GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration

J. R. Gardner*, G. Pleiss*, D. Bindel, K. Q. Weinberger, A. G. Wilson

Presented at NeurIPS-2018

https://gpytorch.ai

Jungtaek Kim (jtkim@postech.ac.kr)

POSTECH
Republic of Korea
https://jungtaek.github.io

April 30, 2021

Table of Contents

Introduction

- The gains in optimization originate in large part from insights in stochastic gradient optimization, effectively trading off unnecessary exactness for speed and in some cases regularization.
- The advantages of modern software frameworks for deep learning include rapid prototyping, easy access to specialty compute hardware (such as GPUs), and blackbox optimization through automatic differentiation.
- However, the tools most commonly used for GP inference do not effectively utilize modern hardware, and new models require significant implementation efforts.
- GPyTorch [Gardner et al., 2018] bridges this gap by introducing a highly efficient framework for Gaussian process inference.

Gaussian Process

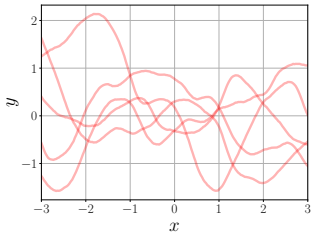
- A collection of random variables, any finite number of which have a joint Gaussian distribution [Rasmussen and Williams, 2006].
- Generally, Gaussian process (GP) is defined as

$$f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \tag{1}$$

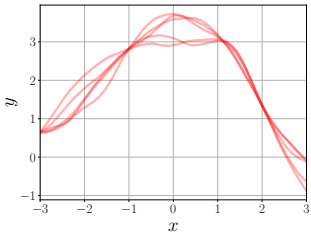
where

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{2}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{3}$$

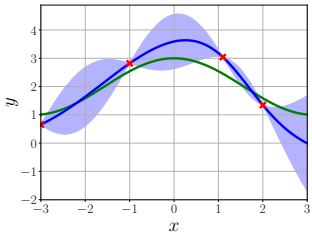
Gaussian Process Regression



(a) From prior function dist.



(b) From posterior function dist.



(c) Predictive dist.

Figure 1: Gaussian process regression for a function $\cos(x) + 2$ with observation noise.

Gaussian Process Regression

- One of popular covariance functions, the squared-exponential covariance function in one dimension is defined as

$$k\left(x, x'\right) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}\left(x - x'\right)^2\right) + \sigma_n^2 \delta_{xx'}, \tag{4}$$

where σ_f is a signal level, l is a length scale and σ_n is a noise level [Rasmussen and Williams, 2006].

- Posterior mean function $\mu(\cdot)$ and covariance function $\Sigma(\cdot)$:

$$\mu(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \boldsymbol{I})^{-1}\mathbf{y}, \tag{5}$$

$$\Sigma(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X}^*) - K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \boldsymbol{I})^{-1}K(\mathbf{X}, \mathbf{X}^*). \tag{6}$$

Gaussian Process Regression

- If non-zero mean prior is given, posterior mean and covariance functions:

$$\mu(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \boldsymbol{I})^{-1}(\mathbf{y} - \mu_p(\mathbf{X})) + \mu_p(\mathbf{X}), \tag{7}$$

$$\Sigma(\mathbf{X}^*) = K(\mathbf{X}^*, \mathbf{X}^*) + K(\mathbf{X}^*, \mathbf{X})(K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \boldsymbol{I})^{-1} K(\mathbf{X}, \mathbf{X}^*), \tag{8}$$

where $\mu_p(\cdot)$ is a prior mean function.

Notation

- Denote that

  $X \in \mathbb{R}^{n \times d}$ where the ith row (i.e., $\mathbf{x}_i$) is the ith training example,

  $[K_{XX}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$,

  $[\mathbf{k}_{X\mathbf{x}^*}]_i = k(\mathbf{x}_i, \mathbf{x}^*)$,

  $\widehat{K}_{XX} = K_{XX} + \sigma_n^2 \boldsymbol{I}$ where σ_n^2 is a noise variance.

Training a Gaussian Process

- Gaussian processes depend on hyperparameters θ (e.g., l, σ_f, and σ_n).
- These parameters are commonly learned by minimization or sampling via the negative log marginal likelihood:

$$L(\theta|X,\mathbf{y}) \propto \log\left|\widehat{K}_{XX}\right| + \mathbf{y}^\top \widehat{K}_{XX}^{-1}\mathbf{y}, \tag{9}$$

$$\frac{\mathrm{d}L}{\mathrm{d}\theta} = -\mathbf{y}^\top \widehat{K}_{XX}^{-1}\frac{\mathrm{d}\widehat{K}_{XX}}{\mathrm{d}\theta}\widehat{K}_{XX}^{-1}\mathbf{y} + \mathrm{Tr}\left(\widehat{K}_{XX}^{-1}\frac{\mathrm{d}\widehat{K}_{XX}}{\mathrm{d}\theta}\right). \tag{10}$$

$$L(\theta\,|\,X,\mathbf{y}) \propto \log\left|\widehat{K}_{XX}\right| - \mathbf{y}^\top \widehat{K}_{XX}^{-1}\mathbf{y}, \quad \frac{dL}{d\theta} = \mathbf{y}^\top \widehat{K}_{XX}^{-1}\frac{d\widehat{K}_{XX}}{d\theta}\widehat{K}_{XX}^{-1}\mathbf{y} + \mathrm{Tr}\left(\widehat{K}_{XX}^{-1}\frac{d\widehat{K}_{XX}}{d\theta}\right). \tag{2}$$

Cholesky Decomposition

- The Cholesky decomposition of a symmetric, positive-definite matrix A decomposes A into a product of a lower triangular matrix L and its transpose:

$$LL^\top = A, \tag{11}$$

  where L is called the Cholesky factor.

- To solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$, we can get the solution as $\mathbf{x} = L^\top \backslash (L \backslash \mathbf{b})$.

- The computation of the Cholesky factor L is considered numerically extremely stable and takes time $n^3/6$.

Gaussian Process Inference through Blackbox Matrix Multiplication

- The goal of this paper is to replace existing inference strategies with a unified framework that utilizes modern hardware efficiently.
- Complex GP models can be used in a blackbox manner without additional inference rules.
- To this end, this method reduces the bulk of GP inference to one of the most efficiently-parallelized computations: matrix-matrix multiplication, which is dubbed Blackbox Matrix-Matrix inference (BBMM).

Required Operations

- $\widehat{K}_{XX}^{-1}\mathbf{y}$
- $\log|\widehat{K}_{XX}|$
- $\mathrm{Tr}(\widehat{K}_{XX}^{-1}\frac{\mathrm{d}\widehat{K}_{XX}}{\mathrm{d}\theta})$

Gaussian Process Inference through Blackbox Matrix Multiplication

- Recently, there is a growing line of research that computes these operations with iterative routines based on matrix-vector multiplications (MVMs).
- The term $\widehat{K}_{XX}^{-1}\mathbf{y}$ can be computed using conjugate gradients (CG).
- The other two terms can be computed using calls to the iterative Lanczos tridiagonalization algorithm.
- BUT, one primary issue is parallelism.

Conjugate Gradients

- The contents are taken from [Shewchuk, 1994].
- The Conjugate Gradients (CG) method solves large systems of linear equations:

$$A\mathbf{x} = \mathbf{b}, \tag{12}$$

where $\mathbf{x}$ is an unknown vector, $\mathbf{b}$ is a known vector, and A is a known, symmetric, positive-definite matrix.

Steepest Descent

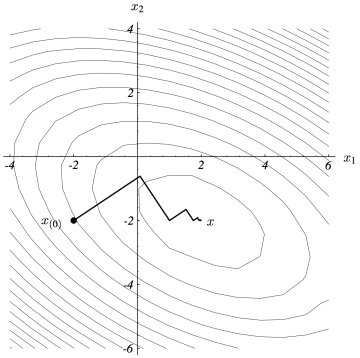


Figure 8: Here, the method of Steepest Descent starts at $[-2,-2]^T$ and converges at $[2,-2]^T$.

- The steepest descent algorithm is

$$\mathbf{r}_{(i)} = \mathbf{b} - A\mathbf{x}_{(i)}, \tag{13}$$

$$\alpha_{(i)} = \frac{\mathbf{r}_{(i)}^\top \mathbf{r}_{(i)}}{\mathbf{r}_{(i)}^\top A\mathbf{r}_{(i)}}, \tag{14}$$

$$\mathbf{x}_{(i+1)} \leftarrow \mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{r}_{(i)}. \tag{15}$$

Conjugate Gradients

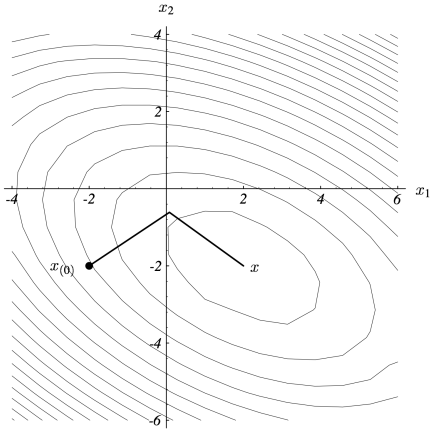


Figure 30: The method of Conjugate Gradients.

- The CG algorithm is

$$\mathbf{d}_{(0)} = \mathbf{r}_{(0)} = \mathbf{b} - A\mathbf{x}_{(0)}, \tag{16}$$

$$\alpha_{(i)} = \frac{\mathbf{r}_{(i)}^{\top}\mathbf{r}_{(i)}}{\mathbf{d}_{(i)}^{\top}A\mathbf{d}_{(i)}}, \tag{17}$$

$$\mathbf{x}_{(i+1)} \leftarrow \mathbf{x}_{(i)} + \alpha_{(i)}\mathbf{d}_{(i)}, \tag{18}$$

$$\mathbf{r}_{(i+1)} \leftarrow \mathbf{r}_{(i)} - \alpha_{(i)}A\mathbf{d}_{(i)}, \tag{19}$$

$$\beta_{(i+1)} = \frac{\mathbf{r}_{(i+1)}^{\top}\mathbf{r}_{(i+1)}}{\mathbf{r}_{(i)}^{\top}\mathbf{r}_{(i)}}, \tag{20}$$

$$\mathbf{d}_{(i+1)} \leftarrow \mathbf{r}_{(i+1)} + \beta_{(i+1)}\mathbf{d}_{(i)}. \tag{21}$$

Modified Batched Conjugate Gradients

- A modified Batched Conjugate Gradients (mBCG) algorithm has been proposed.
- Given random variables $\mathbf{z}_1, \ldots, \mathbf{z}_t$, the outputs of mBCG are

$$[\mathbf{u}_0 \ \mathbf{u}_1 \ \cdots \ \mathbf{u}_t] = \widehat{K}_{XX}^{-1}[\mathbf{y} \ \mathbf{z}_1 \ \ldots \ \mathbf{z}_t], \tag{22}$$

  and the partial Lanczos tridiagonalizations of $\widehat{K}_{XX}$,

$$\tilde{T}_1, \ldots, \tilde{T}_t. \tag{23}$$

- Thus, $\mathbf{u}_0 = \widehat{K}_{XX}^{-1}\mathbf{y}$.

Estimating Three GP Inference Terms

- Finally, three GP inference terms can be esimated:

$$\mathbf{u}_0 = \widehat{K}_{XX}^{-1}\mathbf{y}, \tag{24}$$

$$\mathrm{Tr}\left(\widehat{K}_{XX}^{-1}\frac{\mathrm{d}\widehat{K}_{XX}}{\mathrm{d}\theta}\right) = \mathbb{E}\left[\mathbf{z}_i^\top \widehat{K}_{XX}^{-1}\frac{\mathrm{d}\widehat{K}_{XX}}{\mathrm{d}\theta}\mathbf{z}_i\right] \approx \frac{1}{t}\sum_{i=1}^{t}\left(\mathbf{z}_i^\top \widehat{K}_{XX}^{-1}\right)\left(\frac{\mathrm{d}\widehat{K}_{XX}}{\mathrm{d}\theta}\mathbf{z}_i\right), \tag{25}$$

$$\log|\widehat{K}_{XX}| = \mathrm{Tr}(\log T) = \mathbb{E}\left[\mathbf{z}_i^\top(\log T)\mathbf{z}_i\right] \approx \frac{1}{t}\sum_{i=1}^{t}\mathbf{z}_i^\top \tilde{Q}_i(\log\tilde{T}_i)\tilde{Q}_i^\top\mathbf{z}_i. \tag{26}$$

Preconditioning

- While each iteration of mBCG performs large parallel matrix-matrix operations that utilize hardware efficiently, the iterations themselves are sequential.
- A natural goal for better utilizing hardware is to trade off fewer sequential steps for slightly more effort per step.
- It is accomplished by introducing preconditioning:

$$P^{-1}\widehat{K}_{XX}\mathbf{u} = P^{-1}\mathbf{y}, \tag{27}$$

  instead of $\widehat{K}_{XX}^{-1}\mathbf{y}$.
- Both systems are guaranteed to have the same solution, but the preconditioned system's convergence depends on the conditioning of $P^{-1}\widehat{K}_{XX}$ rather than that of $\widehat{K}_{XX}$.

Preconditioning

- There are two requirements of a preconditioner to be used in general for GP inference.
- First, in order to ensure that preconditioning operations do not dominate running time when using scalable GP methods, the preconditioner should afford roughly linear time solves and space.
- Second, the log determinant of the preconditioner matrix $\log|P|$ should be efficiently computed, because $\log|\widehat{K}_{XX}| = \log|P^{-1}\widehat{K}_{XX}| + \log|P|$ must be comptued.
- For one possible preconditioner, the pivoted Cholesky decomposition is applied.

Programmability with BBMM

- Indeed BBMM is blackbox by nature, only requiring a routine to perform matrix-multiplications with the kernel matrix and its derivative.
- Examples of how existing GP models and scalable approximations can be easily implemented can be provided in this framework:
  1. Bayesian linear regression,
  2. Sparse Gaussian process regression,
  3. Structured kernel interpolation,
  4. Compositions of kernels.
- See `https://gpytorch.ai` for more examples.

Results

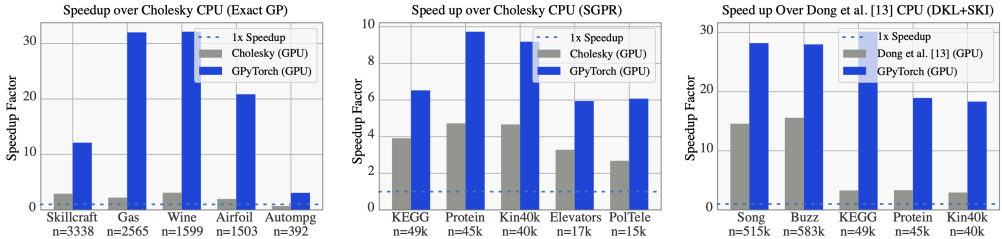


Figure 1: Speedup of GPU-accelerated inference engines. BBMM is in blue, and competing GPU methods are in gray. **Left:** Exact GPs. **Middle:** SGPR [21, 45] – speedup over CPU Cholesky-based inference engines. **Right:** SKI+DKL [50, 52] – speedup over CPU inference of Dong et al. [13].

Results

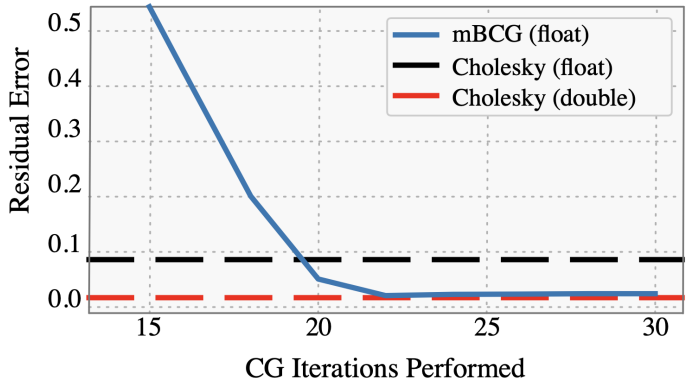


Figure 2: Solve error for mBCG and Cholesky.

Results

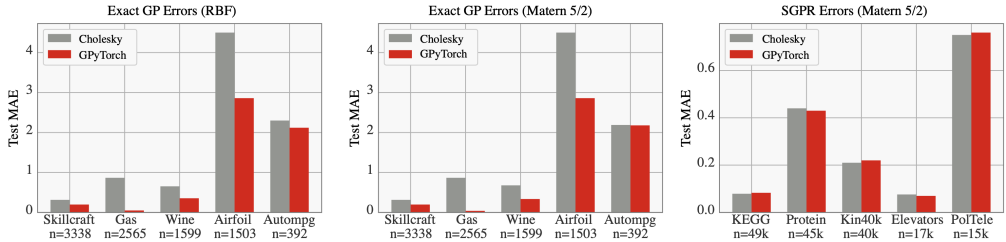


Figure 3: Comparing final Test MAE when using BBMM versus Cholesky based inference. The left two plots compare errors using Exact GPs with RBF and Matern-5/2 kernels, and the final plot compares error using SGPR with a Matern-5/2 kernel on significantly larger datasets.

Results

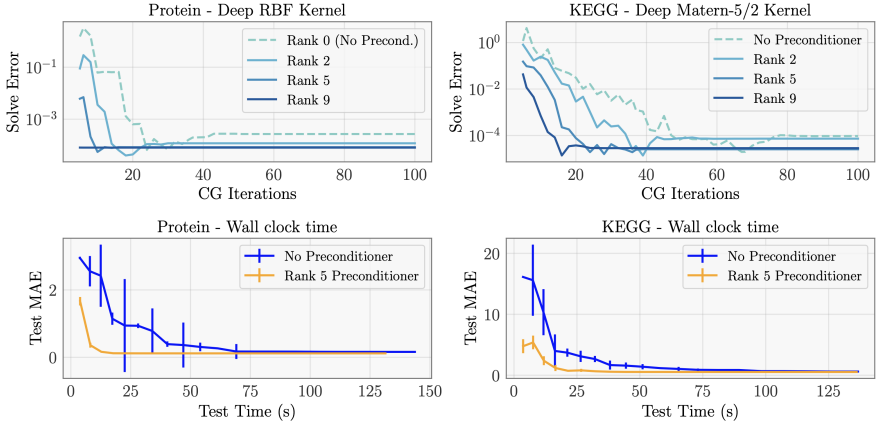


Figure 4: The effect of preconditioning on solve errors $\|K\mathbf{x}^* - \mathbf{y}\|/\|\mathbf{y}\|$ achieved by linear conjugate gradients using no preconditioner versus rank 2, 5, and 9 pivoted Cholesky preconditioners on 2 UCI benchmark datasets using deep RBF and deep Matern kernels. The hyperparameters of K were learned by maximizing the marginal log likelihood on each dataset.

Discussion

- Non-Gaussian likelihoods
- Avoiding the Cholesky decomposition

Takeaway

- If we need to compute matrix inversion and speed up this process, this technique can be used.
- GPyTorch is a well-maintained project in Gaussian process community.
- GPyTorch is straightforwardly utilized in BoTorch.

Thank you.

References I

J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 7576–7586, Montreal, Quebec, Canada, 2018.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.