STAT 538 Paper Review Project
Divide and Conquer Kernel Ridge Regression

Ming-Chun Wu
Department of Statistics
University of Washington
chunwu@uw.edu

Abstract

The demand of scalable machine learning methods are getting higher and higher because of modern large-scale dataset. It is thus important to develop scalable algorithms with both statistical guarantee and computational feasibility. This report review one of the recent development, divide and conquer kernel ridge regression, to show how we can speed up a kernel machine for large scale data while still achieving statistical optimality. Theoretical results are presented along with reproduced experiments of the original paper Zhang et al. [2015].

1 Introduction

Regression problem is widely encountered in many applications, and one of the most important scenarios is unsupervised learning. Given data $((x_i, y_i))_{i=1}^N$ where $x_i \in \mathcal{X}$ and $y_i \in \mathbb{R}$, a regression problem aims to find a function f that produces good prediction of y given x. Under mean squared loss $\mathbb{E}[y - f(x)]^2$, it is well known that the optimal predictor is the conditional expectation $f^*(x) = \mathbb{E}[Y|X = x]$. The simple but widely used linear regression assume the linear model of the conditional expectation $\mathbb{E}[Y|X = x]$. Linear model is simple and easy to be interpreted and thus is popular for inferential tasks. However, for prediction purpose as in unsupervised learning, linear models are usually not powerful enough because the class of all linear model may be too small to contain a good predictor. This is where nonparametric regression came out to solve the challenge.

In nonparametric regression setting, we assume $f \in \mathcal{H}$ where $\mathcal{H}$ is a functional space that may have infinite dimension such that it is big enough to contain a good predictor. This frame work extends beyond squared error loss, give loss function $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ we aim to

$$\min_{f \in \mathcal{H}} \mathbb{E} l(f(x), y)$$

However, the risk $\mathbb{E} l(f(x), y)$ is an oracle quantity since we do not know the probability distribution of (x, y). Instead, we minimize a reasonable estimate of the risk, which leads to the empirical risk minimization problem

$$\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} l(f(x), y)$$

This is a nontrivial optimization problem and usually require additional effort to solve. Fortunately, if we imitate ridge regression by introducing the regularization $\|f\|_{\mathcal{H}}^2$ and regularization parameter λ, the problem becomes

$$\min_{f \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} l(f(x), y) + \lambda \|f\|_{\mathcal{H}}^2$$

If the functional space is nice enough to be a Reproducing Kernel Hilbert Space (RKHS) with kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, then the solution has the form $f(x) = \sum_{i=1}^{N} a_i k(x, x_i)$. All we need is to solve a finite dimensional optimization problem to find a_i's. This nice property makes kernel methods a popular framework to extend linear methods.

Nonetheless, kernel based methods are not easy to apply to large scale dataset since compute the $N \times N$ kernel matrix is computational inefficient. In general we need $\mathcal{O}(N^3)$ time complexity and $\mathcal{O}(N^2)$ memory and thus is is not scalable with large N. Either it takes too much time for computing or there is no single device to store the dataset and to compute the kernel matrix. Recently, more and more researchers have worked on such challenges. Roughly speaking, the general principle is to speed up the computation by approximating the optimal solution. Usually, the price we paid is losing the statistical guarantee of the resulting estimations/predictions. Hence, an important task is to find scalable methods that have certain levels of statistical guarantees. This report review one of the recent development, divide and conquer kernel ridge regression, or called FastKRR, as an attractive example.

FastKRR speeds up kernel ridge regression (KRR), a popular kernel based method, by using distributed implementation and still achieves minimax convergence rate for common kernel functions. It divides the large dataset into m partitions and trains m separate KRR based on these m partitions. In prediction step, FastKRR averages the m distributed KRR as its own predictor. Theory shows that with the careful choice of the partition m and the regularization parameter λ, FastKRR has optimal statistical property. Moreover, because of the distributed nature of FastKRR, we can further speed it up using multiprocessing or parallel computing in practice. Therefore, it is attractive for large scale dataset.

Although the main purpose of the report is to review FastKRR, we will also present related methods that serve as competitors in the experimental studies. One common way to speed up kernel machines is to approximate the kernel matrix by a low-rank surrogate and then implement the kernel machines. Later, we will present random Fourier feature Rahimi and Recht [2008] and Nystrom method Williams and Seeger [2001] for illustration. Moreover, random feature and Nystrom method are used to compare to the FastKRR in the experimental studies.

2 Reproducing Kernel Hilbert Space

Now, we briefly introduce RKHS and the kernels used in this report.

2.1 Kernels

Definition 1 (RKHS)**.** *A function space $\mathcal{H}$ is RKHS if it is a Hilbert space with bounded evaluation functionals. That is, for all $f \in \mathcal{H}$, $f : \mathcal{X} \to \mathbb{R}$, we have $|f(x)| < \infty, \forall x \in \mathcal{X}$.*

As shown in lectures, we have many good properties of RKHS.

Lemma 1 (Equivalent definition of RKHS)**.** *$\mathcal{H}$ is a RKHS if it has a unique reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that k has the reproducing property $f(x) = \langle f, k(\cdot, x) \rangle_{\mathcal{H}}, \forall f \in \mathcal{H}, x \in \mathcal{X}$.*

Definition 2 (psd kernel)**.** *A kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is psd if $\sum_{i,j=1}^{N} a_i a_j k(x_i, x_j) \geq 0, \forall a_i, N$. Moreover, as shown in the lectures, a kernel is psd if and only if it has the reproducing property.*

The kernels used in this report are

- Sobolev kernel of order 1; $k(x, x') = 1 + \min(x, x')$
- Gaussian kernel: $k(x, y) = \exp^{-\frac{\|x-y\|_2^2}{2\sigma^2}}$ where σ is the hyper parameter

The Gaussian kernel is a well known psd kernel. Therefore, we explore more properties of the Sobelev kernel.

Definition 3 (Sobolev WassermanGu [2002])**.** *Sobolev space of order m is defined as the function space $W_m = \{f \in L_2[0,1] : D^m f \in L_2[0,1]\}$ where D^m denotes the m-th weak derivative, and* *with inner product* $\forall f, g \in W_m$

$$\langle f, g\rangle_{\mathcal{H}} = \sum_{k=0}^{m-1} f^{(k)}(0)g^{(k)}(0) + \int_0^1 f^{(m)}(u)g^{(m)}(u)du \tag{1}$$

The corresponding kernel is

$$k(x, y) = \sum_{k=0}^{m-1} \frac{x^k}{k!}\frac{x^k}{k!} + \int_0^{\min(x,y)} \frac{(x-u)^{m-1}}{(m-1)!}\frac{(x-u)^{m-1}}{(m-1)!}du \tag{2}$$

We can show that k is the valid producing kernel by letting $g = k(x, \cdot)$ into (1). Note that $g^{(k)}(0) = \frac{x^k}{k!}, k = 0, 1, \ldots, m-1$ and $g^{(m)}(u) = \frac{[\max(0,x-u)]^{m-1}}{(m-1)!}$ we then get $\langle f, k(x, \cdot)\rangle = f(x)$.

Corollary 1. *psd of Sobolev kernel The kernel* $k(x, y) = 1 + \min(x, y)$ *is psd.*

Proof. This is a special case of Sobolev kernel with $m = 1$. Since we have shown that (2) is the reproducing kernel, by the equivalence of psd kernel and reproducing kernel as stated in definition 2 we then finish the proof. □

The concept of weak derivative is a generalization of derivative, if a function is differentiable, its weak derivative equals to ordinary derivative. Intuitively, the Sobolev space generalizes $\mathcal{C}^m[0, 1]$, set of functions with m continuous derivatives. If $f, g \in \mathcal{C}^m[0, 1]$, apply Taylor expansion with integral form of reminder

$$f(x) = \sum_{k=0}^{m-1} \frac{1}{k!}f^{(k)}(0) + \int_0^x \frac{(x-u)^{m-1}}{(m-1)!}f^{(m)}du \tag{3}$$

Intuitively speaking, if we let $\phi_i(x) = x^k, a_k = \frac{1}{k!}f^{(k)}(0), b_k = \frac{1}{k!}g^{(k)}(0), \lambda_k = (\frac{1}{k!})^2$ and neglect the integral terms, we have

$$f \approx \sum_{k=0}^{m-1} a_k\phi_k(x),\ g \approx \sum_{k=0}^{m-1} b_k\phi_k(x)$$

$$\langle f, g\rangle \approx \sum_{k=0}^{m-1} a_k b_k/\lambda_k,\ k(x, y) \approx \sum_{k=0}^{m-1} \lambda_k\phi_k(x)\phi_k(y)$$

Therefore, the eigenvalues of the Sobolev kernel is

$$\lambda_k = (1/(k!))^2 \leq Ck^{-2v} \tag{4}$$

with $v = 1$ and some constant C. We can see λ_k is the k-th eigenvalue of the kernel k. We say that a kernel has v-polynomial decaying eigenvalues if (4) holds. With this condition, the FastKRR has good statistical property as shown in the next section.

2.2 Kernel Ridge Regression

It is usually difficult to come out a RKHS directly. However, there is a one to one correspondence between psd kernel and RKHS. That is, every psd kernel induces a RKHS and vice versa. With this property, we can define a psd kernel, which is usually easier, to induce a RKHS. Then given $\mathcal{H}$ along with kernel k, the kernel ridge regression problem is defined as

$$\min_{f\in\mathcal{H}} \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2 + \lambda\|f\|_{\mathcal{H}}^2$$

We can apply the so called representer theorem to show that the solution has the form $f(x) = \sum a_i k(x, x_i)$. Let $\mathcal{H} = \mathcal{H}_\perp \bigoplus \mathcal{H}_\parallel$ where $\mathcal{H}_\parallel = \text{span}\{k(\cdot, x_i)\}$. Then $f = f_\perp + f_\parallel$, and by the reproducing property $f(x) = \langle f, k(\cdot, x)\rangle_{\mathcal{H}} = \langle f_\perp + f_\parallel, k(\cdot, x)\rangle_{\mathcal{H}} = \langle f_\perp, k(\cdot, x)\rangle_{\mathcal{H}} + \langle f_\parallel, k(\cdot, x)\rangle_{\mathcal{H}} = \langle f_\parallel, k(\cdot, x)\rangle_{\mathcal{H}}$ and $\|f\|_{\mathcal{H}}^2 = \|f_\perp + f_\parallel\|_{\mathcal{H}}^2 = \|f_\perp\|_{\mathcal{H}}^2 + \|f_\parallel\|_{\mathcal{H}}^2 \geq \|f_\parallel\|_{\mathcal{H}}^2$. Therefore, it is not difficult to see that the solution f must satisfy $f_\perp \equiv 0$ and thus f is a linear combination of $k(\cdot, x_i)$'s. Now, let K

be the $N \times N$ kernel matrix with $K_{ij} = k(x_i, x_j)$, $y = [y_1, y_2, \ldots, y_N]^T$ and $a = [a_1, a_2, \ldots a_N]^T$, we can express the kernel ridge problem as

$$\min_{a \in \mathbb{R}^N} \frac{1}{N}(y - Ka)^T(y - Ka) + \lambda a^T K a \tag{5}$$

To solve the problem, we take gradient with respect to a

$$\begin{aligned} \frac{1}{N} 2K(y - Ka) + 2\lambda K a &= 0 \\ K(y - (K + N\lambda I)a) &= 0 \\ a &= (K + \lambda N I)^{-1} y \end{aligned}$$

So, in the training step, KRR first compute the kernel matrix K and estimate $a = (K + \lambda NI)^{-1}y$. Given x, the predicted label is $\hat{y} = f(x) = \sum_{i=1}^N a_i k(x, x_i)$. In the training set, KRR needs to invert the a $N \times N$ matrix, which takes $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ space. Therefore, it is usually infeasible to apply KRR on large dataset.

3 Divide and Conquer Kernel Ridge Regression

3.1 The Algorithm

To fix the drawback of KRR for large scale dataset, we can divide the original problem into several pieces and tackle them separately. The concept of divide and conquer is widely used in the computer science society to deal with complex problem. The resulting Fast KRR (divide and conquer KRR) is

1. Randomly divide the dataset into m subsets with similar sizes
2. For each subset S_i, fit a kernel ridge regression by solving

$$\hat{f}_i = \arg\min_{f \in \mathcal{H}} \frac{1}{|S_i|} \sum_{(x_i, y_i) \in S_i} (f(x_i) - y_i)^2 + \lambda \|f\|_{\mathcal{H}}^2$$

3. Output the predictor $\bar{f} = \frac{1}{m} \sum \hat{f}_i$.

For each local kernel machine, the time complexity is $\mathcal{O}(N^3/m^3)$ and space $\mathcal{O}(N^2/m^2)$. If we implement the algorithm in serial way, i.e. fit one local KRR at a time, then the overall time complexity is $\mathcal{O}(N^3/m^2)$ with memory $\mathcal{O}(N^2/m^2)$. Another way is to use parallel implementation such that we have time complexity $\mathcal{O}(N^3/m^3)$ with memory $\mathcal{O}(M^2/m)$.

Although the algorithm of divide and conquer KRR is simple to understand, with the careful choice of m and the regularization parameter λ, the method achieve minimax optimal convergence rate.

3.2 Theoretical Guarantee [Zhang et al., 2015]

Given a RKHS $\mathcal{H}$ and its psd kernel $k(x, y)$, suppose we have (Mercer's theorem)

$$k(x, y) = \sum_{k=1}^{\infty} \lambda_k \phi_k(x) \phi_k(y)$$

where λ_k's are nonnegative eigenvalues and ϕ_k's are eigenfunctions. Then we want to control the behaviors of ϕ_k's by the following assumptions.

- Assumption A: For some constant $k \geq 2$, there exists $\rho > 0$ such that $\mathbb{E}[\phi_j(X)^{2k}] \leq \rho^{2k}, \forall j \in \mathbb{N}$
- Assumption B: The true model $f^*(x) = \mathbb{E}[Y|x]$ is in $\mathcal{H}$ and we have bounded variance of noise, i.e. $\mathbb{E}[(Y - f^*(x))^2|x] \leq \sigma^2, \forall x \in \mathcal{X}$

Then for particular kernels like the Sobolev kernel or Gaussian kernel, we have the following statistical guarantees.

Theorem 1 (Polynomial decaying kernel)**.** *Suppose the kernel has v-polynomial decaying eigenvalues, i.e. $\lambda_i \le Ci^{-2v}$ and assumption A and B holds. Use the regularity parameter $\lambda = N^{-\frac{2v}{2v+1}}$ for all local KRR in the FastKRR algorithm, we have the error bound*

$$\mathbb{E}[\|\bar{f} - f\|_2^2] = \mathcal{O}\left(\left(\frac{\sigma^2}{N}\right)^{\frac{2v}{2v+1}}\right) \tag{6}$$

if the number of local KRR satisfies $m \le c\left(\frac{N^{\frac{2(k-4)v-k}{2v+1}}}{\rho^{4k}\log^k N}\right)^{1/(k-2)}$ and c is a constant only depends on v. Moreover, the upper bound has the optimal convergence rate.

Theorem 2 (SubGaussian decaying kernel)**.** *Suppose the kernel has exponentially decaying (or sub Gaussian eigendecay) eigenvalues, i.e. $\lambda_i \le c_1 e^{-c_2 i^2}$ for some constant c_1, c_2 and assumption A and B holds. Use the regularity parameter $\lambda = N^{-1}$ for all local KRR in the FastKRR algorithm, we have the error bound*

$$\mathbb{E}[\|\bar{f} - f\|_2^2] = \mathcal{O}\left(\sigma^2\frac{\sqrt{\log N}}{N}\right) \tag{7}$$

if the number of local KRR satisfies $m \le c\frac{N^{\frac{k-4}{k-2}}}{\rho^{\frac{4k}{k-2}}\log^{\frac{2k-1}{k-2}} N}$ where c is a constant only depends on c_2. Again, the upper bound has the optimal convergence rate.

4 Related Methods

Before reproduce the experiments in the next section, we introduce two competing methods.

4.1 Random Fourier Feature

Let us consider Gaussian kernel to illustrate the idea of random feature method. Given Gaussian kernel

$$k(x, y) = k(x - y) = e^{-\frac{1}{2\sigma^2}\|x-y\|_2^2}$$

by Fourier transform, or the characteristic function of multivariate normal distribution, we know

$$k(x, y) = k(x - y) = \int e^{jw^T(x-y)}dP(w) = \mathbb{E}_w[e^{jw^T(x-y)}]$$

where P is the distribution function of $N(0, \frac{1}{\sigma^2}I)$. We can further simplify to get

$$\begin{aligned} k(x - y) &= \mathbb{E}_w[\cos(w^T(x - y))] + j\mathbb{E}_w[\sin(w^T(x - y))] \\ &= \mathbb{E}_w[\cos(w^T(x - y))] \qquad\qquad k(x - y) \in \mathbb{R} \end{aligned}$$

If we introduce a random variable $b \sim [0, 2\pi]$ and b is independent to w

$$\mathbb{E}_w[\cos(w^T(x - y))] = \mathbb{E}_w[\cos(w^T(x - y))] + \mathbb{E}_{w,b}[\cos(w^T(x + y) + b)]$$

since $\mathbb{E}_{w,b}[\cos(w^T(x+y)+b)] = \mathbb{E}_w\mathbb{E}_b[\cos(w^T(x+y)+b)|w] = \mathbb{E}_w 0 = 0$. Then use trigonometric formula we can get

$$\begin{aligned} k(x, y) &= \mathbb{E}_{w,b}[\sqrt{2}\cos(w^T + b)\sqrt{2}\cos(w^T y + b)] \\ &\approx \sum_{i=1}^{D}\left[\sqrt{\frac{2}{D}}\cos(w_i^T x + b_i)\sqrt{\frac{2}{D}}\cos(w_i^T y + b_i)\right] \end{aligned}$$

where $w_i \overset{i.i.d.}{\sim} N(0, \frac{1}{\sigma^2}I)$ and $b_i \overset{i.i.d.}{\sim} U[0, 2\pi]$. In other words, we can apply the finite dimension random feature transform $\phi : x \to [\sqrt{2/D}\cos(w_i^T x + b_i), i = 1, \dots D]^T \in \mathbb{R}^D$ on the original data, then perform linear regression of the random features.

4.2 Nystrom Method

Nystrom method is another method to find a low rank approximation of the kernel matrix. By the definition of eigenfunction, we have

$$\int k(x,y)\phi_i(\tilde{x})dP(\tilde{x}) = \lambda_i\phi_i(x),\ \int \phi_i(\tilde{x})\phi_j(\tilde{x})dP(\tilde{x}) = \delta_{ij}$$

If we have $\tilde{x}_i \overset{i.i.d.}{\sim} P, i = 1,\ldots,q$, then we have the following approximations

$$\frac{1}{q}\sum_{l=1}^{q} k(x,\tilde{x}_l)\phi_i(\tilde{x}_l) \approx \lambda_i\phi_i(x) \tag{8}$$

$$\frac{1}{q}\sum_{l=1}^{q} \phi_i(\tilde{x}_l)\phi_j(\tilde{x}_l) \approx \delta_{ij} \tag{9}$$

Let $K_{qq} = [k(\tilde{x}_i,\tilde{x}_j)]$ and its eigen decomposition $U\Sigma U^T$. Let u_i be the i-th column of U and σ_i be the i-th eigen value, and compare to (8) and (9) we get

$$\lambda_i = \sigma_i/q,\ [\phi_i(\tilde{x}_j),\ldots,\phi_i(\tilde{x}_q)]^T = \sqrt{q}u_i.$$

Substitute the results back to (8) and (9), we will get

$$\phi_i(x) = \frac{\sqrt{q}}{\sigma_i}[k(x,\tilde{x}_1),k(x,\tilde{x}_2),\ldots,k(x,\tilde{x}_q)]u_i, i = 1,2,\ldots,q$$

With these approximate eigenfunctions $\phi_i, i = 1,2,\ldots,q$, we can approximate the kernel matrix by

$$K \approx \sum_{l=1}^{q} \lambda_l\phi_l(x)\phi_l(x') \tag{10}$$

It is better to express this in matrix form. Suppose we randomly subsample q data points from the original dataset and let K_{qq} be the kernel matrix of the subsamples. Let K be the original $N \times N$ kernel matrix, without lost of generality, we can assume the upper left submatrix of K being K_{qq}, then we have the block form

$$K = \begin{bmatrix} K_{qq} & K_{q(N-q)} \\ K_{(N-q)q} & K_{(N-q)(N-q)} \end{bmatrix} \tag{11}$$

The Nystrom method approximate the original kernel by

$$K \approx \begin{bmatrix} K_{qq} \\ K_{(N-q)q} \end{bmatrix} K_{qq}^{+} \begin{bmatrix} K_{qq} & K_{q(N-q)} \end{bmatrix} \tag{12}$$

where K_{qq}^{+} is the pseudo inverse of K_{qq}.

5 Experiments

5.1 Simulation Studies

We first reproduce the simulation results. Assume $y = f^*(x) + \epsilon, x \in [0,1]$ were $f^*(x) := \min(x, 1-x)$ and $\epsilon\ N(0,\sigma^2)$ with variance $\sigma^2 = 1/5$. The function $f(x)$ is in the Sobolev space with order 1, so we use the kernel $k(x,y) = 1 + \min(x,y)$. From theorem 1 with $v = 1$, we choose the regularity parameter $\lambda = N^{-2/3}$. We measure the oracle mean square error $\|\bar{f} - f^*\|_2^2$ and reproduce the simulations by varying the number of local KRR, m, and the number of samples, N.

As summarized in Fig. 1, as long as the number of local KRR is not too large compare to N, the performance of FastKRR converges to that of KRR. Therefore, FastKRR have comparable statistical performance as KRR while enjoying better computational efficiency. Recall that with the serial implementation of FastKRR, the time and memory complexities are $\mathcal{O}(N^3/m^2)$ and $\mathcal{O}N^2/m^2$ respectively. And for parallel implementation, we have $\mathcal{O}(N^3/m^3)$ time complexity and $\mathcal{O}(N^2/m)$

Figure 1: With regularity parameter $\lambda = N^{-2/3}$.

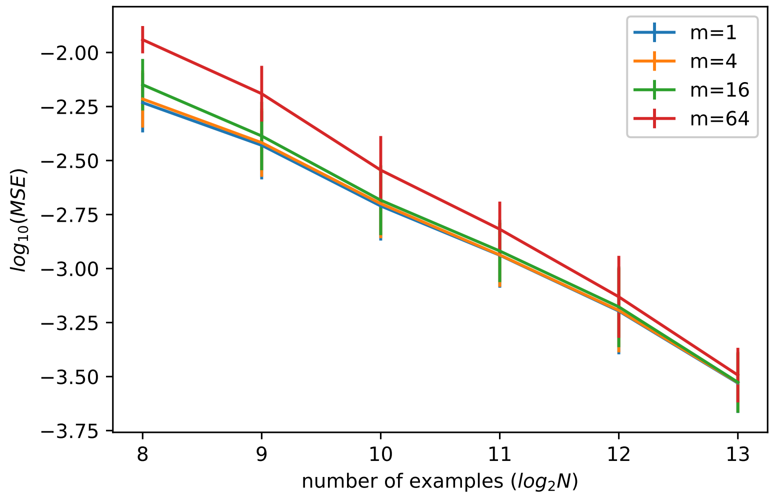
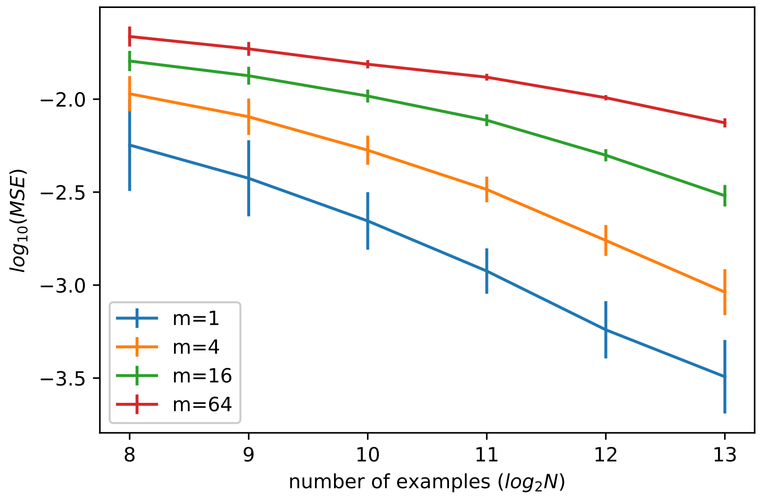


Figure 2: With the wrong $\lambda = (N/m)^{-2/3}$ such that the FastKRR over-regularized the model.



memory. So, we can even trade off in computational resources while maintaining the statistical performance at the same time.

Now, we show the importance of choosing the right regularity parameter λ, we show the result in Fig 2 with the wrong parameter, $\lambda = (N/m)^{-2/3}$, such that the FastKRR over-regularized the model. We see that the FastKRR fails to converge to KRR even as N increases, which suggests the importance of theorem 1 and theorem 2.

As shown in figure 3, the number of partitions required to achieve optimal performance does not grow too slow as N increases.

5.2 Million Song Dataset

We use the Million Song Dataset

Figure 3: We can increase m fairly fast as N increases while achieving good statistical performance.

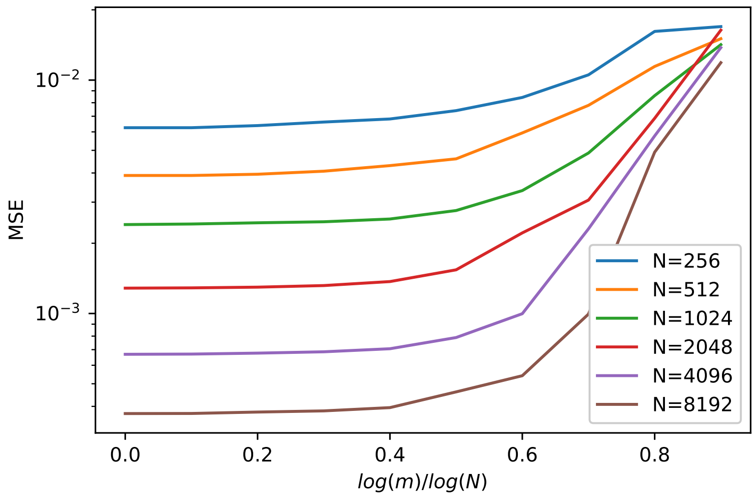
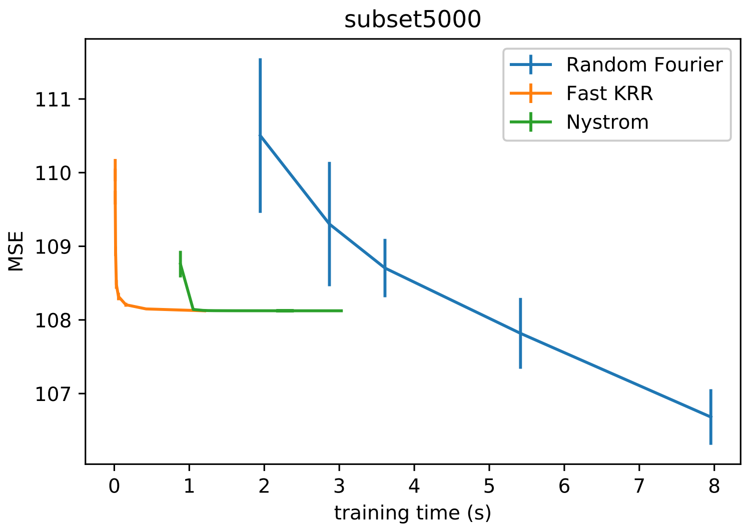


Figure 4: Subset of 5000 songs.



`https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd`

to test the FastKRR algorithm. The original dataset contains 4163715 training examples each represents a song with 90 features and with 51630 testing examples. The label y is the year when the song published, where y is between 1922 to 2011. Our goal is predict the year given the 90 features.

We use Gaussian Kernel with $\sigma = 6$ as suggested by the original paper. The competitors are FastKRR, fast random feature method and Nystrom method. We compare these three methods by measuring both their statistical performance and computational efficiency. We use three different sizes of subsets of the original dataset. We first randomly sample the subset from the original dataset, then use $80\% - 20\%$ split for training set and hold-out test set. The performance measure is the mean squared error of the test set. The experiments are conducted on google cloud compute engine with 8vCPU and 52GB memory. We use the multiprocessing to implement the FastKRR.

As we can see from fig 4, 5 and 6, FastKRR has the best overall performance. The random Fourier feature does not stable and in general has the worst performance. The Nystrom method can achieve

Figure 5: Subset of 10000 songs. Random Fourier feature is not stable.

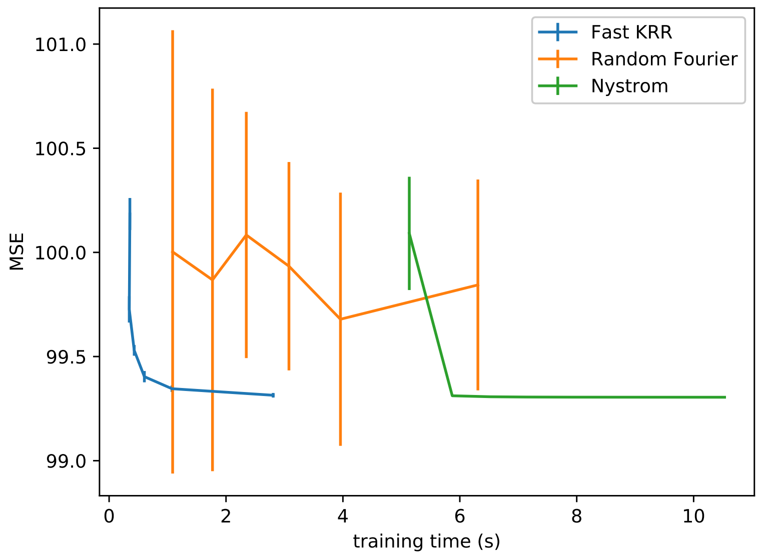
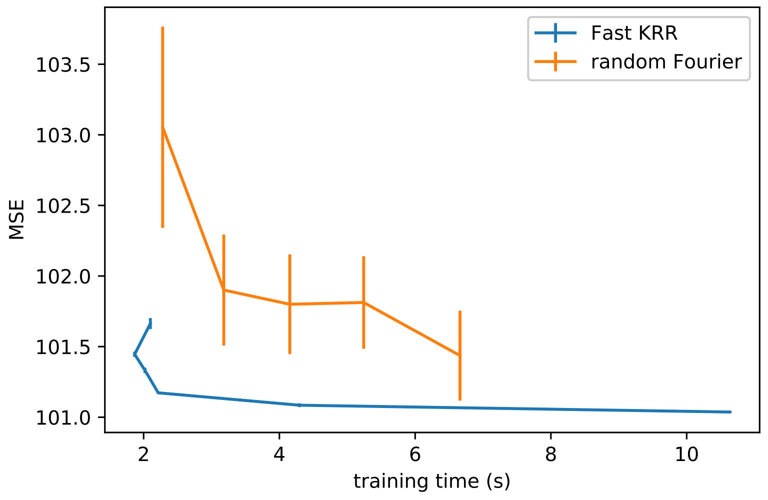
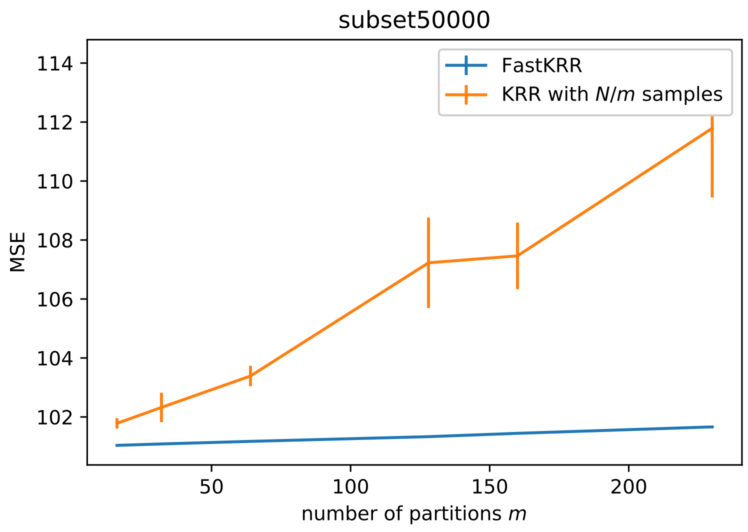


Figure 6: Subset of 50000 songs. The Nystrom method needs to store the $N \times N$ kernel matrix, which leads to extreme high demand of memory. Therefore Nystrom approach is not feasible in this case.



the same statistical accuracy as FastKRR if there is enough memory. Even though Nystrom method approximate the original kernel matrix, it still require to store the $N \times N$ kernel matrix in the training step. This leads to extremely high demand of memory. For example, if $N = 50000$ and use floating number (8 byte in Python) for each entry of the kernel matrix, we need $8 * 50000^2 \approx 10GB$ memory in python. So, we failed to train Nystrom due to the limitation of memory. However, the FastKRR not only improves the speed, it also reduces the required memory. For implementation using multiprocessing, we have $\mathcal{O}(N^2/m))$ and thus it reduce the require memory by factor m. For serial implementation, i.e. train one local KRR each time, we can further reduce the required memory to $\mathcal{O}(N^2/m^2)$.

Figure 7: Comparison of KRR and FastKRR. FastKRR indeed outperform KRR with N/m samples. In parallel implementation, FastKRR use m times more memory (and data) to improve the statistical accuracy.



Moreover, the distributed nature of FastKRR allows us to take the advantage of modern multi-core computer or large-scale distributed clusters. The parallel implementation of FastKRR is not difficult. On the other hand, it is nontrivial to implement Nystrom approach and Fast Fourier feature.

6 Concluding Remarks

The divide and conquer FastKRR speeds up the kernel ridge regression by equally partition the dataset into m subsets, then train m KRR separately and average the m KRR to produce the final predictor. For common kernels like Gaussian kernel, with carefully choices of the regularization parameter and the number of partitions according to the sample size, FastKRR achieves optimal statistical guarantee. Meanwhile, FastKRR gains m^2 factor in both time complexity and memory in the serial implementation. With parallel implementation, FastKRR has m^3 time faster speed and m time less memory than KRR. So, FastKRR serves as an attractive example of scaling kernel based learning method.

References

Gong Gu. *Smoothing Spline ANOVA Models*. Springer, 2002.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008. URL `http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf`.

Larry Wasserman. Lecture notes in function spaces. URL `http://www.stat.cmu.edu/~larry/=sml/functionspaces.pdf`.

Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001. URL `http://papers.nips.cc/paper/1866-using-the-nystrom-method-to-speed-up-kernel-machines.pdf`.

Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *Journal of Machine Learning Research*, 16:3299–3340, 2015. URL `http://jmlr.org/papers/v16/zhang15d.html`.