Embarrassingly Parallel Inference for Gaussian Processes

Michael Minyi Zhang
Department of Statistics and Data Science
The University of Texas at Austin
Austin TX, 78712
michael_zhang@utexas.edu

Sinead A. Williamson
McCombs School of Business
The University of Texas at Austin
Austin TX, 78712
sinead.williamson@mccombs.utexas.edu

Abstract

Training Gaussian process (GP)-based models typically involves an $O(N^3)$ computational bottleneck. Popular methods for overcoming the matrix inversion problem include sparse approximations of the covariance matrix through inducing variables or dimensionality reduction via "local experts". However, these type of models cannot account for both long and short range correlations in the GP functions, and are often hard to implement in a distributed setting. We present an embarrassingly parallel method that takes advantage of the computational ease of inverting block diagonal matrices, while maintaining much of the expressivity of a full covariance matrix. By using importance sampling to average over different realizations of low-rank approximations of the GP model, we ensure our algorithm is both asymptotically unbiased and embarrassingly parallel. We show comparable or improved performance over competing methods, on a range of synthetic and real datasets.

1 Introduction

Many problems in statistics and machine learning can be framed in terms of learning a latent function $f(x)$. For example, in regression problems, we represent our dependent variables as a (noisy) function of our independent variables. In classification, we learn a function that maps an input to a class (or a probability distribution over classes). In parameter optimization, we have some function that maps parameters to their likelihoods, and want to find the optima of this function. However, if we assume the relationship between x and $f(x)$ is nonlinear then learning the latent function is a non-trivial task.

Gaussian processes (GPs) provide a flexible family of distributions over functions, that have been widely adopted for problems including regression, classification and optimization [17, 16, 13] due to their ease of use in modeling latent functions. Unfortunately, inference in GP models with N observations involves repeated inversion of an $N \times N$ matrix, which typically scales $O\left(N^3\right)$. This computational bottleneck has thus far prevented Gaussian processes from being used in so-called "big data" situations.

Two main approaches have been proposed to ameliorate the computational complexity for inference: The first approach aims to reduce the computational burden of inverting an $N \times N$ matrix. Sparse methods approximate the full function f using a function evaluated at $M << N$ inducing points, requiring only a $M \times M$ matrix inversion [11, 2, 7, 12, 14, 8]. However, sparse methods are not well-suited for modeling functions with short length-scales without increasing M, thereby increasing the computational burden.

Local methods, the other approach, learn K separate Gaussian processes, each on N/K-th of the data, and then combine the estimates [15, 1, 10, 3]. This is equivalent to approximating the $N \times N$ covariance matrix with a block-diagonal matrix, which we can invert by inverting the individual N/K-sized blocks. Unlike the sparse methods, local GP inference can model functions with short

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

length-scales, but will poorly approximate functions with long-length scales unless the size of the block in the block-diagonal approximation is increased, which will cause the very problem we want to avoid (see Low et al. [9] for a discussion on weaknesses in sparse and local GP methods).

In this paper, we propose a novel inference algorithm for Gaussian processes that is both scalable and easily distributed. The "Importance Gaussian Process Sampler" (IGPS) approximates the covariance matrix with a distribution over block diagonal matrices. We learn, in parallel, multiple Gaussian processes sampled from this distribution, allowing us to take advantage of the lower inversion cost of a block-diagonal matrix. We then use importance sampling to combine these estimates in a principled manner–the only step in our algorithm requiring global communication. The resulting posterior predictive distribution has a dense covariance matrix, avoiding edge effects common with local methods and allowing for an expressive covariance structure that can model both long- and short-range covariance.

We will start by describing our approach in Section 2, before presenting detailed experimental evaluation in Section 3 to showcase its efficacy versus existing methods. Further extensions and applications are discussed in Section 4.

2 Embarrassingly Parallel Inference with the Importance Gaussian Process Sampler

Our approach is a mixture-of-experts scheme that uses a distribution over partitions which allows for long-range correlations and a dense implied covariance matrix but uses a trivially parallelizable importance-sampling-based inference. We call the resulting combination of covariance matrix and inference algorithm the "Importance Gaussian Process Sampler" (IGPS). We specify our distribution over partitions by assuming that each data point has a latent block indicator $z_i \sim \pi$, where $\pi \sim \text{Dir}_K(\alpha)$ and we fit a Normal-Inverse Wishart mixture model on the location and covariance of the input space. Data points associated with block k have input location $x_i \sim \text{N}(\mu_k, \Gamma_k)$, and compose a single block $\mathbf{\Sigma}_j$ of the covariance matrix $\mathbf{\Sigma}$, associated with parameters Θ_k.

For inference, we independently sample J partitions from $Z^{(j)} \sim p(Z|X, \boldsymbol{\mu}, \mathbf{\Gamma})$ and assign them weights $w_j \propto p(Z|X,Y)/p(Z|X)$. We can then use the weighted samples to obtain asymptotically unbiased[1] estimates of functions of the posterior, such as predictions at new inputs X^*. Calculating the w_j involves integrating over the covariance parameters Θ_k:

$$\frac{p(Z|X,Y)}{p(Z|X)} \propto \frac{p(X,Y|Z)p(Z)}{p(X|Z)p(Z)} = p(Y|X,Z) = \int p(Y|X,Z,\Theta)p(\Theta)d\Theta.$$

We must approximate this intractable integral. In our experiments, we choose a MAP approximation; while this is not as accurate as sampling hyperparameters it is significantly faster, and mirrors the choices made by our comparison methods.

To perform prediction using IGPS, we compute predictions on each importance sample with $P(f_j^*|Z_j-) = \sum_{k=1}^{K} P(f_j^*|Z_j, X^*, Z_j^*, -)P(Z_j^*|X_{k,j}, Z_j)$[2]. We obtain the unnormalized importance weights from the optimization routine for learning hyperparameters on each importance sample. After normalizing the weights, we obtain our final predictions with $P(f^*|-) \approx \sum_{j=1}^{J} w_j P(f_j^*|Z_j, -)$. The overall computational cost of the IGPS, using J importance-weighted samples and K blocks, is $O(JN^3/K^2)$. In Table 1, we compare this with the overall computational cost of the full GP, sparse approximations of FITC and DTC, [12, 14], sparse approximations learned using SVI [5], the Bayesian treed GP (BTGP) [4], and the robust Bayesian committee machine (RBCM) [3].

While the $O(JN^3/K^2)$ cost is $O(J)$ higher than sparse methods[3] and local methods based on a fixed partition such as RBCM, the J samples can be performed and weighted in parallel. We can

[1] Since we are working with self-normalized weights, the estimate has a bias of $O(1/J)$ [6], but a lower variance than the unbiased estimate obtained with $w_j = p(Z|X,Y)/p(Z|X)$.

[2] As a fast approximation, we estimate f_j^* by predicting X^* with the expert corresponding to the MAP estimate of Z_j^* instead of averaging over all K experts

[3] In general, a sparse model with M inducing points obtains comparable accuracy to a local method with N/K local GPs.

Table 1: Comparison of inference complexity, for various methods. N is the number of data points, K is the number of experts, and $M = N/K$ is the number of inducing points. For the Monte Carlo based methods, J is the number of MCMC iterations or importance samples.

	Full GP	Sparse	SVI	BTGP	RBCM	**IGPS**
Complexity	$O(N^3)$	$O(NM^2)$	$O(M^3)$	$O(JN^3/K^2)$	$O(N^3/K^2)$	$O(JN^3/K^2)$
Complexity/Thread	x	x	x	x	$O(N^3/K^3)$	$O(N^3/K^3)$

also make use of the independence of the K partitions to parallelize further, using JK threads each taking $O(N^3/K^3)$. As shown in Table 1, this leads to an equivalent wall-time cost comparable with the distributed RBCM. As we will see in Section 3, the extra computational cost required to ensure a full posterior predictive distribution yields improved performance over methods that are based on a fixed partition.

3 Experimental Evaluation

We begin by evaluating our method on synthetically generated data. In our studies, we will compare our Importance Gaussian Process Sampler against the models listed in Table 1. We consider four settings: (1) Data generated with a long lengthscale, (2) data generated with a short lengthscale, (3) non-stationary data generated from a piecewise periodic function, and (4) a high dimensional input space generated from a Gaussian mixture model with an RBF kernel and long length scale. For (1), (2) and (4), we used a stationary RBF kernel, trained on 1000 data points and evaluated on 500 held-out data points based on predictive likelihood. For (3), we allowed each mixture to have its own hyperparameters. In all methods, except BTGP where we used MCMC, we infer hyperparameters through optimization.

For the sparse methods, we used $M = 50$ inducing points and for the local methods we used $K = 20$ partitions to have a comparable computational complexity. We ran BTGP for 1,000 iterations and for the IGPS we used 100 independent importance-weighted samples. For space-saving purposes, we only show the posterior distributions obtained on experiments (1) in Figure 1. Table 2 shows the test set log likelihood for all four datasets.

We first consider the one-dimensional stationary examples. Looking at the results on the dataset with long-range correlations, we see that the IGPS performs better than the other local methods, and performs nearly as well as the full GP and the sparse FITC approximation. If we look at the dataset with short-range correlation, the sparse methods generally perform worse than the local methods, as expected. And out of the local methods, the IGPS performs closest to the full GP. We also investigate the effect of increasing the number of importance samples and partitions on the predictive log likelihood of the stationary data in Figure 2.

For the non-stationary example, the methods fitted with stationary kernels clearly cannot account for the non-stationary components in the data. While DTC and SVI are more efficient with inducing parameters and can better capture the predictive mean than FITC, but by assuming a stationary covariance they give a poorer test-set performance. The BTGP performs well at capturing the function; though its performance is hampered by slow mixing and lack of convergence.

Also, we consider how the models fare in high dimension. Table 2 shows that the IGPS considerably outperform the other methods in a 50-dimensional setting. As the dimensionality increases, our inputs are more sparsely distributed. The input sparsity also means long-range correlations become more important, which is under-captured using the fixed partition of the RBCM. By contrast, the IGPS is able to obtain good performance by averaging over partitions.

Lastly, our IGPS method's inherently parallelizable nature makes it an appealing choice for the "big data" regime. We modeled a dataset where the goal is to predict flight delay times from a number of covariates.[4] Using the previous settings, We compare against RBCM and SVI—the only other methods in our comparisons that are distributable. We trained our method using 10,000 observations in the training set, 5,000 observations in the test set, and $K = 50$ clusters and experts. Our importance sampling method provides for a richer predictive model due to the averaging over

[4]The dataset is available at `http://stat-computing.org/dataexpo/2009/`

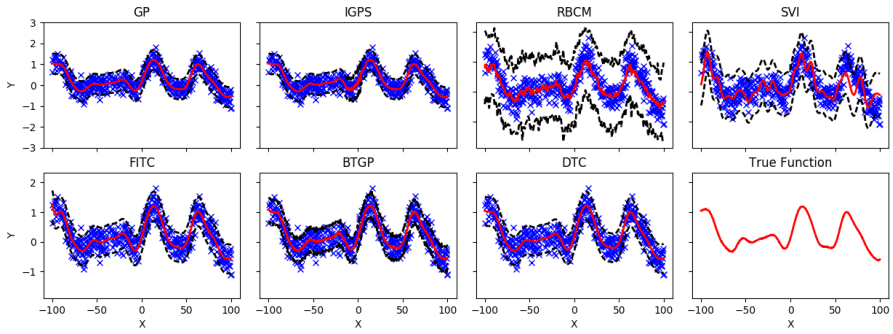

Figure 1: Synthetic (stationary, long length scale)

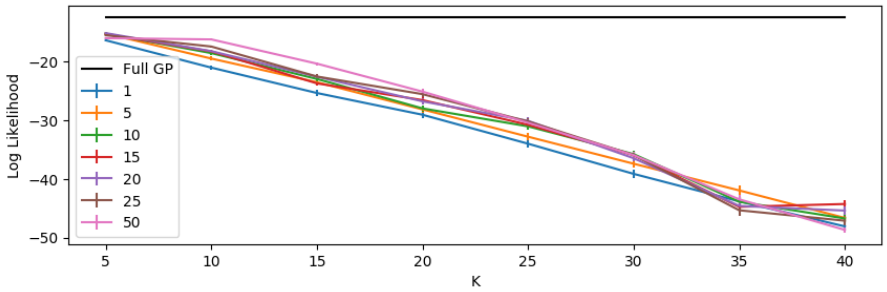

Figure 2: Comparison of test set log likelihood performance with increasing J and K for stationary data.

Table 2: Log likelihoods obtained on the stationary, non-stationary and large-N regression tasks

Data	GP	FITC	**IGPS**	BTGP	RBCM	SVI	DTC
Stationary, long lengthscale	-33.79	-35.28	-49.29	-80.15	-255.23	-444.43	-33.80
Stationary, short lengthscale	-31.32	-539.55	-43.15	-64.16	-280.08	-355.51	-125.92
Non-stationary	-711.57	-1.39e4	291.11	119.65	-1.17e5	-91.95	167.26
High-dimensional	-60.45	-108.71	-93.08	-147.53	-117.31	-141.00	-126.67
Flight Delay	x	x	-4.00e4	x	-7.13e7	-1.53e8	x

importance proposals and we can see the benefit of this in the results of our "big data" analysis. Table 2 shows that the IGPS obtains better predictive performances than other scalable methods.

4 Summary and future work

We have presented an approximation and algorithmic framework for scalable, distributed inference in Gaussian process models that avoids the pitfalls of existing approaches. Our algorithm uses importance sampling to average over the posterior distribution, introducing an $O(J)$ term into the computational complexity where J is the number of samples. In a single-machine setting, this leads to slower computation than sparse or local methods that do not average over predictions. However, independence between the J samples and independence between the K blocks in each sample means our algorithm can be trivially parallelized onto as many as JK threads or machines, with a per-thread computational cost of $O(N^3/K^3)$. This allows us to make efficient use of additional computational resources in a distributed setting, without increasing the total wall-clock time.

Additionally, we note that our algorithm can be used in conjunction with other inference techniques. For example, we could approximate the covariance matrix within a single partition using a sparse approximations, to obtain further speed-ups. An interesting avenue for future research would be to explore both which domains would benefit from additional approximations, and see which non-regression application areas would benefit most from our approach.

References

[1] Cao, Y. and Fleet, D. J. (2014). Generalized product of experts for automatic and principled fusion of Gaussian process predictions. In *Modern Nonparametrics 3: Automating the Learning Pipeline workshop at NIPS*.

[2] Csató, L. and Opper, M. (2002). Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668.

[3] Deisenroth, M. and Ng, J. W. (2015). Distributed Gaussian processes. In *International Conference on Machine Learning*, pages 1481–1490.

[4] Gramacy, R. B. and Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483):1119–1130.

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

[6] Kong, A. (1992). A note on importance sampling using standardized weights. Technical Report 348, University of Chicago, Dept. of Statistics.

[7] Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. In *Neural Information Processing Systems*, pages 625–632.

[8] Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881.

[9] Low, K. H., Yu, J., Chen, J., and Jaillet, P. (2015). Parallel Gaussian process regression for big data: Low-rank representation meets Markov approximation. In *AAAI Conference on Artificial Intelligence*.

[10] Ng, J. W. and Deisenroth, M. P. (2014). Hierarchical mixture-of-experts model for large-scale Gaussian process regression. *arXiv preprint arXiv:1412.3078*.

[11] Smola, A. J. and Bartlett, P. (2001). Sparse greedy Gaussian process regression. *Neural Information Processing Systems*, 13:619–625.

[12] Snelson, E. and Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. In *Neural Information Processing Systems*, pages 1257–1264.

[13] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Neural Information Processing Systems*, pages 2951–2959.

[14] Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, volume 5, pages 567–574.

[15] Tresp, V. (2000). A Bayesian committee machine. *Neural Computation*, 12(11):2719–2741.

[16] Williams, C. K. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

[17] Williams, C. K. and Rasmussen, C. E. (1996). Gaussian processes for regression. In *Neural Information Processing Systems*, pages 514–520. MIT Press.