Tight Kernel Query Complexity of Kernel Ridge Regression and Kernel k-means Clustering

Manuel Fernández V, David P. Woodruff, Taisuke Yasuda

Overview

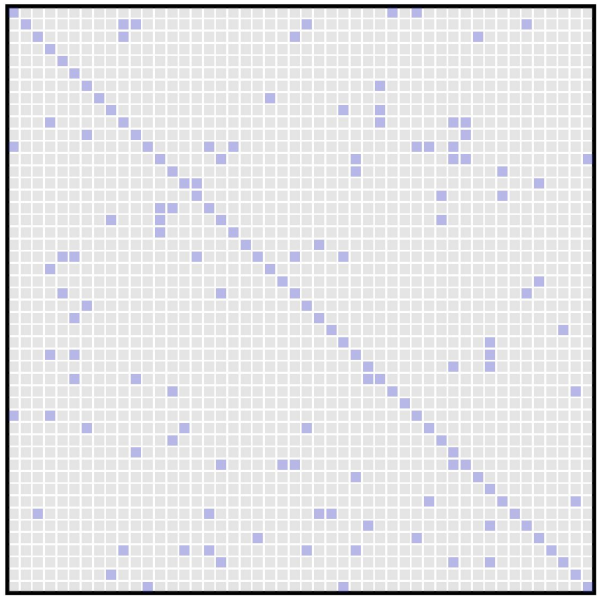
- Preliminaries
- Kernel ridge regression
- Kernel k-means clustering
- Query-efficient algorithm for mixtures of Gaussians

Kernel Method

- Many machine learning tasks can be expressed as a function of the inner product matrix $\mathbf{G}$ of the data points (rather than the design matrix)
- Implicitly apply the exact same algorithm to the data set under a feature map through the use of a *kernel function*
- The analogue of the inner product matrix $\mathbf{G}$ is called the *kernel matrix* $\mathbf{K}$

Kernel Query Complexity

- In this work, we study *kernel query complexity*: the number of entries of the kernel matrix $\mathbf{K}$ read



Kernel Ridge Regression (KRR)

- Kernel method applied to ridge regression

$$\begin{aligned}\boldsymbol{\alpha}_{\text{opt}} &= \underset{\boldsymbol{\alpha}\in\mathbb{R}^n}{\operatorname{argmin}} \|\mathbf{K}\boldsymbol{\alpha} - \mathbf{z}\|_2^2 + \lambda \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} \\ &= (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{z}\end{aligned}$$

- Approximation guarantee

$$\|\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}_{\text{opt}}\|_2 \leq \varepsilon \|\boldsymbol{\alpha}_{\text{opt}}\|_2$$

Query-Efficient Algorithms

- State of the art approximation algorithms have *sublinear* and *data-dependent* runtime and query complexity (Musco and Musco NeurIPS 2017, El Alaoui and Mahoney NeurIPS 2015)
- Sample $\tilde{O}(d_{\text{eff}}^{\lambda}/\varepsilon)$ rows proportionally to ridge leverage scores where

$$d_{\text{eff}}^{\lambda}(\mathbf{K}) := \operatorname{tr}\left(\mathbf{K}(\mathbf{K}+\lambda\mathbf{I}_n)^{-1}\right) = \sum_{i=1}^{r} \frac{\sigma_i^2}{\sigma_i^2+\lambda}$$

- Query complexity $\tilde{O}(nd_{\text{eff}}^{\lambda}/\varepsilon)$

Contribution 1: Tight Lower Bounds for KRR

Theorem (informal)

Any randomized algorithm computing a $(1+\varepsilon)$*-approximate KRR solution with probability at least 2/3 makes at least* $\Omega(nd_{\text{eff}}^{\lambda}/\varepsilon)$ *kernel queries.*

- Effective against randomized and adaptive (data-dependent) algorithms
- Tight up to logarithmic factors

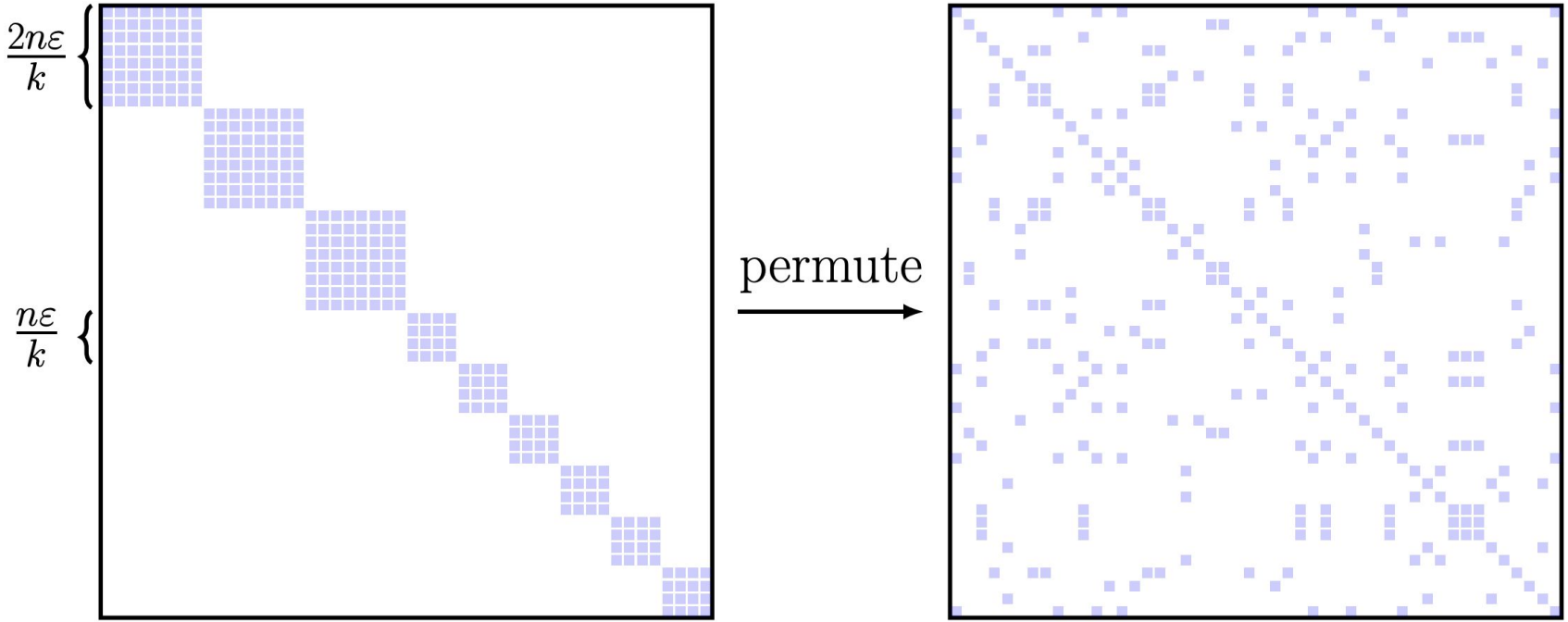
Contribution 1: Tight Lower Bounds for KRR

Proof (sketch)

- By Yao's minimax principle, suffices to prove for deterministic algorithms on a hard input distribution
- Our hard input distribution: all ones vector for the target vector $\mathbf{z}$, regularization $\lambda = n/k$

Contribution 1: Tight Lower Bounds for KRR

- Data distribution μ_{KRR} for the kernel matrix:



Contribution 1: Tight Lower Bounds for KRR

- Inner product matrix of standard basis vectors, $2n\varepsilon/k$ copies of $\mathbf{e}_j$ for the first $k/4\varepsilon$ coordinates, and $n\varepsilon/k$ copies of the next $k/2\varepsilon$
- Half of the data points belong to “large clusters”, the other half belong to “small clusters”
- In order to label a row as “large cluster” or “small cluster”, any algorithm must read $\Omega(k/\varepsilon)$ entries of the row
- In order to label a constant fraction of rows, need to read $\Omega(nk/\varepsilon)$ entries of the kernel matrix

Contribution 1: Tight Lower Bounds for KRR

Lemma

Any randomized algorithm for labeling a constant fraction of rows of a kernel matrix drawn from μ_{KRR} *must read* $\Omega(nk/\varepsilon)$ *kernel entries.*

- Proven using standard techniques

Contribution 1: Tight Lower Bounds for KRR

Reduction

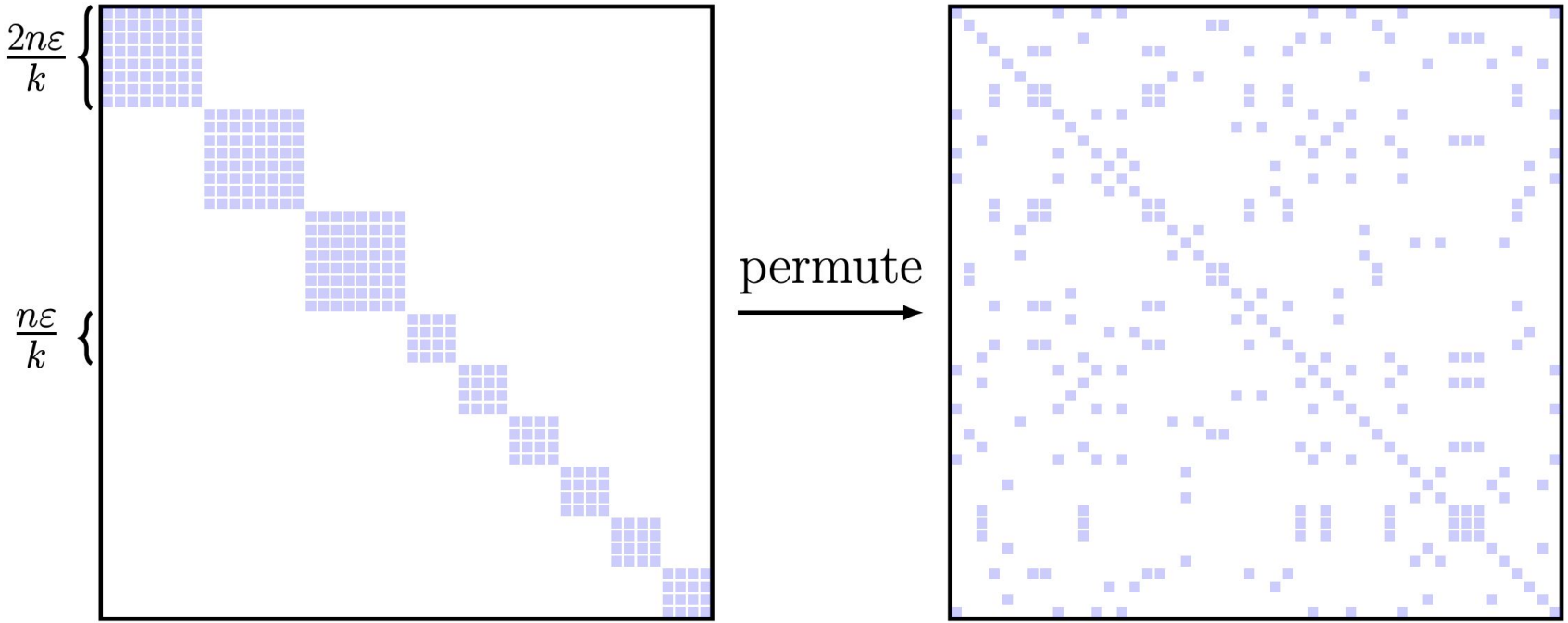
Main Idea: one can just read off the labels of all the rows from the optimal KRR solution, and one can do this for a constant fraction of the rows from an approximate KRR solution.

Contribution 1: Tight Lower Bounds for KRR

- Let $\mathbf{K} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^\top$ be the SVD of the kernel matrix
- The columns are the eigenvectors of $\mathbf{K}$ and the cluster size n_j is the corresponding eigenvalue, and these are orthogonal
- The target vector is the sum of these columns

$$\mathbf{z} = \sum_{j \in [3J/4]} \sqrt{n_j}\mathbf{U}\mathbf{e}_j$$

Contribution 1: Tight Lower Bounds for KRR



Contribution 1: Tight Lower Bounds for KRR

Optimal KRR solution

$$\boldsymbol{\alpha}_{\text{opt}} = (\mathbf{K} + \lambda \mathbf{I}_n)^{-1}\mathbf{z}$$

$$= \sum_{j \in [3J/4]} \frac{1}{n_j + \lambda} \left(\sqrt{n_j}\mathbf{U}\mathbf{e}_j\right)$$

Contribution 1: Tight Lower Bounds for KRR

Optimal KRR solution

$$\mathbf{e}_i^\top \boldsymbol{\alpha}_{\text{opt}} = \begin{cases} (2n\varepsilon/k + n/k)^{-1} = \frac{k/n}{1+2\varepsilon} & \text{if row } i \text{ has block size } 2n\varepsilon/k \\ (n\varepsilon/k + n/k)^{-1} = \frac{k/n}{1+\varepsilon} & \text{if row } i \text{ has block size } n\varepsilon/k \end{cases}$$

Thus, the entries are separated by a multiplicative $(1 \pm \Omega(\varepsilon))$ factor.

Contribution 1: Tight Lower Bounds for KRR

Approximate KRR solution

- By averaging the approximation guarantee over the coordinates, we can still distinguish the cluster sizes for a constant fraction of the coordinates

$$\left\|\hat{\boldsymbol{\alpha}} - \boldsymbol{\alpha}_{\mathrm{opt}}\right\|_2 \leq \varepsilon \left\|\boldsymbol{\alpha}_{\mathrm{opt}}\right\|_2$$

Contribution 1: Tight Lower Bounds for KRR

$$d_{\text{eff}}^{\lambda} = \sum_{j \in [3J/4]} \frac{n_j}{n_j + \lambda} = \Theta\left(\sum_{j \in [3J/4]} \frac{n\varepsilon/k}{n\varepsilon/k + n/k} \right) = \Theta(k)$$

Contribution 1: Tight Lower Bounds for KRR

Remarks

- Settles a variant of an open question of El Alaoui and Mahoney: is the effective statistical dimension a lower bound on the query complexity? (they consider an approximation guarantee on the statistical risk instead of the argmin)
- Techniques extend to any *indicator kernel* function, including all kernels that are a function of the inner product or Euclidean distance
- Lower bound is easily modified to an instance where the top d_{eff}^{λ} singular values scales as the regularization λ

Kernel k-means Clustering (KKMC)

- Kernel method applied to k-means clustering
- Objective: a partition of the data set into k clusters that minimizes the sum of squared distances to the nearest centroid
- For a feature map $\varphi : \mathcal{X} \to \mathcal{F}$, objective function is

$$\operatorname{cost}(\mathcal{C}) := \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} \left\| \varphi(\mathbf{x}) - \boldsymbol{\mu}_j \right\|_{\mathcal{F}}^2$$

$$\boldsymbol{\mu}_j := \tfrac{1}{|C_j|} \textstyle\sum_{\mathbf{x} \in C_j} \varphi(\mathbf{x})$$

Contribution 2: Tight Lower Bounds for KKMC

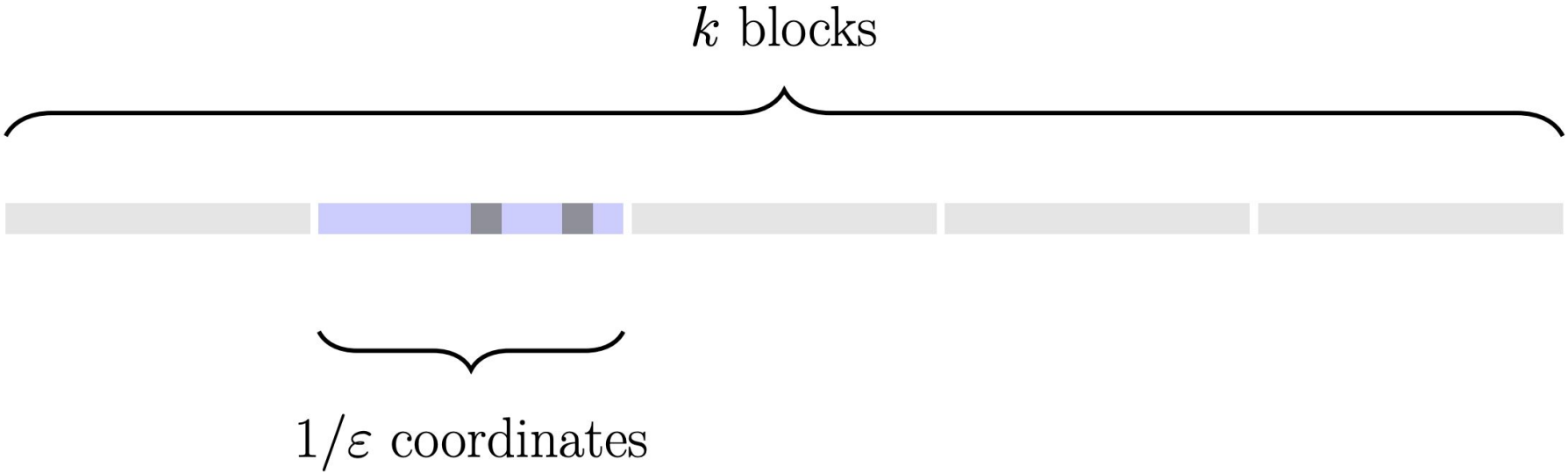
Theorem (informal)

Any randomized algorithm computing a $(1+\varepsilon)$*-approximate KKMC solution with probability at least 2/3 makes at least* $\Omega(nk/\varepsilon)$ *kernel queries.*

- Effective against randomized and adaptive (data-dependent) algorithms
- Tight up to logarithmic factors

Contribution 2: Tight Lower Bounds for KKMC

- Similar techniques, hard distribution is sums of standard basis vectors



Kernel k-means Clustering of Mixtures of Gaussians

- For input distributions encountered in practice, previous lower bound may be pessimistic
- We show that for a mixture of k isotropic Gaussians, we can solve KKMC in only $\tilde{O}(n/\varepsilon)$ kernel queries

Contribution 3: Query-Efficient Algorithm for Mixtures of Gaussians

Theorem (informal)

Given a mixture of k Gaussians with mean separation $\tilde{O}(\sigma)$, there exists a randomized algorithm which returns a $(1+\varepsilon)$- approximate k-means clustering solution reading $\tilde{O}(n/\varepsilon)$ kernel queries with probability at least 2/3.

Contribution 3: Query-Efficient Algorithm for Mixtures of Gaussians

Proof (sketch)

- Learn the means of the Gaussians in $\mathrm{poly}(k, 1/\varepsilon, d)$ samples (Regev and Vijayaraghavan, FOCS 2017)
- Use the learned means to identify the true means of $O(\log n/\varepsilon)$ Gaussians
- Subtract off Gaussians from the same mean from each other to obtain zero-mean Gaussians
- Use the zero-mean Gaussians to sketch the data set in $O(n \log n/\varepsilon)$ samples
- Cluster the sketched data set