Hashing Based Prediction for Large-Scale Kernel Machine

Lijing Lu[1,2][0000−0002−9039−0385], Rong Yin[1,2][0000−0003−1894−7561], Yong Liu[1⋆], and Weiping Wang[1]

[1] Institute of Information Engineering, Chinese Academy of Sciences
[2] School of Cyber Security, University of Chinese Academy of Sciences
{lulijing, yinrong, liuyong, wangweiping}@iie.ac.cn

Abstract. Kernel Machines, such as Kernel Ridge Regression, provide an effective way to construct non-linear, nonparametric models by projecting data into high-dimensional space and play an important role in machine learning. However, when dealing with large-scale problems, high computational cost in the prediction stage limits their use in real-world applications. In this paper, we propose hashing based prediction, a fast kernel prediction algorithm leveraging hash technique. The algorithm samples a small subset from the input dataset through the locality-sensitive hashing method and computes prediction value approximately using the subset. Hashing based prediction has the minimum time complexity compared to the state-of-art kernel machine prediction approaches. We further present a theoretical analysis of the proposed algorithm showing that it can keep comparable accuracy. Experiment results on most commonly used large-scale datasets, even with million-level data points, show that the proposed algorithm outperforms the state-of-art kernel prediction methods in time cost while maintaining satisfactory accuracy.

Keywords: Kernel machine · Locality-sensitive hashing · Kernel prediction.

1 Introduction

Kernel methods have been widely implemented in practice, allowing one to discover non-linear structure by mapping input data points into a feature space, where all pairwise inner products can be computed via a nonlinear kernel function. Kernel machines, such as Kernel Ridge Regression (KRR), have attracted a lot of attention as they can effectively approximate any function or decision boundary with enough training data [17][18][19].

Despite excellent theoretical properties, they have been limited applications in large scale learning because computing the decision function for new test samples is extremely expensive. As the scale of data increases, not only the training time will become longer, but the prediction time will also increase. However,

⋆ Yong Liu is the corresponding author.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

more and more improvement methods devoted to reducing training complexity have been proposed, while fewer algorithms have been designed to improve the performance of the prediction stage [12]. As is known to us all, the Nyström and random feature methods are devoted to reaching faster training and prediction speed by constructing low-rank approximation of kernel matrix. Nyström method [4][14][16] construct a small scale subset of landmark data points by sampling to approximate the raw kernel matrix. Random feature [17][20] maps data into a relative low-dimensional randomized feature space to improve both training and prediction speed. Random sketch [15][23] is another family of techniques that projects the kernel matrix into a small matrix to reduce the computational requirement. Although these methods perform well and are applied into practice widely, they still need huge computational requirements in the prediction stage when faced with large-scale datasets. Based on the innovation of further reducing the computational costs when dealing with large-scale datasets, we attempt to develop a fast prediction algorithm for kernel methods.

Hashing is an efficient algorithm to solve the approximate large-scale nearest Neighbor search problem. The main idea of the hashing method is to construct a family of hash functions to map the data points into a binary feature vector such that the produced hash code preserves the structure of the original space [9]. In recent years, a large number of effective hashing methods have emerged, but they are rarely used in the prediction of kernel machines. Charikar and Siminelakis presented a hashing-based framework for kernel density estimate problem in [5]. Inspired by this paper, we consider applying the hashing algorithm to the prediction stage of kernel machine and proposed the hashing based prediction (HBP) algorithm. The algorithm leverages locality-sensitive hashing (LSH) [6][8] method to search the nearest neighbors of the test data point to approximately compute the decision value in the kernel prediction problem.

Specifically, our HBP algorithm consists of two stages: the pre-processing stage and the query stage. LSH was used to find the neighbors of the test point as the sampled subset in the pre-processing stage. And in the query stage, the samples are used to approximately compute the decision value. We provide a theoretical analysis that we can compute the decision function in $\mathcal{O}(\frac{log(n/\epsilon\tau)}{\tau^{0.5}\epsilon^{2.5}})$ time cost with accuracy guarantees $\|\hat{y} - y^*\|_p \leq \epsilon \cdot (3\tau n^{1/p} + \|y^*\|_p)$, where $\epsilon, \tau \in (0, 1)$. It is novel to use the hash method directly in the KRR problem which is an attempt to break through the traditional method. As will be demonstrated later, experiment results on most commonly used large-scale datasets, even with million-level data points, show that the proposed HBP algorithm outperforms the state-of-art kernel prediction methods in time cost while maintaining satisfactory accuracy.

The remainder of the paper is organized as below: We start with presenting related work in Section 2, and the background material is given in Section 3. In Section 4, we introduce our hashing-based prediction algorithm for the prediction of kernel machine, while in Section 5, we discuss the theoretical analysis. Section 6 presents the experimental results on real-world datasets. The following section is conclusions and the last section presents proof of theoretical results.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

2 Related Work

In this section, we will introduce several important works on reducing time complexity in the prediction stage of KRR, and most of the algorithms can be used to other kernel machines [11]. Practical methods are proposed to overcome the computational and memory bottleneck of kernel machines. One popular family of techniques is based on low-rank approximation of the kernel matrix to improve both training and prediction speed. Nyström [13][14][22] method is one of the most well-known representation methods. Nyström samples a subset C of m columns to approximate the kernel matrix $G \in \mathbb{R}^{n\times n}$. Typically, the subset of columns is randomly selected by uniform sampling without replacement [13][22]. Recently, more and more outstanding extension methods of Nyström are proposed to solve the KRR problem. For example, an accurate and scalable Nyström scheme has been proposed in [14] which first samples a large column subset from the input matrix, but then only performs an approximate SVD on the inner submatrix by using the recent randomized low-rank matrix approximation algorithms. [16] present a new Nyström algorithm based on recursive leverage score sampling. Based on Nyström method, a fast prediction method called DC-Pred++ was proposed in [11]. However, the scale of the subset in Nyström method can't be too small to keep the accuracy, and as a result, the lower bound of computational complexity is limited. Random features [17][20] project the data into a relative low-dimensional feature space where the inner product between a pair of input points approximates their kernel evaluation so as to reduce the computation requirement of training and prediction stage. Another family of techniques is random sketches which improving the computational speed by projecting the kernel matrix into a small matrix [15][23]. As shown in [23], a simple hash method was applied to generate the randomized sketch matrix.

Hashing algorithms [6][8][9] have been a continuously "hot topic" since it birth because of its superior performance in the Approximate Nearest Neighbor Search. However, hashing algorithms have not widely been used in kernel prediction problems. Inspired by the idea of "sample" of the Nyström method, we consider employing Locality-sensitive hashing (LSH) to sample the subset and approximately compute the result using the subset. Compared with the currently optimal and most popular Nyström methods, our HBP method can further greatly reduce the number of samples while ensuring accuracy. HBP reduces computational efficiency from $\mathcal{O}(n)$ to $\mathcal{O}(\frac{log(n/\epsilon\tau)}{\tau^{0.5}\epsilon^{2.5}})$ with less accuracy loss $\|\hat{y} - y^*\|_p \leq \epsilon \cdot (3\tau n^{1/p} + \|y^*\|_p)$.

3 Preliminaries

This paper focuses on the typical kernel machines: KRR. [7][10][21]. Given dataset $\{x_i, y_i\}_{i=1}^n, x_i \in \mathbb{R}^d$, $k(x_i, x_j)$ denotes the kernel function value of the two points x_i, x_j, $\hat{\boldsymbol{\alpha}} \in \mathbb{R}^n$ in the formula is generated in the training process by

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

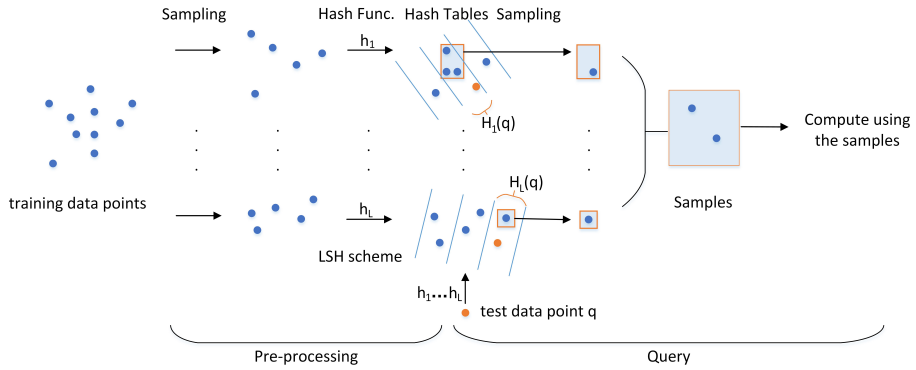

Fig. 1. Schematic diagram of algorithm structure. Given the training data points and a test point q, the HBP algorithm samples some hash functions and construct a separate hash table for each hash function leveraging LSH in the pre-processing stage. In the query stage, for each hash table, we sample a point randomly from the hash bucket that the test point maps to.

solving the below optimization problem:

$$\hat{\boldsymbol{\alpha}} \leftarrow \underset{\boldsymbol{\alpha}}{argmin}\, \boldsymbol{\alpha}^T K \boldsymbol{\alpha} + \lambda \boldsymbol{\alpha}^T \boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T y, \tag{1}$$

where $K \in \mathbb{R}^{n \times n}$ is the kernel matrix with $K_{ij} = k(x_i, x_j)$, $y = [y_1, ..., y_n]^T \in \mathbb{R}^n$ is the response vector, $\lambda > 0$ is the regularization parameter.

Our goal in the prediction process is to compute the decision value of a testing data $\bar{x}$, following the equation:

$$y^* = \sum_{i=1}^{n} \hat{\boldsymbol{\alpha}}_i k\left(x_i, \bar{x}\right), \tag{2}$$

whose time cost is $\mathcal{O}(n)$. The problem to solve Eq.(2) is challenging as n increases. To improve the speed of kernel prediction, we proposed the following HBP algorithm.

4 Hashing Based Prediction

The proposed HBP method is an importance sampling algorithm. The algorithm solves the problem of designing a model that given a set of data points $X = \{x_i, y_i\}_{i=1}^n \subset \mathbb{R}^d$ and a kernel function k, returns the approximation of the decision value of a test point $\bar{x}$: $\sum_{i=1}^n \hat{\boldsymbol{\alpha}}_i k\left(x_i, \bar{x}\right)$. Given a hashing based data structure, HBP performs a two-stage computation to approximate the prediction result. The architecture of HBP is illustrated in Fig.1. In the pre-processing stage, we select data points which close to the test data point as the sampling points. In the query stage, we use the sampling points to estimate the response

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

value. As shown in Fig.1, the hashing technique plays an important role in the proposed method. The hashing technique we leverage is called Localization-sensitive-hashing (LSH). Before explaining the algorithm in detail, we will first introduce the LSH method.

4.1 Localization-sensitive hashing

Localization-sensitive hashing (LSH) is an approximate neighbor search technology. The key idea of LSH is that two adjacent data points in the original data space are mapped by the same projection, the probability that these two data points are still adjacent in new data space is high. For example, the LSH method employed by this paper map a d dimension vector $\boldsymbol{x}$ into a ρ dimension vector consisting entirely of 0 and 1, where $\rho < d$. The vector of ρ dimension is the hash value. If two data points get the same hash value through the hashing technique, it is said that the two data points fall into the same hash bucket. The construction goal of the hash scheme is to make the adjacent points in the original data space fall into the same hash bucket through the hashing function. So the hash functions need to meet two conditions shown in definition 2. We will first introduce the definition of collision probability which indicates the probability that two data points are mapped to the same hash bucket through the given hash function.

Definition 1. *Given a hashing scheme $\mathcal{H}$, the collision probability between two elements $x, \bar{x} \in \mathbb{R}^d$ is defined by $p(x, \bar{x}) := \Pr_{h \sim \mathcal{H}}[h(x) = h(\bar{x})]$, where $h \sim \mathcal{H}$ denote a hash function h is sampled from the hashing scheme $\mathcal{H}$.*

The collision probability of two data points x and $\bar{x}$ is closely related to the distance between the two points. The larger the distance, the smaller the collision probability.

Let $D(\cdot, \cdot)$ be a distance function of elements from a dataset S, and for any $x \in S$, let $\mathcal{B}(x, r)$ denote the set of elements from S within the distance r from x.

Definition 2. *A family of functions $\mathcal{H} = \{h : S \to U\}$ is called (r_1, r_2, p_1, p_2)-sensitive for $D(\cdot, \cdot)$ if for any $x, \bar{x} \in S$*

- *if $\bar{x} \in \mathcal{B}(x, r_1)$ then $p(x, \bar{x}) \geq p_1$,*
- *if $\bar{x} \notin \mathcal{B}(x, r_2)$ then $p(x, \bar{x}) \leq p_2$.*

In order to guarantee the hash functions in a locality-sensitive family to be useful, it has to satisfy that $p_1 > p_2$ and $r_1 < r_2$. The hash methods employed by this paper are presented in algorithm 1.

Proposition 1. *For every $x, \bar{x} \in \mathbb{R}^n$, the LSH family $\mathcal{H}$ constructed by Algorithm 1 satisfies :*

$$p(x, \bar{x}) = e^{-\|x - \bar{x}\|_1 / (2\sigma)} = k(x, \bar{x})^{\frac{1}{2}}. \tag{3}$$

The proof is shown in Section 8.1.

After all input data points have been hashed into hash buckets, one can determine neighbors of query point by hashing the query point and searching elements in the bucket containing that query point.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

Algorithm 1 Localization-sensitive hashing.

Require: Dataset $\{x_i, y_i\}_{i=1}^n$; Dimension of data d; Bandwidth σ.
Ensure: Hash function h.
1: Sample $\rho \sim Poisson(d/(2\sigma))$.
2: Sample ρ dimensions from d dimensions $\zeta_1.....\zeta_\rho \in \{1.....d\}$ at random.
3: Sample ρ reference values $\xi_1.....\xi_\rho \in [0, 1]$ at random.
4: Given a data point x, for every $i = 1.....\rho$, set $b_i = 1$ if $x_{\zeta_i} > \xi_{\zeta_i}$ and $b_i = 0$ otherwise.
5: The hash value is the concatenation of $b_1.....b_\rho$.

4.2 Hashing based prediction

As mentioned before, HBP uses LSH to create a two-stage structure to compute the decision value approximately. Now, we will explain the main algorithm in detail.

Pre-processing In the pre-processing stage, training data points are hashed to different buckets according to the hash function.

1) Randomly construct L hash functions $h_1...h_L$ from hash scheme $\mathcal{H}$.
2) For each hash function h_j, sample a subset from dataset $X_j \in X$, every point in X_j is selected with probability $\delta = \frac{L}{n}$. In order to reduce computational complexity, we only run the hash process on a small subset rather than the whole training set. The result of our theorem proves that the method can still guarantee accuracy.
3) For each hash function, each data point in X_j is mapped to a hash value through the hash function. All data points in X_j of the same hash value constitute a hash bucket. And all hash buckets form a hash table corresponding to the hash function.
4) L hash functions can produce L hash tables.

Query In the query stage, the sampling points were used to estimate the response value.

1) Compute the hash value of the test point $\bar{x}$ through each hash function, and map the test point $\bar{x}$ to a hash bucket for each hash table.
2) For each hash table, randomly select a data point $x^{(j)}, j \in (1, L)$ from the hash bucket to which the test point is mapped if that hash bucket is not empty except for the test point $\bar{x}$. The L hash tables allow us to produce at most L independent samples because if the hash bucket where the test point is mapped is empty except for the test point, we can't get a sample for this hash table.
3) Compute the estimated values using every sample point:

$$Z_j = \frac{\hat{\boldsymbol{\alpha}}_j}{\sum_{j=1}^n \hat{\boldsymbol{\alpha}}_j} \frac{k(x^{(j)}, \bar{x}) \cdot |b_j(\bar{x})|}{\delta \cdot p(x^{(j)}, \bar{x})}, \tag{4}$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

where $b_j(\bar{x}) = \{x \in X'_j : h_j(x) = h_j(\bar{x})\}$ represents the set of the elements in the hash bucket where the point $\bar{x}$ mapped, and $|b_j(\bar{x})|$ represents the number of elements in $b_j(\bar{x})$.

4) Compute the accurate prediction value by averaging all of the samples produced by hash tables:

$$\hat{y} = \frac{1}{L}\sum_{j=1}^{L} Z_j. \tag{5}$$

5) The approximate prediction value can achieve the accuracy of $\|\hat{y} - y^*\|_p \leq \epsilon \cdot (3\tau n^{1/p} + \|y^*\|_p)$ with the time and space cost of $\mathcal{O}(\frac{log(n/\epsilon\tau)}{\tau^{0.5}\epsilon^{2.5}})$.

Algorithm 2 Hashing Based Kernel Prediction.

Require: Dataset $\{x_i, y_i\}_{i=1}^n$; test data point $\bar{x} \in \mathbb{R}^d$; kernel $k(\cdot,\cdot)$; LSH family $\mathcal{H}$; interger $1 \leq L \leq n$; the result of training stage $\hat{\boldsymbol{\alpha}}$;

Ensure: Estimated response value $\hat{y}$.

Pre-processing:
For $j = 1.....L$:
Sample a random hash function h_j from $\mathcal{H}$.
Get sampling dataset $X_j \subset X$, each point of X_j was selected with independent probability $\delta = \frac{L}{n}$.
For every $x \in X_j$, compute the hash value $h_j(x)$.
Query:
For $j = 1.....L$:
Compute the hash value of test data $h_j(\bar{x})$.
Sample a random point $x^{(j)}$ from $b_j(\bar{x}) = \{x \in X'_j : h_j(x) = h_j(\bar{x})\}$.
Let $Z_j \longleftarrow \frac{\hat{\boldsymbol{\alpha}}_j}{\sum_{i=1}^n \hat{\boldsymbol{\alpha}}_j} \cdot \frac{k\left(x^{(j)},\bar{x}\right)\cdot|b_j(\bar{x})|}{\delta \cdot p(x^{(j)},\bar{x})}$.
The prediction value $\hat{y} = \frac{1}{L}\sum_{j=1}^{L} Z_j$.

5 Theoretical Assessments

The problem of our proposed algorithm aims to solve is to obtain an approximation $\hat{y}$ to $y^* = \sum_{i=1}^n \hat{\boldsymbol{\alpha}}_i k(x_i, \bar{x})$. In this section, we introduce the theoretical bound of the proposed algorithm.

Theorem 1. *Given a kernel k, if there exists a distribution $\mathcal{H}$ of hash functions and $M \geq 1$ such that for every $x, \bar{x} \in \mathbb{R}^d$,*

$$M^{-1} \cdot k(x,\bar{x})^{1/2} \leq \Pr_{h\sim\mathcal{H}}[h(x) = h(\bar{x})] \leq M \cdot k(x,\bar{x})^{1/2}, \tag{6}$$

then we can compute an approximate vector $\hat{y}$ for kernel prediction in time $\mathcal{O}(\frac{log(n/\epsilon\tau)}{\tau^{0.5}\epsilon^{2.5}})$ such that with probability at least $1 - n^{-1}$ for all $i \in [n]$ it holds $|\hat{y}_i - y_i^*| \leq 3\epsilon\tau + \epsilon\,|y_i^*|$ and

$$\|\hat{y} - y^*\|_p \leq \epsilon \cdot (3\tau n^{1/p} + \|y^*\|_p). \tag{7}$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

Table 1. Datasets used in this paper.

Dataset	Instance	Feature	Type	Bandwidth
covtype	581,012	54	Multi-Classification	0.1
SUSY	1,000,000	18	Bi-classification	0.1
Census	2,458,285	68	Regression	0.05

The proof of Theorem 1 is in Section 8.2. $\tau \in (0,1)$ in Theorem 1 denotes a lower bound of $\frac{1}{n}\sum_i k(x_i, \bar{x})$. As shown in Eq.(3) in Section 4.1, the hashing scheme constructed by HBP algorithm satisfies the condition in Theorem 1. The above result shows the upper bound of error and the time complexity of the HBP method. As discussed before, the computational complexity of kernel prediction is $\mathcal{O}(n)$, while we need only $\mathcal{O}(\frac{log(n/\epsilon\tau)}{\tau^{0.5}\epsilon^{2.5}})$ with accuracy guarantees $\|\hat{y} - y^*\|_p \leq \epsilon \cdot (3\tau n^{1/p} + \|y^*\|_p)$.

6 Experiments

We evaluate the efficiency and effectiveness of the proposed algorithm by experiments on 3 large-scale datasets. The experiments with these datasets use the Laplacian kernel

$$k(x, y) = e^{-\|x-y\|_1/\sigma}. \tag{8}$$

All of the experiments are conducted on a server with 2.40GHZ Inter(R) Xeon(R) E5-2630 v3 CPU and 32GB of RAM in Matlab.

6.1 Datasets preparation

The performance of our proposed algorithm is presented on three large-scale real-world datasets: covtype[3], SUSY [4] and Census [5], which are generally used in the field of kernel machines [18][19]. The details of the datasets are shown in table 1.

We randomly selected 2.5×10^5 data points on each dataset. The features of datasets have been normalized. 70 percent of instances are used for training experiment and the rest for prediction. We measure the error by calculating root-mean-square error(RMSE) for regression problems and calculating the classification error for classification problems.

6.2 Performance of hashing based prediction

Our experiment contains two parts. Every experiment is repeated 10 times to avoid contingency. The final result is the average value of 10 experiments.

[3] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[4] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[5] https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

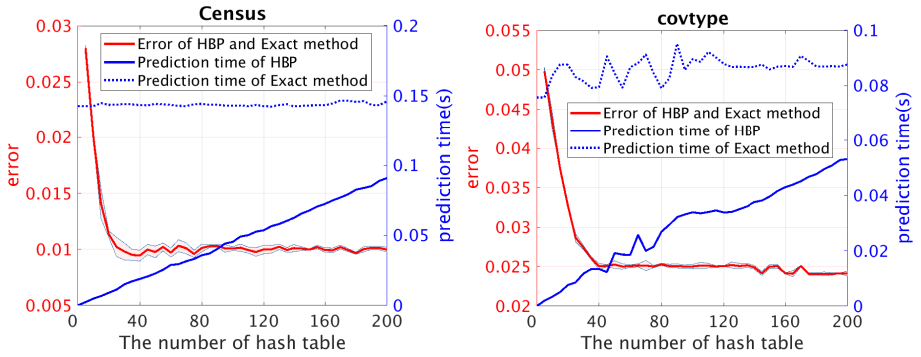

Fig. 2. Validate error between the HBP algorithm and the exact prediction method as well as the average prediction time for each test point of the two methods with respect to the number of hash table L on 2 large scale datasets: Census and covtype.

To focus on the effectiveness of our algorithm in the prediction stage, we consider the model $\hat{\boldsymbol{\alpha}}$ for kernel machine is given in the first part of our experiment. We randomly generate $\hat{\boldsymbol{\alpha}}$ and compare the error between the results of our approximation method and the exact value without using any approximation algorithm as well as the time complexity of the two methods.

Fig.2 shows the error and prediction time concerning the number of hash table L. The horizontal coordinate represents the number of hash table L. The vertical coordinate in the left represents the average errors of the prediction value between our algorithm and the exact algorithm, and the vertical coordinate in the right represent the prediction time of that two algorithms. With the increase of L, the error decreases at the beginning, and the rate of decrease become slow when L increases to a value. The prediction time of HBP increases with the increase of L. Our algorithm has a significant advantage over the accurate algorithm in prediction time.

In the second part of our experiment, we also compare our algorithm with 2 representative methods. Our algorithm employs KRR to finish the training process. For ensuring fairness, we use the same way to tune parameters σ in $2^{[-2:0.5:5]}$ and λ in $2^{[-40:3:8]}$ on each dataset and algorithm. The selected parameters are sufficient to achieve satisfactory results, although they may not be optimal.

The general introduction of the methods used in the experiment is as follows:

1) HBP: Our proposed algorithm, which uses Locality-Sensitive Hashing (LSH) for importance sampling to generate a fast prediction method for kernel machines.
2) Nyström [14]: An accurate and scalable Nyström scheme that made large-scale Nyström approximation possible. The algorithm first samples a large column subset from the input matrix, then only performs an approximate SVD on the inner submatrix.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

Table 2. Comparison of prediction time and test error in solving KRR problem between HBP, Nyström and RLS-Nyström on covtype, SUSY and Census datasets. We bold the numbers of the best algorithm.

Dataset	Metric	HBP	Nyström	RLS-Nyström
covtype	Time(s)	**4.409**	73.154	73.311
	Error	**0.2874 ±0.00003**	1.4919±0.00263	0.7713±0.00890
		$L = 50$	$m = 2500$	$m = 2500$
SUSY	Time(s)	**2.974**	24.130	24.211
	Error	0.6471 ± 0.0000	**0.6388 ±0.00169**	0.6552 ± 0.00036
		$L = 50$	$m = 2500$	$m = 2500$
Census	Time(s)	**5.354**	87.970	87.683
	Error	**0.2867±0.00079**	0.2900 ± 0.00163	1.2297 ± 0.00712
		$L = 50$	$m = 2500$	$m = 2500$

3) Recursive RLS-Nyström [16]: A recently Nystöm scheme using leverage score sampling.

Table 2 shows the experiment result of our algorithm and other methods mentioned before. Our algorithm is at a faster prediction speed than other methods. The larger the scale of data, the more obvious the time advantage of the proposed algorithm is. Simultaneously, HBP keeps the optimal value or just a little gap with the optimal, which validates the effectiveness of our algorithm.

7 Conclusions

We propose an algorithm HBP to effectively reduce the prediction cost of kernel machines on large-scale datasets. The algorithm opens the door of solving the kernel prediction problem by the hashing method. By using Locality-sensitive hashing (LSH) for importance sampling to achieve approximate calculation, the HBP algorithm reduces computational complexity and storage cost with less prediction accuracy loss. The experimental analysis on large-scale datasets showed that our HBP algorithm outperforms previous state-of-art solutions.

Since LSH is the data-independent hash scheme, the result of the approximate search is not extremely accurate. In the future, we intend to replace the LSH with the existing data-dependent hash algorithm to further reduce the prediction errors. For example, the data-dependent hashing schemes designed by [1] and [2] can be attempted to apply to our hashing based prediction method.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (No.61703396, No.61673293), the CCF-Tencent Open Fund,the Youth Innovation Promotion Association CAS,the Excellent Talent Introduction of Institute of Information Engineering of CAS (No.Y7Z0111107),

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

the Beijing Municipal Science and Technology Project (No. Z191100007119002), and the Key Research Program of Frontier Sciences, CAS (No. ZDBS-LY-7024).

8 Proof

In this section, we will provide proofs of some theorems in this paper. Section 8.1 presents the proof of Proposition 1 which references [3]. In section 8.2, we present the proof of Theorem 1 which is our main theoretical result.

8.1 The proof of Proposition 1

Proof. First assume that the data point $x, \bar{x}$ is one-dimensional. For the sake of clarity, we will describe LSH families $\mathcal{H}$ such that $\Pr_{h\sim\mathcal{H}}[h(x) = h(\bar{x})] = e^{-\|x-\bar{x}\|_1/\sigma}$. The Proposition 1 then follows simply by doubling the bandwidth σ. As shown in the Algorithm 1, we randomly sample $\xi \in [0, 1]$ and set the hash value $b(x) = 1$, if $x > \xi$, else $b(x) = 0$.

$$\Pr[b(x) = b(\bar{x})] = 1 - |x - \bar{x}|. \tag{9}$$

Then we consider the case that $x, \bar{x} \in [0, 1]^d$, applying this to a random dimension $\zeta \in \{1.....d\}$, we get:

$$\Pr[b(x) = b(\bar{x})] = \frac{1}{d}\sum_{\zeta=1}^{d}(1 - |x_\zeta - \bar{x}_\zeta|) = 1 - \frac{1}{d}\|x - \bar{x}\|_1 . \tag{10}$$

We repeat ρ times independently, then,

$$\Pr_{h\sim\mathcal{H}}[h(x) = h(\bar{x})] = (1 - \frac{1}{d}\|x - \bar{x}\|_1)^\rho. \tag{11}$$

At last, we sample $\rho \sim Possion(d/\sigma)$,

$$\Pr_{h\sim\mathcal{H}}[h(x) = h(\bar{x})] = \sum_{\rho=0}^{\infty}\frac{e^{-d/\sigma}\cdot(d/\sigma)^\rho}{\rho!}\cdot(1 - \frac{1}{d}\|x - \bar{x}\|_1)^\rho = e^{-\|x-\bar{x}\|_1/\sigma}. \tag{12}$$

8.2 The proof of Theorem 1

Note 1. Given a dataset $X = \{x_1, ..., x_n\} \subset \mathbb{R}^d$ and a test data point $\bar{x} \in \mathbb{R}^d$. $\omega_i = k(x_i, \bar{x})$ denoted the kernel value of data point x_i and test point $\bar{x}$. Let $\mathcal{H}$ be a family of hash function. For every x_i, $p_i = \Pr_{h\sim\mathcal{H}}[h(x_i) = h(\bar{x})]$ denoted the collision probability with $\bar{x}$. $b_h(\bar{x}) = \{i : h(x_i) = h(\bar{x})\}$ is the set of points with the same hash value as $\bar{x}$.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

Proof. In our algorithm, we hash each point only with probability $\delta = 1/(n\tau^{1-\beta})$, where $\tau \leq \frac{1}{n}\sum_i \omega_i$. Set $r_1, ..., r_n$ be Bernoulli random variables with $\Pr[r_i] = \delta$. $b'_h(\bar{x})$ is sparsified counterpart of $b_h(\bar{x})$,

$$b'_h(\bar{x}) = \{i : h(x_i) = h(\bar{x}) \text{ and } r_i = 1\}. \tag{13}$$

We may assume that all elements of $\hat{\boldsymbol{\alpha}}$ are positive otherwise we apply our algorithm to $\hat{\boldsymbol{\alpha}}_+$ and $\hat{\boldsymbol{\alpha}}_-$ separately. The vector $\hat{\boldsymbol{\alpha}}$ can be geometrically divided into $S_1, ..., S_T$ such that all elements in each group differ by at most a factor of two, where $T = log_2(n/\epsilon\tau)$. Therefore, our problem can be expressed as T subproblem.

On the t^{th} subproblem, our estimated response value is:

$$Z_{h,t} = \frac{\hat{\boldsymbol{\alpha}}_I}{A_t} \frac{k(\bar{x}, x_I)}{\delta p_I} |b_h(\bar{x})|, \tag{14}$$

where I is a random index from $H(x) \subseteq S_t$, $A_t = \sum_{i \in S_t} \hat{\boldsymbol{\alpha}}_i$.

Now we bound the variance:

$$\begin{aligned}
\mathrm{Var}[Z_{h,t}] \leq \mathbb{E}\left[(Z_{h,t})^2\right] &= \frac{1}{\delta^2}\mathbb{E}\left[\frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2 / |b'_h(\bar{x})|^2}\right] \\
&= \frac{1}{\delta^2} \mathop{\mathbb{E}}_{h\sim\mathcal{H}} \mathop{\mathbb{E}}_{i\in b'_h(\bar{x})} \left[|b'_h(\bar{x})|^2 \frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2}\right] \\
&= \frac{1}{\delta^2} \mathop{\mathbb{E}}_{h\sim\mathcal{H}} \left[|b'_h(\bar{x})| \sum_{i\in b'_h(\bar{x})} \frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2}\right] \\
&= \frac{1}{\delta^2} \mathop{\mathbb{E}}_{h\sim\mathcal{H}} \left[\sum_j [j \in b'_h(\bar{x})] \sum_i [i \in b'_h(\bar{x})] \frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2}\right] \\
&= \frac{1}{\delta^2} \sum_i \frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2} \sum_j \mathop{\mathbb{E}}_{h\sim\mathcal{H}} [[j \in b'_h(\bar{x})][i \in b'_h(\bar{x})]] \\
&= \frac{1}{\delta^2} \sum_i \frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2} \sum_j \Pr_{h\sim\mathcal{H}} [j \in b'_h(\bar{x}) \& i \in b'_h(\bar{x})].
\end{aligned} \tag{15}$$

The last term can be split into two expressions:

$$\frac{1}{\delta^2} \sum_i \frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2} \sum_{j:j\neq i} \Pr_{h\sim\mathcal{H}} [j \in b'_h(\bar{x}) \& i \in b'_h(\bar{x})] \tag{16}$$

and

$$\frac{1}{\delta^2} \sum_i \frac{\hat{\boldsymbol{\alpha}}_i^2\omega_i^2}{A_t^2 p_i^2} \mathop{\mathbb{E}}_{h\sim\mathcal{H}} [i \in b'_h(\bar{x})]. \tag{17}$$

Since $j \neq i$ in Eq.(16), we have:

$$\Pr_{h\sim\mathcal{H}} [j \in b'_h(\bar{x}) \& i \in b'_h(\bar{x})] = \delta^2 \Pr_{h\sim\mathcal{H}} [j \in b_h(\bar{x}) \& i \in b_h(\bar{x})] \leq \delta^2 p_j.$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

Therefore, Eq.(16) is upper bounded by $\sum_i \frac{\hat{\boldsymbol{\alpha}}_i^2 \omega_i^2}{A_t^2 p_i^2} \sum_j p_j$.

As $0 \leq 2-2\beta \leq 1$ and $0 \leq \beta \leq 1$, the inequalities $\frac{1}{n}\sum_i \omega_i^{2-2\beta} \leq \left(\frac{1}{n}\sum_i \omega_i\right)^{2-2\beta}$ and $\frac{1}{n}\sum_j \omega_j^{\beta} \leq \left(\frac{1}{n}\sum_i \omega_i\right)^{\beta}$ hold.

Using the inequalities and the definition in the theorem $\frac{w_i^\beta}{M} \leq p_i \leq M w_i^\beta$, we have:

$$\sum_i \frac{\hat{\boldsymbol{\alpha}}_i^2 \omega_i^2}{A_t^2 p_i^2} \sum_j p_j \leq \frac{4M^3}{n^2} \sum_i \omega_i^{2-2\beta} \sum_j \omega_j^{\beta} \leq 4M^3 \left(\frac{1}{n}\sum_i \omega_i\right)^{2-\beta}. \tag{18}$$

Let $\frac{1}{n}\sum_i \omega_i = \mu$, Eq.(16) is upper bounded by $4M^3\mu^{2-\beta}$.

We observe that

$$\mathop{\mathbb{E}}_{h\sim\mathcal{H}}\left[i \in b'_h(\bar{x})\right] = p_i \delta, \tag{19}$$

and therefore Eq.(17) is upper bounded by

$$\frac{1}{\delta}\sum_i \frac{\hat{\boldsymbol{\alpha}}_i^2 \omega_i^2}{A_t^2 p_i} \leq \frac{4M}{n^2\delta} \sum_i \omega^{2-\beta}. \tag{20}$$

Since $1 \leq 2-\beta$ and $\omega_i \leq 1$, it is easy to get:

$$\sum_i \omega_i^{2-\beta} \leq \sum_i \omega_i = n\mu. \tag{21}$$

And we have known that $\delta = \frac{1}{n\tau^{1-\beta}} \geq \frac{1}{n\mu^{1-\beta}}$, so we get the upper bound of Eq.(17) is $4M\mu^{2-\beta}$.

Eq.(16)+Eq.(17)

$$\mathrm{Var}\left[Z'\right] \leq 4(M^3+M)\mu^{2-\beta}. \tag{22}$$

It is obvious that the variance of our estimator is at most 4 times larger than the variance bound of kernel density estimate using Hashing-based-estimate method which is $(M^3+M)\cdot\mu^{2-\beta}$. To derive Theorem 1, set $\beta = 1/2$. It is sufficient to obtain a $(1+\epsilon)$-approximation for the t^{th} subproblem over $\mathcal{O}(\frac{1}{\tau^{0.5}\epsilon^{2.5}})$ independent samples. For all $t \in T$, the overhead of the whole process is at most a multiplicative factor $T = log(n/\epsilon\tau)$ compared to the case that we were creating a single data-structure for the same problem. Combining with the conclusion of [5], we obtain a straightforward analysis of the estimation error of the algorithm that for all $i \in [n]$ with probability at least $1-n^{-1}$, it holds $|\hat{y}_i - y_i^*| \leq 3\epsilon\tau + \epsilon|y_i^*|$. Summing over all indices and using triangle inequality gives $\|\hat{y}-y^*\|_p \leq \epsilon \cdot (3\tau n^{1/p} + \|y^*\|_p)$.

References

1. Andoni, A., Laarhoven, T., Razenshteyn, I., Waingarten, E.: Optimal hashing-based time-space trade-offs for approximate near neighbors. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 47–66. Society for Industrial and Applied Mathematics (2017)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37

2. Andoni, A., Razenshteyn, I.: Optimal data-dependent hashing for approximate near neighbors. In: Proceedings of the forty-seventh annual ACM symposium on Theory of computing. pp. 793–801. ACM (2015)
3. Backurs, A., Indyk, P., Wagner, T.: Space and time efficient kernel density estimation in high dimensions. In: Advances in Neural Information Processing Systems. pp. 15773–15782 (2019)
4. Camoriano, R., Angles, T., Rudi, A., Rosasco, L.: Nytro: When subsampling meets early stopping. In: Artificial Intelligence and Statistics. pp. 1403–1411 (2016)
5. Charikar, M., Siminelakis, P.: Hashing-based-estimators for kernel density in high dimensions. In: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS). pp. 1032–1043. IEEE (2017)
6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the twentieth annual symposium on Computational geometry. pp. 253–262. ACM (2004)
7. Van de Geer, S.A.: Applications of empirical process theory, vol. 91. Cambridge University Press Cambridge (2000)
8. Gionis, A., Indyk, P., Motwani, R., et al.: Similarity search in high dimensions via hashing. In: Vldb. vol. 99, pp. 518–529 (1999)
9. Gui, J., Liu, T., Sun, Z., Tao, D., Tan, T.: Fast supervised discrete hashing. IEEE transactions on pattern analysis and machine intelligence **40**(2), 490–496 (2017)
10. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media (2009)
11. Hsieh, C.J., Si, S., Dhillon, I.S.: Fast prediction for large-scale kernel machines. In: Advances in Neural Information Processing Systems. pp. 3689–3697 (2014)
12. Ju, X., Wang, T.: A hash based method for large scale nonparallel support vector machines prediction. Procedia Computer Science **108**, 1281–1291 (2017)
13. Kumar, S., Mohri, M., Talwalkar, A.: Sampling techniques for the nystrom method. In: Artificial Intelligence and Statistics. pp. 304–311 (2009)
14. Li, M., Kwok, J.T., Lu, B.L.: Making large-scale nyström approximation possible. In: ICML. pp. 631–638 (2010)
15. Liu, M., Shang, Z., Cheng, G.: Sharp theoretical analysis for nonparametric testing under random projection. In: Conference on Learning Theory. pp. 2175–2209 (2019)
16. Musco, C., Musco, C.: Recursive sampling for the nystrom method. In: Advances in Neural Information Processing Systems. pp. 3833–3845 (2017)
17. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Advances in neural information processing systems. pp. 1177–1184 (2008)
18. Rudi, A., Camoriano, R., Rosasco, L.: Less is more: Nyström computational regularization. In: Advances in Neural Information Processing Systems. pp. 1657–1665 (2015)
19. Rudi, A., Carratino, L., Rosasco, L.: Falkon: An optimal large scale kernel method. In: Advances in Neural Information Processing Systems. pp. 3888–3898 (2017)
20. Rudi, A., Rosasco, L.: Generalization properties of learning with random features. In: Advances in Neural Information Processing Systems. pp. 3215–3225 (2017)
21. Shawe-Taylor, J., Cristianini, N., et al.: Kernel methods for pattern analysis. Cambridge university press (2004)
22. Williams, C.K., Seeger, M.: Using the nyström method to speed up kernel machines. In: Advances in neural information processing systems. pp. 682–688 (2001)
23. Zhang, X., Liao, S.: Incremental randomized sketching for online kernel learning. In: International Conference on Machine Learning. pp. 7394–7403 (2019)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_37