4 Introducing quantum-inspired linear algebra

We have established a theory of quantum linear algebra based on the block-encoding: you might have noticed that, with block-encodings, we can implicitly manipulate exponentially large matrices in polynomial time. This raises a natural question: perhaps we can harness Nature's linear algebra processor to manipulate data exponentially faster than we can with classical computers. The key work in this line is Harrow, Hassidim, and Lloyd's quantum algorithm for sampling from the solution to a sparse system of linear equations [HHL09]. QML has since rapidly developed into an active field of study with numerous proposals for quantum speedups for machine learning tasks in domains ranging from recommendation systems [KP17] to topological data analysis [LGZ16].

A key tool underlying many QML algorithms is the observation that certain kinds of *data structures* could allow for efficient preparation of block encodings of arbitrary matrices, assuming the ability to query these data structures in superposition (i.e. assuming that the data is in QRAM). In particular, this allows for $\log(rc)$-block encodings of $A/\|A\|_{\mathrm{F}}$. Many QML algorithms relied on this data structure for exponential speedup, with the belief that this additional assumption would not affect it. However, it was discovered that classical algorithms given this data structure can achieve the same results up to polynomial slowdown. These are known as "dequantized" algorithms. The existence of a dequantized algorithm means that its quantum counterpart cannot give exponential speedups on classical data, illuminating the landscape of QML speedups.

Quantum singular value transformation captures essentially all known linear algebraic QML techniques [MRTC21], including all prior dequantized QML algorithms (up to minor technical details), so it is our natural target for dequantizing. We cannot hope to dequantize *all* of QSVT, because with sparse input data encoded appropriately, QSVT can simulate algorithms for BQP-complete problems [JW06; HHL09]. However, we show that we can dequantize the QSVT framework, provided that the input data comes in the state preparation data structure commonly used for quantum linear algebra. Such data structures only allow for efficient QML when the input is low-rank. Nevertheless, they are the only way we know how to run quantum linear algebra on unstructured classical data, so this setting covers all QML algorithms that do not rely on sparsity assumptions. We present a classical analogue of the QSVT framework that is only polynomially slower when the input is low rank, and apply it to dequantize QML algorithms.

Bibliographic note: the results here are all taken from [CGLLTW22].

4.1 Example 1: The swap test, and access models

It seems counterintuitive that classical linear algebra algorithms can perform nearly as well as quantum ones, even on classical data. In some sense, what dequantization shows is that some quantum linear algebra algorithms do not fully exploit "quantumness," since they can be mimicked classically using sampling procedures. We'll investigate a simple example of a quantum linear algebra algorithm: the *swap test* [BCWW01].

Suppose we have two d-dimensional vectors $\phi, \psi \in \mathbb{C}^d$, both with unit norm. We wish to compute their overlap $|\langle\phi|\psi\rangle|^2$. There is a quantum algorithm, the swap test (shown in Fig. 2), to solve this: prepare the $\log(d)$-qubit quantum states $|\phi\rangle = \sum_{i=1}^{d} \phi_i |i\rangle$ and $|\psi\rangle = \sum_{i=1}^{d} \psi_i |i\rangle$, along with one additional qubit in the state $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Then,

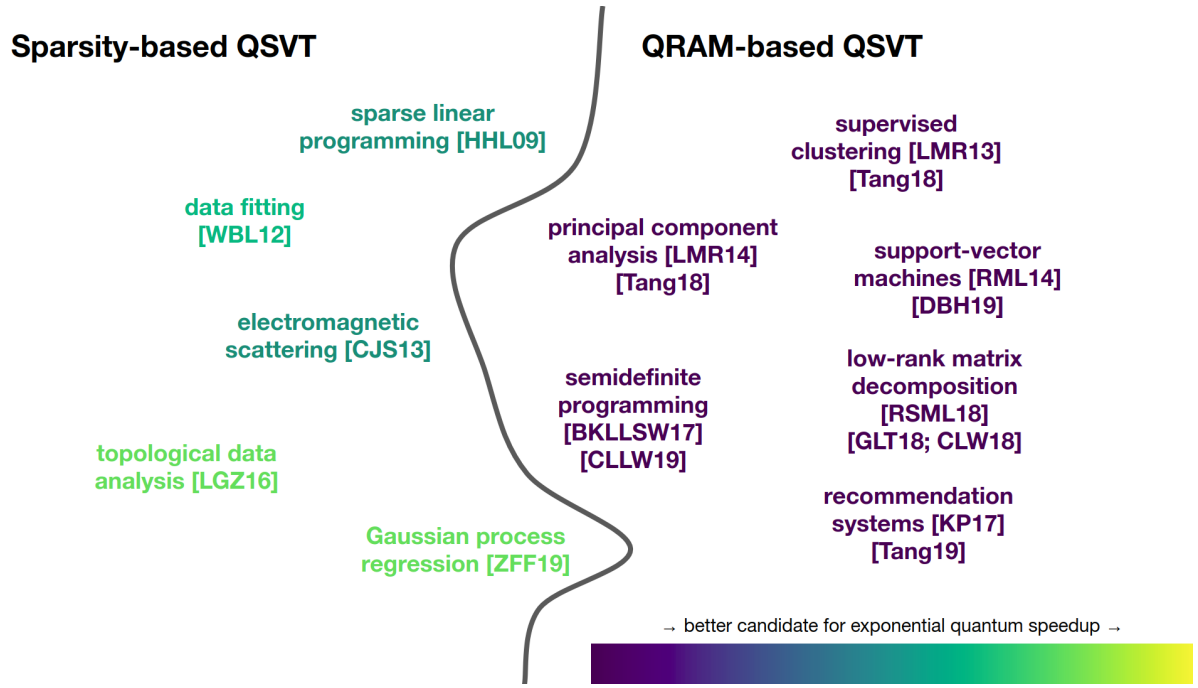

Figure 1: Pictured is the landscape of quantum machine learning algorithms after dequantization. In this thesis, we dequantize all of the algorithms on the right-hand side, showing that they do not give exponential speedups on classical data. All of these algorithms can be placed in the QSVT framework, and in this setting, the dequantized algorithms are precisely the algorithms that do not rely on sparsity assumptions.

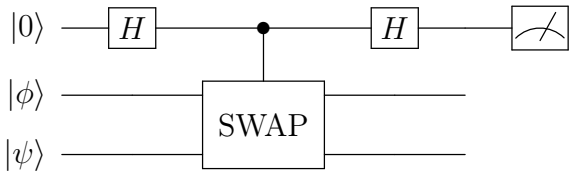

Figure 2: The quantum circuit for the swap test, taken from [BCWW01, Figure 1].

apply a controlled SWAP between $|\phi\rangle$ and $|\psi\rangle$, with the additional qubit as the control, and then measure this qubit in the Hadamard basis; the measurement produces 1 with probability $\frac{1}{2} - \frac{1}{2}|\langle\phi|\psi\rangle|^2$, so we can use it to estimate the overlap. Averaging over more runs of this circuit gives an estimate to 0.01 error with only $\mathcal{O}(\log(d))$ quantum gates and a constant number of copies of the input states. Even approximating overlaps using classical computers requires $\Omega(d)$ time, since we need to read this many entries of the input to distinguish the two cases $\phi = e_i$, $\psi = e_i$ and $\phi = e_i$, $\psi = e_j$. So, we might naively conclude that the swap test achieves an exponential quantum advantage in the task of "computing overlaps". This is not as farfetched a claim as it might appear: the general version of this task, where we wish to estimate $|\langle 0|^{\otimes n} U |0\rangle^{\otimes n}|$ for $U \in \mathbb{C}^{2^n \times 2^n}$ a unitary matrix encoded as a poly(n)-sized quantum circuit, indeed gives a quantum advantage (since this task is BQP-hard). Further, this idea has been proposed before in QML: a preprint of Lloyd, Mohseni, and Rebentrost claims to achieve an exponential quantum advantage for clustering with the swap test, computing the distance of a vector to a centroid by estimating the overlap of states like the above [LMR13].

However, the comparison between $\mathcal{O}(\log(d))$ and $\Omega(d)$ hides the difference in input models: the quantum algorithm requires copies of the states $|\phi\rangle$ and $|\psi\rangle$, and the classical lower bound assumes that we are only given the input vectors as lists of entries. For applications to machine learning, it's reasonable to receive the data in the latter form, since the data is classical (in that it comes from classical sources, as is the case for the vast majority of data). For example, machine learning datasets are stored in this way. This leads us to the question: given ϕ and ψ classically, how can we efficiently prepare their corresponding quantum states? Though state preparation assumptions like these are common in quantum linear algebra, they cannot be satisfied in general: the typical way of satisfying them is to assume pre-processing to load the input into a certain kind of data structure in *quantum random access memory* (QRAM) [GLM08; Pra14; JR23]. QRAM is a speculative piece of quantum hardware which supports storing n bits of data and subsequently querying that data in superposition in (functionally) polylog(n) time, similarly to how we consider classical RAM; for the sake of comparison, we assume the existence of QRAM.[1] If we assume that input is given in this data structure (see Fig. 3) for the sake of the quantum computer, then for a fair comparison, we should give our classical computer this same data structure.

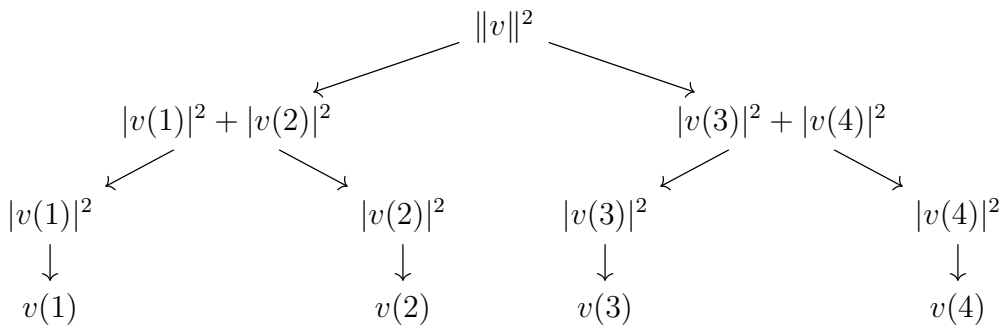

Figure 3: Dynamic data structure used to perform efficient state preparation of a vector $v \in \mathbb{C}^4$. The values displayed are stored in QRAM, along with pointers to other values as designated by the entries. Observe that, by starting from the root of the tree and recursing appropriately, we can sample $i \in [4]$ with probability proportional to $|v(i)|^2$ using only *classical* access to the data structure. See **??** part (b) for more information. A variety of data structures have similar properties, but this one has the advantage of supporting updating entries in $\mathcal{O}(\log n)$ accesses.

If A is in a state preparation data structure in QRAM (like the vector case, see Fig. 4), we can implement a block-encoding of $A/\|A\|_{\mathrm{F}}$ efficiently [GSLW19, Lemma 50]. This type of block-encoding is the one commonly used for quantum linear algebra algorithms on classical data, since it works for arbitrary matrices and vectors, paying only a $\|A\|_{\mathrm{F}}/\|A\|$ (square root of stable rank) factor in sub-normalization.

If ϕ in this data structure, a classical computer can draw independent samples $i \in [n]$ with probability proportional to $|\phi(i)|^2$ with $\mathcal{O}(\log(n))$ accesses. Equipped with this additional

[1] Of course, neither forms of RAM could be "truly" polylog(n) time, since storing n bits of data requires poly(n) space and therefore poly(n) time for the information to travel across that amount of space. The goal would be to optimize QRAM as well as classical RAM, so that accesses can be treated as $\mathcal{O}(\log(n))$ time, the cost of simply writing down the pointer into the data.

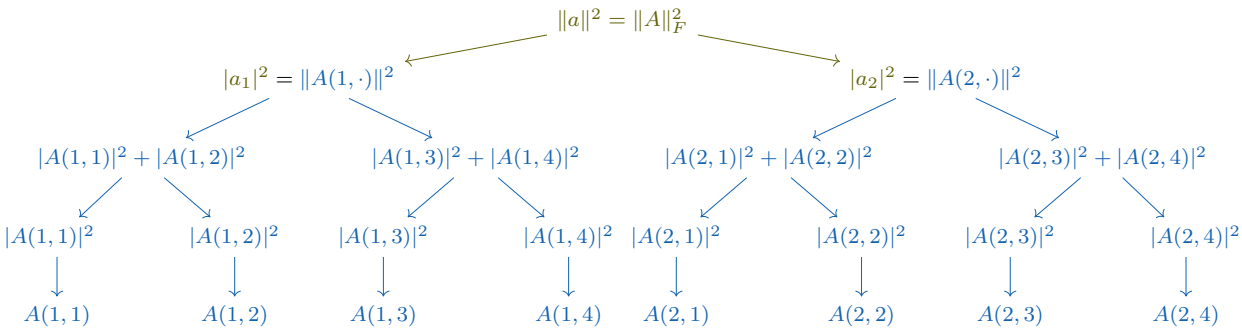

Figure 4: Dynamic data structure for a matrix $A \in \mathbb{C}^{2\times 4}$ discussed in **??** part (b). We compose the data structure for a, the vector of row norms, with the data structure for A's rows.

type of input access, we can estimate the overlap much faster via a Monte Carlo method: pull one sample, s, from $|\phi\rangle$, and then compute the estimator ψ_s/ϕ_s. This estimator has expected value $\langle\phi|\psi\rangle$ and variance 1, so by averaging over a constant number of runs, we can estimate of the overlap to 0.01 error using $O(\log d)$ classical gates, assuming that the entries of ϕ and ψ are specified with $O(\log d)$ bits. The swap test achieves the same dependence on dimension as the dequantized swap test, so it does not give an exponential speedup in this setting. (A more precise analysis would reveal that a quadratic quantum speedup in error is possible, from $\mathcal{O}(1/\varepsilon^2)$ to $\mathcal{O}(1/\varepsilon)$.) This argument against exponential quantum speedup remains valid provided we want to run the quantum algorithm in a setting where we could also perform the quantum-inspired algorithm.

The general principle of the dequantized swap test extends to other QML algorithms. In the typical RAM access model, we assume only that we can query entries efficiently. In other words, we receive our input $v \in \mathbb{C}^n$ as $\mathrm{Q}(v)$ with $\mathbf{q}(v) = 1$.

Definition 4.1 (Query access)**.** For a vector $v \in \mathbb{C}^n$, we have $\mathrm{Q}(v)$, *query access* to v, if for all $i \in [n]$, we can query for $v(i)$. Let $\mathbf{q}(v)$ denote the (time) cost of such a query.

For comparison to quantum algorithms, we assume a stronger input model, sampling and query access, which supports the types of queries we need to perform the overlap estimation algorithm.

Definition 4.2 (Sampling and query access to a vector)**.** For a vector $v \in \mathbb{C}^n$, we have $\mathrm{SQ}(v)$, *sampling and query access* to v, if we can:

1. query for entries of v as in $\mathrm{Q}(v)$;
2. obtain independent samples $i \in [n]$ where the probability of sampling i is $|v(i)|^2/\|v\|^2$;
3. query for $\|v\|$.

Let $\mathbf{sq}(v)$ denote the time cost of any query.

If we only have $\mathrm{Q}(v)$, then responding to queries from $\mathrm{SQ}(v)$ (or preparing the state $|v\rangle$) requires linear-time pre-processing. When quantum algorithms use $|v\rangle$, it's sensible to give classical algorithms access to $\mathrm{SQ}(v)$, since this is $\mathrm{Q}(v)$ with access to computational basis measurements of $|v\rangle$, also known as *importance samples from* v. In fact, as far as we

know, if input data is given *classically*,[2] classical algorithms in the sampling and query model can be run whenever the corresponding algorithms in the quantum model can (**??**). For example, if input is loaded in the QRAM data structure, as commonly assumed in QML in order to satisfy state preparation assumptions [Pra14; Cil+18], then we have log-time sampling and query access to it. So, a fast classical algorithm for a problem in this classical model implies lack of quantum speedup for the problem, at least in the usual settings explored in the QML literature.

As the inner product estimation protocol suggests, $\mathrm{SQ}(v)$ is a much more powerful access model than $\mathrm{Q}(v)$. Classical algorithms can exploit the measurements of input data possible with sampling and query access to speed up linear algebra to become time-independent of the dimension. Specifically, sketching algorithms explore how to use randomness to perform a dimensionality reduction and "sketch" a large matrix A down to a constant-sized matrix normalized submatrix of A which behaves similarly to the full matrix [Woo14]. The computational basis measurements one can produce in the quantum-inspired input model allow for the efficient estimation of matrix products through Monte Carlo methods [DKM06], which can be applied iteratively to produce dequantized algorithms that achieve surprisingly similar bounds to their quantum counterparts. We explore this in our next example.

4.2 Extensibility properties

We can show that *oversampling and query access is approximately closed under arithmetic operations*. These extensibility properties together imply that, given input matrices and vectors in data structures in QRAM, we can get oversampling and query access to low-degree polynomials of the input via closure properties; in the same setting, QSVT gives block-encodings of low-degree polynomials of the input, through similar properties. The classical algorithm's runtime is only polynomially slower than the corresponding quantum algorithm (except in the ε parameter). This dequantizes QSVT.

Definition 4.3. We have *ϕ-oversampling and query access* to a vector $v \in \mathbb{C}^n$, $\mathrm{SQ}_\phi(v)$, if:

1. we can query for entries of v, $\mathrm{Q}(v)$, and;
2. we have sampling and query access to an "entry-wise upper bound" vector $\tilde{v}$, $\mathrm{SQ}(\tilde{v})$, where $\|\tilde{v}\|^2 = \phi\|v\|^2$ and $|\tilde{v}(i)| \geq |v(i)|$ for all indices $i \in [n]$.

Let $\mathbf{sq}_\phi(v)$ denote the time cost of any query.

The parameter ϕ is the classical analogue of α for block-encodings. These appear in running times of algorithms because they correspond to overhead in rejection sampling and post-selection, respectively. More specifically, we will compare what we call (sub)normalization "overhead" between the two, which is the value ϕ in the classical setting and what [GSLW19] denotes as α in the quantum setting.

Lemma 4.4. *Suppose we are given $\mathrm{SQ}_\phi(v)$ and some $\delta \in (0, 1]$. Denote $\overline{\mathbf{sq}}(v) := \phi\,\mathbf{sq}_\phi(v)\log\frac{1}{\delta}$. We can sample from $\mathcal{D}_v$ with probability $\geq 1-\delta$ in $\mathcal{O}(\overline{\mathbf{sq}}(v))$ time. We can also estimate $\|v\|$ to ν multiplicative error for $\nu \in (0, 1]$ with probability $\geq 1-\delta$ in $\mathcal{O}(\frac{1}{\nu^2}\,\overline{\mathbf{sq}}(v))$ time.*

[2] This assumption is important. When input data is quantum (say, it is coming directly from a quantum system), a classical computer has little hope of performing linear algebra on it efficiently, see for example [ACQ22; CHM21].

Proof. Consider the following rejection sampling algorithm to generate samples: sample an index i from $\tilde{v}$, and output it as the desired sample with probability $r(i) \coloneqq \frac{|v(i)|^2}{|\tilde{v}(i)|^2}$. Otherwise, restart. We can perform this: we can compute $r(i)$ in $\mathcal{O}(\mathbf{sq}_\phi(v))$ time and $r(i) \le 1$ since $\tilde{v}$ bounds v.

The probability of accepting a sample in a round is $\sum_i \mathcal{D}_{\tilde{v}}(i) r(i) = \|v\|^2/\|\tilde{v}\|^2 = \phi^{-1}$ and, conditioned on a sample being accepted, the probability of it being i is $|v(i)|^2/\|v\|^2$, so the output distribution is $\mathcal{D}_v$ as desired. So, to get a sample with $\ge 1-\delta$ probability, run rejection sampling for at most $2\phi \log \frac{1}{\delta}$ rounds.

To estimate $\|v\|^2$, notice that we know $\|\tilde{v}\|^2$, so it suffices to estimate $\|v\|^2/\|\tilde{v}\|^2$ which is ϕ^{-1}. The probability of accepting the rejection sampling routine is ϕ^{-1}, so we run $3\nu^{-2}\phi \log \frac{2}{\delta}$ rounds of it for estimating ϕ^{-1}. Let Z denote the fraction of them which end in acceptance. Then, by a Chernoff bound we have

$$\Pr[|Z - \phi^{-1}| \ge \nu\phi^{-1}] \le 2\exp\Big(-\frac{\nu^2 z \phi^{-1}}{2+\nu}\Big) \le \delta,$$

so $Z\|\tilde{v}\|^2$ is a good multiplicative approximation to $\|v\|^2$ with probability $\ge 1-\delta$. □

Lemma 4.5 (Linear combinations, Proposition 4.3 of [Tan19])**.** *Given* $\mathrm{SQ}_{\varphi_t}(v_t) \in \mathbb{C}^n$ *and* $\lambda_t \in \mathbb{C}$ *for all* $t \in [\tau]$*, we have* $\mathrm{SQ}_\phi(\sum_{t=1}^\tau \lambda_t v_t)$ *for* $\phi = \tau \frac{\sum \varphi_t \|\lambda_t v_t\|^2}{\|\sum \lambda_t v_t\|^2}$ *and* $\mathbf{sq}_\phi(\sum \lambda_t v_t) = \sum_{t=1}^\tau \mathbf{sq}(v_t)$ *(after paying* $\mathcal{O}(\sum_{t=1}^\tau \mathbf{sq}_{\varphi_t}(v_t))$ *one-time pre-processing cost to query for norms).*

Proof. Denote $u \coloneqq \sum \lambda_t v_t$. To compute $u(s)$ for some $s \in [n]$, we just need to query $v_t(s)$ for all $t \in [\tau]$, paying $\mathcal{O}(\sum \mathbf{q}(v_t))$ cost. So, it suffices to get $\mathrm{SQ}(\tilde{u})$ for an appropriate bound $\tilde{u}$. We choose

$$\tilde{u}(s) = \sqrt{\tau \textstyle\sum_{t=1}^\tau |\lambda_t \tilde{v}_t(s)|^2},$$

so that $|\tilde{u}(s)| \ge |u(s)|$ by Cauchy–Schwarz, and $\|\tilde{u}\|^2 = \tau \sum_{t=1}^\tau \|\lambda_t \tilde{v}_t\|^2 = \tau \sum_{t=1}^\tau \varphi_t \|\lambda_t v_t\|^2$, giving the desired value of ϕ.

We have $\mathrm{SQ}(\tilde{u})$: we can compute $\|\tilde{u}\|^2$ by querying for all norms $\|\tilde{v}_t\|$, compute $\tilde{u}(s)$ by querying $\tilde{v}_t(s)$ for all $t \in [\tau]$. We can sample from $\tilde{u}$ by first sampling $t \in [\tau]$ with probability $\frac{\|\lambda_t \tilde{v}_t\|^2}{\sum_\ell \|\lambda_\ell \tilde{v}_\ell\|^2}$, and then taking our sample to be $j \in [n]$ from $\tilde{v}_t$. The probability of sampling $j \in [n]$ is correct:

$$\sum_{t=1}^\tau \frac{\|\lambda_t \tilde{v}_t\|^2}{\sum_\ell \|\lambda_\ell \tilde{v}_\ell\|^2} \frac{|\tilde{v}_t(j)|^2}{\|\tilde{v}_t\|^2} = \frac{\sum_{t=1}^\tau |\lambda_t \tilde{v}_t(j)|^2}{\sum_{\ell=1}^\tau \|\lambda_\ell \tilde{v}_\ell\|^2} = \frac{|\tilde{u}(j)|^2}{\|\tilde{u}\|^2}.$$

If we pre-process by querying all the norms $\|\tilde{v}_\ell\|$ in advance, we can sample from the distribution over i's in $\mathcal{O}(1)$ time, using an alias sampling data structure for the distribution (**??**), and we can sample from $\tilde{v}_t$ using our assumed access to it, $\mathrm{SQ}_{\varphi_t}(v_t)$. □

Definition 4.6 (Oversampling and query access to a matrix)**.** For a matrix $A \in \mathbb{C}^{m\times n}$, we have $\mathrm{SQ}(A)$ if we have $\mathrm{SQ}(A(i,\cdot))$ for all $i \in [m]$ and $\mathrm{SQ}(a)$ for $a \in \mathbb{R}^m$ the vector of row norms $(a(i) \coloneqq \|A(i,\cdot)\|)$.

We have $\mathrm{SQ}_\phi(A)$ if we have $\mathrm{Q}(A)$ and $\mathrm{SQ}(\tilde{A})$ for $\tilde{A} \in \mathbb{C}^{m\times n}$ satisfying $\|\tilde{A}\|_{\mathrm{F}}^2 = \phi\|A\|_{\mathrm{F}}^2$ and $|\tilde{A}(i,j)|^2 \ge |A(i,j)|^2$ for all $(i,j) \in [m]\times[n]$.

Let $\mathbf{sq}_\phi(A)$ denote the cost of every query. We omit subscripts if $\phi = 1$.

Lemma 4.7. *Given vectors* $\mathrm{SQ}_{\varphi_u}(u) \in \mathbb{C}^m$ *and* $\mathrm{SQ}_{\varphi_v}(v) \in \mathbb{C}^n$*, we have* $\mathrm{SQ}_\phi(A)$ *for their outer product* $A := uv^\dagger$ *with* $\phi = \varphi_u\varphi_v$ *and* $\mathbf{sq}_\phi(A) = \mathbf{sq}_{\varphi_u}(u) + \mathbf{sq}_{\varphi_v}(v)$.

Proof. We can query an entry $A(i,j) = u(i)v(j)^\dagger$ by querying once from u and v. Our choice of upper bound is $\tilde{A} = \tilde{u}\tilde{v}^\dagger$. Clearly, this is an upper bound on $uv^\dagger$ and $\|\tilde{A}\|_{\mathrm{F}}^2 = \|\tilde{u}\|^2\|\tilde{v}\|^2 = \varphi_u\varphi_v\|A\|_{\mathrm{F}}^2$. We have $\mathrm{SQ}(\tilde{A})$ in the following manner: $\tilde{A}(i,\cdot) = \tilde{u}(i)\tilde{v}^\dagger$, so we have $\mathrm{SQ}(\tilde{A}(i,\cdot))$ from $\mathrm{SQ}(\tilde{v})$ after querying for $\tilde{u}(i)$, and $\tilde{a} = \|\tilde{v}\|^2\tilde{u}$, so we have $\mathrm{SQ}(\tilde{a})$ from $\mathrm{SQ}(\tilde{u})$ after querying for $\|\tilde{v}\|$. □

Using the same ideas as in Lemma 4.5, we can extend sampling and query access of input matrices to linear combinations of those matrices.

Lemma 4.8. *Given* $\mathrm{SQ}_{\varphi^{(t)}}(A^{(t)}) \in \mathbb{C}^{m\times n}$ *and* $\lambda_t \in \mathbb{C}$ *for all* $t \in [\tau]$*, we have* $\mathrm{SQ}_\phi(A) \in \mathbb{C}^{m\times n}$ *for* $A := \sum_{t=1}^\tau \lambda_t A^{(t)}$ *with* $\phi = \tau\frac{\sum_{t=1}^\tau \varphi^{(t)}\|\lambda_t A^{(t)}\|_{\mathrm{F}}^2}{\|A\|_{\mathrm{F}}^2}$ *and* $\mathbf{sq}_\phi(A) = \sum_{t=1}^\tau \mathbf{sq}_{\varphi^{(t)}}(A^{(t)})$ *(after paying* $\mathcal{O}(\sum_{t=1}^\tau \mathbf{sq}_{\varphi^{(t)}}(A^{(t)}))$ *one-time pre-processing cost).*

Lemma 4.9 (Asymmetric matrix multiplication to Frobenius norm error, [DKM06, Lemma 4])**.** *Consider* $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{m\times p}$*, and take* $S \in \mathbb{R}^{r\times m}$ *to be sampled according to* $p \in \mathbb{R}^m$ *a* ϕ*-oversampled importance sampling distribution from* X *or* Y*. Then,*

$$\mathrm{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_{\mathrm{F}}^2] \le \frac{\phi}{r}\|X\|_{\mathrm{F}}^2\|Y\|_{\mathrm{F}}^2$$

Proof. To show the first equation, we use that $\mathrm{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_{\mathrm{F}}^2]$ is a sum of variances, one for each entry (i,j), since $\mathrm{E}[X^\dagger S^\dagger SY - XY]$ is zero in every entry. Furthermore, for every entry (i,j), the matrix expression is the sum of r independent, mean-zero terms, one for each row of S:

$$[X^\dagger S^\dagger SY - XY](i,j) = \sum_{s=1}^r \Big([SX](s,i)^\dagger[SY](s,j) - \frac{1}{r}[X^\dagger Y](i,j)\Big).$$

So, we can use standard properties of variances to conclude that

$$\begin{aligned}\mathrm{E}[\|X^\dagger S^\dagger SY - X^\dagger Y\|_{\mathrm{F}}^2] &= r\cdot\mathrm{E}[\|[SX](1,\cdot)^\dagger[SY](1,\cdot) - \tfrac{1}{r}X^\dagger Y\|_{\mathrm{F}}^2] \le r\cdot\mathrm{E}[\|[SX](1,\cdot)^\dagger[SY](1,\cdot)\|_{\mathrm{F}}^2] \\ &= r\sum_{i=1}^m p(i)\frac{\|X(i,\cdot)^\dagger Y(i,\cdot)\|_{\mathrm{F}}^2}{r^2p(i)^2} = \frac{1}{r}\sum_{i=1}^m \frac{\|X(i,\cdot)\|^2\|Y(i,\cdot)\|^2}{p(i)} \le \frac{\phi}{r}\|X\|_{\mathrm{F}}^2\|Y\|_{\mathrm{F}}^2.\end{aligned}$$

□

Remark 4.10. This implies that, given $\mathrm{SQ}_{\phi_1}(X)$ and $\mathrm{SQ}_{\phi_2}(Y)$, we can get $\mathrm{SQ}_\phi(M)$ for M a sufficiently good approximation to $X^\dagger Y$, with $\phi \le \phi_1\phi_2\frac{\|X\|_{\mathrm{F}}^2\|Y\|_{\mathrm{F}}^2}{\|M\|_{\mathrm{F}}^2}$. This is an approximate closure property for oversampling and query access under matrix products.

Given the above types of accesses, we can compute the sketch S necessary for Lemma 4.9 by taking $p = \mathcal{D}_{\tilde{x}}$ and $q = \mathcal{D}_{\tilde{y}}$), thereby finding a desired $M := X^\dagger S^\dagger SY$. We can compute entries of M with only r queries each to X and Y, so all we need is to get $\mathrm{SQ}(\tilde{M})$ for $\tilde{M}$ the appropriate bound. We choose $|\tilde{M}(i,j)|^2 := r\sum_{\ell=1}^r |[S\tilde{X}](\ell,i)^\dagger[S\tilde{Y}](\ell,j)|^2$; showing that we have $\mathrm{SQ}(M)$ follows from the proofs of Lemmas 4.7 and 4.8, since M is simply a

linear combination of outer products of rows of $\tilde{X}$ with rows of $\tilde{Y}$. Finally, this bound has the appropriate norm. Notating the rows sampled by the sketch as $s_1, \ldots, s_r$, we have

$$\|\tilde{M}\|_{\mathrm{F}}^2 = r\sum_{\ell=1}^{r} \|[S\tilde{X}](\ell,\cdot)\|^2\|[S\tilde{Y}](\ell,\cdot)\|^2 = r\sum_{\ell=1}^{r} \frac{\|\tilde{X}(s_\ell,\cdot)\|^2\|\tilde{Y}(s_\ell,\cdot)\|^2}{r^2(\frac{\|\tilde{X}(s_\ell,\cdot)\|^2}{2\|\tilde{X}\|_{\mathrm{F}}^2} + \frac{\|\tilde{Y}(s_\ell,\cdot)\|^2}{2\|\tilde{Y}\|_{\mathrm{F}}^2})^2}$$
$$\leq \sum_{\ell=1}^{r} \frac{\|\tilde{X}(s_\ell,\cdot)\|^2\|\tilde{Y}(s_\ell,\cdot)\|^2}{r(\frac{\|\tilde{X}(s_\ell,\cdot)\|\|\tilde{Y}(s_\ell,\cdot)\|}{\|\tilde{X}\|_{\mathrm{F}}\|\tilde{Y}\|_{\mathrm{F}}})^2} = \|\tilde{X}\|_{\mathrm{F}}^2\|\tilde{Y}\|_{\mathrm{F}}^2 = \phi_1\phi_2\|X\|_{\mathrm{F}}^2\|Y\|_{\mathrm{F}}^2.$$

4.3 Applying the extensibility properties

Assuming A and b are in appropriate data structures in QRAM, we can implement a $\|A\|_{\mathrm{F}}$-block-encoding of A and prepare copies of $|b\rangle$ efficiently, so we can quantumly produce a sample from Ab in $\mathcal{O}(\frac{\|A\|_{\mathrm{F}}^2\|b\|^2}{\|Ab\|^2})$ time. We can dequantize this algorithm! Classically, under identical assumptions, we can produce a sample from a v such that $\|v - Ab\| \leq \varepsilon\|Ab\|$ in $\mathcal{O}(\frac{\|A\|_{\mathrm{F}}^4\|b\|^4}{\varepsilon^2\|Ab\|^4})$ time, only polynomially slower than quantum.

We note here that a dependence on error ε appears here where it does not in the quantum setting. However, this is not a realizable quantum speedup (except possibly for sampling tasks) since the output is a quantum state: estimating some statistic of the quantum state requires incurring a polynomial dependence on ε. For example, if the goal is to estimate $|\langle v|Ab\rangle|^2$, where v is a given vector, then this can be done with $1/\varepsilon^2$ invocations of a swap test (or $1/\varepsilon$ if one uses amplitude amplification). More generally, distinguishing a state from one ε-far in trace distance requires $\Omega(1/\varepsilon)$ additional overhead, even when given an oracle efficiently preparing that state, so estimating quantities to this sensitivity requires polynomial dependence on ε.

To see this, we first consider a simple case: where b is a constant-sized vector, so $n = \mathcal{O}(1)$. Then we simply wish to sample from a linear combination of columns of A, since $Ab = \sum_{t=1}^{n} b(t)A(\cdot,t)$. If A is in the QRAM data structure (i.e. storing $A^\dagger$ in Fig. 4), then this means its columns are in the vector QRAM data structures (Fig. 3), so classically we have sampling and query access to the columns of A, $\mathrm{SQ}(A(\cdot,t))$ for all $t \in [n]$. This implies we have sampling and query access to Ab, up to some overhead.

Given access to a constant number of vectors $\mathrm{SQ}(A(\cdot,1)), \ldots, \mathrm{SQ}(A(\cdot,n))$, we have access to linear combinations $\mathrm{SQ}_\phi(Ab)$ with $\phi = n\frac{\sum_{t=1}^{n}|b(t)|^2\|A(\cdot,t)\|^2}{\|Ab\|^2} \leq \frac{n\|A\|_{\mathrm{F}}^2\|b\|^2}{\|Ab\|^2}$ and $\mathbf{sq}_\phi(Ab) = \mathcal{O}(n)$ (Lemma 4.5; the inequality follows from Cauchy-Schwarz). Finally, from $\mathrm{SQ}_\phi(Ab)$ we can perform approximate versions of all the queries of $\mathrm{SQ}(Ab)$ with a factor ϕ of overhead (Lemma 4.4). This is possible with rejection sampling: given $\mathrm{SQ}_\phi(v)$, pull a sample i from $\tilde{v}$; accept it with probability $|v(i)|^2/|\tilde{v}(i)|^2$, and restart otherwise; the output will be a sample from v. In particular, we can sample from Ab in $\phi n = \mathcal{O}(n^2\frac{\|A\|_{\mathrm{F}}^2\|b\|^2}{\|Ab\|^2})$ time in expectation, which is good when $n = \mathcal{O}(1)$.

Now, consider when n is too large to iterate over in our linear combination of vectors. In this setting, we can use the *approximate matrix product* property of importance sampling to reduce the number of vectors under consideration. Consider pulling a sample $s \in [n]$ where we sample i with probability $p(i)$. Then $\frac{1}{p(s)}b(s)A(\cdot,s)$, a rescaled random column of A, has expectation $\sum_i b(i)A(\cdot,i) = Ab$. If the sampling distribution is chosen to be $p(i) = \frac{|b(i)|^2}{\|b\|^2}$, an importance sample from $\mathrm{SQ}(b)$, then a variance computation shows

that the average of $\tau = \Theta(\frac{\|A\|_F^2}{\varepsilon^2\|A\|^2})$ copies of this random vector is $\varepsilon\|A\|\|b\|$-close to Ab with probability ≥ 0.9 (Lemma 4.9). This average, which we denote v, is now a linear combination of only τ columns of A, each of which we have sampling and query access. So, we can use the closure properties mentioned before to get $\mathrm{SQ}_\phi(v)$ for $\phi = \mathcal{O}(\frac{\|A\|_F^2\|b\|^2}{\|v\|^2})$ and $\mathbf{sq}_\phi(v) = \mathcal{O}(\tau)$, and a sample from v in $\phi\tau = \mathcal{O}(\frac{\|A\|_F^4\|b\|^2}{\varepsilon^2\|A\|^2\|v\|^2})$ time in expectation. Rescaling ε by a factor of $\frac{\|Ab\|}{\|A\|\|b\|}$ gives the result stated above.

References

[ACQ22] Dorit Aharonov, Jordan Cotler, and Xiao-Liang Qi. "Quantum algorithmic measurement". In: *Nature Communications* 13.1 (Feb. 2022). DOI: 10.1038/s41467-021-27922-0. arXiv: 2101.04634 [quant-ph] (page 5).

[BCWW01] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. "Quantum fingerprinting". In: *Physical Review Letters* 87.16 (Sept. 2001), p. 167902. DOI: 10.1103/physrevlett.87.167902. arXiv: quant-ph/0102001 [quant-ph] (pages 1, 2).

[CGLLTW22] Nai-Hui Chia, András Pal Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. "Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning". In: *Journal of the ACM* 69.5 (Oct. 2022), pp. 1–72. DOI: 10.1145/3549524. arXiv: 1910.06151 [cs.DS] (page 1).

[CHM21] Jordan Cotler, Hsin-Yuan Huang, and Jarrod R. McClean. *Revisiting dequantization and quantum advantage in learning tasks*. 2021. DOI: 10.48550/ARXIV.2112.00811. arXiv: 2112.00811 [quant-ph] (page 5).

[Cil+18] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. "Quantum machine learning: a classical perspective". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (Jan. 2018), p. 20170551. DOI: 10.1098/rspa.2017.0551. arXiv: 1707.08561 (page 5).

[DKM06] P. Drineas, R. Kannan, and M. Mahoney. "Fast Monte Carlo algorithms for matrices I: approximating matrix multiplication". In: *SIAM Journal on Computing* 36.1 (Jan. 2006), pp. 132–157. DOI: 10.1137/s0097539704442684 (pages 5, 7).

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. "Quantum random access memory". In: *Physical Review Letters* 100.16 (2008), p. 160501. DOI: 10.1103/PhysRevLett.100.160501. arXiv: 0708.1879 (page 3).

[GSLW19] András Gilyén, Yuan Su, Guang Hao Low, and Nathan Wiebe. "Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics". In: *Proceedings of the 51st ACM Symposium on the Theory of Computing (STOC)*. ACM, June 2019, pp. 193–204. DOI: 10.1145/3313276.3316366. arXiv: 1806.01838 (pages 3, 5).

[HHL09] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". In: *Physical Review Letters* 103 (15 Oct. 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502 (page 1).

[JR23] Samuel Jaques and Arthur G. Rattew. "QRAM: A survey and critique". In: (May 17, 2023). DOI: 10.48550/ARXIV.2305.10310. arXiv: 2305.10310 [quant-ph] (page 3).

[JW06] Dominik Janzing and Pawel Wocjan. *Estimating diagonal entries of powers of sparse symmetric matrices is BQP-complete*. June 27, 2006. arXiv: quant-ph/0606229 [quant-ph] (page 1).

[KP17] Iordanis Kerenidis and Anupam Prakash. "Quantum recommendation systems". In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*. 2017, 49:1–49:21. DOI: 10.4230/LIPIcs.ITCS.2017.49. arXiv: 1603.08675 (page 1).

[LGZ16] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. "Quantum algorithms for topological and geometric analysis of data". In: *Nature Communications* 7.1 (Jan. 2016), p. 10138. DOI: 10.1038/ncomms10138. arXiv: 1408.3106 (page 1).

[LMR13] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. "Quantum algorithms for supervised and unsupervised machine learning". In: *arXiv* (2013). arXiv: 1307.0411 [quant-ph] (page 2).

[MRTC21] John M. Martyn, Zane M. Rossi, Andrew K. Tan, and Isaac L. Chuang. "Grand unification of quantum algorithms". In: *PRX Quantum* 2 (4 Dec. 2021), p. 040203. DOI: 10.1103/PRXQuantum.2.040203 (page 1).

[Pra14] Anupam Prakash. "Quantum algorithms for linear algebra and machine learning". PhD thesis. University of California at Berkeley, 2014. URL: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-211.pdf (pages 3, 5).

[Tan19] Ewin Tang. "A quantum-inspired classical algorithm for recommendation systems". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing - STOC 2019*. ACM Press, 2019, pp. 217–228. DOI: 10.1145/3313276.3316310. arXiv: 1807.04271 [cs.IR] (page 6).

[Woo14] David P. Woodruff. "Sketching as a tool for numerical linear algebra". In: *Foundations and Trends® in Theoretical Computer Science* 10.1–2 (2014), pp. 1–157. ISSN: 1551-305X. DOI: 10.1561/0400000060 (page 5).