Divide and Conquer Kernel Ridge Regression

Yuchen Zhang  John Duchi  Martin Wainwright

University of California, Berkeley

Conference on Learning Theory 2013
Problem set-up

**Goal** Solve

\[
\text{minimize } \mathbb{E}[(f(x) - y)^2] \\
\text{subject to } f \in \mathcal{H}
\]

where \((x, y) \sim \mathbb{P} \). \(\mathcal{H}\) is a Reproducing Kernel Hilbert Space (RKHS).
Problem set-up

Goal Solve

\[
\text{minimize } \mathbb{E}[(f(x) - y)^2]
\]
subject to \( f \in \mathcal{H} \)

where \((x, y) \sim \mathbb{P}\). \( \mathcal{H} \) is a Reproducing Kernel Hilbert Space (RKHS).

\( \mathcal{H} \) defined by kernel \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \):

\[
\mathcal{H} = \{ f : f = \sum_i \alpha_i k(x_i, \cdot), \ x_i \in \mathcal{X} \}. 
\]
Kernel regression review

1. Linear regression $f(x) = \theta^T x$ only fits linear functions.
Kernel regression review

1. Linear regression $f(x) = \theta^T x$ only fits linear functions.
2. Map $x$ onto a high-dimension space $x \Rightarrow \phi(x)$. Learn a model $f(x) = \theta^T \phi(x)$, so that $f$ is non-linear function of $x$. 

\begin{figure}
\centering
\includegraphics[width=\textwidth]{kernel_regression_figure.png}
\end{figure}
Kernel regression review

\( \phi(x) \) is expensive to compute. Reformulate the algorithm such that only inner product enters

\[
k(x, x') = \langle \phi(x), \phi(x') \rangle.
\]
Kernel regression review

3. $\phi(x)$ is expensive to compute. Reformulate the algorithm such that only inner product enters

$$k(x, x') = \langle \phi(x), \phi(x') \rangle.$$

4. $k$ is the kernel function and should be easy to compute.

Examples:

- Polynomial kernel: $k(x, x') = (1 + x^T x')^d$.
- Gaussian kernel: $k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$.
- Sobolev kernel in $\mathbb{R}^1$: $k(x, x') = 1 + \min(x, x')$. 

Kernel ridge regression

Given finite samples \((x_1, y_1), \ldots, (x_N, y_N)\), minimize the empirical risk

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

as the estimator of \(f^* = \arg\min_{f \in \mathcal{H}} \mathbb{E}[(f(x) - y)^2]\).
Kernel ridge regression

Given finite samples \((x_1, y_1), \ldots, (x_N, y_N)\), minimize the empirical risk

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

as the estimator of \(f^* = \arg\min_{f \in \mathcal{H}} \mathbb{E}[(f(x) - y)^2]\).

This minimization problem has a closed-form solution:

\[
\hat{f} = \sum_{i=1}^{N} \alpha_i k(x_i, \cdot), \quad \text{where} \quad \alpha = (K + \lambda NI)^{-1} y.
\]

\(K\) is the \(N \times N\) kernel matrix defined by \(K_{ij} = k(x_i, x_j)\).
Think about large datasets

The matrix inversion $\alpha = (K + \lambda NI)^{-1} y$ takes $O(N^3)$ time and $O(N^2)$ memory space, which can be prohibitively expensive when $N$ is large.
Think about large datasets

The matrix inversion $\alpha = (K + \lambda NI)^{-1} y$ takes $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory space, which can be prohibitively expensive when $N$ is large.

Fast approaches to compute kernel ridge regression:

1. Low-rank matrix approximation:
   - Kernel PCA [SSM98]
   - Incomplete Cholesky decomposition [FS02]
   - Nystrom sampling [WS01]
Think about large datasets

The matrix inversion $\alpha = (K + \lambda NI)^{-1}y$ takes $O(N^3)$ time and $O(N^2)$ memory space, which can be prohibitively expensive when $N$ is large.

Fast approaches to compute kernel ridge regression:

1. Low-rank matrix approximation:
   - Kernel PCA [SSM98]
   - Incomplete Cholesky decomposition [FS02]
   - Nystrom sampling [WS01]

\[ K \approx \Phi \times \Phi' \]
Think about large datasets

The matrix inversion $\alpha = (K + \lambda NI)^{-1} y$ takes $O(N^3)$ time and $O(N^2)$ memory space, which can be prohibitively expensive when $N$ is large.

Fast approaches to compute kernel ridge regression:

1. Low-rank matrix approximation:
   - Kernel PCA [SSM98]
   - Incomplete Cholesky decomposition [FS02]
   - Nystrom sampling [WS01]

2. Iterative optimization algorithm:
   - Gradient descent [YRC07...]
   - Conjugate gradient methods [BK10]
Think about large datasets

The matrix inversion $\alpha = (K + \lambda NI)^{-1} y$ takes $\mathcal{O}(N^3)$ time and $\mathcal{O}(N^2)$ memory space, which can be prohibitively expensive when $N$ is large.

Fast approaches to compute kernel ridge regression:

1. Low-rank matrix approximation:
   - Kernel PCA [SSM98]
   - Incomplete Cholesky decomposition [FS02]
   - Nystrom sampling [WS01]

2. Iterative optimization algorithm:
   - Gradient descent [YRC07...]
   - Conjugate gradient methods [BK10]

Accuracy of approximate methods vs. exact algorithm is unknown.
Our main idea

Only keep the diagonal blocks, so that the matrix inversion is fast.

\[
K = \begin{pmatrix}
K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\
K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\
K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\
K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} \\
K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} \\
K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66}
\end{pmatrix}
\]
Our main idea

Only keep the diagonal blocks, so that the matrix inversion is fast.

\[ K = \begin{pmatrix}
  K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\
  K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\
  K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\
  K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} \\
  K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} \\
  K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66}
\end{pmatrix} \Rightarrow

\begin{pmatrix}
  K_{33} & K_{36} & K_{34} & K_{32} & K_{31} & K_{35} \\
  K_{63} & K_{66} & K_{64} & K_{62} & K_{61} & K_{65} \\
  K_{43} & K_{46} & K_{44} & K_{42} & K_{41} & K_{45} \\
  K_{23} & K_{26} & K_{24} & K_{22} & K_{21} & K_{25} \\
  K_{13} & K_{16} & K_{14} & K_{12} & K_{11} & K_{15} \\
  K_{53} & K_{56} & K_{54} & K_{52} & K_{61} & K_{55}
\end{pmatrix}

Random Shuffle
Our main idea

Only keep the diagonal blocks, so that the matrix inversion is fast.

\[ K = \begin{pmatrix}
  K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\
  K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\
  K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\
  K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} \\
  K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} \\
  K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66}
\end{pmatrix} \Rightarrow \]

\[ \begin{pmatrix}
  K_{33} & K_{36} & K_{34} & K_{32} & K_{31} & K_{35} \\
  K_{63} & K_{66} & K_{64} & K_{62} & K_{61} & K_{65} \\
  K_{43} & K_{46} & K_{44} & K_{42} & K_{41} & K_{45} \\
  K_{23} & K_{26} & K_{24} & K_{22} & K_{21} & K_{25} \\
  K_{13} & K_{16} & K_{14} & K_{12} & K_{11} & K_{15} \\
  K_{53} & K_{56} & K_{54} & K_{52} & K_{51} & K_{55}
\end{pmatrix} \Rightarrow \begin{pmatrix}
  K_{33} & K_{36} & 0 & 0 & 0 & 0 \\
  K_{63} & K_{66} & 0 & 0 & 0 & 0 \\
  0 & 0 & K_{44} & K_{42} & 0 & 0 \\
  0 & 0 & K_{24} & K_{22} & 0 & 0 \\
  0 & 0 & 0 & 0 & K_{11} & K_{15} \\
  0 & 0 & 0 & 0 & K_{61} & K_{55}
\end{pmatrix} \]

Random Shuffle \hspace{1cm} \text{Block Diagonalize}
Fast Kernel Ridge Regression (Fast-KRR)

Divide-and-conquer approach:

1. Divide samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \) uniformly at random into the \( m \) disjoint subsets \( S_1, \ldots, S_m \).
Fast Kernel Ridge Regression (Fast-KRR)

Divide-and-conquer approach:

1. Divide samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \) uniformly at randomly into the \( m \) disjoint subsets \( S_1, \ldots, S_m \).

2. For each \( i = 1, 2, \ldots, m \), compute the \textit{local KRR estimate}

\[
\hat{f}_i := \arg\min_{f \in \mathcal{H}} \left\{ \frac{1}{|S_i|} \sum_{(x,y) \in S_i} (f(x) - y)^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\},
\]

under-regularized local risk
Fast Kernel Ridge Regression (Fast-KRR)

Divide-and-conquer approach:

1. Divide samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \) uniformly at randomly into the \( m \) disjoint subsets \( S_1, \ldots, S_m \).

2. For each \( i = 1, 2, \ldots, m \), compute the local KRR estimate

\[
\hat{f}_i := \arg\min_{f \in \mathcal{H}} \left\{ \frac{1}{|S_i|} \sum_{(x, y) \in S_i} (f(x) - y)^2 + \lambda \|f\|_H^2 \right\}.
\]

3. Average together the local estimates \( \bar{f} = \frac{1}{m} \sum_{i=1}^{m} \hat{f}_i \).

Time: \( O(N^3) \) \( \Rightarrow O(N^3/m^2) \)

Space: \( O(N^2) \) \( \Rightarrow O(N^2/m^2) \).
Fast Kernel Ridge Regression (Fast-KRR)

Divide-and-conquer approach:

1. Divide samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \) uniformly at randomly into the \( m \) disjoint subsets \( S_1, \ldots, S_m \).

2. For each \( i = 1, 2, \ldots, m \), compute the local KRR estimate

\[
\hat{f}_i := \arg\min_{f \in \mathcal{H}} \left\{ \frac{1}{|S_i|} \sum_{(x, y) \in S_i} (f(x) - y)^2 + \lambda \|f\|_{\mathcal{H}}^2 \right\}.
\]

3. Average together the local estimates \( \bar{f} = \frac{1}{m} \sum_{i=1}^m \hat{f}_i \).

Time: \( \mathcal{O}(N^3) \Rightarrow \mathcal{O}(N^3/m^2) \)

Space: \( \mathcal{O}(N^2) \Rightarrow \mathcal{O}(N^2/m^2) \).
Theoretical result

**Theorem**

With $m$ splits, Fast-KRR achieves the mean square error:

\[
\mathbb{E}[\|\bar{f} - f^*\|_2^2] \leq C \left( \lambda \|f^*\|_{\mathcal{H}}^2 + \frac{\gamma(\lambda)}{N} + \exp\left(-\frac{c \cdot N/m}{\gamma^2(\lambda)}\right) \right)
\]

\(\gamma(\lambda)\) is the effective dimensionality: let \(\mu_1 \geq \mu_2 \geq \ldots\) be the sequence of eigenvalues in kernel \(k\)'s eigen-expansion, then

\[
\gamma(\lambda) = \sum_{k=1}^{\infty} \frac{\mu_k}{\lambda + \mu_k}.
\]

\(\exp\left(-\frac{c \cdot N/m}{\gamma^2(\lambda)}\right)\) is negligibly small when \(m \ll N/\gamma^2(\lambda)\).

We assume \(\|f^*\|_{\mathcal{H}} < \infty\). There is an “oracle” extension to this result.
Theoretical result

Theorem

With $m$ splits, Fast-KRR achieves the mean square error:

$$\mathbb{E}[\|\Bar{f} - f^*\|^2_2] \leq C \left( \lambda \|f^*\|^2_{\mathcal{H}} + \frac{\gamma(\lambda)}{N} + \exp \left( - \frac{c \cdot N/m}{\gamma^2(\lambda)} \right) \right)$$

- $\gamma(\lambda)$ is the effective dimensionality: let $\mu_1 \geq \mu_2 \geq \ldots$ be the sequence of eigenvalues in kernel $k$'s eigen-expansion, then

$$\gamma(\lambda) = \sum_{k=1}^{\infty} \frac{\mu_k}{(\lambda + \mu_k)}.$$ 

We assume $\|f^*\|_{\mathcal{H}} < \infty$. There is an “oracle” extension to this result.
Theoretical result

**Theorem**

With $m$ splits, Fast-KRR achieves the mean square error:

$$
\mathbb{E}[\|\bar{f} - f^*\|_2^2] \leq C \left( \frac{\lambda \|f^*\|_{\mathcal{H}}^2}{N} + \frac{\gamma(\lambda)}{N} + \exp\left(-\frac{c \cdot N/m}{\gamma^2(\lambda)}\right) \right)
$$

- $\gamma(\lambda)$ is the **effective dimensionality**: let $\mu_1 \geq \mu_2 \geq \ldots$ be the sequence of eigenvalues in kernel $k$'s eigen-expansion, then

  $$
  \gamma(\lambda) = \sum_{k=1}^{\infty} \frac{\mu_k}{(\lambda + \mu_k)}.
  $$

- $\exp(-\frac{c \cdot N/m}{\gamma^2(\lambda)})$ is negligibly small when $m \lesssim N/\gamma^2(\lambda)$.

---

We assume $\|f^*\|_{\mathcal{H}} < \infty$. There is an “oracle” extension to this result.
Corollary

For polynomial kernel if $m \leq cN / \log N$ then

$$\mathbb{E}[\|\bar{f} - f^*\|_2^2] = O\left(\frac{1}{N}\right)$$

(minimax optimal rate)

Time: $O(N^3) \Rightarrow O(N \log^2 N)$
Space: $O(N^2) \Rightarrow O(\log^2 N)$
Apply to specific kernels

For polynomial kernel if $m \leq cN/\log N$ then

$$\mathbb{E}[\|\bar{f} - f^*\|_2^2] = O\left(\frac{1}{N}\right)$$  
(minimax optimal rate)

Time: $O(N^3) \Rightarrow O(N \log^2 N)$  
Space: $O(N^2) \Rightarrow O(\log^2 N)$

For Gaussian kernel, if $m \leq cN/\log^2 N$ then

$$\mathbb{E}[\|\bar{f} - f^*\|_2^2] = O\left(\frac{\sqrt{\log N}}{N}\right)$$  
(minimax optimal rate)

Time: $O(N^3) \Rightarrow O(N \log^4 N)$  
Space: $O(N^2) \Rightarrow O(\log^4 N)$
Apply to specific kernels

**Corollary**

For Sobolev kernel of smoothness \( \nu \), if \( m \leq cN^{\frac{2\nu - 1}{2\nu + 1}} / \log N \) then

\[
\mathbb{E}[\| \bar{f} - f^* \|^2] = \mathcal{O}\left(N^{-\frac{2\nu}{2\nu + 1}}\right) \quad \text{(minimax optimal rate)}
\]

Time: \( \mathcal{O}(N^3) \Rightarrow \mathcal{O}(N^{\frac{2\nu + 5}{2\nu + 1}} \log^2 N) \)

Space: \( \mathcal{O}(N^2) \Rightarrow \mathcal{O}(N^{\frac{4}{2\nu + 1}} \log^2 N) \)
Simulation Study

Data \((x, y)\) is generated by \(y = \min(x, 1 - x) + \epsilon\) where \(\epsilon \sim N(0, 0.2)\).
Compare Fast-KRR and exact KRR

We use a Sobolev kernel of smoothness-1 to fit the data.
Compare Fast-KRR and exact KRR

We use a Sobolev kernel of smoothness-1 to fit the data.

Fast-KRR’s performance is very close to exact KRR for $m \leq 16$. 
Threshold for data partitioning

Mean square error is plotted for varied choices of $m$. As long as $m \ll N^{0.45}$, the accuracy is not hurt.
Threshold for data partitioning

Mean square error is plotted for varied choices of $m$.

As long as $m \lesssim N^{0.45}$, the accuracy is not hurt.
Summary

- Divide-and-conquer approach for kernel ridge regression reduces time and space complexity.
- Optimal accuracy is retained.
- Polynomially time complexity in for Sobolev space kernels, nearly linear complexity for finite-rank kernels and Gaussian kernels.
Open Problems

- How to choose optimal partition number in practice?
- Lower bound for sub-sampling rate?
- Does divide-and-conquer work for other kernel-based methods?