Consistent Online Gaussian Process Regression Without the Sample Complexity Bottleneck

Alec Koppel[§]

Abstract— **Gaussian process regression provides a framework for nonlinear nonparametric Bayesian inference applicable across machine learning, robotics, chemical engineering, and other settings. Unfortunately, the computational burden of the posterior mean and covariance scales cubically with the training sample size. Even worse, in the online setting where samples perpetually arrive, this complexity approaches infinity. Thus, popular perception is that Gaussian processes cannot be used with streaming data, and that approximations are required. Motivated by this necessity, we develop the first compression sub-routine for online Gaussian processes that preserves their convergence to the population posterior, i.e., *asymptotic posterior consistency*, while ameliorating their intractable complexity growth with the sample size. We do so by after each sequential Bayesian update, fixing an error neighborhood with respect to the Hellinger metric centered at the current empirical probability measure, and greedily tossing out past kernel dictionary elements until we hit the boundary of this neighborhood. We call the resulting method *Parsimonious Online Gaussian Processes* (POG). When we set the error radius, or compression budget, go to null with the sample size, then exact asymptotic consistency is preserved (Theorem 1i) at the cost of unbounded memory in the limit. On the other hand, for constant compression budget, POG converges to a neighborhood of the population posterior distribution (Theorem 1ii) but with finite memory that is at-worst determined by the metric entropy of the feature space (Theorem 2). Experiments on benchmark data demonstrates that POG exhibits favorable performance in practice.**

I. Introduction

Gaussian process regression [26], (kriging [17]) is a framework for nonlinear nonparametric Bayesian inference widely used in chemical processing [14], robotics [10], and machine learning [26], among other applications. Unfortunately, in the batch setting, its complexity scales cubically N^3 with the training sample size N. Moreover, in the online/stochastic setting where the total number of training samples may be infinite $N \to \infty$ or samples continually arrive in a streaming fashion, this complexity tends to infinity. Thus, popular perception is that Gaussian processes cannot work online. In this work, we upend this perception by developing a method that approximately preserves the distributional properties of Gaussian processes in the online setting while yielding a worst-case complexity that is sample size independent, and is instead determined by the metric entropy of the feature space [38].

Consider the batch setting $N < \infty$. The complexity bottleneck comes from the posterior mean and covariance's dependence on a data matrix, or kernel dictionary, that accumulates all past observations. To address this memory explosion, one must choose a subset $M << N$ possible model points from the training set and "project the posterior distribution" onto the "subspace" defined by these samples. Various criteria for projection have been proposed based on information gain [29], greedy compression [30], Nyström sampling [37], probabilistic criteria [20], [5], and many others [7]. Unfortunately, when $N \to \infty$, all of these methods fail to attain the Bayesian notion of optimality known as *asymptotic posterior consistency* [4], [12]. Posterior consistency means that the empirical distribution tends to its population counterpart, and hence the Gaussian mean and covariance tend to their Bayesian ground truth, as the sample size tends to infinity.

Now, let's shift focus to Gaussian processes in the online setting [27]. When written in a sequential manner, the maximum a posteriori (MAP) mean and covariance updates are parameterized by a kernel dictionary that grows by one every time a new sample arrives. Clearly, this is untenable for streaming applications where samples arrive ad infinitum. The question, then, is how to selectively retain a subset of past training examples to ensure the posterior distribution nearly locks onto the population probability measure, while bringing the memory under control.

Motivated recent efforts to break the curse of kernelization [15], [16] in expected risk minimization [35] (ERM), we do so by seeking a Lyapunov function of the sequential estimates. Then, we seek to combine the sequential MAP with a greedy compression step that throws away as much information as possible while preserving the Lyapunov function's per-step evolution. In ERM, the criterion that must be preserved to ensure convergence while compressing is stochastic descent [6], [21]. In Bayesian MAP estimation, on the other hand, it is posterior contraction of probability measures [12].

Posterior contraction means the empirical posterior probability measure based upon t samples and its population counterpart become closer according to some metric. The Hellinger metric may computed in closed form for multivariate Gaussians [1], and many rate results can be specified with respect to this metric [8], [33], [34], [18], [32]. Thus, we define as a Lyapunov function of the sequential MAP estimates the Hellinger distance between the empirical posterior probability from t samples and its population counterpart. Through this insight, our contributions are to:

- develop a new algorithm called Parsimonious Online Gaussian Processes (POG), which, after each MAP

[§]Alec Koppel is with Computational and Information Sciences Directorate, U.S. Army Research Laboratory, Adelphi, MD 20783, USA alec.e.koppel.civ@mail.mil

update, executes online compression by fixing an error neighborhood of radius ϵ_t in terms of the Hellinger distance around the current posterior, and greedily tossing out as many model points as possible, while staying inside this neighborhood (Section III). This compression is accomplished with a custom destructive variant of matching pursuit [24], [36] that quantifies approximation error according to the Hellinger metric. Here ϵ_t denotes the compression budget [15], and the greedy compression may be viewed as a hard-thresholding projection [23] onto subspaces of posterior distributions.
- establish that POG exactly preserves asymptotic posterior consistency of the standard Gaussian Process MAP step when the compression budget approaches null $\epsilon_t \to 0$ as $t \to \infty$ (Theorem 1i, Section IV).
- mitigate the fact that exact consistency requires infinite memory in the limit, by fixing the compression budget $\epsilon_t = \epsilon > 0$ as constant, in which case we establish that the Bayesian posterior estimates converge to within an ϵ-neighborhood of asymptotic posterior consistency (Theorem 1ii). Moreover, with this budget selection, the size of the kernel dictionary is at-worst finite, and defined by the metric entropy of the feature space (Theorem 2). Thus, we obtain an optimal tradeoff of *approximate consistency* and *model parsimony*.
- experimentally (Section V) demonstrate that POG yields favorable tradeoff between performance and complexity, and behaves comparably to the fully dense GP posterior for an online nonlinear filtering task involving LIDAR data [28].

Related Work We briefly review some related works, as the field of approximate online Gaussian processes is not new. [9] kicked off the field decades ago by proposing an additive decomposition of the likelihood, which they combine with fixed-memory subspace projections. This work has been built upon in various ways, such optimizing over the model points that define the likelihood [20], i.e., pseudo-input optimization [31], and numerous approximation strategies for the Gaussian likelihood that exploit information theoretic properties [19].

Alternate memory reduction methods include greedy selection of points as they arrive, as in [29]. Our approach is similar to greedy forward selection; however, our selection criterion is directly tied to preserving statistical consistency, whereas alternate approaches quantify the merit of a compression in ways at best loosely tied to convergence. There are many other works on this area – see [27], [5], [7] for reviews of challenges and approaches for the online setting, or [3], [26][Ch. 8] for experimental evaluations of offline approximations. We note in contrast to all the aforementioned works, we offer a compression routine that provably converges to near-optimality, and that it yields finite memory approximate posterior distributions.

II. Gaussian Process Regression

In Gaussian process regression [17], [26] there is some function $f(\mathbf{x})$ that models the relationship between random variables $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^p$ and $y \in \mathcal{Y} \subset \mathbb{R}$, i.e., $\hat{y} = f(\mathbf{x})$, that we are trying to estimate upon the basis of N training examples $\mathcal{S} = \{\mathbf{x}_n, y_n\}_{n=1}^N$. Unlike in expected risk minimization (ERM), we do not learn this estimator by solving an optimization problem that defines its merit of fitness, but instead assume that this function $f(\mathbf{x})$ follows some particular parameterized family of distributions, and then we seek to estimate those parameters.

In particular, for Gaussian processes, we place a uniform *prior* on the distribution of $\mathbf{f}_{\mathcal{S}} = [f(\mathbf{x}_n), \cdots, f(\mathbf{x}_N)]$ as a Gaussian distribution, namely, $\mathbf{f}_{\mathcal{S}} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_N)$. Here $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate Gaussian distribution in N dimensions with mean vector $\boldsymbol{\mu} \in \mathbb{R}^N$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$. In Gaussian Process regression, the covariance $\mathbf{K}_N = [\kappa(\mathbf{x}_m, \mathbf{x}_n)]_{m,n=1}^{N,N}$ is constructed from a distance-like kernel function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defined over the product set of the feature space. The kernel expresses some prior about how to measure distance between points, a common example of which is itself the Gaussian, $[\mathbf{K}_N]_{mn} = \kappa(\mathbf{x}_m, \mathbf{x}_n) = \exp\{-\|\mathbf{x}_m - \mathbf{x}_n\|^2/c^2\}$ with bandwidth hyper-parameter c.

In standard Gaussian process regression, we further place a zero-mean Gaussian prior on the noise that corrupts $\mathbf{f}_{\mathcal{S}}$ to form the observation vector $\mathbf{y} = [y_1, \cdots, y_N]$, i.e., $\mathbb{P}(\mathbf{y} \,|\, \mathbf{f}_{\mathcal{S}}) = \mathcal{N}(\mathbf{f}_{\mathcal{S}}, \sigma^2\mathbf{I})$ where σ^2 is some variance parameter. In Section IV we use Π to denote the measure associated with the Gaussian prior. We may integrate out the prior on $\mathbf{f}_{\mathcal{S}}$ to obtain the marginal likelihood for $\mathbf{y}$ as

$$\mathbb{P}(\mathbf{y} \,|\, \mathcal{S}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_N + \sigma^2\mathbf{I}) \tag{1}$$

Upon receiving a new data point $\mathbf{x}_{N+1}$, we make a Bayesian inference $\hat{y}_{N+1}$, not by simply setting a point estimate $f(\mathbf{x}_{N+1}) = \hat{y}_{N+1}$. Instead, we formulate the entire *posterior* distribution for y_{N+1} as:

$$\mathbb{P}(y_{N+1} \,|\, \mathcal{S} \cup \mathbf{x}_{N+1}) = \mathcal{N}\Big(\boldsymbol{\mu}_{N+1 \,|\, \mathcal{S}}, \boldsymbol{\Sigma}_{N+1 \,|\, \mathcal{S}}\Big) \tag{2}$$

where the mean and covariance in (2) are given by

$$\begin{aligned} \boldsymbol{\mu}_{N+1 \,|\, \mathcal{S}} &= \mathbf{k}_{\mathcal{S}}(\mathbf{x}_{N+1})[\mathbf{K}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{y}_N \\ \boldsymbol{\Sigma}_{N+1 \,|\, \mathcal{S}} &= \kappa(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) \\ &\quad - \mathbf{k}_{\mathcal{S}}^T(\mathbf{x}_{N+1})[\mathbf{K}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{k}_{\mathcal{S}}(\mathbf{x}_{N+1}) \end{aligned} \tag{3}$$

The expressions (3) are standard – see [26][Chapter 2]. Here $\mathbf{k}_{\mathcal{S}}(\mathbf{x}) = [\kappa(\mathbf{x}_1, \mathbf{x}); \cdots \kappa(\mathbf{x}_N, \mathbf{x})]$ denotes the empirical kernel map. While this approach to sequential Bayesian inference provides a powerful framework for fitting a mean and covariance envelope around observed data, it requires for each N the computation of $\boldsymbol{\mu}_{N+1 \,|\, \mathcal{S}}$ and $\boldsymbol{\Sigma}_{N+1 \,|\, \mathcal{S}}$ which crucially depend on computing the inverse of the kernel matrix $\mathbf{K}_N$ every time a new data point arrives. It is well-known that matrix inversion has cubic complexity $\mathcal{O}(N^3)$ in the variable dimension N, which may be reduced through use of Cholesky factorization [11] or subspace projections [3] combined with various compression criteria such as information gain [29], mean square error [30], integral approximation for Nyström sampling [37], probabilistic criteria [20], [5], and many others [7].

However, even with this complexity reduction, if one tries to run Gaussian Process regression in true streaming applications where the sample size is not necessarily finite $N \to \infty$, any computational savings is eventually rendered useless unless one ensures the complexity remains independent of the sample size. Thus, our objective is to find an approximate Gaussian process whose memory is sample complexity independent, yet is as close as possible in distribution to the fully infinite (as $N \to \infty$) dense Gaussian process. Next, we shift focus to developing such a method.

III. Online Gaussian Processes

In this section, we derive our new algorithm Parsimonious Online Gaussian Process (POG) which is nothing more than a rewriting of the posterior update (2) in time-series manner, and constructing an online sparsification rule that ensures its complexity remains under control, while also nearly preserving posterior consistency. Define the time series of observations as $\mathcal{S}_t = \{\mathbf{x}_u, y_u\}_{u\leq t}$, and then rewrite (2) in a time-varying way in terms of $\mathcal{S}_t \cup \{\mathbf{x}_{t+1}\}$ as

$$\begin{aligned}\boldsymbol{\mu}_{t+1\,|\,\mathcal{S}_t} &= \mathbf{k}_{\mathcal{S}_t}(\mathbf{x}_{t+1})[\mathbf{K}_t + \sigma^2\mathbf{I}]^{-1}\mathbf{y}_t \\ \boldsymbol{\Sigma}_{t+1\,|\,\mathcal{S}_t} &= \kappa(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \\ &\quad - \mathbf{k}^T_{\mathcal{S}_t}(\mathbf{x}_{t+1})[\mathbf{K}_t + \sigma^2\mathbf{I}]^{-1}\mathbf{k}_{\mathcal{S}_t}(\mathbf{x}_{t+1}).\end{aligned} \tag{4}$$

Observe that this update causes the kernel dictionary $\mathbf{X}_t := [\mathbf{x}_1; \cdots; \mathbf{x}_t] \in \mathbb{R}^{p\times t}$ to grow by one at each iteration i.e. $\mathbf{X}_{t+1} = [\mathbf{X}_t; \mathbf{x}_{t+1}] \in \mathbb{R}^{p\times t}$, and that the posterior estimates at time $t+1$ use *all past observations* $\{\mathbf{x}_u\}_{u\leq t}$ in the kernel dictionary $\mathbf{X}_{t+1}$. Subsequently, we refer to the number of columns in the dictionary matrix as the *model order* M_t. The Gaussian processes posterior estimates at time $t+1$ have model order $M_t = t$. For future reference, denote the posterior distributions of y_t and y_{t+1} defined by (4), as $\rho_t = \mathbb{P}(y_t \,|\, \mathcal{S}_{t-1} \cup \mathbf{x}_t)$ and $\rho_{t+1} = \mathbb{P}(y_{t+1} \,|\, \mathcal{S}_t \cup \mathbf{x}_{t+1})$, respectively.

A. Compressing the Posterior Distributions

Our memory-reduction technique relies on the following conceptual link between sparsity in stochastic programming and Bayesian inference: a typical Lyapunov (potential) function of a online supervised learning algorithm is the mean sub-optimality, which quantifies the "energy" in an optimization algorithm. If the method has been appropriately designed, the Lyapunov function is negative semi-definite, and thus flows to the minimum energy state, yielding convergence to optimality [13]. When sparsity considerations are additionally present, the role of a proximal [2]/hard-thresholding [22] operator in tandem with the descent direction must be analyzed to establish decrement in expectation.

Motivated by the convergence of stochastic gradient hard-thresholding algorithms and their use in sparse nonparametric function estimation [15], we seek a Lyapunov function of the sequential Bayesian MAP update for Gaussian processes, and develop a custom new hard-thresholding projection that preserves its per-step behavior. In Bayesian inference, there is no optimization iterate or gradient, but instead only posterior estimates and their associated distributions. Thus, to quantify convergence, we propose measuring how far the empirical posterior is from its population ground truth according to some metric, which actually quantifies distance from *posterior consistency* [4].

We fix our choice of metric as the Hellinger distance due to the fact that it is easily computable for two multivariate Gaussians [1]. For two continuous distributions $\nu(\mathbf{x})$ and $\lambda(\mathbf{x})$ over feature space $\mathcal{X}$, the Hellinger distance is defined as

$$d_H(\nu, \lambda) := \frac{1}{2}\int_{\mathbf{x}\in\mathcal{X}} \left(\sqrt{\nu(\mathbf{x})} - \sqrt{\lambda(\mathbf{x})}\right)^2 d\mathbf{x}\,, \tag{5}$$

which, when both distributions are normal $\nu = \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\lambda = \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$ [1], takes the form

$$\begin{aligned}&d_H(\nu, \lambda) \\ &= \sqrt{1 - \frac{|\Sigma_1|^{1/4}|\Sigma_2|^{1/4}}{|\bar{\Sigma}|}\exp\left\{-\frac{1}{8}(\mu_1 - \mu_2)\,\bar{\Sigma}^{-1}\,(\mu_1 - \mu_2)\right\}}\end{aligned} \tag{6}$$

where $\bar{\Sigma} = (\Sigma_1 + \Sigma_2)/2$. The distribution defined by the Bayesian updates (2) converges to the underlying population distribution in Hellinger distance with probability 1 with respect to the population posterior distribution (see [8][Theorem 6], [33][Theorem 3.3], [18][Theorem 2]). These posterior contraction results motivate the subsequence design of the compression sub-routine. If we select some other kernel dictionary $\mathbf{D} \in \mathbb{R}^{p\times M}$ rather than $\mathbf{X}_t$ for some model order M, the only difference is that the kernel matrix $\mathbf{K}_t$ in (4) and the empirical kernel map $\mathbf{k}_{\mathcal{S}}(\cdot)$ are substituted by $\mathbf{K}_{\mathbf{DD}}$ and $\mathbf{k}_{\mathbf{D}}(\cdot)$, respectively, where the entries of $[\mathbf{K}_{\mathbf{D},\mathbf{D}}]_{mn} = \kappa(\mathbf{d}_m, \mathbf{d}_n)$, $\mathbf{k}_{\mathbf{D}} = [\kappa(\mathbf{d}_1, \cdot); \cdots; \kappa(\mathbf{d}_M, \cdot)]$ and $\{\mathbf{d}_m\}_{m=1}^M \subset \{\mathbf{x}_u\}_{u\leq t}$. Let's rewrite (4) for sample $(\mathbf{x}_{t+1}, y_{t+1})$ with $\mathbf{D}$ as the kernel dictionary rather than $\mathbf{X}_t$ as

$$\begin{aligned}\boldsymbol{\mu}_{t+1\,|\,\mathbf{D}} &= \mathbf{k}_{\mathbf{D}}(\mathbf{x}_{t+1})[\mathbf{K}_{\mathbf{D},\mathbf{D}} + \sigma^2\mathbf{I}]^{-1}\mathbf{y}_t \\ \boldsymbol{\Sigma}_{t+1\,|\,\mathbf{D}} &= \kappa(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) \\ &\quad - \mathbf{k}^T_{\mathbf{D}}(\mathbf{x}_{t+1})[\mathbf{K}_{\mathbf{D},\mathbf{D}} + \sigma^2\mathbf{I}]^{-1}\mathbf{k}_{\mathbf{D}}(\mathbf{x}_{t+1}).\end{aligned} \tag{7}$$

The question, then, is how to select a sequence of dictionaries $\mathbf{D}_t \in \mathbb{R}^{p\times M_t}$ whose M_t columns comprise a subset of those of $\mathbf{X}_t$ in such a way to preserve asymptotic posterior consistency.

Suppose we have a dictionary $\mathbf{D}_t \in \mathbb{R}^{p\times(M_t)}$ at time t and observe point $\mathbf{x}_{t+1}$. We compute its associated posterior distribution $\rho_{\mathbf{D}_t} := \mathcal{N}(\boldsymbol{\mu}_{t+1\,|\,\mathbf{D}_t}, \boldsymbol{\Sigma}_{t+1\,|\,\mathbf{D}_t})$, where the expressions for the mean and covariance can be obtained by substituting $\mathbf{D} = \mathbf{D}_t$ into (4), assuming that $\mathbf{D}_t$ has already been chosen.

We propose compressing dictionary $\mathbf{D}_t$ of model order M_t to obtain a dictionary $\tilde{\mathbf{D}}_{t+1}$ of smaller model complexity $M_{t+1} \leq M_t$ by executing the update in (7), fixing an error neighborhood centered at $\mathcal{N}(\boldsymbol{\mu}_{t+1\,|\,\mathbf{D}_t}, \boldsymbol{\Sigma}_{t+1\,|\,\mathbf{D}_t})$ in the Hellinger metric. Then, we prune dictionary elements greedily with respect to the Hellinger metric until we hit

Algorithm 1 Parsimonious Online GPs (POG)

```
Require: $\{\mathbf{x}_t, y_t, \epsilon_t\}_{t=0,1,2,\ldots}$
  initialize noise prior $\sigma^2$, empty dictionary $\mathbf{D}_0 = []$
  for $t = 0, 1, 2, \ldots$ do
    Obtain independent training pair $(\mathbf{x}_{t+1}, y_{t+1})$
    Update posterior mean and covariance estimates (7)
      $\boldsymbol{\mu}_{t+1 \,|\, \mathbf{D}_t} = \mathbf{k}_{\mathbf{D}_t}(\mathbf{x}_{t+1})[\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t} + \sigma^2\mathbf{I}]^{-1}\mathbf{y}_t$
      $\boldsymbol{\Sigma}_{t+1 \,|\, \mathbf{D}_t} = \kappa(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}_{\mathbf{D}_t}^T(\mathbf{x}_{t+1})[\mathbf{K}_{\mathbf{D}_t,\mathbf{D}_t} + \sigma^2\mathbf{I}]^{-1}\mathbf{k}_{\mathbf{D}_t}(\mathbf{x}_{t+1})$
    Compress w.r.t. Hellinger metric via Algorithm 2
      $(\boldsymbol{\mu}_{\tilde{\mathbf{D}}_{t+1}}, \boldsymbol{\Sigma}_{\tilde{\mathbf{D}}_{t+1}}, \tilde{\mathbf{D}}_{t+1}) = \textbf{DHMP}(\boldsymbol{\mu}_{t+1 \,|\, \mathbf{D}_t}, \boldsymbol{\Sigma}_{t+1 \,|\, \mathbf{D}_t}, \mathbf{D}_t, \epsilon_t)$
    Revise dictionary via MAP update $\mathbf{D}_{t+1} = [\tilde{\mathbf{D}}_{t+1}, \mathbf{x}_{t+1}]$
  end for
```

the boundary of this error neighborhood. This is a destructive variant of matching pursuit [24], [36] that has been customized to operate with the Hellinger distance, and is motivated by the fact that we can tune its stopping criterion to assure that the intrinsic distributional properties of the Bayesian update are almost unchanged. We call this routine Destructive Hellinger Matching Pursuit (DHMP) with budget parameter ϵ_t.

DHMP with compression budget ϵ_t, summarized in Algorithm 2, operates by taking as input a kernel dictionary $\mathbf{D}_t$ and associated posterior mean and covariances estimates $\boldsymbol{\mu}_{t+1 \,|\, \mathbf{D}_t}, \boldsymbol{\Sigma}_{t+1 \,|\, \mathbf{D}_t}$ and initializing its approximation as the input. Then, it sequentially and greedily removes kernel dictionary elements $\mathbf{d}_j$ according to their ability to that cause the least error in Hellinger distance. Then, it terminates when the resulting distribution $\rho_{\tilde{\mathbf{D}}_{t+1}}$ hits the boundary of an ϵ_t error neighborhood in Hellinger distance from its input. This procedure with input $\mathbf{D} \in \mathbb{R}^{p \times M_t+1}$, $\boldsymbol{\mu}_{t+1 \,|\, \mathbf{D}}, \boldsymbol{\Sigma}_{t+1 \,|\, \mathbf{D}}$ and parameter ϵ_t, is summarized in Algorithm 2.

The full algorithm, Parsimonious Online Gaussian Processes (POG), is summarized as Algorithm 1. It is the standard sequential Bayesian MAP updates of Gaussian process regression (4) with current dictionary $\mathbf{D}_t$ operating in tandem with DHMP (Algorithm 2), i.e., $(\boldsymbol{\mu}_{\tilde{\mathbf{D}}_{t+1}}, \boldsymbol{\Sigma}_{\tilde{\mathbf{D}}_{t+1}}, \tilde{\mathbf{D}}_{t+1}) = \textbf{DHMP}(\boldsymbol{\mu}_{t+1 \,|\, \mathbf{D}_t}, \boldsymbol{\Sigma}_{t+1 \,|\, \mathbf{D}_t}, \tilde{\mathbf{D}}_{t+1}, \epsilon_t)$. After compression completes, the latest sample $\mathbf{x}_{t+1}$ is appended to the dictionary to form the prior for the next iteration as $\mathbf{D}_{t+1} = [\tilde{\mathbf{D}}_{t+1}, \mathbf{x}_{t+1}]$. We denote the Gaussian distribution with mean $\boldsymbol{\mu}_{t+1 \,|\, \mathbf{D}_{t+1}}$ and covariance $\boldsymbol{\Sigma}_{t+1 \,|\, \mathbf{D}_{t+1}}$ at time $t+1$ as $\rho_{\mathbf{D}_{t+1}}$.

The compression budget ϵ_t may be chosen to carefully trade off closeness to the population posterior distribution and model parsimony. This tradeoff is the subject of the subsequent section.

Algorithm 2 Destructive Hellinger Matching Pursuit (DHMP)

```
Require: posterior dist. $\rho_{\tilde{\mathbf{D}}}$ defined by dict. $\mathbf{D} \in \mathbb{R}^{p \times \tilde{M}}$, mean $\boldsymbol{\mu}_{t+1 \,|\, \mathbf{D}}$ covariance $\boldsymbol{\Sigma}_{t+1 \,|\, \mathbf{D}}$, budget $\epsilon_t > 0$
  initialize mean $\boldsymbol{\mu}_{\tilde{\mathbf{D}}} = \tilde{\boldsymbol{\mu}}_{t+1 \,|\, \tilde{\mathbf{D}}}$, cov. $\boldsymbol{\Sigma}_{\mathbf{D}} = \tilde{\boldsymbol{\Sigma}}_{t+1 \,|\, \tilde{\mathbf{D}}}$,
  dict. $\mathbf{D} = \mathbf{D}$ w/ indices $\mathcal{I}$ with model order $M = \tilde{M}$
  while candidate dictionary is non-empty $\mathcal{I} \neq \emptyset$ do
    for $j = 1, \ldots, \tilde{M}$ do
      Compute mean $\boldsymbol{\mu}_{\mathbf{D}_{-j}}$, cov. $\boldsymbol{\Sigma}_{\mathbf{D}_{-j}}$ w/o dict. point $\mathbf{d}_j$
      Find error with dict. element $\mathbf{d}_j$ removed (6)
        $\gamma_j = d_H(\rho_{\mathbf{D}_{-j}}, \rho_{\tilde{\mathbf{D}}})$ .
    end for
    Find minimal error dict. point: $j^* = \text{argmin}_{j \in \mathcal{I}} \gamma_j$
      if minimal approximation error exceeds $\gamma_{j^*} > \epsilon_t$
        stop
      else
        Prune dictionary $\mathbf{D} \leftarrow \mathbf{D}_{\mathcal{I} \setminus \{j^*\}}$
        Revise set $\mathcal{I} \leftarrow \mathcal{I} \setminus \{j^*\}$, model order $M \leftarrow M - 1$.
      end
  end while
  return dictionary $\mathbf{D}$ such that $d_H(\rho_{\mathbf{D}}, \rho_{\tilde{\mathbf{D}}}) \leq \epsilon_t$
```

IV. Balancing Consistency and Parsimony

The foundation of our technical results are the well-developed history of convergence of the empirical posterior (4) to the true population posterior [4]. Various posterior contraction rates are available, but they depend on the choice of prior, the underlying smoothness of the generative process f, the choice of metric [33], [34], [18], the sampling coverage and radius of the feature space [32], among other technicalities. To avoid descending down the rabbit hole of measure theory and functional analysis, we opt for the simplest technical setting we could find for asymptotic posterior consistency of (4), which is stated next.

Lemma 1: [8][Theorem 6(2) and Sec. 6, Ex. 2] Assume that f_0 is the true response function, σ_0^2 is the true noise variance that corrupts observations $\mathbf{y}_t$, and ρ_0 is the associated true population posterior. Assume the following conditions:

i) Suppose training examples $\mathbf{x}_t \in \mathcal{X} \subset \mathbb{R}^p$ are sampled from $\mathcal{X} = [0,1]^p$, and $\{\mathbf{x}_t, \mathbf{y}_t\}_{t=1}^{\infty} \overset{i.i.d.}{\sim} \mathbb{P}_0$, where $\mathbb{P}_0$ is the true joint distribution of $(\mathbf{x}, \mathbf{y})$.
ii) Denote $\lambda([0,1]^p)$ as the Lebesgue measure on the hypercube. There exists a constant $0 < K_p \leq 1$ such that whenever $\lambda([0,1]^p) \geq \frac{1}{K_p t}$, $[0,1]^p$ contains at least one sample $\mathbf{x}_t$.
iii) For $t \geq 1$, the kernel matrix $\mathbf{K}_t$ is positive definite $\mathbf{K}_t \succ \mathbf{0}$.
iv) The kernel $\kappa(\mathbf{x}, \mathbf{x}')$ is of the form $\kappa(\beta\|\mathbf{x} - \mathbf{x}'\|)$ for some strictly positive $\beta > 0$.
v) For each $t \geq 1$, there exist positive constants $0 < \delta < 1/2$ and $b_1, b_2 > 0$, such that the covariance kernel parameter β satisfies $\mathbb{P}\{\beta > t^{\delta}\} < b_1 e^{-b_2 t}$.

Then, for every $\gamma > 0$, the posterior ρ_t defined by (4) is asymptotically consistent, i.e.,

$$\mathbb{P}_\Pi\left\{d_H(\rho_t,\rho_0)<\gamma \,\middle|\, \mathcal{S}_t\right\}\to 1 \text{ a.s. with respect to } \mathbb{P}_0 \quad (8)$$

where the probability $\mathbb{P}_\Pi$ computed in (8) is the Gaussian prior density Π as in Sec. II.

This result forms the foundation of our Lyapunov-style analysis of our Bayesian learning algorithm (Algorithm 1). To establish our main result, we additionally require the following assumption, which says that the per step change in the posterior likelihood is unchanged by compression.

Assumption 1: Define the events $\eta_t := \{d_H(\rho_t,\rho_{t-1})<\gamma \,|\, \mathcal{S}_t\}$ and $\tilde{\eta}_t := \{d_H(\rho_{\tilde{\mathbf{D}}_t},\rho_{\mathbf{D}_{t-1}})<\gamma \,|\, \mathcal{S}_t\}$ for any constant $\gamma>0$. The single step likelihood change with respect to the Gaussian prior (Sec. II, paragraph 2) of the uncompressed posterior is at least as likely as the uncompressed single step likelihood change based upon sample point $(\mathbf{x}_t, y_t)$ is the same, i.e.,

$$\mathbb{P}_\Pi\{\eta_t\}\geq \mathbb{P}_\Pi\{\tilde{\eta}_t\}.$$

where $\mathbb{P}_\Pi$ denotes the prior Gaussian likelihood in Sec. II.

Assumption 1 is reasonable because the uncompressed and compressed updates observe the same sample $(\mathbf{x}_t,\mathbf{y}_t)$ at time t and formulate conditional Gaussian likelihoods based upon them. While they are conditioned on different dictionaries $\mathcal{S}_{t-1}$ and $\mathbf{D}_{t-1}$, the likelihood of the compressed posterior is at least as likely as the fully dense GP. In the analysis, Assumption 1 plays the role of a Bayesian analogue of nonexpansiveness of projection operators. Under this assumption and the conditions of the aforementioned lemma, we can establish almost sure convergence under both diminishing and constant compression budget selections as stated next.

Theorem 1: Under the same conditions as Lemma 1, Algorithm 1 attains the following posterior consistency results almost surely:

i) for decreasing compression budget $\epsilon_t\to 0$, for any $\alpha>0$, as $t\to\infty$, we have $\mathbb{P}_\Pi\{d_H(\rho_{\mathbf{D}_t},\rho_{\mathbf{D}_{t-1}})<\alpha \,|\, \mathcal{S}_t\}\to 1$ w.r.t. population posterior $\mathbb{P}_0$.
ii) for fixed budget $\epsilon_t=\epsilon>0$, ϵ-approximate convergence with respect to the Hellinger metric is attained, i.e., for any $\alpha>0$, as $t\to\infty$, $\mathbb{P}_\Pi\{d_H(\rho_{\mathbf{D}_t},\rho_{\mathbf{D}_{t-1}})+\epsilon<\alpha \,|\, \mathcal{S}_t\}\to 1$ w.r.t. the population posterior $\mathbb{P}_0$.

Proof: See Appendix of [25]. ■

Theorem 1 establishes a formal tradeoff between the choice of compression budget and the accuracy with which we are able to lock onto the population posterior distribution. Specifically, for attenuating compression budget, the algorithm retains more and more sample points as time progresses, such that in the limit it exactly locks onto the fully infinite dimensional population posterior. On the other hand, for constant compression budget, we can converge to a neighborhood of the population posterior, but for this selection we can additionally guarantee that the complexity of the distribution's parameterization never grows out of control. This finite memory property is formalized in the following theorem.

Theorem 2: Suppose Algorithm 1 is run with constant compression budget $\epsilon>0$. Then the model order M_t of the posterior distributions $\rho_{\mathbf{D}_t}$ remains finite for all t, and subsequently the limiting distribution ρ_∞ has finite model complexity M^∞. Moreover, $M_t\leq M^\infty$ for all t.
Proof: See Appendix of [25]. ■

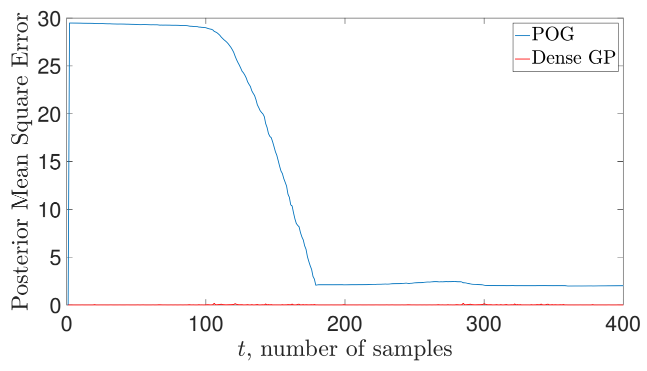
Theorem 2 establishes a unique result for approximate Gaussian processes, namely, that the attainment of approximate posterior consistency comes with the salient feature that the posterior admits a parsimonious representation. In the worst case, the model complexity depends on the metric entropy of the feature space $\mathcal{X}$, rather than growing unbounded with the iteration index t. The combination of Theorems 1ii and 2 establish a Nyquist Sampling Theorem for Gaussian Processes over compact features spaces. In the subsequent section we empirically validate the aforementioned theoretical results on a simple regression task common to autonomous systems.

V. Experiments

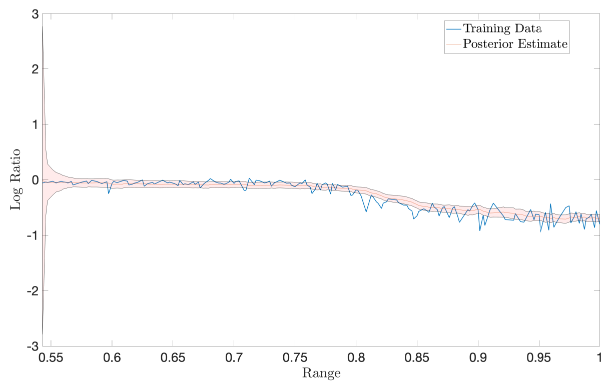
In this section, we consider a simple experimental setup where a sequence of scalar ($p=1$) range measurements are taken by a robot equipped with a light detection and ranging (LIDAR) system as the independent variable x_t. The dependent variable y_t is the (scalar) log-ratio of light received from the two lasers, and may be used to infer presence of obstacles or walls [28]. There are $T=179$ total training examples in this data set. This is a classical setting where linear regression fails and polynomial regression, while attaining a reasonable model fit, reveals little about the physical obstacle, and hence nonparametric approaches may be most appropriate – see [28][Ch. 2.6-2.8].

We run Algorithm 1, POG, for this setting and compare it against the fully dense Gaussian Process (Dense GP) when the noise prior is fixed as $\sigma^2=0.4$ and the compression budget is kept constant $\epsilon_t=\epsilon=5\times 10^{-4}$ for five training epochs. We quantify performance in terms of mean square error and a visualization of the ground truth as compared to the posterior mean estimate together with its 95% (three standard deviations) confidence interval shaded in red. Also visualized is the evolution of the Hellinger distance between subsequent posterior distributions and the model order over time.

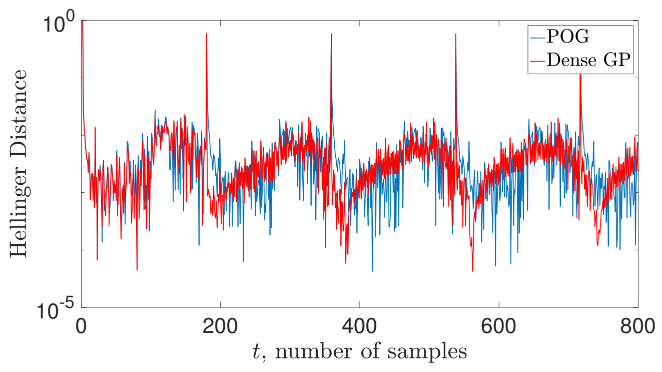
Observe in Figure 1a that POG behaves comparably to the Dense GP, albeit with higher mean square error. The Dense GP is able to track the signal to a very high degree of accuracy, whereas POG tracks it to a bounded neighborhood. This trend is corroborated in the visualization of the evolution of the POG estimator over time as compared to the ground truth in Figure 1b: POG tracks the mean performance of the LIDAR dependent variable y_t very well, and its three standard deviation confidence interval well-encompasses the uncertainty in the data. Not pictured here is the tracking performance of the fully dense GP, which nearly exactly tracks the blue curve. One could argue, however, that this exact adherence is actually overfitting.



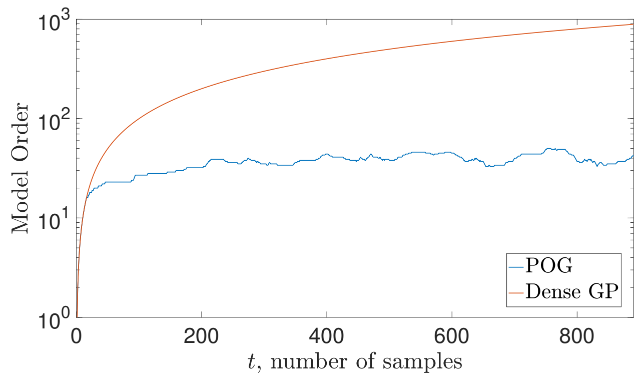
(a) Mean square error of posterior mean $\mu_{\mathbf{D}_t}$ vs. training index t



(b) Actual target and posterior mean $\pm$ three standard deviations



(c) Hellinger Distance of subsequent posteriors vs. training index t



(d) Model Complexity vs. training index t

Fig. 1: A comparison of POG with the fully dense Gaussian Process on the LIDAR data set [28][Ch. 2.6-2.8], a classic nonlinear filtering problem where Bayesian methods are appropriate. Observe that POG attains worse performance than the Dense GP in terms of mean square error (Fig. 1a), but tracks the mean value of the ground truth effectively, and its confidence interval covers all volatility in the data (Fig. 1b). This behavior is attained by sparsifying while ensuring the per-step distributional change with respect to the Hellinger metric of the Bayesian updates are relatively unchanged – see Fig. 1c. Moreover, this sequential sparsification procedure yields orders of magnitude reduction in model complexity of the posterior distribution, as evidenced by Fig. 1d.

In Figure 1c we plot the evolution of the Hellinger distance between subsequent iterates for both POG and the Dense GP. Observe that their behavior is very similar, meaning that the sparsification (Algorithm 2) does not substantially change the statistical behavior of the Gaussian process as more and more data points arrive. On the other hand, this sparsification yields an order of magnitude reduction in model complexity as may be observed in Figure 1d. The magnitude of this reduction will only grow over time: as more and more samples arrive, the Dense GP will store all past observations in its dictionary matrix, whereas the posterior defined by POG remains near-consistent but with a dictionary containing between 30 and 40 samples.

VI. Conclusion

In this work, we investigated the classic scalability bottleneck of Bayesian maximum a posterior estimation using Gaussian processes. In particular, the mean and covariance have computational complexity that is proportionate to the iteration index, and therefore untenable for streaming data scenarios common in autonomous control. We pointed out that statisticians have already proposed valid Lyapunov functions that quantify the asymptotic behavior of Gaussian processes, among which is the Hellinger distance between the current posterior and its population counterpart. Since it is computable in closed form for Gaussians, we restricted focus to the Hellinger distance. By introducing hard-thresholding projections based on matching pursuit, we were able to design sparsification rules that nearly preserve the statistical behavior of Gaussian processes while bringing their memory usage under control. A simple experiment on a streaming data set involving LIDAR to track the distance to an obstacle provided practical evidence that these projections work well in practice and corroborate our theoretical results. Further experimental investigation is required, as is extension of the this work to classification, rather than regression problems.

References

[1] Karim T Abou-Moustafa and Frank P Ferrie. A note on metric properties for some divergence measures: The gaussian case. In *Asian Conference on Machine Learning*, pages 1–15, 2012.

[2] YVES F Atchade, GERSENDE Fort, and ERIC Moulines. On stochastic proximal gradient algorithms. *arXiv preprint arXiv:1402.2365*, 23, 2014.

[3] Anjishnu Banerjee, David B Dunson, and Surya T Tokdar. Efficient gaussian process regression for large datasets. *Biometrika*, 100(1):75–89, 2012.

[4] Andrew Barron, Mark J Schervish, Larry Wasserman, et al. The consistency of posterior distributions in nonparametric problems. *The Annals of Statistics*, 27(2):536–561, 1999.

[5] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse gaussian process approximations. In *Advances in neural information processing systems*, pages 1533–1541, 2016.

[6] Léon Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

[7] Thang D Bui, Cuong Nguyen, and Richard E Turner. Streaming sparse gaussian process approximations. In *Advances in Neural Information Processing Systems*, pages 3299–3307, 2017.

[8] Taeryon Choi and Mark J Schervish. On posterior consistency in nonparametric regression problems. *Journal of Multivariate Analysis*, 98(10):1969–1987, 2007.

[9] Lehel Csató and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[10] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

[11] Leslie Foster, Alex Waagen, Nabeela Aijaz, Michael Hurley, Apolonio Luis, Joel Rinsky, Chandrika Satyavolu, Michael J Way, Paul Gazis, and Ashok Srivastava. Stable and efficient gaussian process calculations. *Journal of Machine Learning Research*, 10(Apr):857–882, 2009.

[12] Subhashis Ghosal, Jayanta K Ghosh, Aad W van der Vaart, et al. Convergence rates of posterior distributions. *The Annals of Statistics*, 28(2):500–531, 2000.

[13] Hassan K Khalil. Noninear systems. *Prentice-Hall, New Jersey*, 2(5):5–1, 1996.

[14] Juš Kocijan. *Modelling and control of dynamic systems using Gaussian process models*. Springer, 2016.

[15] Alec Koppel, Garrett Warnell, Ethan Stump, and Alejandro Ribeiro. Parsimonious online learning with kernels via sparse projections in function space. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pages 4671–4675. IEEE, 2017.

[16] Alec Koppel, Kaiqing Zhang, Hao Zhu, and TM Baser. Projected stochastic primal-dual method for constrained online learning with kernels. *IEEE Trans. Signal Process.(submitted)*, 2018.

[17] Daniel G Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[18] Willem Kruijer, Judith Rousseau, Aad Van Der Vaart, et al. Adaptive bayesian density estimation with location-scale mixtures. *Electronic Journal of Statistics*, 4:1225–1257, 2010.

[19] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *arXiv preprint arXiv:1807.01065*, 2018.

[20] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. Sparse gaussian processes for bayesian optimization. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 517–526. AUAI Press, 2016.

[21] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[22] Nam Nguyen, Deanna Needell, and Tina Woolf. Linear convergence of stochastic iterative greedy algorithms with sparse constraints. *IEEE Transactions on Information Theory*, 63(11):6869–6895, 2017.

[23] Neal Parikh, Stephen Boyd, et al. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[24] Y. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. In *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, 1993.

[25] Hrusikesha Pradhan, Alec Koppel, and Ketan Rajawat. Consistent compressions of online gaussian processes with pseudo-inputs. 2019.

[26] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[27] Stephen Roberts, Michael Osborne, Mark Ebden, Steven Reece, Neale Gibson, and Suzanne Aigrain. Gaussian processes for time-series modelling. *Phil. Trans. R. Soc. A*, 371(1984):20110550, 2013.

[28] David Ruppert, Matt P Wand, and Raymond J Carroll. Semiparametric regression during 2003–2007. *Electronic journal of statistics*, 3:1193, 2009.

[29] Matthias Seeger, Christopher Williams, and Neil Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Artificial Intelligence and Statistics 9*, number EPFL-CONF-161318, 2003.

[30] Alex J Smola and Peter L Bartlett. Sparse greedy gaussian process regression. In *Advances in neural information processing systems*, pages 619–625, 2001.

[31] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

[32] Andrew Stuart and Aretha Teckentrup. Posterior consistency for gaussian process approximations of bayesian posterior distributions. *Mathematics of Computation*, 87(310):721–753, 2018.

[33] Aad W van der Vaart and J Harry van Zanten. Rates of contraction of posterior distributions based on gaussian process priors. *The Annals of Statistics*, pages 1435–1463, 2008.

[34] Aad W van der Vaart, J Harry van Zanten, et al. Adaptive bayesian estimation using a gaussian random field with inverse gamma bandwidth. *The Annals of Statistics*, 37(5B):2655–2675, 2009.

[35] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[36] P. Vincent and Y. Bengio. Kernel matching pursuit. *Machine Learning*, 48(1):165–187, 2002.

[37] Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[38] Ding-Xuan Zhou. The covering number in learning theory. *Journal of Complexity*, 18(3):739–767, 2002.