A quantum-inspired classical algorithm for recommendation systems

Ewin Tang

May 22, 2018

Abstract

A recommendation system, which suggests products to users based on data about user preferences, is typically modeled as a problem of completing a matrix of small rank \( k \). We give the first recommendation system to produce a recommendation in \( O(\text{poly}(k) \text{ polylog}(m, n)) \) time, an exponential improvement on the best known recommendation systems, which run in time linear in \( m \) and \( n \). Our strategy is inspired by a quantum recommendation system by Kerenidis and Prakash: instead of reconstructing a user’s full list of preferences, which is the approach taken by previous recommendation systems, we only seek a randomized sample from said user’s preferences. This allows us to circumvent linear-time bottlenecks. As a consequence, we show that Kerenidis and Prakash’s quantum machine learning algorithm, one of the best candidates for provably exponential speedups in QML, does not in fact give an exponential speedup over classical algorithms.

1 Introduction

1.1 Recommendation Systems

The question driving the study of recommendation systems is a practical one: given sparse, incomplete data on user preferences for products, can one quickly and correctly predict which other products a user will prefer?

To put this problem on theoretically sound footing, we model this problem as follows. We represent the sentiments of \( m \) users towards \( n \) products with an \( m \times n \) matrix \( T \), where \( T_{ij} \) is large if user \( i \) likes product \( j \). We call \( T \) a preference matrix. If we can find high-value entries of \( T \), we can provide good recommendations by outputting the corresponding products. However, we are typically given only a small subsample of entries of \( T \) (say, when a user purchases a product or fills out a survey). Then, finding recommendations for user \( i \) is equivalent to finding large entries of the \( i \)th row of \( T \) given such a subsample.
Obviously, without any restrictions on what $T$ looks like, this problem is intractable. We make this problem tractable through a standard assumption on $T$; namely, that it have low numerical rank $k$ (say, constant or logarithmic in $m$ and $n$). This reflects the intuition that users tend to fall into a small number of classes based on their preferences. With this assumption, it becomes possible to infer information from subsampled data; namely, we can determine the class a user lies in, and use information about that class to give a likely-to-be-good recommendation. Roughly speaking, the field of recommendation systems aims to develop algorithms that do this to get good recommendations in this model, as first seen in [12] and refined further in [3] and [7].

Our main result is a classical recommendation system algorithm whose runtime is exponentially faster than the best-known in the literature. More specifically, given our subsampled data $A$ in a low-overhead dynamic data structure and some user $i$, we can give a random sample from a distribution defined by the $i$th row of a low-rank approximation of $A$ in $O(poly(k) \ polylog(m, n))$ time; given standard assumptions on $A$ and $T$, this sample is likely to be a good recommendation. We do this with a modified version of Frieze, Kannan, and Vempala’s algorithm for low-rank matrix approximation [8], which we prove a novel bound for. This reduces the problem to a straightforward sampling problem, which we solve using usual techniques like rejection sampling.

With the low-rank assumption that $k$ is small, this algorithm gives an exponential improvement over the current literature. Algorithms for recommendation systems typically attempt to reconstruct full rows of $T$, which must take $\Omega(n)$ time. We sidestep this bottleneck by only giving a sample of a good recommendation, instead of a complete list of recommendations. Further, we are able to do this without atypical or unnatural assumptions; in fact, our assumptions are exactly the same as those of Kerenidis and Prakash for their quantum recommendation system [10].

Finally, we note that there is also a substantial body of work on practical recommendation systems, which use a combination of theoretical methods and implementations of intuitive heuristics. Using the terminology from that setting, our algorithm is best described as an SVD-based technique. Work on the Netflix Prize is a useful reference for practical recommendation systems: see [5] for a broad overview of techniques, [11] for a high-level exposition of the SVD technique, and [4] for more technical details.

1.2 Quantum Machine Learning

Our algorithm stemmed from work analyzing Kerenidis and Prakash’s quantum recommendation system [10] in the context of proving exponential speedups for algorithms in quantum machine learning. QML spawned from Harrow, Hassidim, and Lloyd’s 2008 quantum algorithm for solving linear systems [9]. This burgeoning field has produced exciting quantum algorithms that give
hope for finding exponential speedups outside of the now-established gamut of problems related to period finding and Fourier coefficients. However, because of a number of caveats exposited by Aaronson in [1], no QML algorithm is known to have a new, provably exponential speedup over classical algorithms. Kerenidis and Prakash’s work is the first to address all of these caveats in a satisfactory way and give a complete quantum algorithm that can be compared directly to classical algorithms.

When their work was published, their algorithm was exponentially faster than the best-known classical algorithms. It was not known whether this was a provably exponential speedup. Our algorithm gives a classical algorithm that matches that of Kerenidis and Prakash’s (up to small loss in approximation), thus answering this question. While this negative result leaves open the question of exponential speedups in quantum machine learning, the author sees this as a success for the field, by giving inspiration for new classical algorithms with improved time complexity.

1.3 Algorithm Sketch

**Main Result.** There is a classical algorithm that performs just as well as the quantum algorithm in [10], only losing polynomial factors in runtime and ε factors in approximation of output.

The quantum algorithm essentially just computes low-rank matrix approximations: given an input matrix $A$, it produces a quantum state representing a row of a low-rank approximation of $A$. It can then measure this state to sample an entry of that row that is likely to have large magnitude. Kerenidis and Prakash show through specification and analysis of a recommendation system model that this is sufficient to give good recommendations. Crucially, instead of reconstructing an entire row of recommendations, like was done in previous literature, the algorithm only samples for a “high-value” recommendation; the quantum setting makes such a notion natural to consider. Indeed, a constant number of recommendations for a user is nearly invariably sufficient in practice, so this algorithm does not lose any utility.

Our recommendation system keeps the model from Kerenidis and Prakash, but replaces the quantum algorithm with a classical algorithm. Thus, our goal is to give a classical algorithm that, given an input matrix, can sample a high-value entry from a low-rank approximation of that matrix.

We do this by splitting the problem into two parts. First, we show that given a matrix $A$ in a BST data structure (one that [10] uses for their quantum algorithm), we can in fact produce a description for a matrix whose rows do produce good recommendations. This is done via an algorithm by Frieze, Kannan, and Vempala [8] that samples a submatrix whose size is independent of $m$ and $n$ to find a low-rank matrix approximation. We modify this algorithm (with what
we call ModFKV, see Algorithm 1) to work with singular values \( \sigma \), and then prove a novel bound that is compatible with Kerenidis & Prakash’s analysis.

This low-rank matrix approximation is thus guaranteed to output good samples, but we are only given a compact description of this matrix. So, in the second part, we use various sampling techniques to go from the compact description to an actual desired sample (or an approximate sample). These sampling techniques solve the problem of sampling from a projection of a vector onto a subspace, so we call the problem \textsc{ProjectionSampling} (see Algorithm 2). Using this technique we can sample the output in time polylogarithmic in \( m \) and \( n \), as desired.

The rest of the paper proceeds as follows. First, we describe basic notation and definitions in Section 2. Next, we give a brief description of the assumptions and model used for analysis in [10], along with the necessary results for our work. Assumptions on the model are given in Section 3 and assumptions on how the input is given (that is, on the input data structure) are in Section 4. These sections are background and can be safely skimmed, since the parameters introduced there only emerge for the last parts of the analysis of the algorithm. Following background, in Section 5 we explore two notions that will be central to our algorithm: vector sampling and approximate orthogonality. Using these notions, we develop subroutines and lemmas that we use in the algorithm, which is described in Section 6. This is split into the two parts described above, which we combine into a complete algorithm into the final part (see Algorithm 3). Finally, we give some final thoughts and open questions in Section 7 and have deferred, self-contained proofs in the Appendix.

2 Definitions

All nonlocal notation can be found here or in Section 3. \([n] \triangleq \{1, \ldots, n\}\). \( f \preceq g \) denotes the ordering \( f = O(g) \) (and correspondingly for \( \succeq \) and \( \asymp \)). For \( U, V \) subspaces of \( \mathbb{R}^n \), let \( U \leq V \) denote “\( U \) is a subspace of \( V \)”. Unless specified otherwise, for a matrix \( A \), \( A_i \) and \( A^{(i)} \) will refer to the \( i \)th row and column, respectively. \( \|A\|_F \) and \( \|A\|_2 \) will refer to Frobenius and spectral norm, respectively. Norm of a vector will always refer to 2-norm. The absolute value of \( x \in \mathbb{R} \) will be denoted either \( |x| \) or \( \text{abs}(x) \). Occasionally, inequalities of the form \( \|x - y\| \leq \varepsilon \) will be phrased in the form \( x = y + E \), where \( \|E\| \leq \varepsilon \). Thus, the letter \( E \) will refer to some form of error.

For a matrix \( A \in \mathbb{R}^{m \times n} \), let \( A = U \Sigma V^T = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^T \) be the SVD of \( A \). Here, \( U \in \mathbb{R}^{m \times m} \) and \( V \in \mathbb{R}^{n \times n} \) are unitary with columns \( \{u_i\}_{i \in [m]} \) and \( \{v_i\}_{i \in [n]} \), the left and right singular vectors, respectively. Since \( U \) and \( V \) are unitary, these columns form orthonormal bases. \( \Sigma \in \mathbb{R}^{m \times n} \) is diagonal with \( \sigma_i \triangleq \Sigma_{ii} \) and the \( \sigma_i \) nonincreasing and nonnegative.

We will use the function \( \ell \) to indicate splitting the singular vectors along a
singular value.

\[ \ell(c) \triangleq \max\{i \mid \sigma_i \geq c\} \]

For example, \(\sigma_1\) through \(\sigma_{\ell(x)}\) gives all of the singular values that are at least \(x\). This notation suppresses the dependence on \(\sigma_i\), but in all cases it will be clear from context.

\(\Pi\) will always refer to orthogonal projectors. That is, if \(\beta = \{b_1, \ldots, b_d\}\) is an orthonormal basis for \(\text{im } \Pi\), then \(\Pi = \sum_{i=1}^{d} b_i b_i^T = BB^T\) for \(B\) the matrix whose columns are the elements of \(\beta\). We will often conflate \(B\) the matrix of basis vectors and \(\beta\) the basis itself.

### 2.1 Low-Rank Approximations

We will use various techniques to describe low-rank approximations of \(A\). All of these techniques will involve projecting the rows onto some span of right singular vectors.

\[ A_k \triangleq A\Pi_k \quad \text{im } \Pi_k = \text{span}\{v_i \mid i \in [k]\} \]

\[ A_{\geq \sigma} \triangleq A\Pi_{\geq \sigma} \quad \text{im } \Pi_{\geq \sigma} = \text{span}\{v_i \mid i \in [\ell(\sigma)]\} \]

\(A_k\) and \(A_{\geq \sigma}\) correspond to the standard notions of low-rank approximations of \(A\). For example, \(A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T\) and is a rank \(k\) matrix minimizing the Frobenius norm distance from \(A\).

We will need to relax this notion for our purposes, and introduce error \(\Pi_E\).

\[ A_{\geq \sigma, \kappa} \triangleq A\Pi_{\geq \sigma, \kappa} = A(\Pi_{\geq \sigma} + \Pi_E) \text{ for some } \Pi_E \text{ satisfying } \text{im } \Pi_E \leq \text{span}\{v_i \mid \ell(\sigma) < i \leq \ell(\sigma(1 - \kappa))\} \]

In words, \(A_{\geq \sigma, \kappa}\) is the projection of \(A\) onto its singular vectors whose values are at least \(\sigma\), perhaps along with some vectors that still become fairly large after applying \(A\). Such a matrix could be produced, say, from having some \(\kappa\)-like error in estimating singular values when finding the low-rank matrix approximation.

Note that \(A_{\geq \sigma, \kappa}\) is underdefined, since different \(\Pi_E\) will result in different matrices. However, this should be thought of as just an error term, and \(A_{\geq \sigma, \kappa}\) is just one of a number of possible matrices with sufficiently small error.\(^1\) \(\kappa\) should be thought of as some constant between 0 and 1 (1/3 will be the eventual value), and \(\sigma\) should be thought of as very large (say, a constant multiple of \(\|A\|_F\)), so \(A_{\geq \sigma, \kappa}\) always has relatively low rank.

\(^1\)This definition of \(A_{\geq \sigma, \kappa}\) is a relaxation of the definition given in [10]. However, we do not lose any precision in error, since our matrices still have the same rank and always lie in the convex hull of the matrices that satisfied the old definition. In this sense, the notion we give is perhaps a more natural one. There will be no issue conflating the two.
It will also be useful to consider a version with two-sided error; we can do this with a simple transform of parameters:

\[
A_{\sigma, \kappa} \triangleq A_{\geq \sigma(1+\kappa), \frac{1}{1+\kappa}} = A(\Pi_{\geq \sigma(1+\kappa)} + \Pi_E) \text{ for some } \Pi_E \text{ satisfying } \\
\text{im } \Pi_E \leq \text{span}\{v_i \mid \ell(\sigma(1+\kappa)) < i \leq \ell(\sigma(1-\kappa))\}
\]

Since \(0 < \kappa < 1\) is a constant, this only perturbs parameters by constant factors.

### 2.2 Sampling

For a vector \(x \in \mathbb{R}^n\), \(Z_x\) will refer to the probability distribution over \([n]\) satisfying

\[
\Pr[Z_x = i] = \frac{x_i^2}{\|x\|^2}
\]

We will call a sample from \(Z_x\) a sample from \(x\).

Notice the following two basic observations. First, \(Z_x\) is the probability distribution resulting from measuring \(|x\rangle\) in the computational basis. Second, access to samples from \(Z_x\) gives a significant amount of power compared to just being given query access to entries of \(x\). For example, while finding a hidden large entry takes \(\Omega(n)\) queries in the latter case, it can be found with a constant number of samples in the former case.

For probability distributions \(P, Q\) over a (discrete) universe \(X\), the total variation distance between them is defined as

\[
TV(P, Q) = \frac{1}{2} \sum_{x \in X} \left| \Pr[P = x] - \Pr[Q = x] \right|
\]

### 3 Model Assumptions

We will now go through the relevant assumptions necessary for our recommendation system, and the resulting goals for our algorithms. As mentioned above, these are the same assumptions as those in [10]: an exposition of these assumptions are also given there.

#### 3.1 Preference Matrix

As a reminder, given \(m\) users and \(n\) products, the preference matrix \(T \in \mathbb{R}^{m \times n}\) contains the information on whether user \(i\) likes product \(j\) in its \(ij\)th location.

We will use the following model:

**Definition.** If product \(j\) is a good recommendation for user \(i\), \(T_{ij} = 1\). If not, \(T_{ij} = 0\).
We can form such a preference matrix from generic sets of data about recommendations, simply by condensing information down to the binary question of whether a product is a good recommendation or not.

**T has small numerical rank.** That is, \( \|T - T_k\| \leq \rho \|T\|_F \) for some \( k \) and \( \rho \ll 1 \). \( k \) should be thought of as constant (polylog\((m, n)\) at worst).

This assumption comes from the intuition that users decide their preference for products based on a small number of factors (e.g. price, quality, and popularity); while different users may value these factors differently, any set of preferences based primarily on a weighting a small number of factors will lead to small numerical rank.

This assumption is standard across recommendation systems and, more generally, similar tasks in data analysis. Theoretical papers state this assumption explicitly: see [7] and [10] for this assumption in recommendation systems, and [3] for more general contexts. Further, this assumption is implicit in the SVD-based techniques used in practice [11].

The low-rank assumption gives \( T \) structure that is relatively robust; that is, only given a small number of entries about \( T \), \( T \) can be reconstructed fairly well. From a reconstruction of \( T \) one can produce good recommendations, so this is a nice structure that recommendation systems can take advantage of.

**Many users have approximately the same number of preferences.** The low-rank assumption is enough to get some bound on quality of recommendations (see Lemma 3.2 in [10]). However, this bound considers “matrix-wide” recommendations. We would like to give a bound on the probability that an output is not a good recommendation for a particular user.

It is not enough to assume that \( \|T - T_k\|_F \leq \rho \|T\|_F \). For example, in a worst-case scenario, a few users make up the vast majority of the recommendations (say, a few users can be recommended any product, and the rest of the users are only happy with four products). In this scenario, the error of the low rank approximation, \( \rho \|T\|_F \), can exceed the mass of recommendations in the non-heavy users, so the error drowns out any possible information about the vast majority of users that could be gained from the low-rank structure.

In addition to being pathological for user-specific bounds, this scenario is orthogonal to our primary concerns: we aren’t interested in providing recommendations to users that desire very few products or users that desire nearly all products, since doing so is intractable and trivial, respectively. To avoid considering such a pathological case, we restrict our attention to the “typical user”:

**Definition.** For \( T \in \mathbb{R}^{m \times n} \), call \( S \subset [m] \) a subset of rows/users \((\gamma, \zeta)\)-typical (where \( \zeta, \gamma > 0 \)) if \(|S| \geq (1 - \zeta)m \) and, for all \( i \in S \),

\[
\frac{1}{1 + \gamma \frac{\sqrt{m}}{\|T\|_F}} \leq \|T_i\| \leq (1 + \gamma) \frac{\|T\|_F}{\sqrt{m}}
\]
γ and ζ can be chosen as desired to broaden or restrict our idea of typical. We can enforce good values of γ and ζ simply by requiring that users have the same number of preferences; for example, this can be done by defining a good recommendation to be the top 100 products for a user, regardless of utility to the user.

Given this definition, we can give a guarantee on recommendations for typical users.

**Theorem 3.1** (Theorem 3.3 of [10]). For $T \in \mathbb{R}^{m \times n}$, $S$ a ($γ, ζ$)-typical set of users, and a matrix $\tilde{T}$ satisfying $\|T - \tilde{T}\|_F \leq \varepsilon\|T\|_F$, for a chosen parameter $\psi > 0$ there exists some $S' \subset S$ of size at least $(1 - \psi - ζ)m$ such that

$$
\Pr_{i \sim S', j \sim \tilde{T}_i} [(i, j) \text{ is bad}] \leq \frac{\varepsilon^2(1 + \varepsilon)^2}{(1 - \varepsilon)^2 \left(1/(1 + \gamma) - \varepsilon/\sqrt{\psi}\right)^2 (1 - \psi - ζ)}
$$

Namely, the above bound is for a sample from $\tilde{T}_i$ with $i \in S'$, averaged over the rows in $S'$. Note: for intuition, if $\varepsilon$ is sufficiently small compared to the other parameters, this bound becomes

$$
O\left(\frac{\varepsilon^2(1 + γ)^2}{1 - δ - ζ}\right)
$$

**We know k.** More accurately, a rough upper bound for $k$ will suffice. Such an upper bound can be determined from data.

All of the above taken together, we have reduced the problem of “find a good recommendation for a user” to “given some entries from $T$, sample from $\tilde{T}_i$ for some $\tilde{T}$ satisfying $\|T - \tilde{T}\| \leq \varepsilon\|T\|$ for small $\varepsilon.$” Further, if $\tilde{T}$ is an approximation of $T_k$, this is good enough.

### 3.2 Matrix Sampling

We have stated our assumptions on the full preference matrix $T$, but we also need assumptions on the information we are given about $T$.

For example, if we are given information heavily concentrated on a few rows or columns, then we don’t have enough information to give recommendations to most users or recommend most products, respectively.

We will use a model for subsampling for matrix reconstruction given by Achlioptas and McSherry in [2]. In this model, the entries we are given are chosen uniformly over all entries. This model is common for theoretical recommendation systems literature; see [7] and [10].

Specifically, we have the following:
**Definition.** For a matrix $T \in \mathbb{R}^{m \times n}$, let $\hat{T}$ be a random matrix i.i.d. on its entries, where

$$
\hat{T}_{ij} = \begin{cases} 
T_{ij} & \text{with probability } p \\
0 & \text{with probability } 1 - p 
\end{cases} \quad (\ast)
$$

Notice that $E[\hat{T}] = T$.

When the entries of $T$ are bounded, $\hat{T}$ is $T$ perturbed by a random matrix whose entries are approximately i.i.d. Thus, intuition suggests that if $T$ has large singular values, these will not be perturbed much by the random matrix. Formally speaking, this means that if $\|T - T_k\|$ is small, then $\|T - \hat{T}_k\|$ should also be small. This turns out to be true:

**Theorem 3.2** (Reformulation of 1.1, 1.4 given by [10]). Let $T = \mathbb{R}^{m \times n}$, and let $\hat{T}$ be the above-defined random matrix $(\ast)$ where $p = n/(\eta\|T\|_F^2)$ for a parameter $\eta > 0$. Let $b = \max_{ij} |T_{ij}|$. Then with probability at least $1 - \exp(-19(\log n)^4)$, for any $k$,

$$
\|T - \hat{T}_k\|_F \leq \|T - T_k\|_F + 6b^2\eta^2\kappa^3\|T\|_F
$$

In the recommendation systems context, this result means that if we are given $\hat{T}$ as our input, $\hat{T}_k$ is a reconstruction that will give us good samples. This works when $p$ is fairly small: a constant fraction works, and for this theorem, slightly subconstant works as well (say, $1/m$). However, reconstructing $\hat{T}_k$ is hard. For example, if $\hat{T}$’s singular values are close together, detecting such small differences becomes infeasible. Thus, we will relax the low-rank approximation we aim for.

The following is a technical result from [10] that says that, under appropriate circumstances, it’s good enough to reconstruct $\hat{T}_{\geq \sigma, \kappa}$ for appropriate $\sigma$ and $\kappa$.

**Theorem 3.3** (4.3 of [10]). Let $T \in \mathbb{R}^{m \times n}$ and let $\hat{T}$ be the random matrix defined in $(\ast)$, with $p = n/(\eta\|T\|_F^2)$ and $\max_{ij} |T_{ij}| = 1$. Let $\eta, \mu > 0$ be chosen parameters, and let $\sigma = \sqrt{\mu/k}\|\hat{T}\|_F$. Let $\kappa > 0$ be a precision parameter. Then with probability at least $1 - \exp(-19(\log n)^4)$,

$$
\|T - \hat{T}_{\geq \sigma, \kappa}\|_F \leq 3\|T - T_k\|_F + \left(6\eta^2\kappa^3\mu^{-\frac{1}{2}}(2 + (1 - \kappa)^{-\frac{1}{2}} + \sqrt{2(3 - \kappa)}\mu^{\frac{1}{2}}p^{-\frac{1}{2}})\right)\|T\|_F
$$

If $\|T - T_k\|_F \leq \rho\|T\|_F$ and $\|T\|_F \geq \frac{9}{2\sqrt{2}}\sqrt{n\kappa}$, then we can choose $\eta, \mu, \kappa$ such that $\|T - \hat{T}_{\geq \sigma, \kappa}\|_F \leq 3(\rho + \varepsilon)\|T\|_F$.

$\mu$ represents the threshold for detecting singular values; when $\mu = 1/c$, for example, the rank of $T_{\geq \sigma, \kappa}$ will be $ck/(1 - \kappa)$ at most. $\eta$ is the parameter that corresponds to the number of entries given as input, and $\kappa$ is an error parameter.

To get the desired bound, the parameters are $\kappa = 1/3$, $\mu = \varepsilon^2p/8$, and any $\eta \leq \frac{2^{9/4}\eta^3/8\kappa^{3/2}}{3k^{1/4}\|T\|_F^{1/2}}$. 

9
With this theorem, our goal becomes sampling from \((\hat{T}_{\geq \sigma, \kappa})_i\) with \(\sigma, \kappa\) chosen with respect to desired \(\varepsilon\). Since these inputs only depend on \(p, \|\hat{T}\|_F, k, \text{ and } \varepsilon\), and all of these parameters are either known to the user (i.e. \(k, \varepsilon\)) or computable from the input (i.e. \(p, \|\hat{T}\|_F\)), the algorithm does not need to know any of the other parameters in the analysis (\(m\) and \(n\) are implicit in the input)\(^2\).

4 Data Structure

Since we are interested in achieving sublinear bounds for our algorithm, we need to concern ourselves with how our input is given.

In our model, our input is whatever entries we have of the preference matrix \(T\), along with a user \(i\), and the output is a recommendation \(j\) that is likely to be a good recommendation for user \(i\), for most \(i\).

If the entries are given in some unprocessed format (say, for example, via a stream of the known entries of \(T\)), then clearly linear time is required to even parse the stream into a usable form. Even when the input is relatively structured (for example, if we are given the known entries of \(T\) sorted by row and column), there is no hope to sample the low-rank approximation of a generic matrix in sublinear time.

To avoid these issues, we will instead consider our input stored in a low-overhead data structure, which will give the power necessary for a generic sublinear algorithm.

**Lemma 4.1.** There exists a data structure storing a vector \(v \in \mathbb{R}^n\) with \(w\) nonzero entries in \(O(w \log^2(n))\) space, supporting the following operations:

- Reading and updating an entry of the vector in \(O(\log^2 n)\) time
- Finding \(\|v\|^2\) in \(O(1)\) time
- Sampling from \(Z_v\) in \(O(\log^2 n)\) time

\[^2\text{For a user to verify the assumptions of Theorem 3.3 and guarantee its result, bounds for } \|T\|_F \text{ are necessary. Knowledge of } \rho \text{ is helpful to make sense of the result.}\]
This can be achieved through a binary search tree data structure, pictured above for \( n = 4 \). The leaf nodes store \( v_i \) (or equivalently, \( v_i^2 \) and the sign of \( v_i \)), and interior nodes are just the sum of their children. Updates can be done quickly just by updating all of the nodes above a particular leaf. Sampling from \( Z_v \) is possible by starting from the top of the tree and randomly recursing on a subtree dependent on the weight of that subtree. When \( v \) is sparse, the tree can be pruned to only nonzero nodes.

We can adapt the BST data structure to matrices:

**Proposition 4.2.** Consider a matrix \( A \in \mathbb{R}^{m \times n} \), and let \( A_i \) refer to the \( i \)th row of \( A \). Let \( \tilde{A} \in \mathbb{R}^m \) be a vector whose \( i \)th entry is \( \| A_i \|_2^2 \).

There exists a data structure storing a matrix \( A \in \mathbb{R}^{m \times n} \) with \( w \) nonzero entries in \( O(w \log^2(mn)) \) space, supporting the following operations:

- Reading and updating an entry of the matrix in \( O(\log^2 mn) \) time
- Finding \( \tilde{A}_i \) in \( O(\log^2 m) \) time
- Finding \( \| A \|_F^2 \) in \( O(1) \) time
- Sampling from \( Z_{\tilde{A}} \) and \( Z_{A_i} \) in \( O(\log^2 mn) \) time

This can be done by having a copy of a data structure specified by Lemma 4.1 for each row of the matrix. Then the root nodes of these data structure are the entries of \( \tilde{A} \), and a BST is constructed using these as the input vectors. This has all of the desired properties, and in fact, is the data structure Kerenidis and Prakash use to generate arbitrary quantum states (Theorem A.1 in \cite{10}). Thus, our algorithm can operate on the same input (although any data structure supporting the operations detailed in Proposition 4.2 will suffice).

## 5 Notions and Properties

There are several strategies that will be central to the analysis of our algorithm. First is the notion of sampling from a vector, which is both the goal of our algorithm and also used in the techniques we use to solve the sampling tasks involved in our algorithm. Second is the notion of approximate orthonormality, which will be a useful property to maintain; because our algorithm is sublinear in the input, we cannot produce vectors that are exactly orthonormal, but it will be enough to have approximately orthonormal vectors.

### 5.1 Vector Sampling

Recall how we defined sampling from a vector.

**Definition.** For a vector \( x \in \mathbb{R}^n \), let \( Z_x \) refers to the distribution on \( |n| \) such that \( \Pr[Z_x = i] = x_i^2/\|x\|^2 \). We call an \( i \sim Z_x \) a sample from \( x \).
For our algorithm, we will want to be able to perform such samples to various aspects of our matrix $A \in \mathbb{R}^{m \times n}$. Let $\tilde{A} \in \mathbb{R}^m$ refer to the vector where $A_i = \|A_i\|^2$. Then we want the ability to sample from $Z_{\tilde{A}}$ and $Z_A$. This is a standard assumption to make (for examples, see [7] and [8]). We have shown in the previous section that this can be done easily with a data structure (see Proposition 4.2).

Vector sampling has nice properties with respect to the norms that they relate. Namely, closeness of vectors in 2-norm implies closeness of their respective distributions in TV distance.

**Lemma 5.1.** For $x, y \in \mathbb{R}^n$ satisfying $\|x - y\| \leq \varepsilon$, the corresponding distributions $Z_x, Z_y$ satisfy $TV(Z_x, Z_y) \leq 2\varepsilon/\|x\|$.

**Proof.** Let $\hat{x}$ and $\hat{y}$ be the normalized vectors $x/\|x\|$ and $y/\|y\|$.

$$TV(Z_x, Z_y) = \frac{1}{2} \sum_{i=1}^n |\hat{x}_i^2 - \hat{y}_i^2|$$

$$= \frac{1}{2} \langle |\hat{x} - \hat{y}|, |\hat{x} + \hat{y}| \rangle$$

$$\leq \frac{1}{2} \|\hat{x} - \hat{y}\| \|\hat{x} + \hat{y}\|$$

$$\leq \frac{1}{\|x\|} \|x - y - (\|x\| - \|y\|)\hat{y}\|$$

$$\leq \frac{1}{\|x\|} \left( \|x - y\| + \text{abs}(\|x\| - \|y\|) \right)$$

$$\leq \frac{2\varepsilon}{\|x\|}$$

Now, we will give two subroutines that can be performed, supposing access to vector sampling.

First, we show that we can estimate the dot product of two vectors well.

**Proposition 5.2.** Given query access to $x, y \in \mathbb{R}^n$ and sample access to $Z_x$, as well as knowledge of $\|x\|$, $\langle x, y \rangle$ can be estimated to $\|x\|\|y\|\varepsilon$ additive error with $1 - \delta$ probability in $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ queries and samples.

**Proof.** Perform samples in the following way: let the random variable $Z$ be $y_i/x_i$ with probability $\frac{x_i^2}{\|x\|^2}$ (select the index by sampling from $Z_x$).

$$E[Z] = \sum \frac{y_i}{x_i} \frac{x_i^2}{\|x\|^2} = \frac{\sum x_iy_i}{\|x\|^2} = \frac{\langle x, y \rangle}{\|x\|^2}$$

$$Var[Z] = \sum \left( \frac{y_i}{x_i} \right)^2 \frac{x_i^2}{\|x\|^2} = \frac{\sum y_i^2}{\|x\|^2} = \frac{\|y\|^2}{\|x\|^2}$$
Since we know $\|x\|$, we can normalize by it to get a random variable whose mean is $\langle x, y \rangle$ and whose standard deviation is $\sigma = \|x\||y\|$.

The rest follows from standard techniques: we take the median of $6\log \frac{1}{\delta}$ copies of the mean of $\frac{6}{\epsilon^2}$ copies of $Z$ to get within $\epsilon\sigma = \epsilon\|x\||y\|$ with probability $1 - \delta$ in $O(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ accesses.

$$
\Pr\left[ \text{median} - \langle x, y \rangle \geq \epsilon\sigma \right] \leq \Pr\left[ \sum_{i=1}^{6\log \frac{1}{\delta}} 1_{\{\text{mean}_i - \langle x, y \rangle \geq \epsilon\sigma \}} \geq 3\log \frac{1}{\delta} \right] \\
\leq \Pr\left[ \sum_{i=1}^{6\log \frac{1}{\delta}} \text{Bernoulli}_i(1/6) \geq 3(\log \frac{1}{\delta}) \right] \\
\leq e^{-\frac{2^2}{3^2} \log 1/\delta} = \delta
$$

Second, we show that, given sample access to some vectors, we can sample from a linear combination of them.

**Proposition 5.3.** Suppose we are given some $V \in \mathbb{R}^{n \times k}$ with columns $V^{(i)}$. We have query access, as well as sample access to $Z_{V^{(i)}}$ and knowledge of $\|V^{(i)}\|$. Then given $c \in \mathbb{R}^k$ (just as input) and a $C$ satisfying

$$
C \|Vc\|^2 \geq \sum_{i=1}^k |c_i V^{(i)}|^2
$$

we can output a sample from $Vc$ in at most $Ck$ queries in expectation.

$C$ is, to some extent, a measure of orthogonality. If we allow $c$ to be chosen arbitrarily poorly, $C$ is the best when the columns of $V$ are orthogonal ($C = 1$ works), and can be forced to be much larger than 1 otherwise, since cancellation can occur.

**Proof.** The strategy we use is rejection sampling. Suppose we can sample from the distribution $X$, and we want to sample from the distribution $Y$. If there is a $C$ such that for all $i$, $C \Pr[X = i] \geq \Pr[Y = i]$, then rejection sampling gives a sample from $Y$ in at most $C$ queries in expectation.
Let $Z$ be a sample from $Z^{(i)}$ proportional to $\|c_i V^{(i)}\|^2$. Then

$$\Pr[Z = i] = \frac{\sum_{\alpha=1}^{k} \|c_{\alpha} V^{(\alpha)}\|^2 \left( \frac{V^{(\alpha)}}{\|V^{(\alpha)}\|} \right)^2}{\sum_{\beta=1}^{k} \|c_{\beta} V^{(\beta)}\|^2} = \frac{\sum_{\alpha=1}^{k} (c_{\alpha} V^{(\alpha)})^2}{\sum_{\beta=1}^{k} \|c_{\beta} V^{(\beta)}\|^2} \geq \frac{1}{k} \left( \sum_{\alpha=1}^{k} c_{\alpha} V^{(\alpha)} \right)^2 \sum_{\beta=1}^{k} \|c_{\beta} V^{(\beta)}\|^2 \geq \frac{(V c)^2}{kC\|V\|^2} \frac{1}{k} \Pr[Z_{Vc} = i]$$

The result follows.

We will need a technical lemma to allow us to perform rejection sampling when conditions are relaxed slightly. Typically, rejection sampling requires ability to sample and compute probabilities from a distribution $p$, ability to compute probabilities from the target distribution $q$, and a bound on $p/q$. We will relax the requirements for computing probabilities, instead requiring only approximate knowledge of probabilities.

**Lemma 5.4.** Suppose we can sample from the distribution $p(x)$, and we know the value of $p'(x)$, with $a \leq p(x)/p'(x) \leq A$ for all $x$ in the supports of $p, p'$. ($p'$ need not be a probability distribution.) Further suppose that we want to sample from the distribution $q(x)$, and we know the value of $\lambda q(x)$ for $\lambda \leq B$ (with $\lambda$ unknown, $B$ known).

Finally, suppose we know a constant $C$ such that $q(x)/p'(x) \leq C$ for all $C$.

Then we can output a sample from a distribution $q'(x)$ satisfying $TV(q, q') \leq (A - a)/2a$ in $CB/\lambda a$ expected queries.

**Proof.** The algorithm is standard rejection sampling: sample $x$ from $p$, and keep the sample with probability $\lambda q(x)/CBp'(x)$. Notice that we know $\lambda q(x), C, B,$ and $p'(x)$.

Then we accept with probability

$$\sum \frac{\lambda q(x)}{CBp'(x)} p(x) \geq \frac{a\lambda}{CB}$$

and we output the probability distribution satisfying

$$q'(i) = \frac{\Pr[\text{accept on } i]}{\Pr[\text{accept}]} = \frac{\lambda q(i)p(i)}{CBp'(i)} = \frac{q(i)p(i)}{p'(i)}$$
and the TV distance is bounded as follows:

\[
TV(q, q') = \frac{1}{2} \sum_i \text{abs} \left( q(i) - \frac{q(i)p(i)}{p'(i)} \right)
\]

\[
= \frac{1}{2} \sum_i q(i) \text{abs} \left( 1 - \frac{p(i)}{p'(i)} \right) \sum q(x)p(x) \frac{p'(x)}{p'(x)}
\]

\[
\leq \frac{1}{2} \sum_i q(i) \frac{A-a}{a} = \frac{A-a}{2a}
\]

The last step results from the inequality

\[
\frac{a}{A} \leq \frac{p(i)}{p'(i)} \sum q(x)p(x) \frac{p'(x)}{p'(x)} \leq \frac{A}{a}
\]

### 5.2 Approximate Orthonormality

We will not be able to get precisely orthonormal vectors from our algorithm. However, we will define a notion of approximate orthonormality that will work in our case, and prove that such vectors have exactly the properties one would expect from the name.

**Definition.** Call \( V \in \mathbb{R}^{n \times k} \) a set of \( k \) vectors \( \alpha \)-approximately orthonormal for \( \alpha > 0 \) if \( V \) satisfies

\[
I - \frac{\alpha}{k} \mathbf{1} \mathbf{1}^T \preceq V^T V \preceq I + \frac{\alpha}{k} \mathbf{1} \mathbf{1}^T
\]

Here, \( \preceq \) denotes entrywise inequality and \( \mathbf{1} \) denotes the vector of ones. This definition is equivalent to saying that any dot product is at most \( \alpha \) off from the expected result.

Note that we will always assume \( \alpha < 1 \).

**Lemma 5.5.** If \( V \) is \( \alpha \)-approximately orthonormal, then there exist \( k \) orthonormal vectors \( U \) spanning the column space of \( V \) satisfying \( \|U - V\|_F \leq \alpha/\sqrt{2} + O(\alpha^2) \).

**Proof.** Use the QR decomposition \( V = QR \), where \( Q \) is unitary and \( R \) is upper triangular with positive entries on its diagonal. Then

\[
I - \frac{\alpha}{k} \mathbf{1} \mathbf{1}^T \preceq V^T V = \tilde{R}^T \tilde{Q}^T QR = \tilde{R}^T R \preceq I + \frac{\alpha}{k} \mathbf{1} \mathbf{1}^T
\]

Since \( R \in \mathbb{R}^{n \times k} \) is upper triangular, we can think about \( R \) as \( k \times k \). Thus, \( R^T R \) is an approximate Cholesky factorization of \( I \), with error \( \frac{\alpha}{k} \mathbf{1} \mathbf{1}^T \); by Theorem 1 in [6], \( \|R - I\|_F \leq \frac{\alpha}{\sqrt{2}} + O(\alpha^2) \).
We will choose $U = QI$, where $I$ is the $k \times k$ identity extended to $n \times k$. Clearly this satisfies the orthonormality and subspace conditions, and $\|U - V\|_F = \|Q(I - R)\|_F \leq \alpha/\sqrt{2} + O(\alpha^2)$.

The above lemma allows makes the notion of approximate orthonormality very versatile: we can treat these vectors as orthonormal with only $\alpha$ change in Frobenius norm. We give a basic corollary as an example.

**Corollary 5.6.** For $V$ $\alpha$-approximately orthonormal, $\|VV^T - \Pi_V\|_F \lesssim \alpha$, where $\Pi_V$ is the orthogonal projector on the image of $V$.

Essentially, we can treat $VV^T$ as an orthonormal projector. This follows from Lemma 5.5 and a simple calculation:

$$VV^T = (U + E)(U + E)^T = UU^T + UE^T + EU^T + EE^T \Rightarrow \|VV^T - UU^T\|_F \lesssim \alpha$$

Next, we will remark that sampling from a linear combination of approximately orthonormal vectors can be done quickly via sampling, even without knowledge of the norms of these vectors.

**Lemma 5.7** (rejection sampling for approximately orthonormal vectors). Given sample and query access to $V$ a set of $k \alpha$-approximately orthonormal vectors, rejection sampling can output a sample from a distribution $(\alpha + O(\alpha^2))$-close to $Z_{Vc}$ in $k(1 + O(\alpha))$ expected queries.

Notice that when $V$ is fully orthonormal, by Proposition 5.3 $k$ samples always suffice for a sample, matching the above result when $\alpha = 0$.

**Proof.** Use Lemma 5.4, we will show that all of the criteria are satisfied.

We can sample from $p = (\sum c_i^2 Z_{V(i)})/(\sum c_i^2)$ (this is just a linear combination of $Z_{V(i)}$). We can compute $p' = (\sum c_i^2 \|V(i)\|^2 Z_{V(i)})/(\sum c_i^2)$, since this cancels out the normalization constant of $Z_{V(i)}$.

$$\frac{(\sum c_i^2 Z_{V(i)})/(\sum c_i^2)}{(\sum c_i^2 \|V(i)\|^2 Z_{V(i)})/(\sum c_i^2)} = \frac{\sum c_i^2 Z_{V(i)}}{\sum c_i^2 \|V(i)\|^2 Z_{V(i)}}$$

so $a = 1/(1 + \alpha)$ and $A = 1/(1 - \alpha)$.

We want to sample from $q = Z_{Vc}$, and we know the values of $q = \|Vc\|^2 Z_{Vc}$, and the bound $\|Vc\|^2 \leq (1 + O(\alpha))\|c\|^2$.

Finally,

$$\frac{q(i)}{p'(i)} = \frac{(Vc)^2}{\|Vc\|^2} = \frac{\sum c_i^2 \|V(i)\|^2}{\sum (c_i^2 V^{(x)}_i)^2} = \frac{\sum c_i^2 (\sum c_i V^{(x)}_i)^2}{\|VTc\|^2 \sum (c_i V^{(x)}_i)^2} = \frac{k}{1 - O(\alpha)} = k(1 + O(\alpha))$$

The result follows. \qed

16
6 Algorithm

As a reminder, from Section 3 and Section 4 we have determined that the goal of our algorithm is to, given an input matrix $A$ supporting querying and sampling operations (see Proposition 4.2) and a user $i$, sample from the $i$th row of $A_{\geq \sigma, \kappa}$. This is a variant of the standard task of computing a low-rank approximation.

We are able to use a well-known result (Frieze, Kannan, and Vempala’s sampling algorithm for low-rank matrix approximations [8]) to get most of the way there. The rest is done using the sampling techniques we discussed in Section 5.1.

6.1 Finding a Low-Rank Approximation

**Theorem 6.1.** Given a matrix $A \in \mathbb{R}^{m \times n}$ in the BST data structure, along with parameters $\sigma, \kappa, \varepsilon$, where $\kappa$ is constant, there is an algorithm that will output a description of some $D^*$ satisfying $\|D^* - A_{\geq \sigma, \kappa}\|_F \leq \varepsilon \|A\|_F$ with probability $1 - \delta$ and $O\left(poly\left(\frac{\|A\|_F^2}{\sigma^2}, \varepsilon, \log \frac{1}{\delta} \right) \log^2 mn\right)$ query (and time) complexity.

To prove this theorem, we will modify the algorithm given in [8] and show that it satisfies the desired properties. The algorithm is shown below.

**Algorithm 1: ModFKV**

**Input:** Matrix $A \in \mathbb{R}^{m \times n}$ supporting operations in Proposition 4.2, threshold $\sigma$, error parameter $\varepsilon > 0$

**Output:** A description of an output matrix $D^*$

Set $K = \frac{\|A\|_F^2}{\sigma^2}$;

Set $\bar{\varepsilon} = \frac{\varepsilon^2}{\sqrt{K}}$;

Set $p = 10^7 \max\{\frac{K^4}{\varepsilon^4}, \frac{K^2}{\varepsilon^2}\}$;

Independently choose rows $i_1, \ldots, i_p$ according to distribution $Z_{\bar{A}}$;

Independently choose columns $j_1, \ldots, j_p$ according to distribution $Z_{\bar{A}, s}$, where $s$ is uniformly picked from $[p]$;

Let the resulting $p \times p$ submatrix, with row $r$ normalized by $\frac{1}{\sqrt{p \Pr[Z_{\bar{A}} = i_r]}}$ and column $c$ normalized by $\frac{1}{\sqrt{p \Pr[Z_{\bar{A}} = j_c]}}$, be denoted $W$;

Compute the left singular vectors of $W u^{(1)}(1), \ldots, u^{(k)}(1)$ that correspond to singular values $\sigma^{(1)}, \ldots, \sigma^{(k)}$ larger than $\sigma$;

Output $i_1, \ldots, i_p, u^{(1)}(1), \ldots, u^{(k)}(1)$, and $\sigma^{(1)}, \ldots, \sigma^{(k)}$ as the description of the output matrix $D^*$;

We can get the output matrix $D^*$ by projecting $A$ onto a subspace defined by vectors $\hat{V}$. That is, $D^* = AVV^T = A \sum_{t=1}^{k} (\hat{v}^{(t)}) (\hat{v}^{(t)})^T$. The columns of $\hat{V}$ are
a linear combination of rows of $A$. Let $S$ be the submatrix given by restricting
the rows to $i_1, \ldots, i_p$. Then $\hat{v}^{(i)} \triangleq S^T u^{(i)}/\sigma^{(i)}$.

To summarize, ModFKV subsamples a matrix, computes the large singular
vectors of this matrix, and outputs those singular vectors, with the assertion
that the singular vectors of the subsample give a good description of the singular
vectors of the full matrix. This description of the algorithm is simply taken from
Frieze, Kannan, and Vempala’s paper [8], and there more of the underpinnings
are explained.

ModFKV differs from FKV only in that $\sigma$ is taken as input instead of $k$, and
is used as the threshold for the singular vectors. As a result of this change, $K$
replaces $k$ in the subsampling steps, and $\sigma$ replaces $k$ in the SVD step. Notice
that the number of singular vectors taken (denoted $k$) is at most $K$, so in effect,
we are running FKV and just throwing away some of the smaller singular vectors.
The filter step in the original algorithm does nothing in our case, so it is not
shown here.

We will note the following properties of the modified algorithm:

- **ModFKV** has the same time and query bounds as FKV with $K$ in place
  of $k$. This proves the above query and time complexity.

- **ModFKV** only differs from FKV run on $K$ in that **ModFKV** takes fewer
  singular vectors. Thus, because a larger $K$ leads to a larger sample, the
correctness bound for FKV run on $\ell$ holds for the output of **ModFKV**.
In particular, the output $D^*$ is rank at most $k$, and

$$\|A - D^*\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon \|A\|_F^2$$

which is the main bound given in the paper.

- Two facts about the description will be relevant: the $u^{(i)}$ have unit norm
  and the rows of $A$ (and the rows of $S$) support the operations detailed in
  Lemma 4.1.

- By Proposition 4.2, there is only a log$^2 mn$ blowup in number of queries
  when using the BST data structure; it is obvious that we can sample
  exactly according to the sampling assumptions needed to run FKV.

- We use $\bar{\varepsilon}$ instead of $\varepsilon$ as our error parameter, which will be relevant for
  the bounds we get.

**Proposition 6.2.** ModFKV satisfies the following inequality:

$$\ell((1 + \varepsilon^2)\sigma) \leq k \leq \ell((1 - \varepsilon^2)\sigma)$$

**Proof.** ModFKV can compute the first $k$ singular values to within a cumulative
additive error of $\varepsilon \|A\|_F$. (This is stated offhand in [8], but it follows from simple
manipulation of their analysis.)

18
Thus, ModFKV could only conceivably take a singular vector \( v \) such that 
\[
\| A v \| \geq \sigma - \bar{\varepsilon} \| A \|_F = \sigma (1 - \bar{\varepsilon} \| A \|_F / \sigma) = \sigma (1 - \varepsilon^2),
\]
and analogously for the upper bound.

It will be very useful for us to note the nice property of the singular vectors we are projecting upon.

**Proposition 6.3.** The output singular vectors \( \hat{v}_1, \ldots, \hat{v}_k \) are \( \varepsilon^4 \)-approximately orthonormal. Further, \( \| \hat{v}_i \|_2 \geq 1 \).

We defer the proof of this to the appendix, since it uses notation from FKV.

Before we begin the proof that ModFKV outputs a matrix with the desired bounds, we will perform the following slight change. It will be nicer to consider double-sided \( \kappa \)-error in the analysis, rather than single-sided as \([10]\) does. As a reminder from Section 2.1, \( A_{\sigma, \kappa} \) refers to double-sided error: projecting onto 
\[
\{ v_i \mid i \in [\ell(\sigma (1 + \kappa))] \},
\]
and also some vectors in the span of \( \{ v_i \mid \ell(\sigma (1 + \kappa)) < i < \ell(\sigma (1 - \kappa)) \} \).

Now, we prove a bound for ModFKV. To the author’s knowledge, the form of this bound is novel for its context, so it may be of independent interest.

**Theorem 6.4.** \( \| D^* - A_{\sigma, \kappa} \|_F \lesssim \varepsilon \| A \|_F / \kappa \).

While this bound might not intuitively follow from \([8]\), they essentially say the same thing. \([3]\) says that \( D^* \) doesn’t have to be the projection onto the top singular vectors, but the singular vectors that it takes have to be similar in quality. This implies that the vectors of exceptional quality must be taken and the vectors of awful quality cannot be taken, which is essentially Theorem 6.4.

Our proof only relies on \([3]\) and Proposition 6.2, both of which are very typical results, so it is likely that this bound can find use elsewhere.

**Proof.** If we get a bound on \( \| D^* - A_{\sigma_k, \kappa'} \|_F \) for all possible \( k \), then by Proposition 6.2 this is equivalent to a generic bound on \( \| D^* - A_{\sigma, \kappa} \|_F \), where \( \kappa = \kappa' / (1 + \varepsilon^2) \). We will use \( \sigma \) to refer to \( \sigma_k \) for a particular choice of \( k \).

3If the more standard form of the bound, the one given in \([8]\), can be used to prove recommendation system bounds without passing through this form, the recommendation system analysis can be greatly simplified.

4An analogous proof gives the bound \( \| V V^T - \Pi_{\sigma, \kappa} \|_2 \leq \varepsilon \| A \|_F \), which gives a more generally applicable result.
Using the definition of the description, we rewrite the original expression:

\[
\|D^* - A_{\sigma,\kappa'}\|_F^2 = \|U \Sigma V^T (\hat{V} \hat{V}^T - \Pi_{\sigma,\kappa'})\|_F^2 \\
= \|\Sigma V^T (\hat{V} \hat{V}^T - \Pi_{\sigma,\kappa'})\|_F^2 \\
= \min_{m,n} \sum_{i=1}^{\sigma,\kappa'} \sigma_i^2 \|v_i^T (\hat{V} \hat{V}^T - \Pi_{\sigma,\kappa'})\|_F^2 \\
= \min_{m,n} \sum_{i=1}^{\sigma,\kappa'} \sigma_i^2 \|v_i^T (I - \hat{V} \hat{V}^T) - v_i^T (I - \Pi_{\sigma,\kappa'})\|_F^2
\]

This calculation realizes the intuition that \(D^*\) and \(A_{\sigma,\kappa'}\) are close when their corresponding projectors behave in the same way. Let \(a_i = v_i^T (I - \hat{V} \hat{V}^T)\), and \(b_i = v_i^T (I - \Pi_{\sigma,\kappa'})\). Note that

\[
b_i = \begin{cases} 
0 & \sigma_i \geq (1 + \kappa') \sigma \\
v_i^T \Pi_E (1 + \kappa') \sigma > \sigma_i \geq (1 - \kappa') \sigma \\
v_i^T (1 - \kappa') \sigma > \sigma_i
\end{cases}
\]

Where \(\Pi_E\) is as defined in Section 2.1. Using the first and third case, and the fact that orthogonal projectors \(\Pi\) satisfy

\[
\|v - \Pi v\|_F^2 = \|v\|_F^2 - \|\Pi v\|_F^2,
\]

the formula becomes

\[
\|D^* - A_{\sigma,\kappa'}\|_F^2 = \sum_{i=1}^{\ell(\sigma(1+\kappa'))} \sigma_i^2 \|a_i\|^2 + \sum_{i=\ell(\sigma(1+\kappa'))+1}^{\ell(\sigma(1-\kappa'))} \sigma_i^2 \|a_i - b_i\|^2 \\
+ \sum_{i=\ell(\sigma(1-\kappa'))+1}^{\min_{m,n}} \sigma_i^2 (1 - \|a_i\|^2)
\]

**Lemma 6.5.** Let \(N = \min m, n\). The following system of equations holds:

\[
\sum_{i=1}^{N} \sigma_i^2 \|a_i\|^2 \leq \sum_{k+1}^{N} \sigma_i^2 + \bar{\varepsilon} \sum_{i=1}^{N} \sigma_i^2 \sum \|a_i\|^2 \geq N - k
\]

\[\|a_i\|^2 \in [0, 1]\]

\(\sigma_i^2\) are nonincreasing

**Proof.** The first equation follows from \(1\):

\[
\|A - D^*\|_F^2 \leq \|A - A_k\|_F^2 + \bar{\varepsilon} \|A\|_F^2
\]

\[
\|U \Sigma V^T (I - \hat{V} \hat{V}^T)\|_F^2 \leq \sum_{k+1}^{N} \sigma_i^2 + \bar{\varepsilon} \|A\|_F^2
\]

\[
\sum_{i=1}^{N} \sigma_i^2 \|a_i\|^2 \leq \sum_{i=\ell(\sigma)+1}^{N} \sigma_i^2 + \bar{\varepsilon} \|A\|_F^2
\]

20
The second equation is equivalent to the statement \( \|I - \hat{V}\hat{V}^T\|_F^2 \geq N - k \), which is always true for \( \hat{V} \in \mathbb{R}^{n \times k} \).

It turns out that this system of equations is enough to show that the \( \|a_i\|^2 \) behave the way we want them to. The following lemma gives these results (the proof is deferred to the appendix, but it is straightforward since the system is linear).

**Lemma 6.6.** The system of equations in Lemma 6.5 imply the following, for \( 0 < \kappa \leq \frac{1}{2} \):

\[
\sum_{i=1}^{\ell(\sigma(1+\kappa'))} \sigma_i^2 \|a_i\|^2 \leq \bar{\varepsilon} \left(1 + \frac{1}{\kappa'}\right) \|A\|_F^2 \quad \text{and} \quad \sum_{i=\ell(\sigma(1-\kappa'))+1}^{N} \sigma_i^2 (1 - \|a_i\|^2) \leq \bar{\varepsilon} \left(1 + \frac{1}{\kappa'}\right) \|A\|_F^2
\]

Now, applying the top two inequalities:

\[
\|D^* - A_{\sigma,\kappa}\|_F^2 \leq 2\bar{\varepsilon} \left(1 + \frac{1}{\kappa'}\right) \|A\|_F^2 + \sum_{i=\ell(\sigma(1+\kappa'))+1}^{\ell(\sigma(1-\kappa'))} \sigma_i^2 \|a_i - b_i\|^2
\]

We just need to bound the last term in the inequality now. Notice the following:

\[
\sum_{i=\ell(\sigma(1+\kappa'))+1}^{\ell(\sigma(1-\kappa'))} \sigma_i^2 \|a_i - b_i\|^2 = \sum_{i=\ell(\sigma(1+\kappa'))+1}^{\ell(\sigma(1-\kappa'))} \sigma_i^2 \|v_i^T(\hat{V}\hat{V}^T - \Pi_E)\|^2 \leq \sigma^2 (1 + \kappa')^2 \|U^T(\hat{V}\hat{V}^T - \Pi_E)\|^2
\]

where \( U \) is the set of vectors \( v_{\ell(\sigma(1+\kappa'))+1} \) through \( v_{\ell(\sigma(1-\kappa'))} \). Notice that \( \Pi_E \) is the error component of the projection, and this error can be any projection onto a subspace spanned by \( U \). Thus, to bound the above we just need to pick an orthogonal projector \( \Pi_E \) making the norm as small as possible. If \( UU^T\hat{V}\hat{V}^T \) were an orthogonal projection, this would be easy:

\[
\|U^T(\hat{V}\hat{V}^T - UU^T\hat{V}\hat{V}^T)\|_F^2 = 0
\]

However, this is likely not the case. \( UU^T\hat{V}\hat{V}^T \) is close to an orthogonal projector, though, by the lemma below.

\[
\|U^T(\hat{V}\hat{V}^T - M)\|_F^2 = \|U^T(\hat{V}\hat{V}^T - (UU^T\hat{V}\hat{V}^T + E))\|_F^2 = \|U^TE\|_F^2 \leq 6\bar{\varepsilon} \frac{\|A\|_F^2}{\kappa'\sigma^2} + 16 \left(\frac{\bar{\varepsilon} \|A\|_F^2}{\kappa'\sigma^2}\right)^2
\]
Lemma 6.7. Let $\alpha = \frac{\|A\|_F^2}{\kappa^2 \sigma^2}$. For some orthogonal projector $M$,

$$\|M - UU^T \hat{V}^T\|_F^2 \leq 16\alpha + O(\alpha^{3/2})$$

Proof. For ease of notation let $P_1$ be the orthogonal projector onto the first $\ell(\sigma(1+\kappa))$ singular vectors, $P_2 = UU^T$, and $P_3$ be the orthogonal projector onto the rest of the singular vectors. Let $Q = \hat{V}^T$. We are concerned with $P_2Q$.

Notice that all of the matrices discussed above are Hermitian and positive semidefinite. Further, $P_1 + P_2 + P_3 = I$, $\|(I - Q)P_1\|_F^2 \leq \alpha$ and $\|QP_3\|_F^2 \leq \alpha$ (the latter two following from Lemma 6.6). Then

$$P_2Q = Q - P_1 + P_1(I - Q) - P_3Q$$

$$\|P_2Q - (Q - P_1)\|_F \leq \|P_1(I - Q)\|_F + \|P_3Q\|_F \leq 2\sqrt{\alpha}$$

So now it is sufficient to show that $Q - P_1$ is close to a projector matrix.

Lemma 6.8. If a Hermitian $A$ satisfies $\|A^2 - A\|_F \leq \epsilon$, then $\|A - P\|_F \leq \epsilon + 4\epsilon^2$ for some orthogonal projector $P$.

We defer the proof of this to the appendix. The following short computation concludes the proof:

$$(Q - P_1)^2 - (Q - P_1) = Q - QP_1 - P_1Q + P_1 - Q = (I - Q)P_1 + P_1(I - Q)$$

$$\Rightarrow \|Q - P_1\|^2 - (Q - P_1)\|_F \leq 2\sqrt{\alpha}$$

Putting everything together, we have

$$\|D^* - A_{\alpha, \kappa'}\|_F^2 \leq 2\varepsilon(1 + \frac{1}{\kappa^2})\|A\|_F^2 + \sigma^2(1 + \kappa')^2\left(6\frac{\varepsilon\|A\|_F^2}{\kappa^2 \sigma^2} + O\left(\left(\frac{\varepsilon\|A\|_F^2}{\kappa' \sigma^2}\right)^{3/2}\right)\right)$$

$$\leq \left(2\varepsilon(1 + \frac{1}{\kappa^2}) + (1 + \kappa')^2\left(16\frac{\varepsilon}{\kappa^2} + O\left(\left(\frac{\varepsilon}{\kappa'}\right)^{3/2}\right)\right)\right)\|A\|_F^2$$

By taking $\varepsilon = \sigma^2\|A\|_F$, $0 < \kappa' \leq \frac{1}{2}$, and bounding very roughly, we get that the right-hand side is $O(\varepsilon^2\|A\|_F^2/\kappa'^2)$. Thus,

$$\|D^* - A_{\alpha, \kappa'}\|_F \lesssim \frac{\varepsilon\|A\|_F}{\kappa'}$$

giving us the result.

From the above, we can conclude that sampling from rows of $D^*$, where $\|D^* - A_{\alpha, \kappa'}\| \leq \varepsilon\|A\|_F$ will give good recommendations. For an informal explanation why, recall that we are taking $A$ to be our subsampled preference matrix $\hat{T}$. Thus, by Theorem 3.3, $D^*$ is close to $\hat{T}$, and by Theorem 3.1 this gives good recommendations. A rigorous proof is given in Section 6.3.
It’s not obvious how to gain information about $D^*$ just from its description. The following proposition shows that we can perform some basic sampling and querying operations, and that will be enough.

**Proposition 6.9.** The description of $\hat{V}$ allows samples from any $\hat{v}^{(i)}$ in $O(Kp \log n)$ expected queries, and queries for any particular entry in $O(p)$ queries.

**Proof.** As a reminder, $\hat{v}^{(i)} = S^T u^{(i)}/\sigma^{(i)}$, where $S \in \mathbb{R}^{p \times n}$ is a set of rows from $A$ specified by their corresponding indices, $u^{(i)} \in \mathbb{R}^p$ is a given unit vector, and $\sigma^{(i)}$ is a given scalar.

We can query to any particular entry easily: $\hat{V}_{ij} = \hat{v}^{(i)}_j = \frac{1}{\sigma^{(i)}} \sum_{k=1}^p A_{ik} u^{(i)}_k$, so this is just an expression with $2p$ values, all of which can be easily found from $A$ and the description.

For sampling, we can use Proposition 5.3 with constant $K$; we know that the sum of the squared 2-norms of the rows cannot be any more than $\|A\|_F$; the coefficients of the linear combination are at most 1, and the output linear combination has squared 2-norm at least $\sigma^2$, giving a constant of $K$ at worst. Rejection sampling uses queries of rows, which each take $\log n$ queries to the BST data structure.

6.2 Projection Sampling

At this point, we know from the above section that we have some description of a matrix that would give good recommendations if we could only sample from its rows. By 6.9 we know that this description allows for sampling and querying to a set of approximately orthogonal vectors (as shown by 6.3). We also know the row for the user we care about, $A_i$. Our goal is to produce a sample from $A_i\hat{V}\hat{V}^T$ (or, thinking of $A_i$ as a column vector, $\hat{V}\hat{V}^T A_i$); that is, we want to sample from the projection of a vector to some subspace, given the vector and an approximately orthonormal basis. We will call this task projection sampling. It turns out that we can do this, with the vector sampling tools we have developed.

**Theorem 6.10.** Suppose we are given $x \in \mathbb{R}^n$ in the BST data structure and $V \in \mathbb{R}^{n \times p}$ $\alpha$-approximately orthonormal vectors with query and sample access. (We know $\alpha$.) There is an algorithm that can output a sample from a distribution $O(\varepsilon + \alpha)$-close to $Z_{VV^T}x$ in

$$O\left(\frac{p^2 \log p \log \frac{1}{\delta} \|x\|^2}{\varepsilon^2 \|VV^T x\|^2}\right)$$

expected queries (and similar time complexity), assuming $\varepsilon \leq \frac{\|VV^T x\|}{\|x\|}$ for $\varepsilon$ constant.

**Proof.** The approach is simple: to sample from $VV^T x$, estimate $V^T x$, which we can do quickly since $V$ is tall, and then sample from a linear combination of the
Algorithm 2: ProjectionSampling

**Input:** $V \in \mathbb{R}^{n \times p}$ α-app. orthonormal vectors (query, sample access); $x \in \mathbb{R}^n$ (BST data structure); $\varepsilon$ parameter

**Output:** a random sample $s \in [n]$ approximately from $Z_{VV^T x}$

Let $est_0 = x$;

for $i$ from 0 to $c - 1$ do
  Use Proposition 5.2 with parameter $\gamma_i = \frac{\varepsilon\|est_i\|}{\sqrt{p}\|x\|}$ to estimate $\langle V(t), x \rangle$ for all $t \in [p]$;
  Let $est_{i+1}$ be the vector of estimates: $est_{i+1} = \{\langle V(t), x \rangle\}_{t \in [p]}$;
end

Use Lemma 5.7 to get a sample from $\sum (est_{c})_i v_i$ to output;

columns of $V$ according to the estimated $V^T x$. In Section 5.1 we described how to do all of these tasks; in the analysis we will piece them together.

Note that we only know $\|V(i)\|$ approximately (despite being able to query and sample from the vector), so we will have to be somewhat careful in our analysis.

To analyze the for loop, let $err_i = \|V^T x - est_i\| < \text{abs}(\|V^T x\| - \|est_i\|)$. Proposition 5.2 gives $\gamma_i\|x\||V(t)||/\sqrt{p}$ additive error for all $t \in [p]$, and thus a $\gamma_i\|x\|\max_t \|V(t)\|$ error additive error for $est_i$. That is,

$$\|est_{i+1} - V^T x\| \leq \gamma_i \|x\| \max_t \|V(t)\|$$

$$\leq \frac{\varepsilon\|est_i\|\|x\| \max_t \|V(t)\|}{\|x\|}$$

$$\leq \varepsilon\|est_i\|(1 + \alpha + O(\alpha^2))$$

$$err_{i+1} \leq (\|V^T x\| + err_i)(1 + \alpha + O(\alpha^2))$$

Solving this recurrence, we get that

$$err_c = (\varepsilon + O(\varepsilon^2))(1 + \alpha + O(\alpha^2))\|V^T x\| + \varepsilon^c(1 + \alpha + O(\alpha^2))^c\|x\|$$

From here, we can use rejection sampling to pull a sample from $V(est_c)$. We have all the tools we need to use Lemma 5.7 and get $(\alpha + O(\alpha^2))$-close in TV distance to $Z_{V(est_c)}$, which in turn has TV distance to $Z_{VV^T x}$ of

$$\frac{\varepsilon(1 + \alpha)\|V^T x\| + \varepsilon^c(1 + \alpha)\|x\|}{\|VV^T x\|} \approx \varepsilon(1 + 2\alpha) + \varepsilon^c(1 + \alpha)^c \frac{\|x\|}{\|VV^T x\|}$$

$$\approx \varepsilon(1 + \alpha)^c$$

by Lemma 5.1 and the assumption $\varepsilon^{c-1} \lesssim \|VV^T x\|/\|x\|$.

The runtime analysis is straightforward from the propositions; notice that $\delta$ only blows up by a poly($p$) factor. $\alpha$ only occurs in $(1 + \alpha)$-factors, and so doesn’t appear in the runtime.

\[24\]
6.3 Full Algorithm & Running Time

We give the full algorithm below, in Algorithm 3, which glues the two algorithms together and feeds them the correct parameters. We note that it’s possible to store the description $D^*$ given by ModFKV and use it for recommendations for multiple users; if $D^*$ is a good low-rank approximation (which it is with high probability) then correctness will follow for all runs of ProjectionSampling using that description.

Algorithm 3: Recommendations

**Input:** Matrix $A \in \mathbb{R}^{m \times n}$ in a data structure satisfying 4.2, rank parameter $k$, error parameter $\varepsilon > 0$, subsample parameter $p$, user $i \in [m]

**Output:** A sample $s \in [n]

Run ModFKV (1) with inputs $A$, $\sigma = \|A\|_F \sqrt{\varepsilon^2 p/8k}$, $\varepsilon$ to get a description $D^*$;

Run ProjectionSampling (2) with $V = \hat{V}$ simulated from $D^*$ as described in Proposition 6.9, $x = A_i$, and $\varepsilon$ error parameter to get a sample $s \in [n]$;

Output $s$;

**Correctness:** From Theorem 3.3 combined with 6.4 we know that with probability the description given by ModFKV would give good recommendations if we could sample from rows of the matrix corresponding to it. That is, for $D^*$ the matrix defined by the description,

$$\|T - D^*\|_F \leq \|T - \hat{T}_{\geq \sigma, \kappa}\|_F + \|\hat{T}_{\geq \sigma, \kappa} - D^*\|_F \leq 3(\rho + \varepsilon)\|T\|_F + O(\varepsilon)\|\hat{T}\|_F$$

$$\|\hat{T}\|_F = O(\|T\|_F)$$

by a simple Chernoff bound, so by Theorem 3.1 for $(\gamma, \zeta)$-typical users of $T$, there is a subset of those users $S'$ of size at least $(1 - \psi - \zeta)$ such that

$$\Pr_{i \sim u, j \sim S'}[(i, j) \text{ is bad}] \lesssim \frac{(\varepsilon + \rho)^2(1 + \varepsilon + \rho)^2}{(1 - \varepsilon - \rho)^2 (1/(1 + \gamma) - (\varepsilon + \rho)/\sqrt{\psi})^2 (1 - \psi - \zeta)}$$

By Proposition 6.9 we note that the requirements are satisfied to run ProjectionSampling (since $A_i$ is already in the correct data structure, $x$'s requirements are implicitly satisfied). The runtime and correctness bounds for ProjectionSampling given in Theorem 6.10 assert that we sample from a distribution close to the desired distribution for users satisfying $\varepsilon^c \lesssim \|A_i\|/\|VV^TA_i\|$. We will see below that nearly all typical users satisfy that inequality. Let $X_i$ be the output distribution of the algorithm for input $i$, and let $S'' \subset S'$ be the set of typical users that satisfy the inequality. Then

$$\Pr_{i \sim S''}[(i, j) \text{ is bad}] \lesssim \frac{(\varepsilon + \rho)^2(1 + \varepsilon + \rho)^2}{(1 - \varepsilon - \rho)^2 (1/(1 + \gamma) - (\varepsilon + \rho)/\sqrt{\psi})^2 (1 - \psi - \zeta)} + O(\varepsilon)$$
Proposition 6.11. $\|A_i\|/\|VV^T A_i\|$ is constant for most typical users (and further, $> 1/\varepsilon^c$ for a $1 - O(\varepsilon^c)$ fraction of typical users).

The proof for this is given in [10]; it is straightforward, and follows from the fact that

$$E_{i \in S'} \left[ \frac{\|A_i\|^2}{\|(A_{\geq \sigma, \kappa})_i\|^2} \right] \lesssim \frac{(1 + \varepsilon + \rho)^2}{(1 - \psi - \zeta) \left( \frac{1}{1 + \gamma} + \frac{\varepsilon + \rho}{\sqrt{\psi}} \right)^2}$$

For appropriate parameters this is a constant, and so the proposition follows from Markov’s inequality.

Runtime: To see that the runtime is $O(\text{polylog}(m, n) \text{ poly}(k, 1/\varepsilon, \log 1/\delta))$ is easy, since all of the algorithms and their associated subroutines that we use have those time bounds without blowup of input size.

This is subject to two caveats. First, $p$ needs to be constant; by $p$ can be subconstant for Theorem 3.3 to hold, but the time bound is lost for smaller $p$. This is typical for recommendation systems [7, 3, 10]. Second, $\|A_i\|/\|VV^T A_i\|$ needs to be constant. Proposition 6.11 states that this was the case for most users.

7 Further Questions

Since this algorithm is associated both with recommendation systems and quantum machine learning, there are two lines of questioning that are natural follow-ups.

First, we can continue to ask whether quantum machine learning algorithms have provably exponential speedups over classical algorithms. The author believes that an interesting aspect of this to investigate is the core data structures (that is, the QRAMs) used for these algorithms. Most quantum machine learning algorithms require construction of quantum superpositions corresponding to input vectors $|v\rangle$. Since exponential speedups will result in sublinear algorithms, it’s necessary for this construction to be very fast. The data structure given by Kerenidis and Prakash to do this is also very powerful classically (as seen by Proposition 4.2), allowing a sublinear classical algorithm. Any attempt to prove speedups for QML algorithms will have to contend with these data structure issues, making it a good line of attack.

Second, the recommendation system we give is asymptotically polylogarithmic in $m$ and $n$, but there are several aspects of this algorithm that make direct application infeasible in practice. First, the model assumptions are somewhat constrictive, especially the assumption that the subsampling probability is constant. Second, the exponents and constant factors are fairly large (mostly as a result of the constants in [8]). It seems reasonable that both of these issues can be mitigated with less roundabout analysis combined with more sophisticated techniques.
Acknowledgments

Thanks to Prof. Scott Aaronson for advising me and introducing me to this problem. His mentorship and help were integral to this work, as well as to my growth and abilities as a CS researcher. Thanks also to Daniel Liang for providing frequent, useful discussions and for reading through a draft of this document. Quite a few of the insights in this document were generated during discussions with him.

Thanks to Patrick Rall, Sujit Rao, and anybody else that I had enlightening conversations with during the research process. Thanks to Prabhat Nagarajan for the continuing support.

References

A Deferred Proofs

Proposition (Statement of Proposition 6.3). The output singular vectors \( \hat{v}_1, \ldots, \hat{v}_k \) are \( \varepsilon^4 \)-approximately orthonormal. Further, \( \| \hat{v}_i \|^2 \geq 1 \).

Proof. For \( i \neq j \), we have as follows (using notation from [8])

\[
|\hat{v}_i^T \hat{v}_j| = \frac{|u_i S^T u_j|}{\|W^T u_i\| \|W^T u_j\|} \leq \frac{|u_i S^T u_j|}{\frac{\theta \|S\|_F}{\gamma \|W\|_F}} \leq \frac{\theta}{\gamma}
\]

For \( i = j \), we also get a similar bound of \( 0 \leq \| \hat{v}_i \|^2 - 1 \lesssim \frac{\theta}{\gamma} \). (To see that \( \| \hat{v}_i \|^2 \geq 1 \), just notice that \( \|S^T u_i\| \geq \|W^T u_i\| \).)

In the original algorithm’s filter step, \( \gamma \) is chosen to be \( \bar{\varepsilon}/8k \). However, in ModFKV, as noted in its description, the bound for \( \gamma \) still holds for values as large as \( 1/K \) without altering the behavior of the algorithm. Thus, the bound is actually much better than \( \theta/\gamma = O(\bar{\varepsilon}) \), it is \( O(\bar{\varepsilon}^2) \). We need to lose an additional factor of \( k \); since \( \bar{\varepsilon} = \varepsilon^2/\sqrt{K} \), we account for this factor. \( \square \)

Lemma (Statement of Lemma 6.8). If a Hermitian \( A \) satisfies \( \| A^2 - A \|_F \leq \varepsilon \), then \( \| A - P \|_F \leq \varepsilon + 4\varepsilon^2 \) for some orthogonal projector \( P \).

Proof of Lemma 6.8. Use the fact that Hermitian matrices are normal, so \( A = UTU^T \) for unitary \( U \) and diagonal matrix \( \Gamma \), and

\[
A^2 - A = U(\Gamma^2 - \Gamma)U^T \implies \| \Gamma^2 - \Gamma \|_F \leq \varepsilon
\]

From here, consider the entries \( \gamma_i \) of \( \Gamma \), satisfying \( \gamma_i^2 - \gamma_i = c_i \) and \( \sum c_i^2 = \varepsilon^2 \). Thus, \( \gamma = (1 + \sqrt{1 + 4c_i^2})/2 \) which is at most \( c_i + 4c_i^2/2 \) off from \( 0.5 \pm 0.5 \) (aka \( \{0, 1\} \)), using that \( 1 - x/2 - x^2/2 \leq \sqrt{1 - x} \leq 1 - x/2 \). Finally, this means that \( \Gamma \) is off from having only 0s and 1s on the diagonal by \( \sqrt{\sum (c_i + 4c_i^2)^2} \leq \varepsilon + 4\varepsilon^2 \) in Frobenius norm.

If \( \Gamma \) had only 0s and 1s on the diagonal, the resulting \( UTU^T \) would be an orthogonal projector. \( \square \)
Lemma (Statement of Lemma 6.6). The system of equations:
\[
\sum_{i=1}^{N} \sigma_i^2 \|a_i\|^2 \leq \sum_{k+1}^{N} \sigma_i^2 + \bar{\varepsilon} \|A\|_F^2 \quad \sum_{i=1}^{N} \|a_i\|^2 \geq N - k
\]
\[
\|a_i\|^2 \in [0, 1] \quad \sigma_i^2 \text{ are nonincreasing}
\]
imply the following, for \(0 < \kappa \leq \frac{1}{2}\):
\[
\sum_{1}^{\ell(\sigma(1+\kappa))} \sigma_i^2 \|a_i\|^2 \leq \bar{\varepsilon} \left(1 + \frac{1}{\kappa}\right) \|A\|_F^2 \quad \sum_{\ell(\sigma(1+\kappa))+1}^{N} \sigma_i^2 (1 - \|a_i\|^2) \leq \bar{\varepsilon} \left(1 + \frac{1}{\kappa}\right) \|A\|_F^2
\]
\[
\sum_{1}^{\ell(\sigma(1+\kappa))} \|a_i\|^2 \leq \frac{\bar{\varepsilon} \|A\|_F^2}{\kappa \sigma^2} \quad \sum_{\ell(\sigma(1+\kappa))+1}^{N} (1 - \|a_i\|^2) \leq \frac{\bar{\varepsilon} \|A\|_F^2}{\kappa \sigma^2}
\]

Proof. We are just proving straightforward bounds on a linear system; we include the computations for completeness.

Note that \(k = \ell(\sigma)\). The slack in the inequality is always maximized when the second equation is an equality and the weight of \(\|a_i\|^2\) is concentrated on the large-index (small-value) entries. For example, as written in the problem statement, the choice of \(\|a_i\|^2\) maximizing slack is the vector \(1 \geq \ell(\sigma)\); that is,
\[
\|a_i\|^2 = \mathbb{1}_{i \geq \ell(\sigma)} = \begin{cases} 1 & i \geq \ell(\sigma) \\ 0 & \text{otherwise} \end{cases}
\]

We will prove the bounds by adding constraints, and showing when the resulting system is feasible.

Consider adding the constraint \(C = \sum_{1}^{\ell(\sigma(1+\kappa))} \sigma_i^2 \|a_i\|^2\) to the system of equations. We want to determine whether the modified system is still feasible; we can do this by trying the values that maximize slack.

This occurs when weight is on the largest possible indices: when \(\|a_{\ell(\sigma(1+\kappa))}\|^2 = C/\sigma_i^2(\sigma(1+\kappa))\|, \|a_{\ell(\sigma)+1}\|^2 = 1 - C/\sigma_i^2(\sigma(1+\kappa))\|, \text{ and all other } \|a_i\|^2\) are \(\mathbb{1}_{i \geq \ell(\sigma)}\). Notice that \(\|a_{\ell(\sigma(1+\kappa))}\|^2\) could be larger than one and \(\|a_{\ell(\sigma)+1}\|\) could be negative, so we break constraints, but if there is no feasible solution even when relaxing those two constraints, there is certainly no solution to the non-relaxed system.

Thus, we check feasibility (by construction the second equation is satisfied):
\[
C + \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 - \sigma_{\ell(\sigma)+1}^2 (C/\sigma_i^2(\sigma(1+\kappa))) \leq \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 + \bar{\varepsilon} \|A\|_F^2
\]
\[
C \left(1 - \sigma_{\ell(\sigma)+1}^2/\sigma_i^2(\sigma(1+\kappa))\right) \leq \bar{\varepsilon} \|A\|_F^2
\]
\[
C \left(1 - \frac{1}{(1 + \kappa)^2}\right) \leq \bar{\varepsilon} \|A\|_F^2
\]
This gives the bound on $C$.

Analogously, consider the restriction $D = \sum_{i=1}^{N} \sigma_i^2 (1 - \|a_i\|^2)$. Then at best, \( \|a_{i(\sigma(1-\kappa))}\| = 1 - \frac{D}{\sigma_i^2(\sigma(1-\kappa))} \) and the rest follow $\mathbb{1}_{i \geq \ell(\sigma)}$.

\[
\sigma_i^2(\sigma(\sigma(1-\kappa)) + \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 - D \leq \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 + \bar{\varepsilon}\|A\|_F^2,
\]

\[
D \left( \frac{\sigma_i^2(\sigma(\sigma(1-\kappa)) + 1)}{\sigma_i^2(\sigma(1-\kappa))} - 1 \right) \leq \bar{\varepsilon}\|A\|_F^2.
\]

When $c = \sum_{i=1}^{\ell(\sigma(1+\kappa))} \|a_i\|^2$, we modify from $\mathbb{1}_{\geq \ell(\sigma)}$ by taking $\|a_{\ell(\sigma(1+\kappa))}\| = c$ and $\|a_{\ell(\sigma)+1}\| = 1 - c$. Then

\[
c\sigma_i^2(\sigma(1+\kappa)) + \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 - c\sigma_i^2(\sigma(1+\kappa)) + 1 \leq \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 + \bar{\varepsilon}\|A\|_F^2,
\]

\[
c \left( 1 - \frac{\sigma_i^2(\sigma(1+\kappa))}{\sigma_i^2(\sigma(1+\kappa)) + 1} \right) \leq \bar{\varepsilon} \frac{\|A\|_F^2}{\sigma_i^2(\sigma(1+\kappa))},
\]

\[
c \left( 1 - \frac{1}{(1+\kappa)^2} \right) \leq \bar{\varepsilon} \frac{\|A\|_F^2}{\sigma^2(1+\kappa)^2}.
\]

Finally, when $d = \sum_{\ell(\sigma(1-\kappa))}^{N} (1 - \|a_i\|^2)$, we modify from $\mathbb{1}_{\geq \ell(\sigma)}$ by taking $\|a_{\ell(\sigma)}\|^2 = D$ and $\|a_{\ell(\sigma)+1}\| = 1 - D$. Then

\[
d\sigma_i^2(\sigma(1-\kappa)) + \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 - d\sigma_i^2(\sigma(1-\kappa)) + 1 \leq \sum_{\ell(\sigma)+1}^{N} \sigma_i^2 + \bar{\varepsilon}\|A\|_F^2
\]

\[
d \left( 1 - \frac{\sigma_i^2(\sigma(1-\kappa)) + 1}{\sigma_i^2(\sigma(1-\kappa))} \right) \leq \bar{\varepsilon} \frac{\|A\|_F^2}{\sigma_i^2(\sigma(1-\kappa))},
\]

\[
d \left( 1 - (1-\kappa)^2 \right) \leq \bar{\varepsilon} \frac{\|A\|_F^2}{\sigma^2(1+\kappa)^2}.
\]

Altogether, these imply the following bounds:

\[
\sum_{i=1}^{\ell(\sigma(1+\kappa))} \sigma_i^2 \|a_i\|^2 \leq \bar{\varepsilon}(1+\kappa)^2 \frac{\|A\|_F^2}{2\kappa^2}, \quad \sum_{\ell(\sigma(1-\kappa))}^{N} \sigma_i^2 (1 - \|a_i\|^2) \leq \bar{\varepsilon}(1+\kappa)^2 \frac{\|A\|_F^2}{2\kappa^2},
\]

\[
\sum_{i=1}^{\ell(\sigma(1+\kappa))} \|a_i\|^2 \leq \bar{\varepsilon} \frac{\|A\|_F^2}{(2\kappa^2 + \kappa^2)\sigma^2}, \quad \sum_{\ell(\sigma(1-\kappa))}^{N} (1 - \|a_i\|^2) \leq \bar{\varepsilon} \frac{\|A\|_F^2}{(2\kappa^2 - \kappa^2)\sigma^2}.
\]

We get the bounds in the statement by simplifying the above. □