Nonlinear Dimensionality Reduction for Faster Kernel Methods in Machine Learning

Christopher Musco, Massachusetts Institute of Technology

February 27, 2018

RELEVANT PAPER

ICML 2017:

"Random Fourier Features for Kernel Ridge Regression: Approximation Bounds and Statistical Guarantees"

Joint work with:

Haim Avron (TAU)
Michael Kapralov (EPFL)
Cameron Musco (MIT)
Ameya Velingker (EPFL)
Amir Zandieh (EPFL)

Main idea:

Study Fourier kernel approximation methods from a matrix sampling point of view.

OUTLINE

Main idea:

Study Fourier kernel approximation methods from a matrix sampling point of view.

Specifics:

- Analyze Random Fourier Features method (Rahimi, Recht NIPS '07) using techniques based on leverage scores.

OUTLINE

Main idea:

Study Fourier kernel approximation methods from a matrix sampling point of view.

Specifics:

- Analyze Random Fourier Features method (Rahimi, Recht NIPS '07) using techniques based on leverage scores.
- Develop an improved Random Fourier Features method based on this analysis (better in theory and experiments).

OUTLINE

Main idea:

Study Fourier kernel approximation methods from a matrix sampling point of view.

Specifics:

- Analyze Random Fourier Features method (Rahimi, Recht NIPS '07) using techniques based on leverage scores.
- Develop an improved Random Fourier Features method based on this analysis (better in theory and experiments).

Lots of open questions and directions for future work.
Opportunities to combine techniques from randomized linear algebra and Fourier methods.

QUICK REFRESHER ON KERNEL METHODS

Adapt standard linear learning methods (least squares regression, support vector machines, PCA, k-means clustering) to learn nonlinear relationships.

Adapt standard linear learning methods (least squares regression, support vector machines, PCA, k-means clustering) to learn nonlinear relationships.

(theoretically well-understood, multipurpose, widely used)

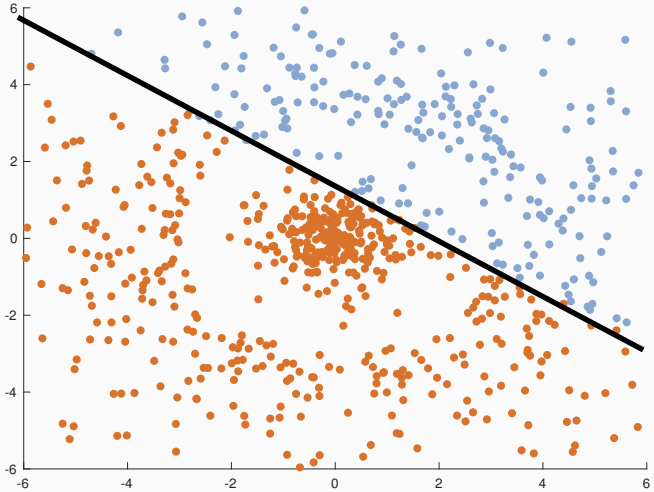
"Lift" data points to a higher dimensional feature space. E.g.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

"Lift" data points to a higher dimensional feature space. E.g.

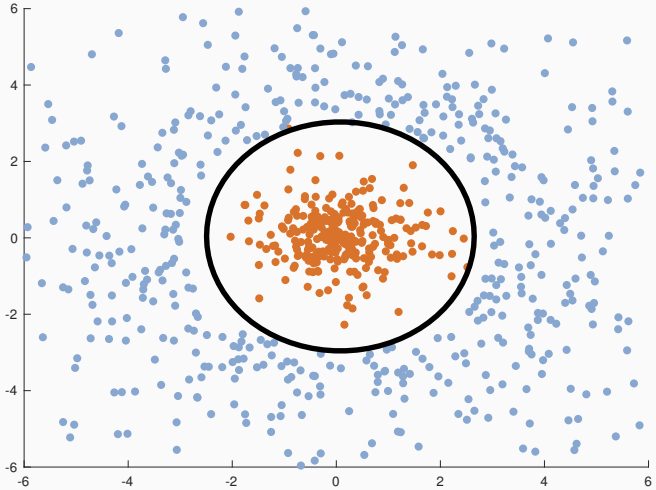
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \quad \Longrightarrow \quad \phi(\mathbf{x}) = \begin{bmatrix} x_1 \\ \vdots \\ x_d \\ x_1 x_1 \\ x_1 x_2 \\ \vdots \\ x_d x_d \end{bmatrix}$$

Linear Classifier



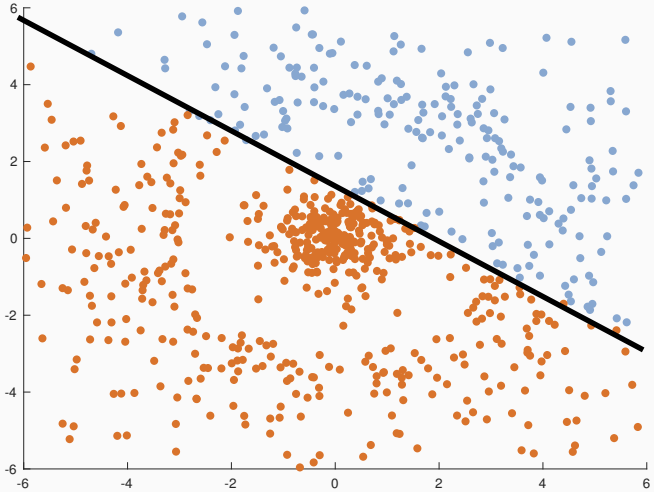
$$x_1 + 2x_2 \geq 6$$

Kernel Classifier



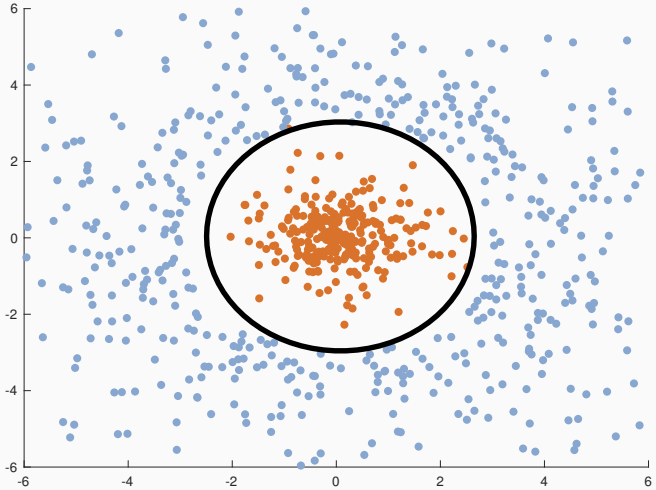
$$x_1^2 + x_2^2 \geq 10$$

Linear Classifier



$$x_1 + 2x_2 \geq 6$$

Kernel Classifier



$$x_1^2 + x_2^2 \geq 10$$

$$\phi(\mathbf{x})_3 + \phi(\mathbf{x})_4 \geq 10$$

THE COST OF KERNEL METHODS

Main computational issue: Forming $\phi(\mathbf{x})$ is intractable even for moderately complex kernels.

E.g. degree q polynomials $\implies$ $O(d^q)$ dimensional vectors.

THE COST OF KERNEL METHODS

Main computational issue: Forming $\phi(\mathbf{x})$ is intractable even for moderately complex kernels.

E.g. degree q polynomials $\implies$ $O(d^q)$ dimensional vectors.

Fix: For common linear learning methods, we only need the kernel dot product for each pair of data points $\mathbf{x}, \mathbf{y}$:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \left\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \right\rangle .$$

THE COST OF KERNEL METHODS

Main computational issue: Forming $\phi(\mathbf{x})$ is intractable even for moderately complex kernels.

E.g. degree q polynomials $\implies$ $O(d^q)$ dimensional vectors.

Fix: For common linear learning methods, we only need the kernel dot product for each pair of data points $\mathbf{x}, \mathbf{y}$:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle .$$

Can often be computed much more quickly that $\phi(\mathbf{x}_i)$, $\phi(\mathbf{x}_j)$.

EXAMPLE: LEAST SQUARES REGRESSION

Input: Data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T$, responses $\mathbf{b} = [b_1, \ldots, b_n]^T$.

EXAMPLE: LEAST SQUARES REGRESSION

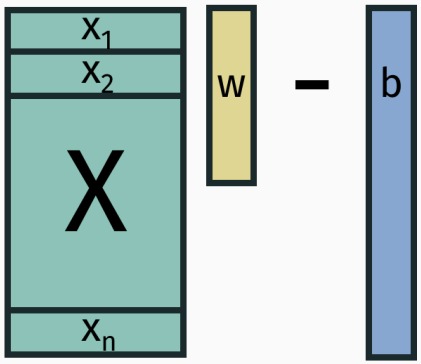
Input: Data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T$, responses $\mathbf{b} = [b_1, \ldots, b_n]^T$.

Solve:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

Predict:

$$b_{new} = \mathbf{w}^{*T} \mathbf{x}_{new}$$



EXAMPLE: LEAST SQUARES REGRESSION

Input: Data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T$, responses $\mathbf{b} = [b_1, \ldots, b_n]^T$.

Solve:

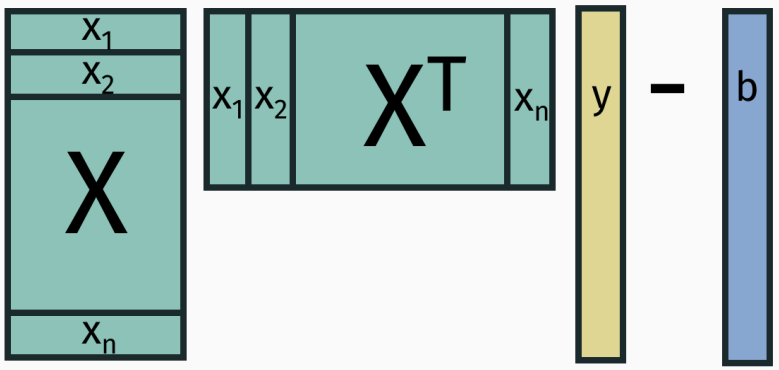
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{w}\|_2^2$$

$$\mathbf{y}^* = \arg\min_{\mathbf{y}} \|\mathbf{X}\mathbf{X}^T\mathbf{y} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{X}^T\mathbf{y}\|_2^2$$

Predict:

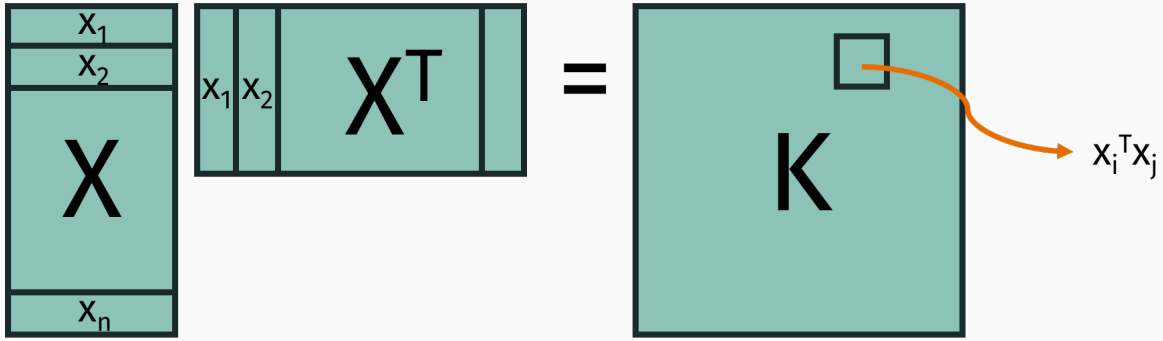
$$b_{new} = \mathbf{w}^{*T}\mathbf{x}_{new}$$

$$b_{new} = \mathbf{y}^{*T}\mathbf{X}\mathbf{x}_{new}$$



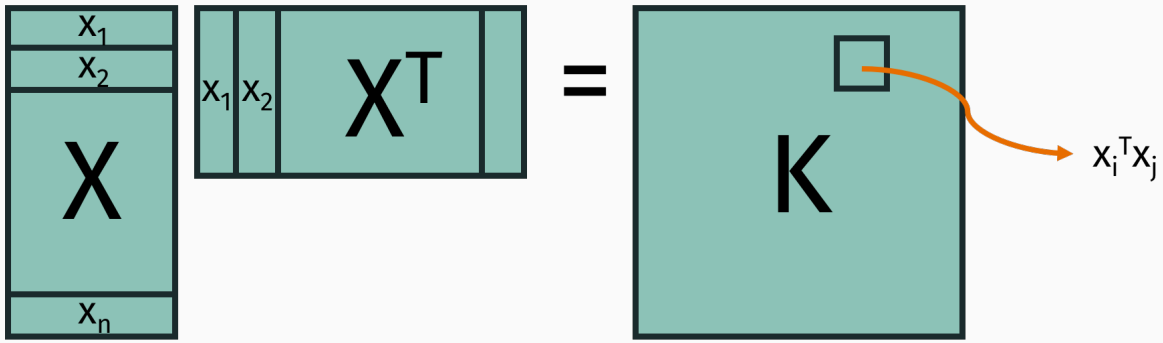
EXAMPLE: LEAST SQUARES REGRESSION

For training, we compute $(\mathbf{K} + \lambda \mathbf{I})^{-1}$ for the kernel matrix $\mathbf{K}$:



EXAMPLE: LEAST SQUARES REGRESSION

For training, we compute $(\mathbf{K} + \lambda \mathbf{I})^{-1}$ for the kernel matrix $\mathbf{K}$:



If we replace each $\mathbf{x}_i$ with $\phi(\mathbf{x}_i)$ for nonlinear learning, we just need to alternatively compute:

$$\mathbf{K}_{i,j} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = k(\mathbf{x}_i, \mathbf{x}_j).$$

THE "KERNEL TRICK"

Kernel dot product can often be computed implicitly without forming $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$. For example:

$$
\begin{aligned}
(1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2 &= (1 + x_1 y_1 + \ldots + x_d y_d)^2 \\
&= (1 + x_1 y_1 + x_1^2 y_1^2 + 2 x_1 y_1 x_2 y_2 + \ldots) \\
&= \left\langle [1, x_1, x_1^2, \sqrt{2} x_1 x_2, \ldots], [1, y_1, y_1^2, \sqrt{2} y_1 y_2, \ldots] \right\rangle \\
&= \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle
\end{aligned}
$$

THE "KERNEL TRICK"

Kernel dot product can often be computed implicitly without forming $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$. For example:

$$
\begin{aligned}
(1 + \langle \mathbf{x}, \mathbf{y} \rangle)^2 &= (1 + x_1 y_1 + \ldots + x_d y_d)^2 \\
&= (1 + x_1 y_1 + x_1^2 y_1^2 + 2 x_1 y_1 x_2 y_2 + \ldots) \\
&= \left\langle [1, x_1, x_1^2, \sqrt{2} x_1 x_2, \ldots], [1, y_1, y_1^2, \sqrt{2} y_1 y_2, \ldots] \right\rangle \\
&= \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle
\end{aligned}
$$

The **kernel function** $k(\mathbf{x}, \mathbf{y}) = \left(1 + \mathbf{x}^T \mathbf{y}\right)^2$ provides an alternative similarity metric to the standard dot product.

THE "KERNEL TRICK"

Kernel learning pipeline for data points $\mathbf{x}_1, \ldots \mathbf{x}_n$:

1. Choose kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$:

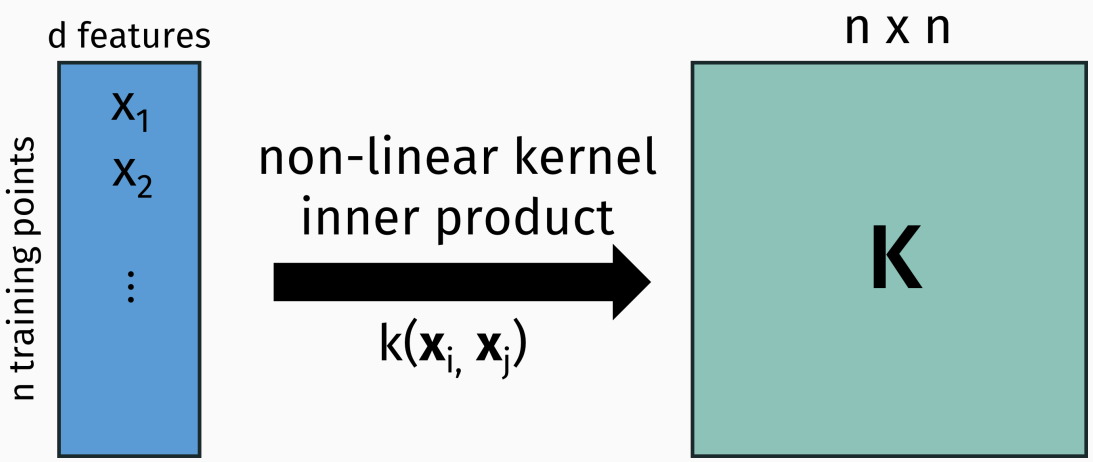
$$(1 + \mathbf{x}_i^T \mathbf{x}_j)^q, \; e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}, \; e^{-\|\mathbf{x}_i - \mathbf{x}_j\|_1}, \; \text{etc.}$$

2. Form $n \times n$ kernel matrix $\mathbf{K}$ with:

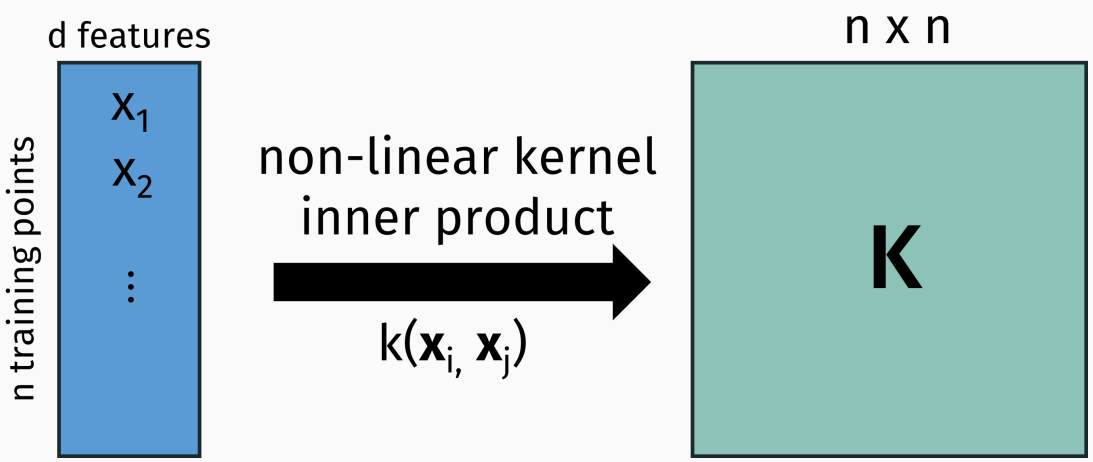
$$\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$$

3. Compute model using $\mathbf{K}$: compute $(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{b}$ for kernel regression, eigendecomposition of $\mathbf{K}$ for kernel PCA, etc.

Even if we avoid explicit feature expansion, kernel methods are slow. Quadratic dependence on the number of data points.

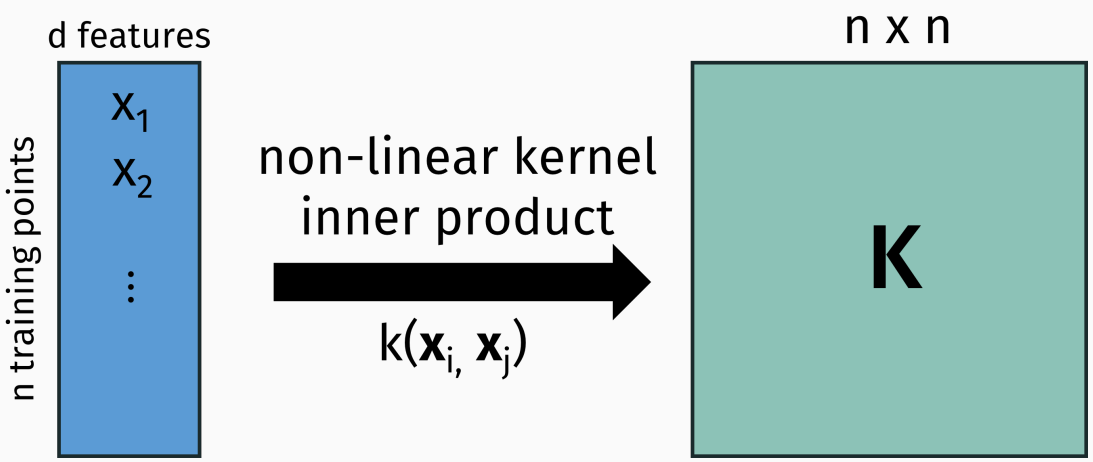


Even if we avoid explicit feature expansion, kernel methods are slow. Quadratic dependence on the number of data points.



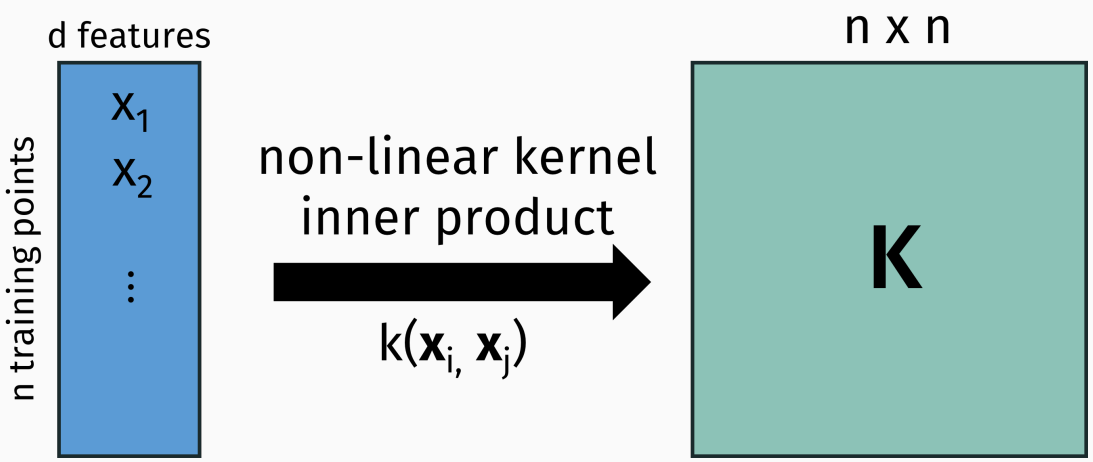
- $n = 100{,}000 \Rightarrow 10$ billion entries $\Rightarrow 80$ GB to store $\mathbf{K}$.

Even if we avoid explicit feature expansion, kernel methods are slow. Quadratic dependence on the number of data points.



- $n = 100{,}000 \Rightarrow 10$ billion entries $\Rightarrow 80$ GB to store $\mathbf{K}$.
- Just writing down $\mathbf{K}$ requires $\Omega(n^2)$ time.

Even if we avoid explicit feature expansion, kernel methods are slow. Quadratic dependence on the number of data points.



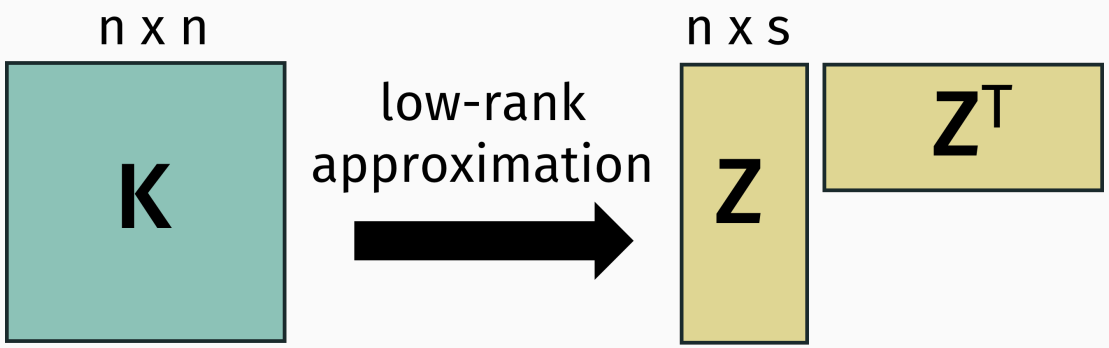
- $n = 100{,}000 \Rightarrow 10$ billion entries $\Rightarrow 80$ GB to store $\mathbf{K}$.
- Just writing down $\mathbf{K}$ requires $\Omega(n^2)$ time.
- Other operations require even more. A single iteration for a linear system solver takes $\Omega(n^2)$ time.

ALGORITHMIC CHALLENGE

New algorithmic ideas are needed to scale kernel methods.
Even for moderately large datasets.

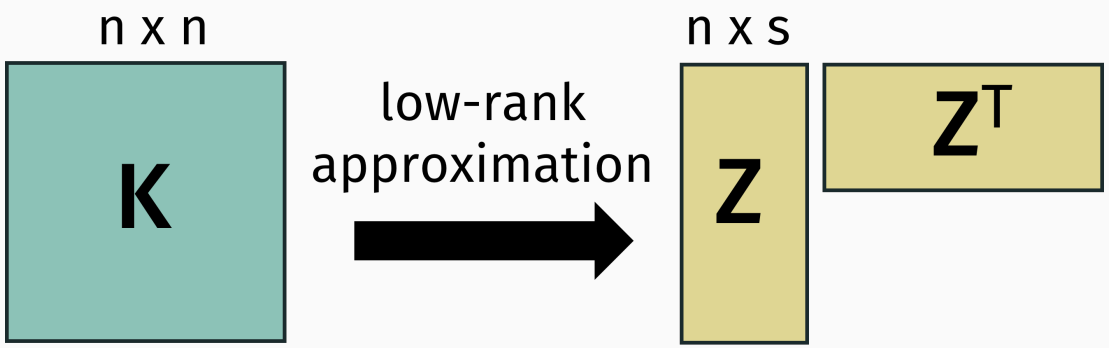
STANDARD APPROACH: LOW-RANK APPROXIMATION

Find approximation $\tilde{K} = ZZ^T$ for K (which is symmetric and positive semidefinite):



STANDARD APPROACH: LOW-RANK APPROXIMATION

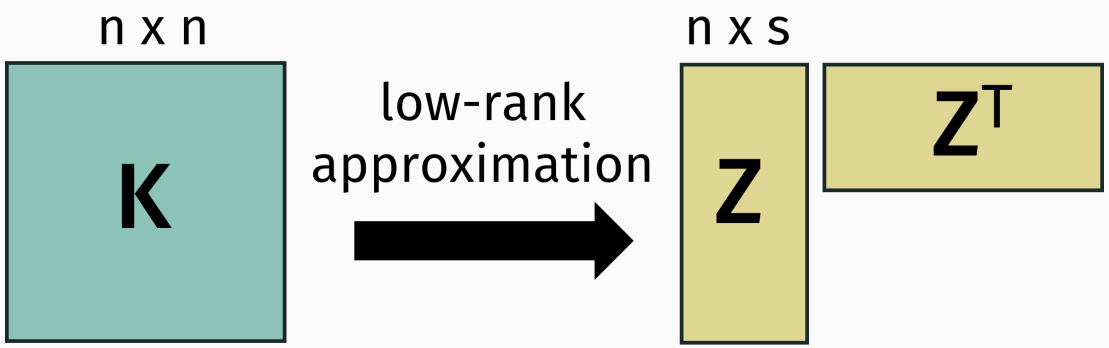
Find approximation $\tilde{\mathbf{K}} = \mathbf{Z}\mathbf{Z}^T$ for $\mathbf{K}$ (which is symmetric and positive semidefinite):



We can typically set $s \ll n$.

STANDARD APPROACH: LOW-RANK APPROXIMATION

Find approximation $\tilde{\mathbf{K}} = \mathbf{Z}\mathbf{Z}^T$ for $\mathbf{K}$ (which is symmetric and positive semidefinite):

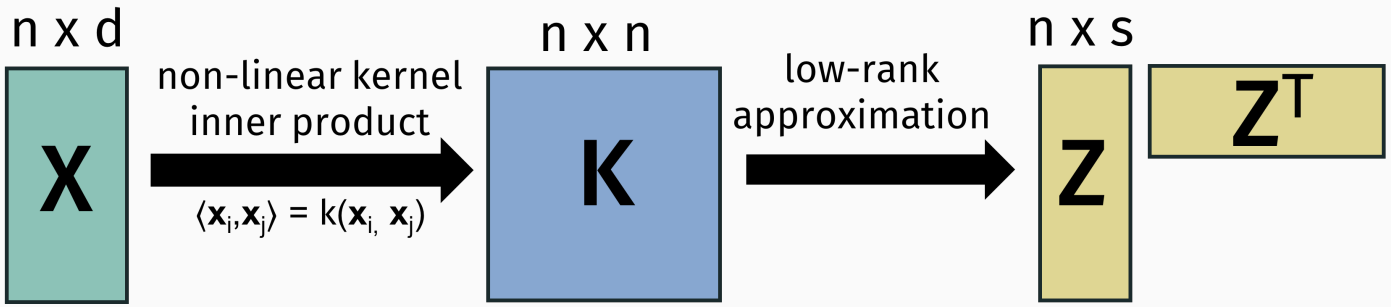


We can typically set $s \ll n$.

- $\mathbf{Z}$ takes $O(ns)$ space to store.
- Orthogonalization, eigendecomposition, and inversion of $\mathbf{Z}\mathbf{Z}^T$ all take just $O(ns^2)$ time.
- $\mathbf{Z}\mathbf{Z}^T\mathbf{x}$ can be computed in $O(ns)$ time.

IMPLICIT LOW-RANK APPROXIMATION

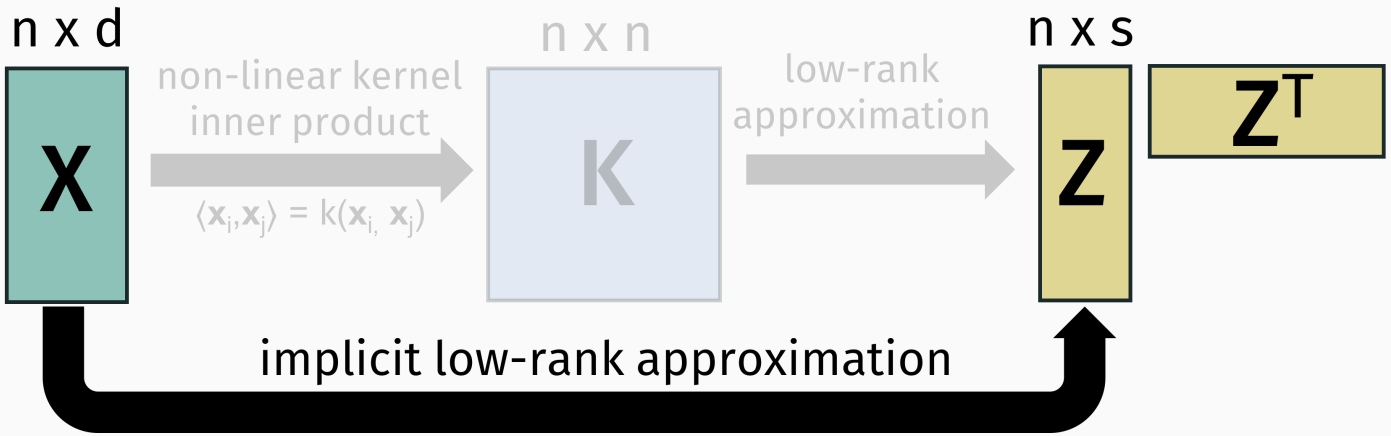
All standard low-rank approximation algorithms take $\Omega(n^2)$ time for kernel matrices: we at least need to compute $\mathbf{K}$.



Goal: Find methods that avoid explicit access to the $\mathbf{K}$.

IMPLICIT LOW-RANK APPROXIMATION

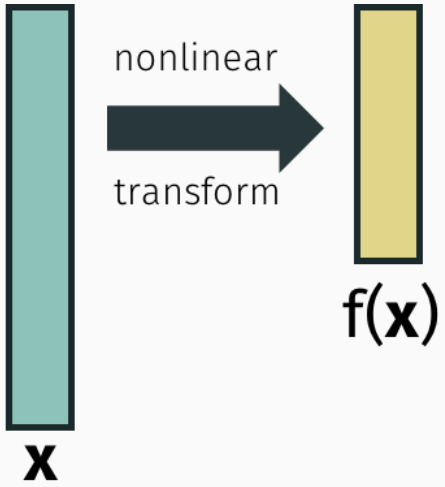
All standard low-rank approximation algorithms take $\Omega(n^2)$ time for kernel matrices: we at least need to compute $\mathbf{K}$.



Goal: Find methods that avoid explicit access to the $\mathbf{K}$.

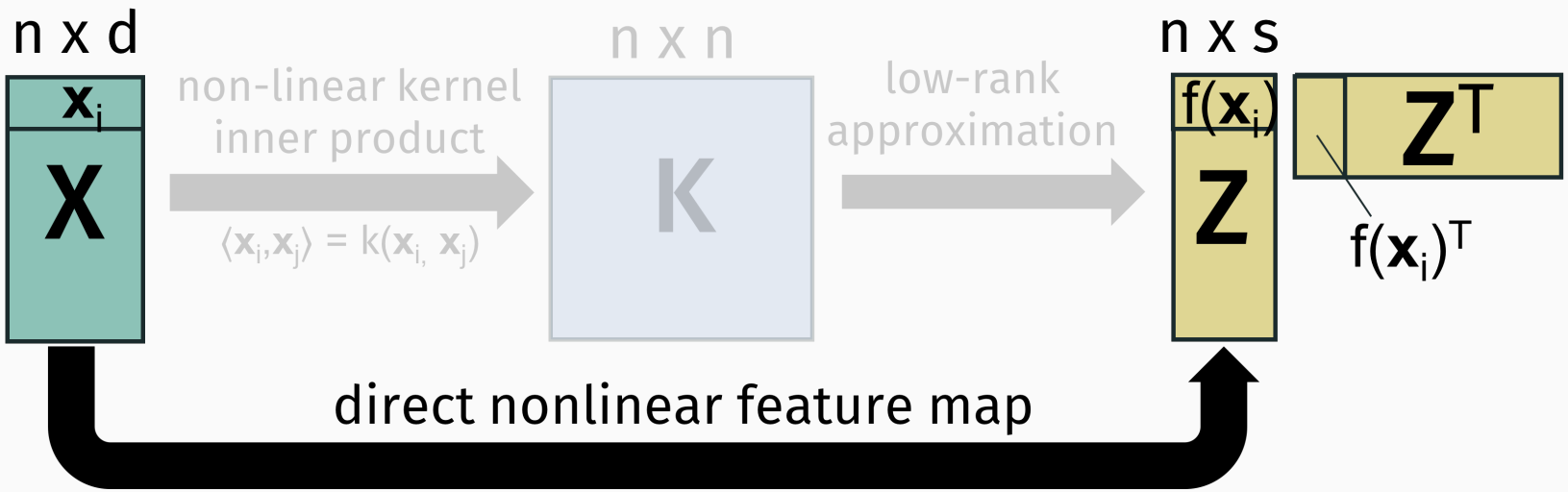
Goal: Find a compact nonlinear function $f(\cdot)$ such that

$$\langle f(\mathbf{x}), f(\mathbf{y}) \rangle \approx k(\mathbf{x}, \mathbf{y})$$

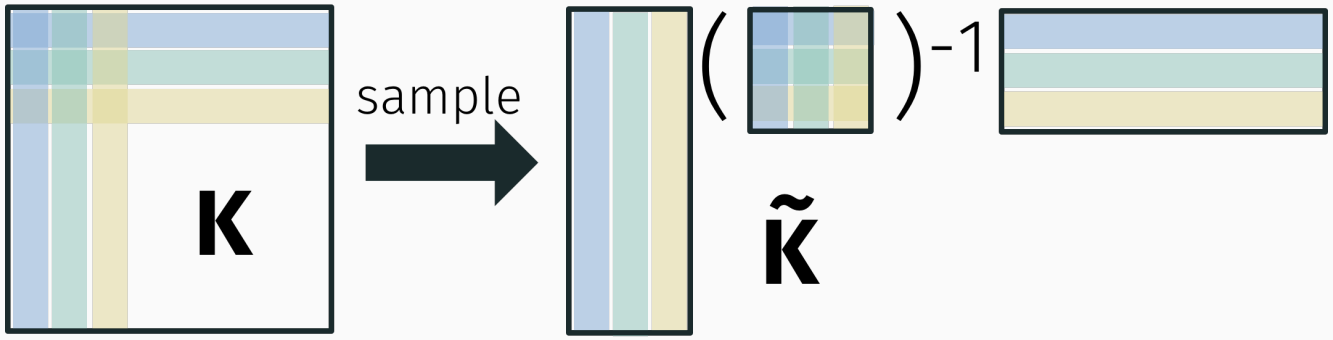


Goal: Find a compact nonlinear function $f(\cdot)$ such that

$$\langle f(\mathrm{x}), f(\mathrm{y})\rangle \approx k(\mathrm{x}, \mathrm{y})$$

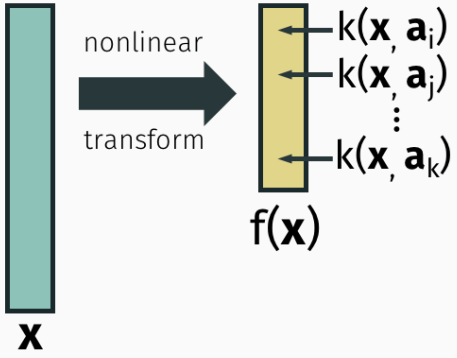


Example: Nyström approximation.



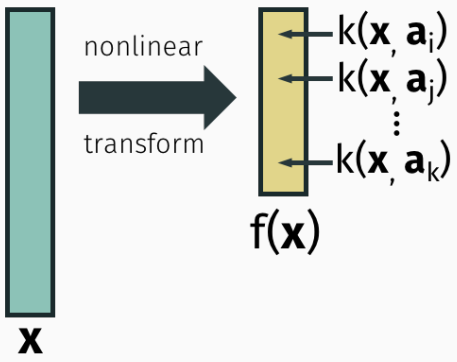
Construct $\tilde{K}$ from subsample of K's columns and rows.

Example: Nyström approximation.



Construct $\tilde{K}$ from subsample of K's columns and rows.

Example: Nyström approximation.



Construct $\tilde{\mathbf{K}}$ from subsample of $\mathbf{K}$'s columns and rows.

Can match optimal k-rank approximation to $(1+\epsilon)$ factor with $O(nk/\epsilon)$ <u>total</u> samples [Musco, Musco, NIPS 2017].

RAHIMI RECHT RANDOM FOURIER FEATURES, NIPS 2007

Suppose $k(\mathbf{x}, \mathbf{y})$ is **shift invariant**.

RAHIMI RECHT RANDOM FOURIER FEATURES, NIPS 2007

Suppose $k(\mathbf{x}, \mathbf{y})$ is **shift invariant**. I.e.

$$k(\mathbf{x}, \mathbf{y}) = k(\boldsymbol{\Delta}) \qquad \text{where} \qquad \boldsymbol{\Delta} = \mathbf{x} - \mathbf{y}.$$

RAHIMI RECHT RANDOM FOURIER FEATURES, NIPS 2007

Suppose $k(\mathbf{x}, \mathbf{y})$ is **shift invariant**. I.e.

$$k(\mathbf{x}, \mathbf{y}) = k(\boldsymbol{\Delta}) \qquad \text{where} \qquad \boldsymbol{\Delta} = \mathbf{x} - \mathbf{y}.$$

RAHIMI RECHT RANDOM FOURIER FEATURES, NIPS 2007

Suppose $k(\mathbf{x}, \mathbf{y})$ is **shift invariant**. I.e.

$$k(\mathbf{x}, \mathbf{y}) = k(\boldsymbol{\Delta}) \qquad \text{where} \qquad \boldsymbol{\Delta} = \mathbf{x} - \mathbf{y}.$$

- Gaussian (RBF) kernel: $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$

RAHIMI RECHT RANDOM FOURIER FEATURES, NIPS 2007

Suppose $k(\mathbf{x}, \mathbf{y})$ is **shift invariant**. I.e.

$$k(\mathbf{x}, \mathbf{y}) = k(\boldsymbol{\Delta}) \qquad \text{where} \qquad \boldsymbol{\Delta} = \mathbf{x} - \mathbf{y}.$$

- Gaussian (RBF) kernel: $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$
- Laplace kernel: $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_1}$

RAHIMI RECHT RANDOM FOURIER FEATURES, NIPS 2007

Suppose $k(\mathbf{x}, \mathbf{y})$ is **shift invariant**. I.e.

$$k(\mathbf{x}, \mathbf{y}) = k(\boldsymbol{\Delta}) \qquad \text{where} \qquad \boldsymbol{\Delta} = \mathbf{x} - \mathbf{y}.$$

- Gaussian (RBF) kernel: $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$
- Laplace kernel: $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_1}$
- Matern kernel, Cauchy kernel, rational quadratic, etc.

RAHIMI RECHT RANDOM FOURIER FEATURES, NIPS 2007

Suppose $k(\mathbf{x}, \mathbf{y})$ is **shift invariant**. I.e.

$$k(\mathbf{x}, \mathbf{y}) = k(\boldsymbol{\Delta}) \qquad \text{where} \qquad \boldsymbol{\Delta} = \mathbf{x} - \mathbf{y}.$$

- Gaussian (RBF) kernel: $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$
- Laplace kernel: $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_1}$
- Matern kernel, Cauchy kernel, rational quadratic, etc.

RANDOM FOURIER FEATURES (RFF) ALGORITHM

Write $k(\boldsymbol{\Delta})$ using its inverse Fourier transform,

$$p(\boldsymbol{\eta}) = \mathcal{F}^{-1} k(\boldsymbol{\Delta}) \qquad k(\boldsymbol{\Delta}) = \int_{\boldsymbol{\eta} \in \mathbb{R}^d} p(\boldsymbol{\eta}) e^{-i\pi \boldsymbol{\eta}^T \boldsymbol{\Delta}} d\boldsymbol{\eta}$$

RANDOM FOURIER FEATURES (RFF) ALGORITHM

Write $k(\boldsymbol{\Delta})$ using its inverse Fourier transform,

$$p(\boldsymbol{\eta}) = \mathcal{F}^{-1}k(\boldsymbol{\Delta}) \qquad k(\boldsymbol{\Delta}) = \int_{\boldsymbol{\eta}\in\mathbb{R}^d} p(\boldsymbol{\eta})e^{-i\pi\boldsymbol{\eta}^T\boldsymbol{\Delta}}d\boldsymbol{\eta}$$

Approximate with finite sum for any $\boldsymbol{\Delta}$:

$$k(\boldsymbol{\Delta}) \approx \frac{1}{S}\sum_{j=1}^{S} c_j e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}$$

RANDOM FOURIER FEATURES (RFF) ALGORITHM

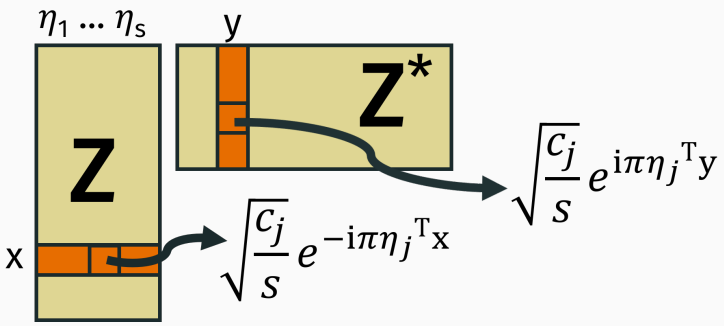
Write $k(\mathbf{\Delta})$ using its inverse Fourier transform,

$$p(\boldsymbol{\eta}) = \mathcal{F}^{-1}k(\mathbf{\Delta}) \qquad k(\mathbf{\Delta}) = \int_{\boldsymbol{\eta}\in\mathbb{R}^d} p(\boldsymbol{\eta})e^{-i\pi\boldsymbol{\eta}^T\mathbf{\Delta}}d\boldsymbol{\eta}$$

Approximate with finite sum for any $\mathbf{\Delta}$:

$$k(\mathbf{\Delta}) \approx \frac{1}{s}\sum_{j=1}^{s} c_j e^{-i\pi\boldsymbol{\eta}_j^T\mathbf{\Delta}}$$

Immediately gives low-rank approximation:



RANDOM FOURIER FEATURES (RFF) ALGORITHM

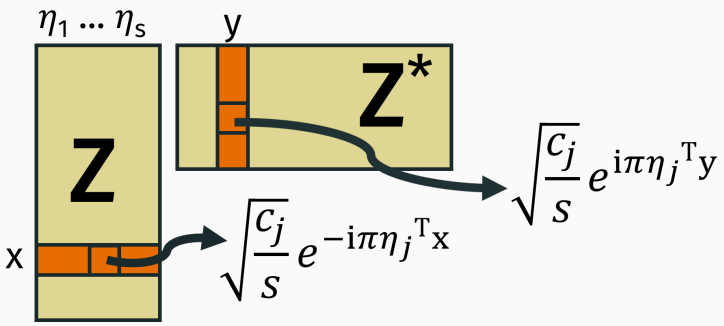
Write $k(\boldsymbol{\Delta})$ using its inverse Fourier transform,

$$p(\boldsymbol{\eta}) = \mathcal{F}^{-1}k(\boldsymbol{\Delta}) \qquad k(\boldsymbol{\Delta}) = \int_{\boldsymbol{\eta}\in\mathbb{R}^d} p(\boldsymbol{\eta})e^{-i\pi\boldsymbol{\eta}^T\boldsymbol{\Delta}}d\boldsymbol{\eta}$$

Approximate with finite sum for any $\boldsymbol{\Delta}$:

$$k(\boldsymbol{\Delta}) \approx \frac{1}{s}\sum_{j=1}^{s} c_j e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}$$

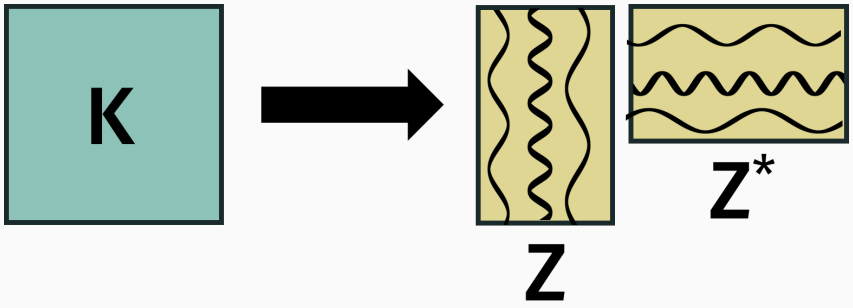
Immediately gives low-rank approximation:



$$Z(x)Z(y)^* = \sum_j \frac{c_j}{s} e^{-i\pi\boldsymbol{\eta}_j^T(x-y)}$$

$$\approx k(x-y)$$

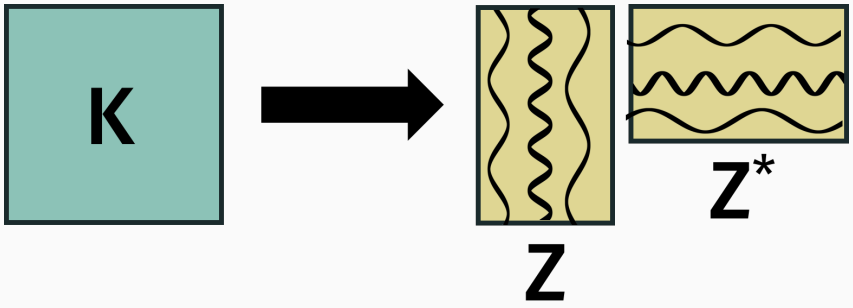
RANDOM FOURIER FEATURES (RFF) ALGORITHM



How do we choose the frequencies in the finite sum?

$$\int_{\boldsymbol{\eta}\in\mathbb{R}^d} p(\boldsymbol{\eta}) e^{-i\pi\boldsymbol{\eta}^T\boldsymbol{\Delta}} d\boldsymbol{\eta} \quad \approx \quad \frac{1}{S}\sum_{j=1}^{S} c_j e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}$$

RANDOM FOURIER FEATURES (RFF) ALGORITHM



How do we choose the frequencies in the finite sum?

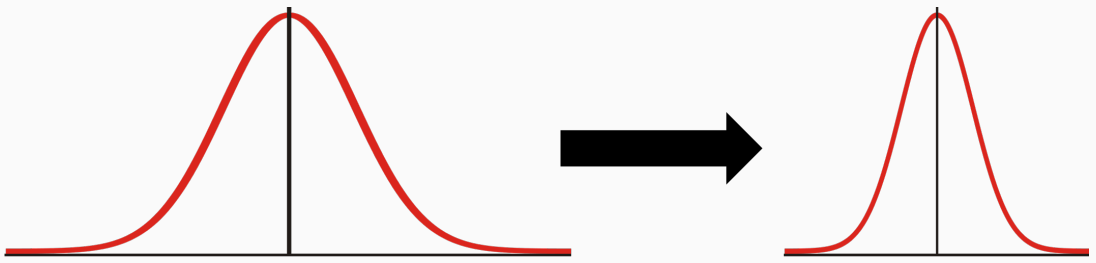
$$\int_{\boldsymbol{\eta}\in\mathbb{R}^d} p(\boldsymbol{\eta}) e^{-i\pi\boldsymbol{\eta}^T\boldsymbol{\Delta}} d\boldsymbol{\eta} \quad \approx \quad \frac{1}{s}\sum_{j=1}^{s} c_j e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}$$

Bochner's Theorem: For <u>shift-invariant, positive semidefinite kernel functions</u>, $p(\boldsymbol{\eta}) \geq 0$ for all $\boldsymbol{\eta}$.

Bochner's Theorem: For shift-invariant, positive semidefinite kernel functions, $p(\boldsymbol{\eta}) \geq 0$ for all $\boldsymbol{\eta}$.

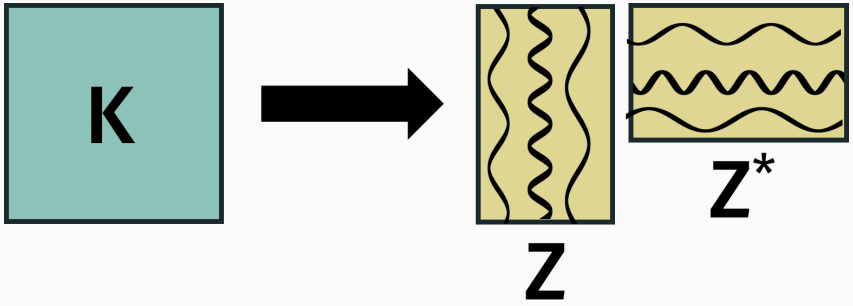
Bochner's Theorem: For shift-invariant, positive semidefinite kernel functions, $p(\boldsymbol{\eta}) \geq 0$ for all $\boldsymbol{\eta}$.

Example (Gaussian kernel):



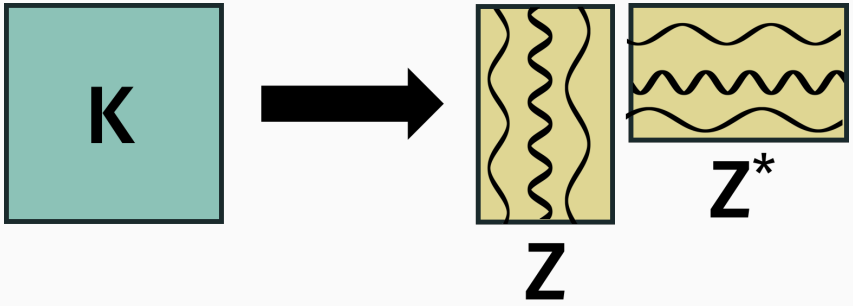
$k(\mathbf{x} - \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$ $\quad$ $\mathrm{p}(\boldsymbol{\eta}) \propto e^{-\|\boldsymbol{\eta}\|^2/4}$

RANDOM FOURIER FEATURES (RFF) ALGORITHM



Sample frequencies from distribution $p(\eta)$.

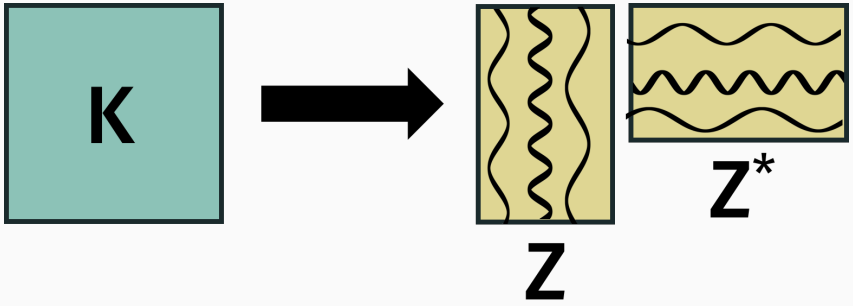
RANDOM FOURIER FEATURES (RFF) ALGORITHM



Sample frequencies from distribution $p(\eta)$.

$$\mathbb{E}\left[\frac{1}{s}\sum_{j=1}^{s} e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}\right] = \mathbb{E}\left[e^{-i\pi\boldsymbol{\eta}^T\boldsymbol{\Delta}}\right]$$

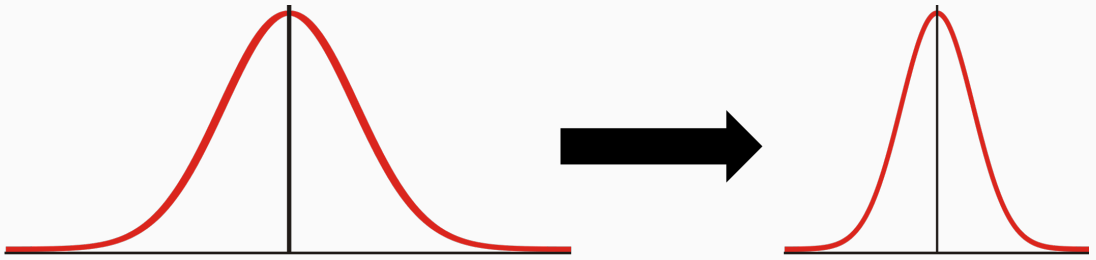
RANDOM FOURIER FEATURES (RFF) ALGORITHM



Sample frequencies from distribution $p(\eta)$.

$$\mathbb{E}\left[\frac{1}{s}\sum_{j=1}^{s} e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}\right] = \mathbb{E}\left[e^{-i\pi\boldsymbol{\eta}^T\boldsymbol{\Delta}}\right] = \int_{\boldsymbol{\eta}\in\mathbb{R}^d} p(\boldsymbol{\eta})e^{-i\pi\boldsymbol{\eta}^T\boldsymbol{\Delta}}d\boldsymbol{\eta}$$

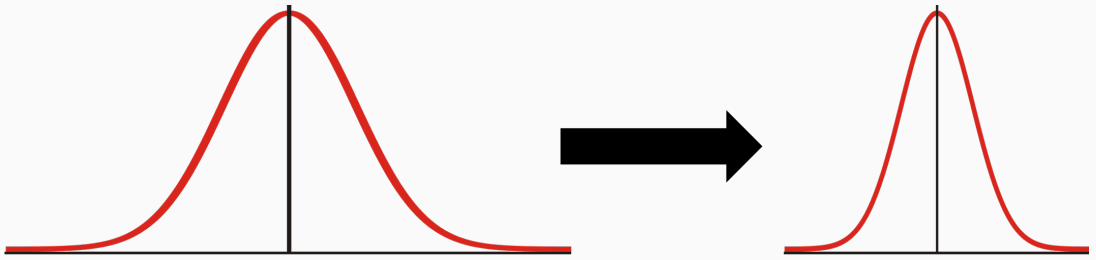
RANDOM FOURIER FEATURES (RFF) ALGORITHM



$k(\mathbf{x}-\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$ $\quad$ $\mathrm{p}(\boldsymbol{\eta}) \propto e^{-\|\boldsymbol{\eta}\|^2/4}$

- Sampling frequencies by $p(\boldsymbol{\eta})$ is correct in expectation.

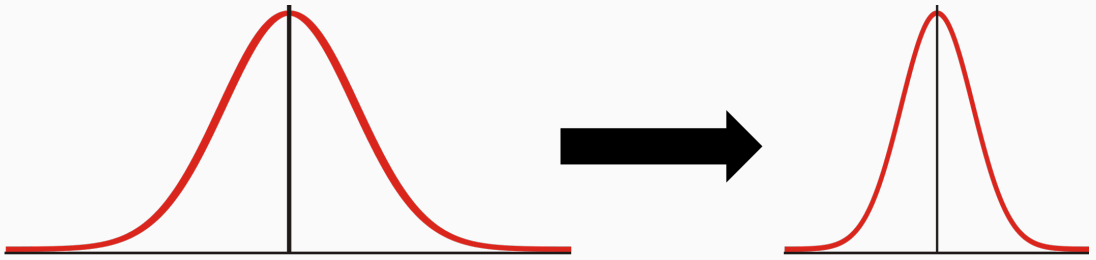
RANDOM FOURIER FEATURES (RFF) ALGORITHM



$k(\mathbf{x}-\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$ $\qquad$ $\mathrm{p}(\boldsymbol{\eta}) \propto e^{-\|\boldsymbol{\eta}\|^2/4}$

- Sampling frequencies by $p(\boldsymbol{\eta})$ is correct in expectation.
- Final sum $\frac{1}{s}\sum_{j=1}^{s} e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}} \approx k(\boldsymbol{\Delta})$ only has terms with magnitude $|e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}| = 1$. Real part ≤ 1. Imaginary part ≤ 1.

RANDOM FOURIER FEATURES (RFF) ALGORITHM

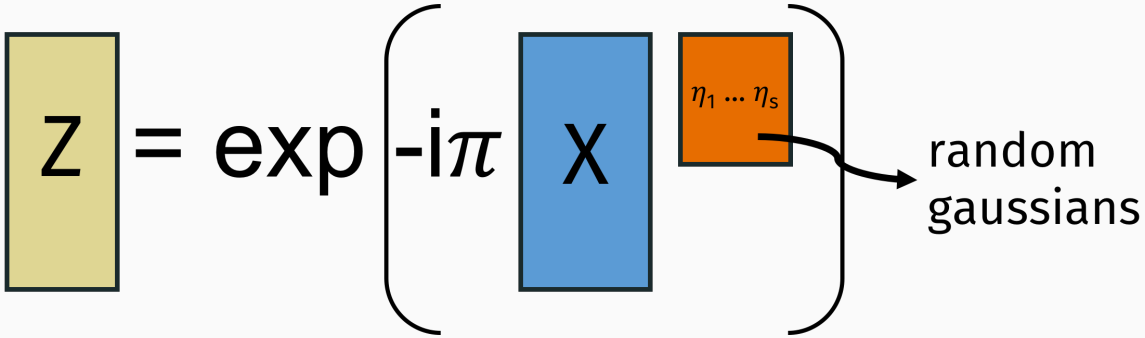


$k(\mathbf{x}-\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2}$ $\qquad$ $\mathrm{p}(\boldsymbol{\eta}) \propto e^{-\|\boldsymbol{\eta}\|^2/4}$

- Sampling frequencies by $p(\boldsymbol{\eta})$ is correct in expectation.
- Final sum $\frac{1}{s}\sum_{j=1}^{s} e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}} \approx k(\boldsymbol{\Delta})$ only has terms with magnitude $|e^{-i\pi\boldsymbol{\eta}_j^T\boldsymbol{\Delta}}| = 1$. Real part ≤ 1. Imaginary part ≤ 1.
- By Chernoff bound, if we take $\mathbf{O}\left(\frac{\log \mathbf{n}}{\epsilon^2}\right)$ samples, we approximate every entry in $\mathbf{K}$ to error $\pm\epsilon$ w.h.p.

RANDOM FOURIER FEATURES (RFF) ALGORITHM

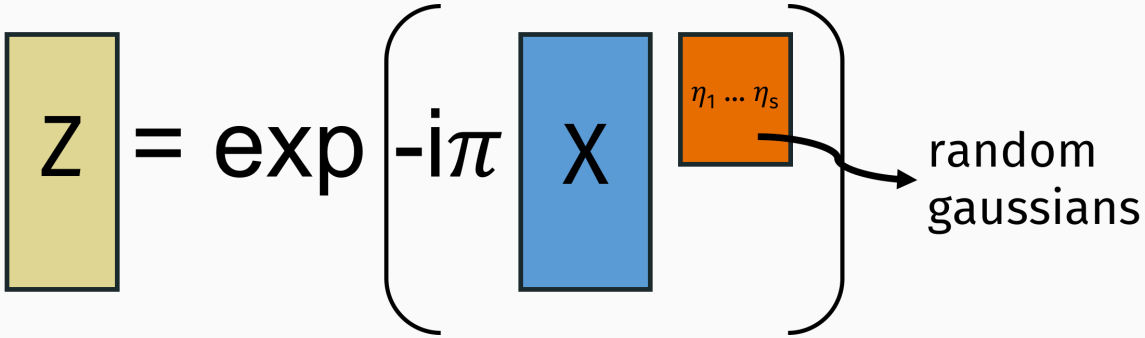
Super simple algorithm. For Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_2^2}$:



$$Z_{x,j} = e^{-i\pi \mathbf{x}^T \boldsymbol{\eta}_j} \text{ for } \boldsymbol{\eta}_j \sim \mathcal{N}$$

RANDOM FOURIER FEATURES (RFF) ALGORITHM

Super simple algorithm. For Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_2^2}$:

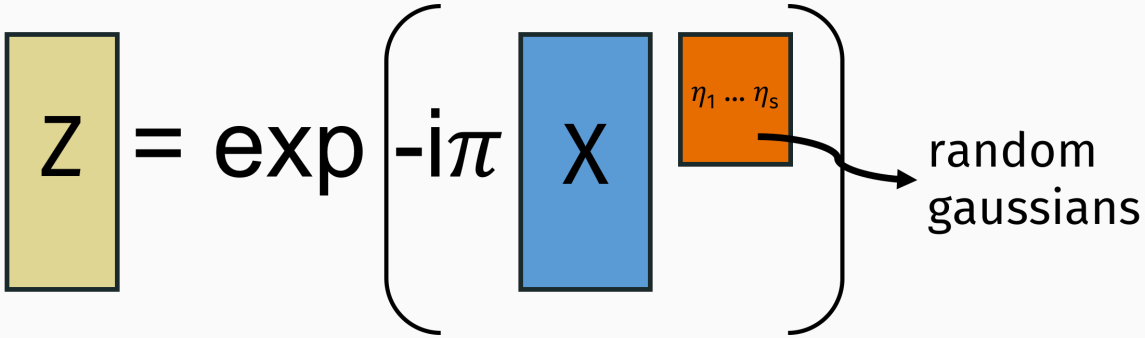


$$Z_{x,j} = e^{-i\pi \mathbf{x}^T \boldsymbol{\eta}_j} \text{ for } \boldsymbol{\eta}_j \sim \mathcal{N}$$

```
G = randn(d, s);
Z = exp(-sqrt(-1)*pi*X*G)/sqrt(s);
```

RANDOM FOURIER FEATURES (RFF) ALGORITHM

Super simple algorithm. For Gaussian kernel $k(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|_2^2}$:

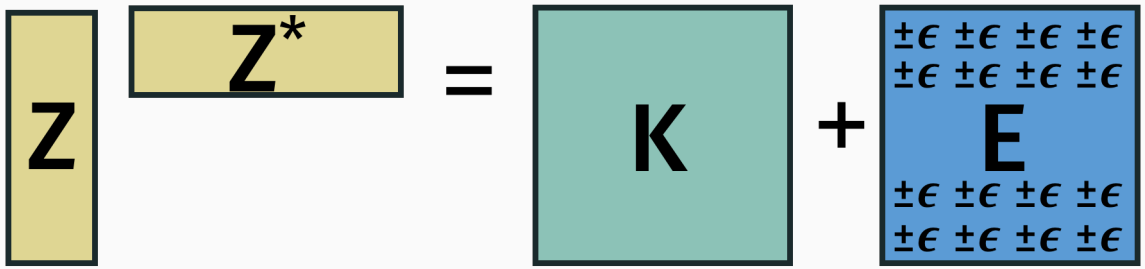


$$Z_{x,j} = e^{-i\pi \mathbf{x}^T \boldsymbol{\eta}_j} \text{ for } \boldsymbol{\eta}_j \sim \mathcal{N}$$

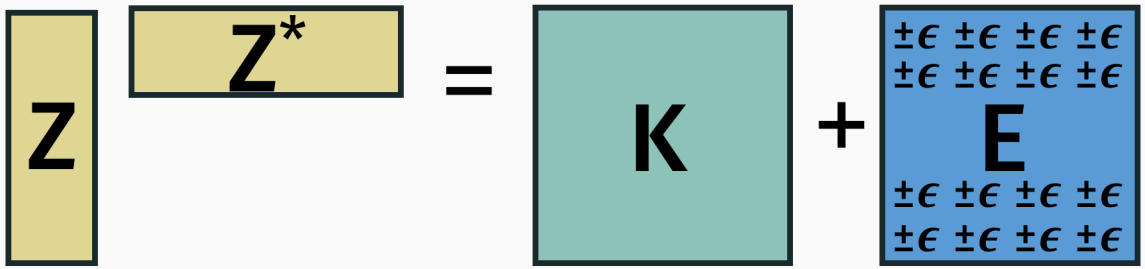
```
G = randn(d, s);
Z = exp(-sqrt(-1)*pi*X*G)/sqrt(s);
```

This is a so-called <u>oblivious</u> sketch.

Want to improve the approximation guarantee:

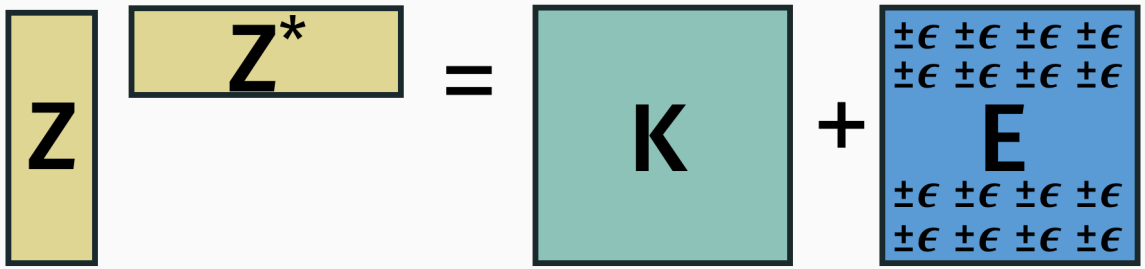


Want to improve the approximation guarantee:



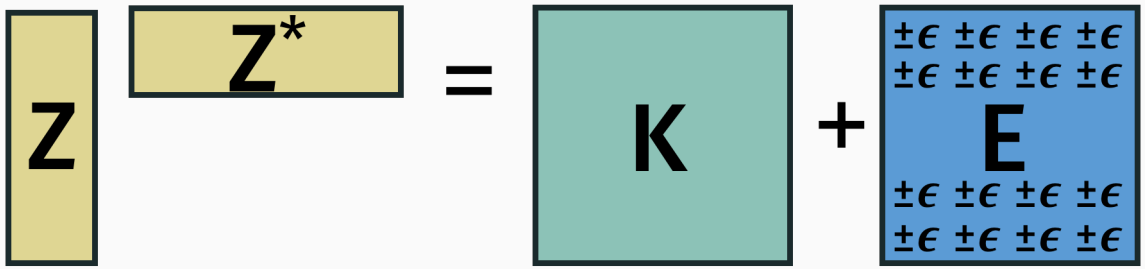
- Does not give bounds on $\|ZZ^* - K\|$ unless we take $\Omega(n^2)$ samples.

Want to improve the approximation guarantee:



- Does not give bounds on $\|\mathbf{Z}\mathbf{Z}^* - \mathbf{K}\|$ unless we take $\Omega(n^2)$ samples.
- No clear implications for downstream learning tasks. E.g., does $(\mathbf{Z}\mathbf{Z}^* + \lambda\mathbf{I})^{-1}$ approximate $(\mathbf{K} + \lambda\mathbf{I})^{-1}$?

Want to improve the approximation guarantee:



- Does not give bounds on $\|\mathbf{Z}\mathbf{Z}^* - \mathbf{K}\|$ unless we take $\Omega(n^2)$ samples.
- No clear implications for downstream learning tasks. E.g., does $(\mathbf{Z}\mathbf{Z}^* + \lambda\mathbf{I})^{-1}$ approximate $(\mathbf{K} + \lambda\mathbf{I})^{-1}$?
- Faster but less accurate than a Nyström approximation with the same number of samples in practice.

STARTING POINT

We want matrix-like error bounds, not entrywise bounds.

STARTING POINT

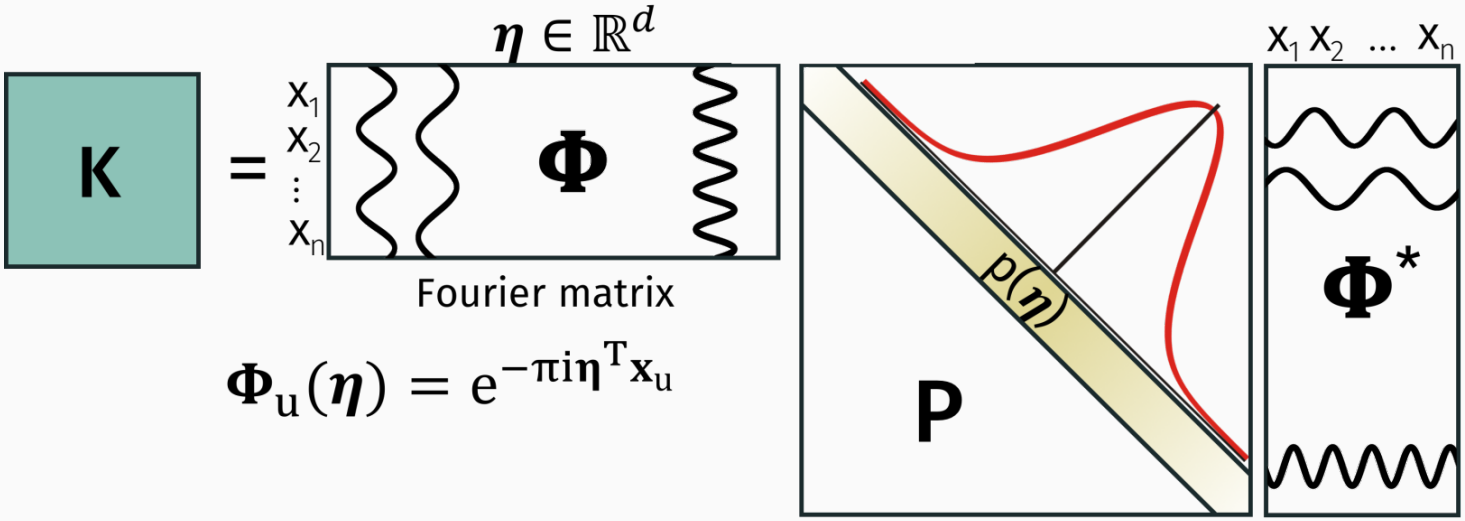
We want matrix-like error bounds, not entrywise bounds.

Key Idea: Analyze the random Fourier features algorithm as a matrix sampling process.

HIGH LEVEL APPROACH

$$k(\mathbf{x}_u - \mathbf{x}_v) = \int_{\boldsymbol{\eta} \in \mathbb{R}^d} p(\boldsymbol{\eta}) e^{-i\pi \boldsymbol{\eta}^T (\mathbf{x}_u - \mathbf{x}_v)} d\boldsymbol{\eta}$$

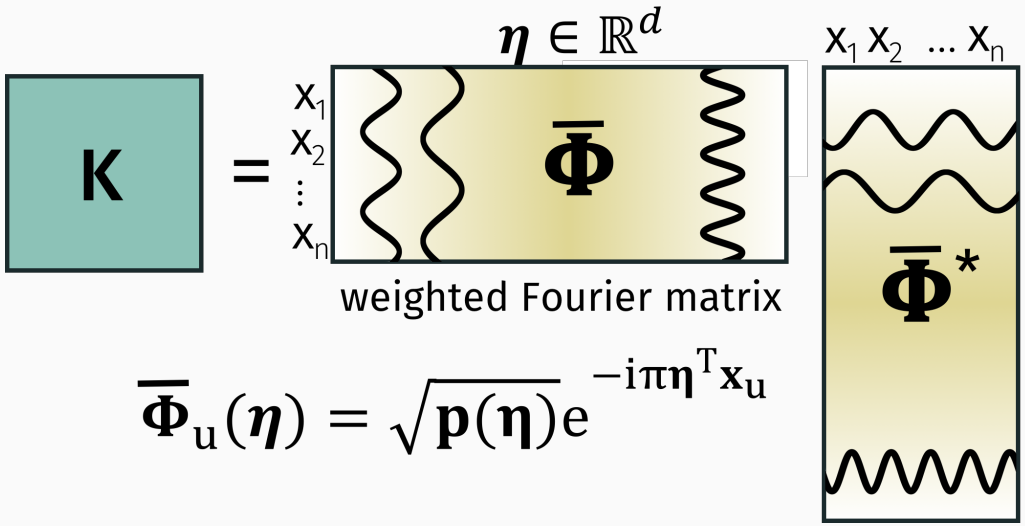
$$k(\mathbf{x}_u - \mathbf{x}_v) = \int_{\boldsymbol{\eta} \in \mathbb{R}^d} p(\boldsymbol{\eta}) e^{-i\pi \boldsymbol{\eta}^T (\mathbf{x}_u - \mathbf{x}_v)} d\boldsymbol{\eta}$$



Kernel Fourier Transform

HIGH LEVEL APPROACH

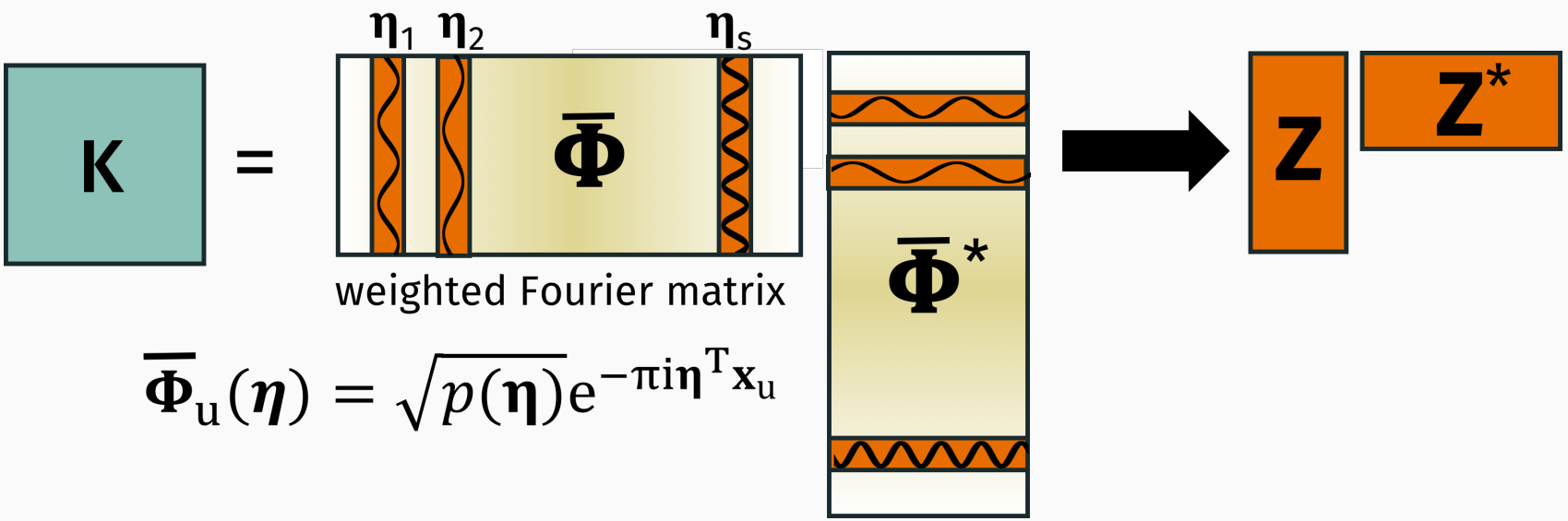
$$k(\mathbf{x}_u - \mathbf{x}_v) = \int_{\boldsymbol{\eta} \in \mathbb{R}^d} p(\boldsymbol{\eta}) e^{-i\pi \boldsymbol{\eta}^T (\mathbf{x}_u - \mathbf{x}_v)} d\boldsymbol{\eta}$$



$$\overline{\mathbf{\Phi}}_{\mathbf{u}}(\boldsymbol{\eta}) = \sqrt{\mathbf{p}(\boldsymbol{\eta})} e^{-i\pi \boldsymbol{\eta}^T \mathbf{x}_u}$$

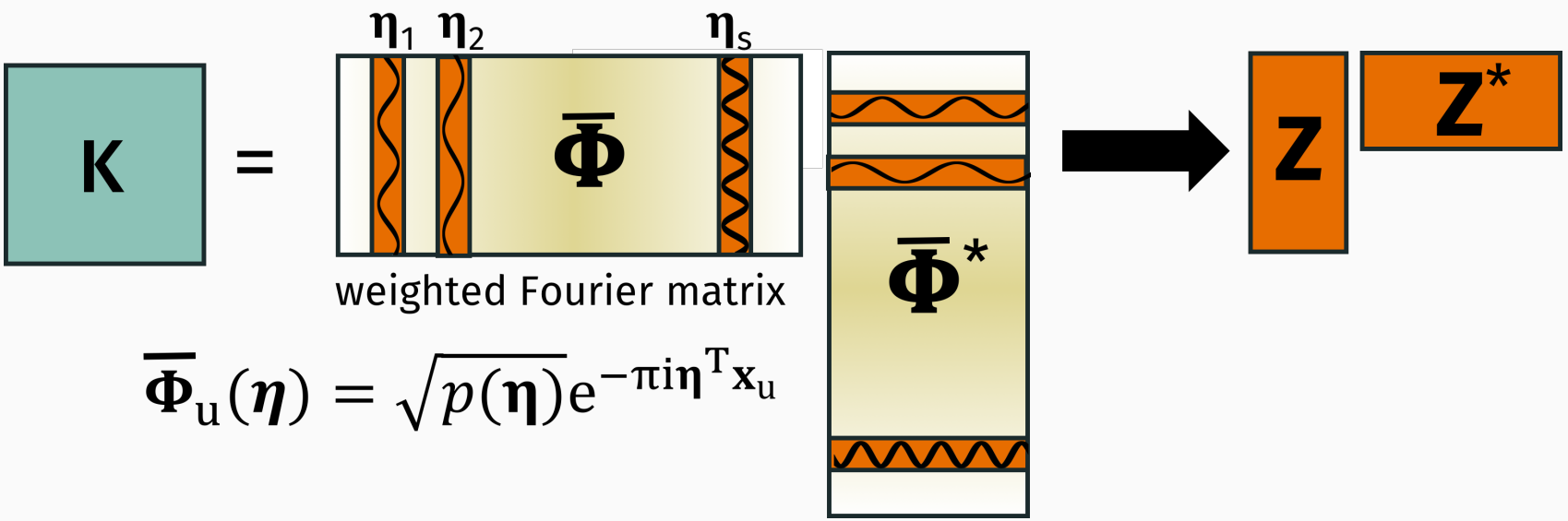
Kernel Fourier Transform

STANDARD RFF = COLUMN NORM SAMPLING



$$\overline{\boldsymbol{\Phi}}_u(\boldsymbol{\eta}) = \sqrt{p(\boldsymbol{\eta})}e^{-\pi i \boldsymbol{\eta}^T \mathbf{x}_u}$$

STANDARD RFF = COLUMN NORM SAMPLING

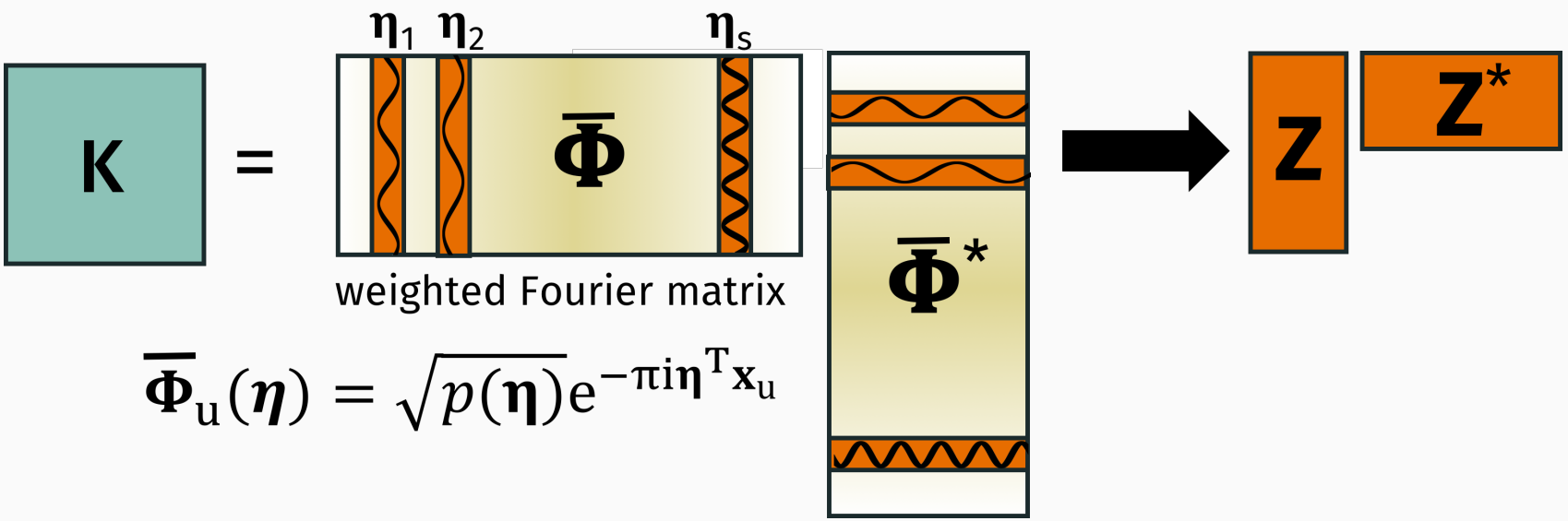


$$\bar{\Phi}_u(\eta) = \sqrt{p(\eta)}e^{-\pi i \eta^T x_u}$$

Standard RFF selects column $\bar{\Phi}(\eta)$ with probability $\propto p(\eta)$.

.

STANDARD RFF = COLUMN NORM SAMPLING

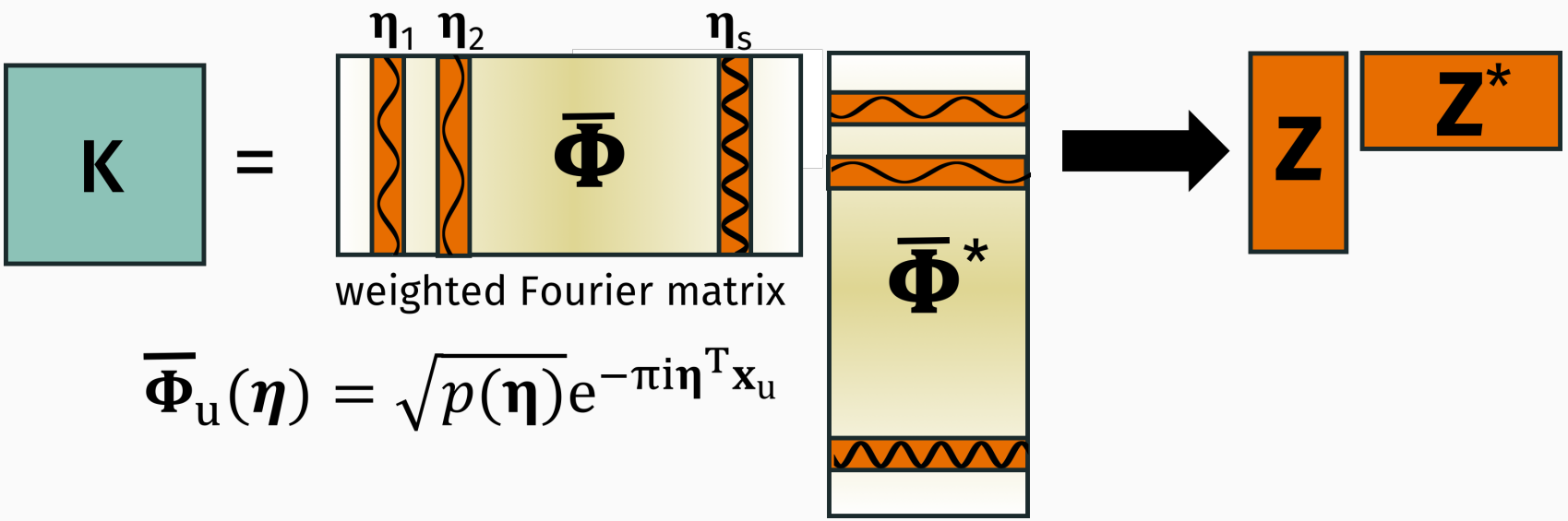


$$\bar{\Phi}_u(\eta) = \sqrt{p(\eta)}e^{-\pi i \eta^T x_u}$$

Standard RFF selects column $\bar{\Phi}(\eta)$ with probability $\propto p(\eta)$.

(with probability proportional to the column's squared norm).

STANDARD RFF = COLUMN NORM SAMPLING



$$\bar{\Phi}_u(\eta) = \sqrt{p(\eta)}e^{-\pi i \eta^T x_u}$$

Standard RFF selects column $\bar{\Phi}(\eta)$ with probability $\propto p(\eta)$.

(with probability proportional to the column's squared norm).

Simple matrix Chernoff already gives better bounds:
$\|K - ZZ^*\|_2 \leq \epsilon$ with $\tilde{O}(n/\epsilon^2)$ samples.

LEVERAGE SCORE SAMPLING

For matrix approximation, norm based sampling probabilities are known to be suboptimal!

LEVERAGE SCORE SAMPLING

For matrix approximation, norm based sampling probabilities are known to be suboptimal!

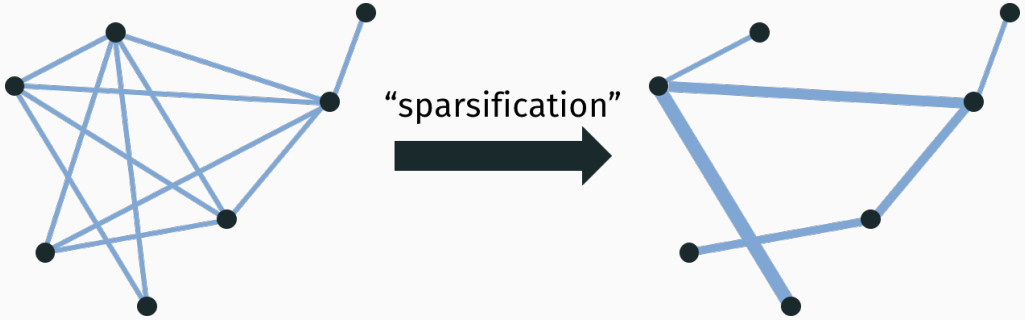
We can obtain better theoretical bounds and empirical performance if we sample by column leverage scores.

LEVERAGE SCORE SAMPLING

For matrix approximation, norm based sampling probabilities are known to be suboptimal!

We can obtain better theoretical bounds and empirical performance if we sample by column leverage scores.

[Spielman, Srivastava, et al. 2008 and on]



(see work on effective resistances for spectral graph sparsification, randomized linear algebra, etc.)

BETTER APPROXIMATION GUARANTEES

λ-Ridge Leverage Score Sampling:

$$(1-\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}) \preceq \mathbf{K} + \lambda \mathbf{I} \preceq (1+\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}).$$

BETTER APPROXIMATION GUARANTEES

λ-Ridge Leverage Score Sampling:

$$(1-\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}) \preceq \mathbf{K} + \lambda \mathbf{I} \preceq (1+\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}).$$

- Since $A \preceq B \Rightarrow B^{-1} \preceq A^{-1}$, $(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I})^{-1} \approx (\mathbf{K} + \lambda \mathbf{I})^{-1}$

BETTER APPROXIMATION GUARANTEES

λ-Ridge Leverage Score Sampling:

$$(1-\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}) \preceq \mathbf{K} + \lambda \mathbf{I} \preceq (1+\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}).$$

- Since $A \preceq B \Rightarrow B^{-1} \preceq A^{-1}$, $(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I})^{-1} \approx (\mathbf{K} + \lambda \mathbf{I})^{-1}$
- Spectral approximation also gives bounds for kernel PCA, k-means, CCA, etc. (Cohen, Musco, Musco '16,'17)

λ-Ridge Leverage Score Sampling:

$$(1-\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}) \preceq \mathbf{K} + \lambda \mathbf{I} \preceq (1+\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}).$$

- Since $A \preceq B \Rightarrow B^{-1} \preceq A^{-1}$, $(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I})^{-1} \approx (\mathbf{K} + \lambda \mathbf{I})^{-1}$
- Spectral approximation also gives bounds for kernel PCA, k-means, CCA, etc. (Cohen, Musco, Musco '16,'17)

Need $\tilde{O}(s_\lambda/\epsilon^2)$ samples where s_λ is the statistical dimension:

$$s_\lambda = \mathrm{tr}(\mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1})$$

BETTER APPROXIMATION GUARANTEES

λ-Ridge Leverage Score Sampling:

$$(1-\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}) \preceq \mathbf{K} + \lambda \mathbf{I} \preceq (1+\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}).$$

- Since $A \preceq B \Rightarrow B^{-1} \preceq A^{-1}$, $(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I})^{-1} \approx (\mathbf{K} + \lambda \mathbf{I})^{-1}$
- Spectral approximation also gives bounds for kernel PCA, k-means, CCA, etc. (Cohen, Musco, Musco '16,'17)

Need $\tilde{O}(s_\lambda/\epsilon^2)$ samples where s_λ is the statistical dimension:

$$s_\lambda = \mathrm{tr}(\mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1}) = \sum_{i=1}^{n} \frac{\sigma_i(\mathbf{K})}{\sigma_i(\mathbf{K}) + \lambda}$$

BETTER APPROXIMATION GUARANTEES

λ-Ridge Leverage Score Sampling:

$$(1-\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}) \preceq \mathbf{K} + \lambda \mathbf{I} \preceq (1+\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}).$$

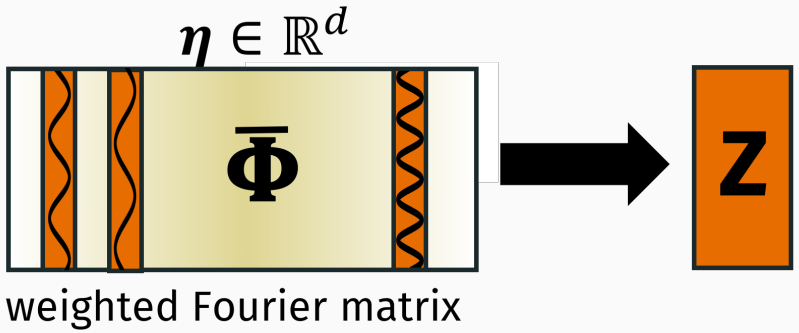
- Since $A \preceq B \Rightarrow B^{-1} \preceq A^{-1}$, $(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I})^{-1} \approx (\mathbf{K} + \lambda \mathbf{I})^{-1}$
- Spectral approximation also gives bounds for kernel PCA, k-means, CCA, etc. (Cohen, Musco, Musco '16,'17)

Need $\tilde{O}(s_\lambda/\epsilon^2)$ samples where s_λ is the statistical dimension:

$$s_\lambda = \mathrm{tr}(\mathbf{K}(\mathbf{K} + \lambda \mathbf{I})^{-1}) = \sum_{i=1}^{n} \frac{\sigma_i(\mathbf{K})}{\sigma_i(\mathbf{K}) + \lambda}$$

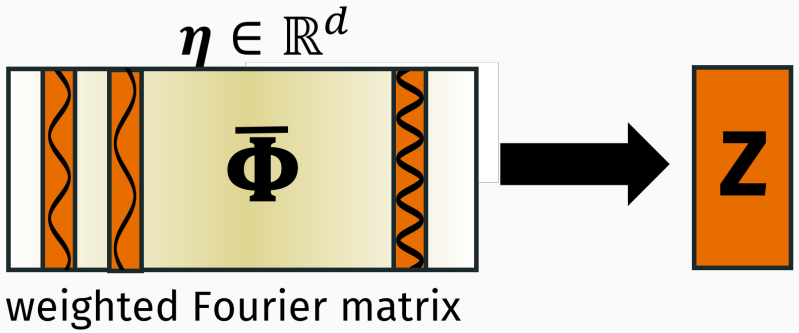
Roughly the number of singular values $\geq \lambda$ plus a term depending on the tail singular values.

What are the ridge leverage scores?



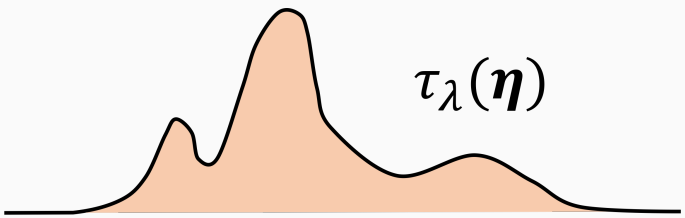
$$\tau_\lambda(\boldsymbol{\eta}) = \bar{\boldsymbol{\Phi}}(\boldsymbol{\eta})^*(\mathsf{K} + \lambda \mathsf{I})^{-1}\bar{\boldsymbol{\Phi}}(\boldsymbol{\eta}).$$

What are the ridge leverage scores?



$$\tau_\lambda(\boldsymbol{\eta}) = \bar{\mathbf{\Phi}}(\boldsymbol{\eta})^*(\mathsf{K} + \lambda \mathsf{I})^{-1}\bar{\mathbf{\Phi}}(\boldsymbol{\eta}).$$

Expensive to invert $\mathsf{K} + \lambda \mathsf{I}$. Even if you could, not at all clear how to efficiently sample from the leverage score distribution.

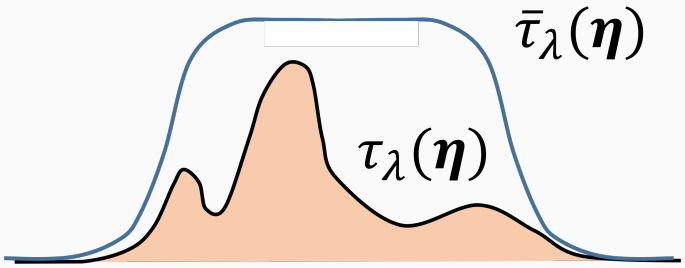


BOUNDING THE FOURIER LEVERAGE SCORES

Goal: Upper bound Fourier ridge leverage scores for common kernels with simple distributions.

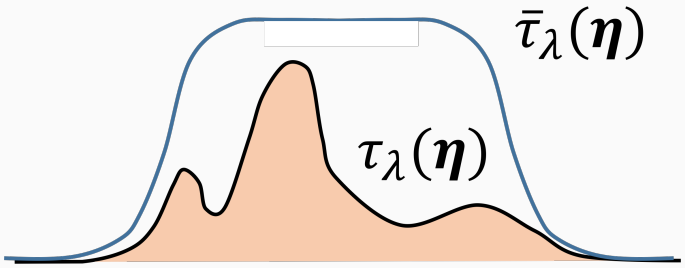
BOUNDING THE FOURIER LEVERAGE SCORES

Goal: Upper bound Fourier ridge leverage scores for common kernels with simple distributions.



BOUNDING THE FOURIER LEVERAGE SCORES

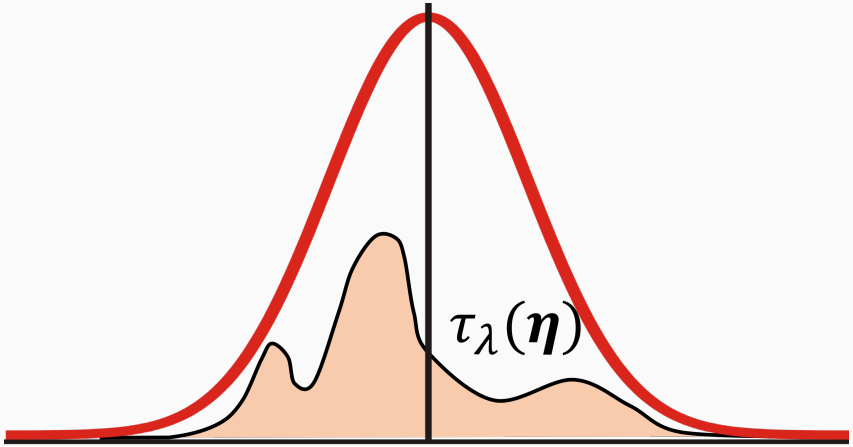
Goal: Upper bound Fourier ridge leverage scores for common kernels with simple distributions.



More closely match leverage score sampling to improve random Fourier features.

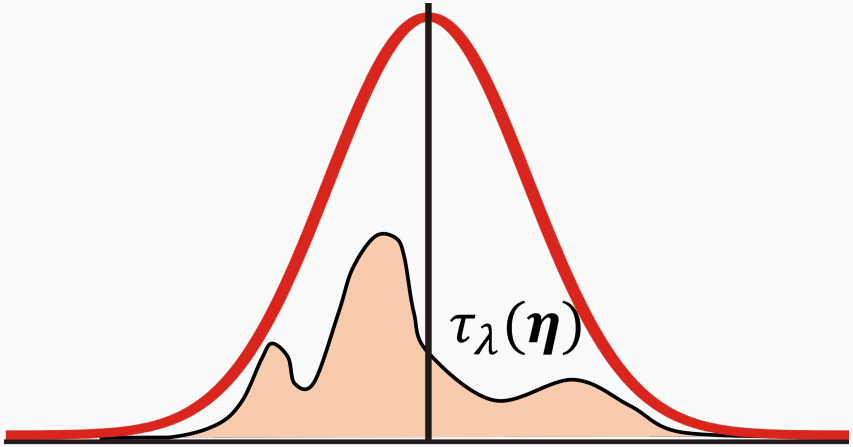
BOUNDING THE FOURIER LEVERAGE SCORES

First observation: Scaled by $\frac{n}{\lambda}$, the standard Rahimi Recht distribution upper bounds the λ-ridge leverage scores.



BOUNDING THE FOURIER LEVERAGE SCORES

First observation: Scaled by $\frac{n}{\lambda}$, the standard Rahimi Recht distribution upper bounds the λ-ridge leverage scores.



Basic Result: Sampling $O(\frac{n}{\lambda} \cdot \frac{1}{\epsilon^2})$ frequencies with RFF gives spectral guarantees.

ALTERNATIVE CHARACTERIZATION OF LEVERAGE SCORES

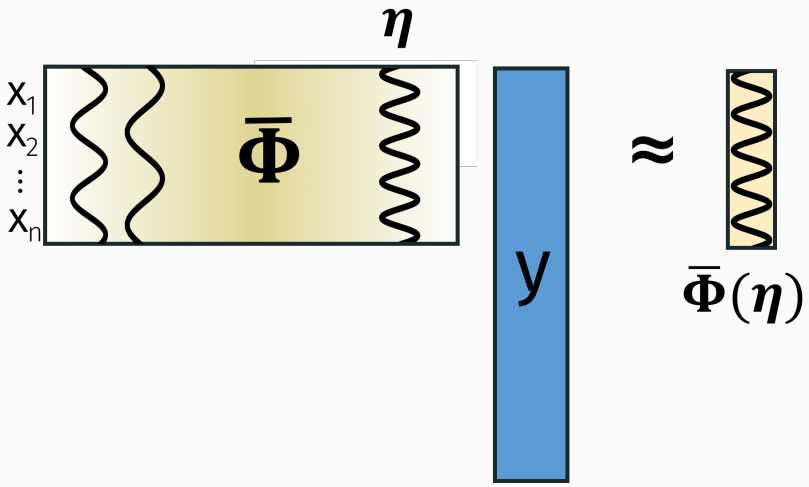
Ridge leverage score $\bar{\mathbf{\Phi}}(\boldsymbol{\eta})^*(\mathsf{K}+\lambda\mathsf{I})^{-1}\bar{\mathbf{\Phi}}(\boldsymbol{\eta})$ solves:

$$\tau_\lambda(\boldsymbol{\eta}) = \min_{\mathbf{y}} \left[\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\mathbf{y} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\mathbf{y}\|_2^2\right].$$

ALTERNATIVE CHARACTERIZATION OF LEVERAGE SCORES

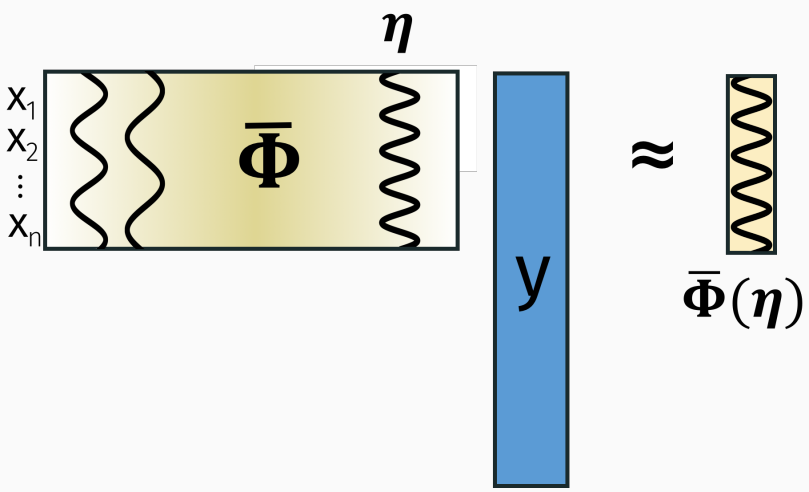
Ridge leverage score $\bar{\mathbf{\Phi}}(\boldsymbol{\eta})^*(\mathsf{K}+\lambda\mathsf{I})^{-1}\bar{\mathbf{\Phi}}(\boldsymbol{\eta})$ solves:

$$\tau_\lambda(\boldsymbol{\eta}) = \min_{\mathsf{y}} \left[\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\mathsf{y} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\mathsf{y}\|_2^2\right].$$



Ridge leverage score $\bar{\mathbf{\Phi}}(\boldsymbol{\eta})^*(\mathsf{K}+\lambda\mathsf{I})^{-1}\bar{\mathbf{\Phi}}(\boldsymbol{\eta})$ solves:

$$\tau_\lambda(\boldsymbol{\eta}) = \min_{\mathsf{y}} \left[\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\mathsf{y} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\mathsf{y}\|_2^2\right].$$

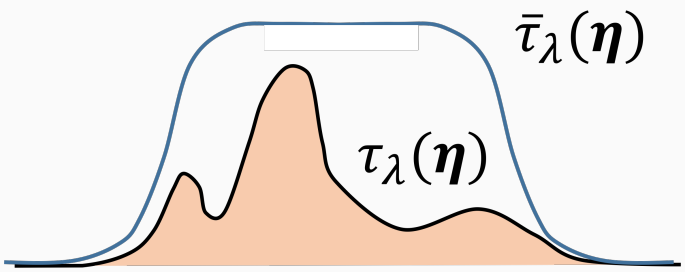


Intuition: y reconstructs frequency $\boldsymbol{\eta}$ from other frequencies.

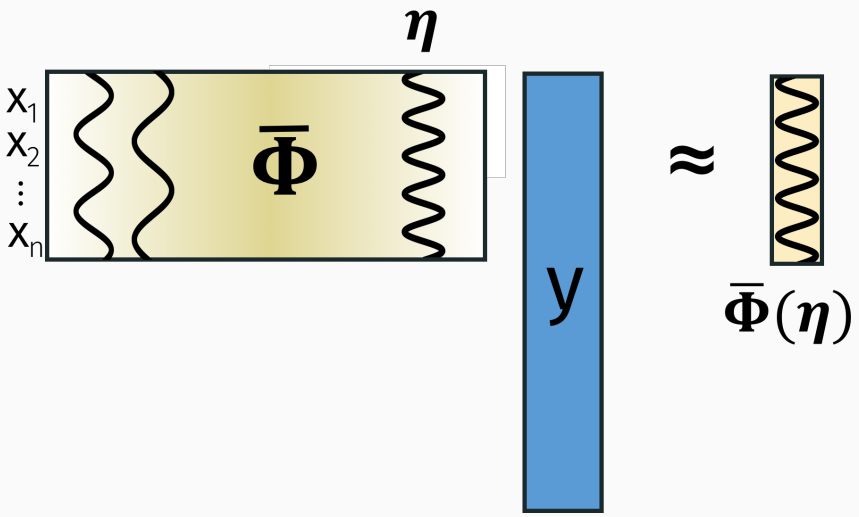
If $\boldsymbol{\eta}$ is "easy" to reconstruct, it is less important to sample.

Approach: Obtain closed form upper bound on leverage scores by exhibiting simple candidate vector $\tilde{\mathbf{y}}$ and noting that:

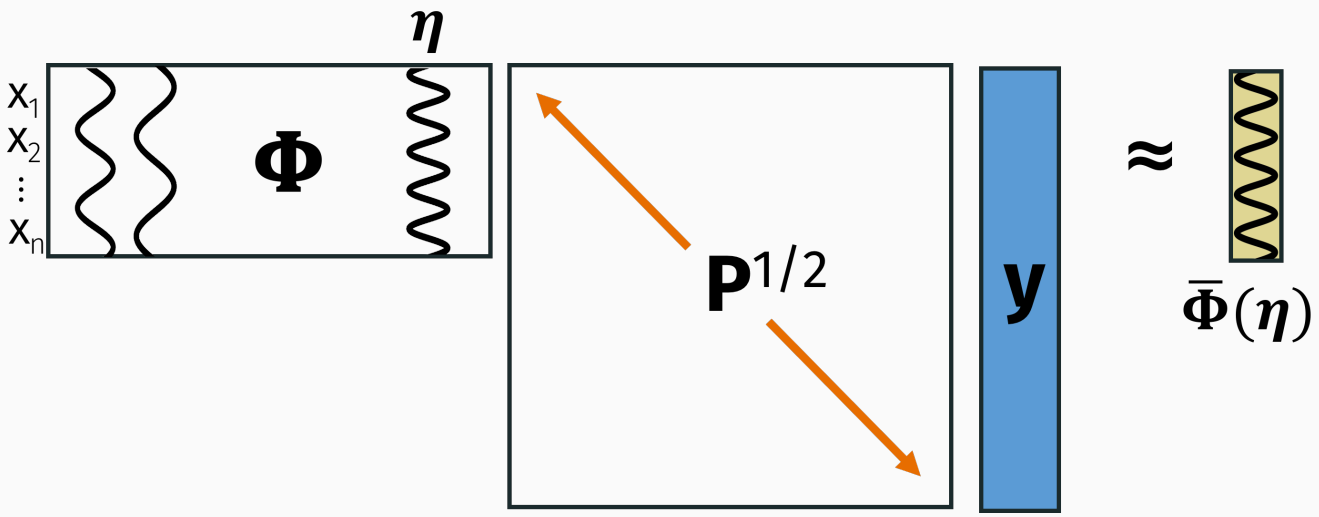
$$\tau_\lambda(\boldsymbol{\eta}) \leq \frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\tilde{\mathbf{y}}\|_2^2.$$



$\bar{\mathbf{\Phi}}\mathbf{y}$ is just the Fourier transform of $\sqrt{\mathbf{P}}\mathbf{y}$ evaluated at $\mathbf{x}_1, \ldots, \mathbf{x}_n$!

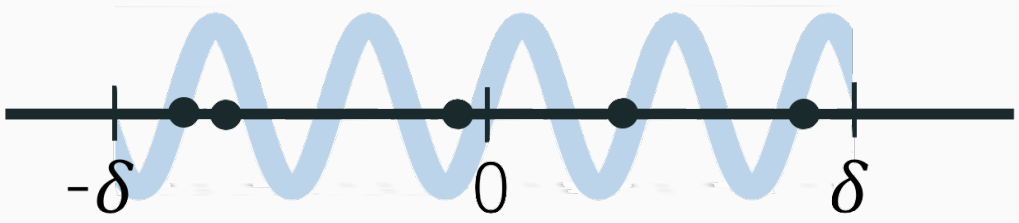


$\bar{\mathbf{\Phi}}\mathbf{y}$ is just the Fourier transform of $\sqrt{\mathbf{P}}\mathbf{y}$ evaluated at $\mathbf{x}_1, \ldots, \mathbf{x}_n$!



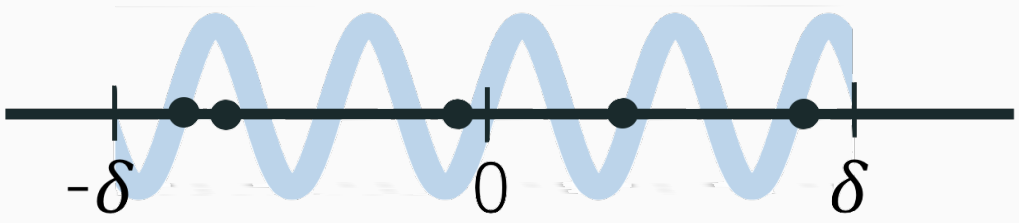
Simplifying Assumption

For remainder of talk: Assume Gaussian kernel, n points in 1-dimension, bounded between $[-\delta, \delta]$.



Simplifying Assumption

For remainder of talk: Assume Gaussian kernel, n points in 1-dimension, bounded between $[-\delta, \delta]$.



We want a function whose Fourier transform matches frequency $\boldsymbol{\eta}$ on these data points.

FIRST ATTEMPT

Need to bound: $\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\tilde{\mathbf{y}}\|_2^2$.

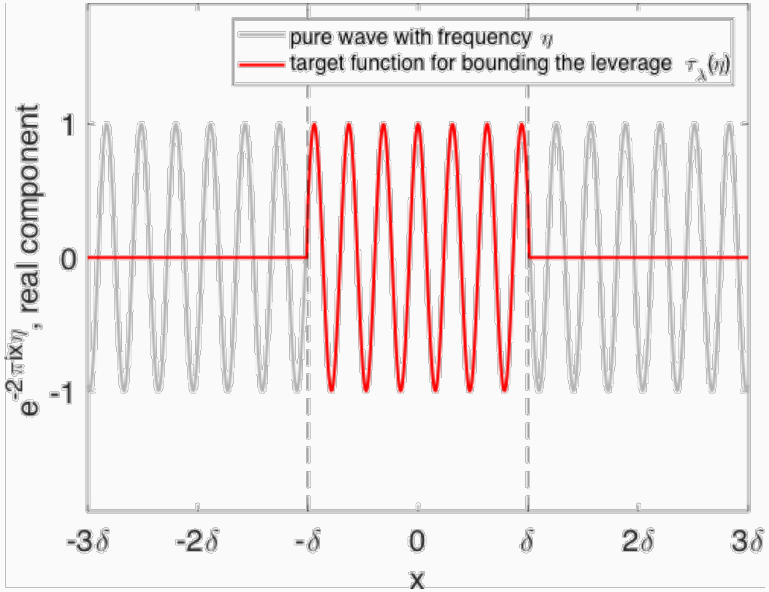
FIRST ATTEMPT

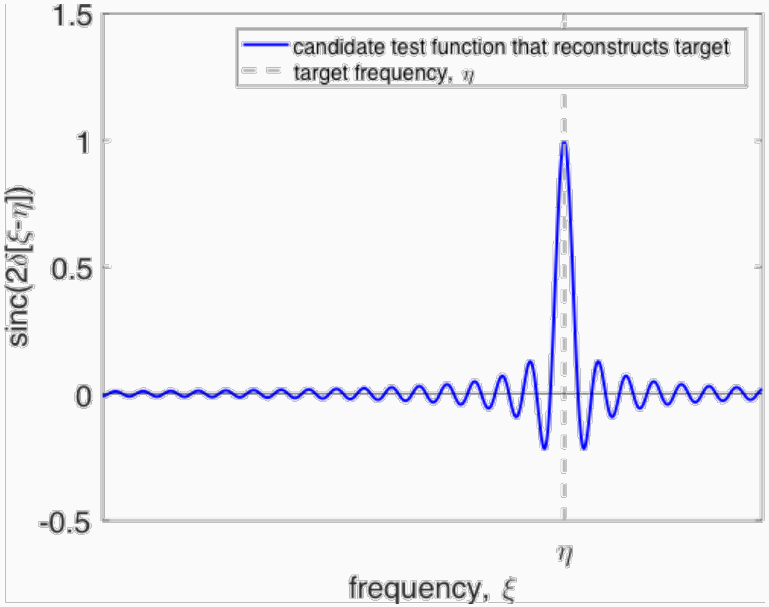
Need to bound: $\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\tilde{\mathbf{y}}\|_2^2$.

Set $\sqrt{\mathbf{P}}\tilde{\mathbf{y}}$ to be a shifted sinc function.

Need to bound: $\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\tilde{\mathbf{y}}\|_2^2$.

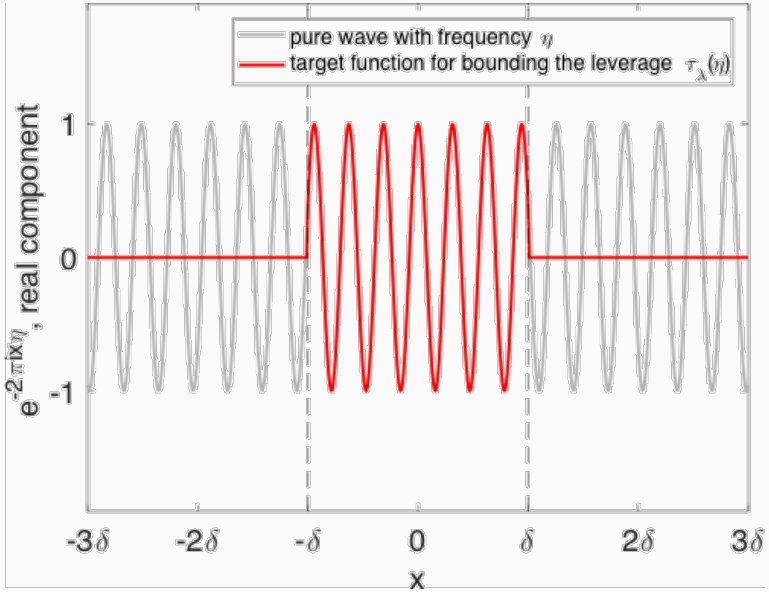
Set $\sqrt{\mathbf{P}}\tilde{\mathbf{y}}$ to be a shifted sinc function.

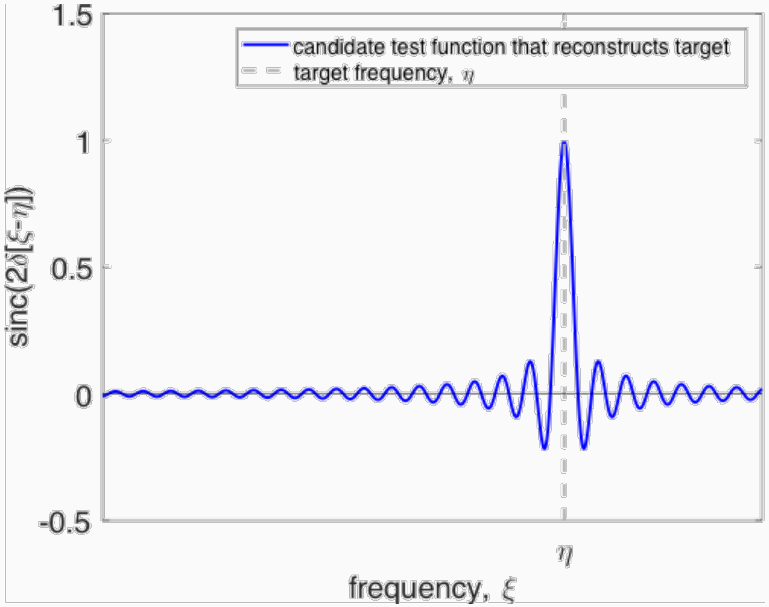




Need to bound: $\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\tilde{\mathbf{y}}\|_2^2$.

Set $\sqrt{\mathbf{P}}\tilde{\mathbf{y}}$ to be a shifted sinc function.

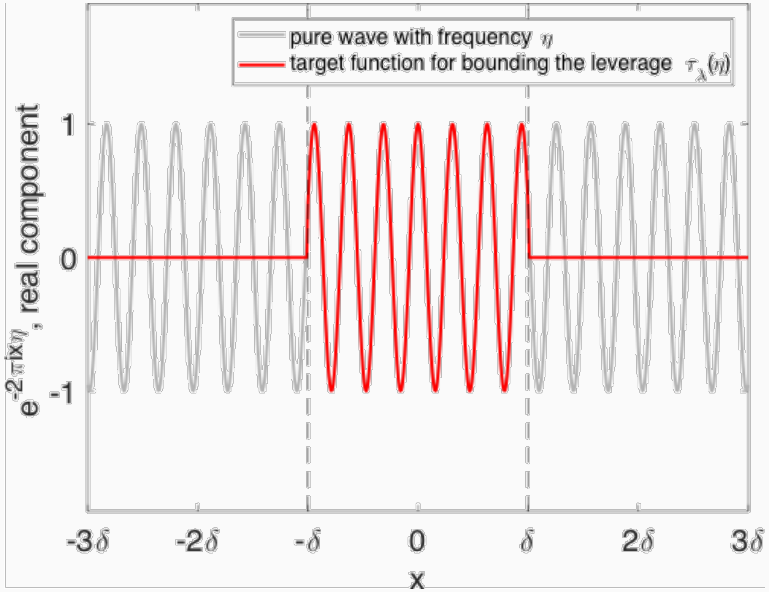




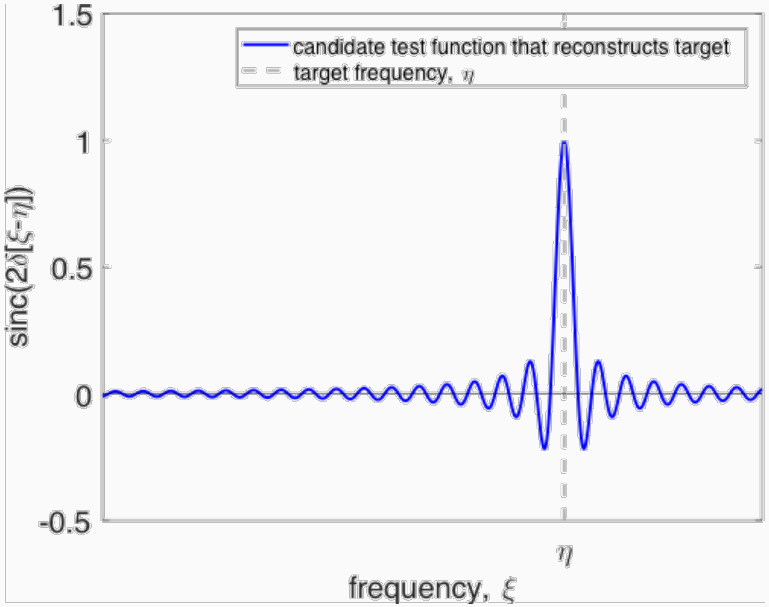
$$\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 = 0!$$

Need to bound: $\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\tilde{\mathbf{y}}\|_2^2$.

Set $\sqrt{\mathbf{P}}\tilde{\mathbf{y}}$ to be a shifted sinc function.



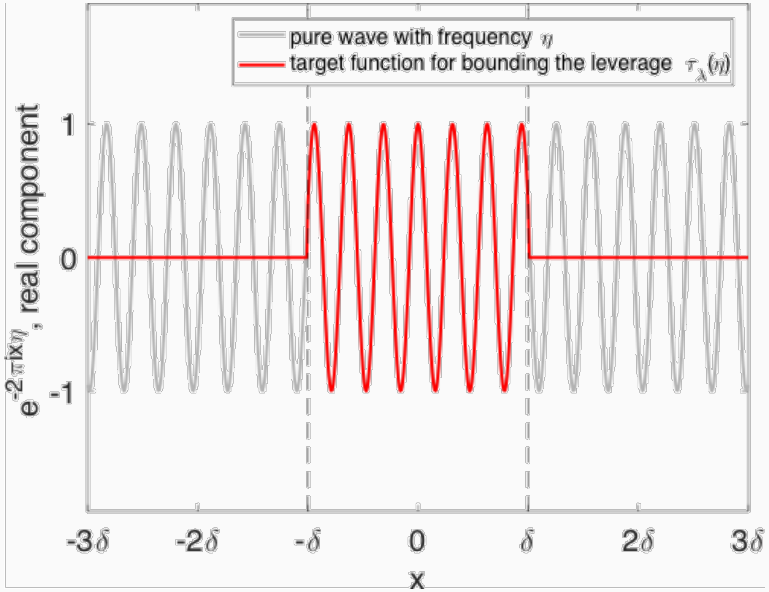
$$\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 = 0!$$

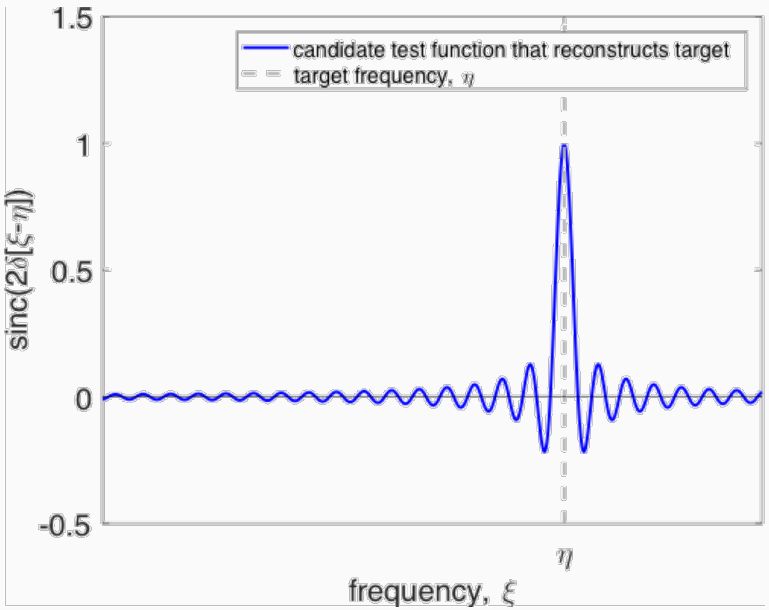


$$\|\sqrt{\mathbf{P}}\tilde{\mathbf{y}}\|_2^2 = O(\delta).$$

Need to bound: $\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 + \|\tilde{\mathbf{y}}\|_2^2$.

Set $\sqrt{\mathbf{P}}\tilde{\mathbf{y}}$ to be a shifted sinc function.



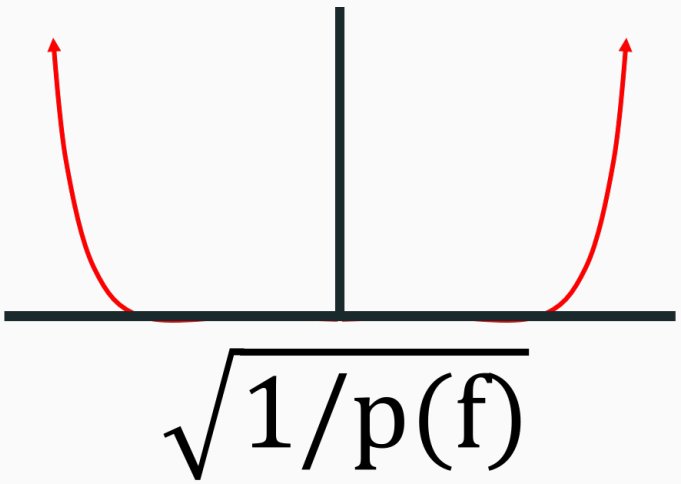


$\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2 = 0!$

$\cancel{\|\sqrt{\mathbf{P}}\tilde{\mathbf{y}}\|_2^2 = O(\delta)}.$

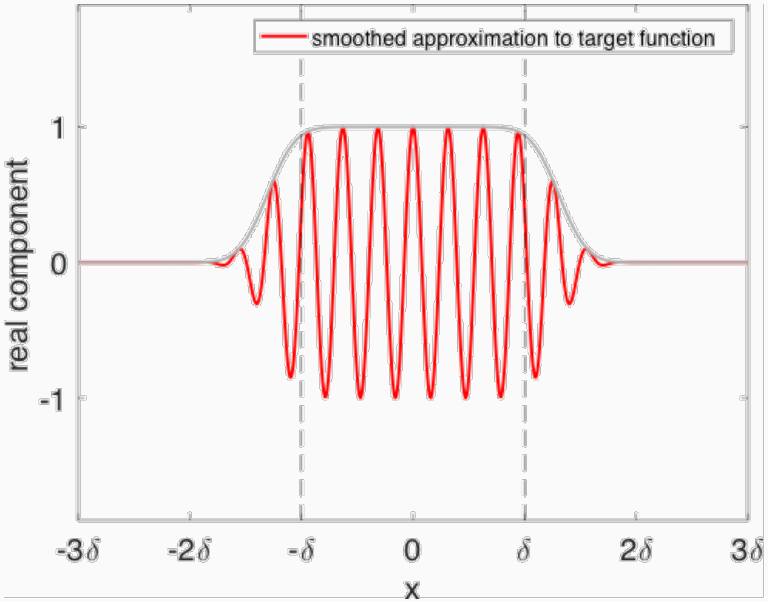
$\|\tilde{\mathbf{y}}\|_2^2$ is much larger!

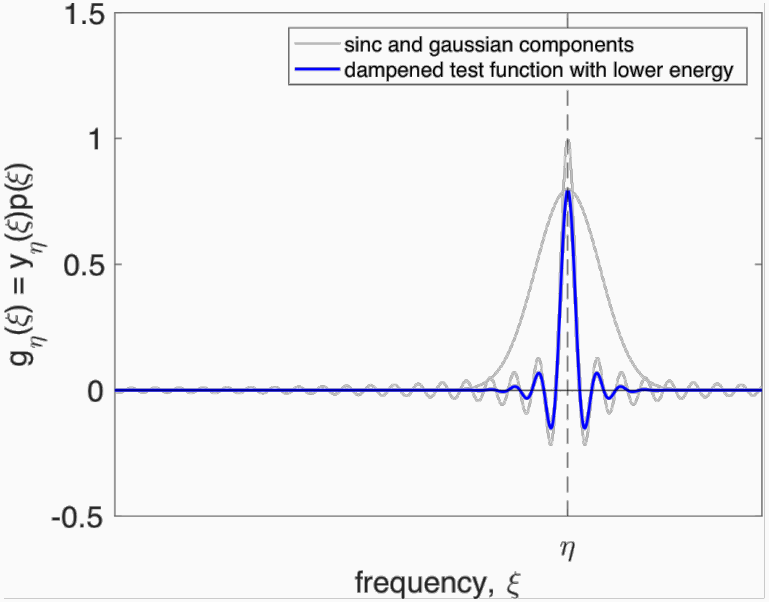
$$\tilde{\mathbf{y}} = \mathbf{P}^{-1/2}\mathbf{P}^{1/2}\tilde{\mathbf{y}} = \mathbf{P}^{-1/2} \cdot \text{sinc function}$$



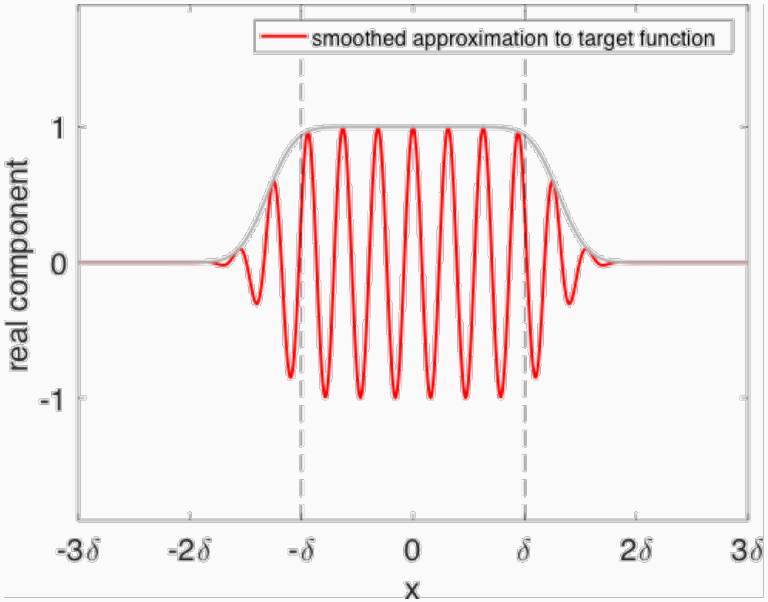
Sinc falls off as $O(1/f)$, but $\frac{1}{\sqrt{p(f)}}$ grows as $e^{O(f^2)}$, so $\|\mathbf{y}\|_2^2$ explodes.

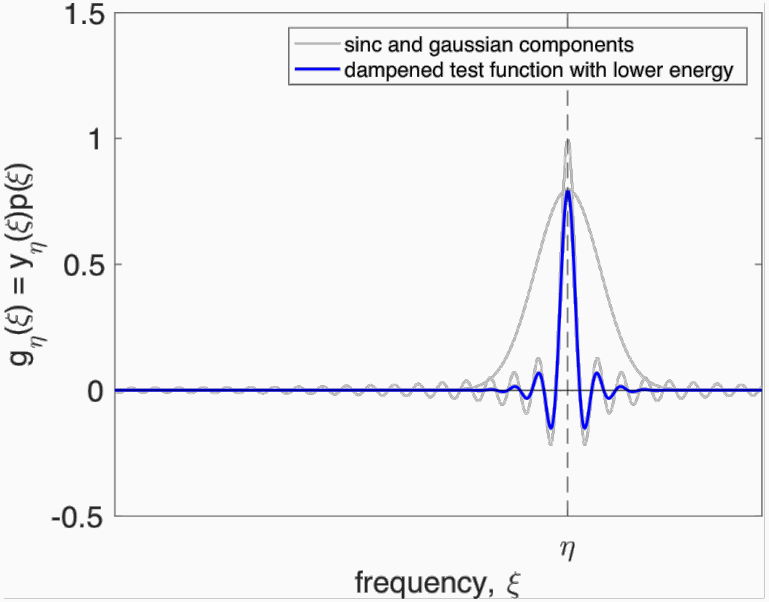
Solution: Dampen sinc with a narrow Gaussian.





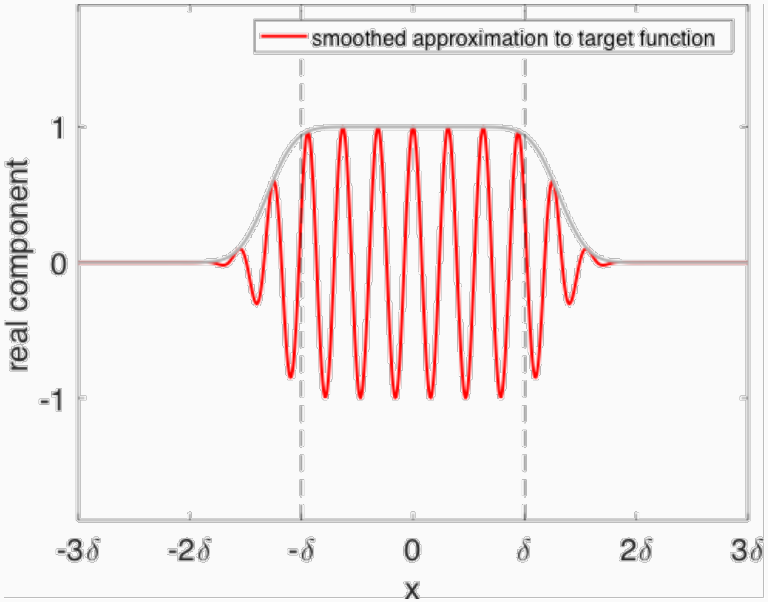
Solution: Dampen sinc with a narrow Gaussian.

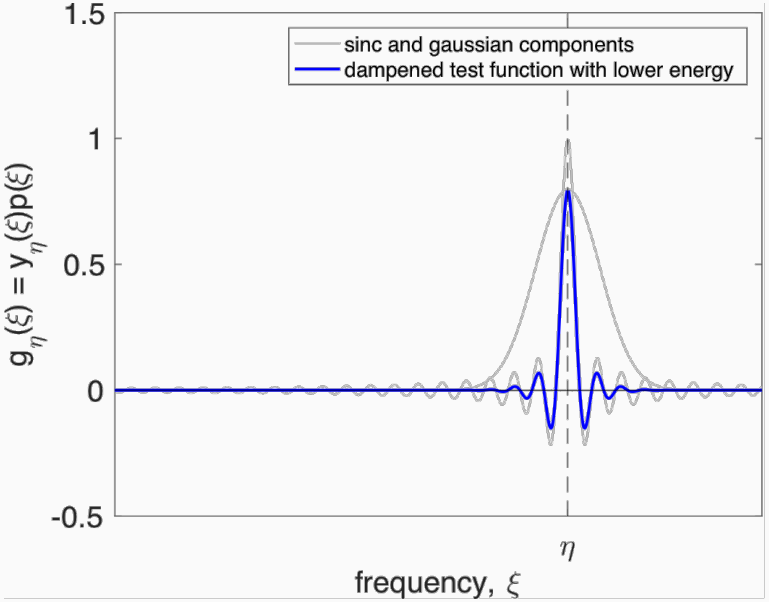




$\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2$ remains very small, $\|\mathbf{y}\|_2^2 \approx O(\delta)$.

Solution: Dampen sinc with a narrow Gaussian.





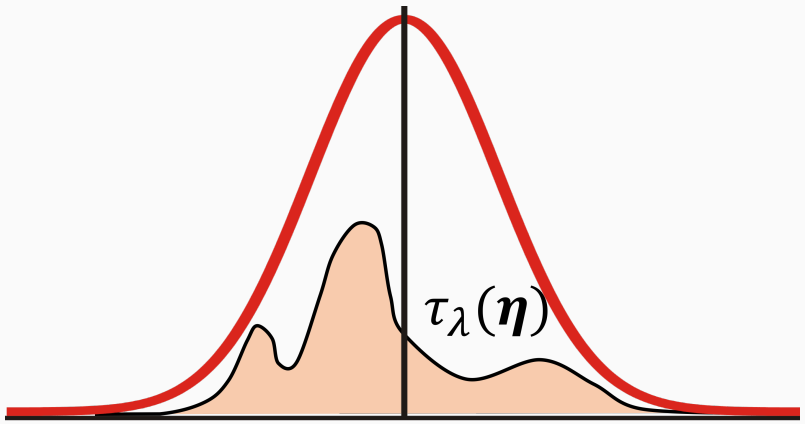
$\frac{1}{\lambda}\|\bar{\mathbf{\Phi}}\tilde{\mathbf{y}} - \bar{\mathbf{\Phi}}(\boldsymbol{\eta})\|_2^2$ remains very small, $\|\mathbf{y}\|_2^2 \approx O(\delta)$.

(as long as $\boldsymbol{\eta}$ is not too large)

FINAL TEST FUNCTION

Easy to sample from approximate leverage distribution for the Gaussian kernel with $x_1, \ldots, x_n \in [-\delta, \delta]^d$:

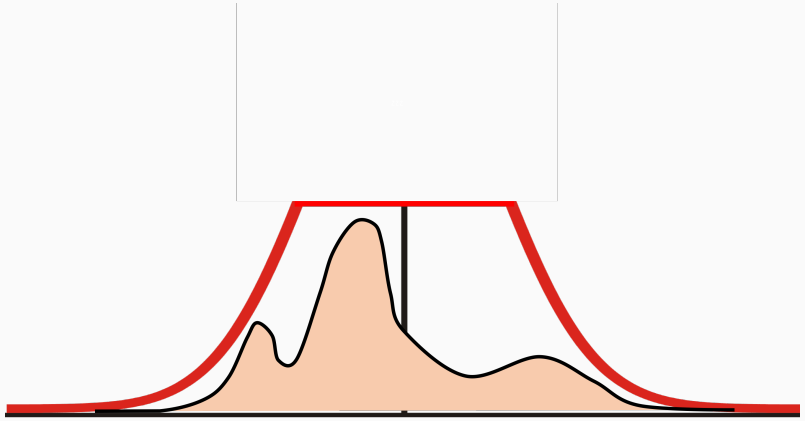
$$\bar{\tau}_\lambda(\boldsymbol{\eta}) = \begin{cases} \tilde{O}(\delta) \text{ when } \boldsymbol{\eta} \leq \sqrt{\log n/\lambda} \\ p(\boldsymbol{\eta}) = e^{-\|\boldsymbol{\eta}\|_2^2/2} \text{ otherwise.} \end{cases}$$



FINAL TEST FUNCTION

Easy to sample from approximate leverage distribution for the Gaussian kernel with $x_1, \ldots, x_n \in [-\delta, \delta]^d$:

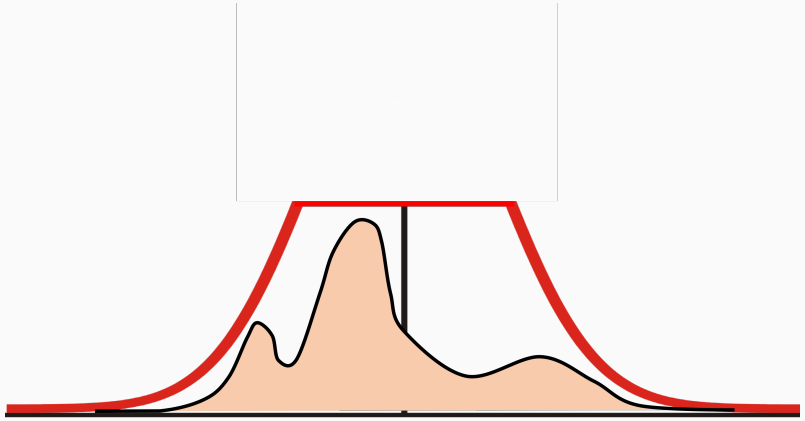
$$\bar{\tau}_\lambda(\boldsymbol{\eta}) = \begin{cases} \tilde{O}(\delta) \text{ when } \boldsymbol{\eta} \leq \sqrt{\log n/\lambda} \\ p(\boldsymbol{\eta}) = e^{-\|\boldsymbol{\eta}\|_2^2/2} \text{ otherwise.} \end{cases}$$



FINAL TEST FUNCTION

Easy to sample from approximate leverage distribution for the Gaussian kernel with $x_1, \ldots, x_n \in [-\delta, \delta]^d$:

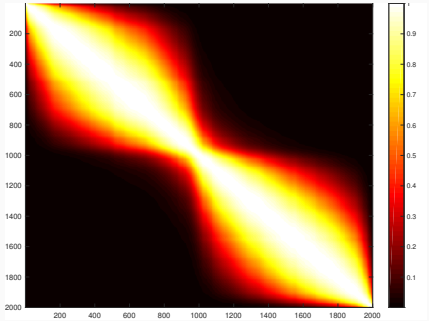
$$\bar{\tau}_\lambda(\boldsymbol{\eta}) = \begin{cases} \tilde{O}(\delta) \text{ when } \boldsymbol{\eta} \leq \sqrt{\log n/\lambda} \\ p(\boldsymbol{\eta}) = e^{-\|\boldsymbol{\eta}\|_2^2/2} \text{ otherwise.} \end{cases}$$



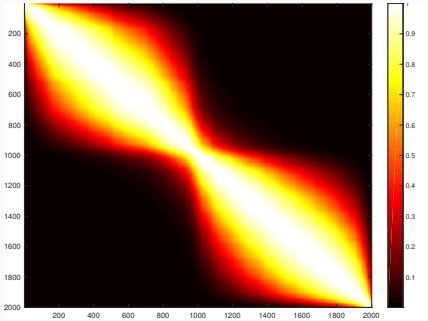
Requires $O(\delta\sqrt{\log(n/\lambda)}) \cdot \frac{1}{\epsilon^2}$ samples for spectral guarantee.

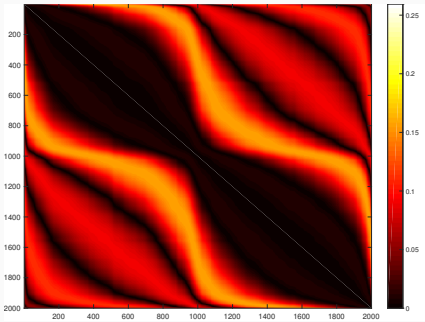
(vs. $O(n/\lambda) \cdot \frac{1}{\epsilon^2}$ for standard random Fourier features.)

Gaussian kernel for two clusters:

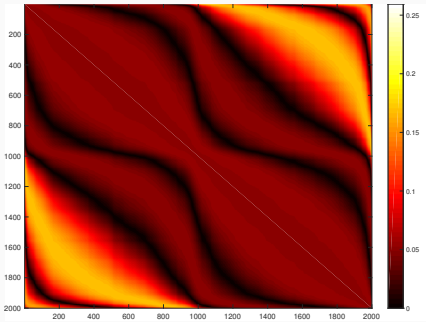


Gaussian kernel for two clusters:



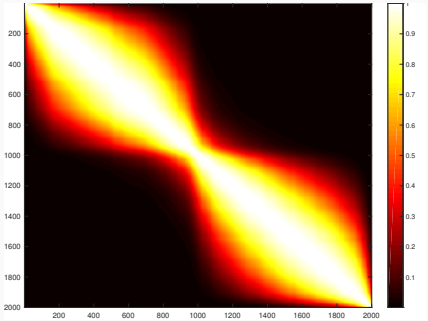


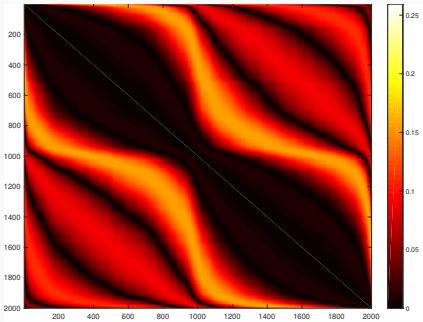
Standard RFF element error.

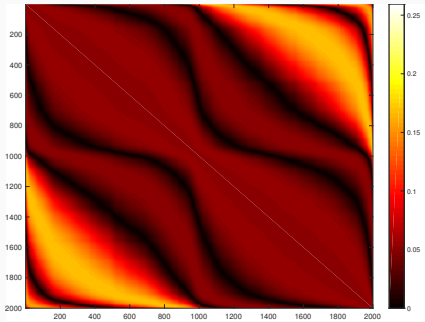


Modified RFF element error.

Gaussian kernel for two clusters:







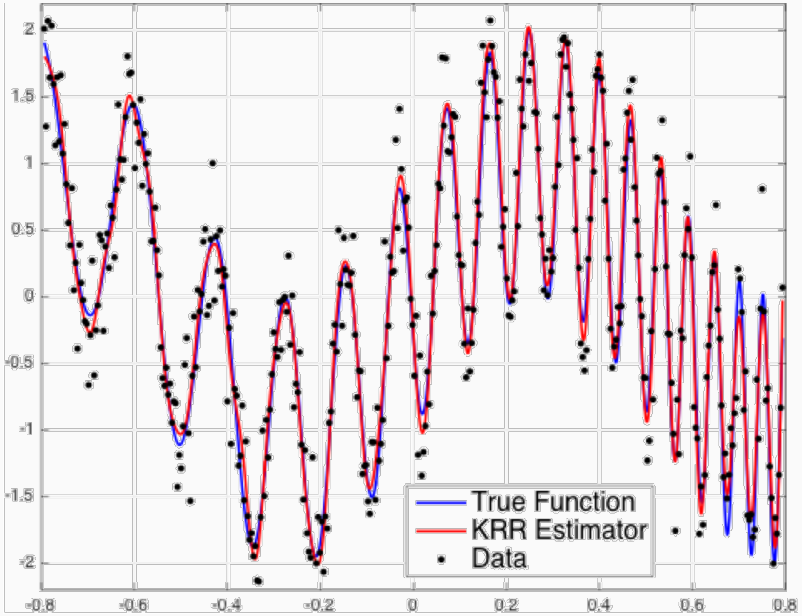
Standard RFF element error.

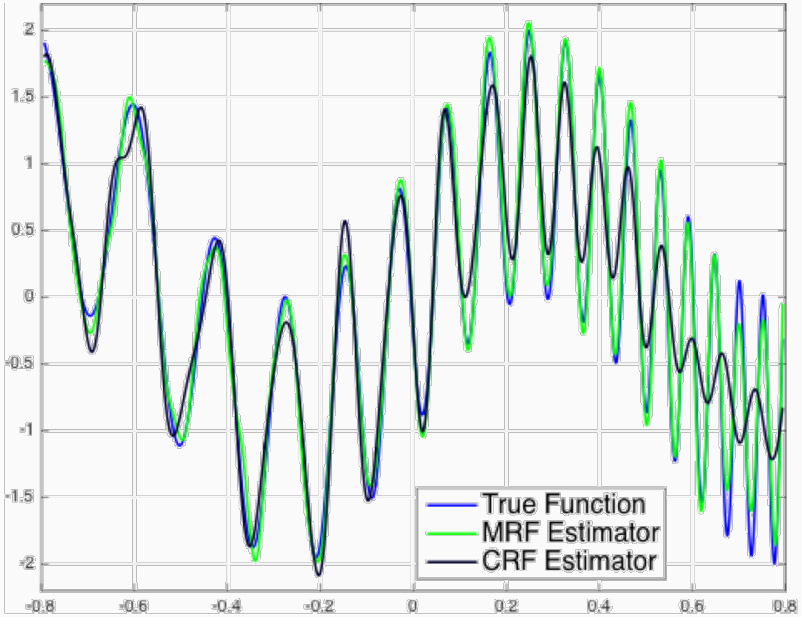
Modified RFF element error.

$$(1-\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}) \preceq \mathbf{K} + \lambda \mathbf{I} \preceq (1+\epsilon)(\mathbf{Z}\mathbf{Z}^* + \lambda \mathbf{I}).$$

Sampling low frequencies relatively less biases error to align with large eigenvectors.

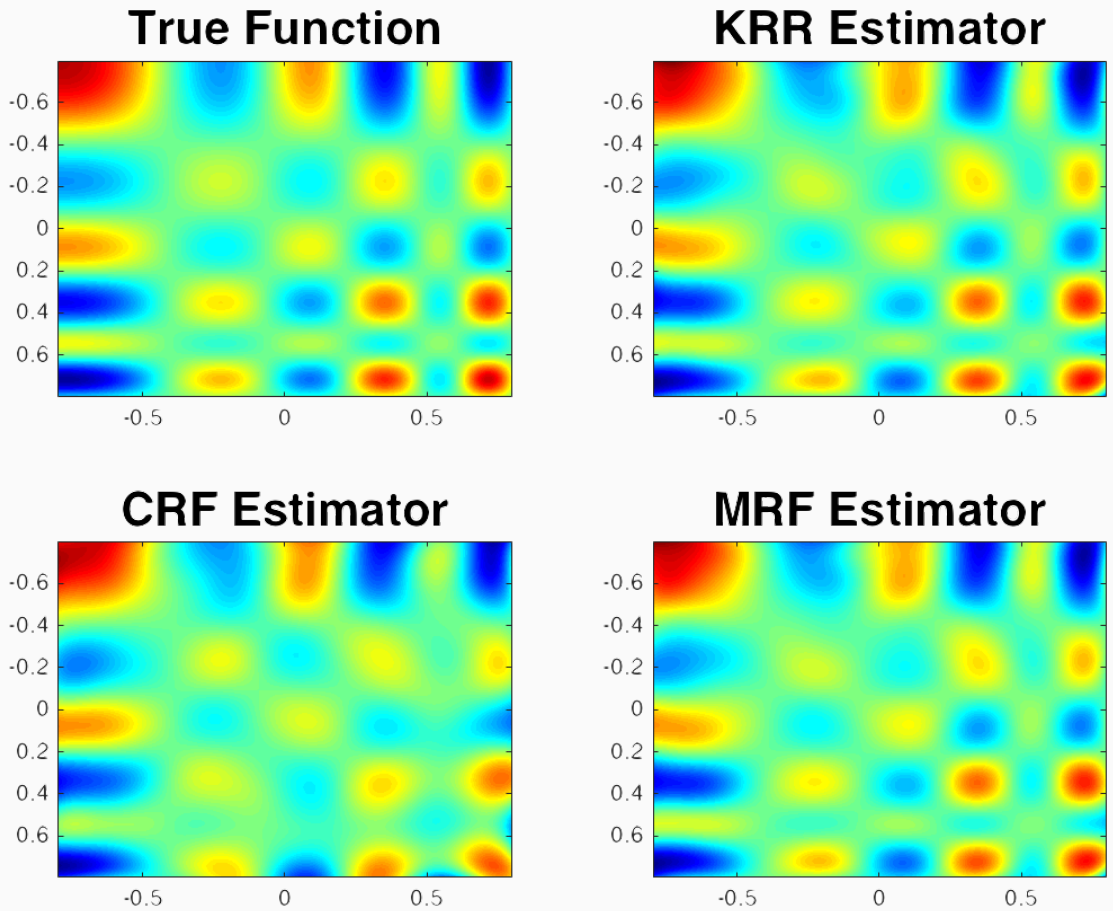
Example of approximate kernel ridge regression to interpolate a synthetic function:





CRF = classic random Fourier features ‘column norm’ sampling, MRF = our modified sampling distribution.

In higher dimensions:



CRF = classic random Fourier features ‘column norm’ sampling,
MRF = our modified sampling distribution.

SUMMARY

SUMMARY

1. Viewed random Fourier features methods as a matrix sampling problem.

SUMMARY

1. Viewed random Fourier features methods as a matrix sampling problem.
2. Used optimization perspective on leverage scores to certify upper bounds on these scores. Reduced score computation to Fourier approximation problem.

SUMMARY

1. Viewed random Fourier features methods as a matrix sampling problem.
2. Used optimization perspective on leverage scores to certify upper bounds on these scores. Reduced score computation to Fourier approximation problem.
3. New sampling distribution under-samples lower frequencies to obtain better kernel approximations.

Major open question: Can we achieve our spectral guarantee with $O(s_\lambda)$ samples in high dimensions for any data set.

OPEN QUESTIONS

Major open question: Can we achieve our spectral guarantee with $O(s_\lambda)$ samples <u>in high dimensions</u> for <u>any data set</u>.

Conjecture: Yes, although maybe with polynomial loss (i.e. $\mathrm{poly}(s_\lambda)$ samples).

Major open question: Can we achieve our spectral guarantee with $O(s_\lambda)$ samples in high dimensions for any data set.

Conjecture: Yes, although maybe with polynomial loss (i.e. $\text{poly}(s_\lambda)$ samples).

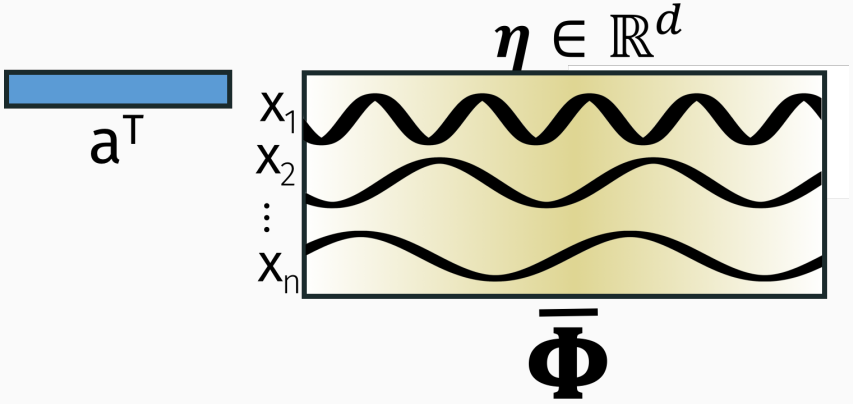
I.e. can we match data-adaptive methods like Nyström obliviously?

Duel view of leverage scores:

$$\tau_\lambda(\boldsymbol{\eta}) = \max_{\mathbf{a}} \frac{\left(\mathbf{a}^T \bar{\boldsymbol{\Phi}}(\boldsymbol{\eta})\right)^2}{\|\mathbf{a}^T \bar{\boldsymbol{\Phi}}\|_2^2 + \lambda \|\mathbf{a}\|_2^2}.$$

Duel view of leverage scores:

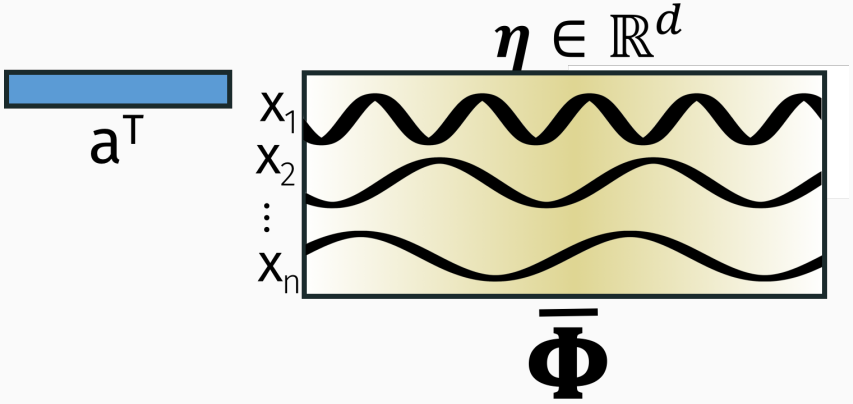
$$\tau_\lambda(\boldsymbol{\eta}) = \max_{\mathbf{a}} \frac{\left(\mathbf{a}^T \bar{\boldsymbol{\Phi}}(\boldsymbol{\eta})\right)^2}{\|\mathbf{a}^T \bar{\boldsymbol{\Phi}}\|_2^2 + \lambda \|\mathbf{a}\|_2^2}.$$



$\mathbf{a}^T \bar{\boldsymbol{\Phi}}$ is an n sparse Fourier function weighted by a Gaussian.

Duel view of leverage scores:

$$\tau_\lambda(\boldsymbol{\eta}) = \max_{\mathbf{a}} \frac{\left(\mathbf{a}^T \bar{\boldsymbol{\Phi}}(\boldsymbol{\eta})\right)^2}{\|\mathbf{a}^T \bar{\boldsymbol{\Phi}}\|_2^2 + \lambda \|\mathbf{a}\|_2^2}.$$



$\mathbf{a}^T \bar{\boldsymbol{\Phi}}$ is an n sparse Fourier function weighted by a Gaussian.

Immediately get leverage score bounds from bounds on smoothness of sparse Fourier functions, e.g. [Chen, Kane, Price, Song FOCS 2016].

Vague open question: Why does this all actually matter for function fitting?

Vague open question: Why does this all actually matter for function fitting?

We were surprised to beat random Fourier features on a kernel regression task.

THANK YOU!