Finite Rank Deep Kernel Learning

Sambarta Dasgupta, Kumar Sricharan, Ashok Srivastava
Central Data Science Organization, Intuit Inc.
{sambarta_dasgupta,sricharan_kumar,ashok_srivastava}@intuit.com

Abstract

Deep kernel learning has emerged as a principled framework to simultaneously forecast and characterize the extent of uncertainty in the forecast in regression problems. Deep kernel learning uses a deep neural network to learn the kernel operator of a Gaussian process, which is used to perform the inference. In this work, we propose a novel way of constructing the deep kernel operator from a set of mutually orthogonal embeddings, which are learned by a deep neural network. The proposed approach on one hand reduces the computational complexity of the deep kernel learning, and simultaneously enables the algorithm to forecast and characterize uncertainty more accurately for complex functions on the other.

1 Introduction

There have been several approaches to characterize uncertainty in forecasting using deep learning e.g. Dropout [1], Bayesian Neural Network [2], Ensemble-based [3], Calibration-based [4], Neural Processes [5], and Deep Kernel Learning [6, 7, 8]. Deep kernel learning (DKL) is a combination of deep neural network (DNN), and Gaussian process (GP). DKL combines the capacity of approximating complex functions by DNN with the flexible uncertainty estimation framework of GP. The GP utilizes a kernel operator, which could be perceived as a similarity function to capture the relationship across various parts of the dataset, to propagate the uncertainty across the dataset. Based on the geometry of the data, there will be an appropriate functional representation of the kernel operator e.g. linear, polynomial, rbf, sinusoidal etc. which can optimally represent the geometry. The selection of the functional representation of the kernel operator is a highly non-trivial task for an arbitrary dataset, as depending on the geometry of the data, one needs to identify the right functional form of the similarity or kernel. In the DKL framework, the DNN is used to learn an embedding, which is acted upon by a radial basis function (rbf) to create the GP kernel [6]. In this way, the kernel representation is made non-parameteric and can potentially model a richer family of kernels compared to the standard kernels used in a pure GP formulation. Another challenge of using GP is the computational complexity, which is $\mathcal{O}(n^3)$, where n is the size of the training data. In order to reduce the computational complexity, the authors [6] have proposed a scheme to approximate the kernel [6].

In this work, we propose a method to create expressive kernels that are capable of capturing complex geometry of the dataset while simultaneously being computationally efficient wrt training and inference. In particular, we construct kernels as a linear combination of a set of simple inner products between orthogonal embeddings which are learned from a DNN. The kernel decomposition ensures that each kernel captures the local geometry and the combined kernel captures the combined geometry of the dataset. This way, we simultaneously do clustering and regression in one framework. In addition, our proposed technique, by virtue of being a linear combination of orthogonal embeddings, has a reduced complexity of $\mathcal{O}(n^2)$ without any approximation of the kernel.

Third workshop on Bayesian Deep Learning (NeurIPS 2018), Montréal, Canada.

2 Finite Rank Deep Kernel Learning

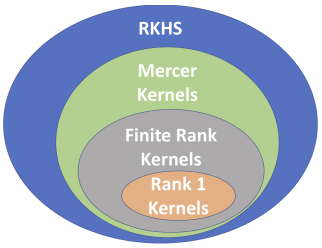


Figure 1: Hierarchies of kernels.

We start from the frameworks for the GP and the DKL. Because of the space limitations, we will briefly review DKL, and refer the reader to [9] (GP), [6] (DKL) for details. Next, we will describe our proposed finite rank deep kernel method, and conclude with experiments that showcase the advantages of the proposed method in terms of both accuracy and efficiency.

2.1 Hierarchies and Richness of Kernel Operators

We will briefly review the hierarchies of the spaces of symmetric and positive definite kernel operators [10]. The Moore Aronszajn Theorem proves that Reproducing Kernel Hilbert Space (RKHS) forms the space of the positive definite kernels [10]. Next, we will consider Mercer kernels, which are formed by a rich subspace of the RKHS, and are continuous in addition to being symmetric and positive definite. Mercer Theorem provides a decomposition of such an arbitrary kernel into the eigen functions, as $K(x,y) \sim \sum_{i=1}^{\infty} \zeta_i \Theta_i(x)\Theta_i(y)$, where Θ_i forms an continuous orthonormal basis of eigen functions, and the eigen values ζ_i's decay asymptotically. A Mercer kernel $K(x,y)$ will be of finite rank R, if it can be expressed as, $K(x,y) = \sum_{r=1}^{R} \zeta_r \Theta_r(x)\Theta_r(y)$. We also note that the rank -1 Mercer kernels forms a smaller subspace of the Mercer kernels, as it can be observed from Fig. 1. In DKL, the authors propose the use of kernel in the form $rbf(\phi(x), \phi(y))$, where the d-dimensional embeddings $\phi(.)$ are learned using a neural network. It can be shown, that rbf is a rank-1 Mercer kernel in the infinite dimension, i.e. in the infinite dimensional case rbf could be expressed as $rbf(x,y) = c(x).c(y)$ for some smooth function $c(.)$. However, to enable efficient inference, the authors have to use a sampling based approximation of this kernel. An alternative would be to use a rank-1 kernel with a finite dimensional embedding $\phi(x)'\phi(y)$, but this kernel does not have enough representational capacity to capture complex functions.

2.2 Finite Rank Kernels

In this work, our goal is to create a kernel with good representational power that is also efficient wrt inference. To this end, we will use finite rank Mercer kernel (of rank R) $K(x,y) = \sum_{r=1}^{R} \phi_r(x,\omega)\phi_r(y,\omega)$. We deploy the DNN (parameterized by ω) to learn the ϕ_rs, which are forced to be mutually orthogonal. We will refer to our method as Finite-Rank Deep Kernel Learning (FRDKL). Next, we will show that this FRDKL kernel simultaneously exhibits the ability to model complex signals while also being computationally efficient. Specifically, in comparison to DKL, FRDKL makes up for loss in representational capacity due to using a finite dimensional embedding through the fact that it employs a finite rank kernel (compared to a rank-1 kernel in DKL). In addition, the linear representation of the FRDKL kernel allows for $\mathcal{O}(n^2)$ inference complexity in FRDKL, which is achieved in DKL only through approximation.

2.2.1 Representational Power

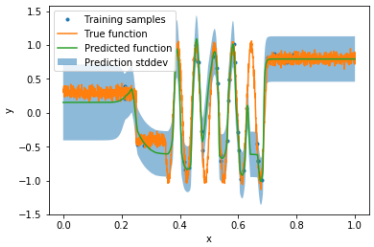
The kernel is modeled as $K(x,y) = \sum_{r=1}^{R} \phi_r(x,w)\phi_r(y,w)$, where $\phi_r(y,w)$'s will be orthogonal to one another for different r's. The DKL algorithm optimizes the negative log likelihood function, based on the kernel operator, $-log\ p(y|x) \sim y^T(K_\gamma + \sigma^2 I)^{-1}y + log|K_\gamma + \sigma^2 I|$. We introduce a penalty term to the cost, as following : $-log\ p(y|x) + \lambda \sum_{r_1,r_2,r_1 \neq r_2} (\phi_{r_1}(x,w)^T \phi_{r_2}(x,w))^2$, where, λ is a weight assigned to the orthogonality objective relative to the log likelihood. It is to be noted that $\sum_{r_1,r_2,r_1 \neq r_2} (\phi_{r_1}(x,w)^T \phi_{r_2}(x,w))^2$ is minimized when the embeddings are orthogonal. By learning this orthogonal representation over a finite rank of embeddings with rank greater than 1, we are able to more accurately capture the geometry of the underlying signal. In particular, these orthogonal eigen-functions would capture different local geometries of the dataset, which once combined would be able to capture the global geometry. Fig. 2a shows a synthetic dataset, which has been forecast very accurately by the FRDKL using a $d = 5$-dimensional embeddings of rank $R = 3$. The corresponding orthogonal embeddings are captured by Fig. 2b. It can be observed that the support of the embedding functions are almost disjoint and as a consequence they are approximately orthogonal.

2.2.2 Computational Complexity

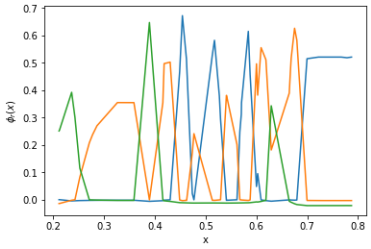
In this section, we would demonstrate the reduction of the computational complexity in the proposed method. For finite number of training data points, we define $\Phi_r = [\phi_r((x_1, w)), \dots \phi_r((x_n, w))]$, the kernel matrix is defined as $K = \sum_{r=1}^{R} \Phi_r . \Phi_r^T$. This orthogonality critically allows us to reduce the computational complexity, by allowing us to efficiently invert the kernel. Typically, evaluation of $K_* \left(K + \sigma^2 I\right)^{-1}$ is the computationally expensive part in GPs ($\mathcal{O}(n^3)$). In our case, by leveraging orthogonality of the embeddings to reduce $K^{-1}(x, y)$ to $\sum_{r=1}^{R} ||\Phi_r||^{-2} . \Phi_r \ \Phi_r^T$, the term $K_* \left(K + \sigma^2 I\right)^{-1}$ can be reduced to

$$K_* \left(K + \sigma^2 I\right)^{-1} = \left(\sum_{r=1}^{R} \Phi_r^* . (\Phi_r^*)^T \right) . \left(\sum_{r=1}^{R} \sigma^2 . ||\Phi_r||^{-2} . \Phi_r \ \Phi_r^T \right),$$

which is $\mathcal{O}(n^2)$ in complexity. The details of the matrix computations could be found in Appendix 5.2. We will show in our experiments that DKL, despite using sampling based approximations, is slower than FRDKL. Finally, we note that in addition to efficient inference, FRDKL also allows for efficient training through stochastic gradient descent by virtue of being able to decompose the loss function over the individual training points (details could be found in the Appendix 5.3).



(a) Sample non-smooth function with forecasts by FRDKL (proposed method).



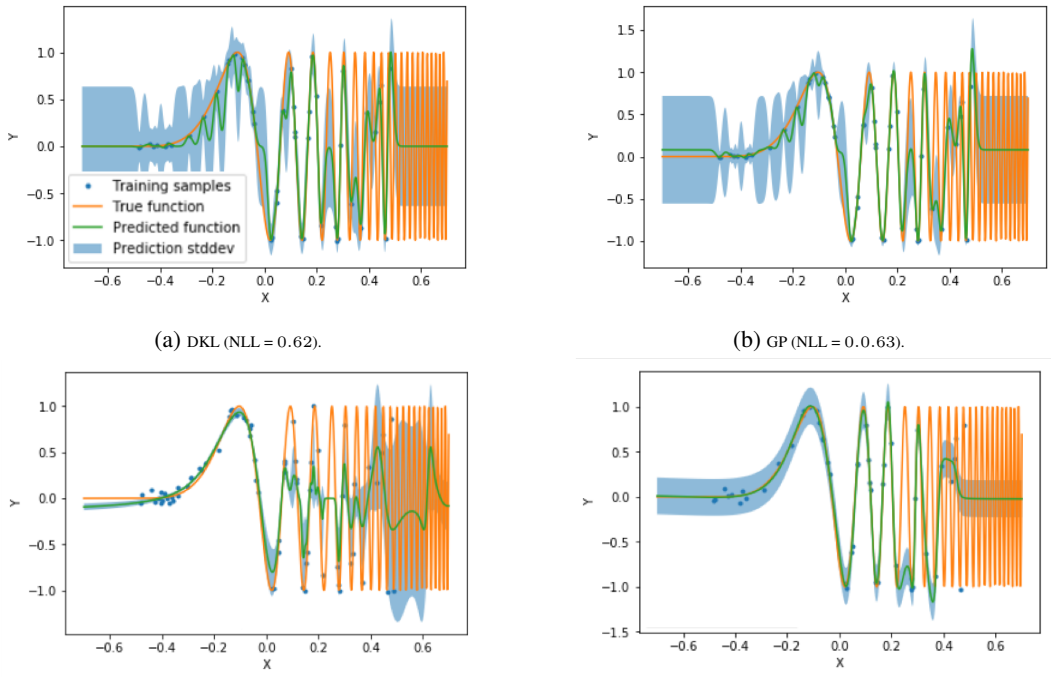
(b) DNN Embeddings produced by FRDKL (proposed method).

Figure 2: Forecasts and embeddings corresponding to a non-smooth function. It can be observed that the embeddings, which the DNN learns, are approximately orthogonal.

3 Simulations

In this section, we will present simulation results for a synthetic sinusoidal dataset, logistic map, and Boston housing dataset. We had performed simulation studies for four algorithms - DKL, Bag of Neural Networks, GP, and FRDKL (proposed method). In the FRDKL, we have used the rank $R = 8$. The DNN architecture, which is used to learn the embeddings, is comprised of 9 hidden layers. The first two are dense layers with tanh and relu activations with $4 * R$ number of units, followed by a dropout layer with dropout probability 0.01. The same layer structure is repeated twice more with number of units $2 * R$ and R respectively. For fair comparison we have used the same network architecture for DKL, by replacing only the output layers activation to rbf. For the bag of NN, we have used the same network architecture with bag size 25. For an arbitrarily complex high dimensional dataset, we would require more complex neural networks. It can be shown theoretically that the kernel of an arbitrary complex high-dimensional dataset could be approximated by a DNN of finite number of layers. The more complex the geometry of the data, potentially deeper the network would be. The details of this sufficiency condition can be found in the form of Theorem 5.1 in the Appendix 5.1. The weight of the orthogonality penalty λ (from subsection 2.2.1) is chosen as 0.3. It can be observed that with increase in the value of λ the embeddings would become more orthogonal with pooer the log-likelihood loss. The first data set is a sinusoidal function [6], whose frequency increases as square of x. Figure 3 contains the plots corresponding to different algorithms for this particular dataset. We have added heteroscedastic noise to the function, where the noise magnitude increases from the left to right. This experiment studies the mixed effect of heteroscedastic noise and nonlinear fluctuation in the dataset. It can be observed that the bag of NN underestimates the confidence intervals near the noisy region. For DKL and GP, we can observe the confidence intervals fluctuates heavily near the noisy region. However, in the FRDKL the confidence bounds are relatively stable and can capture regions of high noise and fluctuations.

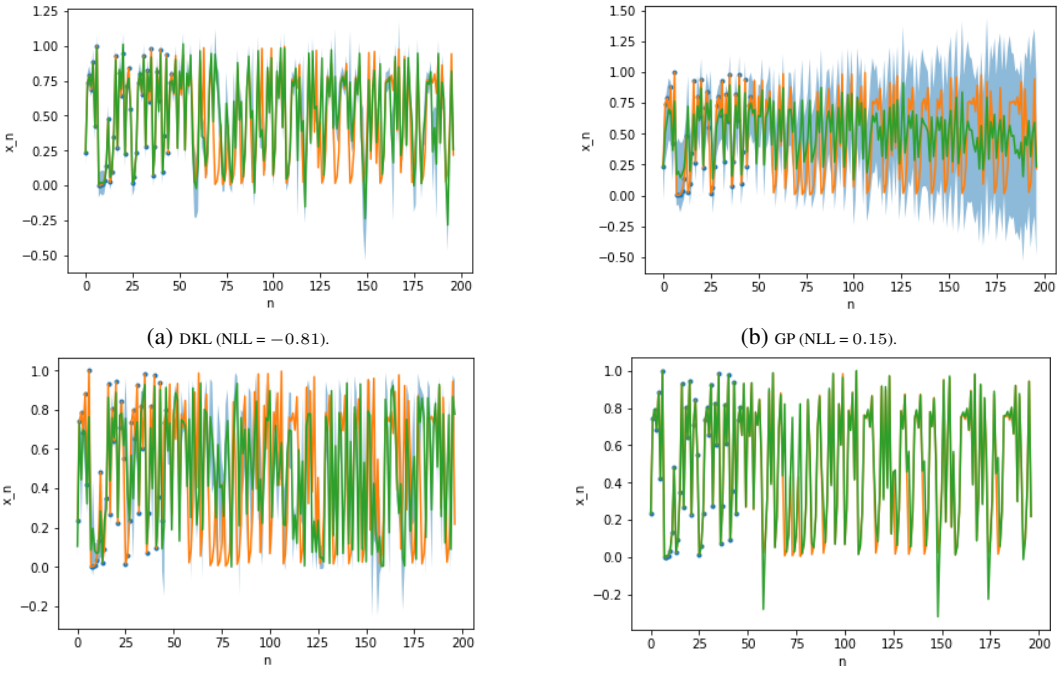
The second example shows the simulation results for Logistic map, which can be found in Fig. 4. We would try to forecast 1-step ahead value of the time series based on past 3 data points. We can show with the aid of dynamical systems theory that 1-step ahead value could be forecast by a smooth function based on the 3 past data points. Logistic map is a chaotic but deterministic dynamical system $x_{n+1} = r \ x_n(1 - x_n)$, where $x_n \in S_1$. We generate the time series data, generated by the system



(a) DKL (NLL = 0.62). (b) GP (NLL = 0.0.63).

(c) NN (NLL = 3.17). (d) FRDKL (NLL = 0.54).

Figure 3: (a-d) Forecasts for sinusoidal data. The uncertainty bounds are $1-\sigma$ limits. FRDKL (proposed method) outperforms competing methods with respect to the negative log likelihood (NLL) although there are singular data points, where it has not succeeded to capture the uncertainty.



(a) DKL (NLL = -0.81). (b) GP (NLL = 0.15).

(c) NN (NLL = 3.31). (d) FRDKL (NLL = -1.18).

Figure 4: (a-d) Mean and $1-\sigma$ uncertainty bounds for logistic map. FRDKL (proposed method) outperforms competing methods in terms NNL.

for $r = 4.1$, which falls in the region of strange attractor [11]. It can be observed that the GP and NN output very wide confidence intervals, and these models erroneously identify chaos as noise. For DKL the bounds are relatively moderate. On the other hand, the proposed FRDKL method correctly estimates the confidence intervals to capture the chaotic time series with very narrow confidence bounds.

We have also performed simulations over Boston housing regression dataset [12, 13, 14] and have compared DKL and FRDKL. In this dataset, based on variables like per capita crime, non-retail business acres per town, nitrogen oxides concentration etc. (13 such features in total), we were to forecast median value of the owner-occupied homes. The dataset has 506 sample data points. We have made a $60 - 40$ split of the dataset into training and test. We have computed the normalized root mean squared error (NRMSE) as a measure of accuracy, which is computed as the RMSE of a predictor divided by the standard error of the samples. The NRMSE < 1 would be the threshold for any predictor performing better than the sample mean. The NRMSE values were found to be 0.41 for DKL and 0.25 for FRDKL. The average CPU time lapsed for one epoch during the model training were 0.32 sec for DKL and 0.079 sec for FRDKL. The inference times were 0.03 sec and 0.01 sec for DKL and FRDKL. In summary, we have demonstrated that FRDKL outperforms competing methods, including DKL, both in terms of accuracy as well as computational efficiency.

4 Conclusion

We have proposed a novel way of representing the kernel operators in DKL as a finite linear combination of simpler dot kernels, which we call as Finite Rank Deep Kernel Learning (FRDKL). We have demonstrated theoretically that our proposed approach would reduce the computational complexity to O(n2), while still enabling us to produce rich kernels that can model complex datasets. We have presented simulation studies over one simulated dataset, logistic map, and the Boston housing dataset, and have found our proposed method to have outperformed competing algorithms. In the future, we will carry out more detailed experimental studies over additional datasets in the UCI repository, and also present a scheme for efficient hyper parameter tuning.

References

[1] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[3] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

[4] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. *arXiv preprint arXiv:1807.00263*, 2018.

[5] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J Rezende, SM Eslami, and Yee Whye Teh. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

[6] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[7] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *arXiv preprint arXiv:1610.08936*, 2016.

[8] Tomoharu Iwata and Zoubin Ghahramani. Improving output uncertainty estimation and generalization in deep learning via neural network gaussian processes. *arXiv preprint arXiv:1707.05922*, 2017.

[9] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pages 63–71. Springer, 2004.

[10] Erwin Kreyszig. *Introductory Functional Analysis with Applications*, volume 1. wiley New York, 1978.

[11] J-P Eckmann and David Ruelle. Ergodic theory of chaos and strange attractors. In *The Theory of Chaotic Attractors*, pages 273–312. Springer, 1985.

[12] David Harrison Jr and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

[13] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*, volume 571. John Wiley & Sons, 2005.

[14] Kaggle. https://www.kaggle.com/c/boston-housing/data.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

5 Appendices

In this section, we would provide some technical details, computations, and the proofs.

5.1 Sufficiency Condition for Deep Neural Networks to Learn Finite Rank Kernels

We would state a sufficiency condition, which describes that a deep neural network of finite depth would be sufficient to learn a Mercer kernel of finite rank to arbitrary precision.

Theorem 5.1 (Deep Neural Network as Universal Approximator of Finite Rank Kernel) *For any Mercer Kernel $K(x,y) = \sum_{r_1=1}^{R_1} \zeta_{r_1} \Theta_{r_1}(x)\Theta_{r_1}(y)$, and an $\epsilon > 0$, there exists an N and a family of neural network regressors with finite number of hidden units and output layer $\phi_{r_2}(z,w)$ s.t.*

$$|K(x,y) - \sum_{r_2=1}^{R_2} \phi_{r_2}(x,w)\phi_{r_2}(y,w)| < \epsilon, \ \ \forall\, R_2 > N, \text{ and } \forall\, x, y \in X,$$

where, $\Theta_{r_1}(x)$, and $\phi_{r_2}(z,w)$ form sets of mutually orthogonal functions. Theorem 5.1 states that an arbitrary smooth Mercer Kernel can be approximated arbitrarily close by a multi-layered neural network. $\Theta_{r_1}(z,w)$s are the output of the neural network, which forms an embedding, which are orthogonal to one another. The proof of the Theorem 5.1 could be attained by extending results from [15], and then extending it for orthogonal eigen-decomposition of the Mercer operators [10].

5.2 Matrix Computations to Reduce Computational Complexity

We would provide the detailed matrix calculations, which would help us reduce the computational complexity of the matrix inversions, utilizing the orthogonality of the embeddings. We have expressed the kernel operator as $K(x,y) = \sum_{r=1}^{R} \phi_r(x,w)\phi_r(y,w)$, where $\phi_r(y,w)$ are orthogonal to each other. For finite number of training data points, we define $\Phi_r = [\phi_r((x_1,w)), \ldots \phi_r((x_n,w))]$, the kernel matrix is defined as $K = \sum_{r=1}^{R} \Phi_r.\Phi_r^T$, which will have the inverse $K^{-1}(x,y) := \sum_{r=1}^{R} \frac{1}{||\Phi_r||^2}.\Phi_r\ \Phi_r^T$, as we can show, using the orthogonal embeddings the following,

$$K(x,y)\ .\ K^{-1}(x,y) = \left(\sum_{r=1}^{R} \Phi_r.\Phi_r^T\right).\left(\sum_{r=1}^{R} \frac{1}{||\Phi_r||^2}.\Phi_r\ \Phi_r^T\right) = I.$$

Next, we would derive the expression for $\left(K + \sigma^2 I\right)^{-1}$. We could express identity matrix as

$$I = \left(\sum_{i=1}^{R} \frac{1}{||\Phi_i||^2}.\Phi_i\ \Phi_i^T + \sum_{j=1}^{n-R} \frac{1}{||\eta_j||^2}.\eta_j\ \eta_j^T\right),$$

where η_j's are orthogonal to each other and form the null space of the subspace spanned by Φ_i's. This construction leads us to the following equation,

$$\left(K + \sigma^2 I\right) = \sum_{i=1}^{R} \frac{\sigma^2}{||\Phi_i||^2}.\Phi_i\ \Phi_i^T + \sigma^2 \sum_{j=1}^{n-R} \frac{1}{||\eta_j||^2}.\eta_j\ \eta_j^T.$$

$$\begin{aligned} K_* \left(K + \sigma^2 I\right)^{-1} &= \sum_{i=1}^{R} \frac{\sigma^2}{||\Phi_i||^2}.\Phi_i\ \Phi_i^T + \sum_{j=1}^{n-R} \frac{\sigma^2}{||\eta_j||^2}.\eta_j\ \eta_j^T, \\ &= \left(\sum_{r=1}^{R} \Phi_r^*.(\Phi_r^*)^T\right).\left(\sum_{r=1}^{R} \frac{\sigma^2}{||\Phi_r||^2}.\Phi_r\ \Phi_r^T\right). \end{aligned}$$

It can be observed that the complexity of the $K_* \left(K + \sigma^2 I\right)^{-1}$ computation would be $\mathcal{O}(n^2)$.

5.3 Decomposition of the Loss Suitable for Stochastic Gradient Descent

Next, we would address the question, whether or not we would be able to decompose the loss function over the training data points, so that the stochastic gradient descent could be used. We have observed that the loss function is comprised of the log-likehood and a penalty function for orthogonal embeddings. The penlaty function is a function of an individual datapoint, and thus we would have to prove the decomposition of the log-likelihood. We would utilize the orthogonality of the embeddings once more to reach the decomposition.

$$\begin{aligned} -log\ p(y|x) &\sim y^T (K_\gamma + \sigma^2 I)^{-1} y + log|K_\gamma + \sigma^2 I|, \\ &= y^T \left(\sum_{r=1}^{R} \Phi_r\ \Phi_r^T + \sigma^2 I\right)^{-1} y + log|\sum_{r=1}^{R} \Phi_r\ \Phi_r^T + \sigma^2 I|, \\ &= y^T \left(\prod_{r=1}^{R} \left(\frac{1}{\sigma^2}\Phi_r\ \Phi_r^T + \sigma^2 I\right)\right)^{-1} y + log|\prod_{r=1}^{R} \left(\frac{1}{\sigma^2}\Phi_r\ \Phi_r^T + \sigma^2 I\right)|, \\ &= y^T \left(\prod_{r=1}^{R} \left(\frac{1}{\sigma^2}\Phi_r\ \Phi_r^T + \sigma^2 I\right)^{-1}\right) y + log|\prod_{r=1}^{R} \left(\frac{1}{\sigma^2}\Phi_r\ \Phi_r^T + \sigma^2 I\right)|. \end{aligned}$$

which leads to the decomposition of the loss function into training data points.

5.4 Deep Orthogonal Bayesian Linear Regression

One can draw parallels between the GP and Bayesian Linear Regression (BLR) [9]. BLR can be shown as equivalent to GP with dot kernel. We can extend the developed technique to BLR, where the regressor becomes a linear combination of orthogonal embeddings, which would be learned by a DNN.

$$y \sim \sum_{r=1}^{R} \phi_r(x) w_r + \epsilon,$$

where, ϵ is the observation noise with variance σ^2, and R is the rank of the combined embedding. It can be shown,

$$f_*|x_*, X, y \sim \mathcal{N}\left(\frac{1}{\sigma^2}\sum_{r=1}^{R} \phi_r(x_*) A^{-1} \Phi_r y,\ \phi_r(x_*) A^{-1} \phi(x_*)\right),$$

where $\Phi_r = \Phi_r(X)$, and $A = \sum_{r=1}^{R} \frac{1}{\sigma^2}\Phi_r \Phi_r^T + \Sigma_p^{-1}$. It can be further modified to,

$$\begin{aligned} f_*|x_*, X, y \sim \mathcal{N}\Bigg(&\sum_{r=1}^{R} \phi_r(x_*)\Sigma_p \Phi_r (K + \sigma^2 I)^{-1} y, \\ &\sum_{r=1}^{R} \left(\ \phi_r(x_*)\Sigma_p \phi_r(x_*) - \phi_r(x_*)\Sigma_p \Phi_r (K + \sigma^2 I)^{-1} \Phi_r^T \Sigma_p \phi_r(x_*)\ \right)\Bigg), \end{aligned} \quad (1)$$

where, $K = \sum_{r=1}^{R} \Phi_r \Sigma_p \Phi_r$. In (1), using the orthogonality of the embeddings, we could reduce the complexity of $(K + \sigma^2 I)^{-1}$. We can formulate a global optimization problem, which would optimize the log likelihood of the BLR with orthogonal embeddings, learned by deep neural network. We could incorporate the orthogonality constraint in the loss function similar to the way it was done for FRDKL.