Gaussian Processes for Classification

Vamshi Kumar Kurva [1]

Abstract

Gaussian processes are non-parametric, probabilistic methods that defines distribution over hypothesis functions. They are probabilistic in the sense that GPs can capture uncertainties in the predictions by modifying the prior based on the data available. However, this probabilistic nature of GPs demand more computational resources and tricks to make it work on huge datasets. This document provides a basic introduction to Gaussian processes, how they are different from traditional ML algorithms, challenges in implementing. We also explore some of the approximation methods and computational tricks to make it implementable and work on huge datasets.Finally we discuss about the connection between deep learning and GPs and how we can improve GP's representation capacity using deep kernels.

1. Introduction

In supervised problems we typically make some assumptions about the data (for e.g in linear regression, we assume that the output is a linear function of features). The assumption fixes our model and the number of parameters. When we observe some data, we find the best parameters by reducing some kind of objective function. These are known as parametric methods. As the complexity of the problem increases, models with higher number of parameters are needed to explain the data well. Non-parametric methods are those which uses the number of parameters depending on the data at hand. Gaussian processes in this sense are non-parametric.

Gaussian processes are generalisation of Gaussian distribution. GPs define distribution over finite random variables which are jointly gaussian. Generally, random variables in a stochastic process are indexed over time, but here random variables are indexed over input space and represent the function values at the given location (Rasmussen & Williams, 2005). Hence, it can be said that GPs define distribution over functions. GPs can be completely specified by the mean vector and covariance matrices. GPs place a prior over a set of hypothesis functions and refines the distribution as we observe more and more data. In this sense, GPs are Bayesian/Probabilistic methods. Since GPs are probabilistic, they can capture the uncertainties in the predictions.

[1]Department of CSA, IISc, Bangalore. Correspondence to: Vamshi Kumar Kurva <vamshikurva@iisc.ac.in>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

2. Literature Survey

To give a flavor of GP classification, we give a brief overview of GPs in regression setting.

2.1. GP Regression

In case of GP regression, the prior over the hypothesis functions is Gaussian. Characteristic of hypothesis functions can be changed by changing the covariance matrix, which can be calculated using the kernel function. A commonly used kernel is RBF kernel with two parameters variance and length. As the length increases, farther points in the space gets more correlated and the functions becomes smoother.

$$k(x, x') = \sigma^2 \exp\Big(-\frac{\|x - x'\|^2}{2l^2}\Big)$$

Any other kernel which gives symmetric positive definite covariance matrix can also be used. Once we observe some data, we can calculate the posterior by conditioning over the observed data. Since Gaussian distributed is closed under conditioning (Görtler et al., 2019), we get a nice analytical formula for calculating the posterior which is given by

$$f_*/f \sim \mathcal{N}(k_*^T K^{-1} f, k_{**} - k_*^T K^{-1} k_*)$$

Here (X, f) is the training data, $k_{**} = K(X_*, X_*), k_* = K(X, X_*)$ and X_* is the test data and f_* are predictions corresponding to the test data. Computing the posterior can be intuitively thought of as sampling from the prior and rejecting the samples that doesn't agree with the observed data (Rasmussen & Williams, 2005). One problem with the computation of posterior is that it involves the calculation of inverse of a covariance matrix. Since calculating inverse is of $\mathcal{O}(N^3)$ complexity, it doesn't scale well with the number of observations. Hence one need to use some computational tricks to calculate inverse as the number of observations

becomes huge. We address this problem in the coming sections.

2.2. GP Classification

In the case of binary classification, output takes discrete values (say $y = +1$ for positive and $y = -1$ for negative) unlike in the regression case where the target variable takes values from real line. Hence we can't directly define a gaussian prior over the target values. The idea is to define a prior over real line and apply a squashing function to limit the values between 0 and 1 to represent the probability. This defines the prior over the target values. It can be thought of as defining probabilities over probabilities.

$$\pi(x) \coloneqq p(y = +1/x) = \sigma(f(x))$$

Here f can be called as latent function since we don't observe the values of f itself. Intuitively, latent function can be thought of as logits (just like in logistic regression case). GP classification is a natural generalisation of logistic regression. Inference is done in two stages. First is to compute the distribution of latent function corresponding to the test samples

$$p(f_*/X, y, x*) = \int \underbrace{p(f_*/X, y, f)}_{\substack{\text{conditional gaussian} \\ \text{from GPR}}} \underbrace{p(f/X, y)}_{\substack{\text{posterior over} \\ \text{latent variable f}}} df \quad (1)$$

Next is to produce a probability prediction using f_*

$$\bar{\pi}_* = p(y_* = 1/X, y.x_*) = \int \sigma(f_*) p(f_*/X, y, x*) df_* \quad (2)$$

The posteriror over latent function after observing the training data (X, y) is given by

$$p(f/X, y) = \frac{p(y/f)p(f/X)}{p(y/X)} \quad (3)$$

where

- $p(f/X)$ - prior given by $\mathcal{N}(0, K_{XX})$
- $p(y/f)$ - likelihood function

$$= \left\{ \begin{array}{ll} \sigma(f) & y = +1 \\ 1 - \sigma(f) = \sigma(-f) & y = -1 \end{array} \right\} = \sigma(yf)$$

- $p(y/X)$ - normalisation constant (independent of f)

Even though prior is gaussian, since the likelihood is non-gaussian, (3) is non-gaussian and this makes the integral in (1) analytically intractable. Similarly (2) is also analytically intractable.

Algorithm 1 Laplace approximation

Input: $p(f/X, y)$ without normalisation constant
$\hat{f} = \arg\max_f p(f/X, y)$
$A = -\nabla^2 p(\hat{f}/X, y)$
$q(f/X, y) \sim \mathcal{N}(f; \hat{f}, A^{-1})$
return $q(f/X, y)$

3. Method

As we have seen the integrals in case of GP classification becomes intractable because of the non-gaussian likelihood. Hence one needs to approximate the integrals to make it tractable. One can approximate the non-gaussian posterior over latent values with the gaussian using the Laplace approximation method.

3.1. Laplace Approximation

Laplace approximation is a method to approximate the distributions for which it is difficult to compute the normalisation constant. In our case also it's difficult to compute this constant since it involves integration over non-gaussian distribution. The idea is to approximate with a gaussian distribution with the mean given by the mode of the distribution we want to approximate. Using the second order taylor approximation of $\log p(fX, y)$ and setting the derivative to zero gives us

$$\hat{f} = K(\nabla \log p(y/\hat{f}))$$

The above equation can't be solved directly, since it is a self-consistent equation, i.e. $\hat{f}$ is expressed in terms of $\hat{f}$ itself. The hessian of the log posterior is given by

$$\begin{aligned} \nabla^2 \log p(f/X, y) &= \nabla^2 \log p(y/f) - K^{-1} \\ &= -(W + K^{-1}) \end{aligned}$$

Hessian of the likelihood $\nabla^2 \log p(y/f)$ is diagonal since y_i depends only on f_i given f_i and and positive definite since sigmoid is a strictly increasing function. This makes the hessian of the log posterior negative definite and hence log posterior is strictly concave and has a unique maxima. We can reach the mode iteratively using Newton's method with the following update equation

$$\begin{aligned} f_{k+1} &= f_k + (K^{-1} + W)^{-1}(\nabla \log p(y/f_k) - K^{-1} f_k) \\ &= (K^{-1} + W)^{-1}(W f_k + \nabla \log p(y/f_k)) \end{aligned}$$

Laplace approximation is simple, but may not a be a good approximation to the true shape of the original distribution if the original distribution is multi-modal or has a skewed peak (Rasmussen & Williams, 2005). Since the log posterior in our case is strictly concave and has a unique maxima,

Laplace approximation may not be so bad. Now, the approximate posterior over the latent values corresponding to the test samples x_* is given by

$$q(f_*/X, y, x_*) = \int p(f_*/X, y, f)q(f/X, y)df$$
$$\sim \mathcal{N}(f_*/\mu_*, \Sigma_*)$$

where

- $\mu_* = k_*^T K^{-1}\hat{f} = k_*^T \nabla \log p(y/\hat{f})$
- $\Sigma_* = k_{**} - k_*^T K^{-1}\hat{f} + k_*^T K^{-1}(K^{-1} + W)^{-1}Kk_*$

Now that we have approximated the posterior over latent values with the gaussian, we can proceed to predict the mean probability of belonging to positive class.

$$\bar{\pi}_* = p(y_* = 1/X, y.x_*) = \int \sigma(f_*)\mathcal{N}(f_*/\mu_*, \Sigma_*)$$
$$= E_{\hat{f}\sim\mathcal{N}(\mu_*,\Sigma_*)}(\sigma(f_*))$$

The expectation can be approximated using Monte Carlo method by generating samples from the distribution and taking the sample average of function values at the samples.

4. How to scale GPs for Big Data?

Various techniques have been proposed over the years on how to scale GPs for huge datasets. Techniques range from using low rank approximations of kernel matrix to using a subset of original data. We give a brief overview about some of these techniques here

4.1. Nystrom Approximation

Nystrom method approximates the PSD matrix with a low rank matrix. We randomly sample $m << n$ points from original data points and assuming that (without loss of generality) the dataset is arranged such that the selected m points come first, the kernel matrix can be written as

$$K = \begin{pmatrix} K_{mm} & K_{m(n-m)} \\ K_{(n-m)m} & K_{(n-m)(n-m)} \end{pmatrix}$$

Nystrom approximation for K is given by (Williams & Seeger, 2001)

$$\tilde{K} = K_{nm}K_{mm}^{-1}K_{mn}$$

K_{mm} is symmetric and positive definite and hence can be eigen decomposed as $K_{mm} = PDP^T$ where P is orthonormal matrix. Nystrom approximation can be written as

$$\begin{aligned}\tilde{K} &= K_{nm}K_{mm}^{-1}K_{mn} \\ &= K_{nm}(PD^{-1}P^T)K_{mn}^T \\ &= (K_{nm}PD^{-\frac{1}{2}})(K_{nm}PD^{-\frac{1}{2}})^T \\ &= Q_{nm}Q_{nm}^T\end{aligned}$$

Now using the matrix inversion lemma the inverse can be calculated as

$$(QQ^T + \sigma_n^2\mathbf{I}_n)^{-1} = \sigma_n^{-2}\mathbf{I}_n - \sigma_n^{-2}Q(\sigma_n^2\mathbf{I}_m + Q^TQ)^{-1}Q^T$$

Note that the above equation only involves the calculation of $m \times m$ matrix. Time complexity to compute the Nystrom approximation is $\mathbf{O}(m^2n)$. Here the Kernel matrix is not systematically replaced by $\tilde{K}$ and hence it may lead to approximated predicted variance to be negative which shouldn't be the case. (Liu et al., 2019).

4.2. Bayesian Committee Machines

BCMs were introduced in to faster GP Regression (Tresp, 2000). The idea is to split the data into partitions and model each partition using a GP and then combine the predictions of all GPs in some probabilistic way during inference. Unlike the other methods which only relies on a subset of data, this method makes use of all the points in the dataset, although it breaks the kernel that we care about. Let $\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_p$ be the p partitions of the dataset $\mathcal{D}$ and f_* be the latent value corresponding to the test sample x_*, then

$$p(f_*/\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_p) \propto p(\mathcal{D}_1, \mathcal{D}_2, ...\mathcal{D}_p/f_*)p(f_*)$$

where $p(f_*)$ is the prior over latent function. BCM makes the conditional independent assumption that $\mathcal{D}_i \perp\!\!\!\perp \mathcal{D}_j/f_*$. Using the conditional independent assumption

$$\begin{aligned}p(f_*/\mathcal{D}_1, \mathcal{D}_2, ..\mathcal{D}_p) &\propto p(\mathcal{D}_1/f_*)p(\mathcal{D}_2/f_*)...p(\mathcal{D}_p/f_*)p(f_*) \\ &\propto \frac{p(\mathcal{D}_1, f_*)}{p(f_*)}\frac{p(\mathcal{D}_2, f_*)}{p(f_*)}...\frac{p(\mathcal{D}_p, f_*)}{p(f_*)}p(f_*) \\ &\propto \frac{p(f_*/\mathcal{D}_1)p(f_*/\mathcal{D}_2)...p(f_*/\mathcal{D}_p)}{p^{p-1}(f_*)} \\ &= c\frac{\prod_{i=1}^p p(f_*/\mathcal{D}_i)}{p^{p-1}(f_*)}\end{aligned}$$

where c is a normalization constant. All the terms in the above equation are Gaussian distributions and hence it is easy to find the predictive mean and variance of f_* and are given by

$$E(f_*/D) = [cov(f_*/\mathcal{D})]\sum_{i=1}^p [cov(f_*/\mathcal{D}_i)]^{-1}E(f_*/\mathcal{D}_i)$$

$$[cov(f_*/\mathcal{D})]^{-1} = -(p-1)K_{**}^{-1} + \sum_{i=1}^p [cov(f_*/\mathcal{D}_i)]^{-1}$$

where K_{**} is the covariance matrix evaluated at test points. Dataset $\mathcal{D}$ can be partitioned in any ways. If m is the partition size we want, we will have $p = n/m$ partitions. We can use the $p-$mean clustering and use the clusters as partitions to get improved performance. Computational complexity of BCM is $\mathcal{O}(pm^3) = \mathcal{O}(m^2n)$.

4.3. Iterative solution of Linear system

Calculating the inverse problem $(K + \sigma_n^2 I)^{-1}$ can be posed as solving the linear system of equations $(K + \sigma_n^2 I)v = y$. Since the matrix is PSD, we can use conjugate gradient method to solve the linear system. To get the correct solution it will take atmost n steps. The idea is to stop the process after certain iterations $k < n$ to get an approximated solution. Time complexity in this case is $\mathcal{O}(n^2 k)$.

5. Deep Kernel GPs

GPs are powerful non-parametric probabilistic methods with the kernel function at it's core whose performance depends on the selection of the kernel function. However standard kernels are limited by their design criteria. More expressive kernels which can find the hidden structure in data automatically can be developed by using deep learning. GPs with deep kernels are referred to as Deep Kernel GPs. Deep Kernel GPs have the flexibility of non-parametric kernel methods combined with the structural properties of Deep learning architectures (Wilson et al., 2015).

5.1. Deep Kernel Learning

Deep Kernel Learning refers to the learning of base kernel hyper-parameters together with the DNN parameters. Let x_i, x_j are inputs, k is a base kernel function parametrized by θ. DKL involves transforming the inputs through non-linearity before applying the kernel.

$$k(x_i, x_j/\theta) \rightarrow k(g(x_i, w), g(x_j, w)|\theta, w)$$

where $g(x, w)$ is the non-linear transformation given a deep network architecture. Deep Kernel Hyper parameters $\{w, \theta\}$ i.e. network parameters w and base kernel hyper parameters θ can be jointly learnt by maximizing the log likelihood $\mathcal{L}$ of the training data under GP. Given the prior $f/X \sim \mathcal{N}(0, K)$ and likelihood $y/f \sim \mathcal{N}(0, \sigma_n^2 \mathbf{I})$, marginal likelihood can be written as $p(y/X) = \int p(y/f)p(f/X)df$. Therefore the marginal log likelihood is given by

$$\mathcal{L} = -\frac{1}{2} y^T (K + \sigma_n^2 \mathbf{I})^{-1} y - \frac{1}{2} \log(K + \sigma_n^2 \mathbf{I}) - \frac{n}{2} \log 2\pi$$

Partial derivatives w.r.to θ

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial K} \frac{\partial K}{\partial \theta}$$

Partial derivatives w.r.to w

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial K} \frac{\partial K}{\partial g(x, w)} \frac{\partial g(x, w)}{\partial w}$$

Absorbing the noise variance $\sigma_n^2 \mathbf{I}$ into the covariance matrix and treating it as a kernel hyper parameter θ, gradient w.r.to kernel matrix can be written as

$$\frac{\partial L}{\partial K_\gamma} = \frac{1}{2}(K_\gamma^{-1} y y^T K_\gamma^{-1} - K_\gamma^{-1})$$

Time complexity to invert the covariance matrix is still $\mathcal{O}(n^3)$. However, it can be used when the no of samples are very low, especially in low data regime. GPs predictive capabilities combined with the Deep kernels were proven effective in Few Shot Learning (Patacchiola et al., 2020).

6. Results

We have done the simplest of the experiments to understand the basics of GPs in regression and classification settings. Our main motive was to understand the kernel, how it affects the prior and posterior. We used only the RBF kernel in all of our experiments (keeping the variance as constant and varying the length hyper-parameter). Our first observation is that as the length increases farther points gets more correlated, the band of influence increases and the samples drawn gets smoother as can be seen from the first and second columns of figure(1). Also from the third column, we can see that posterior becomes more confident(confidence interval almost reduces to the mean) as the length increases. Best hyper parameters can be chosen by maximizing the marginal log likelihood of the data. Figure(2) emphasises the importance of kernel hyper-parameters. In this setting GP has been used for classification. It is evident from the figure that when the length is too less, algorithm failed to draw a proper decision boundary which separates the classes. However as the length parameters increases decision boundary became better and kind of non-linear. These simple experiments highlight the importance of kernel, its hyper-parameters and the power of GP models. Further experiments can also be conducted by using different kernels, non-linear data for classification and also on huge datasets, although we haven't tried out these experiments.

7. Conclusion

GPs are powerful probabilistic methods with the use of kernel function at it's heart. Like all kernel methods, GPs suffer from the problem of scaling with huge datasets because of the storage and computational requirements and like the other probabilistic methods, GPs suffer from intractable integrals. We have addressed some of these issues and provided few of the solutions. However this document barely scratched the surface and this is still an active area of research.

References

Görtler, J., Kehlbeck, R., and Deussen, O. A visual exploration of gaussian processes. *Distill*, 2019. doi: 10.23915/distill.00017. https://distill.pub/2019/visual-exploration-gaussian-processes.

Liu, H., Ong, Y.-S., Shen, X., and Cai, J. When gaussian

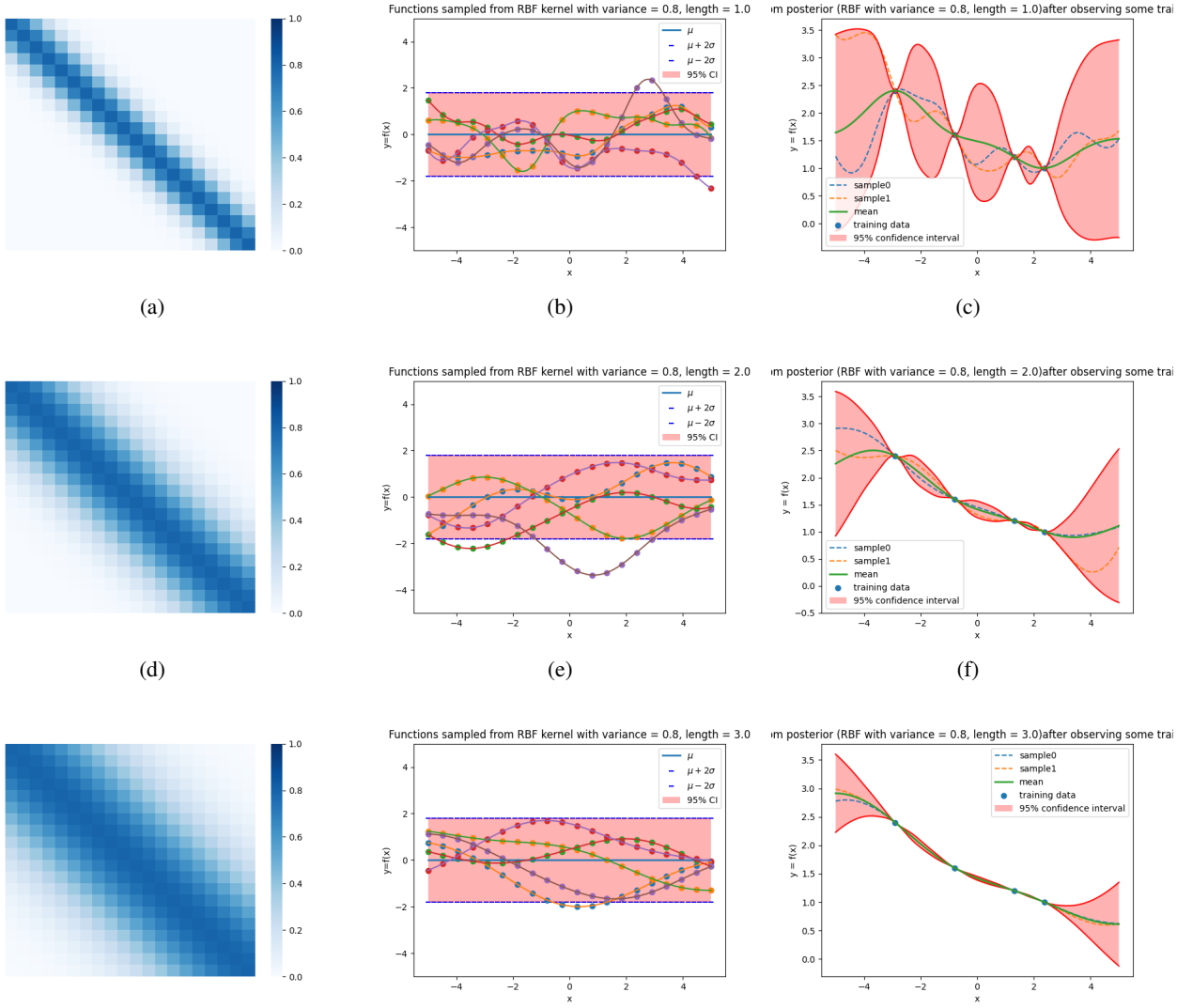

Figure 1. Each row corresponds to one configuration of the kernel(with the varying hyper-parameter length=1.0, 2.0, 3.0 respectively). The first column shows the band of influence . The second column shows 5 samples drawn randomly from the prior. The third column shows 3 samples drawn from the posterior after observing 4 training points along with the mean and 95% confidence interval.

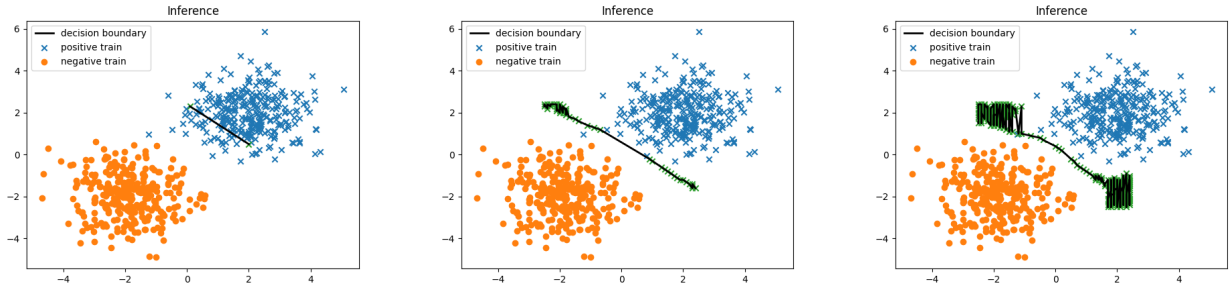

Figure 2. Decision boundaries corresponds to different kernel settings(length=0.5, 1.5, 2.5 respectively) using GPC.

process meets big data: A review of scalable gps, 2019.

Patacchiola, M., Turner, J., Crowley, E. J., O'Boyle, M., and Storkey, A. Bayesian meta-learning for the few-shot setting via deep kernels, 2020.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

Tresp, V. A bayesian committee machine. *NEURAL COMPUTATION*, 12:2000, 2000.

Williams, C. and Seeger, M. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pp. 682–688. MIT Press, 2001.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning, 2015.