Query Efficient Structured Matrix Learning

Conference on Learning Theory 2026

Pratyush Avi

New York University

Joint work with

Noah Amsel[1], Feyza Duman Keles[1], Tyler Chen[1], Chinmay Hegde[1], Cameron Musco[2], Christopher Musco[1], David Persson[3]

[1] New York University [2] UMass Amherst [3] Flatiron Institute

Structured Matrix Learning

Problem: Let $\boldsymbol{A}$ be some target matrix and let $\mathcal{F} \subset \mathbb{R}^{n \times n}$ be a family of $n \times n$ matrices. For approximation factor $\gamma > 1$, find $\tilde{\boldsymbol{B}} \in \mathcal{F}$ satisfying:

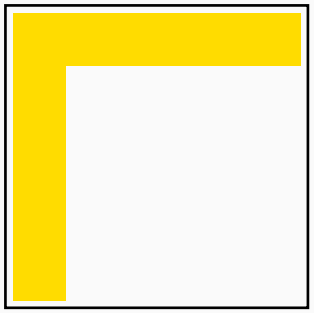
$$\left\|\boldsymbol{A} - \tilde{\boldsymbol{B}}\right\| \leq \gamma \cdot \min_{\boldsymbol{B} \in \mathcal{F}} \|\boldsymbol{A} - \boldsymbol{B}\|$$

Structured Matrix Learning

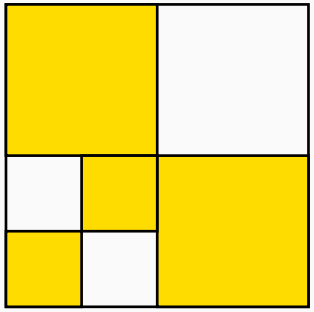
Problem: Let $\boldsymbol{A}$ be some target matrix and let $\mathcal{F} \subset \mathbb{R}^{n \times n}$ be a family of $n \times n$ matrices. For approximation factor $\gamma > 1$, find $\tilde{\boldsymbol{B}} \in \mathcal{F}$ satisfying:

$$\left\|\boldsymbol{A} - \tilde{\boldsymbol{B}}\right\| \leq \gamma \cdot \min_{\boldsymbol{B} \in \mathcal{F}} \|\boldsymbol{A} - \boldsymbol{B}\|$$

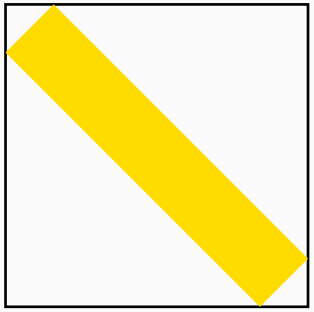
Example families:



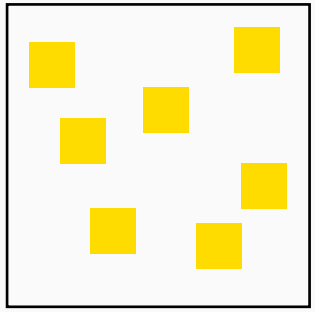
low-rank



hierarchically low-rank



diagonal



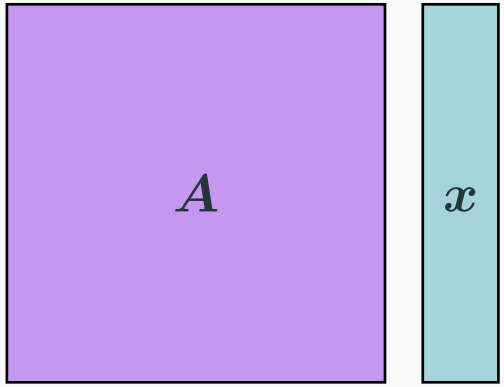
sparse

More e.g.: Toeplitz, butterfly, block-diagonal, tridiagonal, ...

Matrix Learning Access Model

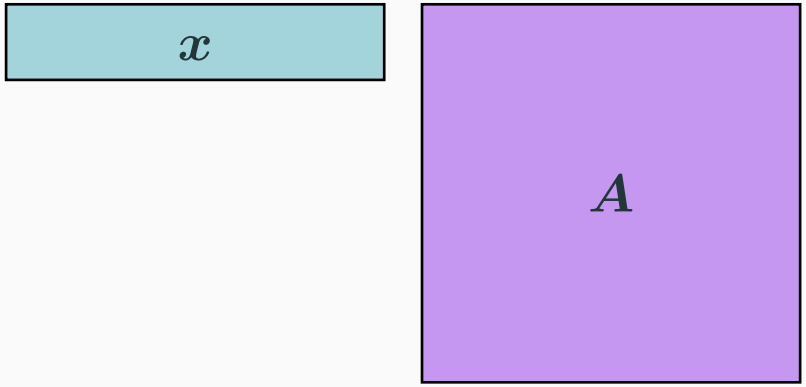
Say we are given access to $\boldsymbol{A}$ via matrix-vector product queries.

$$\boldsymbol{x} \longmapsto \boldsymbol{A}\boldsymbol{x}$$



right query

$$\boldsymbol{x} \longmapsto \boldsymbol{x}^\top \boldsymbol{A}$$

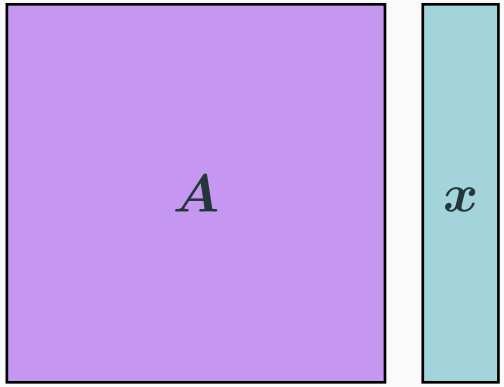


left query

Matrix Learning Access Model

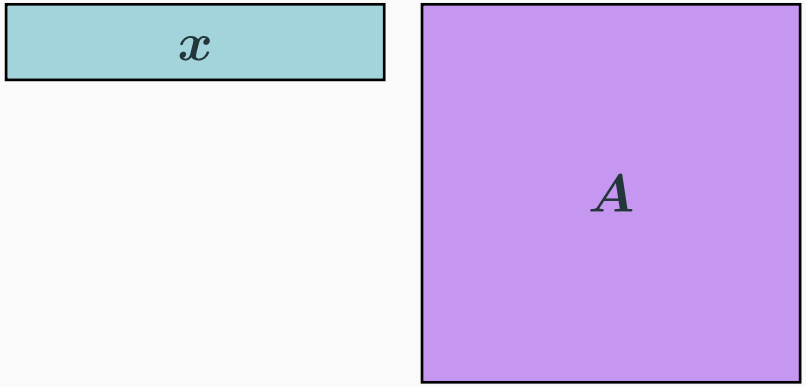
Say we are given access to $\boldsymbol{A}$ via matrix-vector product queries.

$\boldsymbol{x} \longmapsto \boldsymbol{A}\boldsymbol{x}$



right query

$\boldsymbol{x} \longmapsto \boldsymbol{x}^\top \boldsymbol{A}$



left query

Question: How many queries do we need to find a near-optimal approximation from some hypothesis class $\mathcal{F}$?

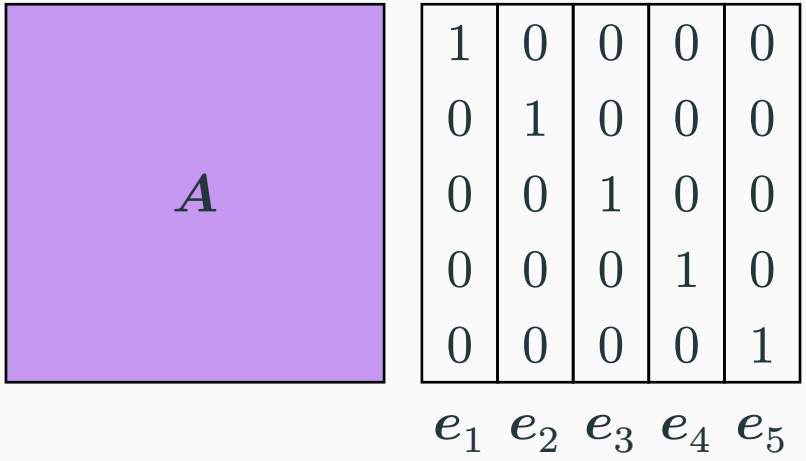
Motivation

In many cases it is natural to have matrix-vector product access to $\boldsymbol{A}$.

Matrix	Oracle for Matrix-Vector Product
Hessian matrix $\boldsymbol{H}$	Backpropagation to compute $\boldsymbol{Hx}$
Inverse matrix $\boldsymbol{B}^{-1}$	Iterative methods to compute $\boldsymbol{B}^{-1}\boldsymbol{x}$
Discretization of Linear PDE operator $\mathcal{L}$	PDE solver to compute $\mathcal{L}\boldsymbol{x}$ $(\boldsymbol{x}, \mathcal{A}\boldsymbol{x})$

Baseline: n queries

n matrix-vector product queries are always sufficient. Just multiply with the n standard basis vectors $\boldsymbol{e}_1, ..., \boldsymbol{e}_n$!



But we can do better...

Example Hypothesis Class - Diagonal Matrices

Let $\mathcal{F}$ be the class of diagonal matrices.

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix}$$

target matrix $\boldsymbol{A}$

We want to find $\tilde{\boldsymbol{B}} \in \mathcal{F}$ satisfying:

$$\left\|\boldsymbol{A} - \tilde{\boldsymbol{B}}\right\| \leq \gamma \cdot \min_{\boldsymbol{B} \in \mathcal{F}} \|\boldsymbol{A} - \boldsymbol{B}\|$$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix}$$

target matrix $\boldsymbol{A}$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix} \quad \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix}$$

target matrix $\boldsymbol{A}$ $\quad$ $\boldsymbol{x}$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix} \quad \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \quad = \quad \begin{bmatrix} 5 \\ -7.8 \\ 2.2 \\ 3 \\ -4 \end{bmatrix}$$

target matrix $\boldsymbol{A}$ $\quad$ $\boldsymbol{x}$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix} \quad \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \quad = \quad \begin{bmatrix} 5 \\ -7.8 \\ 2.2 \\ 3 \\ -4 \end{bmatrix}$$

target matrix $\boldsymbol{A}$ $\quad$ $\boldsymbol{x}$

$$\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x} = \sum_{j=1}^{n} x_j (\boldsymbol{A}\boldsymbol{x})_j =$$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix} \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} 5 \\ -7.8 \\ 2.2 \\ 3 \\ -4 \end{bmatrix}$$

target matrix $\boldsymbol{A}$ $\quad$ $\boldsymbol{x}$

$$\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x} = \sum_{j=1}^{n} x_j (\boldsymbol{A}\boldsymbol{x})_j = \begin{bmatrix} 5 \\ 7.8 \\ -2.2 \\ 3 \\ -4 \end{bmatrix}$$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix} \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} 5 \\ -7.8 \\ 2.2 \\ 3 \\ -4 \end{bmatrix}$$

target matrix $\boldsymbol{A}$ $\quad$ $\boldsymbol{x}$

$$\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x} = \sum_{j=1}^{n} x_j (\boldsymbol{A}\boldsymbol{x})_j = \begin{bmatrix} 5 \\ 7.8 \\ -2.2 \\ 3 \\ -4 \end{bmatrix}$$

Unbiased estimator: $\mathbb{E}\left[x_j (\boldsymbol{A}\boldsymbol{x})_j\right] = \sum_k A_{jk} \mathbb{E}\left[x_j x_k\right] = A_{jj}$

Example Hypothesis Class - Diagonal Matrices

Possible procedure [Bekas, Kokiopoulou, Saad 2007]:

- Draw query vector with random ± 1 entries, i.e., $\boldsymbol{x} \in \{-1, 1\}^n$
- Compute $\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x}$

$$\begin{bmatrix} 5 & .1 & .1 & .1 & .1 \\ .1 & 8 & .1 & .1 & .1 \\ .1 & .1 & -2 & .1 & .1 \\ .1 & .1 & .1 & 3 & .1 \\ .1 & .1 & .1 & .1 & -4 \end{bmatrix} \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} = \begin{bmatrix} 5 \\ -7.8 \\ 2.2 \\ 3 \\ -4 \end{bmatrix}$$

target matrix $\boldsymbol{A}$ $\quad$ $\boldsymbol{x}$

$$\boldsymbol{x} \circ \boldsymbol{A}\boldsymbol{x} = \sum_{j=1}^{n} x_j (\boldsymbol{A}\boldsymbol{x})_j = \begin{bmatrix} 5 \\ 7.8 \\ -2.2 \\ 3 \\ -4 \end{bmatrix}$$

Unbiased estimator: $\mathbb{E}\left[x_j (\boldsymbol{A}\boldsymbol{x})_j\right] = \sum_k A_{jk} \mathbb{E}\left[x_j x_k\right] = A_{jj}$

Repeat $1/\varepsilon$ and average to improve accuracy to get $\gamma = 1 + \varepsilon$ approximation

Existing Results for Popular Classes

Some known results for $\gamma = 1 + \varepsilon$

Structure	Query Complexity	Reference
Diagonal	$O(1/\varepsilon)$	[Bekas et al., 2007] [Dharangutte, Musco 2023]
Rank k	$O\left(k/\varepsilon^{1/3}\right)$	Randomized SVD, [Bakshi et al., 2022]
s-banded	$O(s/\varepsilon)$	[Amsel et al., 2026]
s-sparse rows	$O(s/\varepsilon)$	[Amsel et al., 2026]
rank-k HODLR	$O\left(k \log^4 n/\varepsilon^3\right)$	[Lin, Lu, Ying, 2011] [Chen et al., 2025]
$\vdots$	$\vdots$	$\vdots$

Towards a General Theory of Matrix Learning

Does there exist a general theory to characterize the query complexity of structured matrix approximation?

Something in the spirit of VC dimension, Rademacher complexity, and covering number.

Our Results

Baseline & Setup

We consider:

- Finite hypothesis classes, $\mathcal{F}$
- Perfect matrix-vector product queries

Baseline & Setup

We consider:

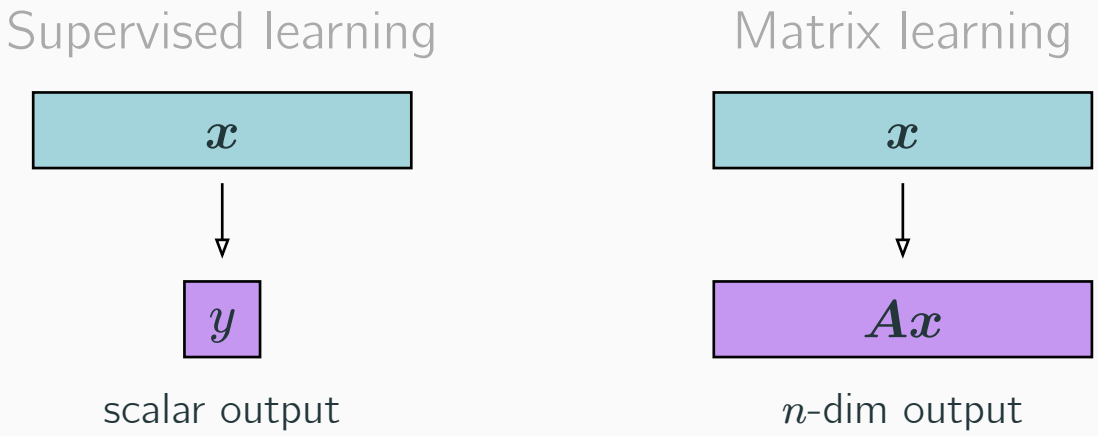
- Finite hypothesis classes, $\mathcal{F}$
- Perfect matrix-vector product queries

Naive result: $O(\log|\mathcal{F}| \ /\varepsilon^2)$ queries are sufficient

Imitates standard learning theory results about finite hypothesis classes.

Matrix Learning vs Supervised Learning

We get significantly more information per query as compared to standard supervised learning with scalar valued outputs.



e.g. of supervised learning setting: rank-1 matrix sensing

Main Result

The multi-output nature of the problem allows us a quadratic improvement in query complexity!

Theorem (Amsel, PA, Duman Keles, Chen, Hegde, Musco, Musco, Persson): Let $\mathcal{F}$ be a finite matrix family. A near-optimal approximation to $\boldsymbol{A}$ can be learned with accuracy 4 with

$$\tilde{O}\left(\sqrt{\log|\mathcal{F}|}\right) \text{ matrix-vectr product queries}$$

with high probability.

Main Result

The multi-output nature of the problem allows us a quadratic improvement in query complexity!

Theorem (Amsel, PA, Duman Keles, Chen, Hegde, Musco, Musco, Persson): Let $\mathcal{F}$ be a finite matrix family. A near-optimal approximation to $\boldsymbol{A}$ can be learned with accuracy 4 with

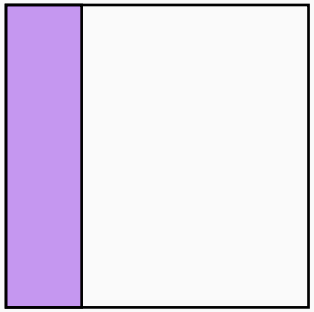
$$\tilde{O}\left(\sqrt{\log|\mathcal{F}|}\right) \text{ matrix-vectr product queries}$$

with high probability.

We show that this result is tight up to log log factors.

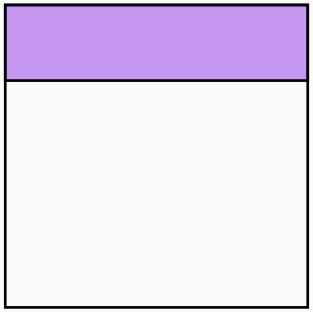
Key Idea

There is some power to using both $\boldsymbol{Ax}$ and $\boldsymbol{x}^\top \boldsymbol{A}$ queries.



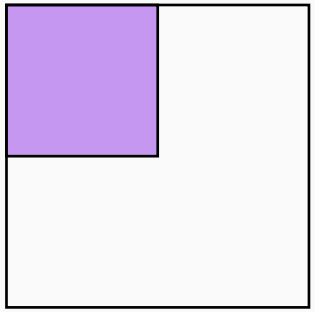
Single $\boldsymbol{Ax}$ query

But n $\boldsymbol{x}^\top \boldsymbol{A}$ queries!



n $\boldsymbol{Ax}$ queries

But single $\boldsymbol{x}^\top \boldsymbol{A}$ queries!



Hard case:

Requires $\sqrt{n}$ from both sides

Query Complexity via Main Result

Family	Family Size	Query Complexity
s-sparse	$2^{O(s)}$	$\tilde{O}(\sqrt{s})$
Constant Rank Butterfly	$2^{O(n)}$	$\tilde{O}(\sqrt{n})$
Diagonal	$2^{O(n)}$	$\tilde{O}(\sqrt{n})$
Rank-k	$2^{O(nk)}$	$\tilde{O}\left(\sqrt{nk}\right)$

Query Complexity via Main Result

Family	Family Size	Query Complexity
s-sparse	$2^{O(s)}$	$\tilde{O}(\sqrt{s})$
Constant Rank Butterfly	$2^{O(n)}$	$\tilde{O}(\sqrt{n})$
Diagonal	$2^{O(n)}$	$\tilde{O}(\sqrt{n})$
Rank-k	$2^{O(nk)}$	$\tilde{O}\left(\sqrt{nk}\right)$

This bound leads to tight results for some families, but loose results for others.

Thank you!

Paper link

https://arxiv.org/abs/2507.19290

Some open questions:

1. Class size may not be the best complexity measure. Is there something better that can achieve instance optimality?
2. Obtain $(1+\varepsilon)$ guarantee instead of constant factor
3. Extend this to other norms, e.g.: spectral norm