Bias-Free Scalable Gaussian Processes via Randomized Truncations

Andres Potapczynski [*1] **Luhuan Wu** [*2] **Dan Biderman** [*1] **Geoff Pleiss** [1] **John P. Cunningham** [1 2]

Abstract

Scalable Gaussian Process methods are computationally attractive, yet introduce modeling biases that require rigorous study. This paper analyzes two common techniques: early truncated conjugate gradients (CG) and random Fourier features (RFF). We find that both methods introduce a systematic bias on the learned hyperparameters: CG tends to underfit while RFF tends to overfit. We address these issues using randomized truncation estimators that eliminate bias in exchange for increased variance. In the case of RFF, we show that the bias-to-variance conversion is indeed a trade-off: the additional variance proves detrimental to optimization. However, in the case of CG, our unbiased learning procedure meaningfully outperforms its biased counterpart with minimal additional computation. Our code is available at `https://github.com/cunningham-lab/RTGPS`.

1. Introduction

Gaussian Processes (GP) are popular and expressive non-parametric models, and considerable effort has gone into alleviating their cubic runtime complexity. Notable successes include inducing point methods (e.g. Snelson & Ghahramani, 2006; Titsias, 2009; Hensman et al., 2013), finite-basis expansions (e.g. Rahimi & Recht, 2008; Mutný & Krause, 2018; Wilson et al., 2020; Loper et al., 2020), nearest neighbor truncations (e.g. Datta et al., 2016; Katzfuss et al., 2021), and iterative numerical methods (e.g. Cunningham et al., 2008; Cutajar et al., 2016; Gardner et al., 2018). Common to these techniques is the classic *speed-bias tradeoff*: coarser GP approximations afford faster but more biased solutions that in turn affect both the model's predictions and learned hyperparameters. While a few papers analyze the bias of inducing point methods (Bauer et al., 2016; Burt et al., 2019), the biases of other approximation techniques, and their subsequent impact on learned GP models, have not been rigorously studied.

Here we scrutinize the biases of two popular techniques – random Fourier features (RFF) (Rahimi & Recht, 2008) and conjugate gradients (CG) (e.g. Cunningham et al., 2008; Cutajar et al., 2016; Gardner et al., 2018). These methods are notable due to their popularity and because they allow dynamic control of the speed-bias tradeoff: at any model evaluation, the user can adjust the number of CG iterations or RFF features to a desired level of approximation accuracy. In practice, it is common to truncate these methods to a fixed number of iterations/features that is deemed adequate. However, such truncation will stop short of an exact (machine precision) solution and potentially lead to biased optimization outcomes.

We provide a novel theoretical analysis of the biases resulting from RFF and CG on the GP log marginal likelihood objective. Specifically, we prove that CG is biased towards hyperparameters that underfit the data, while RFF is biased towards overfitting. In addition to yielding suboptimal hyperparameters, these biases hurt posterior predictions, regardless of the inference method used at test-time. Perhaps surprisingly, this effect is not subtle, as we will demonstrate. Our analysis suggests there is value in debiasing GP learning with CG and RFF. To do so, we turn to recent work that shows the merits of exchanging the speed-bias tradeoff for a *speed-variance tradeoff* (Beatson & Adams, 2019; Chen et al., 2019; Luo et al., 2020; Oktay et al., 2020). These works all introduce a randomization procedure that reweights elements of a fast truncated estimator, eliminating its bias at the cost of increasing its variance.

We thus develop bias-free versions of GP learning with CG and RFF using randomized truncation estimators. In short, we randomly truncate the number of CG iterations and RFF features, while reweighting intermediate solutions to maintain unbiasedness. Our variant of CG uses the **R**ussian **R**oulette estimator (Kahn, 1955), while our variant of RFF uses the **S**ingle **S**ample estimator of Lyne et al. (2015). We believe our **RR-CG** and **SS-RFF** methods to be the first to produce unbiased estimators of the GP log marginal likelihood with $< \mathcal{O}(N^3)$ computation.

[*]Equal contribution (randomly ordered). [1]Zuckerman Institute, Columbia University [2]Statistics Department, Columbia University. Correspondence to: Andres Potapczynski <ap3635@columbia.edu>, Luhuan Wu <lw2827@columbia.edu>, Dan Biderman <db3236@columbia.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

Finally, through extensive empirical evaluation, we find our methods and their biased counterparts indeed constitute a bias-variance tradeoff. Both RR-CG and SS-RFF are unbiased, recovering nearly the same optimum as the exact GP method, while GP trained with CG and RFF often converge to solutions with worse likelihood. We note that bias elimination is not always practical. For SS-RFF, the optimization is slow, due to the large auxiliary variance needed to counteract the slowly decaying bias of RFF. On the other hand, RR-CG incurs a minimal variance penalty, likely due to the favorable convergence properties of CG. In a wide range of benchmark datasets, RR-CG demonstrates similar or better predictive performance compared to CG using the same expected computational time.

To summarize, this work offers three main contributions:

- theoretical analysis of the bias of CG- and RFF-based GP approximation methods (§3)
- RR-CG and SS-RFF: bias-free versions of these popular GP approximation methods (§4)
- results demonstrating the value of our RR-CG and SS-RFF methods (§3 and 5).

2. Background

We consider observed data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ for $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, and the standard GP model:

$$f(\cdot) \sim \mathcal{GP}(\mu(\cdot), k(\cdot,\cdot)),$$
$$\mathbf{y}_i = f(\mathbf{x}_i) + \epsilon_i, \qquad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right)$$

where $k(\cdot,\cdot)$ is the covariance kernel, $\mu(\cdot)$ is set to zero without loss of generality, and hyperparameters are collected into the vector $\boldsymbol{\theta}$, which is optimized as:

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \\ \mathcal{L}(\boldsymbol{\theta}) &= -\log p(\mathbf{y}\,|\,\mathbf{X};\boldsymbol{\theta}) \\ &= \frac{1}{2}\big(\underbrace{\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|}_{\text{model complexity}} + \underbrace{\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}}_{\text{data fit}} + N\log 2\pi \big) \end{aligned} \tag{1}$$

where $\widehat{\mathbf{K}}_{\mathbf{XX}} \in \mathbb{R}^{N\times N}$ is the Gram matrix of all data points with diagonal observational noise:

$$\widehat{\mathbf{K}}_{\mathbf{XX}}[i,j] = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma^2 \mathbb{I}_{i=j}.$$

Following standard practice, we optimize $\boldsymbol{\theta}$ with gradients:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}} = \frac{1}{2}\left(\mathrm{tr}\left\{\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\frac{\partial\widehat{\mathbf{K}}_{\mathbf{XX}}}{\partial\boldsymbol{\theta}}\right\} - \mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\frac{\partial\widehat{\mathbf{K}}_{\mathbf{XX}}}{\partial\boldsymbol{\theta}}\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}\right). \tag{2}$$

Three terms thus dominate the computational complexity: $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$, $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$, and $\mathrm{tr}\{\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\frac{\partial\widehat{\mathbf{K}}_{\mathbf{XX}}}{\partial\boldsymbol{\theta}}\}$. The common approach to computing this triad is the Cholesky factorization, requiring $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ space.

Extensive literature has accelerated the inference and hyperparameter learning of GP. Two very popular strategies are using *conjugate gradients* (Cunningham et al., 2008; Cutajar et al., 2016; Gardner et al., 2018; Wang et al., 2019) to approximate the linear solves in Eq. (2), and *random Fourier features* (Rahimi & Recht, 2008), which constructs a randomized finite-basis approximation of the kernel.

2.1. Conjugate Gradients

To apply conjugate gradients to GP learning, we begin by replacing the gradient in Eq. (2) with a stochastic estimate (Cutajar et al., 2016; Gardner et al., 2018):

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\theta}} \approx \frac{1}{2}\left(\mathbf{z}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\frac{\partial\widehat{\mathbf{K}}_{\mathbf{XX}}}{\partial\boldsymbol{\theta}}\mathbf{z} - \mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\frac{\partial\widehat{\mathbf{K}}_{\mathbf{XX}}}{\partial\boldsymbol{\theta}}\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}\right), \tag{3}$$

where $\mathbf{z}$ is a random variable such that $\mathbb{E}[\mathbf{z}] = 0$ and $\mathbb{E}[\mathbf{z}\mathbf{z}^T] = \mathbf{I}$. Note that the first term constitutes a stochastic estimate of the trace term (Hutchinson, 1989). Thus, stochastic optimization of Eq. (1) can be reduced to computing the linear solves $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{z}$.

Conjugate gradients (CG) (Hestenes et al., 1952) is an iterative algorithm for solving positive definite linear systems $\mathbf{A}^{-1}\mathbf{b}$. It consists of a three-term recurrence, where each new term requires only a matrix-vector multiplication with $\mathbf{A}$. More formally, each CG iteration computes a new term of the following summation:

$$\mathbf{A}^{-1}\mathbf{b} = \textstyle\sum_{i=1}^N \gamma_i \mathbf{d}_i, \tag{4}$$

where the γ_i are coefficients and the $\mathbf{d}_i$ are conjugate search directions (Golub & Van Loan, 2012). N iterations of CG produce all N summation terms and recover the exact solution. In practice, exact convergence may require more than N iterations due to inaccuracies of floating point arithmetic. However, the summation converges exponentially, and so $J \ll N$ iterations may suffice to achieve high accuracy.

CG is an appealing method for computing $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{z}$ due to its computational complexity and its potential for GPU-accelerated matrix products. J iterations takes at most $\mathcal{O}(JN^2)$ time and $\mathcal{O}(N)$ space if the matrix-vector products are performed in a map-reduce fashion (Wang et al., 2019). However, ill-conditioned kernel matrices hinder the convergence rate (Cutajar et al., 2016), and so the J^{th} CG iteration may yield an inaccurate approximation of Eq. (3).

2.2. Random Fourier Features

Rahimi & Recht (2008) introduce a randomized finite-basis approximation to stationary kernels:

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}') \approx \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}') \tag{5}$$

where $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^J$ and $J \ll N$. The RFF approximation relies on Bochner's theorem (Bochner et al., 1959): letting $\tau = \mathbf{x} - \mathbf{x}'$, all stationary kernels $k(\tau)$ on $\mathbb{R}^d$ can be exactly expressed as the Fourier dual of a nonnegative measure $\mathbb{P}(\boldsymbol{\omega})$: $k(\tau) = \int \mathbb{P}(\boldsymbol{\omega}) \exp(i\boldsymbol{\omega}\tau)\, d\boldsymbol{\omega}$. A Monte Carlo approximation of this Fourier transform yields:

$$k(\tau) \approx \frac{2}{J} \sum_{j=1}^{J/2} \exp(i\boldsymbol{\omega}_j \tau), \quad \boldsymbol{\omega}_j \sim \mathbb{P}(\boldsymbol{\omega}),$$

which simplifies to a finite-basis approximation:

$$\mathbf{K}_{\mathbf{XX}} \approx [\boldsymbol{\phi}(\mathbf{x}_1) \ldots \boldsymbol{\phi}(\mathbf{x}_n)][\boldsymbol{\phi}(\mathbf{x}_1) \ldots \boldsymbol{\phi}(\mathbf{x}_n)]^\top,$$
$$\boldsymbol{\phi}(\mathbf{x}) = [\cos(\boldsymbol{\omega}_i^\top \mathbf{x}),\, \sin(\boldsymbol{\omega}_i^\top \mathbf{x})]_{i=1}^{J/2}, \quad \boldsymbol{\omega}_i \sim \mathbb{P}(\boldsymbol{\omega}).$$

For many common kernels, $\mathbb{P}(\boldsymbol{\omega})$ can be computed in closed-form (e.g. RBF kernels have zero-mean Gaussian spectral densities). The approximated log likelihood can be computed in $\mathcal{O}(J^3 + N)$ time and $\mathcal{O}(JN)$ space using the Woodbury inversion lemma and the matrix determinant lemma, respectively. The number of random features $J/2$ is a user choice, with typical values between 100-1000. More features lead to more accurate kernel approximations.

2.3. Unbiased Randomized Truncation

We will now briefly introduce Randomized Truncation Estimators, which are the primary tool we use to unbias the CG and RFF log marginal likelihood estimates. At a high level, assume that we wish to estimate some quantity ψ that can be expressed as a (potentially-infinite) series:

$$\psi = \textstyle\sum_{j=1}^{H} \Delta_j, \quad H \in \mathbb{N} \cup \{\infty\}.$$

Here and in the following sections, Δ_j can either be random or deterministic. To avoid the expensive evaluation of the full summation, a randomized truncation estimator chooses a random term $J \in \{1, \ldots, H\}$ with probability mass function $\mathbb{P}(J) = \mathbb{P}(\mathcal{J} = J)$ after which to truncate computation. In the following, we introduce two means of deriving unbiased estimators by upweighting the summation terms.

The Russian Roulette estimator (Kahn, 1955) obtains an unbiased estimator $\bar{\psi}_J$ by truncating the sum after $J \sim \mathbb{P}(J)$ terms and dividing the surviving terms by their survival probabilities:

$$\bar{\psi}_J = \sum_{j=1}^{J} \frac{\Delta_j}{\mathbb{P}(\mathcal{J} \geq j)} = \sum_{j=1}^{H} \left(\frac{\mathbb{I}_{J \geq j}}{\mathbb{P}(\mathcal{J} \geq j)} \right) \Delta_j, \qquad (6)$$

and, $\mathbb{E}[\bar{\boldsymbol{\psi}}_J] = \sum_{i=1}^{H} \Delta_j = \psi$. (See appendix for further derivation.) The choice of $\mathbb{P}(J)$ determines both the computational efficiency and the variance of $\bar{\psi}_J$. A thin-tailed $\mathbb{P}(J)$ will often truncate sums after only a few terms ($J \ll H$). However, tail events ($J \approx H$) are upweighted inversely to their low survival probability, and so thin-tailed truncation distributions may lead to high variance.

The Single Sample estimator (Lyne et al., 2015) implements an alternative reweighting scheme. After drawing $J \sim \mathbb{P}(J)$, it computes a single summation term Δ_J, which it upweights by $1/\mathbb{P}(J)$:

$$\bar{\psi}_J = \frac{\Delta_J}{\mathbb{P}(J)} = \sum_{j=1}^{H} \left(\frac{\mathbb{I}_{J=j}}{\mathbb{P}(\mathcal{J} = j)} \right) \Delta_j. \qquad (7)$$

This procedure is unbiased, and it amounts to estimating ψ using a single importance-weighted sample from $\mathbb{P}(J)$ (see appendix). Again, $\mathbb{P}(J)$ controls the speed/variance trade-off. We refer the reader to (Beatson & Adams, 2019) for a detailed comparison of these two estimators. We emphasize that both estimators remain unbiased even if Δ_j is a random variable, as long as it is independent from the random truncation integer J.

3. GP Learning with CG and RFF is Biased

Here we prove that early truncated CG and RFF provide biased approximations to the terms comprising the GP log marginal likelihood (Eq. 1). We also derive the bias decay rates for each method. We then empirically demonstrate these biases and show they affect the hyperparameters learned through optimization. Remarkably, we find that the above biases are *highly systematic*: CG-based GP learning favors underfitting hyperparameters while RFF-based learning favors overfitting hyperparameters.

3.1. CG Biases GP Towards Underfitting

In the GP literature, CG has often been considered an "exact" method for computing the log marginal likelihood (Cunningham et al., 2008; Cutajar et al., 2016), as the iterations are only truncated after reaching a pre-specified residual error threshold (e.g. 10^{-10}). However, as CG is applied to ever-larger kernel matrices it is common to truncate the CG iterations before reaching this convergence (Wang et al., 2019). While this accelerates the hyperparameter learning process, the resulting solves and gradients can no longer be considered "exact." In what follows, we show that the early-truncated CG optimization objective is not only approximate but also systematically biased towards underfitting.

To analyze the early-truncation bias, we adopt the analysis of Gardner et al. (2018) that recovers the GP log marginal likelihood (Eq. 1) from the stochastic gradient's CG estimates of $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{z}$ (Eq. 3). Recall the two terms in the log marginal likelihood are the "data fit" term $\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and the "model complexity" term $\log |\widehat{\mathbf{K}}_{\mathbf{XX}}|$. The first term falls directly out of the CG estimate of $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$, while a

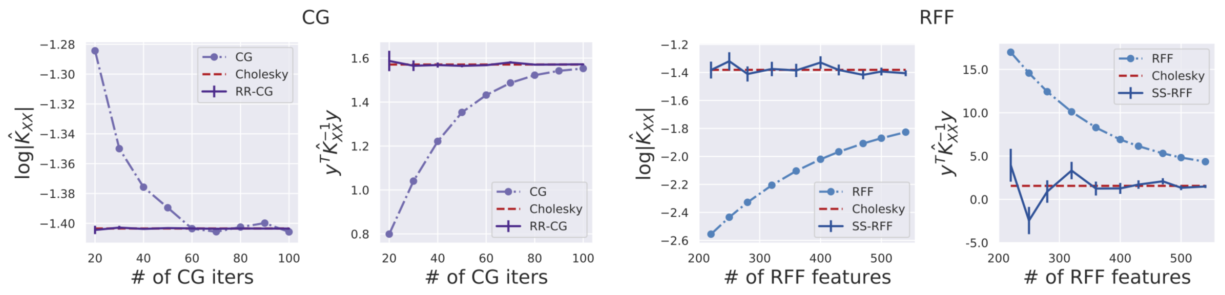

Figure 1. CG (left) systematically overestimates $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ and underestimates $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ whereas RFF (right) does the opposite. The dashed orange line shows the exact values computed by Cholesky. Our unbiased methods, RR-CG (left) and SS-RFF (right), recover the true $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ and $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ values. For these two methods, the x-axis indicates the expected number of iterations/features.

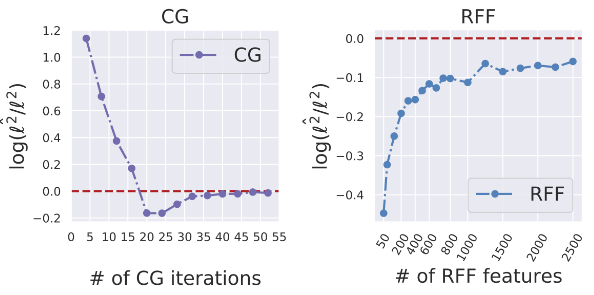

Figure 2. Kernel lengthscale values learned by optimizing (biased) CG and RFF log marginal likelihood approximations. CG overestimates the optimal kernel lengthscale whereas RFF underestimates it. We plot the divergence (in log-ratio scale) between the learned and true lengthscales as a function of the number of CG iterations (left) and of the number of RFF samples (right).

stochastic estimate of $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ can be obtained through the byproducts of CG's computation for $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{z}$. Gardner et al. (2018) show that the CG coefficients in Eq. (4) can be manipulated to produce a partial tridiagonalization: $\mathbf{T}_{\mathbf{z}}^{(J)} = \mathbf{Q}_{\mathbf{z}}^{(J)\top}\widehat{\mathbf{K}}_{\mathbf{XX}}\mathbf{Q}_{\mathbf{z}}^{(J)}$, where $\mathbf{T}_{\mathbf{z}}^{(J)} \in \mathbb{R}^{J\times J}$ is tridiagonal. $\mathbf{T}_{\mathbf{z}}^{(J)}$ can compute the Stochastic Lanczos Quadrature estimate of $\widehat{\mathbf{K}}_{\mathbf{XX}}$ (Ubaru et al., 2017; Dong et al., 2017):

$$\begin{aligned}\log|\widehat{\mathbf{K}}_{\mathbf{XX}}| &= \mathbb{E}\left[\mathbf{z}^T(\log\widehat{\mathbf{K}}_{\mathbf{XX}})\mathbf{z}\right]\\ &\approx \|\mathbf{z}\|^2\mathbf{e}_1^\top\left(\log\mathbf{T}_{\mathbf{z}}^{(J)}\right)\mathbf{e}_1,\end{aligned} \tag{8}$$

where $\log(\cdot)$ is the matrix logarithm and $\mathbf{e}_1$ is the first row of the identity matrix. The following theorem analyzes the bias of these $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ estimates:

Theorem 1. *Let u_J and v_J be the estimates of $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ respectively after J iterations of CG; i.e.:*

$$u_J = \mathbf{y}^\top\left(\textstyle\sum_{i=1}^J\gamma_i\mathbf{d}_i\right), \quad v_J = \|\mathbf{z}\|^2\mathbf{e}_1^\top\left(\log\mathbf{T}_{\mathbf{z}}^{(J)}\right)\mathbf{e}_1.$$

If $J < N$, CG underestimates the inverse quadratic term and overestimates the log determinant in expectation:

$$u_J \le \mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}, \quad \mathbb{E}_{\mathbf{z}}[v_J] \ge \log|\widehat{\mathbf{K}}_{\mathbf{XX}}|. \tag{9}$$

The biases of both terms decay at a rate of $\mathcal{O}(C^{-2J})$, where C is a constant that depends on the conditioning of $\widehat{\mathbf{K}}_{\mathbf{XX}}$.

Proof sketch. The direction of the biases can be proved using a connection between CG and numeric quadrature. u_J and v_J are exactly equal to the J-point Gauss quadrature approximation of $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\|\mathbf{z}\|^2\mathbf{e}_1^\top(\log\mathbf{T}_{\mathbf{z}}^{(N)})\mathbf{e}_1$ represented as Riemann-Stieltjes integrals. The sign of the CG approximation bias follows from the standard Gauss quadrature error bound, which is negative for $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and positive for $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$. The convergence rates are from standard bounds on CG (Golub & Van Loan, 2012) and the analysis of Ubaru et al. (2017). See appendix for a full proof.

Fig. 1 confirms our theoretical analysis and demonstrates the systematic biases of CG. We plot the log marginal likelihood terms for a subset of the PoleTele UCI dataset, varying the number of CG iterations (J) used to produce the estimates. Compared against the exact terms (computed with Cholesky), we see an overestimation of $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ and an underestimation of $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$. These biases are most prominent when using few CG iterations.

We turn to study the effect of CG learning on hyperparameters. Since the log marginal likelihood is a nonconvex function of $\boldsymbol{\theta}$, it is not possible to directly prove how the bias affects $\boldsymbol{\theta}$. Nevertheless, we know intuitively that underestimating $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ de-prioritizes model fit while overestimating $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ over-penalizes model complexity. Thus, the learned hyperparameters will likely underfit the data. Underfitting may manifest in an overestimation of the learned lengthscale ℓ, as low values of ℓ increase the flexibility and the complexity of the model. This hypothesis is empirically confirmed in Fig. 2 (left panel). We train a GP regression model on a toy dataset: $y = x\sin(5\pi x) + \varepsilon$

and $\varepsilon \sim \mathcal{N}(0, 0.01)$. We fix all hyperparameters other than the lengthscale, which is learned using both CG-based optimization and (exact) Cholesky-based optimization. The overestimation of ℓ decays with the number of CG iterations.

3.2. RFF Biases GP Towards Overfitting

Previous work has studied the accuracy of RFF's approximation to the entries of the Gram matrix $k(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$ (Rahimi & Recht, 2008; Sutherland & Schneider, 2015). However, to the best of our knowledge there has been little analysis of nonlinear functions of this approximate Gram matrix, such as $\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y}$ and $\log |\widehat{\mathbf{K}}_{\mathbf{XX}}|$ appearing in the GP objective. Interestingly, we find that RFF systematically biases these terms:

Theorem 2. *Let $\widetilde{\mathbf{K}}_J$ be the RFF approximation with $J/2$ random features. In expectation, $\widetilde{\mathbf{K}}_J$ overestimates the inverse quadratic and underestimates the log determinant:*

$$\mathbb{E}_{\mathbb{P}(\boldsymbol{\omega})} \left[\mathbf{y}^\top \widetilde{\mathbf{K}}_J^{-1} \mathbf{y} \right] \geq \mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y} \tag{10}$$

$$\mathbb{E}_{\mathbb{P}(\boldsymbol{\omega})} \left[\log |\widetilde{\mathbf{K}}_J| \right] \leq \log |\widehat{\mathbf{K}}_{\mathbf{XX}}|. \tag{11}$$

The biases of both terms decay at a rate of $\mathcal{O}(1/J)$.

Proof sketch. The direction of the biases is a straightforward application of Jensen's inequality, noting that $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}$ is a convex function and $\log |\widehat{\mathbf{K}}_{\mathbf{XX}}|$ is a concave function. The magnitude of the bias is derived from a second-order Taylor expansion that closely resembles the analysis of Nowozin (2018). See appendix for full proof.

Again, Fig. 1 confirms the systematic biases of RFF, which decay at a rate proportional to the number of features, as predicted by Thm. 2. Hence, RFF should affect the learned hyperparameters in a manner opposite to CG. Overestimating $\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y}$ emphasizes data fitting while underestimating $\log |\widehat{\mathbf{K}}_{\mathbf{XX}}|$ reduces the model complexity penalty, overall resulting in overfitting behavior. Following the intuition presented in Sec. 3.1, we expect the lengthscale to be underestimated, as empirically confirmed by Fig. 2 (right panel). The figure also illustrates the slow decay of the RFF bias.

4. Bias-free Scalable Gaussian Processes

We debias the estimates of both the GP training objective in Eq. (1) and its gradient in Eq. (2) (as approximated by CG and RFF) using unbiased randomized truncation estimators. To see how such estimators apply to GP hyperparameter learning, we note that both CG and RFF recover the true log marginal likelihood (or an unbiased estimate thereof) in their limits:

Observation 1. *CG recovers the exact log marginal likelihood in expectation in at most N iterations:*

$$\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y} = \mathbf{y}^\top \left(\textstyle\sum_{j=1}^N \gamma_j \mathbf{d}_j \right), \tag{12}$$

$$\log |\widehat{\mathbf{K}}_{\mathbf{XX}}| = \mathbb{E}_{\mathbf{z}} \left[\|\mathbf{z}\|^2 \mathbf{e}_1^\top (\log \mathbf{T}_{\mathbf{z}}^{(N)}) \mathbf{e}_1 \right]. \tag{13}$$

By the law of large numbers, RFF converges almost surely to the exact log marginal likelihood as the number of random features goes to infinity:

$$\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y} = \lim_{J \to \infty} \mathbf{y}^\top \widetilde{\mathbf{K}}_J^{-1} \mathbf{y}, \tag{14}$$

$$\log |\widehat{\mathbf{K}}_{\mathbf{XX}}| = \lim_{J \to \infty} \log |\widetilde{\mathbf{K}}_J|. \tag{15}$$

To maintain the scalability of CG and RFFs while eliminating bias, we express the log marginal likelihood terms in Eqs. (12) to (15) as summations amenable to randomized truncation. We then apply the Russian Roulette and Single Sample estimators of Sec. 2.3 to avoid computing all summation terms while obtaining the same result in expectation.

4.1. Russian Roulette-Truncated CG (RR-CG)

The stochastic gradient in Eq. (3) requires performing two solves: $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y}$ and $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{z}$. Using the summation formulation of CG (Eq. 4), we can write these two solves as series:

$$\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y} = \textstyle\sum_{i=1}^N \gamma_i \mathbf{d}_i, \quad \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{z} = \textstyle\sum_{i=1}^N \gamma_i' \mathbf{d}_i',$$

where each CG iteration computes a new term of the summation. By applying the Russian Roulette estimator from Eq. (6), we obtain the following unbiased estimates:

$$\begin{aligned} \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y} &\approx \textstyle\sum_{j=1}^J (\gamma_j \mathbf{d}_j) / \mathbb{P}(\mathcal{J} \geq j), \quad J \sim \mathbb{P}(J) \\ \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{z} &\approx \textstyle\sum_{j=1}^{J'} (\gamma_j' \mathbf{d}_j') / \mathbb{P}(\mathcal{J} \geq j). \quad J' \sim \mathbb{P}(J'), \end{aligned} \tag{16}$$

These unbiased solves produce an unbiased optimization gradient in Eq. (3); we refer to this approach as Russian Roulette CG (**RR-CG**). With the appropriate truncation distribution $\mathbb{P}(J)$, this estimate affords the same computational complexity of standard CG *without* its bias.

We must compute two independent estimates of $\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y}$ with different $J \sim \mathbb{P}(J)$ in order for the $\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \frac{\partial \widehat{\mathbf{K}}_{\mathbf{XX}}}{\partial \boldsymbol{\theta}} \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1} \mathbf{y}$ term in Eq. (3) to be unbiased. Thus, the unbiased gradient requires 3 calls to RR-CG, as opposed to the 2 CG calls needed for the biased gradient. Nevertheless, RR-CG has the same $\mathcal{O}(JN^2)$ complexity as standard CG – and the additional solve can be computed in parallel with the others.

We can also use the Russian Roulette estimator to compute the log marginal likelihood itself, though this is not strictly necessary for gradient-based optimization. (See appendix.)

Choosing the truncation distribution. Since the Russian Roulette estimator is unbiased for any choice of $\mathbb{P}(J)$, we wish to choose a truncation distribution that balances computational cost and variance[1]. Beatson & Adams (2019) propose the *relative optimization efficiency* (ROE) metric, which is the ratio of the expected improvement of taking an optimization step with our gradient estimate to its computational cost. A critical requirement of the ROE analysis is the expected rate of decay of our approximations in terms of the number of CG iterations J. We summarize our estimates and choices of distribution as follows:

Theorem 3. *The approximation to* $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ *and to* $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ *using RR-CG decays at a rate of* $\mathcal{O}(C^{-2J})$*. Therefore the truncation distribution that maximizes the ROE is* $\mathbb{P}^*(J) \propto C^{-2J}$*, where* C *is a constant that depends on the conditioning of* $\widehat{\mathbf{K}}_{\mathbf{XX}}$*. The expected computation and variance of* $\mathbb{P}^*(J)$ *is finite.*

Proof sketch. Beatson & Adams (2019) show that the truncation distribution that maximizes the ROE is proportional to the rate of decay of our approximation divided by its computational cost. The error of CG decays as $\mathcal{O}(C^{-2J})$, and the cost of each summation term is constant with respect to J. In practice, we vary the exponential decaying rate to control the expectation and variance of $\mathbb{P}(J)$. To further reduce the variance of our estimator, we set a minimum number of CG iterations to be computed, as in (Luo et al., 2020). See appendix for full proof.

In practice, however, we do not have access to C since computing the conditioning of $\widehat{\mathbf{K}}_{\mathbf{XX}}$ is impractical. Yet, we can change the base of the exponential to e and add a temperature parameter λ to rescale the function and control the rate of decay of the truncation distribution as a sensible alternative. Thus, we follow a more general exponential decay distribution:

$$\mathbb{P}(J) \propto e^{-\lambda J}, \qquad J = J_{\min}, \cdots, N \tag{17}$$

where $J_{\min}$ is the minimum truncation number. By varying the values of λ and $J_{\min}$, we can control the expectation and standard deviation of $\mathbb{P}(J)$. In practice we found that having the standard deviation value between 10 and 20 achieves stable GP learning process, which can be obtained by tuning λ between 0.05 and 0.1 for sufficiently large datasets (e.g. $N \geq 500$). We also noticed that the method is not sensitive to these choices of hyperparameters and that they work well across all the experiments. The expected truncation number can be further tuned by varying $J_{\min}$. We emphasize that these choices impact the speed-variance tradeoff. By setting a larger $J_{\min}$ we decrease the speed by requiring more baseline computations but also decrease the variance (since the minimum approximations have the largest deviations from the ground truth).

[1]Solely minimizing variance is not appealing, as this is achieved by $\mathbb{P}(J) = \mathbb{I}_{J=N}$ which has no computational savings.

Toy problem. In Fig. 1 we plot the empirical mean of the RR-CG estimator using 10^4 samples from an exponential truncation distribution. We find that RR-CG produces unbiased estimates of the $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ terms that are indistinguishable from the exact values computed with Cholesky. Reducing the expected truncation iteration $\mathbb{E}(J)$ (x-axis) increases the standard error of empirical means, demonstrating the speed-variance trade-off.

4.2. Single Sample-Truncated RFF (SS-RFF)

Denoting $\widetilde{\mathbf{K}}_j$ as the kernel matrix estimated by j random Fourier features, we can write $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ as the following telescoping series:

$$\log|\widehat{\mathbf{K}}_{\mathbf{XX}}| = \log|\widetilde{\mathbf{K}}_1| + \sum_{j=2}^{N/2-1}\Big(\log|\widetilde{\mathbf{K}}_j| - \log|\widetilde{\mathbf{K}}_{j-1}|\Big) \tag{18}$$

$$+ \log|\widehat{\mathbf{K}}_{\mathbf{XX}}| - \log|\widetilde{\mathbf{K}}_{N/2-1}|$$
$$= \log|\widetilde{\mathbf{K}}_1| + \textstyle\sum_{j=2}^{N/2}\Delta_j, \tag{19}$$

where Δ_j is defined as $\log|\widetilde{\mathbf{K}}_j| - \log|\widetilde{\mathbf{K}}_{j-1}|$ for all $j < N/2$, and $\Delta_{N/2}$ is defined as $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}| - \log|\widetilde{\mathbf{K}}_{N/2-1}|$. Note that each Δ_j is now a random variable, since it depends on the *random* Fourier frequencies ω. Crucially, we only include $N/2$ terms in the series so that no term requires more than $\mathcal{O}(N^3)$ computation in expectation. (For any $j > N/2$, $\widetilde{\mathbf{K}}_j$ is a full-rank matrix and thus is as computationally expensive as the true $\widehat{\mathbf{K}}_{\mathbf{XX}}$ matrix.) We construct a similar telescoping series for $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$.

As with Eq. (16), we approximate the series in Eq. (19) with a randomized truncation estimator, this time using the Single Sample estimator (7):

$$\log|\widehat{\mathbf{K}}_{\mathbf{XX}}| \approx \log|\widetilde{\mathbf{K}}_1| + \Delta_J/\mathbb{P}(J). \tag{20}$$

where J is drawn from the truncation distribution $\mathbb{P}(J)$ with support over $\{2, 3, \ldots N/2\}$. Note that the Single Sample estimate requires computing 3 log determinants ($\log|\widetilde{\mathbf{K}}_1|$, $\log|\widetilde{\mathbf{K}}_{J-1}|$, and $\log|\widetilde{\mathbf{K}}_J|$) for a total of $\mathcal{O}(J^3 + NJ^2)$ computations and $\mathcal{O}(NJ)$ memory. This is asymptotically the same requirement as standard RFF. The Russian Roulette estimator, on the other hand, incurs a computational cost of $\mathcal{O}(NJ^3 + J^4)$ as it requires computing ($\log|\widetilde{\mathbf{K}}_1|$ through $\log|\widetilde{\mathbf{K}}_J|$) which quickly becomes impractical for large J.

A similar Single Sample estimator constructs an unbiased estimate of $\mathbf{y}^\top\widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$. Backpropagating through these Single Sample estimates produces unbiased estimates of the log marginal likelihood gradient in Eq. (2).

Choosing the truncation distribution. For the Single Sample estimator we do not have to optimize the ROE since minimizing the variance of this estimator does not result in a degenerate distribution.

Theorem 4. *The truncation distribution that minimizes the variance of the SS-RFF estimators for* $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ *and* $\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ *is* $\mathbb{P}^*(J) \propto 1/J$. *The expected variance and computation of* $\mathbb{P}^*(J)$ *is finite.*

Proof sketch. The minimum variance distribution can be found by solving a constrained optimization problem. In practice, we can further decrease the variance of our estimator by fixing a minimum value of RFF features to be used in Eq. (19) and by increasing the step size ($c \in \mathbb{N}$) between the elements at each $\Delta_j = \log|\widetilde{\mathbf{K}}_{cJ}| - \log|\widetilde{\mathbf{K}}_{c(J-1)}|$. See appendix for full proof.

For the experiments we started with 500 features and also tried various step sizes $c \in \{1, 10, 100\}$. The variance of the estimator decreases as we increase c since the probability weights will decrease in magnitude. Yet, despite of using the optimal truncation distribution, setting a high number of features and taking long steps $c = 100$, the variance of the estimator still requires taking several steps before converging, making SS-RFF computationally impractical.

Toy problem. Similar to RR-CG, in Fig. 1 we plot the empirical mean of the SS-RFF estimator using 10^4 samples. We find that SS-RFF produces unbiased estimates of the $\mathbf{y}^\top \widehat{\mathbf{K}}_{\mathbf{XX}}^{-1}\mathbf{y}$ and $\log|\widehat{\mathbf{K}}_{\mathbf{XX}}|$ terms. However, these estimates have a higher variance when compared to the estimates of RR-CG. Reducing the expected truncation iteration $\mathbb{E}(J)$ (x-axis) increases the standard error of the empirical means, demonstrating the speed-variance trade-off.

4.3. Analysis of the Bias-free Methods

Randomized truncations and conjugate gradients have existed for many decades (Hestenes et al., 1952; Kahn, 1955), but have rarely been used in conjunction. Filippone & Engler (2015) proposed a method closely related to our RR-CG which performs randomized early-truncation of CG iterations to obtain unbiased posterior samples of the GP covariance parameters. We differ by tackling the GP hyperparameter learning problem: we provide the first theoretical analysis of the biases incurred by both CG and RFF, and proceed to tailor unbiased estimators for each method.

To some extent, randomized truncation methods are antithetical to the original intention of CG: producing deterministic and nearly exact solves. For large-scale applications, where early truncation is necessary for computational tractability, the ability to trade bias for variance is beneficial. This fact is especially true in the context of GP learning, where the bias of early truncation is systematic and cannot be simply explained away as numerical imprecision.

Randomized truncation estimates are often used to estimate infinite series, where it is challenging to design truncation distributions with finite expected computation and/or variance. We avoid such issues since CG and the telescoping RFF summations are both finite.

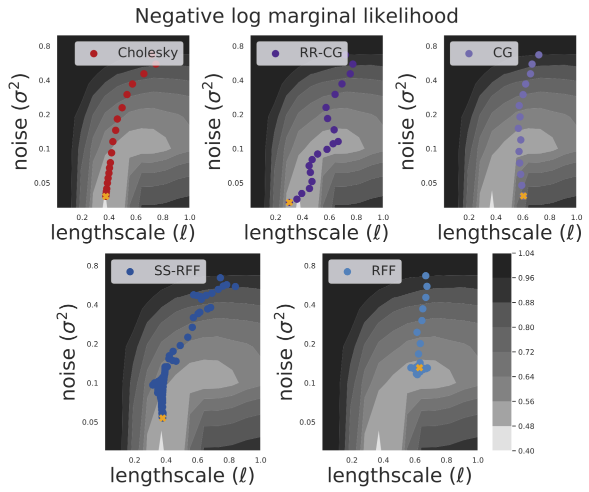

Figure 3. Optimization landscape of a GP with two hyperparameters. The SS-RFF and RR-CG models converge to similar hyperparameter values that are nearly optimal, while the RFF and CG models converge to suboptimal solutions. In addition, the stochastic effect of the randomized truncation is visible in the trajectories of RR-CG and SS-RFF. Moreover, CG (and RR-CG) models truncate after 20 iterations (in expectation); RFF (and SS-RFF) models use 700 features (in expectation).

5. Results

First, we show that our bias-free methods recover nearly the same hyperparameters as exact methods (i.e. Cholesky-based optimization), whereas models that use CG and RFF converge to suboptimal hyperparameters. Since RR-CG and SS-RFF eliminate bias at the cost of increased variance, we then demonstrate the optimization convergence rate and draw conclusions on our methods' applicability. Finally, we compare models optimized with RR-CG against a host of approximate GP methods across a wide range of UCI datasets (Asuncion & Newman, 2007). All experiments are implemented in GPyTorch (Gardner et al., 2018)

Optimization trajectories of bias-free GP. Fig. 3 displays the optimization landscape – the log marginal likelihood of the PoleTele dataset – as a function of (RBF) kernel lengthscale ℓ and noise σ^2. As expected, an exact GP (optimized using Cholesky, Fig. 3 upper left) recovers the optimum. Notably, the GP trained with standard CG and RFF converges to suboptimal hyperparameters (upper right/lower right). RR-CG and SS-RFF models (trained with 20 iterations and 700 features in expectation, respectively) successfully eliminate this bias, and recover nearly

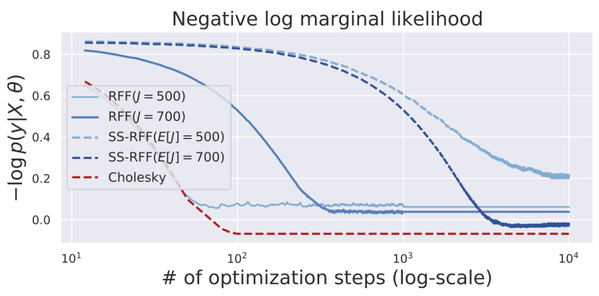

Figure 4. The GP optimization objective for models trained with RFF and SS-RFF. (PoleTele dataset, RBF kernel, Adam optimizer.) RFF models converge to sub-optimal log marginal likelihoods. SS-RFF models converge to (near) optimum values, yet require more than $100\times$ as many optimization steps.

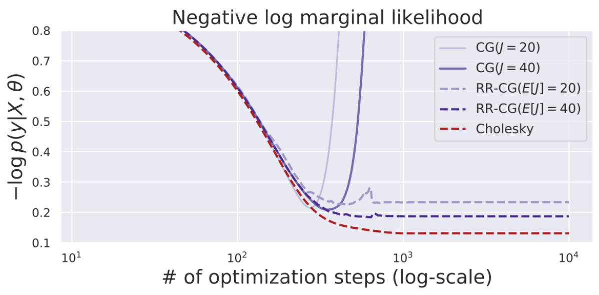

Figure 5. The GP optimization objective for models trained with CG and RR-CG. (Bike dataset, RBF kernel, Adam optimizer.) RR-CG models converge to optimal solutions, while the (biased) CG models diverge. Increasing the expected truncation of RR-CG only slightly improves optimization convergence; models converge in < 100 steps of Adam.

the same parameters as the exact model (upper center/lower left). These plots also show the speed-variance tradeoff of randomized truncation. SS-RFF and RR-CG have noisy optimization trajectories due to auxiliary truncation variance.

Convergence of GP hyperparameter optimization. Figs. 4 and 5 plot the exact GP log marginal likelihood of the parameters learned by each method during optimization. Each trajectory corresponds to a RBF-kernel GP trained on the PoleTele dataset (Fig. 4) and the Bike dataset (Fig. 5).

Fig. 4 shows that RFF models converge to solutions with worse log likelihoods, and more RFF features slow the rate of optimization. Additionally, we see the cost of the auxiliary variance needed to debias SS-RFF: while SS-RFF models achieve better optima than their biased counterparts, they take 2-3 orders of magnitude longer to converge, despite using a truncation distribution that minimizes variance. We thus conclude that SS-RFF has too much variance to be practical for GP hyperparameter learning.

Fig. 5 on the other hand shows that RR-CG is minimally affected by its auxiliary variance. The GP trained with RR-CG converges in roughly 100 iterations, nearly matching Cholesky-based optimization. Decreasing the expected truncation value from $\mathbb{E}[J] = 40$ to 20 slightly slows this convergence. We note that the bias induced by standard CG can be especially detrimental to GP learning. On this dataset, the biased models deviate from their Cholesky counterparts and eventually diverge away from the optimum.

Predictive performance of bias-free GP. Lastly, we compare the predictive performance of GPs that use RR-CG, CG, and Cholesky for hyperparameter optimization. We emphasize that the RR-CG and CG methods only make use of early truncation approximations during training. At test time, we compute the predictive posterior by running CG to a tolerance of $\leq 10^{-4}$, which we believe can be considered "exact" for practical intents and purposes. Additionally, we include four other (biased) scalable GP approximations methods as baselines: **RFF**, Stochastic Variational Gaussian Processes (**SVGP**) (Hensman et al., 2013), generalized Product of Expert Gaussian Processes (**POE**) (Cao & Fleet, 2014; Deisenroth & Ng, 2015), and stochastic gradient-based Gaussian Processes (**sgGP**) (Chen et al., 2020). We note that the RFF, SVGP, and sgGP methods introduce *both* bias and variance, as these methods rely on randomization and approximation.

We use CG with $J = 100$ iterations, and RR-CG with $\mathbb{E}[J] = 100$ expected iterations; both methods use the preconditioner of Gardner et al. (2018). All RFF models use 700 random features. For SVGP, we use 1,024 inducing points and minibatches of size 1,024 as in (Wang et al., 2019). The POE models are comprised of GP experts that are each trained on 1,024 data point subsets. For sgGP, the subsampled datasets are constructed by selecting a random point $\mathbf{x}, y$ and its 15 nearest neighbors as in (Chen et al., 2020). Each dataset is randomly split to 64% training, 16% validation and 20% testing sets. All kernels are RBF with a separate lengthscale per dimension. See appendix for more details.

We report prediction accuracy (RMSE) and negative log likelihood (NLL) in Fig. 6 (see appendix for full tables on predictive performance and training time). We make two key observations: *(i)* RR-CG meaningfully debiases CG. When the bias of CG is not detrimental to optimization (e.g. CG with 100 iterations is close to convergence for the Elevators dataset), RR-CG has similar performance. However, when the CG bias is more significant (e.g. the KEGG dataset), the bias-free RR-CG improves the GP predictive RMSE and NLL. We also include a figure displaying the predictive performance of RR-CG and CG with increasing number of (expected) CG iterations in appendix. *(ii)* RR-CG recovers the same optimum as the "ground-truth" method

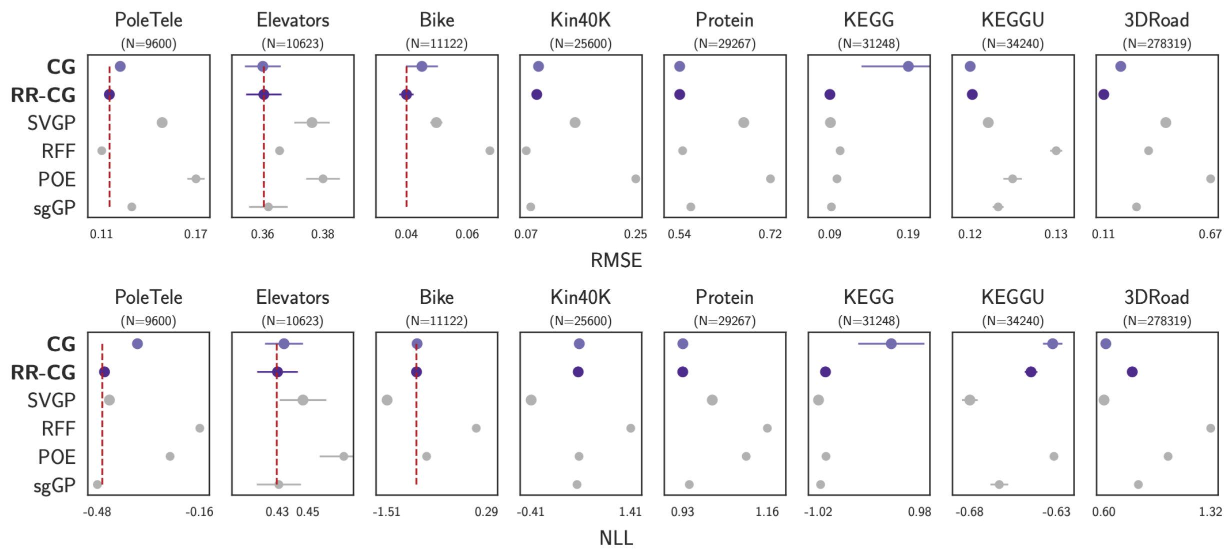

Figure 6. Root-mean-square-error (RMSE) and negative log likelihood (NLL) of GP trained with CG (light purple), RR-CG (dark purple) and various approximate methods (grey). Dashed red lines indicates Cholesky-based GP performance (when applicable). Results are averaged over 3 dataset splits. Missing RFF and sgGP results correspond to (very high) outlier NLL / RMSE values. In almost all experiments, GP learning with RR-CG achieves similar or better performance compared to that with CG at the same computational cost.

(i.e. Cholesky) does, as indicated by the red-dashed line in Fig. 6. This result provides additional evidence that RR-CG achieves unbiased optimization.

While RR-CG obtains the lowest RMSE on all but 2 datasets, we note that the (biased) GP approximations sometimes achieve lower NLL. For example, SVGP has a lower NLL than that of RR-CG on the Bike dataset, despite having a higher RMSE. We emphasize that this is not a failing of RR-CG inference. The SVGP NLL is even better than that of the exact (Cholesky) GP, suggesting a potential model misspecification for this particular dataset. Since SVGP overestimates the observational noise σ^2 (Bauer et al., 2016), it may obtain a better NLL when outliers are abundant. Though we cannot compare against the Cholesky posterior on larger datasets, we hypothesize that the NLL/RMSE discrepancy on these datasets is due to a similar modeling issue.

6. Conclusion

We prove that CG and RFF introduce systematic biases to the GP log marginal likelihood objective: CG-based training will favor underfitting models, while RFF-based training will promote overfitting. Modifying these methods with randomized truncation converts these biases into variance, enabling unbiased stochastic optimization. Our results show that this bias-to-variance exchange indeed constitutes a trade-off. The convergence of SS-RFF is impractically slow, likely due to the truncation variance needed to eliminate RFF's slowly-decaying bias. However, for CG-based training, we find that variance is almost always preferable to bias. Models trained with RR-CG achieve better performance than those trained with standard CG, and tend to recover the hyperparameters learned with exact methods. Though models trained with CG do not always exhibit noticeable bias, RR-CG's negligible computational overhead is justifiable to counteract cases where the bias is significant.

We reported experiments with at most 300K observations for our methods and baselines, which is substantial for GPs. We emphasize that RR-CG can be extended to datasets with over one million data points as in Wang et al. (2019). However, the computational cost is much higher, requiring multiple GPUs for training and testing.

We note that the RR-CG algorithm is not limited to GP applications. Future work should explore applying RR-CG to other optimization problems with large-scale solves.

Acknowledgements

This work was supported by the Simons Foundation, McKnight Foundation, the Grossman Center, and the Gatsby Charitable Trust.

References

Asuncion, A. and Newman, D. Uci machine learning repository, 2007.

Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse gaussian process approximations. In *Advances in Neural Information Processing Systems*, 2016.

Beatson, A. and Adams, R. P. Efficient optimization of loops and limits with randomized telescoping sums. In *International Conference on Machine Learning*, 2019.

Bochner, S. et al. *Lectures on Fourier integrals*, volume 42. Princeton University Press, 1959.

Burt, D., Rasmussen, C. E., and Van Der Wilk, M. Rates of convergence for sparse variational gaussian process regression. In *International Conference on Machine Learning*, pp. 862–871, 2019.

Cao, Y. and Fleet, D. J. Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*, 2014.

Chen, H., Zheng, L., Al Kontar, R., and Raskutti, G. Stochastic gradient descent in correlated settings: A study on gaussian processes. *Advances in Neural Information Processing Systems*, 33, 2020.

Chen, R. T. Q., Behrmann, J., Duvenaud, D., and Jacobsen, J.-H. Residual flows for invertible generative modeling. *Advances in Neural Information Processing Systems*, 2019.

Cunningham, J. P., Shenoy, K. V., and Sahani, M. Fast gaussian process methods for point process intensity estimation. In *International Conference on Machine learning*, pp. 192–199, 2008.

Cutajar, K., Osborne, M. A., Cunningham, J. P., and Filippone, M. Preconditioning kernel matrices. In *International Conference on Machine Learning*, 2016.

Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. Hierarchical nearest-neighbor gaussian process models for large geostatistical datasets. *Journal of the American Statistical Association*, 111(514):800–812, 2016.

Deisenroth, M. and Ng, J. W. Distributed gaussian processes. In *International Conference on Machine Learning*, pp. 1481–1490. PMLR, 2015.

Dong, K., Eriksson, D., Nickisch, H., Bindel, D., and Wilson, A. G. Scalable log determinants for gaussian process kernel learning. In *Advances in Neural Information Processing Systems*, pp. 6327–6337, 2017.

Filippone, M. and Engler, R. Enabling scalable stochastic gradient-based inference for gaussian processes by employing the unbiased linear system solver (ulisse). In *International Conference on Machine Learning*, pp. 1015–1024. PMLR, 2015.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, pp. 7576–7586, 2018.

Golub, G. H. and Van Loan, C. F. *Matrix Computations*. The Johns Hopkins University Press, 4th Edition, 2012.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, 2013.

Hestenes, M. R., Stiefel, E., et al. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(1), 1952.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Kahn, H. Use of different monte carlo sampling techniques. *Rand Corporation*, 1955.

Katzfuss, M., Guinness, J., et al. A general framework for vecchia approximations of gaussian processes. *Statistical Science*, 36(1):124–141, 2021.

Loper, J., Blei, D., Cunningham, J. P., and Paninski, L. General linear-time inference for gaussian processes on one dimension. *arXiv preprint arXiv:2003.05554*, 2020.

Luo, Y., Beatson, A., Norouzi, M., Zhu, J., Duvenaud, D., Adams, R. P., and Chen, R. T. Q. Sumo: Unbiased estimation of log marginal probability for latent variable models. In *International Conference on Learned Representations*, 2020.

Lyne, A.-M., Girolami, M., Atchadé, Y., Strathmann, H., Simpson, D., et al. On russian roulette estimates for bayesian inference with doubly-intractable likelihoods. *Statistical science*, 30(4):443–467, 2015.

Mutnỳ, M. and Krause, A. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, pp. 9005–9016, 2018.

Nowozin, S. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *International Conference on Learned Representations*, 2018.

Oktay, D., McGreivy, N., Aduol, J., Beatson, A., and Adams, R. P. Randomized automatic differentiation. *arXiv preprint arXiv:2007.10412*, 2020.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, pp. 1177–1184, 2008.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, 2006.

Sutherland, D. J. and Schneider, J. G. On the error of random fourier feaures. *Conference on Uncertainty in Artificial Intelligence*, 2015.

Titsias, M. K. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Ubaru, S., Chen, J., and Saad, Y. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Wang, K., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K. Q., and Wilson, A. G. Exact gaussian processes on a million data points. In *Advances in Neural Information Processing Systems*, pp. 14648–14659, 2019.

Wilson, J., Borovitskiy, V., Terenin, A., Mostowsky, P., and Deisenroth, M. Efficiently sampling functions from gaussian process posteriors. In *International Conference on Machine Learning*, pp. 10292–10302, 2020.