An overview of quantum-inspired classical sampling

Ewin Tang

This is an adaptation of a talk I gave at Microsoft Research in November 2018.

I exposit the ℓ^2 sampling techniques used in my recommendation systems work and its follow-ups in dequantized machine learning:

- Tang – *A quantum-inspired algorithm for recommendation systems*
- Tang – *Quantum-inspired classical algorithms for principal component analysis and supervised clustering*;
- Gilyén, Lloyd, Tang – *Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension*;
- Chia, Lin, Wang – *Quantum-inspired sublinear classical algorithms for solving low-rank linear systems.*

The core ideas used are super simple. This goal of this blog post is to break down these ideas into intuition relevant for quantum researchers and create more understanding of this machine learning paradigm.

Contents

Notation is defined in the Glossary.

The intended audience is researchers comfortable with probability and linear algebra (SVD, in particular). Basic quantum knowledge helps with intuition, but is not essential: everything from The model onward is purely classical. The appendix is optional and explains the dequantized techniques in more detail.

An introduction to dequantization

Motivation

The best, most sought-after quantum algorithms are those that take in raw, classical input and give some classical output. For example, Shor's algorithm for factoring takes this form. These *classical-to-classical* algorithms (a term I invented for this post) have the best chance to be efficiently implemented in practice: all you need is a scalable quantum computer. (It's just that easy!)

Nevertheless, many quantum algorithms aren't so nice. Most well-known QML algorithms convert input quantum states to a desired output state or value. Thus, they do not provide a riterations in expectation. Proposition. Foroutine to get necessary copies of these input states (a *state preparation* routine) and a strategy to extract information from an output state. Both are essential to making the algorithm useful.

An example of an algorithm that is not classical-to-classical is the *swap test.* If we have many copies of the quantum states $|a\rangle, |b\rangle \in \mathbb{C}^n$, then the swap test $\mathcal{S}$ estimates their inner product in time polylogarithmic in dimension. While this routine seems much faster than naively computing $\sum_{i=1}^{n} \bar{a}_i b_i$ classically, we can only run this algorithm if we know how to prepare the states $|a\rangle$ and $|b\rangle$. It may well be the case that state preparation is too expensive for input vectors, making the quantum algorithm as slow as the classical algorithm. This illustrates the format and failings of most QML algorithms.

You might then ask: can we fill in the missing routines in QML algorithms to get a classical-to-classical algorithm that's provably fast and useful? This is an open research problem: see Scott Aaronson's piece on QML[1]. We have a variety of partial results towards the affirmative, but as far as I know, they don't answer the question unless you're loose with your definitions of at least one of "classical", "provably fast", or "useful". So let's settle for a simpler question.

How can we compare the speed of quantum algorithms with quantum input and quantum output to classical algorithms with classical input and classical output? Quantum machine learning algorithms can be exponentially faster than the best standard classical algorithms for similar tasks, but this comparison is unfair because the quantum algorithms get outside help through input state preparation. We want a classical model that helps its algorithms stand a chance

[1] Scott Aaronson. *Read the fine print.* Nature Physics 11.4, 2015. Link

against quantum algorithms, while still ensuring that they can be run in nearly all circumstances one would run the quantum algorithm. The answer I propose: **compare quantum algorithms with quantum state preparation to classical algorithms with *sample and query access* to input.**

The model

Before we proceed with definitions, we'll establish some conventions. First, we generally consider our input as being some vector in $\mathbb{C}^n$ or $\mathbb{R}^n$, subject to an access model to be described. Second, we'll only concern ourselves with an algorithm's *query complexity*, the number of accesses to the input. Our algorithms will have query complexity independent of input dimensions and polynomial in other parameters. If we assume that each access costs (say) $O(1)$ or $O(\log n)$, the time complexity is still polylogarithmic in input dimension and at most polynomially worse in other parameters.

Now, we define query access to input; we can get query access simply by having the input in RAM.

Definition. We have *query access* to $x \in \mathbb{C}^n$ (denoted $\mathrm{Q}(x)$) if, given $i \in [n]$, we can efficiently compute x_i.

If we have x stored normally as an array in our classical computer's memory, we have $\mathrm{Q}(x)$ because finding the ith entry of x can be done with the code `x[i]`. This notion of access can represent more than just memory: we can also have $\mathrm{Q}(x)$ if x is *implicitly* described. For example, consider x the vector of squares: $x_i = i^2$ for all i. We can have access to x without writing x in memory. This will be important for the algorithms to come.

Definition. We have *sample and query access* to $x \in \mathbb{C}^n$ (denoted $\mathrm{SQ}(x)$) if we have query access to x; can produce independent random samples $i \in [n]$ where we sample i with probability $|x_i|^2/\|x\|^2$; and can query for $\|x\|$.

Sampling and query access to x will be our classical analogue to assuming quantum state preparation of copies of $|x\rangle$. This should make some intuitive sense: our classical analogue $\mathrm{SQ}(x)$ has the standard assumption of query access to input, along with samples, which are essentially measurements of $|x\rangle$ in the computational basis. Knowledge of $\|x\|$ is for normalization issues, and is often assumed for quantum algorithms as well (though for both classical and quantum algorithms, often approximate knowledge suffices).

Example. Like query access, we can get efficient sample and query access from an explicit memory structure. To get $\mathrm{SQ}(x)$ for a bit vector $x \in \{0,1\}^n$, store the number of nonzero entries z and a sorted array of the 1-indices D. For example, we could store $x = [1\ 1\ 0\ 0\ 1\ 0\ 0\ 0]$ as

$$z, D = 3, \{1, 2, 5\}$$

Then we can find x_i by checking if $i \in D$, we can sample from x by picking an index from D uniformly at random, and we know $\|x\|$, since it's just $\sqrt{z}$. This generalizes to an efficient $O(\log n)$ binary search tree data structure for $\mathrm{SQ}(x)$ for any $x \in \mathbb{C}^n$.

We can also define sample and query access to matrices as just sample and query access to vectors "in" the matrix.

Definition. For $A \in \mathbb{C}^{m\times n}$, $\mathrm{SQ}(A)$ is defined as $\mathrm{SQ}(A_i)$ for A_i the rows of A, along with $\mathrm{SQ}(\tilde{A})$ for $\tilde{A}$ the vector of row norms (so $\tilde{A}_i = \|A_i\|$).

By replacing quantum states with these classical analogues, we form a model based on sample and query access which we codify with the informal definition of "dequantization".

Definition. Let $\mathcal{A}$ be a quantum algorithm with input $|\phi_1\rangle, \dots, |\phi_C\rangle$ and output either a state $|\psi\rangle$ or a value λ. We say we *dequantize* $\mathcal{A}$ if we describe a classical algorithm that, given $\mathrm{SQ}(\phi_1), \dots, \mathrm{SQ}(\phi_C)$, can evaluate queries to $\mathrm{SQ}(\psi)$ or output λ, with similar guarantees to $\mathcal{A}$ and query complexity $\mathrm{poly}(C)$.

That is, given sample and query access to the inputs, we can output sample and query access to a desired vector or a desired value, with at most polynomially larger query complexity.

We justify why this model is a reasonable point of comparison two sections from now, in Implications. Next, though, we will jump into how to build these dequantized protocols.

Quantum for the quantum-less

So far, all dequantized results revolve around three dequantized protocols that we piece together into more useful tasks. In query complexity independent of m and n, we can perform the following:

1. (Inner Product) For $x, y \in \mathbb{C}^n$, given $\mathrm{SQ}(x)$ and $\mathrm{Q}(y)$, we can estimate $\langle x, y\rangle$ to $\|x\|\|y\|\varepsilon$ error with probability $\geq 1-\delta$;
2. (Thin Matrix-Vector) For $V \in \mathbb{C}^{n\times k}, w \in \mathbb{C}^k$, given $\mathrm{SQ}(V^\dagger)$ and $\mathrm{Q}(w)$, we can simulate $\mathrm{SQ}(Vw)$ with $\mathrm{poly}(k)$ queries;
3. (Low-rank Approximation) For $A \in \mathbb{C}^{m\times n}$, given $\mathrm{SQ}(A)$ and some threshold k, we can output a description of a low-rank approximation of A with $\mathrm{poly}(k)$ queries.

   Specifically, our output is $\mathrm{SQ}(S, \hat{U}, \hat{\Sigma})$ for $S \in \mathbb{C}^{\ell\times n}$, $\hat{U} \in \mathbb{C}^{\ell\times k}$, and $\hat{\Sigma} \in \mathbb{C}^{k\times k}$ ($\ell = \mathrm{poly}(k, \frac{1}{\varepsilon})$), and this implicitly describes the low-rank approximation to A, $D := A(S^\dagger \hat{U}\Sigma^{-1})(S^\dagger\hat{U}\Sigma^{-1})^\dagger$ (notice rank $D \leq k$).

   This matrix satisfies the following low-rank guarantee with probability $\geq 1-\delta$: for $\sigma := \sqrt{2/k}\|A\|_F$, and $A_\sigma := \sum_{\sigma_i \geq \sigma} \sigma_i u_i v_i^\dagger$ (using A's SVD),

$\|A - D\|_F^2 \leq \|A - A_\sigma\|_F^2 + \varepsilon^2 \|A\|_F^2$.

This guarantee is non-standard: instead of A_k, we use A_σ. This makes our promise weaker, since it is useless if A has no large singular values.

For intuition, it's helpful to think of D as A multiplied with a "projector" $(S^\dagger \hat{U} \Sigma^{-1})(S^\dagger \hat{U} \Sigma^{-1})^\dagger$ that projects the rows of A onto the columns of $S^\dagger \hat{U} \Sigma^{-1}$, where these columns are "singular vectors" (approximately orthonormal, and with corresponding "singular values" $\hat{\sigma}_1, \ldots, \hat{\sigma}_k$ that are encoded in the diagonal matrix $\hat{\Sigma}$).

The first two protocols are dequantized swap tests and the third is essentially a dequantized variant of phase estimation seen in quantum recommendation systems[2].

Now, we describe how these techniques are used to get the results for recommendation systems, supervised clustering, and low-rank matrix inversion. We defer the important details of models and error analyses to Implications, instead focusing on the algorithms themselves and how they use dequantized protocols.

Supervised clustering

We want to find the distance from a point $u \in \mathbb{R}^n$ to the centroid (average) of a cluster of points $v_1, \ldots, v_{m-1} \in \mathbb{R}^n$. If we assume sample and query access to the data points, computing $\|u - \frac{1}{n}(v_1 + \cdots + v_{m-1})\|$ reduces to solving the following:

Problem. For $M \in \mathbb{R}^{m \times n}, w \in \mathbb{R}^n$, and $\mathrm{SQ}(M, w)$, approximate $(Mw)^T (Mw)$ to additive ε error.

Algorithm. We can write $(Mw)^T Mw$ as the inner product of an order three tensor; through basic tensor arithmetic, it is equal to $\langle u, v \rangle$, where $u, v \in \mathbb{R}^{m \times n \times n}$ are

$$\begin{aligned} u &= \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} M_{ij} \|M^{(k)}\| e_{i,j,k} \text{ and} \\ v &= \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{w_j w_k M_{ik}}{\|M^{(k)}\|} e_{i,j,k}. \end{aligned}$$

Applying the algorithm for inner product (1) gives the desired approximation with $O(\|w\|^2 \|M\|_F^2 \frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ samples and queries.

Recommendation systems

We want to randomly sample a product $j \in [n]$ that is a good recommendation for a particular user $i \in [m]$, given incomplete data on user-product preferences. If we

[2] Iordanis Kerenidis, Anupam Prakash. *Quantum recommendation systems.* arXiv:1603.08675.

store this data in a matrix $A \in \mathbb{R}^{m \times n}$ with sampling and query access, in the right model, finding good recommendations reduces to:

Problem. For a matrix $A \in \mathbb{R}^{m \times n}$ along with a row $i \in [m]$, given $\mathrm{SQ}(A)$, approximately sample from D_i where D is a sufficiently good low-rank approximation of A.

Remark. This task is essentially a variant of PCA, since a low-rank decomposition is dimensionality reduction of the matrix, viewed as a set of row vectors. This is the "dequantized PCA" I refer to in other work[3].

Algorithm. Apply (3) to get $\mathrm{SQ}(S, \hat{U}, \hat{\Sigma})$ for a low-rank approximation $D = AS^T\hat{U}\hat{\Sigma}^{-1}(\hat{\Sigma}^{-1})^T\hat{U}^TS$. It turns out that this low-rank approximation is good enough to get good recommendations. So it suffices to sample from $D_i = A_iS^TMS$, where $A_i \in \mathbb{R}^{1 \times n}, S \in \mathbb{R}^{\ell \times n}, M = \hat{U}\hat{\Sigma}^{-1}(\hat{\Sigma}^{-1})^T\hat{U}^T \in \mathbb{R}^{\ell \times \ell}$ with $\ell = \mathrm{poly}(k)$.

$$\begin{bmatrix} \cdots & A_i & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ S^T \\ \vdots \end{bmatrix} \begin{bmatrix} & M & \end{bmatrix} \begin{bmatrix} \cdots & S & \cdots \end{bmatrix}$$

Approximate A_iS^T to ℓ^2 norm using k inner product protocols (1). Next, compute A_iS^TM with naive matrix-vector multiplication. Finally, sample from $A_iS^T\hat{U}\hat{U}^TS$, which is a thin matrix-vector product (2).

An aside. This gives an exponential speedup over previous classical results from 15-20 years ago[4]. The story here is quite odd. From what I can tell, researchers at the time knew the important (read: hard) part of the algorithm, how to compute low-rank approximations fast, but didn't notice that the resulting knowledge of S and $\hat{U}$ could be used to sample the desired recommendations in sublinear time, which I think is much easier to understand. This gave me anxiety during research, since I figured there was no way this would have been overlooked. I'm glad these fears were unfounded; it's cool that this quantum perspective made this step natural and obvious!

Low-rank matrix inversion

The goal here is to mimic a quantum algorithm that can solve systems of equations $Ax = b$ for A low-rank. The dequantized version of this is:

Problem. For a low-rank matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x \in \mathbb{R}^n$, given $\mathrm{SQ}(A), \mathrm{SQ}(x)$, (approximately) respond to requests for $\mathrm{SQ}(A^+x)$, where A^+ is the pseudoinverse of A.

[3] Ewin Tang. *Quantum-inspired classical algorithms for principal component analysis and supervised clustering.* arXiv:1811.00414, 2018.

[4] Petros Drineas, Iordanis Kerenidis, Prabhakar Raghavan. *Competitive recommendation systems.* STOC, 2002. Link.

Algorithm. Use the low-rank approximation protocol (3) to get $\mathrm{SQ}(S, \hat{U}, \hat{\Sigma})$. From applying the matrix-vector protocol (2), we have $\mathrm{SQ}(\hat{V})$, where $\hat{V} := S^T \hat{U} \hat{\Sigma}^{-1}$; with some analysis we can show that the columns of $\hat{V}$ behave like the right singular vectors of A. Further, $\hat{\Sigma}_{ii}$ behaves like their approximate singular values. Using this information, we can approximate the vector we want to sample from:

$$A^+ x = (A^T A)^+ A^T x \approx \sum_{i=1}^{k} \frac{1}{\hat{\Sigma}_{ii}^2} \hat{v}_i \hat{v}_i^T A^T x$$

We approximate $\hat{v}_i^T A^T x$ to additive error for all i by noticing that $\hat{v}_i^T A^T x = \mathrm{Tr}(A^T x \hat{v}_i^T)$ is an inner product of the order two tensors A^T and $x\hat{v}_i^T$. Thus, we can apply (1), since being given $\mathrm{SQ}(A)$ implies $\mathrm{SQ}(A^T)$ for A^T viewed as a long vector. Finally, using (2), sample from the linear combination using these estimates and $\hat{\sigma}_i$.

Implications

We have just described examples of dequantized algorithms for the following problems:

- Recommendation systems[5][6] (this classical algorithm *exponentially improves* on the previous best!)
- PCA[7][8]
- Supervised clustering[9][10]
- Low-rank matrix inversion[11][12][13]

We address here what to take away from these results.

For quantum computing

The most important conclusion, in my opinion, is a heuristic:

[5] Ewin Tang. *A quantum-inspired algorithm for recommendation systems.* arXiv:1807.04271, 2018.

[6] Iordanis Kerenidis, Anupam Prakash. *Quantum recommendation systems.* arXiv:1603.08675.

[7] Ewin Tang. *Quantum-inspired classical algorithms for principal component analysis and supervised clustering.* arXiv:1811.00414, 2018.

[8] Seth Lloyd, Masoud Mohseni, Patrick Rebentrost. *Quantum principal component analysis.* arXiv:1307.0401, 2013.

[9] Ewin Tang. *Quantum-inspired classical algorithms for principal component analysis and supervised clustering.* arXiv:1811.00414, 2018.

[10] Seth Lloyd, Masoud Mohseni, Patrick Rebentrost. *Quantum algorithms for supervised and unsupervised machine learning.* arXiv:1307.0411, 2013.

[11] Patrick Rebentrost, Adrian Steffens, Iman Marvian, Seth Lloyd. *Quantum singular-value decomposition of nonsparse low-rank matrices.* arXiv:1607.05404 .

[12] András Gilyén, Seth Lloyd, Ewin Tang. *Quantum-inspired low-rank stochastic regression with logarithmic dependence on the dimension.* arXiv:1811.04909, 2018.

[13] Nai-Hui Chia, Han-Hsuan Lin, Chunhao Wang. *Quantum-inspired sublinear classical algorithms for solving low-rank linear systems.* arXiv:1811.04852, 2018.

Heuristic 1. Linear algebra problems in low-dimensional spaces (constant, say, or polylogarithmic) likely can be dequantized.

The intuition for this heuristic is that, if your problem operates in a subspace of such low dimension, the main challenge is "finding" this subspace and rotating to it. Then, we can think about our problem as lying in $\mathbb{C}^d$ where d is small, and can solve it with a simple polynomial-time (in d) algorithm. Finding the subspace is an unordered search problem if you squint, so can't be sped up much by exploiting quantum.

Remark. There are high-dimensional problems that cannot be dequantized; for example, given $\mathrm{SQ}(v)$, it takes $\Omega(n)$ queries to approximately sample from Hv, where H is the Hadamard matrix (this is the Fourier Sampling problem[14]).

Why do we care about dequantizing algorithms? As the name suggests, I argue that this is a reasonable classical analogue to quantum machine learning algorithms.

Heuristic 2. For machine learning problems, SQ assumptions are more reasonable than state preparation assumptions.

That is, the practical task of preparing quantum states is probably always harder than the practical task of preparing sample and query access. Practically, this makes sense, since for state preparation we need, well, quantum computers.

Even assuming the existence of a practical quantum computer, there is evidence that state preparation assumptions are still harder to satisfy than sample and query access, up to polynomial slowdown. For example, preparing a generic quantum state $|v\rangle$ corresponding to an input vector v takes $\Omega(\sqrt{n})$ quantum queries to v in general, while responding to $\mathrm{SQ}(v)$ accesses takes $\Theta(n)$ classical queries. Because dequantized algorithms are polynomial in $\log n$, this means that getting SQ access to a generic vector is much more expensive than running the algorithm.

Of course, we can also consider special classes of vectors where quantum state preparation is easier, but generally SQ access gets proportionally faster as well. For example, we can quickly prepare vectors where all entries have roughly equal magnitude (think vectors whose entries are either $+1$ or -1), but correspondingly, we can compute SQ accesses to such vectors similarly quickly.

On the classical side, the assumption of SQ access is on par with other typical assumptions to make machine learning algorithms sublinear:

- There is a classical dynamic data structure that supports SQ access, fast updates, and sparsity in log time.
- Given an input vector as a list of nonzero entries, sampling from it takes time linear in sparsity.
- k independent samples can be prepared with one pass through the data in $O(k)$ space.

[14] Scott Aaronson and Lijie Chen. *Complexity-theoretic foundations of quantum supremacy experiments.* arXiv:1612.05903, 2016.

To summarize these heuristics: quantum machine learning for *low-dimensional datasets* will probably never get speedups as significant as, say, Shor's algorithm, even in best-case scenarios. Unfortunately, QML for low-dimensional problems were the most practical algorithms in the literature, so with this research it's unclear what the state of the field is today.

The story might not be over, though. We know that quantum computers can "efficiently solve" high-dimensional linear algebra problems[15]; however, this assumes that we have some way to evolve a quantum system precisely according to input data, a much harder problem than the linear algebra itself. Nevertheless, I hold out hope that this result can be applied to achieve exponential speedups in machine learning or elsewhere.

For classical computing

I am cautiously optimistic about the implications of this work for classical computing. The major advantage of dequantized algorithms is sheer speed (asymptotically, at least). However, the issues listed below prevent dequantized algorithms from being strict improvements over current algorithms.

- Gaining SQ access to input typically requires preliminary data processing or the use of a data structure. This means that dequantized algorithms can't be plugged into existing systems without large amounts of computation.
- SQ access to output might not always be useful or practical.
- Current dequantized algorithms have large error compared to standard techniques.
- Current algorithms have large theoretical exponents, so right now we don't know whether they run quickly in practice. I expect we can cut down these exponents greatly.

If I had to guess, the best chance for success in dequantized techniques remains recommendation systems, since speed matters significantly in that context. I view the other algorithms as significantly less likely to see use in practice, though probably more likely than their corresponding quantum algorithms.

Regardless, these works fit nicely into the classical literature: dequantized quantum machine learning is just a nicely modular, quantum-inspired form of randomized numerical linear algebra.

[15] Aram W. Harrow, Avinatan Hassidim, Seth Lloyd. *Quantum algorithm for solving linear systems of equations.* arXiv:0811.3171, 2008.

Appendix: More details

As a reminder, here are the three techniques:

1. Inner Product
2. Thin Matrix-Vector
3. Low-rank Approximation

Below, we explain (1) and (2) fully, and give a rough sketch of (3).

1. Estimating inner products

First, we give a basic way of estimating the mean of an arbitrary distribution with finite variance.

Fact. For $\{X_{i,j}\}$ i.i.d random variables with mean μ and variance σ^2, let

$$Y := \underset{j \in [6 \log 1/\delta]}{\text{median}} \ \underset{i \in [6/\varepsilon^2]}{\text{mean}} \ X_{i,j}$$

Then $|Y - \mu| \leq \varepsilon\sigma$ with probability $\geq 1 - \delta$, using only $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ copies of X.

Proof sketch. The proof follows from two facts: first, the median of $C_1, \dots, C_n$ is at least λ precisely when at least half of the C_i are at least λ; second, Chebyshev's inequality (applied to the mean).

Estimating the inner product is just a basic corollary of this estimator.

Proposition. For $x, y \in \mathbb{C}^n$, given $\mathrm{SQ}(x)$ and $\mathrm{Q}(y)$, we can estimate $\langle x, y \rangle$ to $\varepsilon \|x\| \|y\|$ error with probability $\geq 1 - \delta$ with query complexity $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$.

Proof. Sample s from v and let $Z = x_s v_s \frac{\|v\|^2}{|v_s|^2}$. Apply the Fact with $X_{i,j}$ being independent copies of Z.

2. Thin matrix-vector product with rejection sampling

We first go over rejection sampling, a naive way to efficiently generate samples from a specified distribution from samples from another distribution.

Input: samples from distribution P
Output: samples from distribution Q

1. Pull a sample s from P;
2. Compute $r_s = \frac{Q(s)}{MP(s)}$ for some constant M;
3. Output s with probability r_s and restart otherwise.

Fact. If $r_i \leq 1$ for all i, then the above procedure is well-defined and outputs a sample from Q in M iterations in expectation.

Proposition. For $V \in \mathbb{R}^{n\times k}$ and $w \in \mathbb{R}^k$, given $\mathrm{SQ}(V)$ and $\mathrm{Q}(w)$, we can simulate $\mathrm{SQ}(Vw)$ with expected query complexity $O(k^2 C(V,w))$, where

$$C(V,w) := \frac{\sum_{i=1}^k \|w_i V^{(i)}\|^2}{\|Vw\|^2}.$$

We can compute entries $(Vw)_i$ with $O(k)$ queries.
We can sample using rejection sampling:

- P is the distribution formed by sampling from $V^{(j)}$ with probability proportional to $\|w_j V^{(j)}\|^2$;
- Q is the target Vw.

$$r_i = \frac{(Vw)_i^2}{k\sum_{j=1}^k (w_j V_{ij})^2} = \frac{Q(i)}{kC(V,w)P(i)}$$

Notice that we can compute these r_i's (in fact, despite that we cannot compute probabilities from the target distribution), and that the rejection sampling guarantee is satisfied (via Cauchy-Schwarz).

The probability of success is $\frac{\|Vw\|^2}{k\sum_{i=1}^k \|w_i V^{(i)}\|^2}$. Thus, to estimate the norm of Vw, it suffices to estimate the probability of success of this rejection sampling process. We can view this as estimating the heads probability of a biased coin, where the coin is heads if rejection sampling succeeds and tails otherwise. Through a Chernoff bound, we see that the average of $O(kC(V,w)\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ "coin flips" is in $[(1-\varepsilon)\|Vw\|, (1+\varepsilon)\|Vw\|]$ with probability $\geq 1-\delta$, where each coin flip costs k queries and samples.

3. Low-rank approximation, briefly

Proposition. For $A \in \mathbb{C}^{m\times n}$, given $\mathrm{SQ}(A)$ and some threshold k, we can output a description of a low-rank approximation of A.

Specifically, our output is $\mathrm{SQ}(S, \hat{U})$ for $S \in \mathbb{C}^{\ell\times n}$, $\hat{U} \in \mathbb{C}^{\ell\times k}$ ($\ell = \mathrm{poly}(k, \frac{1}{\varepsilon})$), and this implicitly describes the low-rank approximation to A, $D := A(S^\dagger \hat{U})(S^\dagger \hat{U})^\dagger$ (notice rank $D \leq k$).

This matrix satisfies the following low-rank guarantee with probability $\geq 1-\delta$: for $\sigma := \sqrt{2/k}\|A\|_F$, and $A_\sigma := \sum_{\sigma_i \geq \sigma} \sigma_i u_i v_i^\dagger$ (using SVD),

$$\|A - D\|_F^2 \leq \|A - A_\sigma\|_F^2 + \varepsilon^2 \|A\|_F^2.$$

This algorithm comes from the 1998 paper of Frieze, Kannan, and Vempala[16]. See the recent survey[17] by Kannan and Vempala for a survey of these techniques, and see Woodruff's textbook[18] for a discussion of more general techniques. The form I state above is a simple variant that I discuss in my recommendation systems paper[19].

It would be nice to find an exposition that made this result look quantum-ic, but I haven't figured out a way to do this yet, so instead I'll exposit following the original reference.

We now sketch the algorithm along with intuition: it's most useful to consider the low-rank approximation task as one of finding large approximate singular vectors. First, sample ℓ rows of A according to ℓ^2 norm, and consider the matrix $S \in \mathbb{C}^{\ell \times n}$ of these rows, all renormalized to have the same length. This is the S that we output. One can show that $\|S^T S - A^T A\|_F \leq \varepsilon \|A\|_F^2$ with good probability, which implies that the large right singular vectors of S approximate the large right singular vectors of A.

Next, we can perform the same process to S^T: sample rows of S^T and get a normalized submatrix $W \in \mathbb{R}^{\ell \times \ell}$ such that $\|WW^T - SS^T\|_F \leq \varepsilon \|A\|_F^2$. Since W is a constant-sized matrix, we can compute $\hat{U}$ and $\hat{\Sigma}$, the large left singular vectors and values of W, which approximate the large left singular vectors and values of S. Then, $S^T \hat{U} \hat{\Sigma}^{-1}$ translates these large left singular vectors to their corresponding right singular vectors and rescales them accordingly, giving the approximate singular vectors of A as desired.

[16] Alan Frieze, Ravindran Kannan, Santosh Vempala. *Fast monte-carlo algorithms for finding low-rank approximations. Journal of the ACM*, vol. 51, no. 6, 2004. Link.

[17] Ravindran Kannan and Santosh Vempala. *Randomized algorithms in numerical linear algebra.* Acta Numerica 26, 2017. Link.

[18] David P. Woodruff. *Sketching as a tool for numerical linear algebra.* Foundations and Trends in Theoretical Computer Science 10.1–2, 2014. Link.

[19] Ewin Tang. *A quantum-inspired algorithm for recommendation systems.* arXiv:1807.04271, 2018.

Glossary

For natural numbers m, n, vector $v \in \mathbb{C}^n$ and $A \in \mathbb{C}^{m\times n}$:

$[n]$ denotes $\{1, 2, \dots, n\}$; $O(\cdot)$ and $\Omega(\cdot)$ is big O notation; A_i and $A^{(j)}$ denotes the ith row of A and the jth column of A; $\|v\|$ denotes the ℓ^2 norm of v, $\sqrt{\|v_1\|^2 + \cdots + \|v_n\|^2}$;

$|\psi\rangle$ is bra-ket notation: kets are column vectors $|\psi\rangle \in \mathbb{C}^{n\times 1}$, bras are row vectors $\langle\psi| := (|\psi\rangle)^\dagger$, standard basis vectors are denoted $|1\rangle, \dots, |n\rangle$, and the tensor product of $|\alpha\rangle$ and $|\beta\rangle$ is denoted $|\alpha\rangle|\beta\rangle$. Of course, these are all really quantum states, but that's only relevant for quantum algorithms: for my purposes, I use $|\phi\rangle$ and ϕ interchangeably to refer to vectors. (I ignore normalization, but those issues can be dealt with.)

The singular value decomposition (SVD) of $A \in \mathbb{C}^{m\times n}$ is a decomposition $A = U\Sigma V^\dagger$, where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal. In other words, for u_i and v_i the columns of U and V, respectively, and σ_i the diagonal entries of Σ, $A = \sum \sigma_i u_i v_i^\dagger$. By convention, $\sigma_1 \geq \ldots \geq \sigma_{\min m,n} \geq 0$.

Using A's SVD, we can define basic linear algebraic objects. $\|A\|_2 = \max_{v\in\mathbb{C}^n} \|Av\|/\|v\| = \sigma_1$ is the spectral norm of A. $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |A_{ij}|^2} = \sqrt{\sigma_1^2 + \cdots + \sigma_{\min m,n}^2}$ is the Frobenius norm of A. $A_k = \sum_{i=1}^k \sigma_i u_i v_i^\dagger$ is an optimal rank k approximation to A in both spectral and Frobenius norm. $A^+ = \sum_{\sigma_i\neq 0} \frac{1}{\sigma_i} v_i u_i^\dagger$ is A's pseudoinverse.

I define $\mathrm{SQ}(v)$, $\mathrm{SQ}(A)$, and $\mathrm{Q}(v)$ in An introduction to dequantization.