RANDOMIZED SKETCHES FOR DEEP KERNEL LEARNING

Yijun Yuan

ShanghaiTech University, China.

ABSTRACT

In this paper, we analyze Deep Kernel Learning (DKL) method on a common non-parametric regression model, Kernel Ridge Regression (KRR). According to Randomized Sketches approximation technique, we propose a Sketched DKL (SDKL) method to modified our model allowing for cheap stochastically training. With a small number of sketches, the numerical experiments illustrate the proposed model outperforms than Deep neural network (DNN) but with same computational complexity.

Index Terms— Deep kernel learning, Kernel regression, Randomized sketches

1. INTRODUCTION

Nonparametric regression problem arises in machine learning a couple of decades. The goal of nonparametric regression for a given data set $\{(x_i, y_i)\}_{i=1}^N$ is to estimate the regression function $f(x) = \mathbb{E}[Y|X = x]$ which makes predictions of a response variable $Y = \{y_1, .., y_N\} \in \mathbb{R}$ based on observing a covariate vector $X = \{x_1, .., x_N\} \in \Omega$. If we consider the uncertainties in a standard Gaussian model, the covariate-response pairs are related to the model

$$y_i = f(x_i) + w_i \text{ , for all } i = 1, .., N.$$

Here, Ω denotes the open domain of f with $X \in \Omega$, sequence $\{w_i\}_{i=1}^n$ consists of i.i.d. standard Gaussian variates.

The traditional method to estimate function f is to use kernel ridge regression(KRR). KRR enforces f to belong to reproducing kernel Hilbert space (RKHS) such that f can have some regularity properties. To avoid the $\mathcal{O}(n^3)$ computation cost, many approximation methods has been proposed without losing such statistical properties. The most popular is Nyström method [1] to reduce the computational complexity to $\mathcal{O}(m^2 n)$ with $m \ll n$. After that, [2], [3] introduce an improvement on sampling for Nyström method. To make computation even cheaper, [4] further replace the sampling with randomized sketches on kernel ridge regression (KRR) problem. With choosing a proper sketch matrix, the time complexity and space complexity of KRR can scale as $\mathcal{O}(m^3)$ and $\mathcal{O}(m^2)$ from $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$ with $m \ll n$ on the final m-dimensional quadratic program. However, there is no free lunch. A preprocessing step for computing the approximate kernel takes $\mathcal{O}(n^2 \log m)$ for suitable chosen projection.

While KRR has a rigorous theoretical analyze, in practice, the performance of kernel method is dependent of how the features of data feeding in. So the extraction of feature do make sense in the filed of kernel method. Recent years, deep learning (DL) has achieved a great success on data mining. Motivated by the flexibility of deep neural networks, in which the feature extraction in the data set can be done completely automatically, deep kernel learning method firstly proposed in [5] utilizes deep neural network for feature extraction while combing kernel method for regression.

The works related to deep kernel learning starts with concerning the model based on Gaussian process. [6] introduced an arc-cosine kernel function that mimic the computation in large, multi-layer neural networks and shows how to use the multilayer kernel machine to benefit from deep neural networks. Under the framework of Gaussian process to learn hidden presentation in data, [7] utilize Gaussian process in Baysian neural network structure to construct its kernel. [5] pre-trained the unsupervised deep autoencoders to achieved the feature mapping function, then together with the kernel parameter, minimizing the negative likelihood under Gaussian process to train the model as a whole. Closely, [8] attempts to derive a scalable closed form kernel that can train model jointly under Gaussian process and it is then, the word, Deep kernel learning (DKL) comes into the field. The structure of DKL is depicted in Fig. 1. Its extension, [9] move one step further to enable classification, multi-task learning and stochastic gradient training. While more specifically, current DKL is on top of Gaussian process (GP) because the maximization of a posterior of GP make it reasonable to train the model as a whole.

Multi-layer Multiple kernel learning (MLMKL) is another variant of kernel learning method. In [10], the proposed method moves the first step from multiple kernel learning (MKL), exploring the the multi-layers structure for the combination of multiple kernels. However, it is only for two layers network. Instead of optimizing based on dual objective function, [11] proposes to optimize multi-layers of kernel with leave-one-out error. [12] discussed the inefficiency of existed MLMKL, and introduced to optimize the model with gradient ascent. Recently, [13] presents that, similar to MLMKL, deep kernel learning can be considered as a concatenation of a kernel function with one or more non-linear function. The goal of such structure is to construct new kernel function which is flexible and with high expressibility. Thus, the deep kernel learning would be able to release itself from the structure of Gaussian process.

In this paper, we focus on the standard kernel ridge regression model (KRR). Section 2 reviews the background of DNN and DKL. The main contribution of this paper is presented in Section 3. With a deep architecture to approximate the transformation function for the input of kernel matrix, the proposed method learns the deep feature while solving the regression. Here, we use the Randomized Sketches approach, which was firstly introduced in [4] to reduce the time and space complexity. The performance is illustrated with experiments in Section 4. Section 5 concludes the paper.

2. BACKGROUND

2.1. DNN

Deep neural network aims to find a mapping $y = f(x|\theta)$ from feature to a value, which is equivalent of finding the best θ to approximate the real function f^* [14]. In general, DNN can be represented as a series of concatenated functions

$$f := f_n \circ \cdots \circ f_2 \circ f_1$$

where f_1 and f_n is the first layer and last layer. The resultant optimization problem can be written as

$$\min_{f_1,\cdots,f_n} \frac{1}{N}\sum_{i=1}^{N} \|y_i - f_n \circ \cdots \circ f_2 \circ f_1(x_i)\|_2^2 + \lambda \sum_{j=1}^{n} \|f_j\|^2 ,$$

where $\{x_i, y_i\}_{i\in\{1,\cdots,N\}}$ is the training pairs, λ is the regularization multiplier.

In this paper, several commonly used layers will be involved: 2D-convolution layer (Conv2), ReLU activation (ReLU), Max-pooling layer (MaxPool), and fully-connected layer (fc). Conv2 has been widely used in image tasks for its good property on sparse interactions, parameter sharing and equivariant representations [14]. ReLU, the non-linear function make model easier to adopt to the variety of data. Generally, MaxPool appears in network when Conv2 is utilized. From [14], pooling make the representation approximately invariant to small translations of inputs. Accepted the feature from Conv2, fc can learn a map to obtain the vector feature and provide targets to allows for end-to-end train.

2.2. Deep Kernel Learning

By utilizing the advantage of deep neural network in feature extraction, A deep kernel learning method can be illustrate by reformulating the regression function f as a concatenation of functions, i.e.,

$$f := g \circ h_L \circ \cdots \circ h_1 ,$$

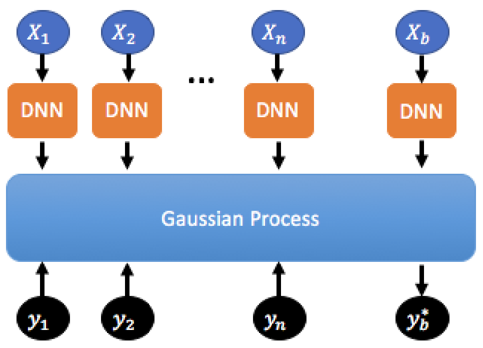

Fig. 1: Structure of DKL. In this figure, $\{(X_i, y_i)\}_{i\in 1,\cdots,n}$ are training set. X_b is the testing sample and y_b^* is its prediction.

functions h_i for all $i = 1, ..., L$ represents the nonlinear map at i-th layer. The optimization problem can be rewritten as

$$\min_{f\in\mathcal{H}_0, h_i\in\mathcal{H}_i} \sum_{i=1}^{N} \|y_i - g \circ h_L \circ \cdots \circ h_1(x_i)\|_2^2 + \lambda_{L+1}\|g\|^2_{\mathcal{H}_{L+1}} + \sum_{i=1}^{L} \lambda_i \|h_i\|^2_{\mathcal{H}_i} \qquad (1)$$

where Reproduced Kernel Hilbert Space (RKHS) $\mathcal{H}_i := H(\Omega_i, \Phi_i)$ for all $i = 1, ..., L$ as well as $\mathcal{H}_{L+1} := H(\Omega_{L+1}, \mathbb{R})$ are denoted by functions $h_i : \Omega_i \to \Phi_i$ and $g : \Phi_L \to \mathbb{R}$. Here, we assume $\Omega_{i+1} \subseteq \Phi_i \subseteq \mathbb{R}^{d_i}$ for all $i = 1, .., L$. Moreover, it is assumed that functions h_i are enforced to be in a vector valued extended RKHS. The most recent work ([13]) analyzes the solution of (1) by utilizing representer theorem from theoretical point of view. For space limitation, we summarize the conclusion proposed in [13] by the following theorem.

Theorem 1 *The solution of* (1) *satisfies* $h_i \in \mathcal{V}_i \subseteq \mathcal{H}_i$ *for all* $i = 1, .., L$ *as well as* $g \in \mathcal{V}_{L+1} \subseteq \mathcal{H}_{L+1}$ *with*

$$\mathcal{V}_i = span\left\{ \mathcal{K}_i(h_{i-1} \circ \cdots h_1(x_i), \cdot)e_k \,\middle|\, \begin{array}{lcl} i & = & 1, ..., N \\ k & = & 1, .., d_i \end{array} \right\},$$

where $\mathcal{K}_i$ *denotes the reproducing kernel of* $\mathcal{H}_i$ *and* $e_k \in \mathbb{R}^{d_i}$ *is the* k*-th unit vector.*

As a consequence, the infinite dimension optimization problem (1) can be transfered as a finite dimension problem, more details refers to [13].

Remark 1 *From the theoretical point of view, [13] gives a systematic analysis of the solution of* (1). *However, the initial formulation of neural network with* h_i *enforced to belong to RKHS limits the structure of deep neural network into a kind of specific form. In practice, with the number of layers* L *increasing, the yielded nonconvex optimization problem is* *scaled up. Different form the standard deep learning optimization problem with a summation form, it can not be solved efficiently with stochastic gradient descent in high dimension due to the dense form. In the next, we will propose a more general form of DKL which is capable of formulating any deep neural networks and being solved by SGD efficiently.*

3. METHOD

This section concerns the deep neural network with a general feature map $h(\cdot; w) : \mathbb{R}^d \to \mathbb{R}^f$ defined by structure parameter $w \in \mathbb{R}^p$.

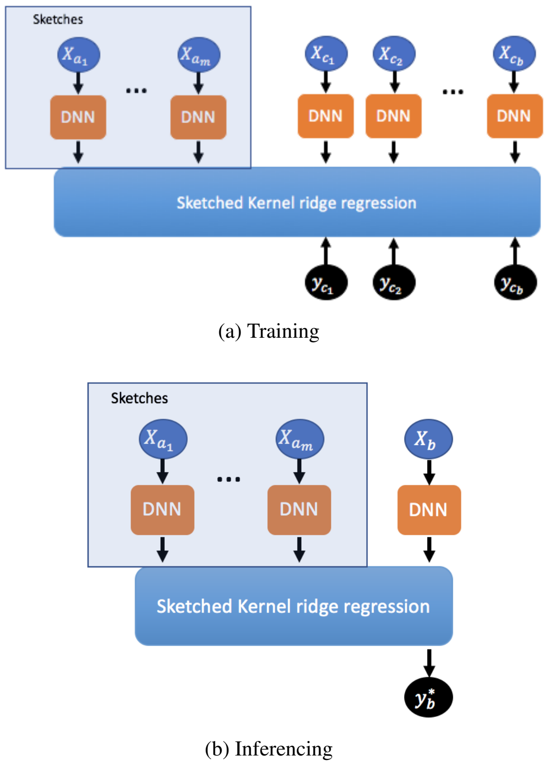

Fig. 2: Structure of SDKL. To distinguish the class, we use $\{a_1, \cdots, a_m\}$, $\{c_1, \cdots, c_b\}$ to indicate the index of sketches set and training batch set.

First, we define the optimization problem to solve as

$$f^*_{X,Y} = \arg\min_{g \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} \|y_i - g(x_i)\|_2^2 + \lambda \|g\|_{\mathcal{H}}^2 \qquad (2)$$

with given RKHS $\mathcal{H} = H(\Omega, \mathbb{R})$ and regularization parameter $\lambda > 0$. The solution of (2) can be directly written down by

$$f^*_{X,Y}(x) = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \alpha_i \mathcal{K}(x_i, x) ,$$

where function $\mathcal{K} : \Omega \times \Omega \to \mathbb{R}$ denotes the reproducing kernel of $\mathcal{H}$, $\alpha = [\alpha_1 \ldots \alpha_N]^\top$ is the solution of the following convex quadratic programming

$$\min_{\alpha} \quad \alpha^\top (K^2 + \lambda K)\alpha - 2\alpha^\top \frac{K}{\sqrt{N}} y ,$$

where matrix $K \in \mathbb{R}^{N \times N}$ is given by entries $K_{ij} = N^{-1}\mathcal{K}(x_i, x_j)$ following [4].

In order to train the deep nerural network and estimate the regression function at $L+1$ layer, we propose to directly plug $h(\cdot; w)$ into problem (2) yielding

$$\min_{g \in \mathcal{H}, w \in \mathbb{R}^p} \frac{1}{N} \sum_{i=1}^{N} \|y_i - g(h(x_i; w))\|_2^2 + \lambda \|g\|_{\mathcal{H}}^2 + \mu \|w\|^2 . \qquad (3)$$

Here, training the neural network by optimizing w is in a sense of optimizing functions h_i, which is consistent with traditional deep kernel learning method. But it has to comment that directly optimizing w is much more straightforward in practice even if there are no analytical solution as discussed in [13]. In analogy with the regularization term of h in (1), we use a standard least square regularization for w with a positive number $\mu > 0$.

Due to the infinite dimension decision varibale f, Problem (3) is still computational intractable. Based on the solution map of (2), we could transfer (3) into a finite dimension optimization problem

$$\min_{\alpha, w} \; \alpha^\top (K(w)^2 + \lambda K(w))\alpha - 2\alpha^\top \frac{K(w)}{\sqrt{N}} y + \mu \|w\|^2, \quad (4)$$

where matrix function $K(w)$ is denoted by entries $K_{ij}(w) = N^{-1}\mathcal{K}(g(x_i; w), g(x_j; w))$.

Note that the dimension of α and w is quite high and slow down the numerical optimization algorithms in practice. Therefore, we consider use sketch matrix S with $m \times n$ to reformulate $\alpha = S^\top \hat{\alpha}$ as randomized sketches method for KRR problem discussed in [4]. The follow-up reformulation of Problem (4) can be written as

$$\begin{aligned} \min_{\hat{\alpha}, w} \quad & \hat{\alpha}^T S(K(w)^2 + \lambda K(w)) S^\top \hat{\alpha} \\ & - 2\hat{\alpha}^\top \frac{SK(w)}{\sqrt{N}} y + \mu \|w\|^2 . \end{aligned} \qquad (5)$$

Problem (5) with a dense quadratic form is not clear what's the advantage in practice. In order to illustrate our idea, a special sketches matrix, sub-sampling sketches, is required. As introduced in [4], $S \in \mathcal{R}^{m \times n}$ holds rows of form $s_i = \sqrt{\frac{n}{m}} p_i$ with vectors $\{p_1, \cdots, p_m\}$, which are chosen randomly and uniformly without replacement from n-dimensional identity matrix. Here, we assume those randomly chosen vectors $\{p_1, \cdots, p_m\}$ are with row indexes $\{a_1, \cdots, a_m\}$, then (5) can be further reduced into

$$\begin{aligned} \min_{\hat{\alpha}, w} \quad & \frac{1}{N} \sum_{i=1}^{N} \|y_i - \frac{1}{\sqrt{N}} \sum_{j=1}^{m} (S^T \hat{\alpha})_{a_j} \mathcal{K}_{i, a_j}(w)\|^2 \\ & + \lambda \hat{\alpha}^T S K S^T \hat{\alpha} + \mu \|w\|^2 . \end{aligned} \qquad (6)$$

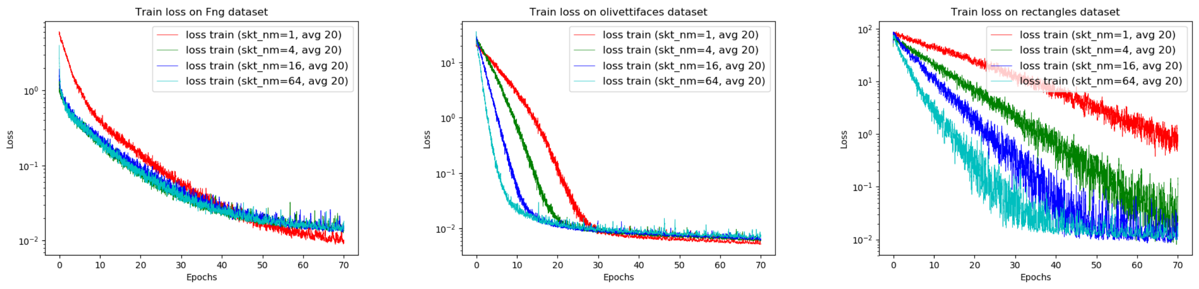

Fig. 3: Loss curve during training for SDKL with semi-log plot. From left to right are on Fng, $Olivettifaces$ and $Rectangle$ dataset. In each figure, four curves demonstrated are with a series of sketches number.

In our implementation, stochastically training only need to pass $1 + m$ inputs through DNN feature function at each iteration with $m \ll N$. Actually, DNN model requires b passes each time by using mini-batch with size b, while $b+m$ passes are adequate for our SDKL. With m that is as small as b, computation cost of our SDKL is on the same level as the DNN model from which its features are obtained.

4. EXPERIMENT

The implementation of the experiment is with python using tensorflow framework [15]. We have utilized the preprocessed dataset from [5]. They are image datasets FGnet, Olivetti Faces and Rectangles with input dimension 1024 and target range $[\log 2, \log 70]$, $[-90, 90]$ and $[2, 31]$ respectively. For olivettiface dataset, considering its target range is relatively large, we rescaled the label to learn into $[1, 10]$ by adding 100 and being divided by 19. Before computing NMSE, we transform it back. During the model training, We randomly select the sketches. In our experiment, sometimes loss will first stick for several epochs and then drop which makes it possible that model are not well trained after a fixed epochs. So to avoid such a problem, if loss does not drop in first 5 epochs, experiment will be reset. All of those models will train 70 epochs with learning rate $1e - 4$ using Adam optimizer [16].

In our implementation, SDKL directly utilized the features before the last fully-connect layer of one simple DNN. And the simple DNN is structured as Conv2-ReLU-MaxPool-Conv2-ReLU-MaxPool-fc-ReLU-fc-ReLU-fc. In KRR, Radial basis function kernel (RBF) are selected to use. Additionally, a series of sketches number m, from small to large, are chosen for SDKL. NMSE for simple DNN and SDKL model with different s has been shown in table 1.

For dataset Fng and $Olivettifaces$, we can find SDKL works better with all of the sketches numbers from small to large. When $s = 16$, 64, our SDKL works better on $Rectangle$. The loss curves during training for SDKL are demonstrated in Fig. 3 with semi-log plot. We can find loss of the cases with better NMSE are converged to the same level. The minimal difference on the tail is caused by m, the larger m will have larger regularity. For $m = 1$, 4 in $Rectangle$ dataset, its NMSE is not as good, right figure in Fig. 3 shows it trained too slow to converge before we stop. From experiment, we can have two observation. First, SDKL can work better than the simple DNN it based on. Then, the number of sketches has effect on the speed of lose dropping while holds the similar performance.

Table 1: NMSE for simple DNN and SDKL by training model 70 epochs.

		Fng	Olivettifaces	Rectangle
DNN		0.04719	0.00711	0.00086
SDKL	m=1	0.03414	0.00499	0.04009
	m=4	0.03021	0.00531	0.00141
	m=16	0.03338	0.00564	0.00035
	m=64	0.03140	0.00579	0.00037

5. CONCLUSION

In summary, we modified the model of DKL with KRR and further adapt randomized sketches approximation on it. From experiment, the introduced SDKL model have a better performance than the simple DNN where its feature is from. For the computation cost, our SDKL can be on the same level as the DNN model which allows for cheaply training and inferencing.

6. REFERENCES

[1] Christopher KI Williams and Matthias Seeger, "Using the nyström method to speed up kernel machines," in *Advances in neural information processing systems*, 2001, pp. 682–688.

[2] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar, "Sampling methods for the nyström method," *Journal of Machine Learning Research*, vol. 13, no. Apr, pp. 981–1006, 2012.

[3] Alex Gittens and Michael W Mahoney, "Revisiting the nyström method for improved large-scale machine learning," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3977–4041, 2016.

[4] Yun Yang, Mert Pilanci, and Martin J Wainwright, "Randomized sketches for kernels: Fast and optimal non-parametric regression," *arXiv preprint arXiv:1501.06195*, 2015.

[5] Wen-bing Huang, Deli Zhao, Fuchun Sun, Huaping Liu, and Edward Y Chang, "Scalable gaussian process regression using deep neural networks.," in *IJCAI*, 2015, pp. 3576–3582.

[6] Youngmin Cho and Lawrence K Saul, "Kernel methods for deep learning," in *Advances in neural information processing systems*, 2009, pp. 342–350.

[7] Andreas Damianou and Neil Lawrence, "Deep gaussian processes," in *Artificial Intelligence and Statistics*, 2013, pp. 207–215.

[8] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing, "Deep kernel learning," in *Artificial Intelligence and Statistics*, 2016, pp. 370–378.

[9] Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing, "Stochastic variational deep kernel learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2586–2594.

[10] Jinfeng Zhuang, Ivor W Tsang, and Steven CH Hoi, "Two-layer multiple kernel learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 909–917.

[11] Eric V Strobl and Shyam Visweswaran, "Deep multiple kernel learning," in *Machine Learning and Applications (ICMLA), 2013 12th International Conference on*. IEEE, 2013, vol. 1, pp. 414–417.

[12] Ilyes Rebai, Yassine BenAyed, and Walid Mahdi, "Deep multilayer multiple kernel learning," *Neural Computing and Applications*, vol. 27, no. 8, pp. 2305–2314, 2016.

[13] Bastian Bohn, Michael Griebel, and Christian Rieger, "A representer theorem for deep kernel learning," *arXiv preprint arXiv:1709.10441*, 2017.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436, 2015.

[15] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., "Tensorflow: a system for large-scale machine learning.," in *OSDI*, 2016, vol. 16, pp. 265–283.

[16] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.