_{\texttt{fb}[t]})$ and $\widetilde{\alpha}_{\texttt{fb}[t]}$ remain fixed in-batch, the UCBs $\widetilde{u}_t(\mathbf{x}, \mathcal{S}_{\texttt{fb}[t]})$ are strictly decreasing in-batch. Therefore, after selecting $\mathbf{x}_{t+1}$ we only need to recompute $\widetilde{u}_{t+1}(\mathbf{x}_{t+1}, \mathcal{S}_{\texttt{fb}[t]})$ and the UCBs for arms that had $\widetilde{u}_t(\mathbf{x}_i, \mathcal{S}_{\texttt{fb}[t]}) \geq \widetilde{u}_{t+1}(\mathbf{x}_{t+1}, \mathcal{S}_{\texttt{fb}[t]})$ to guarantee that we are still selecting the $\arg\max$ correctly. Although it is only an implementation detail, this lazy UCB evaluation brings important practical speedups. If instead our condition on L5 determines that the ap-

Algorithm 1 BBKB

Require: Arm set $\mathcal{A}$, $\{\widetilde{\alpha}_{\texttt{fb}[t]}\}_{t=1}^T$, T
1: Sample $\mathbf{x}_1$ uniformly, receive $\mathbf{y}_1$
2: Initialize $\mathcal{S}_1 = \{\mathbf{x}_1\}$, $\texttt{fb}[1] = 1$
3: **for** $t = \{1, \dots, T-1\}$ **do**
4: Select $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}_i \in \mathcal{A}} \widetilde{\mathbf{u}}_t(\mathbf{x}_i, \mathcal{S}_{\texttt{fb}[t]})$
5: **if** $1 + \sum_{s=\texttt{fb}[t]+1}^{t+1} \widetilde{\sigma}_{\texttt{fb}[t]}(\mathbf{x}_s, \mathcal{S}_{\texttt{fb}[t]}) \leq \widetilde{C}$ **then**
6: $\texttt{fb}[t+1] = \texttt{fb}[t]$, i.e. $\mathcal{S}_{\texttt{fb}[t+1]} = \mathcal{S}_{\texttt{fb}[t]}$
7: Update $\widetilde{\mathbf{u}}_{t+1}(\mathbf{x}_t, \mathcal{S}_{\texttt{fb}[t+1]})$ with the new $\widetilde{\sigma}_{t+1}$
8: Update $\widetilde{\mathbf{u}}_{t+1}(\mathbf{x}_i, \mathcal{S}_{\texttt{fb}[t+1]})$ for all $\{\mathbf{x}_i : \widetilde{u}_t(\mathbf{x}_i, \mathcal{S}_{\texttt{fb}[t]}) \geq \widetilde{u}_{t+1}(\mathbf{x}_{t+1}, \mathcal{S}_{\texttt{fb}[t]})\}$
9: **else**
10: $\texttt{fb}[t+1] = t+1$
11: Initialize $\mathcal{S}_{\texttt{fb}[t+1]} = \emptyset$
12: **for** $\mathbf{x}_s \in \mathbf{X}_{\texttt{fb}[t+1]}$ **do**
13: Set $\widetilde{p}_{\texttt{fb}[t+1],s} = \overline{q} \cdot \widetilde{\sigma}^2_{\texttt{fb}[t]}(\mathbf{x}_s)$
14: Draw $z_{\texttt{fb}[t+1],s} \sim Bernoulli(\widetilde{p}_{\texttt{fb}[t+1],s})$
15: If $z_{\texttt{fb}[t+1],s} = 1$, add $\mathbf{x}_s$ in $\mathcal{S}_{\texttt{fb}[t+1]}$
16: **end for**
17: Get feedbacks $\{y_s\}_{s=\texttt{fb}[t]+1}^{\texttt{fb}[t+1]}$
18: Update $\widetilde{\mu}_{\texttt{fb}[t+1]}$ and $\widetilde{\sigma}_{\texttt{fb}[t+1]}$ for all $\mathbf{x}_i \in \mathcal{A}$ with $\mathbf{y}_{\texttt{fb}[t+1]}$ and $\mathcal{S}_{\texttt{fb}[t+1]}$
19: **end if**
20: **end for**

proximation error might be growing too much
it is time to update the sparse GP approximation, i.e. to resparsify. To do this we employ BKB's posterior sampling procedure in L11-16 (see [4] for more details), and then fully update both the posterior and UCBs for all arms since changing the dictionary makes the lazy updates invalid. Note that BBKB extends and generalizes both BKB and GP-BUCB. If instead of performing the check in L5 we update dictionary and feedback at each step BBKB is equivalent to BKB, up to an improved $\widetilde{\alpha}_t$ as we will see in the next section. If $\mathcal{S}_t = \mathbf{X}_t$ we recover GP-BUCB, but with an improved terminating rule for batches.

3. Computational and Regret Analysis

To quantify the complexity of the optimization process we will denote with $d_{\text{eff}}(\mathbf{X}_t) \triangleq \sum_{s=1}^{t} \sigma_t(\mathbf{x}_s)$ the so-called *effective dimension* of the set of arms selected so far. Intuitively, if captures the effective number of parameters in f, i.e. f can be represented using roughly $d_{\text{eff}}(\mathbf{X}_t)$ weights. Moreover, it is connected to the maximal mutual information gain γ_T [16], and $\widetilde{\mathcal{O}}(\gamma_T) \approxeq \widetilde{\mathcal{O}}(d_{\text{eff}}(\mathbf{X}_T))$ [2, Lem. 1].

Computational analysis. BBKB's complexity can be easily computed (see Appendix B) based on the size of the dictionary/embedding $m_{\texttt{fb}[t]} = |\mathcal{S}_{\texttt{fb}[t]}|$, and by knowing the number B of resparsification steps since they are more expensive than normal steps. However, m_t and B are random quantities, and to guarantee that BBKB is scalable we must prove that they are small w.h.p.

Theorem 1 *Given $\delta \in (0,1)$, $1 \leq \widetilde{C}$, and $1 \leq \lambda$, run* BBKB *with $\overline{q} \geq 8\log(4T/\delta)$. Then, w.p. $1-\delta$*

1) For all $t \in [T]$ we have $|\mathcal{S}_t| \leq 9\widetilde{C}(1+\kappa^2/\lambda)\overline{q}d_{\mathit{eff}}(\mathbf{X}_t)$.

2) Moreover, the total number of resparsification performed by BBKB *is at most $\mathcal{O}(d_{\mathit{eff}}(\mathbf{X}_t))$.*

3) As a consequence, BBKB *runs in at most $\widetilde{\mathcal{O}}(TAd_{\mathit{eff}}(\mathbf{X}_t)^2)$ near-linear time.*

By summing the well understood complexity of the basic linear algebra primitives used to implement BBKB, we can show that it requires $\mathcal{O}(\sum_t Am_t^2 + \max_t B(Am_t^2 + m_t^3 + tm_t))$ time and $\mathcal{O}(\max_t tm_t + m_t^2)$ space. Combining this with the bounds provided by part (1) and (2) of Theorem 1, we obtain part (3), i.e. the proof that BBKB runs provably in near-linear time. Among sequential GP-Opt algorithms, BBKB is not only much faster than the original GP-UCB $\widetilde{\mathcal{O}}(AT^3)$ runtime, but also much faster when compared to BKB's quadratic $\widetilde{\mathcal{O}}(T\max\{A,T\}d_{\text{eff}}^2)$. BBKB's runtime also improves over GP-Opt algorithm that are specialized for stationary kernels (e.g. Gaussian), such as QFF-TS [11] $\widetilde{\mathcal{O}}(TA2^d d_{\text{eff}}^2)$ runtime, without making any assumption on the kernel and without an exponential dependencies on the input dimension d. When compared to batch algorithms, such as GP-BUCB, the improvement is even sharper as all existing batch GP-Opt algorithms that are provably no-regret [6, 7, 15] also share GP-UCB's $\widetilde{\mathcal{O}}(AT^3)$ runtime. One of the central elements of this result is BBKB's adaptive batch terminating condition. As a comparison, GP-BUCB [7] adaptively terminates a batch based on the product $\prod_{s=\texttt{fb}[t]+1}^{t+1}(1+\sigma_{\texttt{fb}[t]}(\mathbf{x}_s)$, which always give smaller batches than BBKB's due to Weierstrass product inequality $1 + \sum_{s=\texttt{fb}[t]+1}^{t+1}\sigma_{\texttt{fb}[t]}(\mathbf{x}_s) \leq \prod_{s=\texttt{fb}[t]+1}^{t+1}(1+\sigma_{\texttt{fb}[t]}(\mathbf{x}_s)$. Finally, note that the A factor reported in the runtime is pessimistic, since BBKB recomputes only a small fraction of UCB's at each step thanks to lazy evaluations.

Regret analysis. To provide regret bounds for BBKB we must guarantee that our $\widetilde{\mathbf{u}}_t = \widetilde{\mu}_{\texttt{fb}[t]} + \widetilde{\alpha}_t\widetilde{\sigma}_t$ is a valid upper confidence bound. This is true at the beginning of a batch, since we have all

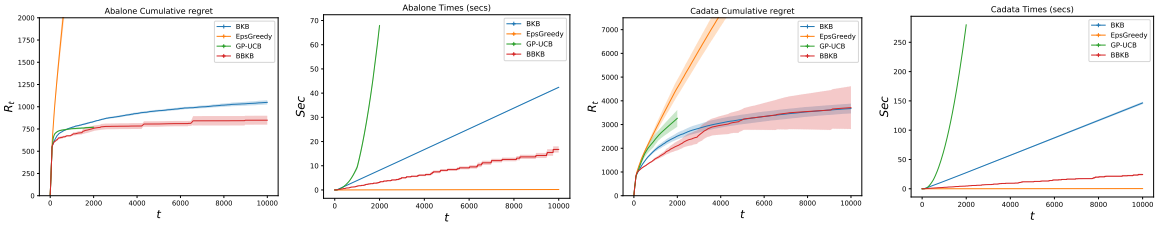

Fig. 1: Comulative regret and time on Abalone and Cadata datasets

feedback and just performed a resparsification, which makes BBKB's UCB equal to that of BKB and therefore valid [4]. This can change during a batch where, due to both the lack of feedback and delay in resparsification, $\widetilde{\mu}_{\mathtt{fb}[t]}$ remains fixed while $\widetilde{\sigma}_t^2$ keeps decreasing, potentially becoming smaller than $f(\mathbf{x}_*)$ and losing its UCB property. The key insight to solve this is to prove that, thanks to the adaptive batch termination rule, even in the worst case $\widetilde{\sigma}_t^2$ is still larger than $\widetilde{\sigma}_{\mathtt{fb}[t]}^2/\widetilde{C}$. Then, inspired by [7], it is sufficient to rescale all $\widetilde{\sigma}_t^2$ by $\widetilde{C}$ to make them a valid UCB, balancing not only the feedback delay but also the resparsification delay introduced by BBKB.

Theorem 2 *Assume $\|f\|_{\mathcal{H}} \leq F < \infty$. For any desired, $0 < \delta < 1$, $1 \leq \lambda$, $1 \leq \widetilde{C}$, if we run* BBKB *with $\overline{q} \geq 72\widetilde{C}\log(4T/\delta)$, $\widetilde{\alpha}_{\mathtt{fb}[t]} = \widetilde{C}\widetilde{\beta}_{\mathtt{fb}[t]}$, and*

$$\widetilde{\beta}_{\mathtt{fb}[t]} = 2\xi\sqrt{\textstyle\sum_{s=1}^{\mathtt{fb}[t]} \log\left(1 + 3\widetilde{\sigma}^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s)\right) + \log\left(1/\delta\right)} + \left(1+\sqrt{2}\right)\sqrt{\lambda}F,$$

then, with prob. $1-\delta$, BBKB*'s regret is bounded as*

$$R_T^{\text{BBKB}} \leq 55\widetilde{C}^2\,\overbrace{\widetilde{C}\,\sqrt{T}\left(\xi\textstyle\sum_{t=1}^{T}\sigma^2_{t-1}(\mathbf{x}_t) + \xi\log(1/\delta) + \sqrt{\lambda F^2\textstyle\sum_{t=1}^{T}\sigma^2_{t-1}(\mathbf{x}_t)}\right)}^{R_T^{\text{GP-UCB}}}.$$

Since $\widetilde{C}$ is a small constant chosen by the user, Theorem 2 shows that BBKB achieves essentially the same regret as GP-BUCB and GP-UCB, but at a fraction of the computational cost. Note that $\sum_{t=1}^{T}\sigma^2_{t-1}(\mathbf{x}_t) \approx d_{\text{eff}} \approx \gamma_T$ [16, Lem. 5.4], and therefore any bound on d_{eff} or γ_T can be applied also to BBKB. (e.g. $\gamma_T \leq \mathcal{O}(\log(T)^{d+1})$ for the Gaussian kernel [16]). Such a tight bound is achieved thanks in part to a new confidence interval radius $\widetilde{\beta}_t$. In particular Calandriello et al. [4] contains an extra $\log\det(\mathbf{K}_T/\lambda + \mathbf{I}) \leq d_{\text{eff}}(\lambda, \mathbf{X}_T)\log(T)$ bounding step that we do not have to make. While in the worst case this is only a $\log(T)$ improvement, empirically the data adaptive bound seems to lead to much better regret.

4. Experiments

For each experimental result we report mean and std over 10 repetitions. The experiments are run on a single core. We compare BBKB on the Abalone ($A = 4177$, $d = 8$) and Cadata ($A = 20640$, $d = 8$) datasets. BBKB uses a Gaussian kernel with width $\sigma^2 = 5$, $F = 20$, $\delta = 1/T$, $\overline{q} = 2$, $\lambda = 0.2$ on a horizon of $T = 10^4$ iterations. In Figure 1 we compare BBKB with EpsGreedy, GP-UCB and BKB (with the same parameters) in terms of regret R_t and time (secs). We can see how BBKB is able to achieve sub-linear regret for both datasets in a fraction of the time of the competitors. Moreover the time is measured without parallelization during the batches of BBKB, which grows up to 3700 and 3900 in size for Abalone and Cadata respectively (see Figure 2 in Appendix), giving BBKB a potential further speedup.

ACKNOWLEDGMENTS

This material is based upon work supported by the Center for Brains, Minds and Machines (CBMM), funded by NSF STC award CCF-1231216, and the Italian Institute of Technology. We gratefully acknowledge the support of NVIDIA Corporation for the donation of the Titan Xp GPUs and the Tesla k40 GPU used for this research. L. R. acknowledges the financial support of the AFOSR projects FA9550-17-1-0390 and BAA-AFRL-AFOSR-2016-0007 (European Office of Aerospace Research and Development), and the EU H2020-MSCA-RISE project NoMADS - DLV-777826.

References

[1] Ahmed El Alaoui and Michael W. Mahoney. Fast randomized kernel methods with statistical guarantees. In *Neural Information Processing Systems*, 2015.

[2] Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Second-order kernel online convex optimization with adaptive sketching. In *International Conference on Machine Learning*, 2017.

[3] Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Distributed adaptive sampling for kernel matrix approximation. In *AISTATS*, 2017.

[4] Daniele Calandriello, Luigi Carratino, Alessandro Lazaric, Michal Valko, and Lorenzo Rosasco. Gaussian process optimization with adaptive sketching: Scalable and no regret. In *Conference on Learning Theory*, 2019.

[5] Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *International Conference on Machine Learning*, pages 844–853, 2017.

[6] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[7] Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014.

[8] Philipp Hennig and Christian J. Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[9] Jonathan H Huggins, Trevor Campbell, Mikołaj Kasprzak, and Tamara Broderick. Scalable gaussian process inference with finite-data mean and variance guarantees. *International Conference on Artificial Intelligence and Statistics*, 2019.

[10] Tarun Kathuria, Amit Deshpande, and Pushmeet Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

[11] Mojmir Mutny and Andreas Krause. Efficient High Dimensional Bayesian Optimization with Additivity and Quadrature Fourier Features. In S. Bengio, H. Wallach, H. Larochelle,

K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9019–9030. Curran Associates, Inc., 2018.

[12] Joaquin Quinonero-Candela, Carl Edward Rasmussen, and Christopher KI Williams. Approximation methods for gaussian process regression. *Large-scale kernel machines*, pages 203–224, 2007.

[13] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. ISBN 978-0-262-18253-9. OCLC: ocm61285753.

[14] Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. In *Advances in Neural Information Processing Systems*, pages 1583–1591, 2014.

[15] Amar Shah and Zoubin Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, pages 3330–3338, 2015.

[16] Niranjan Srinivas, Andreas Krause, Matthias Seeger, and Sham M Kakade. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, pages 1015–1022, 2010.

[17] David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

Appendix A. Preliminary results

We present in Appendix A.1 a sketch of the proof to guide the reader to the more detailed version in Appendix A.2

A.1. Sketch of the Proof

Compared to GP-UCB, to provide guarantees for BBKB we must control several sources of error, some novel: not only we have delayed feedback due to batching as in [7], and approximation error due to the dictionary as in [4], but we also delay resparsification updates to $\mathcal{S}_t$ to the end of a each batch. Moreover, we have to guarantee that the size of the dictionary and the overall number of resparsification remains small, in order to guarantee a near-linear amortized runtime.

The first tool used in our analysis connects $\sigma_{\mathtt{fb}[t]}$ to the approximation $\widetilde{\sigma}_{\mathtt{fb}[t]}$ at the beginning of each batch, extending [4, Thm. 1] to our setting with delayed resparsifications.

Lemma 3 *Under the same conditions as Theorem 1, w.p.* $1-\delta$, $\forall\ \mathtt{fb}[t] \in [T]$ *and* $\forall\ \mathbf{x} \in \mathcal{A}$ *we have* $\sigma^2_{\mathtt{fb}[t]}(\mathbf{x})/3 \leq \widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S}_{\mathtt{fb}[t]}) \leq 3\sigma^2_{\mathtt{fb}[t]}(\mathbf{x})$.

Controlling accuracy at the beginning of the batch however is not enough, and the stochastic argument used for Theorem 3 break down when the dictionary is not updated. Instead, we can use this novel *deterministic* relationship that holds even in a worst case scenario.

Lemma 4 *For any kernel* k*, dictionary* $\mathcal{S}$*, set of points* $\mathbf{X}_t$*,* $\mathbf{x}_i \in \mathcal{A}$*, and* $\mathtt{fb}[t] < t$*,*

$$\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S})/(1 + \textstyle\sum_{s=\mathtt{fb}[t]}^{t} \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S})) \leq \widetilde{\sigma}_t(\mathbf{x}, \mathcal{S}) \leq \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S}).$$

Following Theorem 4, BBKB terminates each batch when $1 + \sum_{s=\mathtt{fb}[t]}^{t} \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S}_{\mathtt{fb}[t]}) \leq \widetilde{C}$, and therefore its estimates $\widetilde{\sigma}_t$ remain accurate during the batch up to a constant $\widetilde{C}$. Combining Theorems 3 and 4, and the terminating condition we can also control the exact posterior w.h.p.

Lemma 5 *Under the same conditions as Theorems 3 and 4,* $\sigma_{\mathtt{fb}[t]}(\mathbf{x})/(3\widetilde{C}) \leq \sigma_t(\mathbf{x}) \leq \sigma_{\mathtt{fb}[t]}(\mathbf{x})$.

Combined, these results allow us to accurately compare approximate and exact posteriors, both at the beginning and during each batch. We leverage them to prove that BBKB is scalable and no-regret.

A.2. Proof of Preliminary Results

In this section we collect most results related to providing guarantees that exact and approximate posteriors remain close during the whole optimization process.

Several results presented in this appendix are easier to express and prove using the so-called feature-space view of a GP [13]. In particular, to every covariance $k(\cdot,\cdot)$ and reproducing kernel Hilbert space $\mathcal{H}$ we can associate a *feature map* $\boldsymbol{\phi}(\cdot)$ such that $k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal \boldsymbol{\phi}(\mathbf{x}_j)$, and that $k(\mathbf{x}_i, \mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal \boldsymbol{\phi}(\mathbf{x}_i) = \|\boldsymbol{\phi}(\mathbf{x}_i)\|^2$. Let $\boldsymbol{\Phi}(\mathbf{X}_t) = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_t)]^\intercal$ be the map where each row corresponds to a row of the matrix $\mathbf{X}_t$ after the application of $\boldsymbol{\phi}(\cdot)$. Finally, given operator $\mathbf{A}$, we use $\|\mathbf{A}\|$ to indicate its ℓ_2 operator norm, also known as sup norm. For symmetric positive semi-definite matrices, this corresponds simply to its largest eigenvalue.

Using the feature-space view of a GP we can introduce an important reformulation of the posterior variance $\sigma_t^2(\mathbf{x}_i)$

$$\sigma_t^2(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal(\boldsymbol{\Phi}(\mathbf{X}_t)^\intercal\boldsymbol{\Phi}(\mathbf{X}_t) + \lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{x}_i).$$

In particular, this quadratic form is well known in randomized numerical linear algebra as ridge leverage scores (RLS) [1], and used extensively in linear sketching algorithms [17]. Therefore, some of the results we will present now are inspired from this parallel literature. For example the proof of Theorem 3, restated here for convenience, is based on concentration results for RLS sampling.

Lemma 6 *Under the same conditions as Theorem 1, w.p.* $1-\delta$, $\forall\ \mathtt{fb}[t] \in [T]$ *and* $\forall\ \mathbf{x} \in \mathcal{A}$ *we have* $\sigma^2_{\mathtt{fb}[t]}(\mathbf{x})/3 \leq \widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S}_{\mathtt{fb}[t]}) \leq 3\sigma^2_{\mathtt{fb}[t]}(\mathbf{x})$.

Proof We briefly show here that we can apply Calandriello et al. [4, Thm. 1]'s result from sequential RLS sampling to our batch setting. In particular, [4, Thm. 1] gives identical guarantees as Theorem 3, but only when the dictionary is resparsified at each step, and we must compensate for the delays.

At a high level, their result shows that given a so-called (ε, λ)-accurate dictionary $\mathcal{S}_t$ it is possible to sample a (ε, λ)-accurate dictionary $\mathcal{S}_{t+1}$ using the posterior variance estimator $\widetilde{\sigma}_t(\mathbf{x}, \mathcal{S}_t)$ from Equation 2. Since all other guarantees directly follow from (ε, λ)-accuracy, we only need to show that the same inductive argument holds if we apply it on a batch-by-batch basis instead of a step-by-step basis. To simplify, we will also only consider the case $\varepsilon = 1/2$. For more details, we refer the reader to the whole proof in [4, App. C].

In particular, consider the state of the algorithm at the beginning of the first batch, i.e. just after initialization ended. Since the subset $\mathcal{S}_1 = \mathbf{X}_1$ includes all arms pulled so far (i.e. $\mathbf{x}_1$) it clearly perfectly preserves $\mathbf{X}_1$, and is therefore infinitely accurate and also $(1/2, \lambda)$-accurate. Note that Calandriello et al. [4] make the same reasoning for their base case.

Thereafter, assume that $\mathcal{S}_{\mathtt{fb}[t]}$ is $(1/2, \lambda)$-accurate, and let $t' > \mathtt{fb}[t]$ be the time step where we resparsify the dictionary (i.e. the beginning of the following batch) such that $\mathtt{fb}[t'-1] = \mathtt{fb}[t]$ and $\mathtt{fb}[t'] = t'$. To guarantee that $\mathcal{S}_{\mathtt{fb}[t']}$ is also $(1/2, \lambda)$-accurate we must guarantee that the probabilities $\widetilde{p}_{\mathtt{fb}[t']}$ used to sample are at least as large as the true posterior $\sigma^2_{\mathtt{fb}[t']}$ scaled by a factor $24\log(4T/\delta)$, i.e. $\widetilde{p}_{\mathtt{fb}[t']} \geq (24\log(4T/\delta)) \cdot \sigma_{\mathtt{fb}[t']}$. From the inductive assumption we know that $\mathcal{S}_{\mathtt{fb}[t]}$ is $(1/2, \lambda)$-accurate, and therefore Theorem 3 holds and $\widetilde{\sigma}_{\mathtt{fb}[t]} \geq \sigma_{\mathtt{fb}[t]}/3 \geq \sigma^2_{\mathtt{fb}[t']}/3$, since it is a well known property of RLS and σ_t that they are non-increasing in t [3]. Adjusting $\overline{q}$ to match this condition, we guarantee that we are sampling at least as much as required by Calandriello et al. [4], and therefore achieve the same accuracy guarantees. ■

Before moving on to Theorem 4 and Theorem 5, we will first consider exact posterior variances $\sigma_t^2(\mathbf{x}_i)$, which represent a simpler case since we do not have to worry about the presence of a dictionary. The following Lemma will form a blueprint for the derivation of Theorem 4.

Lemma 7 *For any kernel* k*, set of points* $\mathbf{X}_t$*,* $\mathbf{x}_i \in \mathcal{A}$*, and* $\mathtt{fb}[t] < t$*,*

$$\sigma_t^2(\mathbf{x}_i) \leq \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i) \leq \left(1 + \textstyle\sum_{s=\mathtt{fb}[t]+1}^{t} \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_s)\right)\sigma_t^2(\mathbf{x}_i)$$

Proof Denote with $\mathbf{A} = \boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal\boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]}) + \lambda\mathbf{I}$, and with $\mathbf{B} = \boldsymbol{\Phi}(\mathbf{X}_{[\mathtt{fb}[t]+1,t]})$ the concatenation of only the arms between rows $\mathtt{fb}[t]+1$ and t, i.e. in the context of BBKB $\boldsymbol{\Phi}(\mathbf{X}_{[\mathtt{fb}[t]+1,t]})$ contains the arms in the current batch. Then we have $\sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal\mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_i)$ and

$$\sigma_t^2(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal\left(\boldsymbol{\Phi}(\mathbf{X}_t)^\intercal\boldsymbol{\Phi}(\mathbf{X}_t) + \lambda\mathbf{I}\right)^{-1}\boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal\left(\mathbf{A} + \mathbf{B}^\intercal\mathbf{B}\right)^{-1}\boldsymbol{\phi}(\mathbf{x}_i),$$

We can now collect $\mathbf{A}$ to obtain

$$\begin{aligned}\sigma_t^2(\mathbf{x}_i) &= \boldsymbol{\phi}(\mathbf{x}_i)^\intercal(\mathbf{A}+\mathbf{B}^\intercal\mathbf{B})^{-1}\boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal\mathbf{A}^{-1/2}(\mathbf{I}+\mathbf{A}^{-1/2}\mathbf{B}^\intercal\mathbf{B}\mathbf{A}^{-1/2})^{-1}\mathbf{A}^{-1/2}\boldsymbol{\phi}(\mathbf{x}_i)\\ &\geq \lambda_{\min}\left((\mathbf{I}+\mathbf{A}^{-1/2}\mathbf{B}^\intercal\mathbf{B}\mathbf{A}^{-1/2})^{-1}\right)\boldsymbol{\phi}(\mathbf{x}_i)^\intercal\mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_i)\\ &= \lambda_{\min}\left((\mathbf{I}+\mathbf{A}^{-1/2}\mathbf{B}^\intercal\mathbf{B}\mathbf{A}^{-1/2})^{-1}\right)\sigma_{\mathtt{fb}[t]}(\mathbf{x}_i).\end{aligned}$$

Focusing on the first part

$$\begin{aligned}\lambda_{\min}\left((\mathbf{I}+\mathbf{A}^{-1/2}\mathbf{B}^\intercal\mathbf{B}\mathbf{A}^{-1/2})^{-1}\right) &= \frac{1}{\lambda_{\max}\left(\mathbf{I}+\mathbf{A}^{-1/2}\mathbf{B}^\intercal\mathbf{B}\mathbf{A}^{-1/2}\right)}\\ &= \frac{1}{1+\lambda_{\max}(\mathbf{A}^{-1/2}\mathbf{B}^\intercal\mathbf{B}\mathbf{A}^{-1/2})} = \frac{1}{1+\lambda_{\max}(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\intercal)}.\end{aligned}$$

Expanding the definition of $\mathbf{B}$, and using $\lambda_{\max}(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\intercal) \leq \mathrm{Tr}(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\intercal)$ due to the fact that $\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\intercal$ is PSD we have

$$\lambda_{\max}(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\intercal) \leq \mathrm{Tr}(\mathbf{B}\mathbf{A}^{-1}\mathbf{B}^\intercal) = \sum_{j=\mathtt{fb}[t]+1}^{t}\boldsymbol{\phi}(\mathbf{x}_j)\mathbf{A}^{-1}\boldsymbol{\phi}(\mathbf{x}_j) = \sum_{j=\mathtt{fb}[t]+1}^{t}\sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_j).$$

Putting it all together, and inverting the ratio

$$\sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i) \leq \left(1+\textstyle\sum_{s=\mathtt{fb}[t]+1}^{t}\sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_s)\right)\sigma_t^2(\mathbf{x}_i),$$

while to obtain the other side we simply observe that $\mathbf{A}+\mathbf{B}^\intercal\mathbf{B} \succeq \mathbf{A}$ since $\mathbf{B}^\intercal\mathbf{B} \succeq \mathbf{0}$ and therefore $(\mathbf{A}+\mathbf{B}^\intercal\mathbf{B})^{-1} \preceq \mathbf{A}^{-1}$ and $\sigma_t^2(\mathbf{x}_i) \leq \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i)$. ■

We are now ready to prove Theorem 4, which we now restate.

Lemma 8 *For any kernel k, dictionary $\mathcal{S}$, set of points $\mathbf{X}_t$, $\mathbf{x}_i \in \mathcal{A}$, and $\mathtt{fb}[t] < t$,*

$$\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x},\mathcal{S})/(1+\textstyle\sum_{s=\mathtt{fb}[t]}^{t}\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x},\mathcal{S})) \leq \widetilde{\sigma}_t(\mathbf{x},\mathcal{S}) \leq \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x},\mathcal{S}).$$

Proof Note that our approximate posterior can be similarly formulated in a feature-space view. Let $\mathbf{P} = \boldsymbol{\Phi}(\mathbf{X}_\mathcal{S})^\intercal(\boldsymbol{\Phi}(\mathbf{X}_\mathcal{S})\boldsymbol{\Phi}(\mathbf{X}_\mathcal{S})^\intercal)^+\boldsymbol{\Phi}(\mathbf{X}_\mathcal{S})$ be the projection on the arms in the arbitrary dictionary $\mathcal{S}$. Then, referring to [4] for more details, we have

$$\widetilde{\sigma}_t^2(\mathbf{x}_i,\mathcal{S}) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal(\mathbf{P}\boldsymbol{\Phi}(\mathbf{X}_t)^\intercal\boldsymbol{\Phi}(\mathbf{X}_t)\mathbf{P}+\lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{x}_i) = \boldsymbol{\phi}(\mathbf{x}_i)^\intercal(\widetilde{\mathbf{A}}+\widetilde{\mathbf{B}}^\intercal\widetilde{\mathbf{B}})^{-1}\boldsymbol{\phi}(\mathbf{x}_i),$$

where we denote with $\widetilde{\mathbf{A}} = \mathbf{P}\boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal\boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]})\mathbf{P}+\lambda\mathbf{I}$ our approximation of $\mathbf{A}$ and with $\widetilde{\mathbf{B}} = \boldsymbol{\Phi}(\mathbf{X}_{[\mathtt{fb}[t]+1,t]})\mathbf{P}$ our approximation of $\mathbf{B}$. Denote $\widetilde{\boldsymbol{\phi}}(\mathbf{x}) \triangleq \mathbf{P}\boldsymbol{\phi}(\mathbf{x})$. With the exact same reasoning as in the proof of Theorem 7 we can derive

$$\begin{aligned}\widetilde{\sigma}_t^2(\mathbf{x}_i,\mathcal{S}) &= \boldsymbol{\phi}(\mathbf{x}_i)^\intercal(\widetilde{\mathbf{A}}+\widetilde{\mathbf{B}}^\intercal\widetilde{\mathbf{B}})^{-1}\boldsymbol{\phi}(\mathbf{x}_i) \geq \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_i,\mathcal{S})\lambda_{\min}\left((\mathbf{I}+\widetilde{\mathbf{A}}^{-1/2}\widetilde{\mathbf{B}}^\intercal\widetilde{\mathbf{B}}\widetilde{\mathbf{A}}^{-1/2})^{-1}\right)\\ &\geq \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_i,\mathcal{S})/\left(1+\mathrm{Tr}(\widetilde{\mathbf{B}}\widetilde{\mathbf{A}}^{-1}\widetilde{\mathbf{B}}^\intercal)\right) \geq \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_i,\mathcal{S})/\left(1+\textstyle\sum_{s=\mathtt{fb}[t]+1}^{t}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)\widetilde{\mathbf{A}}^{-1}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)\right).\end{aligned}$$

This is still not exactly what we wanted, as $\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)\widetilde{\mathbf{A}}^{-1}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s) \neq \boldsymbol{\phi}(\mathbf{x}_s)\widetilde{\mathbf{A}}^{-1}\boldsymbol{\phi}(\mathbf{x}_s) = \widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}_s,\mathcal{S})$, but we can apply the following Lemma, which we will prove later, to connect the two quantities.

Lemma 9 *Denote with* $\mathbf{P}^\perp = \mathbf{I} - \mathbf{P}$ *the orthogonal projection on the complement of* $\mathbf{P}$. *We have*

$$\boldsymbol{\phi}(\mathbf{x}_s)^\intercal \widetilde{\mathbf{A}}^{-1} \boldsymbol{\phi}(\mathbf{x}_s) = \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)^\intercal \widetilde{\mathbf{A}}^{-1} \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s) + \lambda^{-1} \boldsymbol{\phi}(\mathbf{x}_s)^\intercal \mathbf{P}^\perp \boldsymbol{\phi}(\mathbf{x}_s) \geq \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)^\intercal \widetilde{\mathbf{A}}^{-1} \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)$$

Putting it together and inverting the bound we have

$$\begin{aligned}
\widetilde{\sigma}_t^2(\mathbf{x}_i, \mathcal{S}) &\geq \widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}) / \left(1 + \textstyle\sum_{s=\mathtt{fb}[t]+1}^{t} \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s) \widetilde{\mathbf{A}}^{-1} \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)\right) \\
&\geq \widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}) / \left(1 + \textstyle\sum_{s=\mathtt{fb}[t]+1}^{t} \boldsymbol{\phi}(\mathbf{x}_s) \widetilde{\mathbf{A}}^{-1} \boldsymbol{\phi}(\mathbf{x}_s)\right) \\
&\geq \widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}) / \left(1 + \textstyle\sum_{s=\mathtt{fb}[t]+1}^{t} \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_s, \mathcal{S})\right).
\end{aligned}$$

To obtain the other side of the bound, we simply observe that $\widetilde{\mathbf{A}} + \widetilde{\mathbf{B}}^\intercal \widetilde{\mathbf{B}} \succeq \widetilde{\mathbf{A}}$ and therefore $\widetilde{\sigma}_t^2(\mathbf{x}_i, \mathcal{S}) \leq \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S})$. ■

Finally, combining Theorems 3 and 4, we can prove Theorem 5, which we now restate.

Lemma 10 *Under the same conditions as Theorems 3 and 4,* $\sigma_{\mathtt{fb}[t]}(\mathbf{x})/(3\widetilde{C}) \leq \sigma_t(\mathbf{x}) \leq \sigma_{\mathtt{fb}[t]}(\mathbf{x})$.

Proof Note that Theorems 4 and 7 followed a deterministic derivation based only on linear algebra and therefore held in any case, including the worst. To prove Theorem 5 we must instead rely on the high probability event and guarantees from Theorem 3, and therefore this statement holds only for BBKB run with the correct $\overline{q}$ value and using the reported batch termination condition. The derivation is straightforward

$$\begin{aligned}
\sigma_t^2(\mathbf{x}) &\overset{(a)}{\geq} \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i) / \left(1 + \textstyle\sum_{s=\mathtt{fb}[t]+1}^{t} \sigma_{\mathtt{fb}[t]}(\mathbf{x}_s)\right) \\
&\overset{(b)}{\geq} \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i) / \left(1 + 3\textstyle\sum_{s=\mathtt{fb}[t]+1}^{t} \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[t]})\right) \\
&\geq \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i) / \left(3\left(1 + \textstyle\sum_{s=\mathtt{fb}[t]+1}^{t} \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[t]})\right)\right) \overset{(c)}{\geq} \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_i)/(3\widetilde{C}),
\end{aligned}$$

where (a) is due to Theorem 7, (b) is due to Theorem 3, and (c) is due to the fact that by construction each batch is terminated at a step t where $1 + \sum_{s=\mathtt{fb}[t]+1}^{t} \widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[t]}) \leq \widetilde{C}$ still holds. ■

To conclude the section, we report the proof of Theorem 9

Proof [Proof of Theorem 9] We have

$$\begin{aligned}
\boldsymbol{\phi}(\mathbf{x}_s)^\intercal \widetilde{\mathbf{A}}^{-1}_{\mathtt{fb}[t]} \boldsymbol{\phi}(\mathbf{x}_s) &= \boldsymbol{\phi}(\mathbf{x}_s)^\intercal (\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]}) \widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal + \lambda \mathbf{I})^{-1} \boldsymbol{\phi}(\mathbf{x}_s) \\
&= \boldsymbol{\phi}(\mathbf{x}_s)^\intercal (\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]}) \widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal + \lambda \mathbf{P} + \lambda \mathbf{P}^\perp)^{-1} \boldsymbol{\phi}(\mathbf{x}_s) \\
&\overset{(a)}{=} \boldsymbol{\phi}(\mathbf{x}_s)^\intercal \left((\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]}) \widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal + \lambda \mathbf{P})^{-1} + (\lambda \mathbf{P}^\perp)^{-1}\right) \boldsymbol{\phi}(\mathbf{x}_s) \\
&\overset{(b)}{=} \boldsymbol{\phi}(\mathbf{x}_s)^\intercal (\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]}) \widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal + \lambda \mathbf{P})^{-1} \boldsymbol{\phi}(\mathbf{x}_s) + \lambda^{-1} \boldsymbol{\phi}(\mathbf{x}_s)^\intercal \mathbf{P}^\perp \boldsymbol{\phi}(\mathbf{x}_s)
\end{aligned}$$

where (a) is due to the fact that $\mathbf{P}^\perp$ is complementary to both $\mathbf{P}$ and $\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})$ since $\mathrm{Im}(\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})) \subseteq \mathrm{Im}(\mathbf{P})$, and (b) is due to the fact that $\mathbf{P}^\perp$ is a projection and therefore equal to its inverse. We focus

now on the first term

$$\begin{aligned}
&\boldsymbol{\phi}(\mathbf{x}_s)^\intercal(\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal + \lambda\mathbf{P})^{-1}\boldsymbol{\phi}(\mathbf{x}_s)\\
&\overset{(a)}{=} \boldsymbol{\phi}(\mathbf{x}_s)^\intercal(\mathbf{P}\boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]})\boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal\mathbf{P} + \lambda\mathbf{P})^{-1}\boldsymbol{\phi}(\mathbf{x}_s)\\
&\overset{(b)}{=} \boldsymbol{\phi}(\mathbf{x}_s)^\intercal\mathbf{P}(\mathbf{P}\boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]})\boldsymbol{\Phi}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal\mathbf{P} + \lambda\mathbf{P})^{-1}\mathbf{P}\boldsymbol{\phi}(\mathbf{x}_s)\\
&\overset{(c)}{=} \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)^\intercal(\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal + \lambda\mathbf{P})^{-1}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)\\
&\overset{(d)}{=} \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)^\intercal(\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})^\intercal + \lambda\mathbf{I})^{-1}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)\\
&\overset{(e)}{=} \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)^\intercal\widetilde{\mathbf{A}}_{\mathtt{fb}[t]}^{-1}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)
\end{aligned}$$

where (a) is the definition of $\widetilde{\boldsymbol{\Phi}}_{\mathtt{fb}[t]}(\mathbf{X}_{\mathtt{fb}[t]})$, (b) is because we can collect $\mathbf{P}$ and extract it from the inverse, (c) is the definition of $\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)$, (d) is because $\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)$ lies in $\mathrm{Im}(\mathbf{P})$ and therefore placing $\mathbf{P}$ or $\mathbf{I}$ in the inverse is indifferent, and (e) is the definition of $\widetilde{\mathbf{A}}_{\mathtt{fb}[t]}$. Putting it together

$$\boldsymbol{\phi}(\mathbf{x}_s)^\intercal\widetilde{\mathbf{A}}_{\mathtt{fb}[t]}^{-1}\boldsymbol{\phi}(\mathbf{x}_s) = \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)^\intercal\widetilde{\mathbf{A}}_{\mathtt{fb}[t]}^{-1}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s) + \lambda^{-1}\boldsymbol{\phi}(\mathbf{x}_s)^\intercal\mathbf{P}^\perp\boldsymbol{\phi}(\mathbf{x}_s) \geq \widetilde{\boldsymbol{\phi}}(\mathbf{x}_s)^\intercal\widetilde{\mathbf{A}}_{\mathtt{fb}[t]}^{-1}\widetilde{\boldsymbol{\phi}}(\mathbf{x}_s),$$

since $\boldsymbol{\phi}(\mathbf{x}_s)^\intercal\mathbf{P}^\perp\boldsymbol{\phi}(\mathbf{x}_s)$ is a norm and therefore non-negative. ■

Appendix B. Proofs from Section 3

B.1. Complexity analysis (proof of Theorem 1)

We restate Theorem 1 for completeness.

Theorem 1 *Given $\delta \in (0,1)$, $1 \leq \widetilde{C}$, and $1 \leq \lambda$, run* BBKB *with $\overline{q} \geq 8\log(4T/\delta)$. Then, w.p. $1-\delta$*

1) *For all $t \in [T]$ we have $|\mathcal{S}_t| \leq 9\widetilde{C}(1+\kappa^2/\lambda)\overline{q}d_{\mathit{eff}}(\mathbf{X}_t)$.*

2) *Moreover, the total number of resparsification performed by* BBKB *is at most $\mathcal{O}(d_{\mathit{eff}}(\mathbf{X}_t))$.*

3) *As a consequence,* BBKB *runs in at most $\widetilde{\mathcal{O}}(TAd_{\mathit{eff}}(\mathbf{X}_t)^2)$ near-linear time.*

Proof The proof will be divided in three parts, one for each of the statements.

Bounding $|\mathcal{S}_t|$. The first part of the result concerns space guarantees for $\mathcal{S}_t$. Let us consider again a step $t' > \mathtt{fb}[t]$ where we perform a resparsification (i.e. be the beginning of the following batch) such that $\mathtt{fb}[t'-1] = \mathtt{fb}[t]$ and $\mathtt{fb}[t'] = t'$. Conversely from Theorem 3, where we had to show that our inclusion probabilities $\widetilde{p}_{\mathtt{fb}[t]}$ were not much smaller than $\sigma^2_{\mathtt{fb}[t']}$, here we have to show that they are not much larger than $\sigma^2_{\mathtt{fb}[t']}$. This is because our goal is to sample $\mathcal{S}_{\mathtt{fb}[t']}$ according to $\sigma^2_{\mathtt{fb}[t']}$, and if our sampling probabilities $\widetilde{p}_{\mathtt{fb}[t]} \propto \widetilde{\sigma}_{\mathtt{fb}[t]} \propto \sigma_{\mathtt{fb}[t]}$ are much larger than necessary we are going to wastefully include a number of points larger than necessary. Since BBKB gets computationally heavy if the dictionary gets too large, we want to prove that this does not happen w.h.p.

We begin by invoking Theorem 3 to bound $\widetilde{\sigma}_{\mathtt{fb}[t]} \leq 3\sigma_{\mathtt{fb}[t]}$. The second step is to split the quantity of interest in two parts: one from $\mathtt{fb}[t]$ until the end of the batch $\mathtt{fb}[t'] - 1$, and the crucial step from $\mathtt{fb}[t'] - 1$ to $\mathtt{fb}[t]$

$$\widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S}_{\mathtt{fb}[t]}) \leq 3\sigma^2_{\mathtt{fb}[t]}(\mathbf{x}) = 3 \overbrace{\frac{\sigma^2_{\mathtt{fb}[t]}(\mathbf{x})}{\sigma^2_{\mathtt{fb}[t']-1}(\mathbf{x})}}^{(a)} \overbrace{\frac{\sigma^2_{\mathtt{fb}[t']-1}(\mathbf{x})}{\sigma^2_{\mathtt{fb}[t']}(\mathbf{x})}}^{(b)} \sigma^2_{\mathtt{fb}[t']}(\mathbf{x}).$$

Since $\mathtt{fb}[t]$ and $\mathtt{fb}[t'] - 1$ are both in the same batch, we can use BBKB's batch termination condition and Theorem 5 to bound (a) as $\sigma^2_{\mathtt{fb}[t]}(\mathbf{x})/\sigma^2_{\mathtt{fb}[t']-1}(\mathbf{x}) \leq 3\widetilde{C}$. However, (b) crosses the batch boundaries and does not satisfy the terminating condition. Instead, we will re-use the worst-case guarantees of Theorem 7 to bound the single step increase as

$$\sigma^2_{\mathtt{fb}[t']-1}(\mathbf{x})/\sigma^2_{\mathtt{fb}[t']}(\mathbf{x}) \leq (1 + \sigma^2_{\mathtt{fb}[t']}(\mathbf{x}))\sigma^2_{\mathtt{fb}[t']}(\mathbf{x}) \leq (1 + \kappa^2/\lambda)\sigma^2_{\mathtt{fb}[t']}(\mathbf{x}),$$

where we used the fact that the posterior variance can never exceed κ^2/λ, as can be easily derived from the definition. Putting it all together we have

$$\widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}, \mathcal{S}_{\mathtt{fb}[t]}) \leq 3\sigma^2_{\mathtt{fb}[t]}(\mathbf{x}) \leq 3 \cdot 3\widetilde{C} \cdot (1 + \kappa^2/\lambda) \cdot \sigma^2_{\mathtt{fb}[t']}(\mathbf{x}) = 9\widetilde{C}(1 + \kappa^2/\lambda)\sigma^2_{\mathtt{fb}[t']}(\mathbf{x}), \quad (3)$$

and our overestimate error constant is $9\widetilde{C}(1 + \kappa^2/\lambda)$, which when plugged into [4, Thm. 1] gives us

$$|\mathcal{S}_t| \leq 9\widetilde{C}(1 + \kappa^2/\lambda) \cdot 9(1 + \kappa^2/\lambda)\overline{q}d_{\text{eff}}(\mathbf{X}_t) = 81\widetilde{C}(1 + \kappa^2/\lambda)^2\overline{q}d_{\text{eff}}(\mathbf{X}_t).$$

Bounding the total number of resparsifications. The most expensive operation that BBKB can perform is the GP resparsification, and to guarantee low amortized runtime we now prove that we do not do it too frequently. For this, we will leverage the terminating condition of each batch, since a resparsification is triggered only at the end of each batch.

In particular, we know that if BBKB resparsifies at step t, such that $\mathtt{fb}[t] = t$. Then $1 + \sum_{s=\mathtt{fb}[t-1]+1}^{t-1} \widetilde{\sigma}^2_{\mathtt{fb}[t-1]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[t-1]}) \leq \widetilde{C}$ to not have triggered it at the step before, while we have the opposite inequality $\widetilde{C} < 1 + \sum_{s=\mathtt{fb}[t-1]+1}^{\mathtt{fb}[t]} \widetilde{\sigma}^2_{\mathtt{fb}[t-1]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[t-1]})$ if we include the last term $\widetilde{\sigma}^2_{\mathtt{fb}[t-1]}(\mathbf{x}_{\mathtt{fb}[t]}, \mathcal{S}_{\mathtt{fb}[t-1]})$. Moreover, we have one of such inequalities for each batch in the optimization process. Indicating the number of batches with B, and summing over all the inequalities

$$B\widetilde{C} \leq B + \textstyle\sum_{t=1}^{T} \widetilde{\sigma}^2_{\mathtt{fb}[t]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]})\mathbb{I}\{t \neq \mathtt{fb}[t]\} + \widetilde{\sigma}^2_{\mathtt{fb}[t-1]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t-1]})\mathbb{I}\{t = \mathtt{fb}[t]\},$$

where we have used the indicator function $\mathbb{I}\{\cdot\}$ to differentiate between normal steps and resparsification steps since at the resparsification step we are still using the posterior only w.r.t. the previous choiches $\mathtt{fb}[t-1]$, and more importantly the old dictionary $\mathcal{S}_{\mathtt{fb}[t-1]}$, since the resparsification happens only after the check. However, the only thing that matters to be able to apply Theorem 3 is that the subscript of the posterior $\widetilde{\sigma}_{\mathtt{fb}[t]}$ and of the dictionary $\mathcal{S}_{\mathtt{fb}[t]}$ coincide, so we can further upper bound

$$B\widetilde{C} \leq B + 3\textstyle\sum_{t=1}^{T} \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_t)\mathbb{I}\{t \neq \mathtt{fb}[t]\} + \sigma^2_{\mathtt{fb}[t-1]}(\mathbf{x}_t)\mathbb{I}\{t = \mathtt{fb}[t]\}.$$

Finally, we again exploit the bound $\sigma^2_{\mathtt{fb}[t-1]}(\mathbf{x}_t) \leq 3\widetilde{C}(1 + \kappa^2/\lambda)\sigma^2_t(\mathbf{x}_t)$, we derived in Equation 3 for the evolution of RLS across a whole batch to bound the elements in the summation where $t = \mathtt{fb}[t]$, and apply Theorem 5 to the elements where $t \neq \mathtt{fb}[t]$. We obtain

$$\begin{aligned}
B\widetilde{C} &\leq B + 3\textstyle\sum_{t=1}^{T} \sigma^2_{\mathtt{fb}[t]}(\mathbf{x}_t)\mathbb{I}\{t \neq \mathtt{fb}[t]\} + \sigma^2_{\mathtt{fb}[t-1]}(\mathbf{x}_t)\mathbb{I}\{t = \mathtt{fb}[t]\} \\
&\leq B + 3\textstyle\sum_{t=1}^{T} 3\widetilde{C}\sigma^2_t(\mathbf{x}_t)\mathbb{I}\{t \neq \mathtt{fb}[t]\} + 3\widetilde{C}(1 + \kappa^2/\lambda)\sigma^2_t(\mathbf{x}_t)\mathbb{I}\{t = \mathtt{fb}[t]\} \\
&\leq B + 9\widetilde{C}(1 + \kappa^2/\lambda)\textstyle\sum_{t=1}^{T} \sigma^2_t(\mathbf{x}_t).
\end{aligned}$$

Reshuffling terms and normalizing we obtain

$$B \leq \frac{\widetilde{C}}{\widetilde{C}-1} 9(1+\kappa^2/\lambda) \sum_{t=1}^{T} \sigma_t^2(\mathbf{x}_t),$$

and using the fact that $\sum_{t=1}^{T} \sigma_t^2(\mathbf{x}_t) \leq \mathcal{O}(d_{\text{eff}}(\mathbf{X}_T)\operatorname{polylog}(T)) = \widetilde{\mathcal{O}}(d_{\text{eff}}(\mathbf{X}_T))$ from [2, Lem. 3], we obtain our result.

Complexity analysis. Now that we have a bound on the size of the dictionary, and on the frequency of the resparsifications, we only need to quantify how much each operation costs and amortize it over T iterations.

The resparsification steps are more computationally intensive. Resampling the new $\mathcal{S}_{\mathtt{fb}[t+1]}$ takes $\mathcal{O}(\min\{A,t\})$, as we reuse the variances computed at the beginning of the batch. Given the new embedding function $\mathbf{z}_{\mathtt{fb}[t+1]}(\cdot)$, we must first recomputing the embeddings for all arms in $\mathcal{O}(Am_t^2 + m_t^3)$, and then update all variances in $\mathcal{O}(Am_t^2 + m_t^3)$. Finally, updating the means takes $\mathcal{O}(tm_t + m_t^3)$ time. Overall, a resparsification step requires $\mathcal{O}(Am_t^2 + m_t^3 + tm_t)$, since in all cases of interest $m_t \leq d_{\text{eff}} \ll A$.

In each non-resparsification step, updating the variances requires $\mathcal{O}(m_t^2)$ to update the inverse of $\mathbf{V}_t$ and $\mathcal{O}(m_t^2)$ for each $\widetilde{\sigma}_t(\mathbf{x}_i)$ updated. While the updated actions can be as large as $\mathcal{O}(A)$, lazy evaluations usually require to update just a few entries of $\widetilde{\mathbf{u}}_t$.

Using again B to indicate the number of batches during the optimization, i.e. the number of resparsifications, the overall complexity of the algorithm is thus $\mathcal{O}(\sum_{t=1}^{T} Am_t^2 + \max_{t=1}^{T} B(Am_t^2 + m_t^3 + tm_t))$. Using the dictionary size guarantees of Theorem 1 we can further upper bound this to $\widetilde{\mathcal{O}}(B(Ad_{\text{eff}}^2 + d_{\text{eff}}^3 + Td_{\text{eff}}) + TAd_{\text{eff}}^2)$, and using the bound on resparsifications that we just derived we obtain the final complexity $\widetilde{\mathcal{O}}(TAd_{\text{eff}}^2 + d_{\text{eff}}^4)$ where we used the fact that $d_{\text{eff}} \leq \widetilde{\mathcal{O}}(T)$. ■

B.2. Regret analysis (proof of Theorem 2)

We will leverage the following result from [4]. This is a direct rewriting of their statement with two small modifications. First we express the statement in terms of confidence intervals on the function $f(\mathbf{x})$ rather than in their feature-space view of the GPs. Second, we do not upper bound $\log\det(\mathbf{K}_t/\lambda + \mathbf{I}) \leq \mathcal{O}(\log(t)\sum_{s=1}^{t} \widetilde{\sigma}_t^2(\mathbf{x}_s))$. Calandriello et al. [4] use this upper bound for computational reasons, but as we will see we can obtain a tighter (i.e. without the $\log(t)$ factor) alternative bound that is still efficient to compute.

Proposition 11 ([4, App. D, Thm. 9]) *Under the same assumptions of Theorem 2, with probability at least $1-\delta$ and for all $\mathbf{x}_i \in \mathcal{A}$ and $\mathtt{fb}[t] \geq 1$*

$$\widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]}) - \beta_{\mathtt{fb}[t]}\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]}) \leq f(\mathbf{x}_i) \leq \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]}) + \beta_{\mathtt{fb}[t]}\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]})$$

with

$$\beta_{\mathtt{fb}[t]} \triangleq 2\xi\sqrt{\log\det(\mathbf{K}_{\mathtt{fb}[t]}/\lambda + \mathbf{I}) + \log(1/\delta)} + \left(1+\sqrt{2}\right)\sqrt{\lambda}F$$

We can bound $\log\det(\mathbf{K}_{\mathtt{fb}[t]}/\lambda + \mathbf{I})$ as follows. Consider $\mathbf{K}_s$ as a block matrix split between the s-th column and row, i.e. the latest arm pulled, and all other $s-1$ rows and columns. Then using

Schur's determinant identity, we have that

$$\begin{aligned}\det(\mathbf{K}_s/\lambda + \mathbf{I}) &= \det(\mathbf{K}_{s-1}/\lambda + \mathbf{I}) \det\left(1 + k(\mathbf{x}_s, \mathbf{x}_s) - \mathbf{k}_{s-1}(\mathbf{x}_s)^\intercal (\mathbf{K}_{s-1}/\lambda + \mathbf{I})^{-1} \mathbf{k}_{s-1}(\mathbf{x}_s)\right) \\ &= \det(\mathbf{K}_{s-1}/\lambda + \mathbf{I}) \left(1 + \sigma^2_{s-1}(\mathbf{x}_s)\right).\end{aligned}$$

Combining this with the fact that $\sigma^2_{s-1}(\mathbf{x}_s) \leq \sigma^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s)$, and unrolling the product into a sum using the logarithm we obtain

$$\log\det(\mathbf{K}_{\mathtt{fb}[t]}/\lambda + \mathbf{I}) = \sum_{s=1}^{\mathtt{fb}[t]} \log(1 + \sigma^2_{s-1}(\mathbf{x}_s)) \leq \sum_{s=1}^{\mathtt{fb}[t]} \log(1 + \sigma^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s)).$$

We can further upper bound $\sigma^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s) \leq 3\widetilde{\sigma}^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[s-1]})$ using Theorem 3, and obtain

$$\beta_{\mathtt{fb}[t]} \leq \widetilde{\beta}_{\mathtt{fb}[t]} \triangleq 2\xi\sqrt{\textstyle\sum_{s=1}^{\mathtt{fb}[t]} \log\left(1 + 3\widetilde{\sigma}^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[s-1]})\right) + \log(1/\delta)} + (1+\sqrt{2})\sqrt{\lambda}F$$

This gives us that at all steps t where $t = \mathtt{fb}[t]$ (i.e. right after a resparsification)

$$\widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]}) - \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]}) \leq f(\mathbf{x}_i) \leq \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]}) + \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_i, \mathcal{S}_{\mathtt{fb}[t]})$$

We can bound the instantaneous regret $r_t = f(\mathbf{x}_*) - f(\mathbf{x}_t)$ as follows. First we bound

$$\begin{aligned}f(\mathbf{x}_*) &\leq \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_*, \mathcal{S}_{\mathtt{fb}[t]}) + \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_*, \mathcal{S}_{\mathtt{fb}[t]}) \\ &\overset{(a)}{\leq} \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_*, \mathcal{S}_{\mathtt{fb}[t]}) + \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{C}\widetilde{\sigma}_{t-1}(\mathbf{x}_*, \mathcal{S}_{\mathtt{fb}[t]}) \\ &\overset{(b)}{\leq} \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) + \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{C}\widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]})\end{aligned}$$

where (a) is due to Theorem 4, and (b) is due to the greediness of $\mathbf{x}_t$ w.r.t. $\widetilde{\mathbf{u}}_t$. Similarly, we can bound

$$\begin{aligned}f(\mathbf{x}_t) &\geq \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) - \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{\sigma}_{\mathtt{fb}[t]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) \\ &\geq \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) - \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{C}\widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}).\end{aligned}$$

Putting it together

$$\begin{aligned}R_T &= \sum_{t=1}^{T} r_t = \sum_{t=1}^{T} f(\mathbf{x}_*) - f(\mathbf{x}_t) \\ &\leq \sum_{t=1}^{T} \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) + \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{C}\widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) - \widetilde{\mu}_{\mathtt{fb}[t]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) + \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{C}\widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) \\ &= 2\sum_{t=1}^{T} \widetilde{\beta}_{\mathtt{fb}[t]}\widetilde{C}\widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) \\ &\leq 2\widetilde{C}\widetilde{\beta}_{\mathtt{fb}[T]} \sum_{t=1}^{T} \widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}).\end{aligned} \tag{4}$$

We first focus on bounding $\widetilde{\beta}_{\mathtt{fb}[T]} \leq \widetilde{\beta}_T$, starting from bounding a part of it as

$$\sum_{s=1}^{T} \log\left(1 + 3\widetilde{\sigma}^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s)\right) \overset{(a)}{\leq} 3\sum_{s=1}^{T} \widetilde{\sigma}^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s) \overset{(b)}{\leq} 9\sum_{s=1}^{T} \sigma^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s) \overset{(c)}{\leq} 21\widetilde{C}\sum_{s=1}^{T} \sigma^2_{s-1}(\mathbf{x}_s).$$

where we used (a) the fact that $\log(1+x) \leq x$, (b) Theorem 3, and (c) Theorem 5. Plugging it back into the definition of $\widetilde{\beta}_T$ we have

$$\begin{aligned}\widetilde{\beta}_T &= 2\xi\sqrt{\textstyle\sum_{s=1}^{\mathtt{fb}[T]} \log\left(1 + 3\widetilde{\sigma}^2_{\mathtt{fb}[s-1]}(\mathbf{x}_s, \mathcal{S}_{\mathtt{fb}[s-1]})\right) + \log(1/\delta)} + (1+\sqrt{2})\sqrt{\lambda}F \\ &\leq 2\xi\sqrt{21\widetilde{C}\textstyle\sum_{s=1}^{T} \sigma^2_{s-1}(\mathbf{x}_s) + \log(1/\delta)} + (1+\sqrt{2})\sqrt{\lambda}F\end{aligned}$$

Going back to Equation 4, the summation $\sum_{t=1}^{T} \widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]})$ can be also bounded as

$$\begin{aligned}\sum_{t=1}^{T} \widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) &\overset{(a)}{\leq} \sqrt{T}\left(\sum_{t=1}^{T} \widetilde{\sigma}^2_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]})\right)^{1/2} \overset{(b)}{\leq} \sqrt{T}\left(\sum_{t=1}^{T} \widetilde{\sigma}^2_{\mathtt{fb}[t-1]}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]})\right)^{1/2} \\ &\overset{(c)}{\leq} \sqrt{3}\sqrt{T}\left(\sum_{t=1}^{T} \sigma^2_{\mathtt{fb}[t-1]}(\mathbf{x}_t)\right)^{1/2} \overset{(d)}{\leq} 3\widetilde{C}\sqrt{T}\left(\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)\right)^{1/2},\end{aligned}$$

using (a) Cauchy-Schwarz, (b) the fact that $\widetilde{\sigma}^2_{t-1}(\mathbf{x}_t) \leq \widetilde{\sigma}^2_{\mathtt{fb}[t-1]}(\mathbf{x}_t)$ by Theorem 4, (c) Theorem 3, and (d) Theorem 5. Putting it all together

$$\begin{aligned}R_T &\leq 2\widetilde{C} \cdot \widetilde{\beta}_{\mathtt{fb}[T]} \cdot \textstyle\sum_{t=1}^{T} \widetilde{\sigma}_{t-1}(\mathbf{x}_t, \mathcal{S}_{\mathtt{fb}[t]}) \\ &\leq 2\widetilde{C} \cdot \widetilde{\beta}_T \cdot 3\widetilde{C}\sqrt{T}\left(\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)\right)^{1/2} \\ &\leq 2\widetilde{C} \cdot \left(2\xi\sqrt{21\widetilde{C}\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t) + \log(1/\delta)} + (1+\sqrt{2})\sqrt{\lambda}F\right) \cdot 3\widetilde{C}\sqrt{T}\left(\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)\right)^{1/2} \\ &\leq 2\widetilde{C} \cdot \left(2\xi\sqrt{21\widetilde{C}\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)} + 2\xi\sqrt{\log(1/\delta)} + (1+\sqrt{2})\sqrt{\lambda}F\right) \cdot 3\widetilde{C}\sqrt{T}\left(\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)\right)^{1/2} \\ &\leq 55\widetilde{C}^2\sqrt{T} \cdot \left(\xi\sqrt{\widetilde{C}\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)} + \xi\sqrt{\log(1/\delta)} + \sqrt{\lambda}F\right) \cdot \left(\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)\right)^{1/2} \\ &\leq 55\widetilde{C}^2\sqrt{T} \cdot \left(\xi\sqrt{\widetilde{C}\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)} + \xi\log(1/\delta) + \sqrt{\lambda}F\right) \cdot \left(\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)\right)^{1/2} \\ &\leq 55\widetilde{C}^2 \cdot \widetilde{C} \cdot \sqrt{T}\left(\xi\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t) + (\xi\log(1/\delta) + F)\sqrt{\lambda\textstyle\sum_{t=1}^{T} \sigma^2_{t-1}(\mathbf{x}_t)}\right).\end{aligned}$$

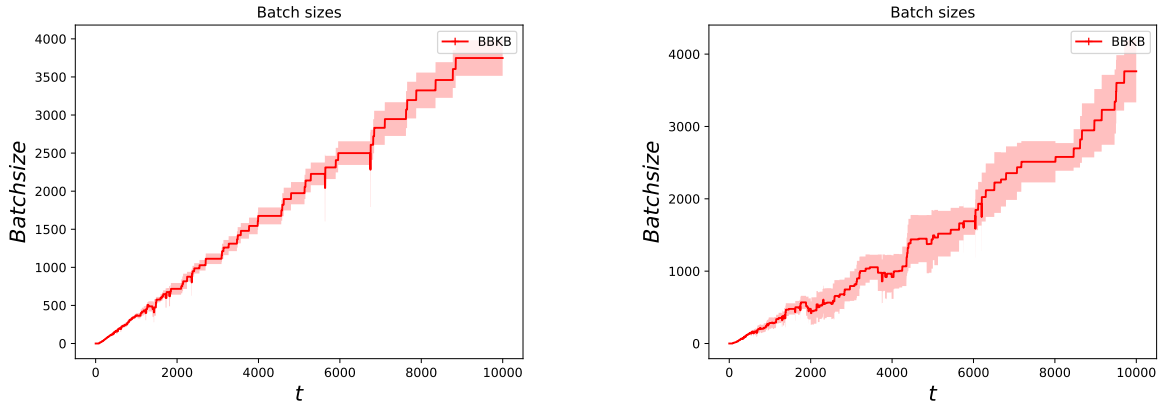

Fig. 2: Batch size of Abalone (left) and Cadata (right)