Divide and Conquer Kernel Ridge Regression

Yuchen Zhang  John Duchi  Martin Wainwright

University of California, Berkeley

COLT 2013
Problem set-up

**Goal** Solve the following problem:

\[
\begin{align*}
\text{minimize} & \quad \mathbb{E}[(f(x) - y)^2] \\
\text{subject to} & \quad f \in \mathcal{H}
\end{align*}
\]

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

**Goal** Solve the following problem:

\[
\begin{align*}
\text{minimize} \quad & \mathbb{E}[(f(x) - y)^2] \\
\text{subject to} \quad & f \in \mathcal{H}
\end{align*}
\]

where \((x, y)\) is sampled from joint distribution \(\mathbb{P}\), and \(\mathcal{H}\) is a Reproducing Kernel Hilbert Space (RKHS).

\(\mathcal{H}\) is defined by a kernel function \(k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}\). Formally:

\[
\mathcal{H} = \{ f : f = \sum_{i=1}^{\infty} \alpha_i k(x_i, \cdot), \; x_i \in \mathcal{X} \}.
\]
Kernel trick review

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

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

2. For non-linear problems, the trick is to map $x$ onto a high-dimension feature space $x \Rightarrow \phi(x)$, then learn a model $f(x) = \theta^T \phi(x)$, so that $f$ is a non-linear function of $x$. 
Kernel trick review

Usually, $\phi(x)$ is an infinite-dimensional vector or it is expensive to compute. We can reformulate the learning algorithm such that the input vector enters only in the form of inner product

$$k(x, x') = \phi(x)^T \phi(x').$$
Kernel trick review

3 Usualy, $\phi(x)$ is a infinite-dimensional vector or it is expensive to compute. We can reformulate the learning algorithm such that the input vector enters only in the form of inner product

$$k(x, x') = \phi(x)^T \phi(x').$$

4 $k$ is called 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 $N$ samples $(x_1, y_1), \ldots, (x_N, y_N)$, we want to compute the empirical minimizer

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

as an estimate to $f^* = \arg\min_{f \in \mathcal{H}} \mathbb{E}[(f(x) - y)^2]$. 

Kernel ridge regression

Given \( N \) samples \((x_1, y_1), \ldots, (x_N, y_N)\), we want to compute the empirical minimizer

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

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

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 $\alpha = (K + \lambda N I)^{-1} y$ 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.
   - Incomplete Cholesky decomposition
   - Nystrom sampling.

2. Iterative optimization algorithm:
   - Gradient descent.
   - Conjugate gradient methods.
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.
   - Incomplete Cholesky decomposition
   - Nystrom sampling.

2. **Iterative optimization algorithm:**
   - Gradient descent.
   - Conjugate gradient methods.

However, none of these method can be shown achieving the same level of accuracy as the exact algorithm does.
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.

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

Random Shuffle  Block Diagonalize
Fast kernel ridge regression (Fast-KRR)

We propose a divide-and-conquer approach:

1. Divide the set of samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \) evenly and uniformly at randomly into the \( m \) disjoint subsets:

\[
S_1, \ldots, S_m \subset \mathcal{X} \times \mathbb{R}.
\]

Computation time: \( O\left(\frac{N^3}{m^2}\right) \); memory space: \( O\left(\frac{N^2}{m^2}\right) \).
Fast kernel ridge regression (Fast-KRR)

We propose a divide-and-conquer approach:

1. Divide the set of samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \) evenly and uniformly at randomly into the \( m \) disjoint subsets:

\[
S_1, \ldots, S_m \subset \mathcal{X} \times \mathbb{R}.
\]

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

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

\( \lambda \|f\|^2_{\mathcal{H}} \) under-regularized

local risk
Fast kernel ridge regression (Fast-KRR)

We propose a divide-and-conquer approach:

1. Divide the set of samples \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \) evenly and uniformly at randomly into the \( m \) disjoint subsets:

   \[ S_1, \ldots, S_m \subset \mathcal{X} \times \mathbb{R}. \]

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

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

3. Average together the local estimates and output \( \bar{f} = \frac{1}{m} \sum_{i=1}^{m} \hat{f}_i \).
Fast kernel ridge regression (Fast-KRR)

We propose a divide-and-conquer approach:

1. Divide the set of samples \( \{ (x_1, y_1), \ldots, (x_N, y_N) \} \) evenly and uniformly at randomly into the \( m \) disjoint subsets:

   \[
   S_1, \ldots, S_m \subset X \times \mathbb{R}.
   \]

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

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

   (under-regularized local risk)

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

   Computation time: \( \mathcal{O}(N^3/m^2) \); memory space: \( \mathcal{O}(N^2/m^2) \).
Theoretical result

**Theorem**

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

$$
\mathbb{E}[\|\bar{f} - f^*\|_2^2] \leq C \left( \lambda \|f^*\|_H^2 + \gamma(\lambda) \frac{N}{N} \right) + T(\lambda, m)
$$

where $\lambda$ is the regularization parameter, $f^*$ is the true function, $\gamma(\lambda)$ is the effective dimensionality, $\lambda \|f^*\|_H$ represents the squared bias, and $N$ is the sample size. The term $T(\lambda, m)$ becomes a higher-order negligible term when $m$ is below the threshold $m^* \simeq N/\gamma(\lambda)$. $\gamma(\lambda)$ is the effective dimensionality: let $\mu_1 \geq \mu_2 \geq \ldots$ be the sequence of eigenvalues in kernel $k$'s eigen-expansion, then $\gamma(\lambda) = \sum_{k=1}^{\infty} \frac{\mu_k}{\lambda + \mu_k}$. 

Yuchen Zhang (UC Berkeley)
Theoretical result

Theorem

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

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

- $\lambda \|f^*\|_{\mathcal{H}}^2$ is the bias introduced by regularization.

$\mathcal{H}$ is the bias introduced by regularization.

$T(\lambda, m)$ becomes a higher-order negligible term when $m$ is below the threshold $m^* \approx \frac{N}{\gamma(\lambda)}$. 

$\gamma(\lambda)$ is the effective dimensionality: let $\mu_1 \geq \mu_2 \geq ...$ be the sequence of eigenvalues in kernel $k$'s eigen-expansion, then $\gamma(\lambda) = \sum_{k=1}^{\infty} \frac{\mu_k}{\lambda + \mu_k}$. 

Theoretical result

**Theorem**

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

$$
\mathbb{E}[\|\bar{f} - f^*\|_2^2] \leq C \left( \lambda \|f^*\|_{\mathcal{H}}^2 \text{ squared bias} + \frac{\gamma(\lambda)}{N} \text{ variance} \right) + T(\lambda, m)
$$

- $\lambda \|f^*\|_{\mathcal{H}}^2$ is the bias introduced by regularization.
- $T(\lambda, m)$ becomes a higher-order negligible term when $m$ is below the threshold $m^* \approx N/\gamma^2(\lambda)$. 

Yuchen Zhang (UC Berkeley)  
Divide and Conquer KRR  
COLT 2013
Theoretical result

**Theorem**

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

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

- $\lambda \|f^*\|_{\mathcal{H}}^2$ is the bias introduced by regularization.
- $T(\lambda, m)$ becomes a higher-order negligible term when $m$ is below the threshold $m^* \approx N/\gamma^2(\lambda)$.
- $\gamma(\lambda)$ is the **effective dimensionality**: let $\mu_1 \geq \mu_2 \geq \ldots$ be the sequence of eigenvalues in kernel $k$’s eigen-expansion, then

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

**Theorem**

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

$$ \mathbb{E} [ \| \tilde{f} - f^* \|_2^2 ] \leq C \left( \lambda \| f^* \|_\mathcal{H}^2 + \frac{\gamma(\lambda)}{N} \right) + T(\lambda, m) $$

- $\lambda \| f^* \|_\mathcal{H}^2$ is the bias introduced by regularization.
- $T(\lambda, m)$ becomes a higher-order negligible term when $m$ is below the threshold $m^* \approx N/\gamma^2(\lambda)$.
- $\gamma(\lambda)$ is the *effective dimensionality*: let $\mu_1 \geq \mu_2 \geq \ldots$ be the sequence of eigenvalues in kernel $k$'s eigen-expansion, then

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

<table>
<thead>
<tr>
<th>Corollary</th>
</tr>
</thead>
<tbody>
<tr>
<td>For polynomial kernel if $m \leq cN/\log N$ then</td>
</tr>
</tbody>
</table>

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

Time: $\mathcal{O}(N^3) \Rightarrow \mathcal{O}(N \log^2 N)$  \hspace{1cm} Space: $\mathcal{O}(N^2) \Rightarrow \mathcal{O}(\log^2 N)$
Apply to specific kernels

\textbf{Corollary}

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

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

Time: \( \mathcal{O}(N^3) \Rightarrow \mathcal{O}(N \log^2 N) \)  
Space: \( \mathcal{O}(N^2) \Rightarrow \mathcal{O}(\log^2 N) \)

\textbf{Corollary}

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

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

Time: \( \mathcal{O}(N^3) \Rightarrow \mathcal{O}(N \log^4 N) \)  
Space: \( \mathcal{O}(N^2) \Rightarrow \mathcal{O}(\log^4 N) \)
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

\[
\mathbb{E}[\| \tilde{f} - f^* \|^2_2] = \mathcal{O}\left(N^{-\frac{2\nu}{2\nu + 1}}\right) \quad \text{(minimax optimal rate)}
\]

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 \(y = \min(x, 1 - x) + \varepsilon\) where \(\varepsilon \sim N(0, 0.2)\).
Compare Fast-KRR and exact KRR

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

Divide and Conquer KRR
COLT 2013 13 / 15
Compare Fast-KRR and exact KRR

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

<table>
<thead>
<tr>
<th>Total number of samples (N)</th>
<th>Mean square error</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>10^{-4}</td>
</tr>
<tr>
<td>512</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>1024</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>2048</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>4096</td>
<td>10^{-3}</td>
</tr>
<tr>
<td>8192</td>
<td>10^{-3}</td>
</tr>
</tbody>
</table>

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 $m \ll N$, 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.
Conclusion

- We propose a divide-and-conquer approach for kernel ridge regression that leads to substantial reduction in computation time and memory space.
Conclusion

- We propose a divide-and-conquer approach for kernel ridge regression that leads to substantial reduction in computation time and memory space.
- The proposed algorithm archives the optimal convergence rate for the full sample size $N$, as long as the partition number $m$ is not too large.
Conclusion

- We propose a divide-and-conquer approach for kernel ridge regression that leads to substantial reduction in computation time and memory space.
- The proposed algorithm archives the optimal convergence rate for the full sample size $N$, as long as the partition number $m$ is not too large.
- As concrete examples, our theory guarantees that $m$ may grow polynomially in $N$ for Sobolev spaces, and grow nearly linearly in $N$ for finite-rank kernels and Gaussian kernels.