Unbiased Stochastic Optimization for Gaussian Processes on Finite Dimensional RKHS

Anonymous authors
Paper under double-blind review

Abstract

Current methods for stochastic hyperparameter learning in *Gaussian Processes* (GPs) rely on approximations, such as computing biased stochastic gradients or using inducing points in stochastic variational inference. However, when using such methods, we are not guaranteed to converge to a stationary point of the true marginal likelihood. In this work, we propose algorithms for exact stochastic inference of GPs with kernels that induce a Reproducing Kernel Hilbert Space (RKHS) of moderate finite dimension. Our approach can also be extended to infinite dimensional RKHSs at the cost of forgoing exactness. Both for finite and infinite dimensional RKHSs, our method achieves better experimental results than existing methods when memory resources limit the feasible batch size and the possible number of inducing points.

1 Introduction

Gaussian Processes (GPs) provide a powerful probabilistic framework that has been applied to a wide range of learning applications, such as multi-task learning (Alvarez et al., 2012; Liu et al., 2018a), active learning (Liu et al., 2018b), semi-supervised learning (Jean et al., 2018), and reinforcement learning (Srinivas et al., 2010; Shahriari et al., 2015). These successes can be attributed to the natural way in which uncertainty is incorporated into the predictions via a Bayesian interpretation. However, hyperparameter learning scales poorly. For a general covariance function (kernel function in the parlance of kernel methods), the computational cost grows as $O(n^3)$, where n denotes the number of samples, and the storage resources grow as $O(n^2)$. As a consequence, approximations are required for any modern application that involves GPs and big data.

It is therefore unsurprising that much research effort has been devoted to approximate methods for learning GPs. For a comprehensive overview of this research, we refer the reader to the work of Liu et al. (2020). The most expensive operation in training GPs is the inversion of the $n \times n$ kernel matrix, which is required during the maximization of the marginal likelihood and for the computation of the posterior of the responses. As a result, much research has been devoted to providing a cheap approximation of the kernel matrix.

Broadly speaking, approximations of GPs can be split into *data dependent* and *data independent* methods. Data dependent approximations usually come with valuable probabilistic interpretations. An important work in this line of research is the seminal work of Quinonero-Candela & Rasmussen (2005), which provides a unified view on previous work by using the concept of inducing points. Since then quite a few follow-up works have used inducing points, while exploiting the tool of variational inference as a theoretical platform (Titsias, 2009; Nguyen et al., 2014; Wilson & Nickisch, 2015; Zhao & Sun, 2016). This approach is closely related to the Nyström approximation of kernel matrices (Zhao & Sun, 2016).

In contrast, data independent methods rely on approximating the kernel function itself. Typically, it is approximated by an inner product between low dimensional vectors (Rahimi & Recht, 2007; Yang et al., 2015; Shustin & Avron, 2021).

Approximating the inverse of the kernel matrix is enough for frequentist kernel methods such as Kernel Ridge Regression. However, in order to harness the full power of Bayesian kernel methods such as *Gaussian Process*

Regression (GPR), we also need to be able to maximize the marginal likelihood; for that, an approximation of the inverse of the kernel matrix is not enough due to an additional log-determinant term. This is especially true in cases where the covariance function depends on a large number of parameters, e.g., evaluating the covariance function involves a forward pass of a deep neural network (Wilson et al., 2016b; Calandra et al., 2016; Wilson et al., 2016a). In such cases, it is important to be able to use stochastic optimization based on mini-batches; otherwise, the cost of making a pass over the entire dataset typically proves too expensive. Moreover, it is well appreciated in the literature that when the model involves deep neural networks, i.e., when the covariance function is defined by a neural network, it is crucial to use stochastic gradients, as they enable more efficient optimization and better generalization (Goodfellow et al., 2016).

Currently, two main approaches are used for stochastic hyperparameter learning in GPs. The first relies on the inducing-points framework and employs *Stochastic Variational Inference* (SVI) (Hoffman et al., 2013; Hensman et al., 2013; Hoang et al., 2015; Wilson et al., 2016a). The second approach is more direct: it is based on computing stochastic batches while ignoring the fact that they provide only biased estimates of gradients. Interestingly, recent work shows that despite the bias, given a large enough batch size, the direct approach produces almost optimal models (in terms of the marginal likelihood) (Chen et al., 2020).

Both the SVI approach and the direct approach suffer from several disadvantages. For example, consider the case in which the covariance function is the inner product between two feature maps of moderate dimension. This can be either because this decomposition serves as an approximation to another covariance function with an infinite dimensional *Reproducing Kernel Hilbert Space* (RKHS), or because it has been defined this way to begin with, e.g., as an inner product between features that are created by passing the data through a neural network. Either way, using the SVI approach amounts to imposing an additional unnecessary approximation that comes from the need to use inducing points. As for the second approach, its applicability is highly dependent on our ability to use large batches (Chen et al., 2016). This can be a serious impediment in several cases, such as optimization on weak edge devices.

In this paper, we propose two stochastic optimization algorithms based on mini-batches for maximizing the marginal likelihood of GPs (i.e., learning the hyperparameters). The first algorithm is based on reframing the problem as a nonconvex-concave minimax problem. We then leverage recent advancements in the theory of solving such problems (Boţ & Böhm, 2023; Lin et al., 2020; Luo et al., 2020) to propose a concrete algorithm. The second algorithm is based on writing the loss function in compositional form, i.e., as the composition of a function and the expected value of another function. We then use the recently introduced *Stochastic Compositional Gradient Descent* method (Wang et al., 2017). Our novel algorithms are applicable for covariance functions connected with an RKHS of moderate dimension and guarantee convergence of the marginal likelihood to a local minimum for any batch size without the need for further approximations. In the infinite dimensional case (e.g., Gaussian covariance function), one can use our method on top of a low rank approximation of the covariance function, e.g., using the *Random Fourier Features* method (Rahimi & Recht, 2007). Our experiments show that not only is our method superior to existing methods for stochastic optimization of the marginal likelihood in the finite dimensional case when the batches have a moderate size, but it is also superior to the existing methods in the infinite dimensional case if the restriction on batch size is more severe. However, in the infinite dimensional case, the models found by our method are no longer optimal.

Additional Related Work. Apart from the extensive literature on scaling GPs using approximations, several works focus on exact inference using sophisticated distributed algorithms (Nguyen et al., 2019; Wang et al., 2019). In this context, the biased stochastic gradient proposed by Chen et al. (2020) can be considered an economical method for exact inference, given that the covariance function and the system enable computation in large enough batches.

2 Preliminaries

2.1 Notations and Basic Definitions

For a function $f{:}\mathcal{U} \to \mathbb{R}$ and $U = (u_1, \ldots, u_m) \in \mathcal{U}^m$, $\mathbf{f} = f(U)$ is a vector in $\mathbb{R}^m$ such that $\mathbf{f}_i = f(u_i)$. Similarly, for a binary function $k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, $K = k(U, U)$ is a matrix in $\mathbb{R}^{m \times m}$ such that $K_{ij} = k(u_i, u_j)$. Given a size m vector $\mathbf{b}$ and an index sequence $\mathcal{S} = (s_1, \ldots, s_r) \in \{1, \ldots, m\}^r$ (so $r = |\mathcal{S}|$), we use $\mathbf{b}_\mathcal{S}$ to denote the size r vector such that the ith coordinate of $\mathbf{b}_\mathcal{S}$ is equal to the s_ith coordinate of $\mathbf{b}$. In a similar way, if C is an $m \times n$ matrix, then $C_\mathcal{S}$ is a $r \times n$ matrix such that the ith row of $C_\mathcal{S}$ is equal to the s_ith row of C. Finally, if $\mathcal{R} = (r_1, \ldots, r_q) \in \{1, \ldots, n\}^q$, then $C_{\mathcal{S}\mathcal{R}}$ is a $r \times q$ matrix such that the (i, j) coordinate of $C_{\mathcal{S}\mathcal{R}}$ is equal to the (s_i, r_j) coordinate of C.

For a square matrix A we use the notation $|A|$ to denote the determinant of A. If $\mathcal{S}$ is a finite sequence or a set then $|\mathcal{S}|$ denotes its length or size. Whenever we use $\langle A, B \rangle$, where A and B are real matrices, $\langle \cdot, \cdot \rangle$ symbolizes the Frobenius inner product, which is defined as

$$\langle A, B \rangle = \mathbf{Tr}\left(AB^T\right).$$

If A is a real matrix, then $\|A\|$ is the Frobenius norm of A.

$$\|A\| = \sqrt{\langle A, A \rangle}.$$

For a vector $\mathbf{v}$, $\|\mathbf{v}\|$ is the Euclidean norm of $\mathbf{v} \in \mathbb{R}^q$.

For any closed convex set $\Omega \subseteq \mathbb{R}^q$ and $\mathbf{v} \in \mathbb{R}^q$, $\mathrm{proj}_\Omega(\mathbf{v}) = \arg\min_{\mathbf{v}' \in \Omega} \|\mathbf{v}' - \mathbf{v}\|$ is the Euclidean projection of $\mathbf{v}$ onto Ω.

2.2 Hyperparameter Learning in Gaussian Process Regression

Let $\mathcal{X}$ be a feature space and let $k_\alpha : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite covariance function parameterized by hyperparameters $\alpha \in \mathbb{R}^m$. For any $\alpha \in \mathbb{R}^m$, let f_α be a random function on $\mathcal{X}$ distributed as a zero mean GP whose covariance is k_α, that is, for any $j \in \mathbb{N}$, $U \in \mathcal{X}^j$:

$$f_\alpha(U) \sim \mathcal{N}(0, k_\alpha(U, U)).$$

Let $X \in \mathcal{X}^n$, $\mathbf{y} \in \mathbb{R}^n$ be a training set, where n is the number of training samples. In *Gaussian Process Regression* (GPR), it is assumed that $\mathbf{y}$ is a sample of the random vector $f_{\bar{\alpha}}(X) + \epsilon$, where $\epsilon \sim \mathcal{N}\left(0, \bar{\sigma}^2 I_n\right)$ and $\bar{\alpha}, \bar{\sigma}^2$ are the true hyperparameters of the model.

In GPR, hyperparameters are learned by solving a maximum likelihood type II problem, i.e., maximizing the marginal likelihood

$$p\left(\mathbf{y}|X, \alpha, \sigma^2\right) = \mathcal{N}\left(\mathbf{y}|0, K(\alpha) + \sigma^2 I\right),$$

with respect to α and $\sigma^2 > \sigma^2_{\min} > 0$, where $K(\alpha) = k_\alpha(X, X)$. The maximization of the marginal likelihood $p\left(\mathbf{y}|X, \alpha, \sigma^2\right)$ is equivalent to the minimization of

$$\mathbf{y}^T \left(K(\alpha) + \sigma^2 I\right)^{-1} \mathbf{y} + \log\left|K(\alpha) + \sigma^2 I\right|.$$

See Rasmussen (2003) for details. In this work, we further assume that the covariance function has the following form:

$$k_\alpha(x, x') = \phi_\alpha(x)^T \phi_\alpha(x'),$$

for some feature map $\phi_\alpha : \mathcal{X} \to \mathbb{R}^d$. It can be shown that for all $\lambda > 0$, $V \in \mathbb{R}^{n \times d}$, and $\mathbf{b} \in \mathbb{R}^d$, we have that

$$\mathbf{b}^T \left(VV^T + \lambda I\right)^{-1} \mathbf{b} = \min_{\mathbf{w}} \frac{1}{\lambda} \|V\mathbf{w} - \mathbf{b}\|^2 + \|\mathbf{w}\|^2,$$

(see Appendix B). As a result, maximizing $p(\mathbf{y}|X, \alpha, \sigma)$ is equivalent to minimizing

$$\ell(\theta) = \frac{1}{\sigma^2} \|Z(\alpha)\mathbf{w} - \mathbf{y}\|^2 + \|\mathbf{w}\|^2 + \log|F(\theta)| + (n - d)\log\sigma^2,$$

where

$$\begin{aligned}
\theta &= (\mathbf{w}, \alpha, \beta)\,, \\
\beta &= \log\left(\sigma^2 - \sigma_{\min}^2\right), \\
Z(\alpha) &= \left(\phi_\alpha(x_1), \ldots, \phi_\alpha(x_n)\right)^T \in \mathbb{R}^{n\times d}, \\
F(\theta) &= Z(\alpha)^T Z(\alpha) + \sigma^2 I \in \mathbb{R}^{d\times d}.
\end{aligned}$$

The goal of this work is to propose algorithms for minimizing $\ell(\theta)$ using a stochastic gradient method based on mini-batches. This would have been straightforward if we could write

$$\nabla_\theta \log|F(\theta)| = \sum_{i=1}^{n} G(\theta; x_i)\,,$$

for some function $G(\cdot;\cdot)$. However, there is no obvious decomposition of this form.

3 Stochastic Optimization for Gaussian Processes

This section contains our main contribution: two novel stochastic mini-batch-based algorithms for minimizing $\ell(\theta)$. Before detailing our approaches, we make a few additional notations:

$$\begin{aligned}
g_i(\theta) &= \frac{1}{\sigma^2}\left(\phi_\alpha(x_i)^T \mathbf{w} - y_i\right)^2 + \frac{1}{n}\|\mathbf{w}\|^2 + \frac{1}{n}(n-d)\log\sigma^2, \\
g(\theta) &= \sum_{i=1}^{n} g_i(\theta)\,, \\
F_i(\theta) &= \phi_\alpha(x_i)\,\phi_\alpha(x_i)^T + \frac{1}{n}\sigma^2 I, \\
h(A) &= \log|A|\,.
\end{aligned}$$

So, we can write,

$$\begin{aligned}
F(\theta) &= \sum_{i=1}^{n} F_i(\theta)\,, \\
\ell(\theta) &= g(\theta) + h(F(\theta))\,.
\end{aligned}$$

3.1 A Minimax Approach

Our first step is to replace the problem

$$\min_\theta g(\theta) + h(F(\theta))\,,$$

with the equivalent problem

$$\begin{aligned}
&\min_{\theta, A} g(\theta) + h(A) \\
&\text{s.t. } A = F(\theta)\,.
\end{aligned} \tag{1}$$

The next step is to replace the last problem with a parameterized problem, such that the hard constraint is replaced with a penalty term, and the penalty term is driven to infinity. To do so, let us first define the optimization problems

$$\min_\zeta \ell_\mu(\zeta)$$

where $\zeta = (\theta, A)$ and

$$\ell_\mu(\zeta) = g(\theta) + h(A) + \mu\frac{\|A - F(\theta)\|}{\|A\|}.$$

Now, suppose that Problem (1) admits an optimal solution. Let the sequences $\{\mu_k\}$ and $\{\zeta_k\}$ satisfy $\mu_k \to \infty$ and, for each k, let ζ_k minimize $\ell_{\mu_k}(\zeta)$. If $\zeta^\star$ is an accumulation point of $\{\zeta_k\}$, then $\zeta^\star$ is an optimal solution of Problem (1) (Ruszczyński, Theorem 6.6).

We remark that, rather than minimizing only $\|A - F(\theta)\|$, we use the modified objective described above. Empirically, this reduces convergence to undesirable stationary points, e.g., runs in which $F(\theta)$ becomes spuriously small without improving downstream performance.

Unfortunately, the term $\|A - F(\theta)\|$ exhibits the same issues as $h(F(\theta)) = |F(\theta)|$ and does not allow for straightforward unbiased mini-batch-based stochastic gradients. In order to handle this, we use the fact that for an element v in Euclidean space, we have that $\|v\| = \max_{\|u\|\leq 1} \langle u, v\rangle$. Thus, minimizing $\ell_\mu(\zeta)$ is equivalent to

$$\min_{\zeta} \max_{\|B\|\leq 1} \Psi(\zeta, B; \mu), \tag{2}$$

where

$$\Psi(\zeta, B; \mu) = g(\theta) + h(A) + \mu \frac{\langle B, A - F(\theta)\rangle}{\|A\|},$$

Now, we can write

$$\Psi(\zeta, B; \mu) = \sum_{i=1}^{n} \psi(\zeta, B; x_i, \mu), \tag{3}$$

where

$$\psi(\zeta, B; x_i, \mu) = g_i(\theta) + \frac{1}{n} h(A) + \mu \frac{\left\langle B, \frac{1}{n}A - F_i(\theta)\right\rangle}{\|A\|},$$

Thus, Problem (2) naturally admits unbiased stochastic gradients based on mini-batches taken separately for the minimization and maximization parts.

For numerical stability, we keep the (learned) noise variance bounded away from zero and ensure A stays uniformly well-conditioned by enforcing

$$\sigma^2 > \sigma^2_{\min} > 0 \quad , \sigma^2 I \preceq A \preceq A_{\max}.$$

In addition, we clip the model parameters $\theta = (\mathbf{w}, \alpha, \beta) = (\mathbf{w}, \alpha, \log(\sigma^2 - \sigma^2_{\min}))$ to a box

$$\theta_{\min} \preceq \theta \preceq \theta_{\max}.$$

Altogether, the feasible set for $\zeta = (\theta, A)$ is

$$\Omega_1 = \Big\{(\theta, A) \;\Big|\; \theta_{\min} \preceq \theta \preceq \theta_{\max},\ \sigma^2 I \preceq A \preceq A_{\max}\Big\},$$

where $\sigma^2_{\min} > 0$, $\sigma^2_{\max}$, $\theta_{\min}$, $\theta_{\max}$, and $A_{\max}$ are fixed (non-learned) hyperparameters.

Let $\Omega_2 = \left\{B \in \mathbb{R}^{d\times d} \mid \|B\| \leq 1\right\}$. The algorithm we propose is based on the following update rule for solving Problem (2):

1. $\zeta_{t+1} = \operatorname{proj}_{\Omega_1}\left(\zeta_t - a\nabla_\zeta \frac{n}{s} \sum_{i\in\mathcal{S}_{t+1}} \psi(\zeta_t, B_t; x_i, \mu)\right),$
2. $B_{t+1} = \operatorname{proj}_{\Omega_2}\left(B_t + b\nabla_B \frac{n}{s} \sum_{i\in\bar{\mathcal{S}}_{t+1}} \psi(\zeta_{t+1}, B_t; x_i, \mu)\right).$

$$\mu_{k+1} = \mathcal{U}(\mu_k), \qquad \mu_k \uparrow \infty.$$

where $\mathcal{S}_t$ and $\bar{\mathcal{S}}_t$ are sets of s indices randomly chosen from $\{1, \ldots, n\}$, independently from all previous iterations. Recent work by Boţ & Böhm (2023) shows that in the case of a minimax problem where the

Algorithm 1 Stochastic Minimax (Alternating) for GP Hyperparameter Learning (Outer-Loop Penalty)

Require: Dataset $(X, \mathbf{y})$ of size n, feature map ϕ_α, batch size s
Require: Inner iterations T, outer penalty stages K
Require: Step sizes $a, b > 0$
Require: Penalty initialization $\mu_0 > 0$ and a nondecreasing update rule $\mu_{k+1} = \mathcal{U}(\mu_k)$
Require: **(Fixed hyperparameters defining the constraint sets)** box bounds $\theta_{\min}, \theta_{\max}$ and a PSD upper bound $A_{\max} \succeq 0$.

Definitions: ($\preceq$ is elementwise for vectors and Loewner order for matrices; I_d is the $d \times d$ identity)

$$\Omega_1 = \Big\{\zeta = (\theta, A) : \ \theta_{\min} \preceq \theta \preceq \theta_{\max}, \ \sigma^2 I_d \preceq A \preceq A_{\max}\Big\}$$

$$\Omega_2 = \{B \in \mathbb{R}^{d\times d} : \|B\| \le 1\}$$

$$g_i(\theta) = \tfrac{1}{\sigma^2}\big(\phi_\alpha(x_i)^\top \mathbf{w} - y_i\big)^2 + \tfrac{1}{n}\|\mathbf{w}\|^2 + \tfrac{1}{n}(n-d)\log\sigma^2$$

$$F_i(\theta) = \phi_\alpha(x_i)\phi_\alpha(x_i)^\top + \tfrac{1}{n}\sigma^2 I_d, \quad h(A) = \log|A|$$

$$\psi(\zeta, B; x_i, \mu) = g_i(\theta) + \tfrac{1}{n}h(A) + \mu\,\frac{\langle B, \ \frac{1}{n}A - F_i(\theta)\rangle}{\|A\|}$$

Initialize $\zeta \leftarrow (\theta_0, A_0) \in \Omega_1$, $B \leftarrow B_0 \in \Omega_2$, and $\mu \leftarrow \mu_0$
for $k = 0$ **to** $K-1$ **do** ▷ outer penalty loop
  for $t = 0$ **to** $T-1$ **do** ▷ inner minimax loop at fixed μ
    Sample mini-batches $\mathcal{S}, \ \bar{\mathcal{S}} \subset \{1, \dots, n\}$ with $|\mathcal{S}| = |\bar{\mathcal{S}}| = s$ independently
    (Min step) Compute

$$G_\zeta \leftarrow \frac{n}{s}\sum_{i\in\mathcal{S}} \nabla_\zeta \psi(\zeta, B; \, x_i, \mu)$$

      Update $\zeta \ \leftarrow \ \mathrm{proj}_{\Omega_1}\big(\,\zeta - a\,G_\zeta\,\big)$
    (Max step) Compute

$$G_B \leftarrow \frac{n}{s}\sum_{i\in\bar{\mathcal{S}}} \nabla_B \psi(\zeta, B; \, x_i, \mu)$$

      Update $B \ \leftarrow \ \mathrm{proj}_{\Omega_2}\big(\,B + b\,G_B\,\big)$
  end for
  (Penalty update) $\mu \ \leftarrow \ \mathcal{U}(\mu)$ ▷ e.g., $\mu \leftarrow \gamma\mu$ with $\gamma \ge 1$; constant-μ uses $\gamma = 1$
end for
Return $\theta = (\mathbf{w}, \alpha, \log(\sigma^2))$ from ζ

maximization is of a concave function over a convex constraint, for correctly chosen step sizes a and b, [1] with a few additional mild conditions on the objective function, an algorithm based on the above update rule visits a point that is at an $O(\epsilon)$ distance from a stationary point in $O\left(\epsilon^{-8}\right)$ iterations (more details can be found in Appendix B.2).

While the projection onto Ω_2 is just the Euclidean projection onto the Frobenius unit ball, the projection onto Ω_1 consists of (i) simple coordinate-wise clipping of the vector parameters in θ and (ii) spectral clipping of A to satisfy the matrix inequalities $\sigma^2 I_d \preceq A \preceq A_{\max}$. Concretely, letting $A = UDU^\top$ be an eigendecomposition and assuming $A_{\max}$ is diagonal (e.g., $A_{\max} = a_{\max} I_d$), we compute

$$\mathrm{proj}_{\{\,\sigma^2 I_d \preceq M \preceq A_{\max}\,\}}(A) = U \ \mathrm{diag}\Big(\min\big(\max(D_{11}, \sigma^2), (A_{\max})_{11}\big), \dots, \min\big(\max(D_{dd}, \sigma^2), (A_{\max})_{dd}\big)\Big)U^\top.$$

This projection therefore requires an eigenvalue decomposition of a $d \times d$ matrix, but it does not change the overall per-iteration asymptotic complexity; see Appendix A for details.

The Minimax approach is summarized in Algorithm 1.

[1] The setup in Boţ & Böhm (2023) is more general.

3.2 Stochastic Compositional Gradient Descent Approach

Consider a loss function $\ell : \Theta \to \mathbb{R}$ of the form $\ell(\theta) = v(u(\theta))$ where $u : \Theta \to \mathbb{R}^p$ and $v : \mathbb{R}^p \to \mathbb{R}$ are differentiable functions, and assume that $u(\theta) = \mathbb{E}_\omega \, \tilde{u}(\theta; \omega)$ for a differentiable function $\tilde{u}(\theta; \omega)$ that depends on a random variable ω. *Stochastic Compositional Gradient Descent* (SCGD) (Wang et al., 2017) is an intuitive algorithm that alternates between two steps: updating the solution θ_t by a stochastic gradient iteration, and estimating $u(\theta_t)$ using an iterative weighted average of past values. More precisely, the update rule of SCGD is given by:

$$\eta_{t+1} = (1 - b_t)\, \eta_t + b_t \, \tilde{u}(\theta_t; \omega_t),$$
$$\theta_{t+1} = \theta_t - a_t \, J_t^\top \, \nabla v(\eta_{t+1}).$$

where $\omega_1, \omega_2, \ldots$ are samples from ω in an i.i.d. manner, and $a_0, a_1, \ldots, b_0, b_1, \ldots$ degrees of freedom in the algorithm. Under a few additional standard conditions on $u, v, \tilde{u}$, Wang et al. (2017) establish an iteration–complexity bound for basic SCGD in the sense that, if we define

$$T_\epsilon := \min\Big\{ t \geq 0 : \inf_{0 \leq s \leq t} \mathbb{E}\big[\|\nabla \ell(\theta_s)\|^2\big] \leq \epsilon \Big\},$$

then, for power-law step sizes $a_t = (t+1)^{-\gamma}$ and $b_t = (t+1)^{-\delta}$ with $\delta < \gamma < 2\delta$, we have

$$T_\epsilon = \mathcal{O}\Big(\epsilon^{-1/q}\Big), \qquad q = \min\{\, 1 - \gamma,\ \gamma - \delta,\ 2\delta - \gamma,\ \gamma \,\}.$$

In particular, taking $(\gamma, \delta) = \left(\frac{3}{4}, \frac{1}{2}\right)$ yields $q = \frac{1}{4}$ and hence recovers $T_\epsilon = \mathcal{O}\big(\epsilon^{-4}\big)$ (see Wang et al. (2017, Theorem 8) and Appendix B, Theorem 3).

In order to use SCGD for our loss, we define:

$$u(\theta) = (g(\theta), F(\theta)),$$
$$v(u_1, u_2) = u_1 + h(u_2),$$
$$\tilde{u}(\theta; \omega) = \frac{n}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} (g_i(\theta), F_i(\theta)),$$

where $\omega = \mathcal{S}$ is a random set of indices chosen from $\{1, \ldots, n\}$. With that, given the fact that

$$\frac{\partial \log |A|}{\partial A} = A^{-1}$$

we can write an explicit update rule for our SCGD-based algorithm:

$$\tilde{F}_{t+1} = (1 - b_t)\, \tilde{F}_t \ + \ b_t \, \frac{n}{|\mathcal{S}_{t+1}|} \sum_{i \in \mathcal{S}_{t+1}} F_i(\theta_t),$$

$$\theta_{t+1} = \theta_t \ - \ a_t \, \frac{n}{|\mathcal{S}_{t+1}|} \sum_{i \in \mathcal{S}_{t+1}} \nabla_\theta \big[g_i(\theta) + \big\langle \tilde{F}_{t+1}^{-1}, F_i(\theta) \big\rangle \big] \big|_{\theta = \theta_t}.$$

The SCGD approach is summarized in Algorithm 2. A formal convergence theorem can be found in Appendix B, Theorem 3.

4 Comparison to Existing Methods

In this work, we suggest two novel methods for the stochastic optimization of the marginal likelihood. We recognize two main competing methods that also optimize the marginal likelihood stochastically. The simplest among them is what we henceforth refer to as *Biased Stochastic Gradient Descent* (BSGD) (Chen et al., 2020). The idea in BSGD is to take the gradient of the marginal likelihood using data only from the current batch, ignoring the fact that this produces only a biased estimate of the full gradient. On the

Algorithm 2 SCGD for GP Hyperparameter Learning

Require: Dataset $(X, \mathbf{y})$ of size n, feature map ϕ_α, batch size s, iterations T
Require: Step-size sequences $\{a_t\}_{t=0}^{T-1}$ and $\{b_t\}_{t=0}^{T-1}$ (e.g., $a_t = (t+1)^{-3/4}$, $b_t = (t+1)^{-1/2}$)
Notation: $d = \dim\phi_\alpha(\cdot)$, I_d is the $d \times d$ identity ($\preceq$ is elementwise for vectors)
Per-sample terms:

$$g_i(\theta) = \frac{1}{\sigma^2}\big(\phi_\alpha(x_i)^\top \mathbf{w} - y_i\big)^2 + \frac{1}{n}\|\mathbf{w}\|^2 + \frac{1}{n}(n-d)\log\sigma^2$$

$$F_i(\theta) = \phi_\alpha(x_i)\phi_\alpha(x_i)^\top + \frac{1}{n}\sigma^2 I_d$$

Initialize $\theta_0 \in \Theta$ and a SPD tracker $\tilde{F}_0 \succ 0$ (e.g., $\tilde{F}_0 \leftarrow \sigma_0^2 I_d$)
for $t = 0$ **to** $T-1$ **do**
  Sample mini-batch $\mathcal{S}_{t+1} \subset \{1, \ldots, n\}$, $|\mathcal{S}_{t+1}| = s$
  (Exponential tracking of $F(\theta)$)

$$\tilde{F}_{t+1} \;\leftarrow\; (1-b_t)\,\tilde{F}_t \;+\; b_t\,\frac{n}{s}\sum_{i\in\mathcal{S}_{t+1}} F_i(\theta_t)$$

  Compute $\tilde{F}_{t+1}^{-1}$ (e.g., via Cholesky of $\tilde{F}_{t+1}$)
  (Gradient step)

$$G_\theta \leftarrow \frac{n}{s}\sum_{i\in\mathcal{S}_{t+1}} \nabla_\theta\Big[g_i(\theta) + \langle \tilde{F}_{t+1}^{-1},\, F_i(\theta)\rangle\Big]\Big|_{\theta=\theta_t}$$

  $\theta_{t+1} \;\leftarrow\; \theta_t - a_t\, G_\theta$
end for
Return $\theta_T = (\mathbf{w}_T, \alpha_T, \beta_T)$

other hand, *the Scalable Variational Gaussian Process* (SVGP) of Hensman et al. (2015) is a sophisticated approach that approximates the original inference problem using inducing points and stochastic variational inference.

Unlike our algorithms, the effectiveness of each of the competing methods in approximating the solution to the true problem depends on memory consumption. In the case of BSGD, the bias in stochastic gradients shrinks as the batch size increases. On the other hand, the inference quality of SVGP crucially depends on the number of inducing points that need to be processed forward and backward by a neural network at each iteration.

4.1 Complexity Analysis

Let C and M be the computational and memory complexities in the computation of $\frac{\partial\phi_\alpha(x)}{\partial\alpha}$, and let s be the batch size and s' be the maximum of the batch size and the number of inducing points in SVGP.

For a fair comparison under a fixed memory budget, we set the number of inducing points in SVGP to be on the same order as the mini-batch size s (and in our experiments we simply set it equal to s). Each inducing point must be processed by the same feature map (i.e., requires a forward and backward pass through the network), so it incurs essentially the same activation and gradient storage cost as an additional training example. Consequently, the dominant per-iteration memory usage scales roughly with the total number of points passed through the network, and choosing the inducing set size to match s keeps the memory footprint balanced and the comparisons meaningful.

The computational complexity of computing one optimization iteration of both BSGD and SVGP is $O\left(s'^3\right) + O\left(s'^2 d\right) + s'\mathrm{C}$, while both of our algorithms take $O\left(d^3\right) + O\left(s'd^2\right) + s'\mathrm{C}$ per iteration. For storage, BSGD and SVGP take $O\left(s^2\right) + O\left(sd\right) + s\mathrm{M}$, while ours takes $O\left(d^2\right) + O\left(sd\right) + s\mathrm{M}$.

Typically, M is relatively large, as it is the storage used by *Automatic Differentiation* (AD) to compute the gradient of a neural network. In small devices, this means one might have to use small batches. Our

Table 1: Complexity analysis for one optimization iteration: C and M are the computational and the memory complexity in the computation of $\frac{\partial \phi_\alpha(x)}{\partial \alpha}$. s is the batch size.

Algorithm	Computations	Storage
SVGP	$O\left(s^3\right)+O\left(s^2 d\right)+s\mathrm{C}$	$O\left(s^2\right)+O\left(sd\right)+s\mathrm{M}$
BSGD	$O\left(s^3\right)+O\left(s^2 d\right)+s\mathrm{C}$	$O\left(s^2\right)+O\left(sd\right)+s\mathrm{M}$
Minimax	$O\left(d^3\right)+O\left(sd^2\right)+s\mathrm{C}$	$O\left(d^2\right)+O\left(sd\right)+s\mathrm{M}$
SCGD	$O\left(d^3\right)+O\left(sd^2\right)+s\mathrm{C}$	$O\left(d^2\right)+O\left(sd\right)+s\mathrm{M}$

approach aims not to harm the exactness of the algorithm in this case. Naturally, to use our algorithms effectively, the feature dimension $d = \dim\phi_\alpha(x)$ must remain moderate, since memory scales as $O(d^2)$ and computation as $O(d^3)$.

4.2 Relation to BSGD

Our approach is close to the BSGD approach of Chen et al. (2020). BSGD is based on the following update formula:

$$\theta_{t+1} = \theta_t - a_t \nabla l\left(\theta_t; \mathcal{S}_t\right),$$

where $\mathcal{S}_t$ are an independent random batch of indices and

$$\ell\left(\theta; \mathcal{S}\right) = \mathbf{y}_{\mathcal{S}}^T \left(K_{\mathcal{S}\mathcal{S}}\left(\theta\right) + \sigma^2 I\right)^{-1} \mathbf{y}_{\mathcal{S}} + \log\left|K_{\mathcal{S}\mathcal{S}}\left(\theta\right) + \sigma^2 I\right|.$$

In our case, since $k_\alpha\left(x\right) = \phi_\alpha\left(x\right)^T \phi_\alpha\left(x\right)$, we can write $l\left(\theta; \mathcal{S}\right)$ as

$$\ell\left(\theta; \mathcal{S}\right) = \sum_i g_i\left(\theta\right) + h\left(\sum_i F_i\left(\theta\right)\right).$$

This leads to the update formula

$$\theta_{t+1} = \theta_t - a_t \sum_{i\in\mathcal{S}} \nabla\left[g_i\left(\theta_t\right) + \left\langle \tilde{F}_t^{-1}, F_i\left(\theta\right)\right\rangle\right], \text{ where } \tilde{F}_t = \sum_{i\in\mathcal{S}} F_i\left(\theta_t\right).$$

Writing the update formula of the BSGD algorithm this way emphasizes the fact that the only difference between the BSGD and the SCGD algorithms lays in the way in which $F\left(\theta_t\right)$ is approximated: using $\tilde{F}_t = \sum_{i\in\mathcal{S}} F_i\left(\theta_t\right)$ in BSGD, and $\tilde{F}_{t+1} = \left(1 - b_t\right)\tilde{F}_t + b_t \frac{n}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} F_i\left(\theta_t\right)$ in SCGD. Intuitively, the exponential smoothing that occurs SCGD should provide additional numerical robustness beyond the theoretical advantage of converging to a stationary point without an error which depends on the batch size. We can also see that in BSGD the scale of $\tilde{F}_t$ is incorrect because it misses the multiplication of $\sum_{i\in\mathcal{S}} F_i\left(\theta_t\right)$ by $\frac{n}{|\mathcal{S}|}$.

5 Experimental Result

5.1 Recovery of Known Parameters

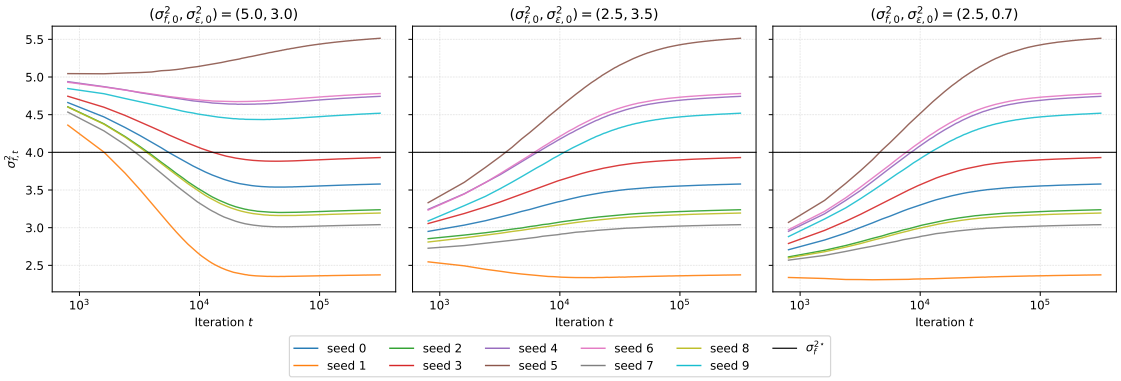
In this set of experiments, we follow the synthetic experiment of Chen et al. (2020). We sample $n = 1024$ one-dimensional inputs $x_i \overset{i.i.d.}{\sim} \mathcal{N}(0, 5^2)$ and generate responses from a zero-mean GP with two variance parameters,

$$\mathbf{y} \sim \mathcal{N}\big(0,\ \sigma_f^2 K_f + \sigma_\varepsilon^2 I_n\big),$$

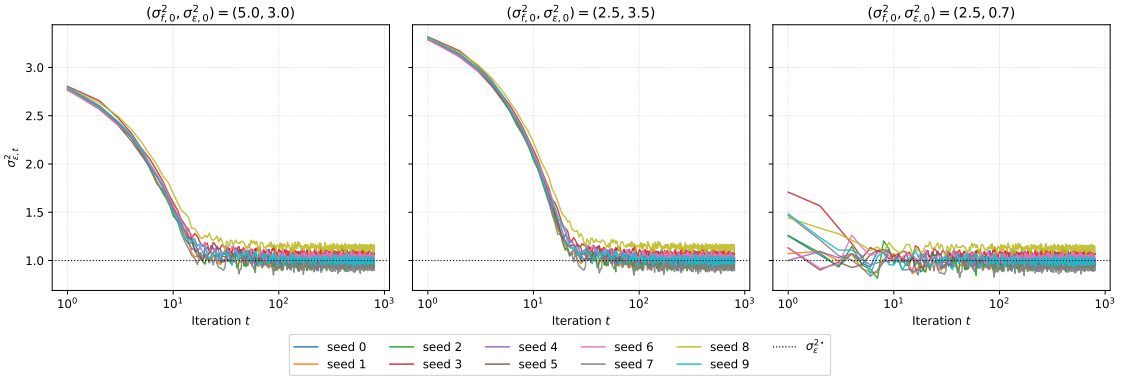
where the lengthscale is fixed to $\ell = 0.5$ and the ground-truth values are $\sigma_f^2 = 4$ and $\sigma_\varepsilon^2 = 1$.

In order to stay in the exact finite dimensional setting of this paper, we construct K_f via a fixed feature map $\varphi : \mathbb{R} \to \mathbb{R}^d$ with $d = 128$ such that

$$k_f(x, x') = \langle \varphi(x), \varphi(x')\rangle, \qquad K_f = k_f(X, X) = \varphi(X)\varphi(X)^T.$$



(a) σ_f^2



(b) σ_ε^2

Figure 1: Recovery of known variance parameters with SCGD (finite dimensional setting, SCGD+ORF). Each color corresponds to a different random seed. The horizontal black lines denote the ground-truth values $\sigma_f^2 = 4$ (top) and $\sigma_\varepsilon^2 = 1$ (bottom).

Concretely, we obtain φ using *Orthogonal Random Features* (ORF) (Yu et al., 2016) with the same lengthscale ℓ. With this, the learnable kernel magnitude can be written in the form assumed throughout the paper,

$$k_\alpha(x, x') = \phi_\alpha(x)^T \phi_\alpha(x'), \qquad \phi_\alpha(x) = \sqrt{\sigma_f^2}\, \varphi(x),$$

so here $\alpha = \sigma_f^2$ is the only kernel hyperparameter we learn, and the observation-noise variance corresponds to $\sigma^2 = \sigma_\varepsilon^2$ in our notation. We optimize $\theta = (\mathbf{w}, \sigma_f^2, \sigma_\varepsilon^2)$ using Algorithm 2 with mini-batches of size $s = 128$.

We used step-size schedules of the form $a_t = a_0\, t^{-0.3}$ and $b_t = b_0\, t^{-0.2}$ with $a_0 = 10^{-3}$ and $b_0 = 0.1$. We ran SCGD for 2×10^6 epochs and, for readability, plot the first 5×10^4 epochs (i.e., 4×10^5 stochastic updates). We repeated the experiment over 10 independent random seeds and, as in Chen et al. (2020), we considered three different initializations for the variance parameters: $(\sigma_{f,0}^2, \sigma_{\varepsilon,0}^2) \in \{(5.0, 3.0), (2.5, 3.5), (2.5, 0.7)\}$.

Figure 1 shows that SCGD converges to a neighborhood of the true parameters from all three initial points. As in Chen et al. (2020), the trajectories of σ_ε^2 are markedly more concentrated around the truth than those of σ_f^2, reflecting the fact that the signal-variance estimate exhibits a larger statistical variability. Most importantly for our purposes, this experiment is carried out in the exact inner-product setting $K_f = \varphi(X)\varphi(X)^T$, and therefore the observed behavior reflects optimization of the true objective by unbiased mini-batch gradients (rather than an artifact of submatrix bias).

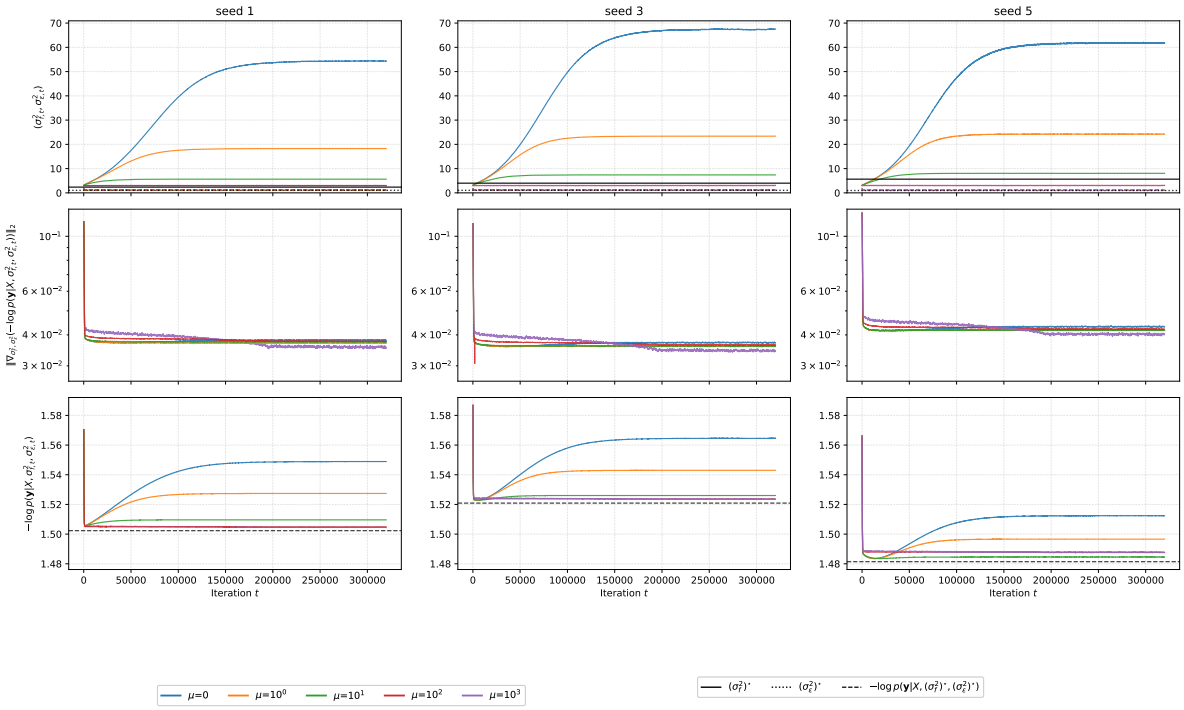


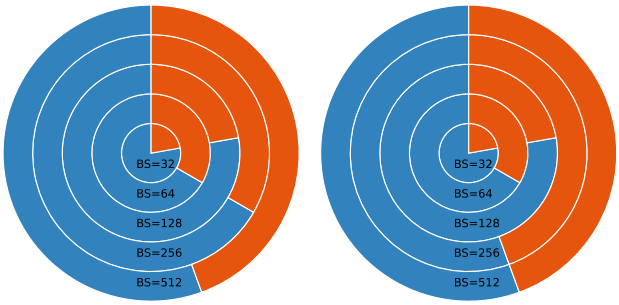
Figure 2: Effect of the penalty weight μ for MINIMAX on the synthetic recovery experiment. Each column corresponds to a different random seed. Top: MINIMAX estimates of σ_f^2 (solid) and σ_ε^2 (dashed) as a function of iteration, for several fixed choices of μ. Middle: exact full-batch gradient norm. Bottom: exact per-sample negative log marginal likelihood.

MINIMAX. We also repeat the above experiment using our MINIMAX procedure (Algorithm 1). Although the penalized formulation recovers the original objective as $\mu \to \infty$, increasing μ is well known to make penalty methods progressively ill-conditioned in practice (Nocedal & Wright, 2006). We therefore treat μ as a tunable constant and study its effect on how closely MINIMAX recovers the same stationary point as full-batch optimization of the exact objective.

We use the same dataset $(X, \mathbf{y})$ and the same ORF feature map φ (and therefore the same finite dimensional kernel $K_f = \varphi(X)\varphi(X)^\top$), and we fix the mini-batch size to $s = 128$. We initialize the hyperparameters at $(\sigma_f^2, \sigma_\varepsilon^2) = (3.0, 2.0)$ and run MINIMAX for 40,000 iterations with constant step sizes $a = 10^{-5}$ and $b = 4 \times 10^{-4}$. We sweep the penalty parameter over $\mu \in \{0, 1, 10, 10^2, 10^3\}$ and repeat each setting over three random seeds.

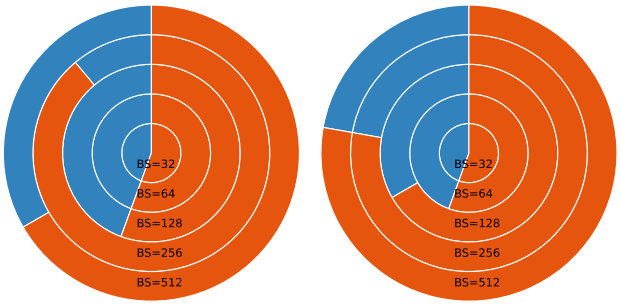
For each seed, we also compute a reference solution by running full-batch gradient descent on the *exact* negative log marginal likelihood until convergence, and denote the resulting stationary point by $(\sigma_f^2)^\star$ and $(\sigma_\varepsilon^2)^\star$.

Figure 2 shows a clear monotone effect of the penalty parameter: larger values of μ yield iterates whose variance estimates and attained negative log marginal likelihood track the full-batch reference increasingly closely. Overall, this sweep illustrates the practical role of μ in MINIMAX: increasing μ improves fidelity to the exact objective. However, taking μ too large can lead to numerical instability; while this is not observed in Figure 2, in our experiments, such issues begin to appear for μ somewhere above 10^3.



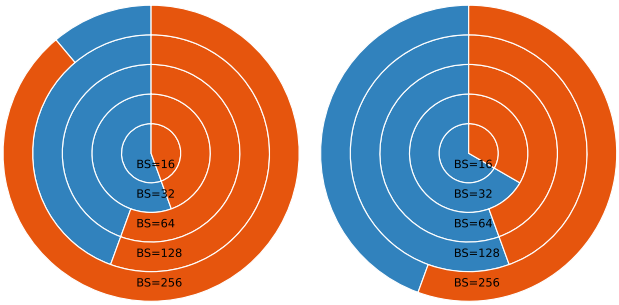
MINIMAX SCGD

Figure 3: Using the linear kernel, how many times our algorithms achieved better test RMSE than the existing methods. The left side of the pie charts (blue) is ours.



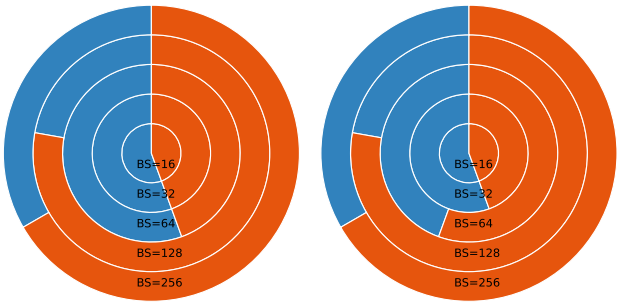
MINIMAX SCGD

Figure 4: Using the linear kernel, how many times our algorithms achieved better marginal likelihood than the existing methods. The left side of the pie charts (blue) is ours.



MINIMAX SCGD

Figure 5: Using the Gaussian kernel, how many times our algorithms achieved better test RMSE than the existing methods. The left side of the pie charts (blue) is ours.



MINIMAX SCGD

Figure 6: Using the Gaussian kernel, how many times our algorithms achieved better marginal likelihood than the existing methods. The left side of the pie charts (blue) is ours.

5.2 GP Regression

In this set of experiments, we consider a covariance function of the form

$$k_\alpha\left(x, x'\right) = k'_u\left(g_w\left(x\right), g_w\left(x'\right)\right),$$

where g_w is a neural network comprising two fully connected layers, both with an output dimension of 128 and a ReLU activation function , and k'_u, which is either the linear kernel

$$k'_u\left(z, z'\right) = \left\langle z, z'\right\rangle,$$

or the Gaussian kernel parametrized with two hyperparameters, length scale u_1 and magnitude u_2, that is

$$k'_u\left(z, z'\right) = u_2 e^{-\frac{\left\|z-z'\right\|^2}{2u_1^2}}.$$

Note that the accuracy of both competing methods depends on memory consumption. In the case of BSGD, the stochastic gradient becomes more unbiased as the batch size increases. On the other hand, the quality of the inference of the SVGP algorithm depends on the number of inducing points that need to be processed, both forward and backward, by the neural network at each iteration.

Setup: The experiment is designed to observe the influence of batch size on the results. We used 9 regression datasets from the UCI repository (Asuncion & Newman, 2007), all with a number of samples above 14,000 and fewer than 60,000. We tested the algorithms using different batch sizes: for the linear kernel, we examined batch sizes of 32, 64, 128, 256, and 512; for the Gaussian kernel, we examined batch sizes of 16, 32, 64, 128, and 256. For the SVGP, we set the number of inducing points equal to the batch size. Under a fixed per-iteration compute/memory budget, SVGP admits a trade-off between mini-batch size and the number of inducing points; a systematic study of this allocation is beyond the scope of this work. For our algorithms, the MINIMAX (Subsection 3.1.) and the SCGD (Subsection 3.2.), we approximated the Gaussian kernel k'_u using the *Random Fourier Features* method with random features φ of dimension 1000 such that

$$k'_u\left(z, z'\right) \approx \left\langle \varphi\left(z\right), \varphi\left(z'\right)\right\rangle.$$

We ran each test on five different splits of 90% for training and 10% for testing. We used AdaDelta for all methods, and for each combination of dataset, split, and method, we employed grid search to select the learning rate that achieves minimal marginal likelihood. For the SCGD algorithm, we fixed $b_t = 0.9$. Note that this is very close to 1, which means that SCGD becomes quite similar to BSGD, and much of the improvement comes just from the correct scaling of $\tilde{F}_t$. For MINIMAX , we fixed $\mu_t = 1.0$. we found it enough for achieving good results, despite the fact that, in theory, it should be increased in an outer loop. We ran each algorithm for 100 epochs and used the hyperparameters from the iteration in which the marginal likelihood achieved its minimum value.

Result with linear kernel - the exact case: It seems (Figure 4 and Figure 7) that the fact that our algorithm performs exact stochastic optimization brings a significant improvement over existing methods in optimizing the marginal likelihood. As expected, we can see that this advantage is more significant when the batch size is smaller. However, a lower negative log marginal likelihood does not always translate to a lower MSE on the test set, as we can see in Figure 3 and Figure 9. We observed that sometimes, by using a suboptimal learning rate that does not achieve the minimal loss, our algorithms can attain better results in terms of test RMSE. However, since our work is focused on optimization, we do not use procedures such as early stopping or cross validation, which could potentially improve the results from the test RMSE perspective.

Result with Gaussian kernel: Since our algorithms, MINIMAX and SCGD, use approximated Gaussian kernel based random Fourier features, we are no longer performing an exact optimization in this case. However, we can see in Figure 6 and Figure 8. that, although we use an approximated kernel, when the

restrictions on batch size are high, our methods perform better than the existing methods in optimizing the marginal likelihood. The advantages of our methods in optimization are also reflected in the test error (see Figure 5 and Figure 10).

We see that in both cases, the finite dimensional RKHS and the infinite dimensional RKHS, our approach, unlike the existing inference algorithms, shows no degradation in the results of our algorithms with the decrease in batch size. This property can be vital for inference on weak edge devices where memory restrictions limit the possible batch size.

Compute power used: This experiment required roughly 20 days of g5.48xlarge machine time on AWS.

5.3 Attention-Based GP Regression

To better illustrate when our methods are most useful, we repeat the linear-kernel GP regression experiment on a subset of the UCI datasets, but replace the feature map used by the kernel. Concretely, as in the previous experiment we define a linear kernel on top of learned representations,

$$k(x, x') = \langle g_w(x),\, g_w(x') \rangle,$$

and we keep the GP model; the only change is that g_w is now an attention-based network rather than the two-layer MLP used earlier. This setting is particularly relevant under memory constraints: the activation memory footprint of self-attention grows with the mini-batch size, making large batches substantially more expensive. We therefore compare how peak GPU memory increases with the batch size (Figure 13). In addition, to evaluate optimization speed, we plot the best (lowest) negative log marginal likelihood achieved up to a given training epoch (Figure 11) and up to a given wall-clock time (Figure 12). Overall, we observe that in this attention-based setting, SCGD in particular often provides a favorable trade-off between memory usage and convergence speed relative to the baselines.

In this experiment, we used the ADAM optimizer with a grid search of the learning rates.

The network (attention feature map). Let p_{in} denote the number of input coordinates. The attention feature map $g_w : \mathbb{R}^{p_{\text{in}}} \to \mathbb{R}^d$ operates on the standardized input $x \in \mathbb{R}^{p_{\text{in}}}$ as a feature-wise Transformer encoder. We first apply a LayerNorm over the p_{in} input coordinates and then tokenize each scalar feature into a d-dimensional token using a feature-specific affine embedding: for $j \in \{1, \ldots, p_{\text{in}}\}$,

$$t_j \;=\; x_j\, w_j + b_j \in \mathbb{R}^d,$$

where $w_j, b_j \in \mathbb{R}^d$ are learned parameters. We prepend a learned `[CLS]` token and feed the resulting sequence of length $p_{\text{in}} + 1$ through $L = 2$ Transformer encoder layers with model width $d = d_{\text{model}} = 32$, $H = 4$ attention heads per layer, and a position-wise feed-forward subnetwork of width $d_{\text{ff}} = 2d$ (i.e., mlp_ratio = 2). Each encoder layer uses a pre-LayerNorm (pre-norm) configuration and GELU nonlinearities. Dropout is disabled (dropout= 0) to keep the feature map deterministic. The final representation $g_w(x)$ is the LayerNorm-normalized `[CLS]` embedding after the last encoder layer. We do not use positional encodings or attention masks, since the tokens correspond to input dimensions rather than an ordered sequence. For initialization, the tokenizer weights are sampled with Xavier-uniform initialization, tokenizer biases are initialized to zero, and the `[CLS]` token is initialized from $\mathcal{N}(0, 0.02^2)$.

6 Conclusion

In many cases, the covariance function of a GP is defined as an inner product between features of finite and moderate dimensions. In such cases, the problem of minimizing the negative-log-marginal-likelihood takes the shape of a standard ridge regression problem with a non-standard regularization term in the form of the log-determinant of the covariance matrix of the representations plus $\sigma^2 I$. In this work, we developed two techniques that enable the solution of this problem using stochastic mini-batches, which, unlike existing methods, do not depend on large batches to be exact. When inference involves forward and backward passes of a feedforward neural net, this property is of great importance and can enable such inference architectures on weak edge devices.

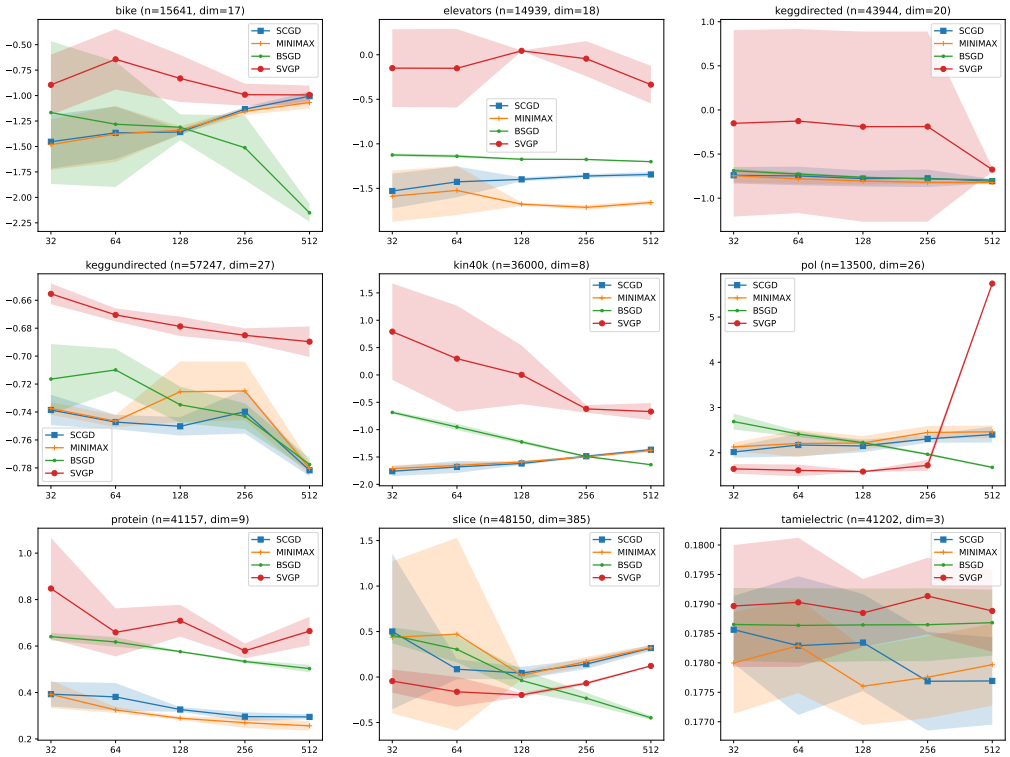


Figure 7: The negative marginal log likelihood divided by n as a function of the batch size when using the

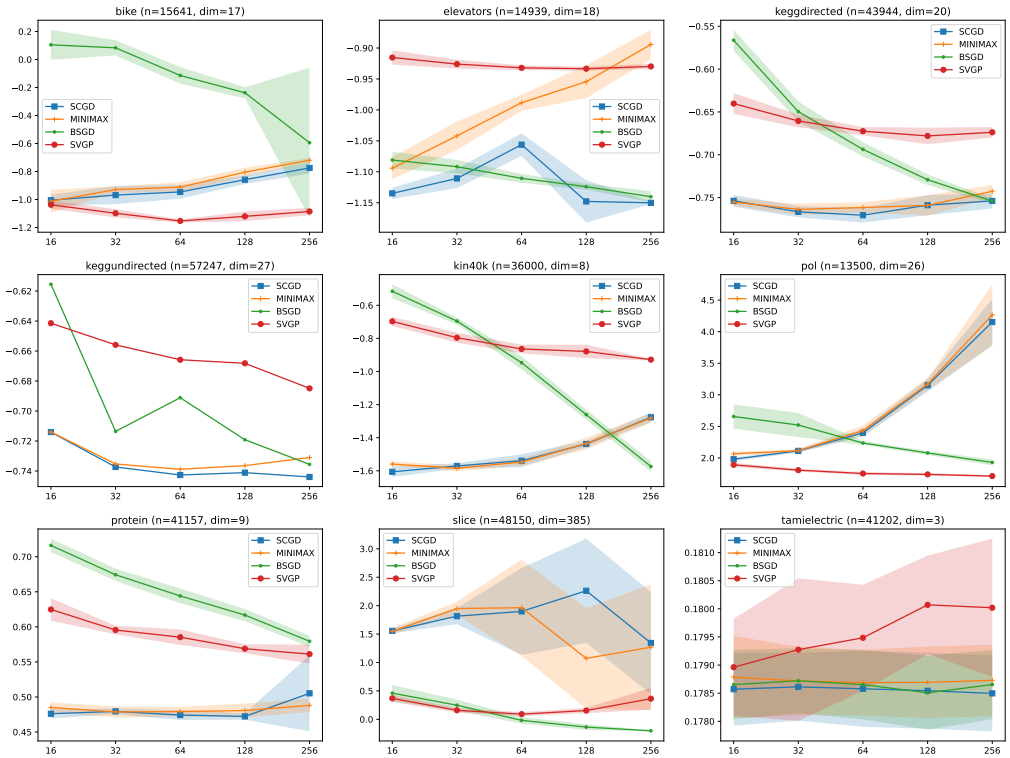


Figure 8: The negative marginal log likelihood divided by n as a function of the batch size when using the Gaussian kernel, plotted with standard deviation margins.

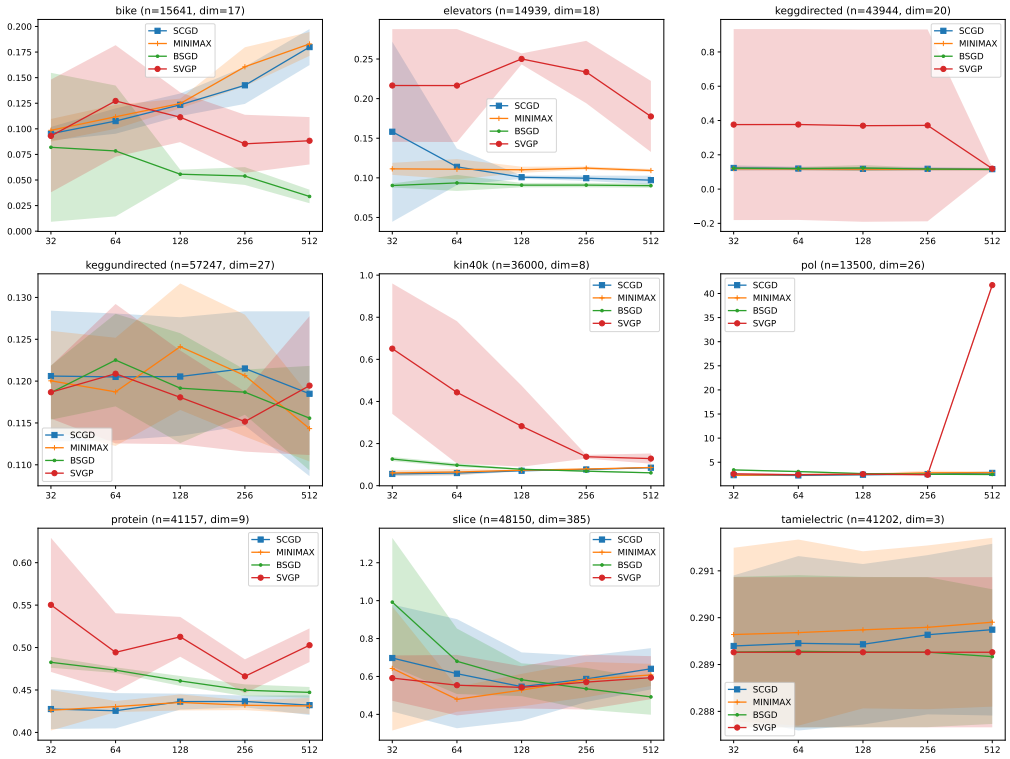


Figure 9: The RMSE as a function of the batch size when using the linear kernel, plotted with standard

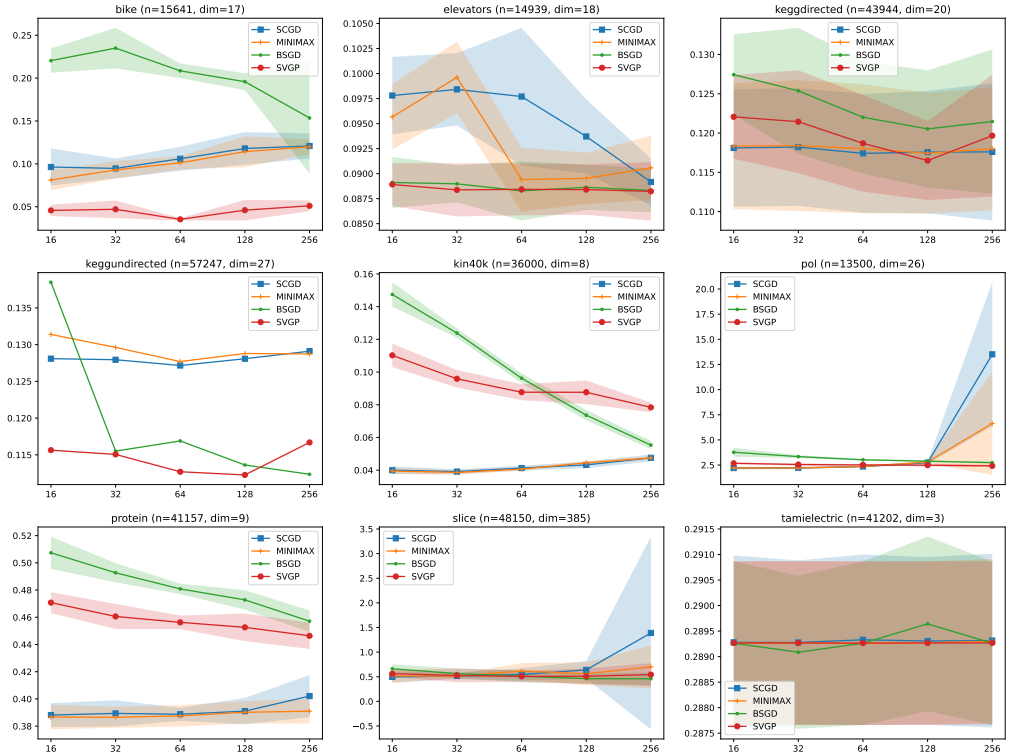


Figure 10: The RMSE as a function of the batch size when using the Gaussian kernel, plotted with standard deviation margins.

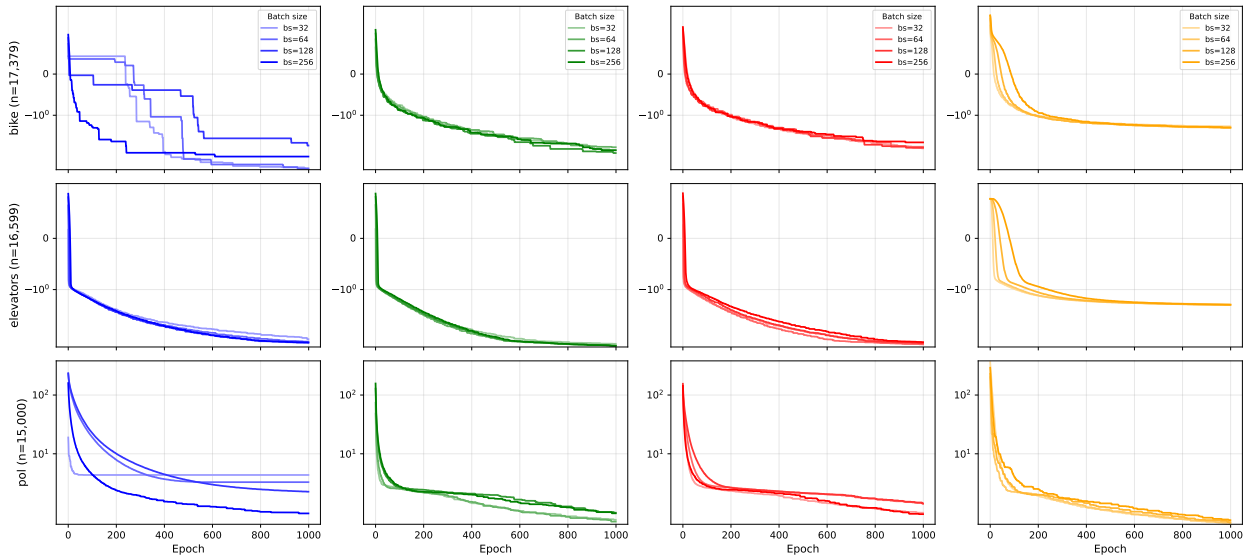


Figure 11: Negative log marginal likelihood as a function of training epoch.

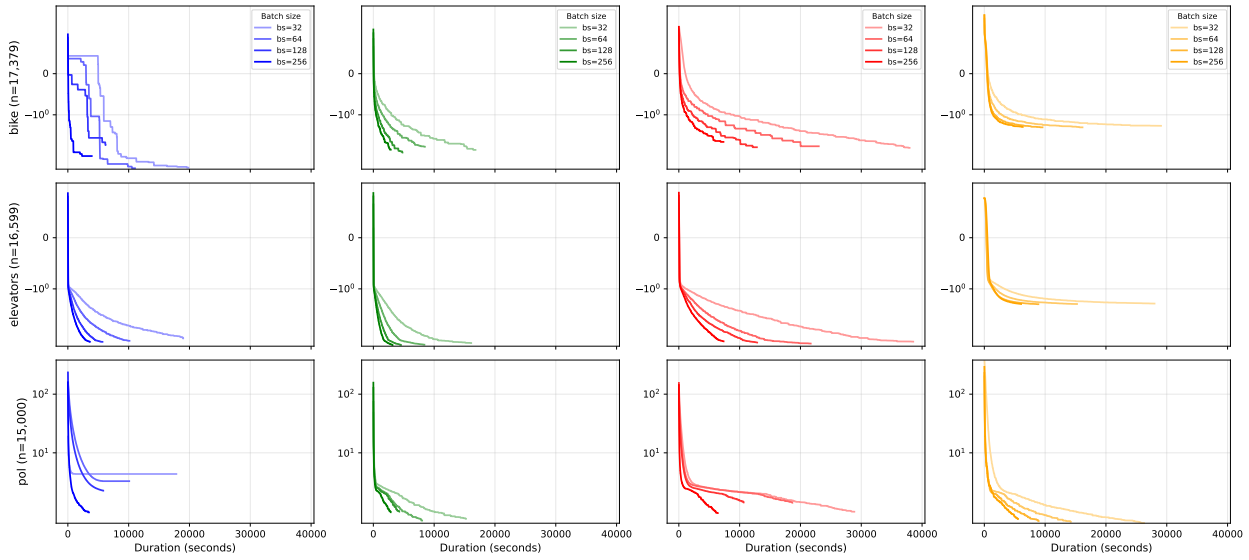


Figure 12: Negative log-likelihood as a function of duration.

Limitations: Both MINIMAX (Algorithm 1) and SCGD (Algorithm 2) maintain and repeatedly factorize a $d \times d$ matrix (either A or $\tilde{F}_t$). This yields an additional per-iteration cost of $O(d^3)$ time (e.g., eigenvalue decomposition / Cholesky) and $O(d^2)$ memory, beyond the cost of computing $\phi_\alpha(x)$ and its Jacobian. Consequently, our methods are most attractive when the feature dimension $d = \dim\phi_\alpha(x)$ is *moderate* and when memory constraints force small mini-batches, but they can be less competitive when d must be very large (e.g., when using a large number of random features to approximate an infinite-dimensional kernel). In addition, the penalized formulation in MINIMAX introduces a trade-off through μ: larger μ enforces $A \approx F(\theta)$ more strongly but can make the optimization increasingly ill-conditioned, while smaller μ improves numerical stability but may yield stationary points that are farther from those of the constrained objective. Finally, when resources allow large batches (or many inducing points), biased or variational baselines can often achieve comparable performance at lower linear-algebra cost, reducing the practical advantage of exact mini-batch gradients.

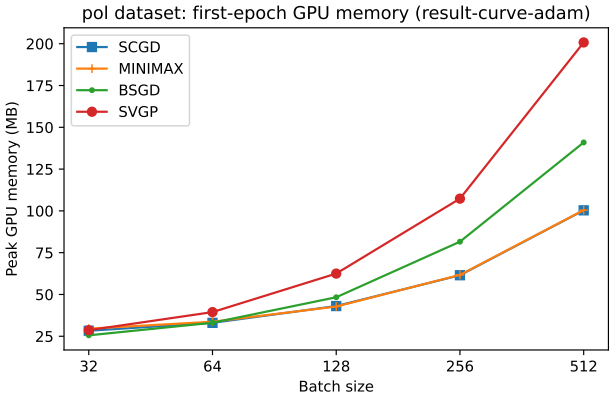


Figure 13: Peak GPU memory usage as a function of batch size.

References

MA Alvarez, L Rosasco, and ND Lawrence. Kernels for vector-valued functions: a review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, 2012.

Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

Radu Ioan Boţ and Axel Böhm. Alternating proximal-gradient steps for (stochastic) nonconvex-concave minimax problems. *SIAM Journal on Optimization*, 33(3):1884–1913, 2023.

Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3338–3345. IEEE, 2016.

Hao Chen, Lili Zheng, Raed Al Kontar, and Garvesh Raskutti. Stochastic gradient descent in correlated settings: A study on gaussian processes. *Advances in Neural Information Processing Systems*, 33, 2020.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, pp. 282–290, Arlington, Virginia, USA, 2013. AUAI Press.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pp. 351–360. PMLR, 2015.

Trong Nghia Hoang, Quang Minh Hoang, and Bryan Kian Hsiang Low. A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data. In *International Conference on Machine Learning*, pp. 569–578. PMLR, 2015.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

Neal Jean, Sang Michael Xie, and Stefano Ermon. Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/9d28de8ff9bb6a3fa41fddfdc28f3bc1-Paper.pdf`.

Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6083–6093. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/lin20a.html`.

Haitao Liu, Jianfei Cai, and Yew-Soon Ong. Remarks on multi-output gaussian process regression. *Knowledge-Based Systems*, 144:102–121, 2018a.

Haitao Liu, Yew-Soon Ong, and Jianfei Cai. A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. *Structural and Multidisciplinary Optimization*, 57(1):393–416, 2018b.

Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

Luo Luo, Haishan Ye, Zhichao Huang, and Tong Zhang. Stochastic recursive gradient descent ascent for stochastic nonconvex-strongly-concave minimax problems. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20566–20577. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/ecb47fbb07a752413640f82a945530f8-Paper.pdf`.

Duc-Trung Nguyen, Maurizio Filippone, and Pietro Michiardi. Exact gaussian process regression with distributed computations. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 1286–1295, 2019.

Trung V Nguyen, Edwin V Bonilla, et al. Collaborative multi-output gaussian processes. In *UAI*, pp. 643–652. Citeseer, 2014.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Joaquin Quinonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959, 2005.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'07, pp. 1177–1184, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pp. 63–71. Springer, 2003.

Andrzej P. Ruszczyński. *Nonlinear Optimization*. Princeton University Press. ISBN 978-1-4008-4105-9.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

Paz Fink Shustin and Haim Avron. Gauss-legendre features for gaussian process regression. *arXiv preprint arXiv:2101.01137*, 2021.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In Johannes Fürnkranz and Thorsten Joachims (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 1015–1022, Haifa, Israel, June 2010. Omnipress. URL `http://www.icml2010.org/papers/422.pdf`.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial intelligence and statistics*, pp. 567–574. PMLR, 2009.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32:14648–14659, 2019.

Mengdi Wang, Ethan X Fang, and Han Liu. Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Mathematical Programming*, 161(1-2):419–449, 2017.

Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning*, pp. 1775–1784. PMLR, 2015.

Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. *Advances in Neural Information Processing Systems*, 29:2586–2594, 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pp. 370–378. PMLR, 2016b.

Zichao Yang, Andrew Wilson, Alex Smola, and Le Song. A la carte–learning fast kernels. In *Artificial Intelligence and Statistics*, pp. 1098–1106. PMLR, 2015.

Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. *Advances in neural information processing systems*, 29, 2016.

Jing Zhao and Shiliang Sun. Variational dependent multi-output gaussian process dynamical systems. *The Journal of Machine Learning Research*, 17(1):4134–4169, 2016.

A Complexity Analysis

We analyze the computational complexity of each of the algorithms. We ignore the complexity of the computation of $\phi_\alpha(x_i)$ and $\frac{\phi_\alpha(x_i)}{\partial\alpha}$ as they are the same for all algorithms. We let d be the dimension of $\phi_\alpha(x_i)$ and s be the batch size.

A.1 Minimax

Computational Complexity Forward

Component	Complexity
g_i	$O(d)$
F_i	$O(d^2)$
$h(A)$	$O(d^3)$
$\frac{\left\langle B, \frac{1}{n}A - F_i(\theta)\right\rangle}{\|A\|}$	$O(d^2)$
Total	$O(d^3) + O(sd^2)$

Computational Complexity Backward

Component	Complexity
$\frac{\partial g_i}{\partial w}$	$O(d)$
$\frac{\partial g_i}{\partial \phi_i}$	$O(d)$
$\frac{\partial h}{A} = A^{-1}$	$O(d^3)$
$\frac{\partial}{\partial B}\frac{\left\langle B, \frac{1}{n}A\right\rangle}{\|A\|} = \frac{\frac{1}{n}A}{\|A\|}$	$O(d^2)$
$\frac{\partial}{\partial A}\frac{\left\langle B, \frac{1}{n}A\right\rangle}{\|A\|} = \frac{\frac{1}{n}B\|A\| - \left\langle B, \frac{1}{n}A\right\rangle \frac{A}{\|A\|}}{\|A\|^2}$	$O(d^2)$
$\frac{\partial}{\partial \phi_i}\frac{\langle B, F_i\rangle}{\|A\|} = \frac{2B\phi_i}{\|A\|}$	$O(d^2)$
$proj_{\{\sigma^2 I_d \preceq M \preceq A_{\max}\}}(A)$	$O(d^3)$
Total	$O(d^3) + O(sd^2)$

Memory Complexity Forward

Component	Complexity
g_i	$O(d)$
A, B, F_i	$O(d^2)$
Total	$O(d^2)$

Memory Complexity Backward

Component	Complexity
$\frac{\partial g_i}{\partial w}$	$O(d)$
$\frac{\partial g_i}{\partial \phi_i}$	$O(d)$
$\frac{\partial h}{A} = A^{-1}$	$O(d^2)$
$\frac{\partial}{\partial B}\frac{\left\langle B, \frac{1}{n}A\right\rangle}{\|A\|} = \frac{\frac{1}{n}A}{\|A\|}$	$O(d^2)$
$\frac{\partial}{\partial A}\frac{\left\langle B, \frac{1}{n}A\right\rangle}{\|A\|} = \frac{\frac{1}{n}B\|A\| - \left\langle B, \frac{1}{n}A\right\rangle \frac{A}{\|A\|}}{\|A\|^2}$	$O(d^2)$
$\frac{\partial}{\partial \phi_i}\frac{\langle B, F_i\rangle}{\|A\|} = \frac{2B\phi_i}{\|A\|}$	$O(d)$
$proj_{\{\sigma^2 I_d \preceq M \preceq A_{\max}\}}(A)$	$O(d^2)$
Total	$O(d^2) + O(sd)$

A.2 SCGD

Computational Complexity Forward

Component	Complexity
g_i	$O(d)$
$\tilde{F}^{-1}$	$O\left(d^3\right)$
F_{i}	$O\left(d^2\right)$
Total	$O\left(d^3\right)$

Computational Complexity Backward

Component	Complexity
$\frac{\partial g_i}{\partial w}$	$O(d)$
$\frac{\partial g_i}{\partial \phi_i}$	$O(d)$
$\frac{\partial}{\partial \phi_i}\left\langle \tilde{F}_t^{-1}, F_i\right\rangle = 2\phi_i \tilde{F}_t^{-1}$	$O\left(d^2\right)$
Total	$O\left(sd^2\right)$

Memory Complexity Forward

Component	Complexity
g_i	$O(d)$
F_i	$O\left(d^2\right)$
$\tilde{F}$	$O\left(d^2\right)$
Total	$O\left(d^2\right)$

Memory Complexity Backward

Component	Complexity
$\frac{\partial g_i}{\partial w}$	$O(d)$
$\frac{\partial g_i}{\partial \phi_i}$	$O(d)$
$\frac{\partial}{\partial \phi_i}\left\langle \tilde{F}_t^{-1}, F_i\right\rangle = 2\phi_i \tilde{F}_t^{-1}$	$O(d)$
Total	$O(sd)$

A.3 BSGD

$$\frac{\partial}{\partial Z}\left[\mathbf{y}^T\left(K+\sigma^2 I\right)^{-1}\mathbf{y}+\log\left|K+\sigma^2 I\right|\right] = -2\left(K+\sigma^2 I\right)^{-1}\mathbf{y}\mathbf{y}^T\left(K+\sigma^2 I\right)^{-1}Z + 2\left(K+\sigma^2 I\right)^{-1}Z$$

Computational complexity: $O\left(s^3\right)+O\left(s^2 d\right)$.

Memory complexity: $O\left(s^2\right)+O(sd)$

B Missing Proofs

Theorem 1. *For all $\lambda > 0$, $V \in \mathbb{R}^{n\times d}$, and $\mathbf{b} \in \mathbb{R}^d$ we have*

$$\mathbf{b}^T\left(VV^T+\lambda I\right)^{-1}\mathbf{b} = \min_{\mathbf{w}} \frac{1}{\lambda}\left\|V\mathbf{w}-\mathbf{b}\right\|^2 + \left\|\mathbf{w}\right\|^2.$$

Proof. Let $f(\mathbf{w}) = \frac{1}{\lambda}\|V\mathbf{w} - \mathbf{b}\|^2 + \|\mathbf{w}\|^2$. Since $\lambda > 0$, f is strongly convex and has a unique minimizer. The first-order condition gives

$$\nabla f(\mathbf{w}) = \frac{2}{\lambda}V^T(V\mathbf{w} - \mathbf{b}) + 2\mathbf{w} = 0 \quad \Longleftrightarrow \quad (V^TV + \lambda I_d)\mathbf{w} = V^T\mathbf{b},$$

hence

$$\hat{\mathbf{w}} = (V^TV + \lambda I_d)^{-1}V^T\mathbf{b}.$$

Now evaluate at $\hat{\mathbf{w}}$:

$$\begin{aligned} f(\hat{\mathbf{w}}) &= \frac{1}{\lambda}\|V\hat{\mathbf{w}} - \mathbf{b}\|^2 + \|\hat{\mathbf{w}}\|^2 \\ &= \frac{1}{\lambda}\Big(\hat{\mathbf{w}}^TV^TV\hat{\mathbf{w}} - 2\mathbf{b}^TV\hat{\mathbf{w}} + \mathbf{b}^T\mathbf{b} + \lambda\hat{\mathbf{w}}^T\hat{\mathbf{w}}\Big) \\ &= \frac{1}{\lambda}\Big(\mathbf{b}^T\mathbf{b} - \big(2\mathbf{b}^TV\hat{\mathbf{w}} - \hat{\mathbf{w}}^T(V^TV + \lambda I_d)\hat{\mathbf{w}}\big)\Big). \end{aligned}$$

Using $(V^TV + \lambda I_d)\hat{\mathbf{w}} = V^T\mathbf{b}$, we have $\hat{\mathbf{w}}^T(V^TV + \lambda I_d)\hat{\mathbf{w}} = \mathbf{b}^TV\hat{\mathbf{w}}$, so

$$f(\hat{\mathbf{w}}) = \frac{1}{\lambda}\big(\mathbf{b}^T\mathbf{b} - \mathbf{b}^TV\hat{\mathbf{w}}\big) = \frac{1}{\lambda}\mathbf{b}^T\Big(I_n - V(V^TV + \lambda I_d)^{-1}V^T\Big)\mathbf{b}.$$

Finally, by the matrix inversion lemma,

$$(VV^T + \lambda I_n)^{-1} = \frac{1}{\lambda}\Big(I_n - V(V^TV + \lambda I_d)^{-1}V^T\Big),$$

so $f(\hat{\mathbf{w}}) = \mathbf{b}^T(VV^T + \lambda I_n)^{-1}\mathbf{b}$. □

B.1 SCGD

Stochastic oracle and the notation $\tilde{\nabla}$. Following Wang et al. (2017), we consider compositional objectives $\ell(x) = (v \circ u)(x)$ with $u : \mathbb{R}^p \to \mathbb{R}^m$ and $v : \mathbb{R}^m \to \mathbb{R}$. We assume access to a sampling oracle that, given x, returns: (i) a random sample $u_\omega(x) \in \mathbb{R}^m$ with $\mathbf{E}[u_\omega(x)] = u(x)$; (ii) a random matrix $\tilde{\nabla}u_\omega(x) \in \mathbb{R}^{p\times m}$ whose jth column is a (stochastic) gradient/subgradient of the scalar map $x \mapsto [u_\omega(x)]_j$; and (iii) a (possibly stochastic) gradient $\nabla v_\nu(y)$ of the outer function. When u_ω is differentiable, we may take $\tilde{\nabla}u_\omega(x) = \nabla u_\omega(x)$. We also follow Wang et al. (2017) in the notation $\partial\ell(x)$: if ℓ is convex, $\partial\ell(x)$ denotes the convex subdifferential; if ℓ is differentiable (possibly nonconvex), we interpret $\partial\ell(x) = \{\nabla\ell(x)\}$.

Assumption 1 (Wang et al., 2017, Assumption 1)**.** *Let $\ell(x) = (v \circ u)(x)$, where $u : \mathbb{R}^p \to \mathbb{R}^m$ and $v : \mathbb{R}^m \to \mathbb{R}$. Let $C_u, C_v, V_u, L_v > 0$ be positive scalars.*

(i) The outer function v is continuously differentiable, the inner function u is continuous, the feasible set $X \subseteq \mathbb{R}^p$ is closed and convex, and there exists at least one optimal solution $x^\star$ to the problem $\min_{x\in X}\ell(x)$.

(ii) The random variables $(\omega_0, \nu_0), (\omega_1, \nu_1), \ldots$ are i.i.d. with a complete probability measure, and

$$\mathbf{E}\big[u_{\omega_0}(x)\big] = u(x), \qquad \mathbf{E}\Big[\tilde{\nabla}u_{\omega_0}(x)\,\nabla v_{\nu_0}(u(x))\Big] \in \partial\ell(x), \qquad \forall x \in X.$$

(iii) The function $u(\cdot)$ is Lipschitz continuous with parameter C_u, and the samples $u_{\omega_k}(\cdot)$ and $\tilde{\nabla}u_{\omega_k}(\cdot)$ have bounded second moments such that, with probability 1,

$$\mathbf{E}\big[\|\tilde{\nabla}u_{\omega_0}(x)\|^2 \mid \nu_0\big] \le C_u, \qquad \mathbf{E}\big[\|u_{\omega_0}(x) - u(x)\|^2\big] \le V_u, \qquad \forall x \in X.$$

(iv) The functions v and v_ν have Lipschitz continuous gradients such that, with probability 1,

$$\mathbf{E}\big[\|\nabla v_{\nu_0}(y)\|^2\big] \le C_v, \qquad \|\nabla v_{\nu_0}(y) - \nabla v_{\nu_0}(\bar{y})\| \le L_v\|y - \bar{y}\|, \qquad \forall y, \bar{y} \in \mathbb{R}^m.$$

Theorem 2 (Wang et al. 2017, Theorem 4: Convergence rate of basic SCGD for nonconvex problems)**.** *Assume Assumption 1 holds, ℓ is Lipschitz differentiable (i.e., $\nabla\ell$ is L_ℓ-Lipschitz), and $X = \mathbb{R}^p$. Run* basic SCGD *(Algorithm 1 of Wang et al., 2017) with step sizes $\alpha_k = k^{-a}$ and $\beta_k = k^{-b}$ where $a, b \in (0, 1)$ satisfy $b < a < 2b$. Define*

$$T_\varepsilon := \min\Big\{k \geq 0 : \inf_{0\leq t\leq k} \mathbf{E}\big[\|\nabla\ell(x_t)\|^2\big] \leq \varepsilon\Big\}.$$

Then $T_\varepsilon = \mathcal{O}(\varepsilon^{-1/q})$, where $q = \min\{1-a,\ a-b,\ 2b-a,\ a\}$. In particular, $(a, b) = (\frac{3}{4}, \frac{1}{2})$ yields $T_\varepsilon = \mathcal{O}(\varepsilon^{-4})$.

Theorem 3 (SCGD for our GP objective)**.** *Let $\theta = (\mathbf{w}, \alpha, \beta) \in \mathbb{R}^p$ and define*

$$\sigma^2(\beta) = \sigma^2_{\min} + \exp(\beta) \quad (> 0).$$

Recall

$$\ell(\theta) = g(\theta) + \log\det(F(\theta)), \qquad F(\theta) = \sum_{i=1}^n u_{2,i}(\theta), \qquad g(\theta) = \sum_{i=1}^n u_{1,i}(\theta),$$

where, for $i = 1, \ldots, n$,

$$\begin{aligned} u_{1,i}(\theta) &:= \sigma(\beta)^{-2}\big(\phi_\alpha(x_i)^\top \mathbf{w} - y_i\big)^2 \ + \ \tfrac{1}{n}\|\mathbf{w}\|^2 \ + \ \tfrac{1}{n}(n-d)\log(\sigma^2(\beta)), \\ u_{2,i}(\theta) &:= \phi_\alpha(x_i)\phi_\alpha(x_i)^\top \ + \ \tfrac{1}{n}\sigma^2(\beta)\, I_d. \end{aligned}$$

Let $\mathrm{vec}(\cdot)$ *stack the entries of a matrix into a vector and let* unvec *denote its inverse. Define the inner map*

$$u(\theta) = \big(u_1(\theta),\, \mathrm{vec}(u_2(\theta))\big), \quad \textit{where} \quad u_1(\theta) := \sum_{i=1}^n u_{1,i}(\theta),\ \ u_2(\theta) := \sum_{i=1}^n u_{2,i}(\theta),$$

and the outer map

$$v(s, z) = s + \log\det(\mathrm{unvec}(z)), \qquad \mathrm{dom}(v) = \{(s, z) : \mathrm{unvec}(z) \succ 0\}.$$

Then $\ell(\theta) = (v \circ u)(\theta)$ and $X = \mathbb{R}^p$.

Assume:

(H1) *$x \mapsto \phi_\alpha(x)$ is C^1 in α and both $\phi_\alpha(x)$ and $\partial\phi_\alpha(x)/\partial\alpha$ are uniformly bounded on the (random) set of iterates $\{\theta_t\}_{t\geq 0}$.*

(H2) *Mini-batches $\mathcal{S}_t$ are drawn i.i.d. uniformly from $\{1, \ldots, n\}$.*

Run basic SCGD specialized to this composite:

$$\tilde{u}_{2,t+1} = (1 - b_t)\tilde{u}_{2,t} \ + \ b_t \frac{n}{|\mathcal{S}_{t+1}|} \sum_{i\in\mathcal{S}_{t+1}} u_{2,i}(\theta_t),$$

$$\theta_{t+1} = \theta_t - \ a_t \frac{n}{|\mathcal{S}_{t+1}|} \sum_{i\in\mathcal{S}_{t+1}} \nabla_\theta\big[u_{1,i}(\theta_t) + \big\langle \tilde{u}^{-1}_{2,t+1},\, u_{2,i}(\theta_t)\big\rangle\big]\,.$$

With stepsizes $a_t = t^{-a}$ and $b_t = t^{-b}$ satisfying $b < a < 2b$, Assumption 1 holds for (v, u) above by taking $\tilde{\nabla} u_\omega(\theta) = \nabla u_\omega(\theta)$ (Jacobian of the mini-batch inner map), and the nonconvex SCGD rate (Theorem 2) applies:

$$\min_{0\leq t\leq T} \mathbf{E}\big[\|\nabla\ell(\theta_t)\|^2\big] \ \leq \ C\,T^{-q}, \qquad q = \min\{\,1 - a,\ a - b,\ 2b - a,\ a\,\}.$$

In particular, with $(a, b) = (\frac{3}{4}, \frac{1}{2})$, the basic SCGD iterates reach an ε-stationary point in $\mathcal{O}(\varepsilon^{-4})$ iterations.

Proof. We verify Assumption 1 for the pair (v, u).

A1(i): regularity and feasibility. For any θ,

$$u_2(\theta) = \sum_{i=1}^{n} \phi_\alpha(x_i)\phi_\alpha(x_i)^\top + \sigma^2(\beta) I_d \succeq \sigma^2(\beta)\, I_d \succeq \sigma_{\min}^2 I_d,$$

so $\log\det(\cdot)$ is C^∞ on the range of u_2 and its matrix-gradient satisfies

$$\nabla_A \log\det(A) = A^{-\top} \quad (\text{and } = A^{-1} \text{ if } A \text{ is symmetric}).$$

Therefore $v(s, z) = s + \log\det(\mathrm{unvec}(z))$ is C^1 on range(u). Also, each $u_{1,i}(\theta)$ and $u_{2,i}(\theta)$ is continuous in θ under (H1) and the smooth map $\beta \mapsto \sigma^2(\beta)$, hence u is continuous.

The feasible set in Theorem 3 is $X = \mathbb{R}^p$, which is closed and convex. Existence of an optimal solution is assumed as in Assumption 1(i).

A1(ii): unbiased oracle and chain rule in expectation. Let $\mathcal{S}$ be a uniform i.i.d. mini-batch and define the inner sample map

$$u_\omega(\theta) \equiv \tilde{u}(\theta; \mathcal{S}) = \Big(\frac{n}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} u_{1,i}(\theta),\ \mathrm{vec}\Big(\frac{n}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} u_{2,i}(\theta)\Big)\Big).$$

Then $\mathbf{E}[\tilde{u}(\theta; \mathcal{S})] = u(\theta)$ by uniform sampling.

Definition of $\tilde{\nabla}$. Because $\tilde{u}(\cdot; \mathcal{S})$ is differentiable in our setting, we take

$$\tilde{\nabla} u_\omega(\theta) \ := \ \nabla_\theta \tilde{u}(\theta; \mathcal{S}),$$

the Jacobian of the mini-batch inner map.

Now write $A = \mathrm{unvec}(z)$. Then

$$\nabla v(s, z) = \big(1,\ \mathrm{vec}(A^{-\top})\big),$$

and by the usual chain rule,

$$\nabla \ell(\theta) = \nabla_\theta u(\theta)\, \nabla v(u(\theta)).$$

Taking expectations yields

$$\mathbf{E}\big[\tilde{\nabla} u_\omega(\theta)\, \nabla v(u(\theta))\big] = \nabla_\theta u(\theta)\, \nabla v(u(\theta)) = \nabla \ell(\theta) \in \partial \ell(\theta),$$

which is exactly Assumption 1(ii) (and here the inclusion is equality since ℓ is smooth).

A1(iii): Lipschitz u and bounded second moments. Under (H1), along the iterates the quantities $\phi_\alpha(x_i)$ and $\partial\phi_\alpha(x_i)/\partial\alpha$ are uniformly bounded, and $\sigma^2(\beta) \geq \sigma_{\min}^2$. Hence the Jacobians $\nabla_\theta u_{1,i}(\theta)$ and $\nabla_\theta u_{2,i}(\theta)$ are uniformly bounded along the iterates, which implies u is Lipschitz on the set visited by $\{\theta_t\}$. Moreover, $\tilde{u}(\theta; \mathcal{S})$ and $\nabla_\theta \tilde{u}(\theta; \mathcal{S})$ are averages of bounded quantities, hence they have uniformly bounded second moments (in particular, the conditional moment bound in Assumption 1(iii) holds).

A1(iv): Lipschitz ∇v on range(u). Over range(u) we have $\lambda_{\min}(A) \geq \sigma_{\min}^2$. For any $A, B \succ 0$,

$$\|A^{-1} - B^{-1}\| = \|A^{-1}(B - A)B^{-1}\| \leq \|A^{-1}\|\, \|B^{-1}\|\, \|A - B\|.$$

Since $\|A^{-1}\| \leq 1/\sigma_{\min}^2$ and $\|B^{-1}\| \leq 1/\sigma_{\min}^2$ on the range,

$$\|A^{-1} - B^{-1}\| \leq \sigma_{\min}^{-4}\, \|A - B\|.$$

Thus $A \mapsto A^{-\top}$ is Lipschitz on range(u), and consequently ∇v is Lipschitz there.

Finally, $\nabla \ell$ is Lipschitz on the set visited by $\{\theta_t\}$ because it is the product/chain of bounded and Lipschitz maps ∇u and ∇v on that set. Therefore the conditions of Theorem 2 apply and the stated rate follows. □

Algorithm 3 Proximal alternating GDA *(Algorithm 2.1 in Boţ & Böhm, 2023)*

Require: Initial point $(x_0, y_0) \in \mathbb{R}^d \times \mathbb{R}^n$, stepsizes $\eta_x, \eta_y > 0$

1: **for** $k = 0, 1, 2, \dots$ **do**
2: $\quad x_{k+1} \leftarrow \operatorname{prox}_{\eta_x f}\big(x_k - \eta_x\, G_x(x_k, y_k)\big)$
3: $\quad y_{k+1} \leftarrow \operatorname{prox}_{\eta_y h}\big(y_k + \eta_y\, G_y(x_{k+1}, y_k)\big)$
4: **end for**

Here G_x and G_y denote the appropriate (sub)gradients or stochastic (sub)gradient estimators in the deterministic or stochastic settings, respectively.

B.2 Minimax

Assumption 2 (Nonconvex–concave structure and smoothness (Assump. 3 in Boţ & Böhm, 2023))**.** *Let $\Phi : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ be the coupling term in the minimax template $\min_x \max_y \{f(x) + \Phi(x, y) - h(y)\}$.*

(i) For every x, the map $y \mapsto \Phi(x, y)$ is concave and $L_{\nabla\Phi}$–smooth: $\|\nabla_y \Phi(x, y) - \nabla_y \Phi(x, y')\| \le L_{\nabla\Phi} \|y - y'\|$.

(ii) For every y, the map $x \mapsto \Phi(x, y)$ is ρ–weakly convex, i.e., $x \mapsto \Phi(x, y) + \frac{\rho}{2}\|x\|^2$ is convex.

Assumption 3 (Lower bounded regularized max function (Assump. 4 in Boţ & Böhm, 2023))**.** *The regularized max function $\psi(x) := f(x) + \varphi(x)$ with $\varphi(x) := \max_y \{\Phi(x, y) - h(y)\}$ is bounded below: $\inf_x \psi(x) > -\infty$.*

Assumption 4 (Lipschitz continuity in x (Assump. 5 in Boţ & Böhm, 2023))**.** *There exists $L > 0$ such that for all x, x' and all $y \in \operatorname{dom} h$, $\big|\Phi(x, y) - \Phi(x', y)\big| \le L\|x - x'\|$.*

Assumption 5 (Convex regularizers and bounded dual domain (Assump. 6 in Boţ & Böhm, 2023))**.** *The functions f and h are proper, l.s.c., convex. Moreover, either f is L_f–Lipschitz on its (open) domain or $f = \iota_{\mathcal{X}}$ for a nonempty closed convex set $\mathcal{X}$, and $\operatorname{dom} h$ is bounded with diameter $D_h < \infty$.*

Assumption 6 (Stochastic oracles: unbiasedness and bounded variance (Assump. 1–2 in Boţ & Böhm, 2023))**.** *The stochastic (sub)gradient estimators G_x, G_y used in alternating GDA satisfy:*

(i) Unbiasedness: $\mathbb{E}[G_x] = \nabla_x \Phi(x, y)$ and $\mathbb{E}[G_y] = \nabla_y \Phi(x, y)$.

(ii) Bounded variance: $\mathbb{E}\|G_x - \nabla_x \Phi(x, y)\|^2 \le \sigma^2_{\text{stoch}}$ and $\mathbb{E}\|G_y - \nabla_y \Phi(x, y)\|^2 \le \sigma^2_{\text{stoch}}$.

Indicator functions and projection as a proximal operator For a (nonempty) set $\mathcal{C} \subseteq \mathbb{R}^p$, we use the extended–real valued *indicator* $\iota_{\mathcal{C}} : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$,

$$\iota_{\mathcal{C}}(z) \;:=\; \begin{cases} 0, & z \in \mathcal{C}, \\ +\infty, & z \notin \mathcal{C}. \end{cases}$$

With this notation, $\operatorname{prox}_{\eta f}(u) := \arg\min_z \{\, f(z) + \frac{1}{2\eta}\|z - u\|^2 \,\}$ specializes to Euclidean projection when $f = \iota_{\mathcal{C}}$.

Definition (Moreau–envelope ε–stationarity). Let $\psi : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ be proper, l.s.c., and ρ–weakly convex, and fix any $\lambda \in (0, \rho^{-1})$. We say that x is *ε–stationary for ψ in the Moreau sense* if

$$\big\|\nabla \psi_\lambda(x)\big\| \;\le\; \varepsilon, \qquad \text{where} \quad \psi_\lambda(x) \;=\; \min_z \Big\{\, \psi(z) + \tfrac{1}{2\lambda}\|z - x\|^2 \,\Big\}.$$

Equivalently, with $\hat{x} := \operatorname{prox}_{\lambda\psi}(x)$, we have $\nabla \psi_\lambda(x) = \lambda^{-1}(x - \hat{x})$ and hence $\|x - \hat{x}\| \le \lambda\, \varepsilon$.

Lemma 1 ((Boţ & Böhm, 2023, Lemma 2.3))**.** *Let x be ε-stationary for the proper, ρ-weakly convex and l.s.c. function ψ, i.e., $\|\nabla \psi_\lambda(x)\| \le \varepsilon$ with $\lambda \in (0, \rho^{-1})$. Then there exist a point $\hat{x}$ such that $\|x - \hat{x}\| \le \varepsilon\lambda$ and $\operatorname{dist}\big(0, \partial\psi(\hat{x})\big) \le \varepsilon$.*

Theorem 4 (Stochastic alternating prox–GDA for nonconvex–concave problems (Boţ & Böhm 2023, Theorem 3.2))**.** *Under Assumptions 2–6, consider Algorithm 3. Let* $\phi(x) = \max_y\{\Phi(x,y) - h(y)\}$ *and* $\psi(x) = f(x) + \phi(x)$*. For any* $\lambda \in (0, \rho^{-1})$*, the Moreau envelope* ψ_λ *is* C^1*, and the iterates satisfy*

$$\min_{0\le t\le K} \mathbb{E}\big[\|\nabla\psi_\lambda(x_t)\|\big] \;\le\; \varepsilon \quad \textit{whenever} \quad K = \mathcal{O}\big(\varepsilon^{-8}\big)\,,$$

with constants depending only on problem Lipschitz/variance parameters and diameters of the constraint sets.

Mapping to the Boţ & Böhm (2023) template. Set the primal variable as $x \equiv \zeta = (\theta, A)$ with $\theta = (\mathbf{w}, \alpha, \sigma^2)$ and the dual variable as $y \equiv B \in \mathbb{R}^{d\times d}$. Define the convex constraint sets

$$\mathcal{X} \equiv \Omega_1 := \big\{\, \zeta :\; A \succeq \sigma^2 I,\; \sigma^2 \ge \sigma^2_{\min},\; \zeta \preceq \zeta_{\max} \,\big\}, \qquad \mathcal{Y} \equiv \Omega_2 := \big\{\, B :\; \|B\| \le 1 \,\big\},$$

and take $f = \iota_{\Omega_1}$, $h = \iota_{\Omega_2}$. Let

$$\Phi(\zeta, B) := g(\theta) + \log|A| \;+\; \mu\,\frac{\langle B,\; A - F(\theta)\rangle}{\|A\|},$$

so that

$$\phi(\zeta) = \max_{B\in\Omega_2} \Phi(\zeta, B) = g(\theta) + \log|A| + \mu\,\frac{\|A - F(\theta)\|}{\|A\|}.$$

Hence the regularized max-function is $\psi(\zeta) = \iota_{\Omega_1}(\zeta) + \phi(\zeta) = \iota_{\Omega_1}(\zeta) + \ell_\mu(\zeta)$, where $\ell_\mu(\zeta)$ is precisely our penalized objective in Section 3.1. With this identification, Algorithm 1 is exactly the projected alternating GDA of Boţ & Böhm (2023) applied to $f + \Phi - h$.

Constraint-set properties. Recall the constraint sets Ω_1 and Ω_2 defined above. By construction, Ω_1 is nonempty, compact, and convex, and every $\zeta = (\theta, A) \in \Omega_1$ satisfies $A \succeq \sigma^2 I_d$ with $\sigma^2 \in [\sigma^2_{\min}, \sigma^2_{\max}]$. Moreover, $\Omega_2 = \{B \in \mathbb{R}^{d\times d} : \|B\| \le 1\}$ is nonempty, compact, and convex. We will use these basic facts repeatedly in the proof.

Theorem 5 (Convergence of Algorithm 1)**.** *Let* Ω_1 *and* Ω_2 *be the constraint sets defined above, and set* $\psi(\zeta) = \iota_{\Omega_1}(\zeta) + \ell_\mu(\zeta)$*. Assume:*

- (H2) *The feature map* $x \mapsto \phi_\alpha(x)$ *is* C^1 *in* α*, and both* $\phi_\alpha(x)$ *and* $\partial\phi_\alpha(x)/\partial\alpha$ *are uniformly bounded on* Ω_1 *for all training points.*
- (H3) *Mini-batches are sampled i.i.d. (uniformly) and the resulting stochastic gradients used in the two projected steps are unbiased with uniformly bounded second moments.*

Then there exists $\rho > 0$ *such that, for any* $\lambda \in (0, \rho^{-1})$*, the Moreau envelope* ψ_λ *of* $\psi(\zeta) = \iota_{\Omega_1}(\zeta) + \ell_\mu(\zeta)$ *is* C^1*, and the iterates of Algorithm 1 with constant stepsizes chosen as in Boţ & Böhm (2023, Thm. 3.2) satisfy*

$$\min_{0\le t\le K} \mathbb{E}\big[\|\nabla\psi_\lambda(\zeta_t)\|\big] \;\le\; \varepsilon \qquad \textit{whenever} \qquad K = \mathcal{O}\big(\varepsilon^{-8}\big)\,.$$

Equivalently, Algorithm 1 reaches an ε*–stationary point of the penalized objective* ℓ_μ *(in the Moreau–envelope sense) within* $\mathcal{O}(\varepsilon^{-8})$ *stochastic gradient evaluations.*

Proof (verification of the reused assumptions). Under the mapping $x \equiv \zeta$, $y \equiv B$ described above, we check the items required by Assumptions 2–6 of Boţ & Böhm (2023) for our choices of Φ, $f = \iota_{\Omega_1}$ and $h = \iota_{\Omega_2}$.

A2(i): For fixed ζ, $B \mapsto \Phi(\zeta, B)$ is affine:

$$\nabla_B\Phi(\zeta, B) = \mu\,\frac{A - F(\theta)}{\|A\|},$$

which is independent of B. Hence it is concave and $L_{\nabla\Phi} = 0$–smooth on Ω_2.

A2(ii): On Ω_1 the functions $\theta \mapsto g(\theta)$ and $A \mapsto \log|A|$ are C^1 with Lipschitz gradients: $A \succeq \sigma_{\min}^2 I$ bounds $\|A^{-1}\|$, and (H2) bounds derivatives of $F(\theta)$. The penalty coupling

$$\zeta \mapsto \frac{\langle B,\, A - F(\theta)\rangle}{\|A\|}$$

is C^1 on Ω_1 because $\|A\| \geq \|A\|_2 \geq \sigma_{\min}^2 > 0$, and its gradient is Lipschitz on the compact set $\Omega_1 \times \Omega_2$ (all ingredients $A, F(\theta), B$ are uniformly bounded by (H1)–(H2)). Therefore $\nabla_\zeta \Phi$ is globally Lipschitz on $\Omega_1 \times \Omega_2$ with some constant L_ζ, so $x \mapsto \Phi(x, y)$ is ρ–weakly convex with $\rho := L_\zeta$.

A3: By (H1), Ω_1 is bounded and $A \succeq \sigma_{\min}^2 I$; thus $\log|A|$ is bounded below, $g(\theta)$ is bounded below, and $\|A - F(\theta)\|/\|A\|$ is bounded. Hence $\psi(\zeta) = \iota_{\Omega_1}(\zeta) + \ell_\mu(\zeta)$ is bounded below.

A4: On $\Omega_1 \times \Omega_2$ we already observed $\nabla_\zeta \Phi$ is bounded; hence $|\Phi(\zeta, B) - \Phi(\zeta', B)| \leq L\|\zeta - \zeta'\|$ with some $L < \infty$ independent of $B \in \Omega_2$.

A5: $f = \iota_{\Omega_1}$ and $h = \iota_{\Omega_2}$ are proper, l.s.c., convex. The domain of h equals Ω_2, whose diameter is finite.

A6: By mini-batch sampling (H3) and linearity of expectation, the stochastic gradients used in the two projected steps are unbiased. Uniform boundedness of gradients on $\Omega_1 \times \Omega_2$ implies a finite variance bound σ^2.

Thus Assumptions 2–6 hold for our problem data under the above mapping to the template of Boţ & Böhm (2023). Applying Theorem 4 (with $x \equiv \zeta$, $y \equiv B$) yields the claimed $\mathcal{O}(\varepsilon^{-8})$ complexity for $\min_{t \leq K} \mathbb{E}\|\nabla \psi_\lambda(\zeta_t)\|$. Since $\psi = \iota_{\Omega_1} + \ell_\mu$, this is precisely stationarity of the penalized objective ℓ_μ in the Moreau–envelope sense. □