SUBLINEAR TIME LOW-RANK APPROXIMATION OF POSITIVE SEMIDEFINITE MATRICES

Cameron Musco (MIT) and David P. Woodruff (CMU)
Our Contributions:

• A near optimal low-rank approximation for any positive semidefinite (PSD) matrix can be computed in sublinear time (i.e. without reading the full matrix).
• Concrete: Significantly improves on previous, roughly linear time approaches for general matrices, and bypasses a trivial linear time lower bound for general matrices.
• High Level: Demonstrates that PSD structure can be exploited in a much stronger way than previously known for low-rank approximation. Opens the possibility of further advances in algorithms for PSD matrices.
Our Contributions:

- A near optimal low-rank approximation for any positive semidefinite (PSD) matrix can be computed in sublinear time (i.e. without reading the full matrix).
Our Contributions:

- A near optimal low-rank approximation for any positive semidefinite (PSD) matrix can be computed in sublinear time (i.e. without reading the full matrix).
- **Concrete**: Significantly improves on previous, roughly linear time approaches for general matrices, and bypasses a trivial linear time lower bound for general matrices.
Our Contributions:

- A near optimal low-rank approximation for any positive semidefinite (PSD) matrix can be computed in sublinear time (i.e. without reading the full matrix).
- **Concrete**: Significantly improves on previous, roughly linear time approaches for general matrices, and bypasses a trivial linear time lower bound for general matrices.
- **High Level**: Demonstrates that PSD structure can be exploited in a much stronger way than previously known for low-rank approximation. Opens the possibility of further advances in algorithms for PSD matrices.
Low-rank approximation is one of the most widely used methods for general matrix and data compression.
Low-rank approximation is one of the most widely used methods for general matrix and data compression.

\[
\begin{align*}
A & \rightarrow N \otimes M^T \\
\text{n x d} & \rightarrow \text{n x k} \times \text{k x d}
\end{align*}
\]
Low-rank approximation is one of the most widely used methods for general matrix and data compression.

- Closely related to principal component analysis, spectral embedding/clustering, and low-rank matrix completion.
Low-rank approximation is one of the most widely used methods for general matrix and data compression.

• Closely related to principal component analysis, spectral embedding/clustering, and low-rank matrix completion.

Important Special Case: $A$ is positive semidefinite (PSD). I.e.

$$x^T A x \geq 0, \forall x \in \mathbb{R}^n$$
Low-rank approximation is one of the most widely used methods for general matrix and data compression.

• Closely related to principal component analysis, spectral embedding/clustering, and low-rank matrix completion.

Important Special Case: $A$ is positive semidefinite (PSD). I.e.

$$x^T Ax \geq 0, \forall x \in \mathbb{R}^n$$

• Includes graph Laplacians, Gram matrices and kernel matrices, covariance matrices, Hessians for convex functions.
An optimal low-rank approximation can be computed via the singular value decomposition (SVD).
An optimal low-rank approximation can be computed via the singular value decomposition (SVD).

$$A_{k} = \arg \min_{B} \text{rank}(B) = k \quad A_{k} B_{k} F = \sum_{i=1}^{k} \sigma_{i} u_{i} v_{i}^T$$

Unfortunately, computing the SVD takes $O(n^2 d^2)$ time.
An optimal low-rank approximation can be computed via the singular value decomposition (SVD).
An optimal low-rank approximation can be computed via the singular value decomposition (SVD).

\[ A_k = \arg \min_{B: \text{rank}(B) = k} \| A - B \|_F \]
An optimal low-rank approximation can be computed via the singular value decomposition (SVD).

\[ \mathbf{A}_k = \arg \min_{\mathbf{B}: \text{rank}(\mathbf{B}) = k} \| \mathbf{A} - \mathbf{B} \|_F = \sqrt{\sum_{i,j} (A_{ij} - B_{ij})^2} \]
An optimal low-rank approximation can be computed via the singular value decomposition (SVD).

\[ A_k = \arg \min_{B: \text{rank}(B)=k} \| A - B \|_F = \sqrt{\sum_{i,j} (A_{ij} - B_{ij})^2} \]

- Unfortunately, computing the SVD takes \( O(nd^2) \) time.
Recent work on matrix sketching gives state-of-the-art runtimes
Recent work on matrix sketching gives state-of-the-art runtimes

**Theorem (Clarkson, Woodruﬀ’13)**

*There is an algorithm which in $O(\text{nnz}(A) + n \cdot \text{poly}(k, 1/\epsilon))$ time outputs $N \in \mathbb{R}^{n \times k}, M \in \mathbb{R}^{d \times k}$ satisfying with prob. $99/100$:*

$$\|A - NM^T\|_F \leq (1 + \epsilon)\|A - A_k\|_F.$$
Recent work on matrix sketching gives state-of-the-art runtimes

**Theorem (Clarkson, Woodruff ’13)**

There is an algorithm which in $O(\text{nnz}(A) + n \cdot \text{poly}(k, 1/\epsilon))$ time outputs $N \in \mathbb{R}^{n \times k}, M \in \mathbb{R}^{d \times k}$ satisfying with prob. 99/100:

$$\|A - NM^T\|_F \leq (1 + \epsilon)\|A - A_k\|_F.$$  

- When $k, 1/\epsilon$ are not too large, runtime is linear in input size.
Recent work on matrix sketching gives state-of-the-art runtimes

Theorem (Clarkson, Woodruff ’13)

There is an algorithm which in \(O(\text{nnz}(A) + n \cdot \text{poly}(k, 1/\epsilon))\) time outputs \(N \in \mathbb{R}^{n \times k}, M \in R^{d \times k}\) satisfying with prob. 99/100:

\[
\|A - NM^T\|_F \leq (1 + \epsilon)\|A - A_k\|_F.
\]

- When \(k, 1/\epsilon\) are not too large, runtime is linear in input size.
- Best known runtime for both general and PSD matrices.
Theorem (Main Result – Musco, Woodruff ‘17)

There is an algorithm running in \( \tilde{O} \left( \frac{nk^2}{\epsilon^4} \right) \) time which, given PSD \( A \), outputs \( N, M \in \mathbb{R}^{n \times k} \) satisfying with probability 99/100:

\[
\| A - NM^T \|_F \leq (1 + \epsilon) \| A - A_k \|_F.
\]
Theorem (Main Result – Musco, Woodruff ‘17)

There is an algorithm running in $\tilde{O}\left(\frac{n k^2}{\epsilon^4}\right)$ time which, given PSD $A$, outputs $N, M \in \mathbb{R}^{n \times k}$ satisfying with probability 99/100:

$$\|A - NM^T\|_F \leq (1 + \epsilon)\|A - A_k\|_F.$$  

• Compare to CW‘13 which takes $O(\text{nnz}(A)) + n \cdot \text{poly}(k, 1/\epsilon)$.  

SUBLINEAR TIME LOW-RANK APPROXIMATION
Theorem (Main Result – Musco, Woodruff ‘17)

There is an algorithm running in $\tilde{O}\left(\frac{nk^2}{\epsilon^4}\right)$ time which, given PSD $A$, outputs $N, M \in \mathbb{R}^{n \times k}$ satisfying with probability $99/100$:

$$\|A - NM^T\|_F \leq (1 + \epsilon)\|A - A_k\|_F.$$ 

• Compare to CW‘13 which takes $O(\text{nnz}(A)) + n \cdot \text{poly}(k, 1/\epsilon)$. 

SUBLINEAR TIME LOW-RANK APPROXIMATION
Theorem (Main Result – Musco, Woodruff ‘17)

There is an algorithm running in $\tilde{O}\left(\frac{nk^2}{\epsilon^4}\right)$ time which, given $PSD$ $A$, outputs $N, M \in \mathbb{R}^{n \times k}$ satisfying with probability $99/100$:

$$\|A - NM^T\|_F \leq (1 + \epsilon)\|A - A_k\|_F.$$ 

- Compare to CW‘13 which takes $O(\text{nnz}(A)) + n \cdot \text{poly}(k, 1/\epsilon)$.
- If $k, 1/\epsilon$ are not too large compared to $\text{nnz}(A)$, our runtime is significantly sublinear in the size of $A$.  


For general matrices, $\Omega(\text{nnz}(A))$ time is required.
For general matrices, $\Omega(\text{nnz}(A))$ time is required.

- Randomly place a single entry which dominates $A$’s Frobenius norm.
For general matrices, $\Omega(\text{nnz}(A))$ time is required.

- Randomly place a single entry which dominates $A$’s Frobenius norm.
For general matrices, $\Omega(\text{nnz}(A))$ time is required.

- Randomly place a single entry which dominates $A$’s Frobenius norm.
- Finding it with constant probability requires reading at least a constant fraction of the non-zero entries in $A$. 

![Matrix Diagram](image.png)
For general matrices, $\Omega(\text{nnz}(A))$ time is required.

- Randomly place a single entry which dominates $A$’s Frobenius norm.
- Finding it with constant probability requires reading at least a constant fraction of the non-zero entries in $A$.

- Lower bound holds for any approximation factor and even rules out $o(\text{nnz}(A))$ time for weaker guarantees.
For general matrices, $\Omega(\text{nnz}(A))$ time is required.

- Randomly place a single entry which dominates $A$’s Frobenius norm.
- Finding it with constant probability requires reading at least a constant fraction of the non-zero entries in $A$.

- Lower bound holds for any approximation factor and even rules out $o(\text{nnz}(A))$ time for weaker guarantees.

\[ \| A - NM^T \|_F \leq (1 + \epsilon) \| A - A_k \|_F \]
For general matrices, $\Omega(\text{nnz}(A))$ time is required.

- Randomly place a single entry which dominates $A$’s Frobenius norm.
- Finding it with constant probability requires reading at least a constant fraction of the non-zero entries in $A$.

- Lower bound holds for any approximation factor and even rules out $o(\text{nnz}(A))$ time for weaker guarantees.

\[ \| A - NM^T \|_F \leq \| A - A_k \|_F + \epsilon \| A \|_F. \]
Observation: For PSD $A$, we have for any entry $a_{ij}$:

$$a_{ij} \leq \max(a_{ii}, a_{jj})$$

since otherwise $(e_i - e_j)^T A (e_i - e_j) < 0.$
Observation: For PSD $A$, we have for any entry $a_{ij}$:

$$a_{ij} \leq \max(a_{ii}, a_{jj})$$

since otherwise $(e_i - e_j)^T A (e_i - e_j) < 0$.

- So we can find any ‘hidden’ heavy entry by looking at its corresponding diagonal entries.
Observation: For PSD $A$, we have for any entry $a_{ij}$:

$$a_{ij} \leq \max(a_{ii}, a_{jj})$$

since otherwise $(e_i - e_j)^T A (e_i - e_j) < 0$.

• So we can find any ‘hidden’ heavy entry by looking at its corresponding diagonal entries.

Question: How can we exploit additional structure arising from positive semidefiniteness to achieve sublinear runtime?
Very Simple Fact: Every PSD matrix $A \in \mathbb{R}^{n \times n}$ can be written as $B^T B$ for some $B \in \mathbb{R}^{n \times n}$. 
**Very Simple Fact:** Every PSD matrix $A \in \mathbb{R}^{n \times n}$ can be written as $B^T B$ for some $B \in \mathbb{R}^{n \times n}$.

- $B$ can be any matrix square root of $A$, e.g. if we let $V \Sigma V^T$ be the eigendecomposition of $A$, we can set $B = \Sigma^{1/2} V^T$. 

**Very Simple Fact:** Every PSD matrix $A \in \mathbb{R}^{n \times n}$ can be written as $B^T B$ for some $B \in \mathbb{R}^{n \times n}$.

- $B$ can be any matrix square root of $A$, e.g. if we let $V \Sigma V^T$ be the eigendecomposition of $A$, we can set $B = \Sigma^{1/2} V^T$.
- Letting $b_1, \ldots, b_n$ be the columns of $B$, the entries of $A$ contain every pairwise dot product $a_{ij} = b_i^T b_j$. 

![Diagram showing matrices and dot products](image)
The fact that $\mathbf{A}$ is a Gram matrix places a variety of geometric constraints on its entries.
EVERY PSD MATRIX IS A GRAM MATRIX

The fact that $A$ is a Gram matrix places a variety of geometric constraints on its entries.

- The heavy diagonal observation is just one example. By Cauchy-Schwarz:

\[ a_{ij} = b_i^T b_j \leq \sqrt{(b_i^T b_i) \cdot (b_j^T b_j)} = \sqrt{a_{ii} \cdot a_{jj}} \leq \max(a_{ii}, a_{jj}). \]
The fact that $A$ is a Gram matrix places a variety of geometric constraints on its entries.

- The heavy diagonal observation is just one example. By Cauchy-Schwarz:

$$a_{ij} = b_i^T b_j \leq \sqrt{(b_i^T b_i) \cdot (b_j^T b_j)} = \sqrt{a_{ii} \cdot a_{jj}} \leq \max(a_{ii}, a_{jj}).$$

**Another View:** $A$ contains a lot of information about the column span of $B$ in a very compressed form — with every pairwise dot product stored as $a_{ij}$. 
**Question:** Can we compute a low-rank approximation of $\mathbf{B}$ using $o(n^2)$ column dot products? I.e. $o(n^2)$ accesses to $\mathbf{A}$?
**Question:** Can we compute a low-rank approximation of \( B \) using \( o(n^2) \) column dot products? I.e. \( o(n^2) \) accesses to \( A \)?

**Why?** \( B \) has the same (right) singular vectors as \( A \), and its singular values are closely related: \( \sigma_i(B) = \sqrt{\sigma_i(A)} \).
Question: Can we compute a low-rank approximation of $B$ using $o(n^2)$ column dot products? I.e. $o(n^2)$ accesses to $A$?

Why? $B$ has the same (right) singular vectors as $A$, and its singular values are closely related: $\sigma_i(B) = \sqrt{\sigma_i(A)}$.

- So the top $k$ singular vectors are the same for the two matrices. An optimal low-rank approximation for $B$ thus gives an optimal low-rank approximation for $A$. 
**Question:** Can we compute a low-rank approximation of $B$ using $o(n^2)$ column dot products? I.e. $o(n^2)$ accesses to $A$?

**Why?** $B$ has the same (right) singular vectors as $A$, and its singular values are closely related: $\sigma_i(B) = \sqrt{\sigma_i(A)}$.

- So the top $k$ singular vectors are the same for the two matrices. An **optimal** low-rank approximation for $B$ thus gives an optimal low-rank approximation for $A$.
- Things will be messier once we introduce approximation, but this simple idea will lead to a sublinear time algorithm for $A$. 


Theorem (Deshpande, Vempala '06)

For any $B \in \mathbb{R}^{n \times n}$, there exists a subset of $\tilde{O}(\frac{k^2}{\epsilon})$ columns whose span contains $Z \in \mathbb{R}^{n \times k}$ satisfying:

$$k_B Z Z^T B k_F \leq (1 + \epsilon) k_B Z k_F.$$

Adaptive Sampling

Initially, start with an empty column subset $S := \{\}$. For $t = 1, \ldots, \tilde{O}(\frac{k^2}{\epsilon})$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability $\frac{k b_i}{P_S b_i k_F}$. 

$$11$$
Theorem (Deshpande, Vempala ‘06)

For any $\mathbf{B} \in \mathbb{R}^{n \times n}$, there exists a subset of $\tilde{O}(k^2 / \epsilon)$ columns whose span contains $\mathbf{Z} \in \mathbb{R}^{n \times k}$ satisfying:

$$
\| \mathbf{B} - \mathbf{Z} \mathbf{Z}^T \mathbf{B} \|_F \leq (1 + \epsilon) \| \mathbf{B} - \mathbf{B}_k \|_F
$$
Theorem (Deshpande, Vempala ‘06)

For any $B \in \mathbb{R}^{n \times n}$, there exists a subset of $\tilde{O}(k^2/\epsilon)$ columns whose span contains $Z \in \mathbb{R}^{n \times k}$ satisfying:

$$\|B - ZZ^T B\|_F \leq (1 + \epsilon)\|B - B_k\|_F$$

Adaptive Sampling

Initially, start with an empty column subset $S := \{}$. For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

- Let $P_S$ be the projection onto the columns in $S$.
- Add $b_i$ to $S$ with probability $\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^{n} \|b_i - P_S b_i\|^2}$. 
Theorem (Deshpande, Vempala ‘06)

For any $\mathbf{B} \in \mathbb{R}^{n \times n}$, there exists a subset of $\tilde{O}(k^2/\epsilon)$ columns whose span contains $\mathbf{Z} \in \mathbb{R}^{n \times k}$ satisfying:

$$\|\mathbf{B} - \mathbf{ZZ}^T \mathbf{B}\|_F \leq (1 + \epsilon) \|\mathbf{B} - \mathbf{B}_k\|_F$$

Adaptive Sampling

Initially, start with an empty column subset $S := \{\}$. For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

- Let $\mathbf{P}_S$ be the projection onto the columns in $S$.
- Add $\mathbf{b}_i$ to $S$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_S \mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i - \mathbf{P}_S \mathbf{b}_i\|^2}$. 
Adaptive Sampling

Initially, start with an empty column subset $S := \{\}$. For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$. Add $b_i$ to $S$ with probability

$$\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2}.$$
Adaptive Sampling

Initially, start with an empty column subset $S := \{\}$.

For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability

$$\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2} = \frac{\|b_i\|^2}{\sum_{i=1}^n \|b_i\|^2} = \frac{a_{ii}}{\text{tr}(A)}.$$
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$.

For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$. Add $b_i$ to $S$ with probability

$\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2} = \frac{\|b_i\|^2}{\sum_{i=1}^n \|b_i\|^2} = \frac{a_{ii}}{\text{tr}(A)}$. 
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$.

For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability

$$\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2} = \frac{\|b_i\|^2}{\sum_{i=1}^n \|b_i\|^2} = \frac{a_{ii}}{\text{tr}(A)}.$$
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$. For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability $\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^{n} \|b_i - P_S b_i\|^2}$.
Adaptive Sampling
Initially, start with an empty column subset $S := \emptyset$.
For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$
Let $P_S$ be the projection onto the columns in $S$.
Add $b_i$ to $S$ with probability
$$\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2}.$$
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$.

For $t = 1, \ldots, \tilde{O}(k^2/\varepsilon)$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability $rac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2}$.
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$.

For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability $\frac{||b_i - P_S b_i||^2}{\sum_{i=1}^n ||b_i - P_S b_i||^2}$.
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$.

For $t = 1, \ldots, \tilde{O}(k^2/\varepsilon)$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability

$$\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2}.$$
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$. For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$. Add $b_i$ to $S$ with probability $\frac{\|b_i - P_Sb_i\|^2}{\sum_{i=1}^{n} \|b_i - P_Sb_i\|^2}$. 
Adaptive Sampling
Initially, start with an empty column subset $S := \emptyset$.
For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$
Let $P_S$ be the projection onto the columns in $S$.
Add $b_i$ to $S$ with probability $\frac{\|b_i - P_Sb_i\|^2}{\sum_{i=1}^n \|b_i - P_Sb_i\|^2}$.
Adaptive Sampling

Initially, start with an empty column subset $S := \emptyset$.

For $t = 1, \ldots, \tilde{O}(k^2/\epsilon)$

Let $P_S$ be the projection onto the columns in $S$.

Add $b_i$ to $S$ with probability

$$\frac{\|b_i - P_S b_i\|^2}{\sum_{i=1}^n \|b_i - P_S b_i\|^2}.$$
Theorem (Factor Matrix Low-Rank Approximation)

There is an algorithm using $\tilde{O}(nk^2/\epsilon)$ accesses to $A = B^T B$ which computes $Z \in \mathbb{R}^{n \times k}$ satisfying with probability $99/100$:

$$\|B - ZZ^TB\|_F \leq (1 + \epsilon)\|B - B_k\|_F.$$
Theorem (Factor Matrix Low-Rank Approximation)

There is an algorithm using $\tilde{O}(nk^2/\epsilon)$ accesses to $A = B^TB$ which computes $Z \in \mathbb{R}^{n \times k}$ satisfying with probability $99/100$:

$$\|B - ZZ^TB\|_F \leq (1 + \epsilon)\|B - B_k\|_F.$$

- How does this translate to low-rank approximation of $A$ itself?
Lemma

If \( \| B - ZZ^TB \|_F^2 \leq \left( 1 + \frac{\epsilon^{3/2}}{\sqrt{n}} \right) \| B - B_k \|_F^2 \), then for \( A = B^TB \):

\[
\| A - B^TZZ^TB \|_F^2 \leq (1 + \epsilon)\| A - A_k \|_F^2.
\]
Lemma

If \( \| B - ZZ^T B \|_F^2 \leq \left( 1 + \frac{\epsilon^{3/2}}{\sqrt{n}} \right) \| B - B_k \|_F^2 \), then for \( A = B^T B \):

\[
\| A - B^T ZZ^T B \|_F^2 \leq (1 + \epsilon) \| A - A_k \|_F^2.
\]
Lemma

If \( \| B - ZZ^T B \|_F^2 \leq \left( 1 + \frac{\epsilon^{3/2}}{\sqrt{n}} \right) \| B - B_k \|_F^2 \), then for \( A = B^T B \):

\[
\| A - ASCS^T A^T \|_F^2 \leq (1 + \epsilon) \| A - A_k \|_F^2.
\]
Lemma

If \( \|B - ZZ^T B\|_F^2 \leq \left(1 + \frac{\epsilon^{3/2}}{\sqrt{n}}\right) \|B - B_k\|_F^2 \), then for \( A = B^T B \):

\[
\|A - ASCS^T A^T\|_F^2 \leq (1 + \epsilon)\|A - A_k\|_F^2.
\]

- This gives a low-rank approximation algorithm which accesses just \( \tilde{O}\left(\frac{nk^2}{\epsilon^{3/2}/\sqrt{n}}\right) = n^{3/2} \cdot \text{poly}(k, 1/\epsilon) \) entries of \( A \).
Lemma

If \( \|B - ZZ^T B\|_F^2 \leq \left(1 + \frac{\epsilon^{3/2}}{\sqrt{n}}\right) \|B - B_k\|_F^2 \), then for \( A = B^T B \):

\[
\|A - ASCS^T A^T\|_F^2 \leq (1 + \epsilon)\|A - A_k\|_F^2.
\]

- This gives a low-rank approximation algorithm which accesses just \( \tilde{O}\left(\frac{nk^2}{\epsilon^{3/2}/\sqrt{n}}\right) = n^{3/2} \cdot \text{poly}(k, 1/\epsilon) \) entries of \( A \).
- Our best algorithm accesses just \( \tilde{O}\left(\frac{nk}{\epsilon^{2.5}}\right) \) entries of \( A \) and runs in \( \tilde{O}\left(\frac{nk^2}{\epsilon^4}\right) \) time.
Recall that our algorithm accesses the diagonal of $\mathbf{A}$ along with $\tilde{O}(k^2/\epsilon)$ columns.
Recall that our algorithm accesses the diagonal of $A$ along with $\tilde{O}(k^2\sqrt{n})$ columns.
Recall that our algorithm accesses the diagonal of $A$ along with $\tilde{O}(k^2 \sqrt{n})$ columns.
Recall that our algorithm accesses the diagonal of $A$ along with $\tilde{O}(k^2 \sqrt{n})$ columns.

- If we take fewer columns, we can miss a $\sqrt{n} \times \sqrt{n}$ block which contains a constant fraction of $A$'s Frobenius norm.
Solution: Sample both rows and columns of $A$. 
Solution: Sample both rows and columns of $A$.

- Instead of adaptive sampling we use ridge leverage scores, which can also be computed using an iterative sampling scheme making $\tilde{O}(nk)$ accesses to $A$ (Musco, Musco ’17).
Solution: Sample both rows and columns of $A$.

- Instead of adaptive sampling we use ridge leverage scores, which can also be computed using an iterative sampling scheme making $\tilde{O}(nk)$ accesses to $A$ (Musco, Musco ’17).
- Same intuition – select a diverse set of columns which span a near-optimal low-rank approximation of the matrix.
Solution: Sample both rows and columns of $\mathbf{A}$.

- Instead of adaptive sampling we use ridge leverage scores, which can also be computed using an iterative sampling scheme making $\tilde{O}(nk)$ accesses to $\mathbf{A}$ (Musco, Musco ’17).
- Same intuition – select a diverse set of columns which span a near-optimal low-rank approximation of the matrix.
- Sample $\mathbf{A} \mathbf{S}$ is a projection-cost-preserving sketch for $\mathbf{A}$ [Cohen et al ’15,’17]. For any rank-$k$ projection $\mathbf{P}$,

\[ \| \mathbf{A} \mathbf{S} - \mathbf{P} \mathbf{A} \mathbf{S} \|^2_F = (1 \pm \epsilon) \| \mathbf{A} - \mathbf{P} \mathbf{A} \|^2_F. \]
Recover low-rank approximation using two-sided sampling and projection-cost-preserving sketch property.
Recover low-rank approximation using two-sided sampling and projection-cost-preserving sketch property.
Recover low-rank approximation using two-sided sampling and projection-cost-preserving sketch property.
• View each entry of $A$ as encoding a large amount of information about its square root $B$. In particular $a_{ij} = b_i^T b_j$. 
• View each entry of $A$ as encoding a large amount of information about its square root $B$. In particular $a_{ij} = b_i^T b_j$.

• Use this view to find a low-rank approximation to $B$ using sublinear accesses to $A$. 

• Obtain near-optimal complexity using ridge leverage scores to sample both rows and columns of $A$. 
• View each entry of $\mathbf{A}$ as encoding a large amount of information about its square root $\mathbf{B}$. In particular $a_{ij} = b_i^T b_j$.

• Use this view to find a low-rank approximation to $\mathbf{B}$ using sublinear accesses to $\mathbf{A}$.

• Since $\mathbf{B}$ has the same singular vectors as $\mathbf{A}$ and $\sigma_i(\mathbf{B}) = \sqrt{\sigma_i(\mathbf{A})}$, a low-rank approximation of $\mathbf{B}$ can used to find one for $\mathbf{A}$, albiet with a $\sqrt{n}$ factor loss in quality.
• View each entry of $\mathbf{A}$ as encoding a large amount of information about its square root $\mathbf{B}$. In particular $a_{ij} = b_i^T b_j$.

• Use this view to find a low-rank approximation to $\mathbf{B}$ using sublinear accesses to $\mathbf{A}$.

• Since $\mathbf{B}$ has the same singular vectors as $\mathbf{A}$ and $\sigma_i(\mathbf{B}) = \sqrt{\sigma_i(\mathbf{A})}$, a low-rank approximation of $\mathbf{B}$ can used to find one for $\mathbf{A}$, albeit with a $\sqrt{n}$ factor loss in quality.

• Obtain near-optimal complexity using ridge leverage scores to sample both rows and columns of $\mathbf{A}$. 
OPEN QUESTIONS

• What else can be done for PSD matrices? We give applications to ridge regression, but what other linear algebraic problems require a second look?

• Are there other natural classes of matrices that admit sublinear time low-rank approximation?

• Starting points are matrices that break the $\Omega(n \sqrt{\text{nnz}(A)})$ time lower bound: e.g. binary matrices, diagonally dominant matrices.

• What can we do when we have PSD matrices with additional structure? E.g. kernel matrices.
• What else can be done for PSD matrices? We give applications to ridge regression, but what other linear algebraic problems require a second look?

• Are there other natural classes of matrices that admit sublinear time low-rank approximation?

• Starting points are matrices that break the $\mathcal{O}(\text{nnz}(A))$ time lower bound: e.g. binary matrices, diagonally dominant matrices.

• What can we do when we have PSD matrices with additional structure? E.g. kernel matrices.
• What else can be done for PSD matrices? We give applications to ridge regression, but what other linear algebraic problems require a second look?
• Are there other natural classes of matrices that admit sublinear time low-rank approximation?
Open Questions

- What else can be done for PSD matrices? We give applications to ridge regression, but what other linear algebraic problems require a second look?
- Are there other natural classes of matrices that admit sublinear time low-rank approximation?
  - Starting points are matrices that break the $\Omega(\text{nnz}(A))$ time lower bound: e.g. binary matrices, diagonally dominant matrices.
OPEN QUESTIONS

• What else can be done for PSD matrices? We give applications to ridge regression, but what other linear algebraic problems require a second look?

• Are there other natural classes of matrices that admit sublinear time low-rank approximation?
  • Starting points are matrices that break the $\Omega(\text{nnz}(A))$ time lower bound: e.g. binary matrices, diagonally dominant matrices.

• What can we do when we have PSD matrices with additional structure? E.g. kernel matrices.
Thanks! Questions?