INPUT SPARSITY TIME LOW-RANK APPROXIMATION VIA RIDGE LEVERAGE SCORE SAMPLING

Michael B. Cohen, Cameron Musco and Christopher Musco

Massachusetts Institute of Technology, EECS.
SODA 2017.

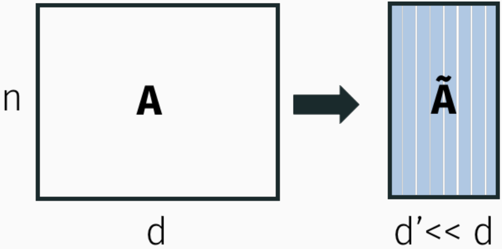
- Randomized methods such as importance sampling, linear sketching, and stochastic gradient descent have led to recent breakthroughs on very well studied problems. E.g. least squares regression, low-rank approximation

- Randomized methods such as importance sampling, linear sketching, and stochastic gradient descent have led to recent breakthroughs on very well studied problems. E.g. least squares regression, low-rank approximation
- Supported by new results in random matrix theory and understanding of how to use these results algorithmically.

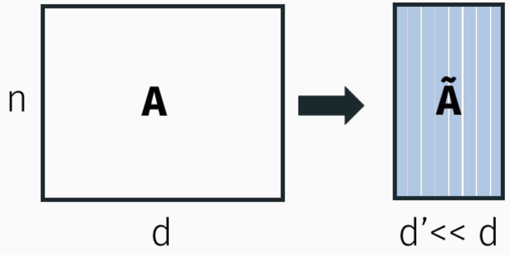
- Randomized methods such as importance sampling, linear sketching, and stochastic gradient descent have led to recent breakthroughs on very well studied problems. E.g. least squares regression, low-rank approximation
- Supported by new results in random matrix theory and understanding of how to use these results algorithmically.
- Closely tied to work on graph sparsification, fast laplacian solvers, streaming algorithms, compressed sensing, etc.

[Clarkson Woodruff STOC ’13]: Sparse Random Projections

[Clarkson Woodruff STOC '13]: Sparse Random Projections

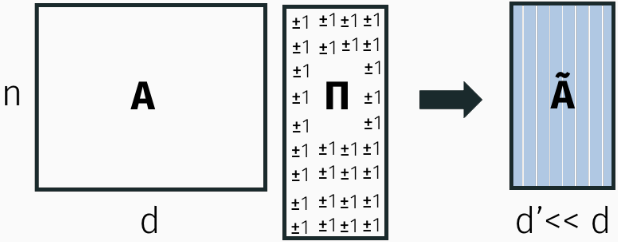


[Clarkson Woodruff STOC '13]: Sparse Random Projections



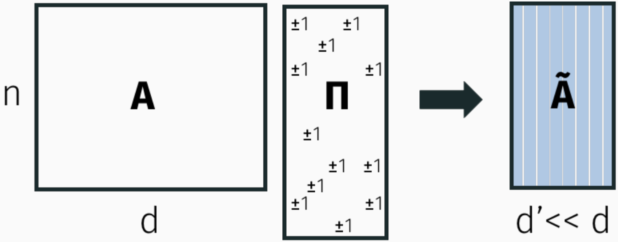
- Solution for Ã ⇒ approximate solution for A for problems like linear system solving, low-rank approximation, etc.

[Clarkson Woodruff STOC '13]: Sparse Random Projections



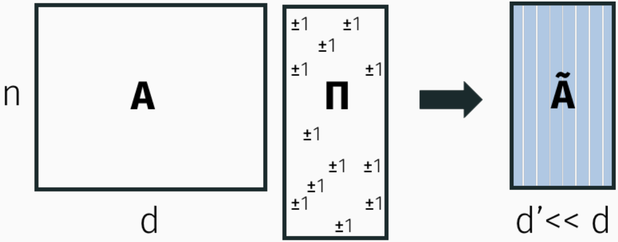
- Solution for $\tilde{A} \Rightarrow$ approximate solution for A for problems like linear system solving, low-rank approximation, etc.

[Clarkson Woodruff STOC '13]: Sparse Random Projections

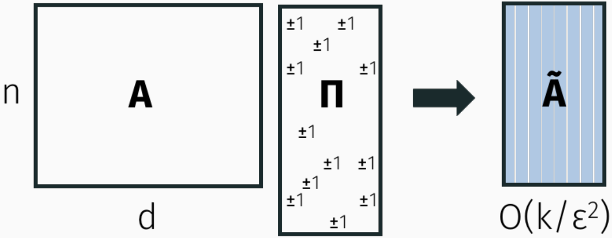


- Solution for $\tilde{\mathbf{A}} \Rightarrow$ approximate solution for $\mathbf{A}$ for problems like linear system solving, low-rank approximation, etc.

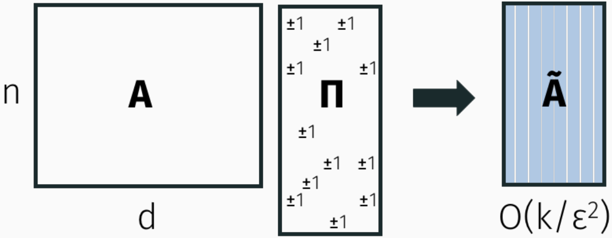
[Clarkson Woodruff STOC '13]: Sparse Random Projections



- Solution for $\tilde{\mathbf{A}} \Rightarrow$ approximate solution for $\mathbf{A}$ for problems like linear system solving, low-rank approximation, etc.
- $O(\text{nnz}(\mathbf{A}))$ to compute $\mathbf{A\Pi}$ plus lower order terms = input sparsity time



Set $\mathbf{Q} \leftarrow$ top k left singular vectors of $\tilde{\mathbf{A}}$.



Set $\mathbf{Q} \leftarrow$ top k left singular vectors of $\tilde{\mathbf{A}}$.

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^T\mathbf{A}\|_F^2 \leq (1+\epsilon) \min_{\mathbf{B} \mid \text{rank}(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_F^2$$

Runtime:

Runtime:

- $\mathbf{\Pi}$ only has $O(1)$ non-zeros per column.

Runtime:

- $\mathbf{\Pi}$ only has $O(1)$ non-zeros per column.
- $O(\text{nnz}(\mathbf{A}))$ time to compute $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{\Pi}$.

Runtime:

- $\mathbf{\Pi}$ only has $O(1)$ non-zeros per column.
- $O(\text{nnz}(\mathbf{A}))$ time to compute $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{\Pi}$.
- $O(nk^2/\epsilon^4)$ time to compute $\tilde{\mathbf{A}}$'s top singular vectors

Runtime:

- $\mathbf{\Pi}$ only has $O(1)$ non-zeros per column.
- $O(\text{nnz}(\mathbf{A}))$ time to compute $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{\Pi}$.
- $O(nk^2/\epsilon^4)$ time to compute $\tilde{\mathbf{A}}$'s top singular vectors

$$\text{Total: } O(\text{nnz}(\mathbf{A})) + \underbrace{n \cdot \text{poly}(k, 1/\epsilon)}_{\text{lower order}}$$

Runtime:

- $\mathbf{\Pi}$ only has $O(1)$ non-zeros per column.
- $O(\text{nnz}(\mathbf{A}))$ time to compute $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{\Pi}$.
- $O(nk^2/\epsilon^4)$ time to compute $\tilde{\mathbf{A}}$'s top singular vectors

$$\text{Total: } O(\text{nnz}(\mathbf{A})) + \underbrace{n \cdot \text{poly}(k, 1/\epsilon)}_{\text{lower order}}$$

- Many improvements. See [Avron Clarkson Woodruff '16] for best low order terms.

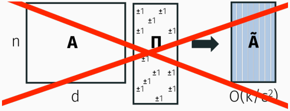
Runtime:

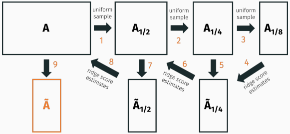
- $\mathbf{\Pi}$ only has $O(1)$ non-zeros per column.
- $O(\text{nnz}(\mathbf{A}))$ time to compute $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{\Pi}$.
- $O(nk^2/\epsilon^4)$ time to compute $\tilde{\mathbf{A}}$'s top singular vectors

$$\text{Total: } O(\text{nnz}(\mathbf{A})) + \underbrace{n \cdot \text{poly}(k, 1/\epsilon)}_{\text{lower order}}$$

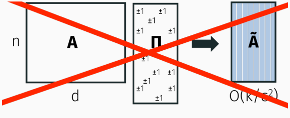
- Many improvements. See [Avron Clarkson Woodruff '16] for best low order terms.
- Compare with $\tilde{O}(\text{nnz}(\mathbf{A}) \cdot k/\sqrt{\epsilon})$ for iterative methods.

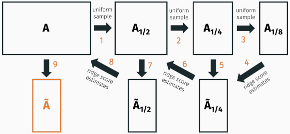
Main Result: Input sparsity time low-rank approximation without sparse random projections.





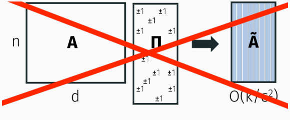
Main Result: Input sparsity time low-rank approximation without sparse random projections.

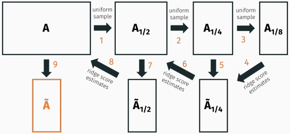




- Column subset selection in single-pass streams.

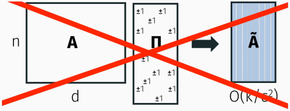
Main Result: Input sparsity time low-rank approximation without sparse random projections.

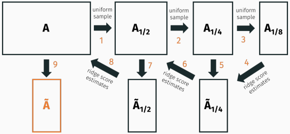




- Column subset selection in single-pass streams.
- Linear time algorithms for Nyström kernel approximation [Musco Musco '16].

Main Result: Input sparsity time low-rank approximation without sparse random projections.

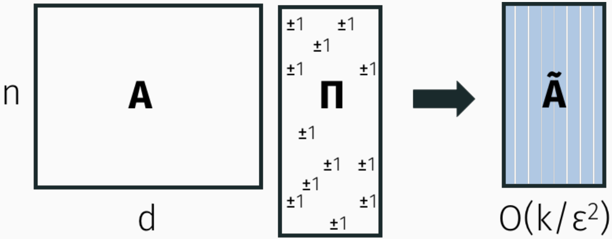




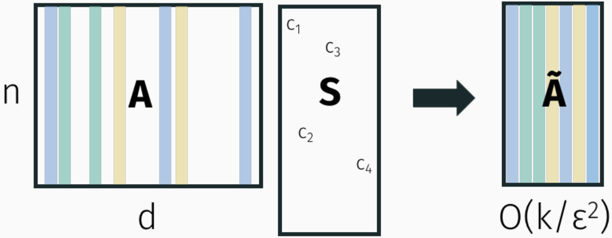
- Column subset selection in single-pass streams.
- Linear time algorithms for Nyström kernel approximation [Musco Musco '16].
- Sublinear time, relative error algorithms for low-rank approximation of PSD matrices [Musco Woodruff '16]

DIMENSIONALITY REDUCTION VIA IMPORTANCE SAMPLING

DIMENSIONALITY REDUCTION VIA IMPORTANCE SAMPLING

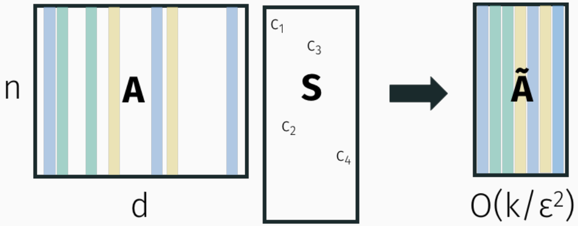


DIMENSIONALITY REDUCTION VIA IMPORTANCE SAMPLING

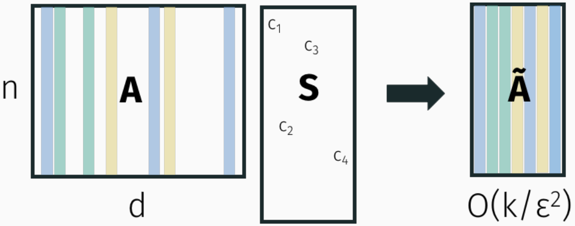


Extremely simple and efficient... once **S** is known.

Variations on statistical leverage scores give a sketch $\tilde{\mathbf{A}}$ that is sufficient for near-optimal low-rank approximation.

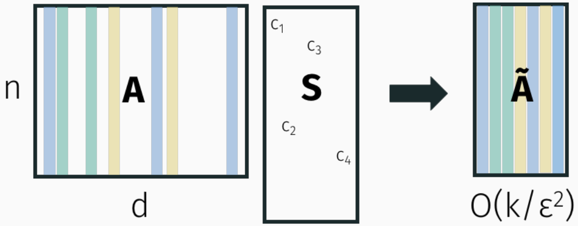


Variations on statistical leverage scores give a sketch $\tilde{\mathbf{A}}$ that is sufficient for near-optimal low-rank approximation.

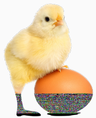


But computing these scores seems as hard as low-rank approximation itself.

Variations on statistical leverage scores give a sketch $\tilde{A}$ that is sufficient for near-optimal low-rank approximation.



But computing these scores seems as hard as low-rank approximation itself.



1. Brief discussion of techniques
2. Why care about sampling?

Leverage scores are the natural sampling probabilities for relative error matrix approximation.

Leverage scores are the natural sampling probabilities for relative error matrix approximation.

$$\tau(\mathbf{a}_i) = \mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{a}_i.$$

Leverage scores are the natural sampling probabilities for relative error matrix approximation.

$$\tau(\mathbf{a}_i) = \mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{a}_i.$$

Intuition: Measure uniqueness of column. $\tau(\mathbf{a}_i) = \min \|\mathbf{y}\|_2^2$ such that $\mathbf{A}\mathbf{y} = \mathbf{a}_i$.

Leverage scores are the natural sampling probabilities for relative error matrix approximation.

$$\tau(\mathbf{a}_i) = \mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{a}_i.$$

Intuition: Measure uniqueness of column. $\tau(\mathbf{a}_i) = \min \|\mathbf{y}\|_2^2$ such that $\mathbf{A}\mathbf{y} = \mathbf{a}_i$.

Sampling $\tilde{O}(\text{rank}(\mathbf{A})/\epsilon^2)$ columns by leverage scores gives spectral approximation:

$$(1-\epsilon)\mathbf{A}\mathbf{A}^T \preceq \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T \preceq (1+\epsilon)\mathbf{A}\mathbf{A}^T.$$

Naively, applying $(\mathbf{A}\mathbf{A}^T)^{-1}$ to compute $\mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{a}_i$ is expensive.

Naively, applying $(\mathbf{A}\mathbf{A}^T)^{-1}$ to compute $\mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{a}_i$ is expensive.

- But leverage scores are robust. E.g. uniformly sampling 1/2 the columns of $\mathbf{A}$ will not change leverage scores too much on average.

Naively, applying $(\mathbf{A}\mathbf{A}^T)^{-1}$ to compute $\mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{a}_i$ is expensive.

- But leverage scores are robust. E.g. uniformly sampling 1/2 the columns of $\mathbf{A}$ will not change leverage scores too much on average.
- Leads to $O(\text{nnz}(\mathbf{A}))$ time recursive sampling algorithm for leverage score approximation [Cohen, Lee, Musco, Musco, Peng, Sidford '15].

Naively, applying $(\mathbf{A}\mathbf{A}^T)^{-1}$ to compute $\mathbf{a}_i^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{a}_i$ is expensive.

- But leverage scores are robust. E.g. uniformly sampling 1/2 the columns of $\mathbf{A}$ will not change leverage scores too much on average.
- Leads to $O(\mathrm{nnz}(\mathbf{A}))$ time recursive sampling algorithm for leverage score approximation [Cohen, Lee, Musco, Musco, Peng, Sidford '15].
- Input sparsity time regression without sparse projections.

"Subspace Scores" [Drineas, Mahoney, Muthukrishnan '08], [Sarló '06]:

$$\tau(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}\mathbf{A}^T)^+\mathbf{a}_i$$

"Subspace Scores" [Drineas, Mahoney, Muthukrishnan '08], [Sarló '06]:

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}_k\mathbf{A}_k^T)^+\mathbf{a}_i$$

where $\mathbf{A}_k = \arg\min_{\mathbf{B}|\operatorname{rank}(\mathbf{B})=k} \|\mathbf{A}-\mathbf{B}\|_F^2$.

"Subspace Scores" [Drineas, Mahoney, Muthukrishnan '08], [Sarló '06]:

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}_k\mathbf{A}_k^T)^+\mathbf{a}_i$$

where $\mathbf{A}_k = \arg\min_{\mathbf{B}|\operatorname{rank}(\mathbf{B})=k} \|\mathbf{A} - \mathbf{B}\|_F^2$.

- Gives additional error depending on $\|\mathbf{A} - \mathbf{A}_k\|_F^2 \implies$ good enough for near optimal low-rank approximation.

Computing Subspace Scores:

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}_k\mathbf{A}_k^T)^+\mathbf{a}_i$$

Computing Subspace Scores:

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}_k\mathbf{A}_k^T)^+\mathbf{a}_i$$

- Suffices to replace $\mathbf{A}_k$ with any near-optimal low-rank approximation $\tilde{\mathbf{A}}_k$.

Computing Subspace Scores:

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}_k\mathbf{A}_k^T)^+\mathbf{a}_i$$

- Suffices to replace $\mathbf{A}_k$ with any near-optimal low-rank approximation $\tilde{\mathbf{A}}_k$.
- But this is what we want to compute in the first place! Hence all nnz($\mathbf{A}$) time sampling algorithms rely critically on sparse random projections.

Computing Subspace Scores:

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}_k\mathbf{A}_k^T)^+\mathbf{a}_i$$

- Suffices to replace $\mathbf{A}_k$ with any near-optimal low-rank approximation $\tilde{\mathbf{A}}_k$.
- But this is what we want to compute in the first place! Hence all nnz($\mathbf{A}$) time sampling algorithms rely critically on sparse random projections.
- Further, subspace scores are unstable. $\mathbf{A}_k$ (an even an approximation to it) can change completely due to small perturbations in $\mathbf{A}$. Hard to make recursive sampling approaches work.

Key Idea:

Key Idea: Truncation $\Rightarrow$ Regularization

Key Idea: Truncation $\Rightarrow$ Regularization

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}_k\mathbf{A}_k^T)^+\mathbf{a}_i$$

Key Idea: Truncation $\Rightarrow$ Regularization

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}\mathbf{A}^T + \lambda \mathbf{I})^+ \mathbf{a}_i$$

where $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$.

Key Idea: Truncation $\Rightarrow$ Regularization

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}\mathbf{A}^T + \lambda \mathbf{I})^+ \mathbf{a}_i$$

where $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$. [Alaoui Mahoney '16]

Key Idea: Truncation $\Rightarrow$ Regularization

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}\mathbf{A}^T + \lambda \mathbf{I})^+ \mathbf{a}_i$$

where $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$. [Alaoui Mahoney '16]

- Ridge 'washes out' rather than completely removes contributions from small singular directions.

Key Idea: Truncation $\Rightarrow$ Regularization

$$\tau_k(\mathbf{a}_i) = \mathbf{a}_i(\mathbf{A}\mathbf{A}^T + \lambda \mathbf{I})^+ \mathbf{a}_i$$

where $\lambda = \frac{\|\mathbf{A}-\mathbf{A}_k\|_F^2}{k}$. [Alaoui Mahoney '16]

- Ridge 'washes out' rather than completely removes contributions from small singular directions.
- These are just the standard leverage scores of $[\mathbf{A}, \sqrt{\lambda}\mathbf{I}]$! Computable using the recursive sampling algorithms of [CLMMPS '15].

Standard arguements show that sampling $\tilde{O}(k/\epsilon^2)$ columns by their ridge leverage scores gives an approximation:

$$(1-\epsilon)\mathbf{A}\mathbf{A}^T - \epsilon\lambda\mathbf{I} \preceq \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T \preceq (1+\epsilon)\mathbf{A}\mathbf{A}^T + \epsilon\lambda\mathbf{I}.$$

Standard arguements show that sampling $\tilde{O}(k/\epsilon^2)$ columns by their ridge leverage scores gives an approximation:

$$(1-\epsilon)\mathbf{A}\mathbf{A}^T - \epsilon\lambda\mathbf{I} \preceq \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T \preceq (1+\epsilon)\mathbf{A}\mathbf{A}^T + \epsilon\lambda\mathbf{I}.$$

- We show that this is enough for $\tilde{\mathbf{A}}$'s top singular vector space to approximate that of $\mathbf{A}$.

Standard arguements show that sampling $\tilde{O}(k/\epsilon^2)$ columns by their ridge leverage scores gives an approximation:

$$(1-\epsilon)\mathbf{A}\mathbf{A}^T - \epsilon\lambda\mathbf{I} \preceq \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T \preceq (1+\epsilon)\mathbf{A}\mathbf{A}^T + \epsilon\lambda\mathbf{I}.$$

- We show that this is enough for $\tilde{\mathbf{A}}$'s top singular vector space to approximate that of $\mathbf{A}$.
- Specifically, show $\tilde{\mathbf{A}}$ is a good projection-cost-preserving sketch of $\mathbf{A}$ [Cohen Elder Musco Musco Persu '15].

Standard arguements show that sampling $\tilde{O}(k/\epsilon^2)$ columns by their ridge leverage scores gives an approximation:

$$(1-\epsilon)\mathbf{A}\mathbf{A}^T - \epsilon\lambda\mathbf{I} \preceq \tilde{\mathbf{A}}\tilde{\mathbf{A}}^T \preceq (1+\epsilon)\mathbf{A}\mathbf{A}^T + \epsilon\lambda\mathbf{I}.$$

- We show that this is enough for $\tilde{\mathbf{A}}$'s top singular vector space to approximate that of $\mathbf{A}$.
- Specifically, show $\tilde{\mathbf{A}}$ is a good projection-cost-preserving sketch of $\mathbf{A}$ [Cohen Elder Musco Musco Persu '15].
- Also achieve near optimal column subset selection via a connection between ridge scores and adaptive sampling [Deshpande Rademacher Vempala Wang '06].

Low-Rank Approximation via Ridge Leverage Scores: Sampling $\mathbf{A}$ using the leverage scores of $(\mathbf{A} + \lambda \mathbf{I})$ give near optimal sized sketches for low-rank approximation.

Low-Rank Approximation via Ridge Leverage Scores: Sampling $\mathbf{A}$ using the leverage scores of $(\mathbf{A} + \lambda \mathbf{I})$ give near optimal sized sketches for low-rank approximation.

- Scores can be computed in input sparsity time via iterative approximation algorithms.

Low-Rank Approximation via Ridge Leverage Scores: Sampling $\mathbf{A}$ using the leverage scores of $(\mathbf{A}+\lambda\mathbf{I})$ give near optimal sized sketches for low-rank approximation.

- Scores can be computed in input sparsity time via iterative approximation algorithms.

Corollary: $O(\text{nnz}(\mathbf{A})) + \text{poly}(k, \epsilon)$ time to compute $\widetilde{\mathbf{B}}$ with:

$$\|\mathbf{A}-\widetilde{\mathbf{B}}\|_F^2 \leq (1+\epsilon) \min_{\mathbf{B} \mid \text{rank}(\mathbf{B})=k} \|\mathbf{A}-\mathbf{B}\|_F^2$$

WHY SAMPLING?

Why do we care about avoiding sparse random projections in the first place?

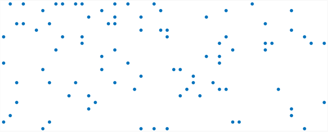
Original Motivation: Match $O(\text{nnz}(\mathbf{A}))$ time random projection algorithms for matrix preconditioning and over-constrained linear regression.

- Li Miller Peng '13
- Cohen Lee Musco Musco Peng Sidford '15.

Reason #1: Sampling Preserves Structure and Sparsity.

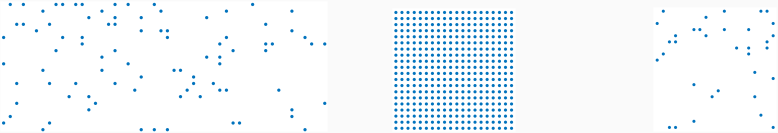
Reason #1: Sampling Preserves Structure and Sparsity.

Original Data General Sketch Column Sample



Reason #1: Sampling Preserves Structure and Sparsity.

Original Data General Sketch Column Sample



Even when $\mathbf{A}$ is sparse, $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{\Pi}$ will be dense. Limits compression for very sparse matrices.

Reason #1: Sampling Preserves Structure and Sparsity

Results for regression used in new work on sparsifying and solving Laplacian and SDD systems:

- Lee, Peng, Spielman '15.
- Kyng, Lee, Peng, Sachdeva, Spielman '16
- Jindal, Kolev '16

Reason #2: Sampling works in settings where random projection does not apply.

Reason #2: Sampling works in settings where random projection does not apply.

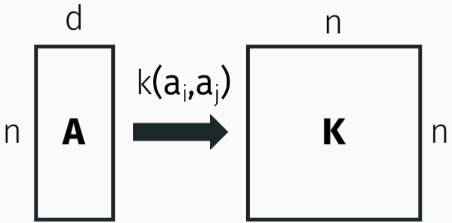
In this paper: Applications to single-pass streaming algorithms for the column subset selection problem.

Reason #2: Sampling works in settings where random projection does not apply.

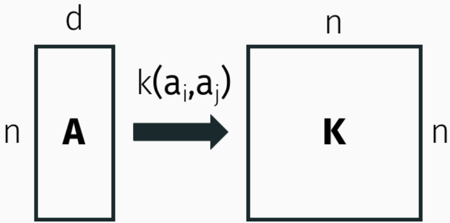
In this paper: Applications to single-pass streaming algorithms for the column subset selection problem.

In follow up work:

- [Musco Musco '16]: Linear time kernel matrix approximation.
- [Musco Woodruff '16]: Sublinear time relative-error low-rank approximation of PSD matrices.

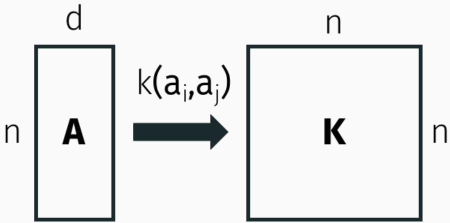


$$K_{i,j} = k(a_i, a_j),\ K = \phi(A)^T\phi(A)$$



$$K_{i,j} = k(a_i, a_j),\ K = \phi(A)^T\phi(A)$$

- Working with full $n \times n$ kernel matrix often prohibitive. Low-rank approximation is important for efficient kernel ridge regression, kernel PCA, kernel k-means clustering, etc.

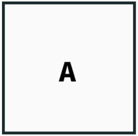


$$K_{i,j} = k(a_i, a_j),\ K = \phi(A)^T\phi(A)$$

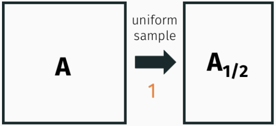
- Working with full $n \times n$ kernel matrix often prohibitive. Low-rank approximation is important for efficient kernel ridge regression, kernel PCA, kernel k-means clustering, etc.
- Sketching K directly requires $\Omega(n^2)$ kernel evaluations.

How can we avoid this using sampling?

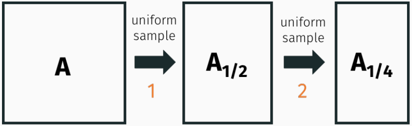
How can we avoid this using sampling?



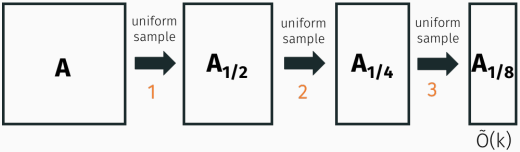
How can we avoid this using sampling?



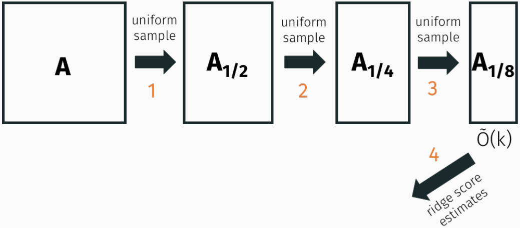
How can we avoid this using sampling?



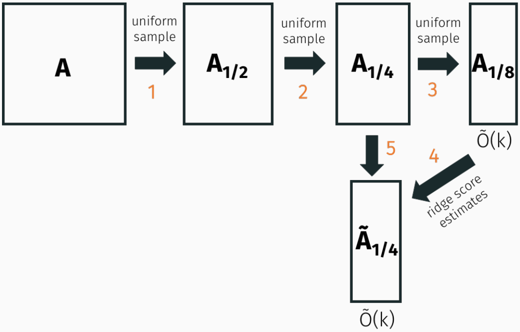
How can we avoid this using sampling?



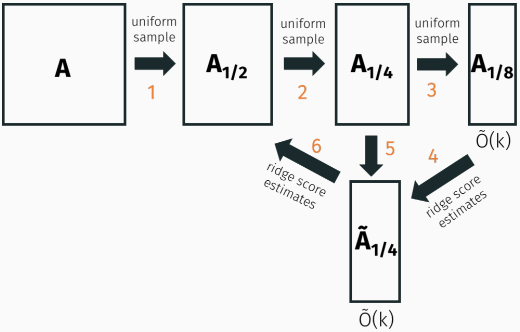
How can we avoid this using sampling?



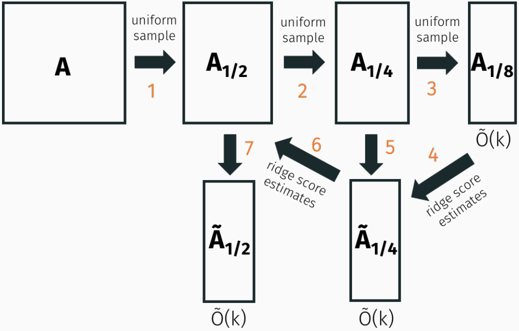
How can we avoid this using sampling?



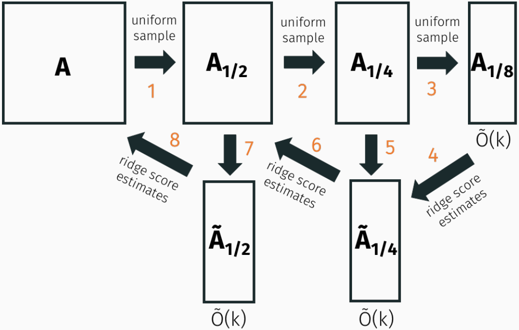
How can we avoid this using sampling?



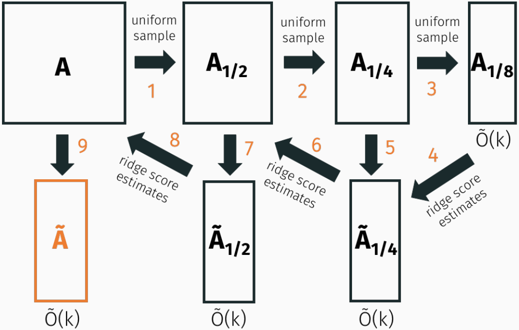
How can we avoid this using sampling?



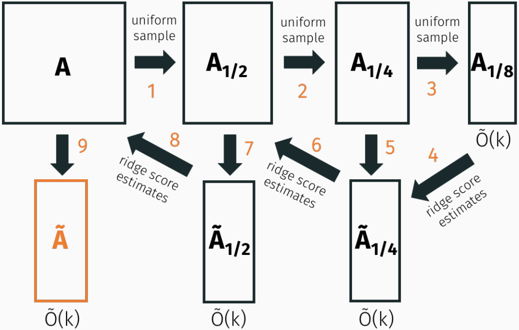
How can we avoid this using sampling?



How can we avoid this using sampling?

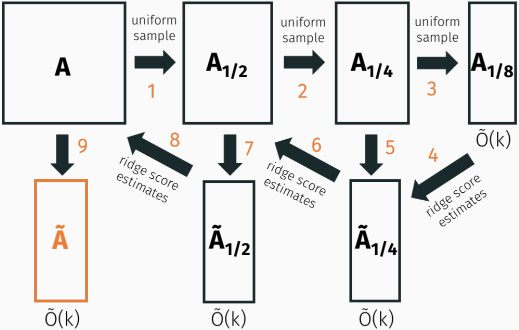


How can we avoid this using sampling?



- $O(nk)$ dot products per level $\Rightarrow \tilde{O}(nk)$ kernel evaluations if we set $\mathbf{A} = \mathbf{K}^{1/2}$ so $\mathbf{A}\mathbf{A}^T = \mathbf{K}$.

How can we avoid this using sampling?



- $O(nk)$ dot products per level $\Rightarrow \tilde{O}(nk)$ kernel evaluations if we set $\mathbf{A} = \mathbf{K}^{1/2}$ so $\mathbf{A}\mathbf{A}^T = \mathbf{K}$.
- Lets us find a low-rank approximation for $\mathbf{K}^{1/2}$ without constructing all of $\mathbf{K}$.

Summary: Input sparsity time linear algebra is not just about sparse random embeddings. Results can also achieved via leverage score sampling.

Summary: Input sparsity time linear algebra is not just about sparse random embeddings. Results can also achieved via leverage score sampling.

Open Questions:

- Empirical evaluation, especially for kernel applications.
- Other methods of achieving input sparsity time? Deterministic?
- Further applications?