Using Scalable Gaussian Processes for Online Learning

Feasibility Study - MATH 7270 Project

Maxwell Jenquin - May 2nd, 2019

This project report details an investigation into the speed of KISS-GP, a scalable Gaussian process approximation which offers data-linear training and constant-time predictions with few assumptions on the covariance structure of input data. In particular, the suitability of this method for online time series regression is probed, and found to be sufficient for many tasks even without GPU acceleration. After discussing scaling relative to the naïve implementation, KISS-GP trained with an L-BFGS optimizer is put to the test on two noisy regression tasks with different challenges. Results indicate that up to 900 new inputs per second may be handled without issue, and parallel implementation is sure to provide even faster methods.

1 Introduction

Gaussian processes (GPs) are one of the fundamental tools of Bayesian data modeling. These models are attractive due to their flexibility, analytical tractability, and the intuitive way they encode inductive biases in practice: as Bayesian kernel machines.

As argued by Wilson and Nickisch [1], these qualities of Gaussian process models make them interesting options for big data, and by extension online applications where data arrive and models must be deployed, fit, and evaluated in real-time. However $\mathcal{O}(n^3)$ cost and $\mathcal{O}(n^2)$ storage required for the naïve implementation renders most problems beyond a few thousand data points intractable, necessitating various approximations and structure-exploiting methods for reducing computational expense. The majority of previous work on GP models for online contexts rely on specific covariance structure assumptions to achieve fast updates or low-order training. However KISS-GP, introduced in [1], provides scalable GP inference with fewer such assumptions. The goal of this project is to probe the viability of KISS-GP for use in an online context.

1.1 Gaussian Processes

A GP is defined mathematically as a stochastic process $f = \{f_i\}_{i\in\mathcal{I}}$ such that any finite subset of its random variables has a joint multivariate Gaussian distribution (hence Gaussian process). That is, for any such subset we have

$$f \sim \mathcal{GP} \Rightarrow \boldsymbol{f} = \{f_{i_1}, \cdots, f_{i_n}\} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

for some mean vector $\boldsymbol{\mu}$ and covariance matrix Σ. Therefore, a Gaussian process model for a scalar function f is specified by defining a functional form for the mean $\mu(\cdot)$, and a covariance function or "kernel", $k(\cdot,\cdot)$ such that the covariance between $f(x_1)$ and $f(x_2)$ is $k(x_1, x_2)$.

Regression for a GP model is performed by optimizing the marginal log-likelihood,

$$\mathcal{L}(\theta) = p(\boldsymbol{y}|\boldsymbol{x}, \theta) = -\frac{1}{2}\log|K + \sigma^2 I| - \frac{1}{2}\boldsymbol{y}^T(K + \sigma^2 I)^{-1}\boldsymbol{y} - \frac{n}{2}\log 2\pi.$$

where σ is a noise variance estimate, data are given by (x_i, y_i), and K is the matrix defined by $K_{i,j} = k(x_i, x_j)$. Optimization is performed with respect to σ and the hyperparameters of $k(\cdot,\cdot)$ and $\mu(\cdot)$, typically a length scale and scale factor. Note that $\mu = 0$ is the standard choice for

mean functions, yielding no hyperparameters. Many common kernels are stationary, meaning they are functions only of distance between inputs - this is a central assumption made by KISS-GP. The expense of training the model is rooted in the computation of the log-determinant and the linear solve in terms 1 and 2 of the log-likelihood. In the naïve implementation this is performed using a Cholesky decomposition, which is $\mathcal{O}(n^3)$.

Once regression has been completed, predictive distributions are analytically available: Given some set of test inputs $\boldsymbol{x}^*$, the result is a Gaussian distribution $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{x}, \boldsymbol{y}) = \mathcal{N}(\mu^*, \Sigma^*)$, where

$$\mu^* = \boldsymbol{k}^{*T}(K + \sigma^2 I)^{-1}\boldsymbol{y}, \quad \Sigma^* = k(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}^{*T}(K + \sigma^2 I)^{-1}\boldsymbol{k}^* + \sigma^2 I$$

and $\boldsymbol{k}^*$ is the matrix comprised of column vectors $k(\boldsymbol{x}^*, x_i)$ for each $x_i \in \boldsymbol{x}$. It is precisely because this analytic form of the predictive distribution is available that GP models are useful and interesting to data scientists and mathematicians alike.

1.2 The Challenge of Online Learning

Typically when tackling a problem in online learning, constant-time model updates are the goal of a proposed method. Such methods scale well with data to arbitrary size, and allow storage to become the primary bottleneck in practice. In contrast, Gaussian processes are globally-fit models, where optimal hyperparameters vary with the entire dataset and are not amenable to local variation or sequential updates without significant approximation in the style of Csató and Opper [2]. Therefore an online capable GP model based on KISS-GP must inevitably settle for stitching together smaller models which are local in time or "windowed", providing a change surface between these models on their overlap. Fortunately because linear combinations of Gaussian processes are themselves Gaussian processes, this is a simple task, but nonetheless requires vastly improved efficiency over $\mathcal{O}(n^3)$ in order to take place in real-time and scale to realistic rates of data acquisition.

2 Related Work

As early as 2002, Csató and Opper [2] proposed a "sparse" approach to GP models for online contexts, using data sub-sampling and a Bayesian update algorithm to approximate the posterior distribution. Using variational inference and KL-divergence estimates, the authors re-parameterize their posterior approximation and enforce sparsity by projecting onto a low-dimensional subspace of the feature Hilbert space implied by the kernel function. Connections were drawn between an equivalent model and kernelized least mean squares regression in 2016 by Van Vaerenbergh et al. [3], although kernelized least mean squares does not provide posterior distribution estimates.

In 2009, Nguyen et al. [4] developed a "local" approach to GP models for online adaptive control problems, namely to partition data into a large number of smaller models, using a Gaussian kernel distance metric

$$w_k(x) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_k)^T W (\boldsymbol{x} - \boldsymbol{c}_k) \right)$$

to pre-classify new data before re-fitting a small model at a much lower expense than a global model. That is, an incoming data point $\boldsymbol{x}$ was pre-clustered to one of k smaller models with means $\boldsymbol{c}_k$ before re-fitting and inference were performed, and predictions from this conglomerate model could then be made using the (diagonal) weight matrix W to combine predictions of local

models.

A sparsity-based GP model aimed at online contexts was developed in 2011 by Ranganathan et al. [5], exploiting a property of commonly used kernel functions - namely decay with respect to distance between inputs. This model assumed sparsity of the covariance matrix, allowing for efficient storage and updates in the form of an upper-triangular Cholesky factor (non-singular covariance matrices always have such a factorization). Using training example pruning (referred to by the authors as "downdates") the model could be updated in constant time using a moment-based KL-divergence minimization between the analytic posterior and the model. For this approach, hyperparameter training was reduced in cost to $\mathcal{O}(n)$.

So far a common theme in this existing work has been assumptions on covariance structure. One early attempt (2005) at an efficient GP implementation which assumed no structure was made by Leithead et al. [6]. At the time the "default" optimization routine was Polak-Ribère nonlinear conjugate gradient descent, a status cemented by its use in code which accompanied Rasmussen and Williams' popular textbook [7][8]. To drastically reduce the number of $\mathcal{O}(n^3)$ tasks required during hyperparameter tuning, a BFGS quasi-newton descent scheme was proposed, which iteratively updated the inverse covariance matrix so as to avoid inverting it at each optimization step. In addition an efficient and error-bounded log-det approximation was employed to complete the method. While this approach required plenty of intellectual overhead the performance gains were around a factor of 5-9 (though the method was competing with now-outdated implementations).

Another fully dense and global model was developed by Ambikasaran, Greengard et al. [9] in 2014, inspired by matrix approximation techniques developed from the fast multipole algorithm. This approach offers $\mathcal{O}(n \log^2 n)$-cost covariance matrix inversion and fast determinant evaluation, providing further tractability in high dimensions and for large data using few assumptions about the kernel function. This approach is promising for online data analysis given its complexity scaling but is not investigated here.

3 Structured Kernel Interpolation and KISS-GP

Inducing point methods such as those described in a unified framework by Quiñonero-Candela and Rasmussen in 2005 [10] seek to reduce the complexity of GP inference by choosing $m \ll n$ points in the data domain to act as anchors, interpolating the model from those points to the full data set. This concept proved powerful enough to gather interest for years to come, and in 2015 a generalization of these methods, structured kernel interpolation (SKI), was introduced by Wilson and Nickisch [1]. These methods all approximate the kernel of a GP model using some covariance matrix factorization, but SKI does so in a unified, structure-exploiting manner applicable to other kernel machines. Specifically, if $\boldsymbol{x}$ is the set of n parameter-space locations of training data and U is an evenly spaced grid of size m on the same domain, SKI makes the approximation

$$[k(\boldsymbol{x}, U)] = K_{\boldsymbol{x},U} \approx W K_{U,U}$$

for some weight matrix W. When combined with existing inducing point concepts this yields full covariance matrix factorizations in terms of W and grid-spaced $K_{U,U}$.

To obtain the methodology labeled KISS-GP, W is chosen as an extremely sparse matrix (e.g. linear or local cubic interpolation from the grid to $\boldsymbol{x}$), and the underlying GP model is required to have a stationary kernel (one depending only on distance between inputs). In that situation, the matrix $K_{U,U}$ will have Toeplitz structure (in 1 dimension) or Kronecker structure (in higher

dimensions). Both of these structures can be exploited for fast linear solves and matrix-vector products. In this project we apply KISS-GP to time-series data, so we discuss only Toeplitz structure exploitation.

3.1 Fast Toeplitz Methods

As detailed by Wilson in 2014 (though certainly known previously) [11], Toeplitz matrices K of size m can be uniquely embedded into circulant matrices C of size $2(m+1)$:

$$C = \begin{bmatrix} K & S \\ S^T & A \end{bmatrix} \Rightarrow C \begin{bmatrix} \boldsymbol{w} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} K\boldsymbol{w} \\ \mathbf{0} \end{bmatrix}$$

Thus, because the Fourier transform diagonalizes circulant matrices, matrix-vector products $K\boldsymbol{w}$ can be computed in $\mathcal{O}(m \log m)$ computations and $\mathcal{O}(m)$ memory, which is the complexity of the transform itself. In addition, linear solves may be performed with the same complexity, using preconditioned conjugate gradients, which make use of only matrix-vector products and converge to machine precision typically after a very small number of iterations $j \ll m$ for matrices with well-clustered eigenvalues. In practice, this process is very fast even for large problems and a number of grid points m approaching n.

Using this structure in $K_{U,U}$, a KISS-GP model requires $\mathcal{O}(mn + m \log m)$ computation to train, and a Lanczos estimate-based approximation for computing variances provides predictions in complexity which is constant with respect to n after pre-computation (which is linear in n).

4 Experiment Design

To examine the suitability of KISS-GP for online contexts, two data-generating functions were created to challenge a simple stationary kernel in different ways:

$$g_1(t) = 50\cos(t/25)\cos(t/64) + \sqrt{t}\sin\left(\frac{\sqrt{t}}{2\log t}\right), \qquad t > 1$$

$$g_2(t) = 10t^{\frac{1}{4}}\cos\left(\frac{t^{1.1}}{75}\right) + 2t^{0.4}\sin(t/4) \qquad t > 0$$

These functions are both quasi-periodic and would confound a simple periodic kernel, and so the GP model employed used a kernel with the structure:

$$k(x, x') = k_{\text{rbf}}(x, x')k_{\text{per}}(x, x') = \alpha \exp\left(-\frac{\|x - x'\|}{2\beta^2}\right)\exp\left(-\frac{2\sin^2(\pi\|x - x'\|/p)}{\gamma^2}\right)$$

A Gaussian process with the above kernel will produce functions with a largely periodic structure but which vary spatially in a smooth fashion. Hyperparameters of the kernel function are α (scale), β (length scale of smooth variation), p (periodicity), and γ (length scale of periodic term). As usual the mean function was assumed to be zero.

The challenges posed by g_1 and g_2 are evident from their plots (see figure 1) - g_1 has no clear sinusoidal signal, although its length scale is consistent and quasi-periodicity is visible. Meanwhile g_2 varies on two length scales, each with its own period, and exhibits additional signal growth. These functions are designed to be "just difficult enough" for the GP model to learn such that it can match its hyperparameters to concrete aspects of the generating function, but not without ambiguity and structural subtlety that confounds the results at poor local optima, especially in the presence of noise.

Two phases of experiments were performed. In the first phase, noise-free data of size 5,000 were sampled from both generating functions, once with grid spacing of size 1 and five times with exponential inter-arrival spacing with mean 1, e.g. distributed as

$$(t_{i+1} - t_i) \sim \exp(-(t_{i+1} - t_i)), \text{ for } t_{i+1} > t_i$$

For each methodology examined, a model was fit to the first k data points in the 5,000-sample data set, for each of the twelve data sets (six for each generating function, one being grid-spaced and five having different exponential spacing). Then an evenly spaced grid of 500 points was used as an evaluation set. For each generating function, fit times and relative pointwise error on the evaluation set were reported, and averaged across the exponentially-distributed data sets. This experiment gave an account of the scaling observed for each methodology, informing the choice of optimization scheme used in phase 2. The functions g_1 and g_2 were designed through trial and error, with phase 1 of experimentation as the proving ground for fit quality.

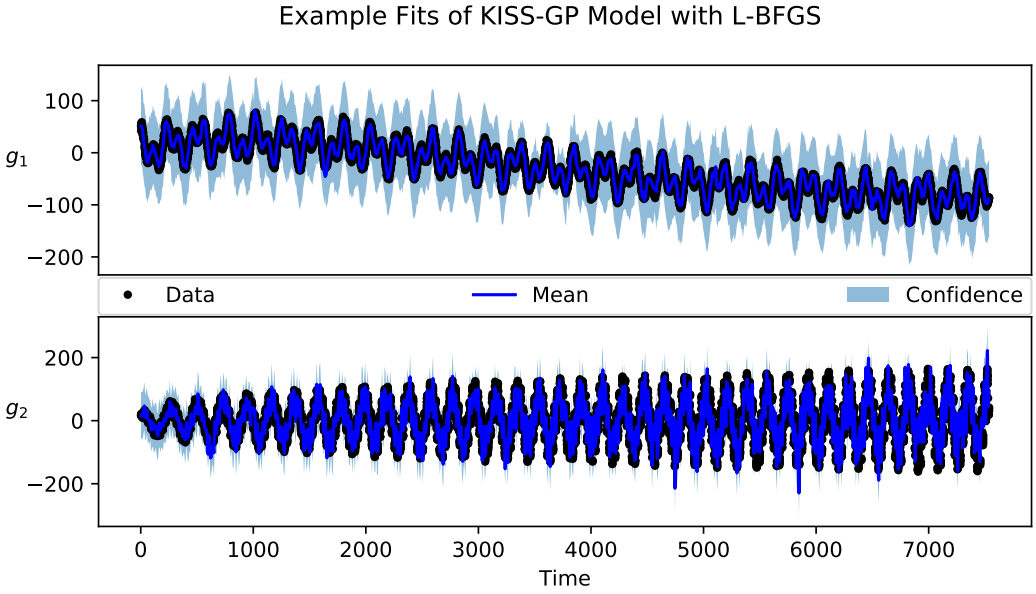

Figure 1: Example fits of the KISS-GP model trained with L-BFGS to 7,500 data points executed as in phase 2, with predictive mean and 95% confidence interval of the posterior distribution plotted alongside generating data from g_1 (top) and g_2 (bottom).

In phase 2, the most efficient methodology was put to a more strenuous scaling test. In increments of 1,000 data points and up to a training set size of 30,000, 30 noisy samples (pointwise-independent normal noise with variance 3) were taken from one of the generating functions, each with independently-generated exponential spacing. Fits were performed on each of these recording both the average time-to-evaluation as a function of data size and the fit quality (measured as the mean point-wise relative error on the evaluation set). This phase provided insight into the rate of data acquisition that a KISS-GP model could "keep up with" for each generating function. Due to the ratio of length scale to signal amplitude involved, it was expected that fits to g_2 would suffer more in accuracy than fits to g_1 in the presence of minor noise.

Methodologies investigated were the naïve implementation, trained with the Adam optimizer [12] using 150 training iterations, and the described KISS-GP implementation, trained three ways: with the same Adam optimizer at 150 iterations, the Adam optimizer at 50 iterations (observed to be sufficient), and a GPyTorch-implemented L-BFGS quasi-Newton optimizer with

careful exit conditions. Here we only mention that Adam is a stochastic gradient-based method with low-order moment correction which is suitable for large problems.

4.1 GPyTorch Implementation

Computation for this project used the Python library GPyTorch [13], built on the popular machine learning framework PyTorch. This library implements KISS-GP and many other techniques in cutting edge GP modeling in an efficient framework, and was chosen for those reasons and because it is amenable to GPU parallelization for future work and further efficiency gains. Times presented in the results section are wall-clock times as measured on a Dell Optiplex 7050 tower with an Intel I7-7700 CPU @3.60 GHz and 16GB DDR4 RAM.

5 Results

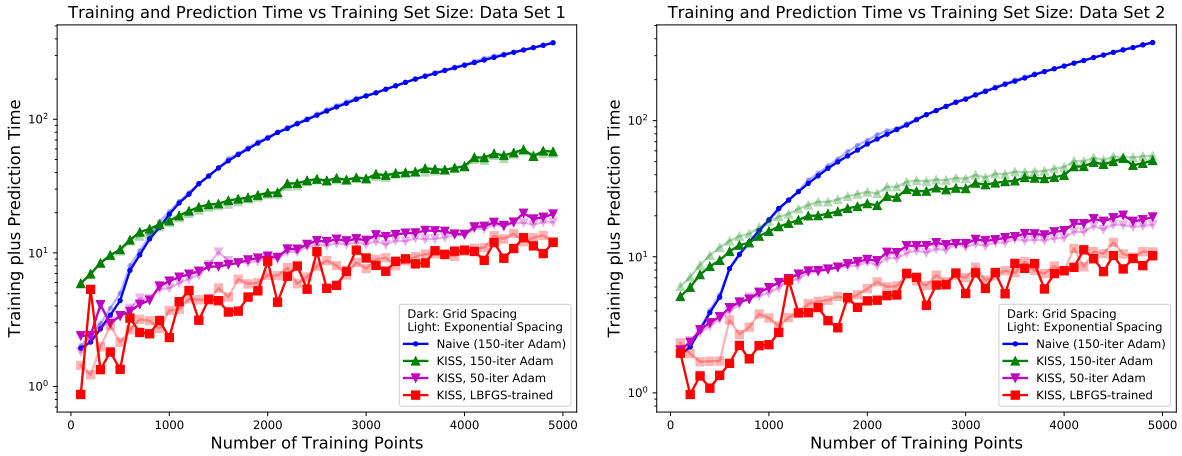

Figure 2: Phase 1 results for data sets drawn from g_1 (left) and g_2 (right). Note that each color corresponds to one methodology, with the lighter data representing the irregularly-spaced trials and the darker data representing the grid-spaced trials. Post-2,000 data point slopes are approximately 10^{-4}. (naive: 5.5, KISS-150: 2.3, KISS-50: 2.0, KISS-LBFGS: 2.6).

Results of phase 1 testing are found in figure 2, and provide a clear ranking of methodologies by efficiency as n increases - also notice that no clear differences in training time are present between data sets for this phase. As expected the naïve implementation slows significantly as n approaches 5,000, but the KISS-GP models suffer much less. The 150-iteration Adam-trained model performs slower than its 50-iteration counterpart, as expected (150 iterations were needed for the naïve method to converge while only 50 were needed for KISS-GP, hence the inclusion of both for comparison). Meanwhile the L-BFGS optimizer's ability to exit early when converged combined with fast convergence near local optima provides a higher-variance but overall lower training time than the Adam optimizer. L-BFGS training frequently converged in less than 10 iterations, often 5 or fewer.

As such, an L-BFGS trained KISS-GP model was the methodology used for phase 2 testing, the results of which are summarized in table 1. Here it becomes clear that the model fit data from g_1 significantly better, as evidenced by the mean error over the 30 trials per n. However, it should be noted that the high mean errors for the g_2 data set come from more frequent outliers, and that when properly converged the model accurately estimated g_2 as well. From the average

time to fit each data set size, hypothetical input rates measured in data points per second were calculated assuming a desired 5% overlap between consecutive model "windows", and are the main results of this project.

	Data from $g_1(t)$			Data from $g_2(t)$		
n (thousands)	time (s)	mean error	input rate	time (s)	mean error	input rate
1	2.398845	2.529934	396	2.095944	1.098058	453
5	5.140201	1.147658	924	6.75071	1.213397	704
10	11.08531	0.797631	857	13.65495	1.819256	696
15	15.59625	0.611085	914	19.11884	1.147181	745
20	28.34102	0.718122	670	35.42328	1.483744	536
25	34.61692	0.601228	686	44.02118	2.296331	540
29	40.5879	0.528249	679	46.90926	0.716015	587

Table 1: Selected phase 2 results for g_1 and g_2 using the KISS-GP model optimized with L-BFGS, along with calculated approximation to average rate of data input (in new data points per second) the model could keep up with at a window size of n.

From these input rates, it is clear that efficiency peaks for a window size of about 5,000 data points, and then again for a window size of about 15,000 data points, with a well-suited model (as in the case of this model applied to data from g_1) reaching over 900 new inputs per second as its maximum capacity.

6 Discussion and Conclusions

This project report presented a study of KISS-GP in the context of online time series regression, and investigated the possibility of designing a methodology for such problems that relied on very few assumptions about the covariance structure of underlying data.

At the maximum observed input rate of approximately 900 new samples per second, a large swath of online learning tasks are accessible, although tasks involving signal processing (and therefore a high-fidelity sample rate) are still out of scope. Still, between small savings in the software design process and the potentially huge speed boost provided by GPU parallelization, it is likely that KISS-GP is capable of keeping up with all but the most strenuous of online tasks. Further savings in intensive applications may be found when data are known to arrive at fixed time intervals, in which case the covariance matrix has a natural Toeplitz structure without interpolation, removing several stored variables and many function calls from the process. In short, KISS-GP is indeed sufficiently fast and appropriately scalable for online contexts.

References

[1] A. Wilson and H. Nickisch, "Kernel interpolation for scalable structured gaussian processes (kiss-gp)," in *International Conference on Machine Learning*, pp. 1775–1784, 2015.

[2] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.

[3] S. Van Vaerenbergh, J. Fernandez-Bes, and V. Elvira, "On the relationship between online gaussian process regression and kernel least mean squares algorithms," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, 2016.

[4] D. Nguyen-Tuong, J. R. Peters, and M. Seeger, "Local gaussian process regression for real time online model learning," in *Advances in Neural Information Processing Systems*, pp. 1193–1200, 2009.

[5] A. Ranganathan, M.-H. Yang, and J. Ho, "Online sparse gaussian process regression and its applications," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 391–404, 2011.

[6] W. Leithead, Y. Zhang, and D. Leith, "Efficient gaussian process based on bfgs updating and logdet approximation," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 1305–1310, 2005.

[7] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT Press Cambridge, MA, 2006.

[8] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*, pp. 63–71, Springer, 2003.

[9] S. Ambikasaran, D. Foreman-Mackey, L. Greengard, D. W. Hogg, and M. O'Neil, "Fast direct methods for gaussian processes," *arXiv preprint arXiv:1403.6015*, 2014.

[10] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[11] A. G. Wilson, *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] 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*, 2018.