SUBLINEAR TIME LOW-RANK APPROXIMATION OF POSITIVE SEMIDEFINITE MATRICES

Cameron Musco (MIT) and David P. Woodruff (CMU)

Our Contributions:

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:

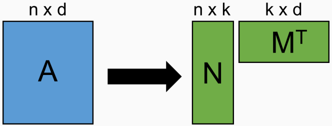
- A near optimal low-rank approximation for any positive semidefinite (PSD) matrix can be computed in sublinear time (i.e. without reading the full matrix).
- **Concrete:** Significantly improves on previous, roughly linear time approaches for general matrices, and bypasses a trivial linear time lower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

Our Contributions:

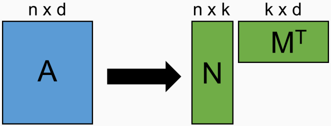
- A near optimal low-rank approximation for any positive semidefinite (PSD) matrix can be computed in sublinear time (i.e. without reading the full matrix).
- **Concrete:** Significantly improves on previous, roughly linear time approaches for general matrices, and bypasses a trivial linear time lower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

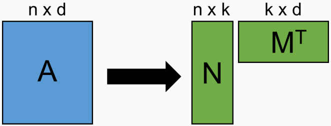


Low-rank approximation is one of the most widely used methods for general matrix and data compression.



- 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.



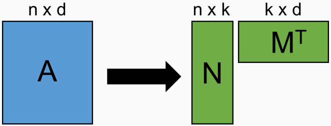
- Closely related to principal component analysis, spectral embedding/clustering, and low-rank matrix completion.

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

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

.

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.

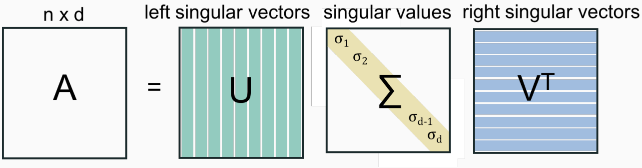
Important Special Case: A is positive semidefinite (PSD). I.e.

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

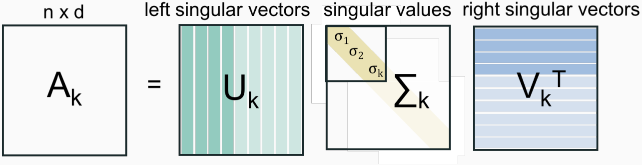
- Includes graph Laplacians, Gram matrices and kernel matrices, covariance matrices, Hessians for convex functions.

An optimal low-rank approximation can be computed via the singular value decomposition (SVD).

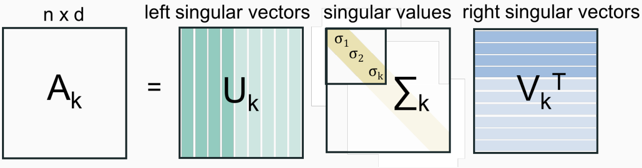
An optimal low-rank approximation can be computed via the singular value decomposition (SVD).



An optimal low-rank approximation can be computed via the singular value decomposition (SVD).

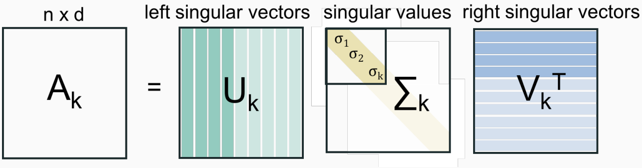


An optimal low-rank approximation can be computed via the singular value decomposition (SVD).



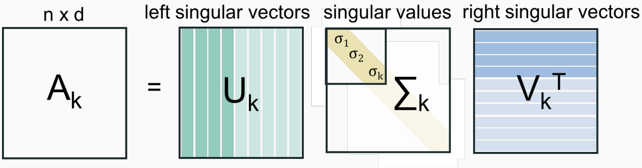
$$\mathbf{A}_k = \operatorname*{arg\,min}_{\mathbf{B}:\text{rank}(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_F$$

An optimal low-rank approximation can be computed via the singular value decomposition (SVD).



$$\mathbf{A}_k = \underset{\mathbf{B}:\text{rank}(\mathbf{B})=k}{\arg\min} \|\mathbf{A}-\mathbf{B}\|_F = \sqrt{\sum_{i,j}(\mathbf{A}_{ij}-\mathbf{B}_{ij})^2}$$

An optimal low-rank approximation can be computed via the singular value decomposition (SVD).



$$\mathbf{A}_k = \arg\min_{\mathbf{B}:\text{rank}(\mathbf{B})=k} \|\mathbf{A}-\mathbf{B}\|_F = \sqrt{\sum_{i,j}(\mathbf{A}_{ij}-\mathbf{B}_{ij})^2}$$

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

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F.$$

Recent work on matrix sketching gives state-of-the-art runtimes

Theorem (Clarkson, Woodruff '13)

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

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F.$$

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

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F.$$

- 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* $\mathbf{A}$, *outputs* $\mathbf{N}, \mathbf{M} \in \mathbb{R}^{n \times k}$ *satisfying with probability* $99/100$*:*

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F.$$

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* $\mathbf{A}$, *outputs* $\mathbf{N}, \mathbf{M} \in \mathbb{R}^{n \times k}$ *satisfying with probability* $99/100$*:*

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F.$$

- Compare to CW'13 which takes $O(\mathrm{nnz}(\mathbf{A})) + n \cdot \mathrm{poly}(k, 1/\epsilon)$.

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* $\mathbf{A}$*, outputs* $\mathbf{N}, \mathbf{M} \in \mathbb{R}^{n \times k}$ *satisfying with probability* $99/100$*:*

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F.$$

- Compare to CW'13 which takes $O(\mathrm{nnz}(\mathbf{A})) + n \cdot \mathrm{poly}(k, 1/\epsilon)$.

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* $\mathbf{A}$, *outputs* $\mathbf{N}, \mathbf{M} \in \mathbb{R}^{n \times k}$ *satisfying with probability* $99/100$*:*

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F.$$

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

LOWER BOUND FOR GENERAL MATRICES

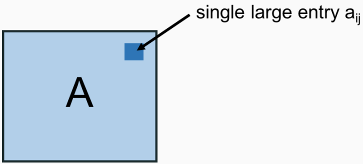
For general matrices, $\Omega(\mathrm{nnz}(\mathbf{A}))$ time is required.

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

- Randomly place a single entry which dominates $\mathbf{A}$'s Frobenius norm.

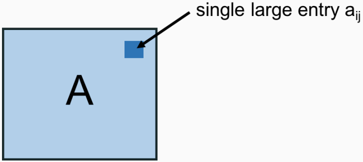
For general matrices, $\Omega(\mathrm{nnz}(\mathbf{A}))$ time is required.

- Randomly place a single entry which dominates $\mathbf{A}$'s Frobenius norm.



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

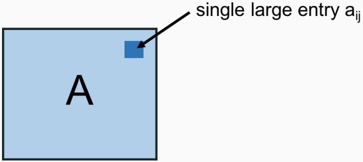
- Randomly place a single entry which dominates $\mathbf{A}$'s Frobenius norm.
- Finding it with constant probability requires reading at least a constant fraction of the non-zero entries in $\mathbf{A}$.



LOWER BOUND FOR GENERAL MATRICES

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

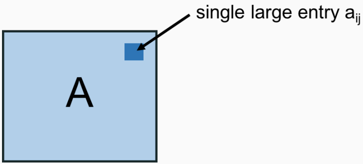
- Randomly place a single entry which dominates $\mathbf{A}$'s Frobenius norm.
- Finding it with constant probability requires reading at least a constant fraction of the non-zero entries in $\mathbf{A}$.



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

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

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



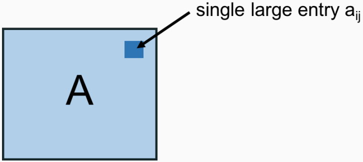
- Lower bound holds for any approximation factor and even rules out $o(\mathrm{nnz}(\mathbf{A}))$ time for weaker guarantees.

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F$$

LOWER BOUND FOR GENERAL MATRICES

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

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



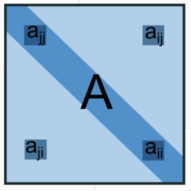
- Lower bound holds for any approximation factor and even rules out $o(\text{nnz}(\mathbf{A}))$ time for weaker guarantees.

$$\|\mathbf{A} - \mathbf{N}\mathbf{M}^T\|_F \leq \|\mathbf{A} - \mathbf{A}_k\|_F + \epsilon\|\mathbf{A}\|_F.$$

Observation: For PSD $\mathbf{A}$, we have for any entry $\mathbf{a}_{ij}$:

$$\mathbf{a}_{ij} \leq \max(\mathbf{a}_{ii}, \mathbf{a}_{jj})$$

since otherwise $(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{A}(\mathbf{e}_i - \mathbf{e}_j) < 0$.



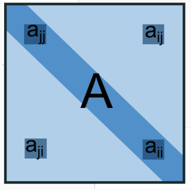
WHAT ABOUT FOR PSD MATRICES?

Observation: For PSD $\mathbf{A}$, we have for any entry $\mathbf{a}_{ij}$:

$$\mathbf{a}_{ij} \leq \max(\mathbf{a}_{ii}, \mathbf{a}_{jj})$$

since otherwise $(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{A} (\mathbf{e}_i - \mathbf{e}_j) < 0$.

- So we can find any 'hidden' heavy entry by looking at its corresponding diagonal entries.



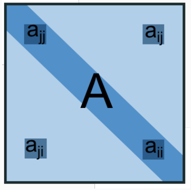
WHAT ABOUT FOR PSD MATRICES?

Observation: For PSD $\mathbf{A}$, we have for any entry $\mathbf{a}_{ij}$:

$$\mathbf{a}_{ij} \leq \max(\mathbf{a}_{ii}, \mathbf{a}_{jj})$$

since otherwise $(\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{A}(\mathbf{e}_i - \mathbf{e}_j) < 0$.

- 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 $\mathbf{A} \in R^{n \times n}$ can be written as $\mathbf{B}^T\mathbf{B}$ for some $\mathbf{B} \in \mathbb{R}^{n \times n}$.

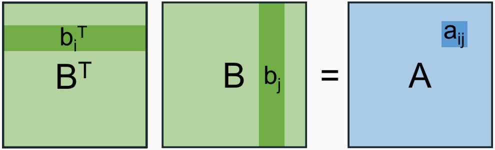
Very Simple Fact: Every PSD matrix $\mathbf{A} \in R^{n\times n}$ can be written as $\mathbf{B}^T\mathbf{B}$ for some $\mathbf{B} \in \mathbb{R}^{n\times n}$.

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

EVERY PSD MATRIX IS A GRAM MATRIX

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

- $\mathbf{B}$ can be any matrix square root of $\mathbf{A}$, e.g. if we let $\mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T$ be the eigendecomposition of $\mathbf{A}$, we can set $\mathbf{B} = \boldsymbol{\Sigma}^{1/2}\mathbf{V}^T$.
- Letting $\mathbf{b}_1, ..., \mathbf{b}_n$ be the columns of $\mathbf{B}$, the entries of $\mathbf{A}$ contain every pairwise dot product $\mathbf{a}_{ij} = \mathbf{b}_i^T\mathbf{b}_j$.



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 $\mathbf{A}$ is a Gram matrix places a variety of geometric constraints on its entries.

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

$$\mathbf{a}_{ij} = \mathbf{b}_i^T \mathbf{b}_j \leq \sqrt{(\mathbf{b}_i^T \mathbf{b}_i) \cdot (\mathbf{b}_j^T \mathbf{b}_j)} = \sqrt{\mathbf{a}_{ii} \cdot \mathbf{a}_{jj}} \leq \max(\mathbf{a}_{ii}, \mathbf{a}_{jj}).$$

EVERY PSD MATRIX IS A GRAM MATRIX

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

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

$$\mathbf{a}_{ij} = \mathbf{b}_i^T \mathbf{b}_j \leq \sqrt{(\mathbf{b}_i^T \mathbf{b}_i) \cdot (\mathbf{b}_j^T \mathbf{b}_j)} = \sqrt{\mathbf{a}_{ii} \cdot \mathbf{a}_{jj}} \leq \max(\mathbf{a}_{ii}, \mathbf{a}_{jj}).$$

Another View: $\mathbf{A}$ contains a lot of information about the column span of $\mathbf{B}$ in a very compressed form – with every pairwise dot product stored as $\mathbf{a}_{ij}$.

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 $\mathbf{B}$ using $o(n^2)$ column dot products? I.e. $o(n^2)$ accesses to $\mathbf{A}$?

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

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 $\mathbf{A}$?

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

- So the top k singular vectors are the same for the two matrices. An optimal low-rank approximation for $\mathbf{B}$ thus gives an optimal low-rank approximation for $\mathbf{A}$.

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 $\mathbf{A}$?

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

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

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

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

Adaptive Sampling

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

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n}\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.

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:*

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

Adaptive Sampling

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

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

Let $\mathbf{P}_\mathcal{S}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_\mathcal{S}\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i - \mathbf{P}_\mathcal{S}\mathbf{b}_i\|^2}$.

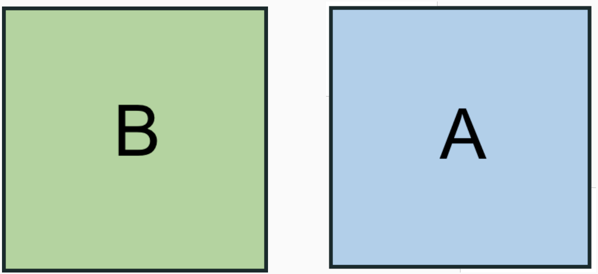
Adaptive Sampling

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

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n}\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.



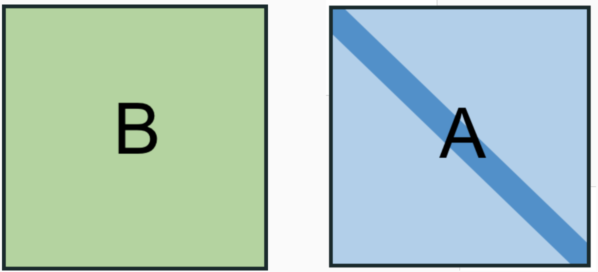
Adaptive Sampling

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

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2} = \frac{\|\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i\|^2} = \frac{\mathbf{a}_{ii}}{\mathrm{tr}(\mathbf{A})}$.



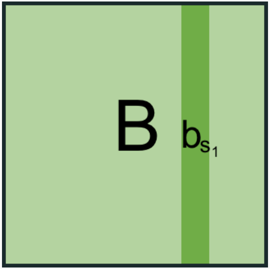
Adaptive Sampling

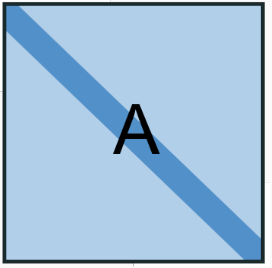
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2} = \frac{\|\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i\|^2} = \frac{\mathbf{a}_{ii}}{\mathrm{tr}(\mathbf{A})}$.





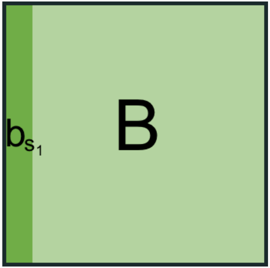
Adaptive Sampling

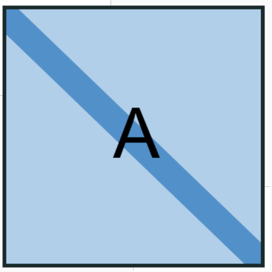
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2} = \frac{\|\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i\|^2} = \frac{\mathbf{a}_{ii}}{\mathrm{tr}(\mathbf{A})}$.





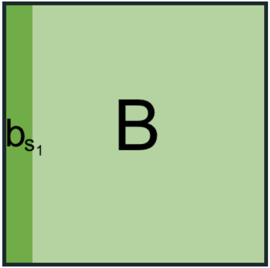
Adaptive Sampling

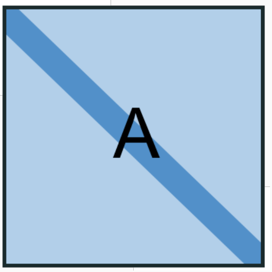
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n} \|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.





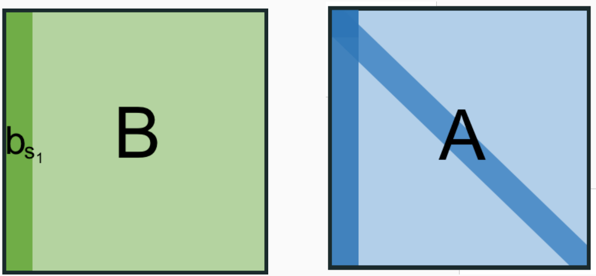
Adaptive Sampling

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

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n}\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.



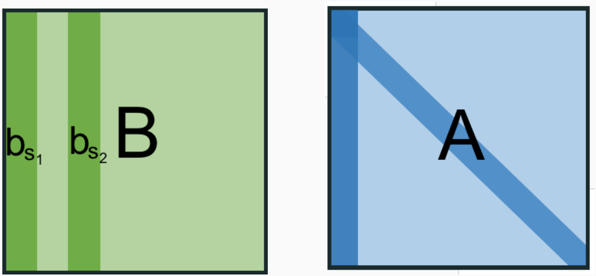
Adaptive Sampling

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

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n} \|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.



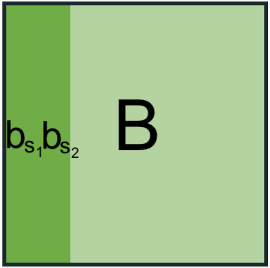
Adaptive Sampling

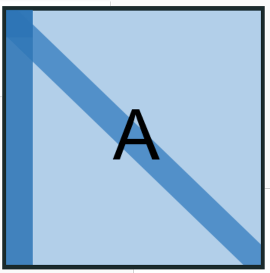
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.





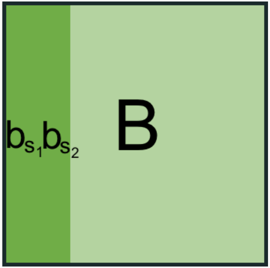
Adaptive Sampling

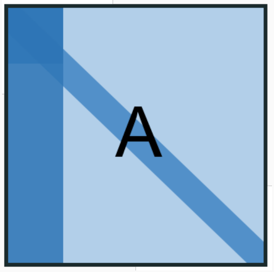
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n} \|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.





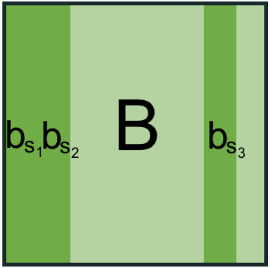
Adaptive Sampling

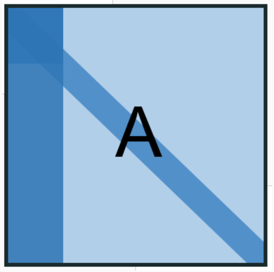
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n}\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.





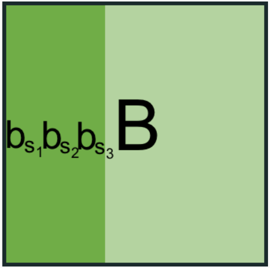
Adaptive Sampling

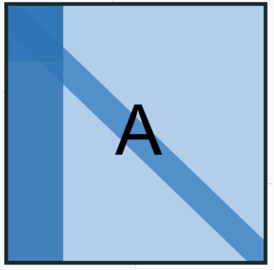
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_{\mathcal{S}}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}{\sum_{i=1}^{n}\|\mathbf{b}_i - \mathbf{P}_{\mathcal{S}}\mathbf{b}_i\|^2}$.





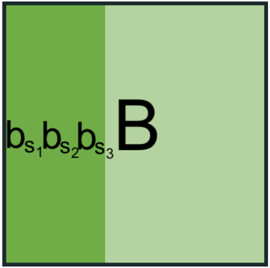
Adaptive Sampling

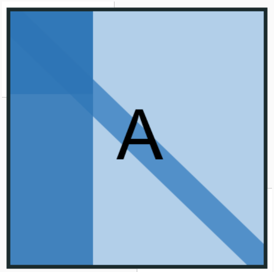
Initially, start with an empty column subset $\mathcal{S} := \{\}$.

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

Let $\mathbf{P}_\mathcal{S}$ be the projection onto the columns in $\mathcal{S}$.

Add $\mathbf{b}_i$ to $\mathcal{S}$ with probability $\frac{\|\mathbf{b}_i - \mathbf{P}_\mathcal{S}\mathbf{b}_i\|^2}{\sum_{i=1}^n \|\mathbf{b}_i - \mathbf{P}_\mathcal{S}\mathbf{b}_i\|^2}$.





Theorem (Factor Matrix Low-Rank Approximation)

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

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

Theorem (Factor Matrix Low-Rank Approximation)

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

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

- How does this translate to low-rank approximation of $\mathbf{A}$ itself?

Lemma

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

$$\|\mathbf{A} - \mathbf{B}^T\mathbf{Z}\mathbf{Z}^T\mathbf{B}\|_F^2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F^2.$$

Boosting to a PSD matrix approximation

Lemma

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

$$\|\mathbf{A} - \mathbf{B}^T\mathbf{Z}\mathbf{Z}^T\mathbf{B}\|_F^2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F^2.$$

BOOSTING TO A PSD MATRIX APPROXIMATION

Lemma

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

$$\|\mathbf{A} - \mathbf{A}\mathbf{S}\mathbf{C}\mathbf{S}^T\mathbf{A}^T\|_F^2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F^2.$$

BOOSTING TO A PSD MATRIX APPROXIMATION

Lemma

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

$$\|\mathbf{A} - \mathbf{A}\mathbf{S}\mathbf{C}\mathbf{S}^T\mathbf{A}^T\|_F^2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F^2.$$

- 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 $\mathbf{A}$.

BOOSTING TO A PSD MATRIX APPROXIMATION

Lemma

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

$$\|\mathbf{A} - \mathbf{ASCS}^T\mathbf{A}^T\|_F^2 \leq (1+\epsilon)\|\mathbf{A} - \mathbf{A}_k\|_F^2.$$

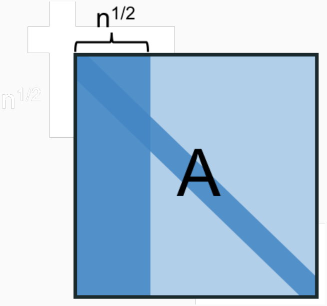
- This gives a low-rank approximation algorithm which accesses just $\tilde{O}\left(\frac{nk^2}{\epsilon^{3/2}/\sqrt{n}}\right) = n^{3/2} \cdot \text{poly}(k, 1/\epsilon)$ entries of $\mathbf{A}$.
- Our best algorithm accesses just $\tilde{O}\left(\frac{nk}{\epsilon^{2.5}}\right)$ entries of $\mathbf{A}$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

LIMITATIONS OF COLUMN SAMPLING

Recall that our algorithm accesses the diagonal of **A** along with $\tilde{O}(k^2/\epsilon)$ columns.

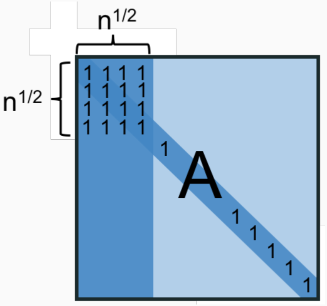
Recall that our algorithm accesses the diagonal of $\mathbf{A}$ along with $\tilde{O}(k^2\sqrt{n})$ columns.

Recall that our algorithm accesses the diagonal of **A** along with $\tilde{O}(k^2\sqrt{n})$ columns.



LIMITATIONS OF COLUMN SAMPLING

Recall that our algorithm accesses the diagonal of $\mathbf{A}$ 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 $\mathbf{A}$'s Frobenius norm.

Solution: Sample both rows and columns of $\mathbf{A}$.

COLUMN AND ROW SAMPLING

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).

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.

Solution: Sample both rows and columns of $\mathbf{A}$.

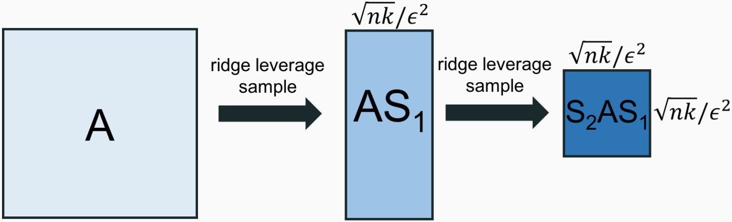
- Instead of adaptive sampling we use ridge leverage scores, which can also be computed using an iterative sampling scheme making $\tilde{O}(nk)$ accesses to $\mathbf{A}$ (Musco, Musco '17).
- Same intuition – select a diverse set of columns which span a near-optimal low-rank approximation of the matrix.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

$$\|\mathbf{AS} - \mathbf{PAS}\|_F^2 = (1 \pm \epsilon)\|\mathbf{A} - \mathbf{PA}\|_F^2.$$

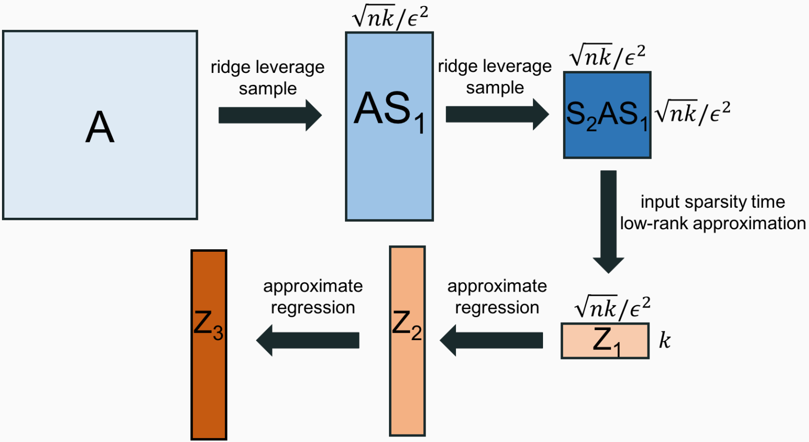
FINAL ALGORITHM

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 $\mathbf{A}$ as encoding a large amount of information about its square root $\mathbf{B}$. In particular $\mathbf{a}_{ij} = \mathbf{b}_i^T \mathbf{b}_j$.

- View each entry of $\mathbf{A}$ as encoding a large amount of information about its square root $\mathbf{B}$. In particular $\mathbf{a}_{ij} = \mathbf{b}_i^T \mathbf{b}_j$.
- Use this view to find a low-rank approximation to $\mathbf{B}$ using sublinear accesses to $\mathbf{A}$.

- View each entry of $\mathbf{A}$ as encoding a large amount of information about its square root $\mathbf{B}$. In particular $\mathbf{a}_{ij} = \mathbf{b}_i^T \mathbf{b}_j$.
- 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.

SUMMARY OF MAIN IDEAS

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

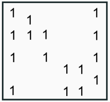
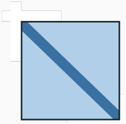
OPEN QUESTIONS

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

- 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?

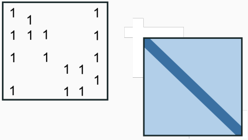
OPEN QUESTIONS

- 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 $\Omega(\mathrm{nnz}(\mathbf{A}))$ 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 $\Omega(\text{nnz}(\mathbf{A}))$ 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?