Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning

Nai-Hui Chia[1], András Gilyén[2], Tongyang Li[3], Han-Hsuan Lin[1], Ewin Tang[4], and Chunhao Wang[1]

[1]University of Texas at Austin
[2]California Institute of Technology
[3]University of Maryland
[4]University of Washington

arXiv:1910.06151

QIP 2020 – Jan 6, 2020

Background of QML

- Representatives of *polylog-time* QML algorithms
  - [HHL09] sparse matrix inversion
  - [WBL12] data fitting
  - [GZL12] PageRank-style problems
  - [LMR13] supervised clustering
  - [LMR14] principal component analysis
  - [RML14] support-vector machines
  - [ZFF15] Gaussian process regression
  - [CD16] discriminant analysis
  - [KP17] recommendation system

Background of QML

- Representatives of *polylog-time* QML algorithms
  - [HHL09] sparse matrix inversion
  - [WBL12] data fitting
  - [GZL12] PageRank-style problems
  - [LMR13] supervised clustering
  - [LMR14] principal component analysis
  - [RML14] support-vector machines
  - [ZFF15] Gaussian process regression
  - [CD16] discriminant analysis
  - [KP17] recommendation system
- Core: Singular value transformation [Gil+19]
  $f^{(SV)}(A) = \sum_{i=1}^{r} f(\sigma_i) v_i u_i^\dagger$

Background of QML

- Representatives of *polylog-time* QML algorithms
  - [HHL09] sparse matrix inversion
  - [WBL12] data fitting
  - [GZL12] PageRank-style problems
  - [LMR13] supervised clustering
  - [LMR14] principal component analysis
  - [RML14] support-vector machines
  - [ZFF15] Gaussian process regression
  - [CD16] discriminant analysis
  - [KP17] recommendation system
- Core: Singular value transformation [Gil+19]
  $f^{(SV)}(A) = \sum_{i=1}^{r} f(\sigma_i) v_i u_i^\dagger$
- Techniques:
  - Sparse: HHL [HHL09] — $A^{-1} \to f^{(SV)}(A)$
  - **Low-rank (meaningful applications)**: Quantum PCA [LMR14] — $e^{-iA} \to f^{(SV)}(A)$

I/O models of QML algorithms

$$|\text{input vector}\rangle \longrightarrow |\text{output vector}\rangle$$

$$|v\rangle := \sum_{i=1}^{n} \frac{v(i)}{\|v\|} |i\rangle$$

I/O models of QML algorithms

$$|\text{input vector}\rangle \longrightarrow |\text{output vector}\rangle$$

$$|v\rangle := \sum_{i=1}^{n} \frac{v(i)}{\|v\|} |i\rangle$$

The HHL algorithm: $A \in \mathbb{C}^{n \times n}$ sparse, well-conditioned

$$|b\rangle \longrightarrow \left|A^{-1}b\right\rangle$$

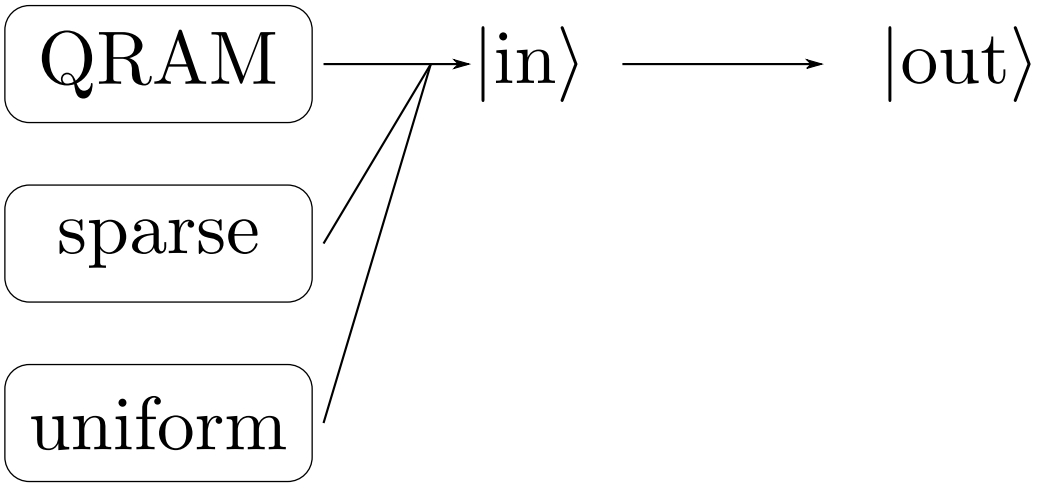
I/O models of QML algorithms (cont.)

But how do we prepare the input and make use of the output?

$$|\text{in}\rangle \longrightarrow |\text{out}\rangle$$

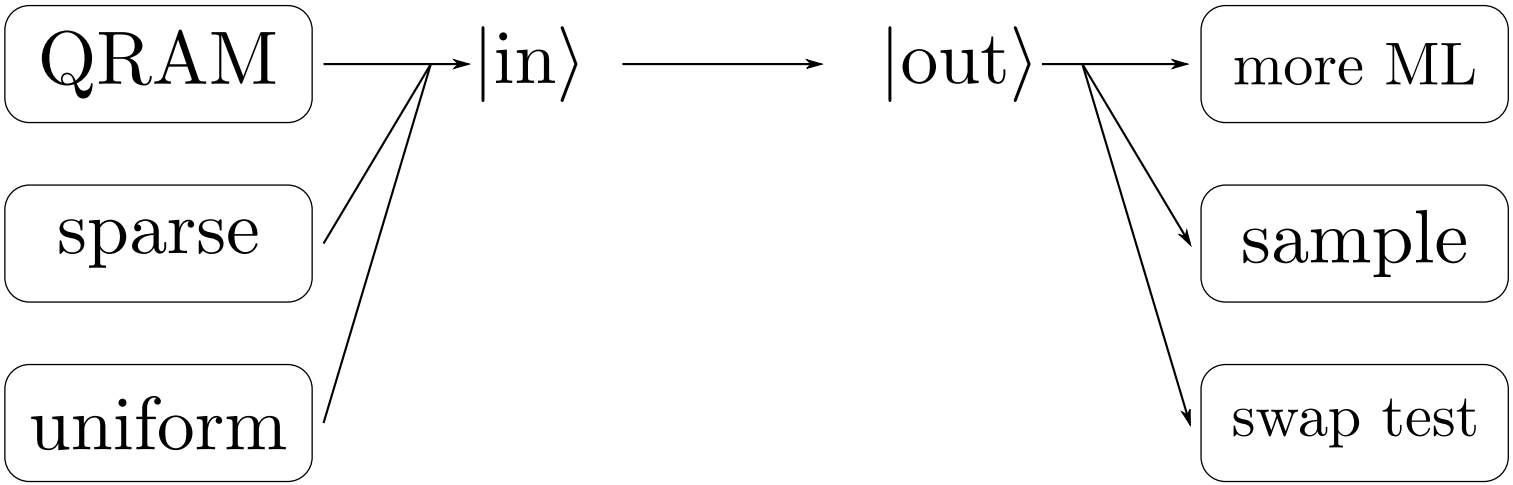
I/O models of QML algorithms (cont.)

But how do we prepare the input and make use of the output?



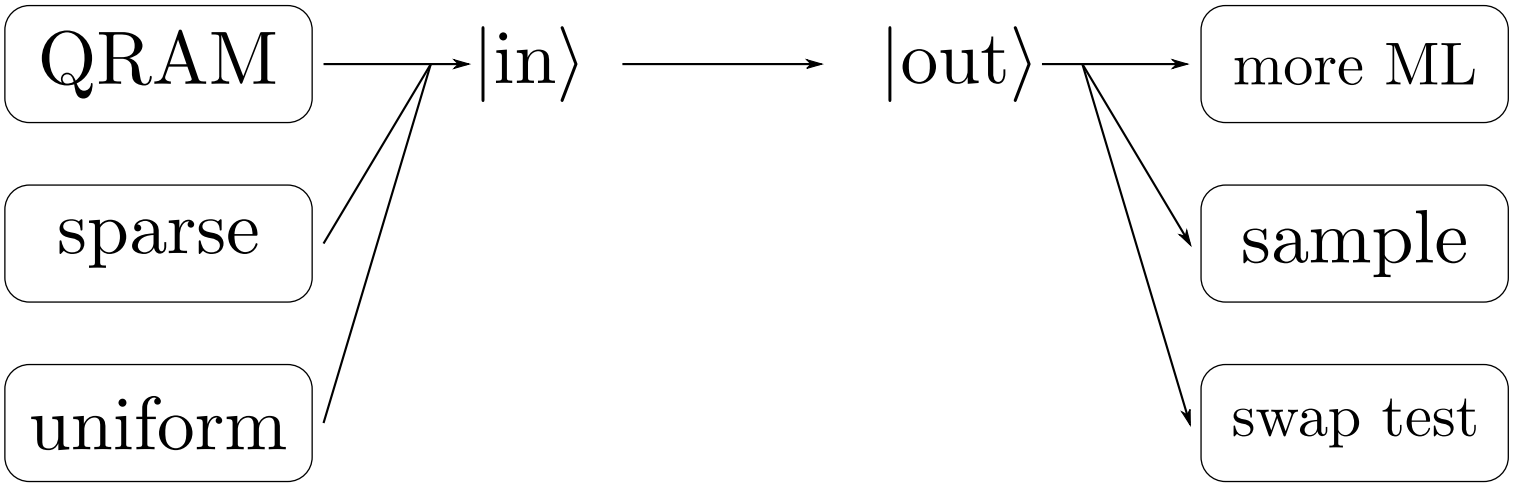
I/O models of QML algorithms (cont.)

But how do we prepare the input and make use of the output?



I/O models of QML algorithms (cont.)

But how do we prepare the input and make use of the output?



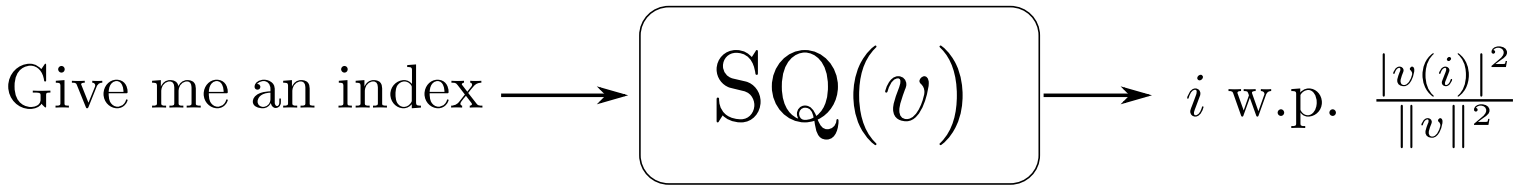
Do we have the classical analogue?

Sampling-query access of a vector

Classical analogue: measuring a state $\rightarrow$ sampling a vector
The **sampling-query** access to v, $\mathrm{SQ}(v)$:

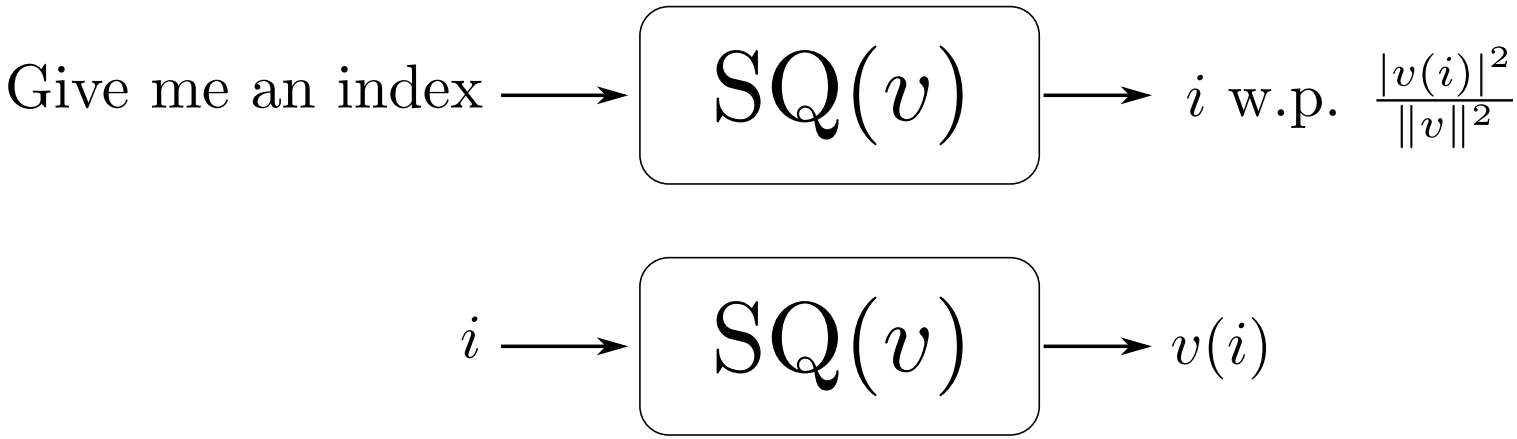
Sampling-query access of a vector

Classical analogue: measuring a state $\rightarrow$ sampling a vector
The **sampling-query** access to v, $\mathrm{SQ}(v)$:



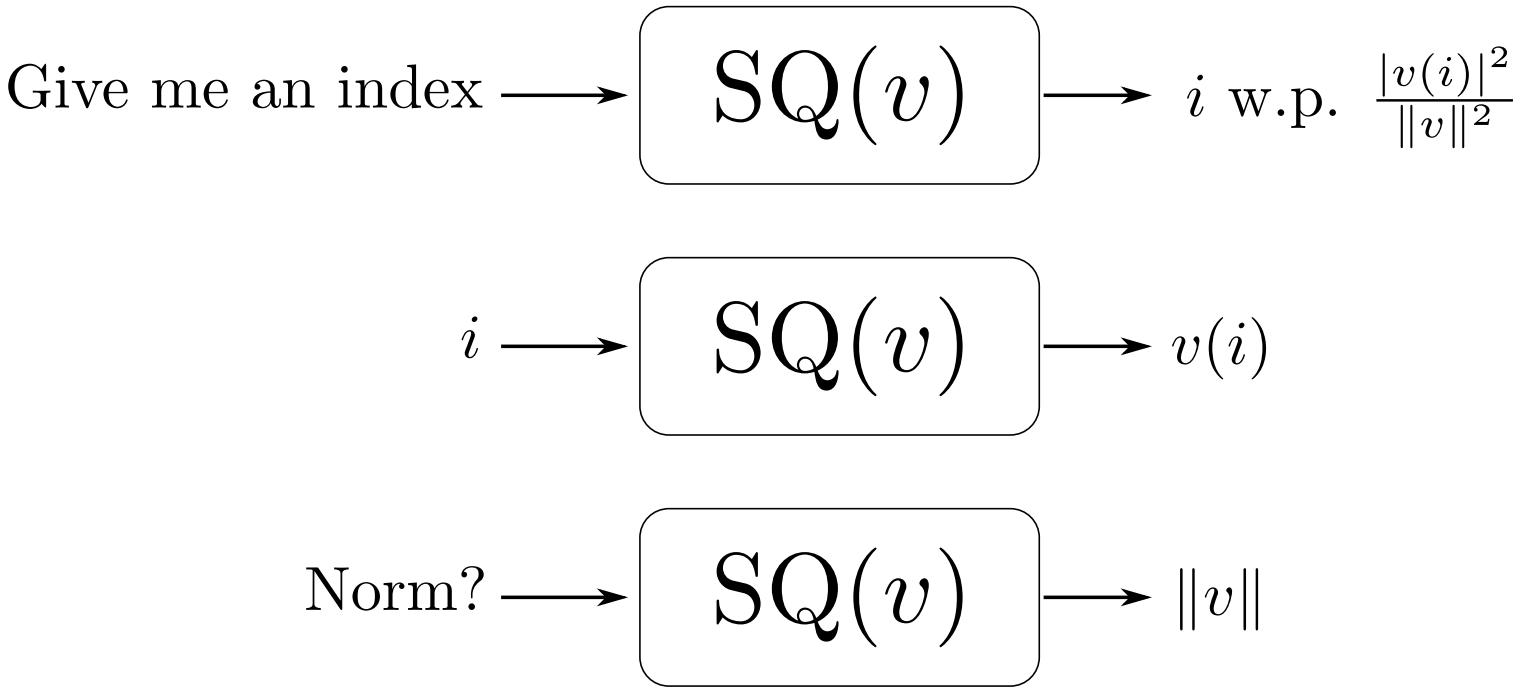
Sampling-query access of a vector

Classical analogue: measuring a state $\rightarrow$ sampling a vector
The **sampling-query** access to v, $\mathrm{SQ}(v)$:



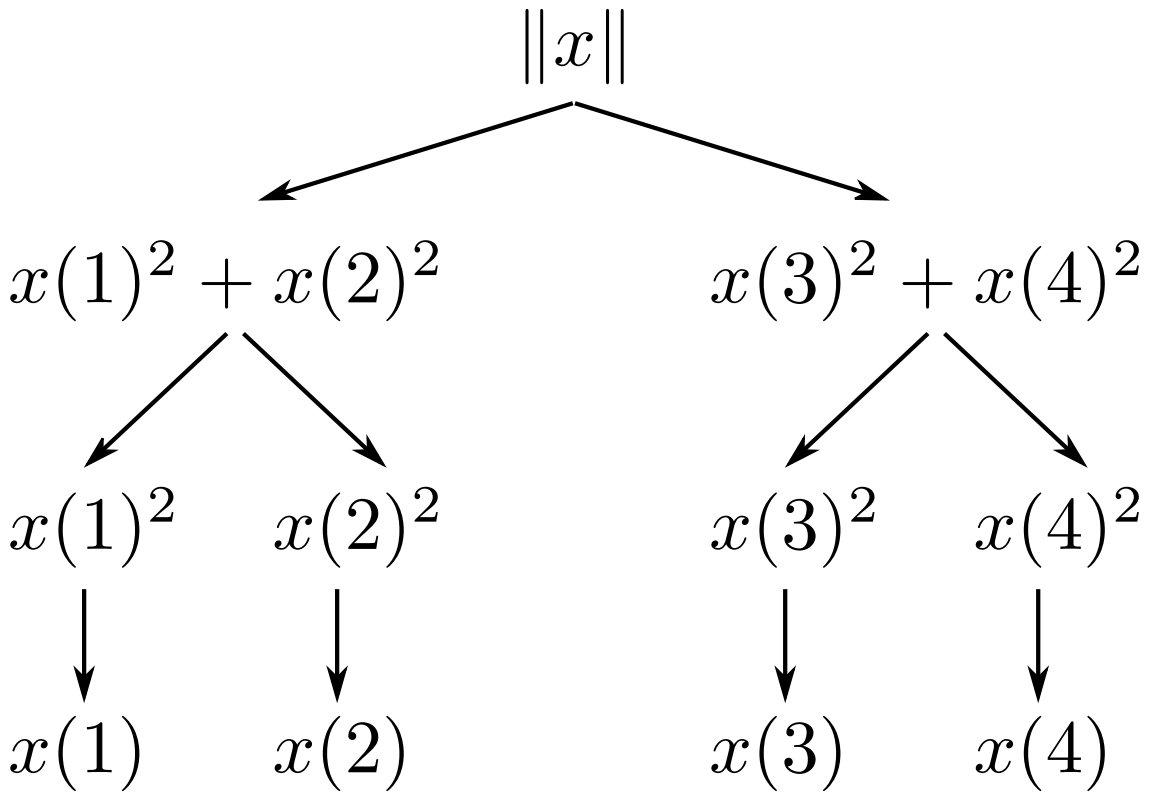
Sampling-query access of a vector

Classical analogue: measuring a state $\rightarrow$ sampling a vector
The **sampling-query** access to v, $\mathrm{SQ}(v)$:



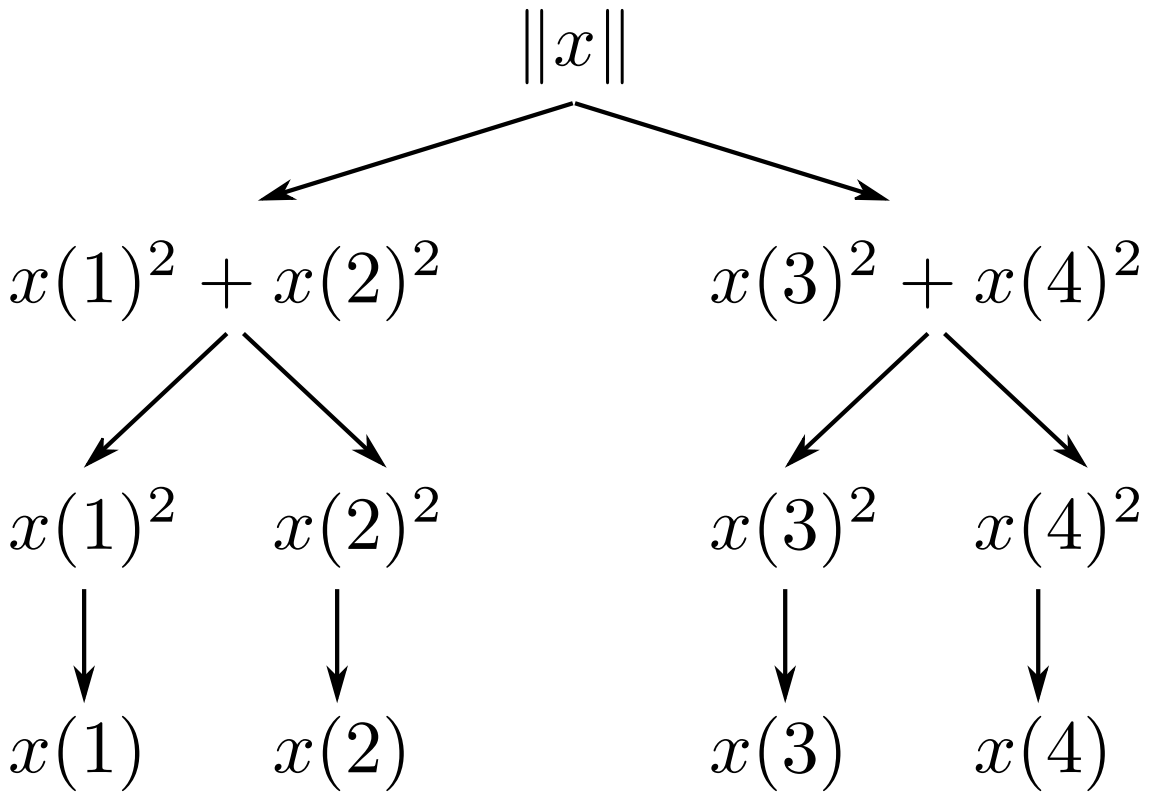
SQ access inspired by QML

[KP17] Preparing $|x\rangle = \sum_{i=1}^{4} \frac{x(i)}{\|x\|}|i\rangle$ from QRAM



SQ access inspired by QML

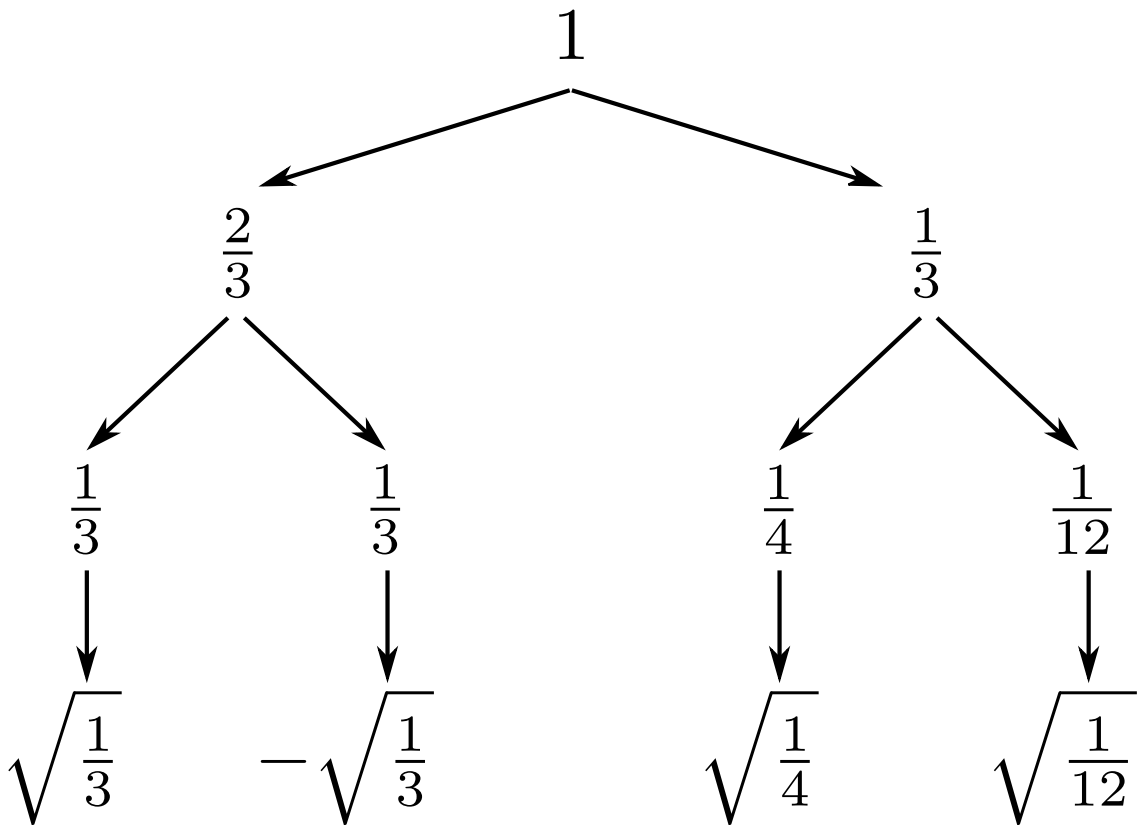
[KP17] Preparing $|x\rangle = \sum_{i=1}^{4} \frac{x(i)}{\|x\|}|i\rangle$ from QRAM



Data structure in QRAM allows for $O(\log n)$-time state preparation

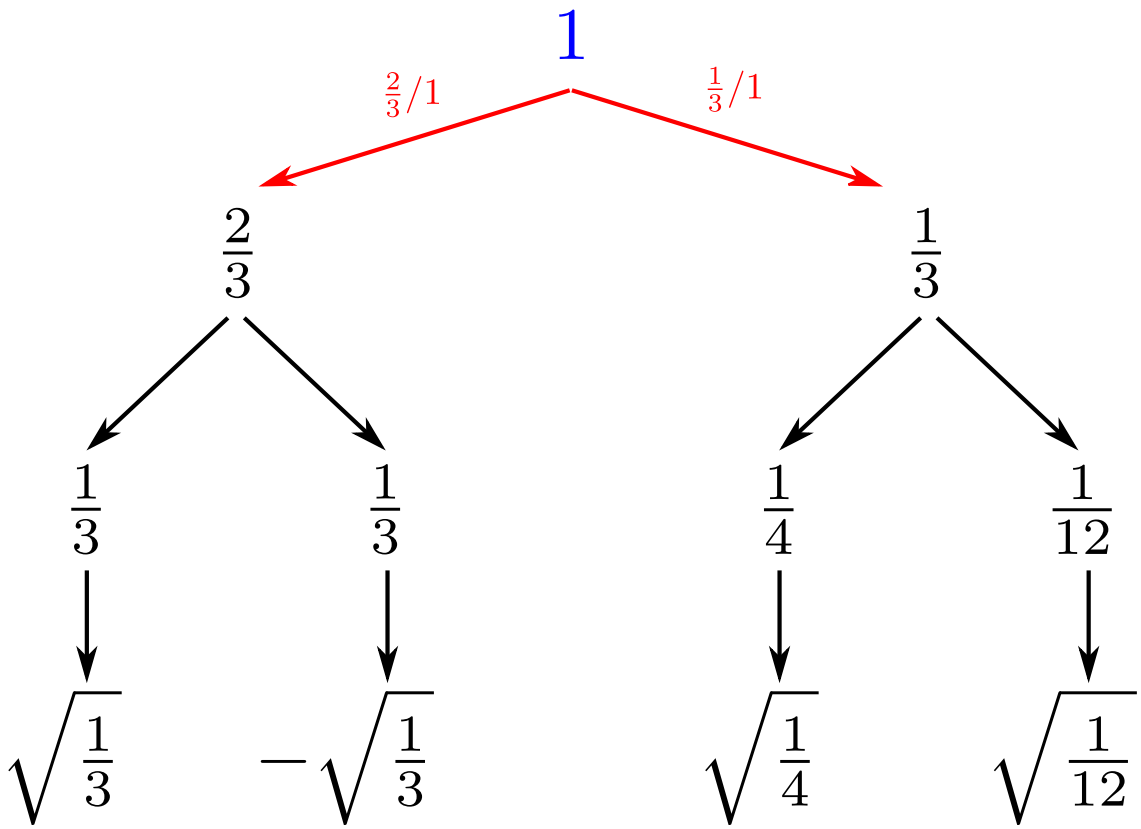
SQ access inspired by QML (cont.)

$x = (\sqrt{1/3}, -\sqrt{1/3}, \sqrt{1/4}, \sqrt{1/12}) \in \mathbb{R}^n$ stored in data structure



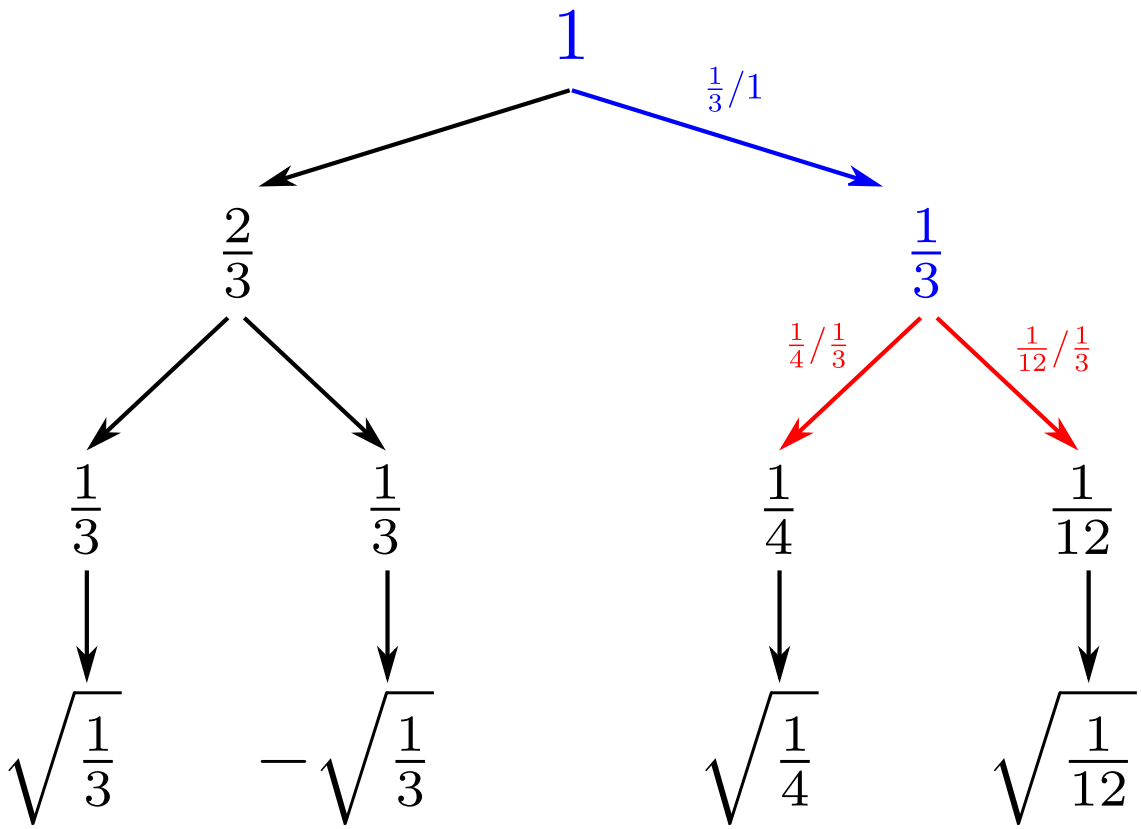
SQ access inspired by QML (cont.)

$x = (\sqrt{1/3}, -\sqrt{1/3}, \sqrt{1/4}, \sqrt{1/12}) \in \mathbb{R}^n$ stored in data structure



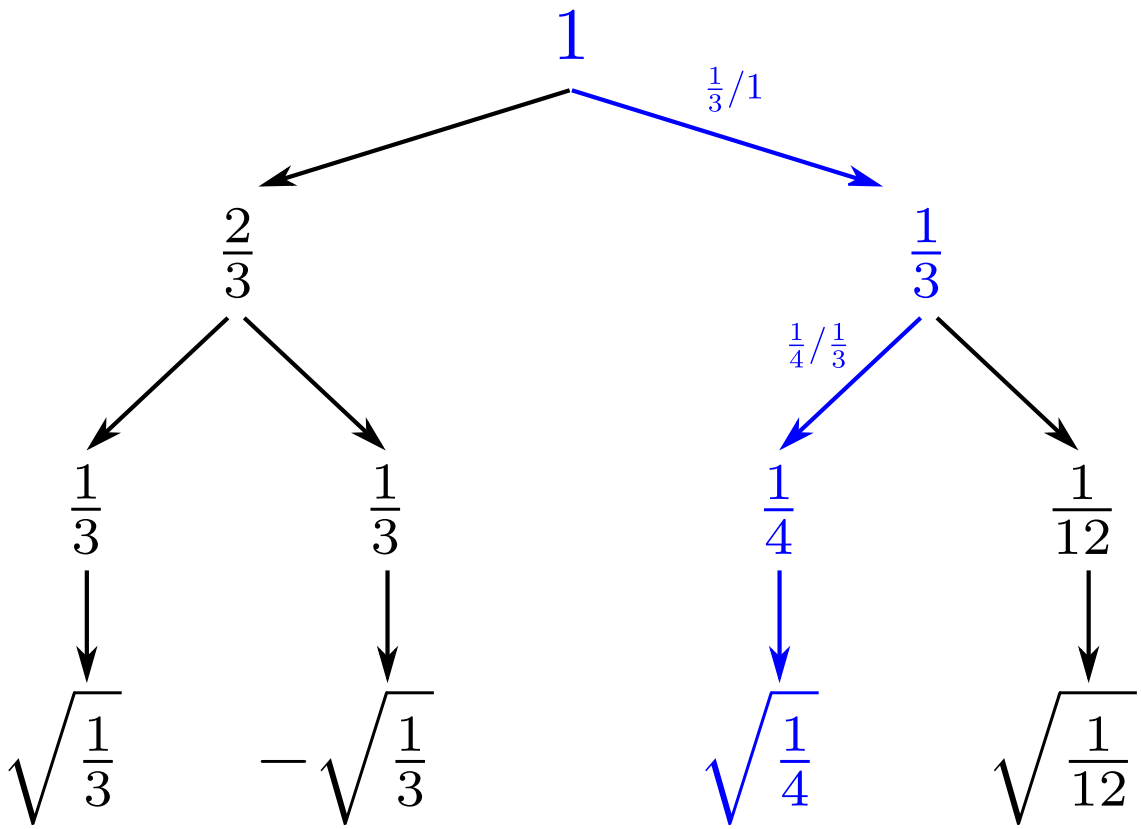
SQ access inspired by QML (cont.)

$x = (\sqrt{1/3}, -\sqrt{1/3}, \sqrt{1/4}, \sqrt{1/12}) \in \mathbb{R}^n$ stored in data structure



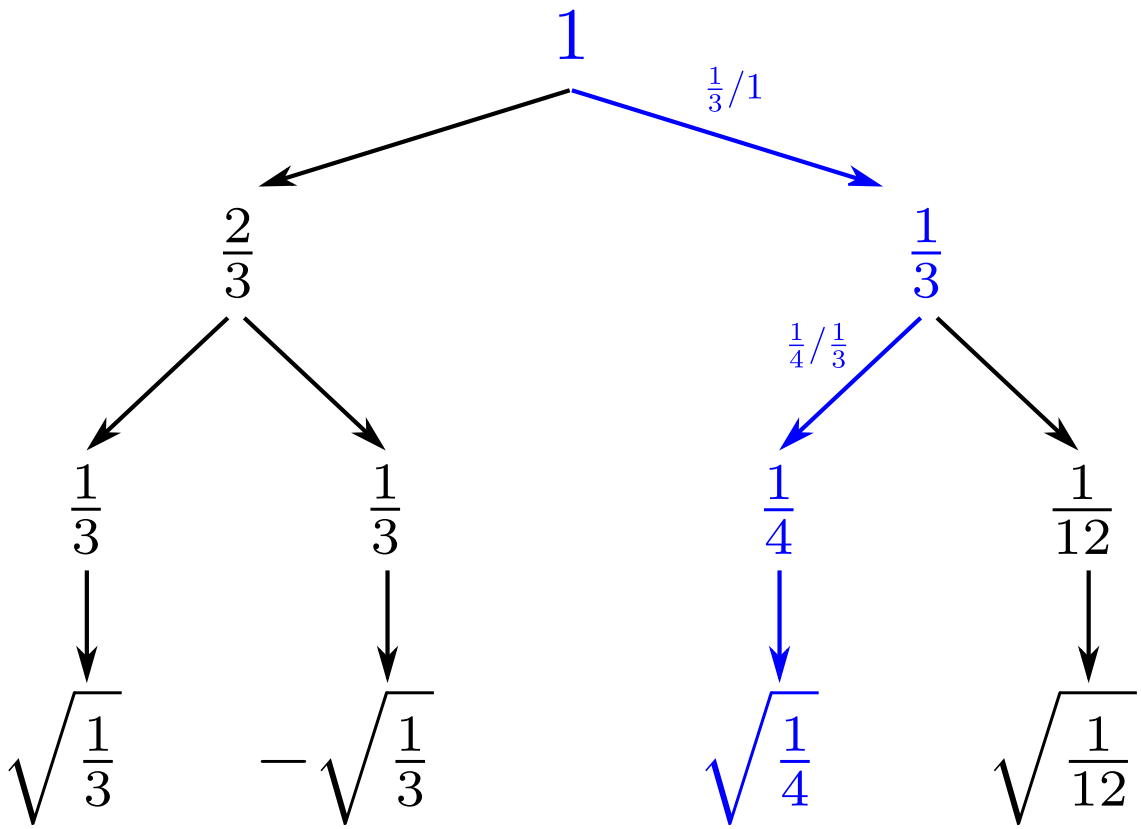
SQ access inspired by QML (cont.)

$x = (\sqrt{1/3}, -\sqrt{1/3}, \sqrt{1/4}, \sqrt{1/12}) \in \mathbb{R}^n$ stored in data structure



SQ access inspired by QML (cont.)

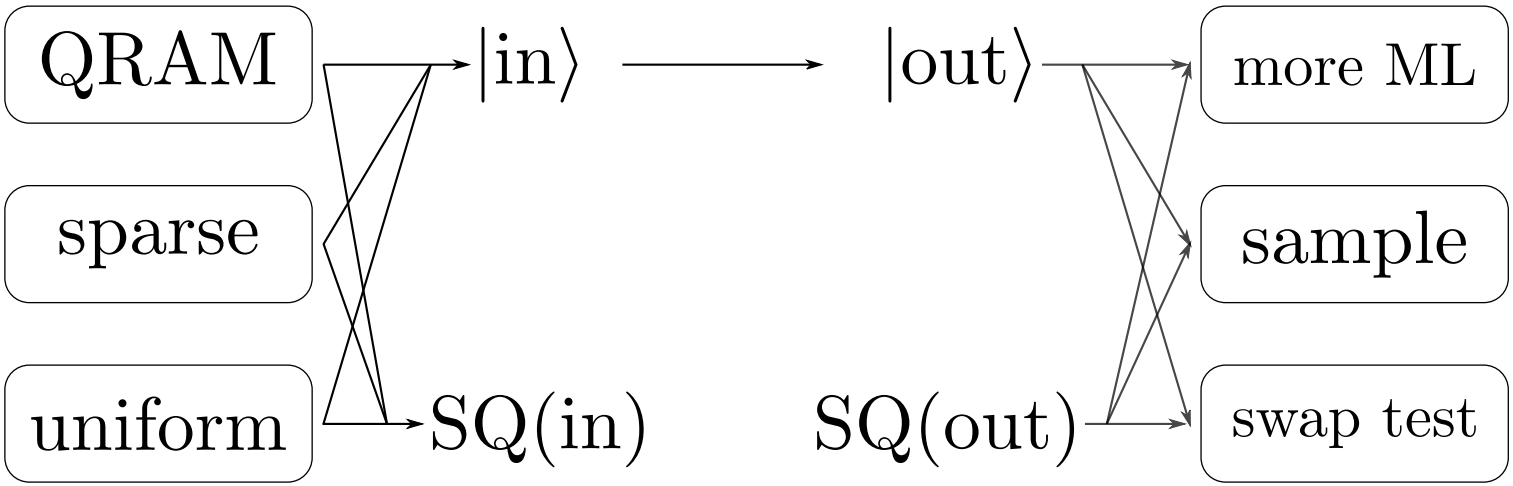
$x = (\sqrt{1/3}, -\sqrt{1/3}, \sqrt{1/4}, \sqrt{1/12}) \in \mathbb{R}^n$ stored in data structure



Data structure for x can simulate $O(\log n)$-time $\mathrm{SQ}(x)$

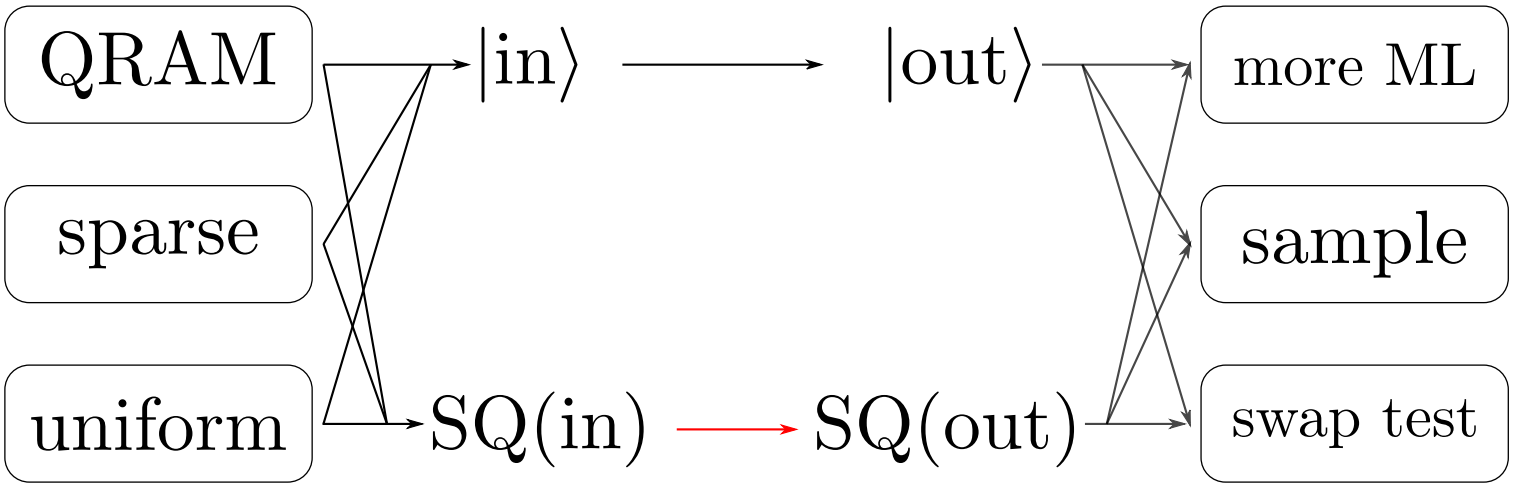
Quantum-inspired algorithms

Discovered by [Tan18a]



Quantum-inspired algorithms

Discovered by [Tan18a]



More definitions

- $\mathrm{Q}(v)$: query access of v. Input i, output $v(i)$.

More definitions

- $\mathrm{Q}(v)$: query access of v. Input i, output $v(i)$.
- $\mathrm{SQ}(A)$: $\mathrm{SQ}(A(1,\cdot)), \ldots, \mathrm{SQ}(A(n,\cdot))$, and $\mathrm{SQ}(\tilde{A})$ where $\tilde{A} = (\|A(1,\cdot)\|, \ldots, \|A(n,\cdot)\|)$

More definitions

- $\mathrm{Q}(v)$: query access of v. Input i, output $v(i)$.
- $\mathrm{SQ}(A)$: $\mathrm{SQ}(A(1,\cdot)), \ldots, \mathrm{SQ}(A(n,\cdot))$, and $\mathrm{SQ}(\tilde{A})$ where $\tilde{A} = (\|A(1,\cdot)\|, \ldots, \|A(n,\cdot)\|)$

Proposition

There exists a data structure that allows for $O(\log n)$-time $\mathrm{SQ}(A)$

Previous quantum-inspired algorithms

Dequantizing low-rank QML algorithms:

- Recommendation systems [KP17] $\rightarrow$ [Tan18a]
- Supervised clustering [LMR13] $\rightarrow$ [Tan18b]
- Principal component analysis [LMR14] $\rightarrow$ [Tan18b]
- Low-rank linear matrix inversion [Reb+18] $\rightarrow$ [GLT18], [CLW18]
- Low-rank SDP solver [Bra+17] $\rightarrow$ [Chi+19]
- Support vector machine [RML14] $\rightarrow$ [DBH19]
- Non-negative matrix factorization [Du+18] $\rightarrow$ [Che+19]

Previous quantum-inspired algorithms

Dequantizing low-rank QML algorithms:

- Recommendation systems [KP17] $\rightarrow$ [Tan18a]
- Supervised clustering [LMR13] $\rightarrow$ [Tan18b]
- Principal component analysis [LMR14] $\rightarrow$ [Tan18b]
- Low-rank linear matrix inversion [Reb+18] $\rightarrow$ [GLT18], [CLW18]
- Low-rank SDP solver [Bra+17] $\rightarrow$ [Chi+19]
- Support vector machine [RML14] $\rightarrow$ [DBH19]
- Non-negative matrix factorization [Du+18] $\rightarrow$ [Che+19]

This work:

SQ-access $\implies$ singular value transformation $\implies$ ML applications

Previous quantum-inspired algorithms

Dequantizing low-rank QML algorithms:

- Recommendation systems [KP17] $\rightarrow$ [Tan18a]
- Supervised clustering [LMR13] $\rightarrow$ [Tan18b]
- Principal component analysis [LMR14] $\rightarrow$ [Tan18b]
- Low-rank linear matrix inversion [Reb+18] $\rightarrow$ [GLT18], [CLW18]
- Low-rank SDP solver [Bra+17] $\rightarrow$ [Chi+19]
- Support vector machine [RML14] $\rightarrow$ [DBH19]
- Non-negative matrix factorization [Du+18] $\rightarrow$ [Che+19]

This work:

SQ-access $\implies$ singular value transformation $\implies$ ML applications

New application

- Hamiltonian simulation (sampling access). Previous work [Rud+18] only has query access and relies on sparsity
- Discriminant analysis [CD16]

Technical tools

SQ for matrix-vector multiplication

Lemma ([Tan18a])

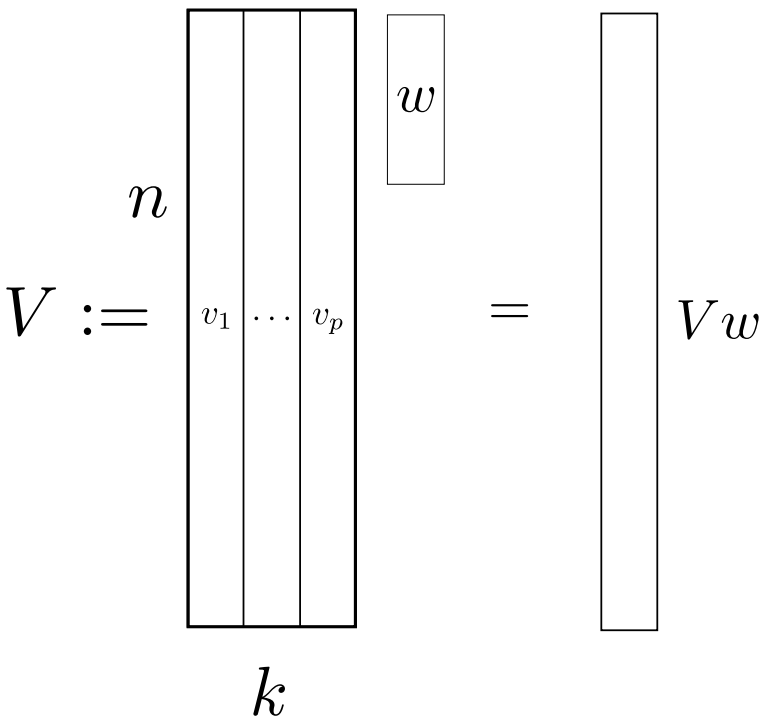
$V \in \mathbb{C}^{n \times k}, w \in \mathbb{C}^k$. *Then* $\mathrm{SQ}(V^\dagger)$ *and* $\mathrm{Q}(w) \implies \mathrm{SQ}(Vw)$

Technical tools

SQ for matrix-vector multiplication

Lemma ([Tan18a])

$V \in \mathbb{C}^{n \times k}, w \in \mathbb{C}^k$. *Then* $\mathrm{SQ}(V^\dagger)$ *and* $\mathrm{Q}(w) \implies \mathrm{SQ}(Vw)$

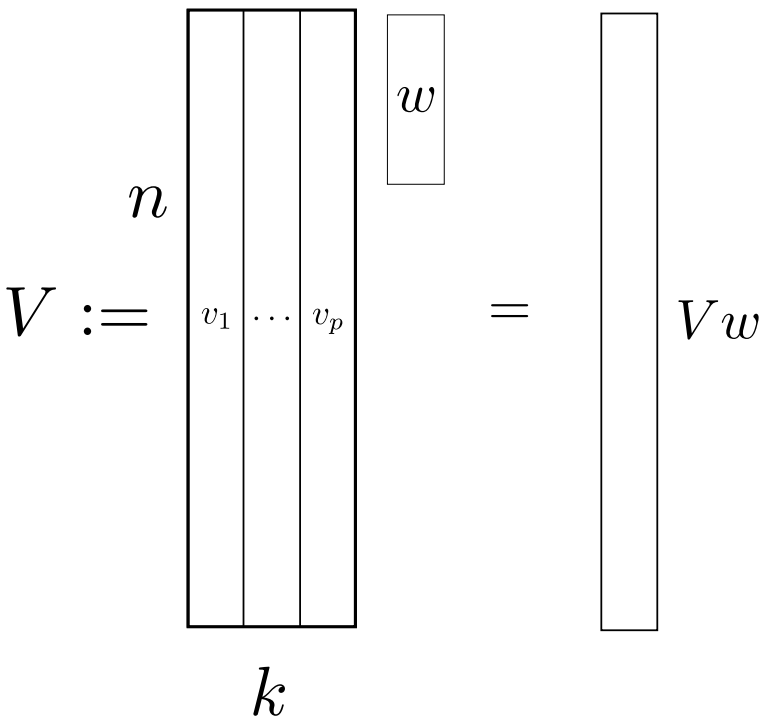


Technical tools

SQ for matrix-vector multiplication

Lemma ([Tan18a])

$V \in \mathbb{C}^{n \times k}, w \in \mathbb{C}^k$. *Then* $\mathrm{SQ}(V^\dagger)$ *and* $\mathrm{Q}(w) \implies \mathrm{SQ}(Vw)$



Performance depends on orthogonality of V

Technical tools (cont.)

Estimating $v^\dagger A w$

Lemma

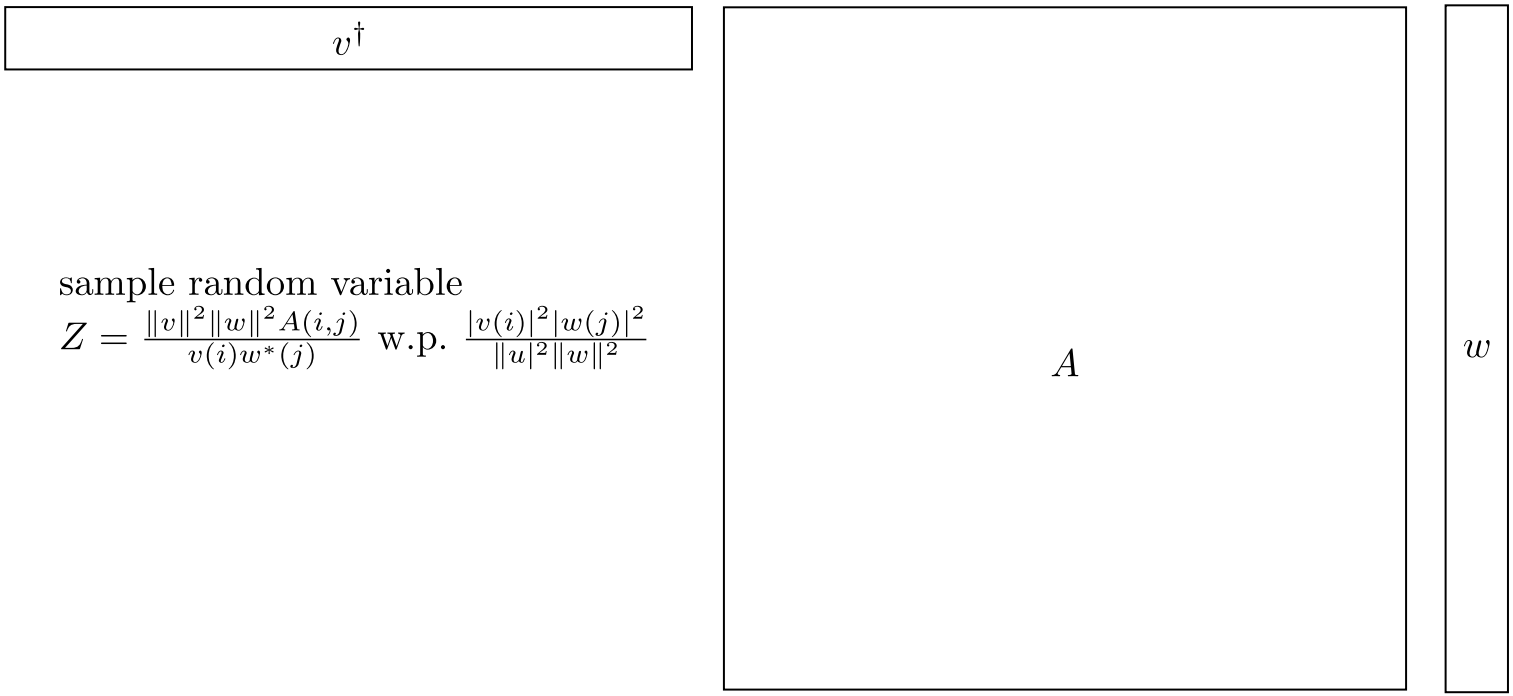
$\mathrm{SQ}(v), \mathrm{SQ}(w)$, *and* $\mathrm{Q}(A) \implies$ *an estimate of* $v^\dagger A w$ *within additive error* ϵ *with high probability in time* $O(\|v\| \|w\| \|A\|_F / \epsilon^2)$.

Technical tools (cont.)

Estimating $v^\dagger A w$

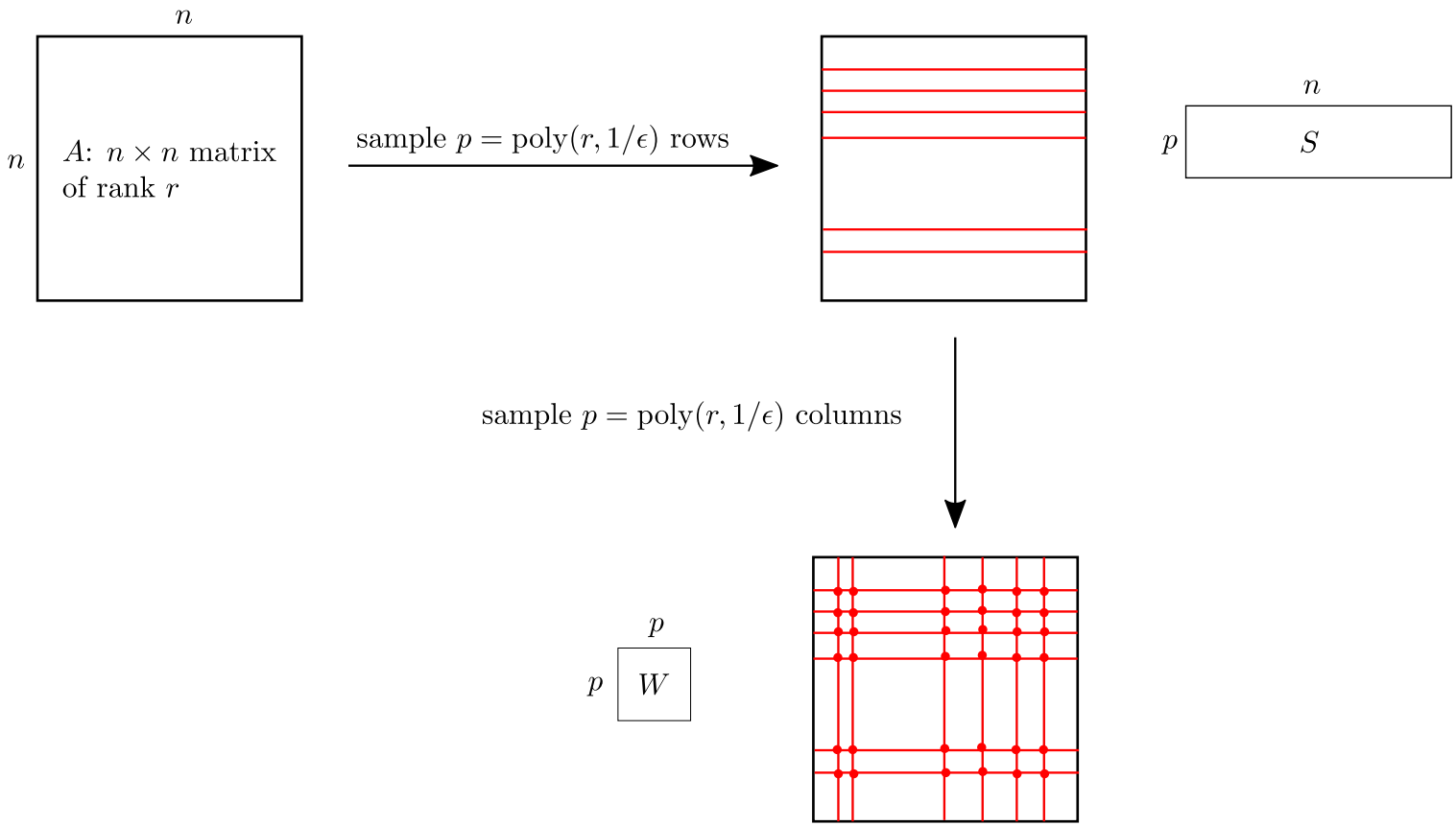
Lemma

$\mathrm{SQ}(v), \mathrm{SQ}(w)$, *and* $\mathrm{Q}(A) \implies$ *an estimate of* $v^\dagger A w$ *within additive error* ϵ *with high probability in time* $O(\|v\| \, \|w\| \, \|A\|_F / \epsilon^2)$.



Low-rank approximation

[FKV04] Sample a small submatrix $A \to S \to W$



Example of subsampling

$$A = \begin{pmatrix} -2.04 & 0.02 & 0.79 & 0.3 & 2.04 & 0.68 & -0.37 & 0.4 & -0.08 & 0.66 \\ -3.44 & 1.5 & 2.68 & -0.93 & 4.04 & -0.02 & -0.78 & 0.96 & -0.58 & -1.99 \\ 5.25 & -2.78 & -3.3 & -1.97 & -4.99 & -5.27 & 1.16 & 1.38 & -3.31 & 2.16 \\ -0.03 & 3.14 & 2.49 & -0.33 & 1.04 & 0.81 & -0.21 & -0.84 & 1.8 & -3.86 \\ -2.51 & 2.45 & 3.35 & -0.48 & 3.87 & 0.17 & -0.59 & 0.62 & 0.48 & -1.63 \\ -0.89 & -0.79 & -1.5 & 1.41 & -0.97 & 3.14 & -0.19 & -1.58 & 1.37 & -0.68 \\ 0.42 & 1.31 & 1.36 & -0.12 & 0.69 & -0.47 & 0.06 & 0.14 & 0.55 & -0.18 \\ -0.22 & 2.63 & 1.94 & 0.78 & 0.85 & 2.08 & -0.18 & -1.3 & 2.58 & -2.31 \\ 0.0 & 1.29 & 1.09 & 0.84 & 0.6 & 1.17 & -0.02 & -0.6 & 1.6 & 0.06 \\ 3.69 & 1.48 & 1.27 & -1.99 & -1.2 & -4.43 & 0.69 & 1.24 & -1.13 & -0.39 \end{pmatrix}$$

$$R = \begin{pmatrix} -94.51 & 41.21 & 73.63 & -25.55 & 111. & -0.55 & -21.43 & 26.38 & -15.93 & -54.67 \\ -7.33 & 87.64 & 64.64 & 25.99 & 28.32 & 69.31 & -6. & -43.32 & 85.97 & -76.97 \\ 87.04 & -46.09 & -54.71 & -32.66 & -82.73 & -87.37 & 19.23 & 22.88 & -54.87 & 35.81 \\ -35.23 & -31.27 & -59.38 & 55.82 & -38.4 & 124.31 & -7.52 & -62.55 & 54.24 & -26.92 \end{pmatrix}$$

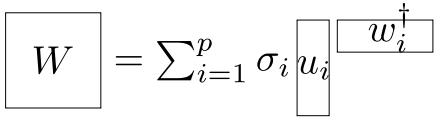
$$R = \begin{pmatrix} -94.51 & 41.21 & 73.63 & -25.55 & 111. & -0.55 & -21.43 & 26.38 & -15.93 & -54.67 \\ -7.33 & 87.64 & 64.64 & 25.99 & 28.32 & 69.31 & -6. & -43.32 & 85.97 & -76.97 \\ 87.04 & -46.09 & -54.71 & -32.66 & -82.73 & -87.37 & 19.23 & 22.88 & -54.87 & 35.81 \\ -35.23 & -31.27 & -59.38 & 55.82 & -38.4 & 124.31 & -7.52 & -62.55 & 54.24 & -26.92 \end{pmatrix}$$

$$C = \begin{pmatrix} 106.02 & 57.6 & -129.5 & 106.02 \\ 93.08 & -94.6 & -10.04 & 93.08 \\ -78.78 & 49.96 & 119.26 & -78.78 \\ -85.51 & -136.6 & -48.28 & -85.51 \end{pmatrix}$$

Image credit: Juan Miguel Arrazola

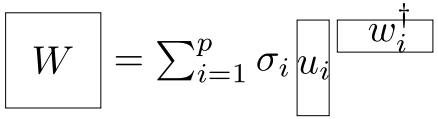
Low-rank projection from subsampling

Compute SVD of W:

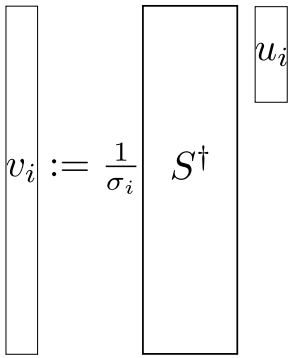


Low-rank projection from subsampling

Compute SVD of W:

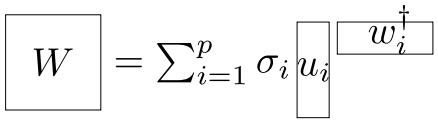


Define vector v_i:

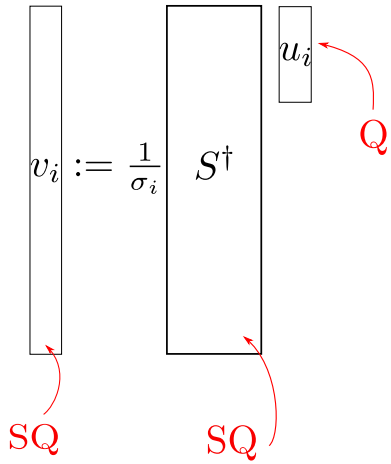


Low-rank projection from subsampling

Compute SVD of W:

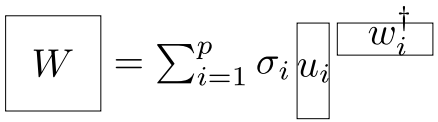


Define vector v_i:

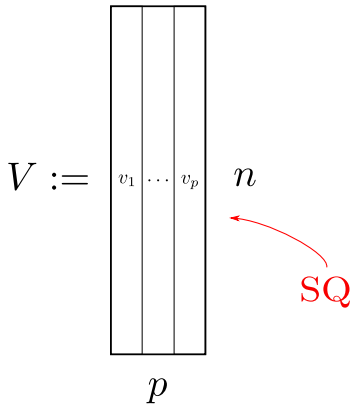


Low-rank projection from subsampling

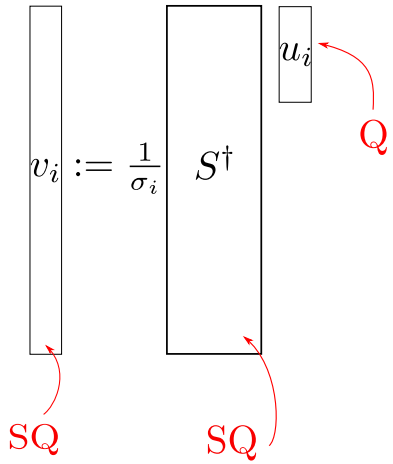
Compute SVD of W:



Define the $n \times p$ matrix V:

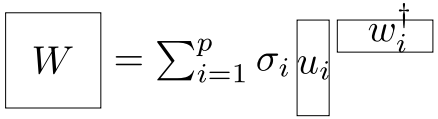


Define vector v_i:

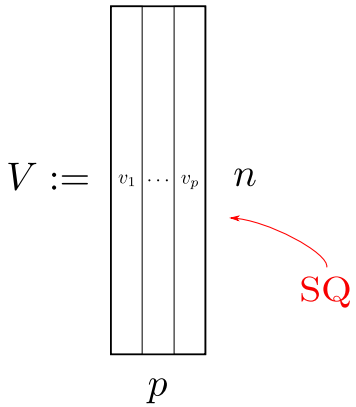


Low-rank projection from subsampling

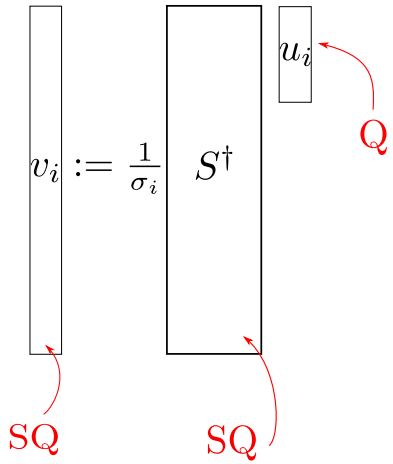
Compute SVD of W:



Define the $n \times p$ matrix V:



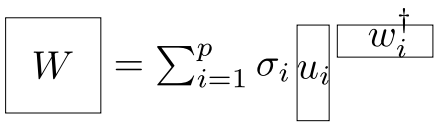
Define vector v_i:



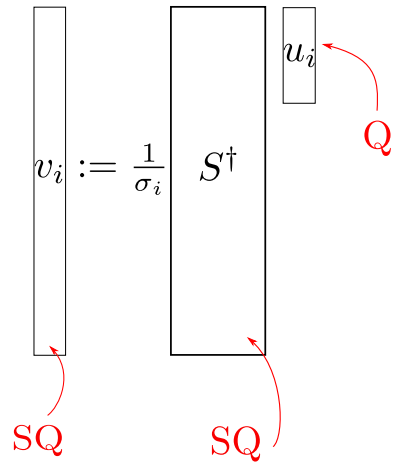
[FKV04]: $\left\| A - AVV^\dagger \right\|_F^2 \leq \epsilon \left\| A \right\|_F^2$

Low-rank projection from subsampling

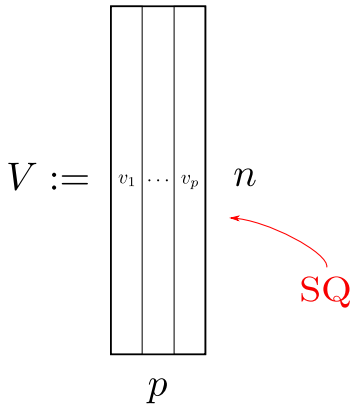
Compute SVD of W:



Define vector v_i:



Define the $n \times p$ matrix V:



[FKV04]: $\left\|A - AVV^\dagger\right\|_F^2 \leq \epsilon \|A\|_F^2$

Our extension: $\left\|\sum_j A - (\sum_j A)VV^\dagger\right\|_F \leq \epsilon \sum_j \|A_j\|_F^2$

Towards singular value transformation — asymmetric approximation

How to do compute $f(A_1 + \cdots + A_\tau)$?

Towards singular value transformation — asymmetric approximation

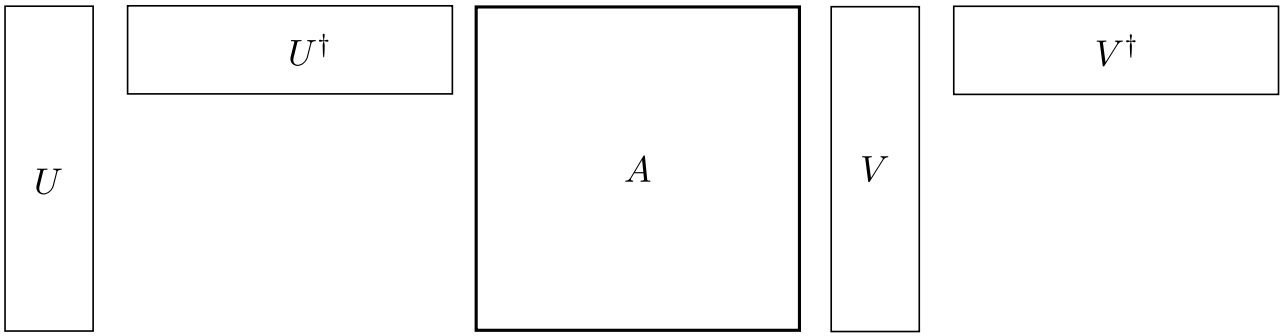
How to do compute $f(A_1 + \cdots + A_\tau)$?

By [FKV04], $\begin{cases} A \approx AVV^\dagger \\ A \approx UU^\dagger A \end{cases} \implies A \approx UU^\dagger AVV^\dagger$

Towards singular value transformation — asymmetric approximation

How to do compute $f(A_1 + \cdots + A_\tau)$?

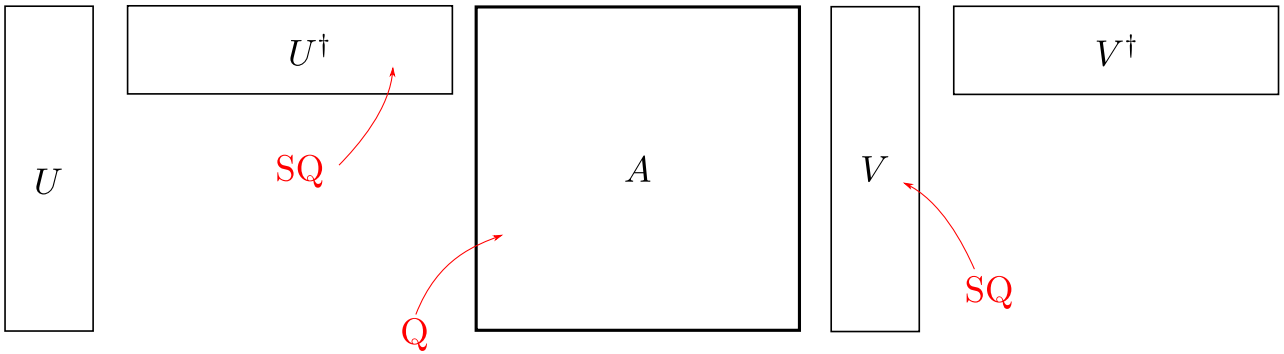
By [FKV04], $\begin{cases} A \approx AVV^\dagger \\ A \approx UU^\dagger A \end{cases} \implies A \approx UU^\dagger AVV^\dagger$



Towards singular value transformation — asymmetric approximation

How to do compute $f(A_1 + \cdots + A_\tau)$?

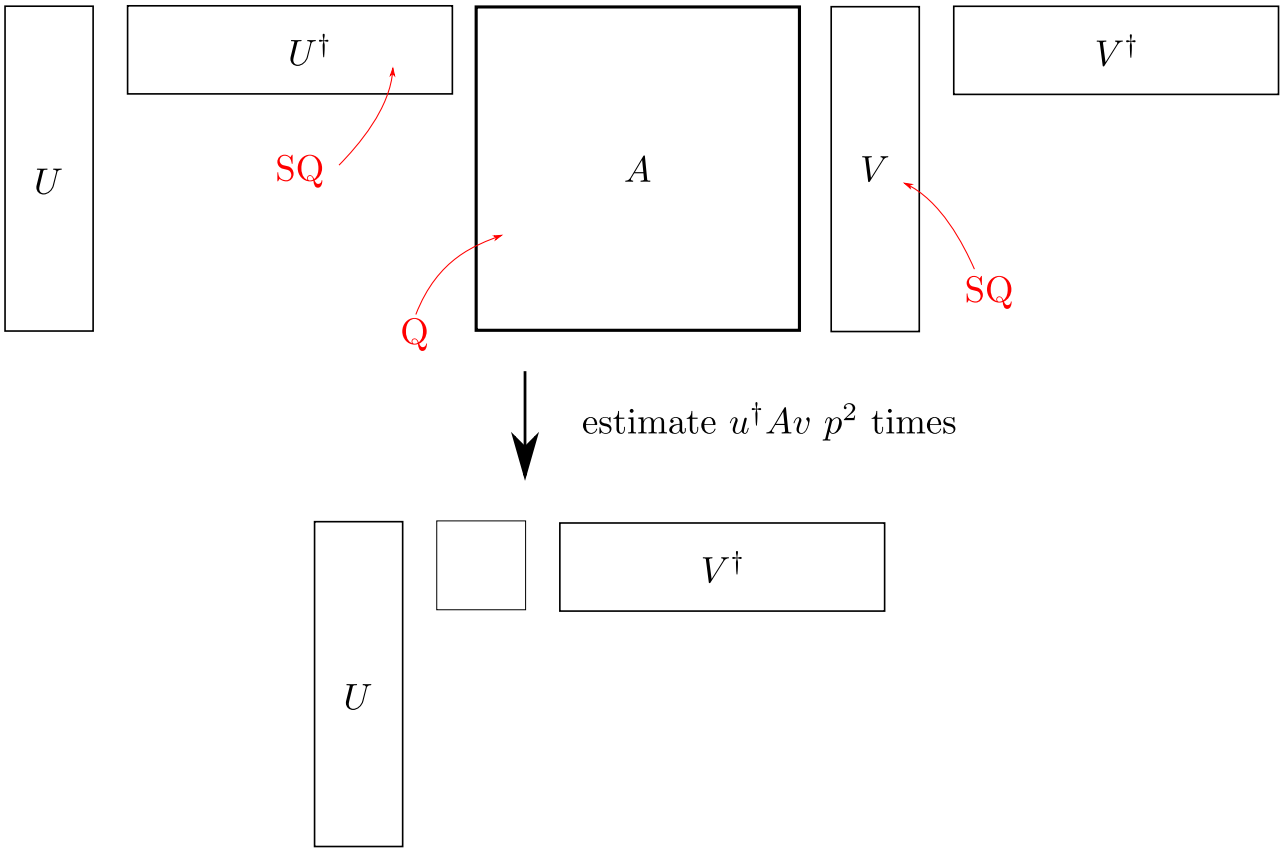
By [FKV04], $\begin{cases} A \approx AVV^\dagger \\ A \approx UU^\dagger A \end{cases} \implies A \approx UU^\dagger AVV^\dagger$



Towards singular value transformation — asymmetric approximation

How to do compute $f(A_1 + \cdots + A_\tau)$?

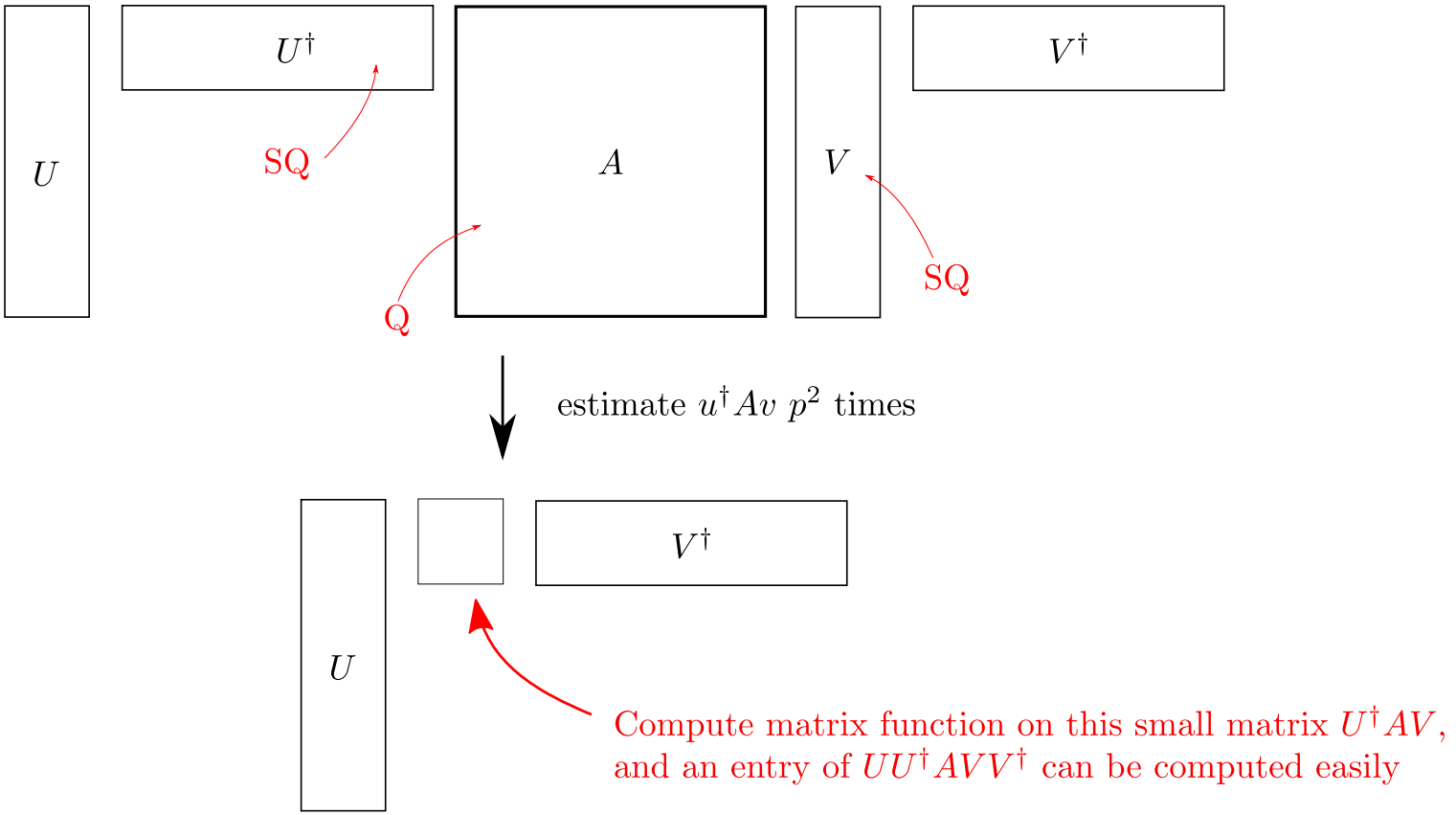
By [FKV04], $\begin{cases} A \approx AVV^\dagger \\ A \approx UU^\dagger A \end{cases} \implies A \approx UU^\dagger AVV^\dagger$



Towards singular value transformation — asymmetric approximation

How to do compute $f(A_1 + \cdots + A_\tau)$?

By [FKV04], $\begin{cases} A \approx AVV^\dagger \\ A \approx UU^\dagger A \end{cases} \implies A \approx UU^\dagger AVV^\dagger$



Main results

- Singular value transformation for f L-Lipschitz continuous:
  $|f(x) - f(y)| \leq L|x - y|$
  $\mathrm{SQ}(A_1), \ldots, \mathrm{SQ}(A_\tau) \to \mathrm{SQ}(f(A))$ where $A = A_1 + \cdots + A_\tau$

Theorem (Informal)

Given $\mathrm{SQ}(A_1), \ldots, \mathrm{SQ}(A_\tau)$, *and* L*-Lipschitz continuous* f. *In* $O(\mathrm{poly}(\sum_j \|A_j\|_F^2, \tau, L)\,\mathrm{polylog}(n))$ *time, we can obtain* $\mathrm{SQ}(B)$ *such that* $B \approx f(A_1 + \cdots + A_\tau)$.

Main results

- Singular value transformation for f L-Lipschitz continuous:
  $|f(x) - f(y)| \leq L|x - y|$
  $\mathrm{SQ}(A_1), \ldots, \mathrm{SQ}(A_\tau) \to \mathrm{SQ}(f(A))$ where $A = A_1 + \cdots + A_\tau$

Theorem (Informal)

Given $\mathrm{SQ}(A_1), \ldots, \mathrm{SQ}(A_\tau)$, *and* L*-Lipschitz continuous* f. *In* $O(\mathrm{poly}(\sum_j \|A_j\|_F^2, \tau, L)\,\mathrm{polylog}(n))$ *time, we can obtain* $\mathrm{SQ}(B)$ *such that* $B \approx f(A_1 + \cdots + A_\tau)$.

- Matrix multiplication
  $\mathrm{SQ}(A)$ and $\mathrm{SQ}(B) \to \mathrm{SQ}(AB)$

Theorem (Informal)

Given $\mathrm{SQ}(A)$ *and* $\mathrm{SQ}(B)$, *In* $O(\mathrm{poly}(\|A\|_F, \|B\|_F)\,\mathrm{polylog}(n))$ *time, we can obtain* $\mathrm{SQ}(C)$ *such that* $C \approx AB$.

Applications — Hamiltonian simulation

- Problem: given *Hamiltonian* $H \in \mathbb{C}^{n \times n}$ and evolution time t, for any initial state $|\psi\rangle$, approximate $e^{-iHt}|\psi\rangle$.

Applications — Hamiltonian simulation

- Problem: given *Hamiltonian* $H \in \mathbb{C}^{n \times n}$ and evolution time t, for any initial state $|\psi\rangle$, approximate $e^{-iHt}|\psi\rangle$.
- Quantum: $O((t\|H\| + \text{polylog}(1/\epsilon))\text{polylog}(n))$.

Applications — Hamiltonian simulation

- Problem: given *Hamiltonian* $H \in \mathbb{C}^{n \times n}$ and evolution time t, for any initial state $|\psi\rangle$, approximate $e^{-iHt}|\psi\rangle$.
- Quantum: $O((t\|H\| + \text{polylog}(1/\epsilon))\text{polylog}(n))$.
- $f(x) = e^{itx}$ is t-Lipschitz.

Applications — Hamiltonian simulation

- Problem: given *Hamiltonian* $H \in \mathbb{C}^{n \times n}$ and evolution time t, for any initial state $|\psi\rangle$, approximate $e^{-iHt}|\psi\rangle$.
- Quantum: $O((t\|H\| + \mathrm{polylog}(1/\epsilon))\mathrm{polylog}(n))$.
- $f(x) = e^{itx}$ is t-Lipschitz.
- Classical: given $\mathrm{SQ}(H)$ and $\mathrm{SQ}(|\psi\rangle)$, $\mathrm{SQ}(e^{-iHt}|\psi\rangle)$ can be obtained in $O(\mathrm{poly}(t, \|H\|_F, 1/\epsilon)\,\mathrm{polylog}(n))$ time.

Applications — Low-rank matrix inversion

- Problem: given matrix $A \in \mathbb{C}^{m \times n}$ and vector $|b\rangle \in \mathbb{C}^m$, approximate $A^{-1}|b\rangle$.

Applications — Low-rank matrix inversion

- Problem: given matrix $A \in \mathbb{C}^{m \times n}$ and vector $|b\rangle \in \mathbb{C}^m$, approximate $A^{-1}|b\rangle$.
- Quantum: the HHL algorithm (sparse) and quantum PCA (low-rank)

Applications — Low-rank matrix inversion

- Problem: given matrix $A \in \mathbb{C}^{m \times n}$ and vector $|b\rangle \in \mathbb{C}^m$, approximate $A^{-1}|b\rangle$.
- Quantum: the HHL algorithm (sparse) and quantum PCA (low-rank)
- $f(x) = \begin{cases} 0, & \text{for } 0 \leq x < \theta(1-\xi) \\ \frac{1}{\xi\theta^2}(x - (\theta(1-\xi)), & \text{for } \theta(1-\xi) \leq x < \theta \\ \frac{1}{x}, & \text{for } \theta \leq x \end{cases}$ is $\frac{1}{\theta\xi}$-Lipschitz.

Applications — Low-rank matrix inversion

- Problem: given matrix $A \in \mathbb{C}^{m \times n}$ and vector $|b\rangle \in \mathbb{C}^m$, approximate $A^{-1}|b\rangle$.
- Quantum: the HHL algorithm (sparse) and quantum PCA (low-rank)
- $f(x) = \begin{cases} 0, & \text{for } 0 \leq x < \theta(1-\xi) \\ \frac{1}{\xi\theta^2}(x - (\theta(1-\xi)), & \text{for } \theta(1-\xi) \leq x < \theta \\ \frac{1}{x}, & \text{for } \theta \leq x \end{cases}$ is $\frac{1}{\theta\xi}$-Lipschitz.
- Classical: Given $\mathrm{SQ}(A_1), \ldots, \mathrm{SQ}(A_\tau)$, then $\mathrm{SQ}((\sum_\ell A)^{-1}|b\rangle)$ can be obtained in $O(\mathrm{polylog}(m,n)\,\mathrm{poly}(\sum_\ell \|A_\ell\|_F^2, 1/\epsilon))$ time.

Applications — Discriminant analysis

Data set: $X = \{x_1, \ldots, x_m\} \in \mathbb{R}^{m \times n}$ each belonging to one of k classes.
μ_c: the mean of class $c \in [k]$ and $\overline{x}$: mean of all data points.

$$S_B = \sum_{c=1}^{k} (\mu_c - \overline{x})(\mu_c - \overline{x})^T$$

$$S_W = \sum_{c=1}^{k} \sum_{x \in c} (\mu_c - x)(\mu_c - x)^T.$$

- Problem: given $\mathrm{SQ}(X)$, find the largest p eigenvalues and eigenvectors of $S_W^{-1} S_B$.

Applications — Discriminant analysis

Data set: $X = \{x_1, \ldots, x_m\} \in \mathbb{R}^{m \times n}$ each belonging to one of k classes.
μ_c: the mean of class $c \in [k]$ and $\overline{x}$: mean of all data points.

$$S_B = \sum_{c=1}^{k} (\mu_c - \overline{x})(\mu_c - \overline{x})^T$$

$$S_W = \sum_{c=1}^{k} \sum_{x \in c} (\mu_c - x)(\mu_c - x)^T.$$

- Problem: given $\mathrm{SQ}(X)$, find the largest p eigenvalues and eigenvectors of $S_W^{-1} S_B$.
- Quantum: Quantum PCA

Applications — Discriminant analysis

Data set: $X = \{x_1, \ldots, x_m\} \in \mathbb{R}^{m \times n}$ each belonging to one of k classes. μ_c: the mean of class $c \in [k]$ and $\overline{x}$: mean of all data points.

$$S_B = \sum_{c=1}^{k} (\mu_c - \overline{x})(\mu_c - \overline{x})^T$$

$$S_W = \sum_{c=1}^{k} \sum_{x \in c} (\mu_c - x)(\mu_c - x)^T.$$

- Problem: given $\mathrm{SQ}(X)$, find the largest p eigenvalues and eigenvectors of $S_W^{-1} S_B$.
- Quantum: Quantum PCA
- Classical: Given $\mathrm{SQ}(X_1), \ldots, \mathrm{SQ}(X_k)$, then $\mathrm{SQ}(S_W^{-1} S_B)$ can be obtained in $O(\mathrm{polylog}(m, n)\, \mathrm{poly}(\|X\|_F, 1/\epsilon))$ time.

Applications — Discriminant analysis

Data set: $X = \{x_1, \ldots, x_m\} \in \mathbb{R}^{m \times n}$ each belonging to one of k classes. μ_c: the mean of class $c \in [k]$ and $\overline{x}$: mean of all data points.

$$S_B = \sum_{c=1}^{k} (\mu_c - \overline{x})(\mu_c - \overline{x})^T$$

$$S_W = \sum_{c=1}^{k} \sum_{x \in c} (\mu_c - x)(\mu_c - x)^T.$$

- Problem: given $\mathrm{SQ}(X)$, find the largest p eigenvalues and eigenvectors of $S_W^{-1} S_B$.
- Quantum: Quantum PCA
- Classical: Given $\mathrm{SQ}(X_1), \ldots, \mathrm{SQ}(X_k)$, then $\mathrm{SQ}(S_W^{-1} S_B)$ can be obtained in $O(\mathrm{polylog}(m, n) \, \mathrm{poly}(\|X\|_F, 1/\epsilon))$ time.
- $S_W^{-1} S_B$ similar to $S_B^{1/2} S_W^{-1} S_B^{1/2}$: finding right singular vectors of $S_W^{-1/2} S_B^{1/2}$

Thank you!