A quantum-inspired classical algorithm for recommendation systems

Ewin Tang

July 13, 2018

Abstract

A recommendation system suggests products to users based on data about user preferences. It is typically modeled by a problem of completing an $m \times n$ matrix of small rank k. We give the first classical algorithm to produce a recommendation in $O(\text{poly}(k)\,\text{polylog}(m,n))$ time, which is an exponential improvement on previous algorithms that run in time linear in m and n. Our strategy is inspired by a quantum algorithm by Kerenidis and Prakash: like the quantum algorithm, instead of reconstructing a user's full list of preferences, we only seek a randomized sample from the user's preferences. Our main result is an algorithm that samples high-weight entries from a low-rank approximation of the input matrix in time independent of m and n, given natural sampling assumptions on that input matrix. As a consequence, we show that Kerenidis and Prakash's quantum machine learning (QML) algorithm, one of the strongest candidates for provably exponential speedups in QML, does not in fact give an exponential speedup over classical algorithms.

Contents

1 Introduction

1.1 Recommendation Systems

Given 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 use the following model, first introduced in 2001 by Kumar et al. [16] and refined further by Azar et al. [4] and Drineas et al. [9] 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 to the corresponding users. However, we are typically given only a small subsample of entries of T (which we learn when a user purchases or otherwise interacts with a product). Then, finding recommendations for user i is equivalent to finding large entries of the ith row of T given such a subsample.

Obviously, without any restrictions on what T looks like, this problem is ill-posed. We make this problem tractable through the standard assumption that T is close to a matrix of small rank k (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. We can determine the classes a user lies in, and use information about those classes to give a likely-to-be-good recommendation.

Our main result is a classical recommendation system algorithm whose runtime is exponentially faster than the best-known in the literature. 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 ith row of a low-rank approximation of A in $O(\text{poly}(k)\,\text{polylog}(m,n))$ time; given standard assumptions on A and T, this sample is likely to be a good recommendation.

With the low-rank assumption that k is small, our algorithm gives an exponential improvement over the current literature. Algorithms for recommendation systems typically attempt to reconstruct full rows of T [16, 9, 3], which must take $\Omega(n)$ time. We sidestep this bottleneck by only giving a sample of good recommendations, instead of a complete list. Knowing a constant number of recommendations for a user is almost always sufficient in practice, so this algorithm does not lose any utility. Further, we are able to do this without unnatural assumptions: our assumptions are exactly the same as those of Kerenidis and Prakash for their quantum recommendation system algorithm [13].

Comparison with related recommendation systems. The literature on theoretical recommendation systems is fairly scant, because such algorithms quickly run up against the limits of their model, despite them being somewhat far from the results seen in practice. We hope that this algorithm, by presenting a novel sampling-based technique with a much faster asymptotic runtime, helps to improve the connection between theory and practice and provides an avenue for further research.

Our model is most similar to the model given in 2002 by Drineas et al. [9]. However, that algorithm's main goal is minimizing the number of user preferences necessary to generate good recommendations; we discuss in Appendix B how to adapt our algorithm to that model to get similar results. Other approaches include combinatorial techniques [16, 3] and the use of mixture models [14].

Although our model assumptions are not directly comparable to the assumptions for these recommendation systems, roughly speaking, they are at least of similar quality. Our assumptions on the preference matrix are weak, but this comes with the tradeoff of stronger assumptions on our knowledge of the preference matrix. Further, we consider our matrix sampling assumptions to be weak in this applied context, where choice of data structure is reasonable to consider. Such assumptions also bear resemblance to assumptions used in previous recommendation systems (Section 4 of [9]).

Finally, we acknowledge the substantial body of work on practical recommendation systems, which applies a combination of theoretical methods and implementations of intuitive heuristics. Using the terminology from that setting, our algorithm is SVD-based and uses a latent factor model. Work on the Netflix Prize is a useful reference for practical recommendation systems: see Bennett et al. [7] for a broad overview of techniques, Koren et al. [15] for a high-level exposition of the SVD technique, and Bell et al. [5] for more technical details.

1.2 Quantum Machine Learning

Our algorithm stems from failed efforts to prove that Kerenidis and Prakash's quantum recommendation system algorithm [13] achieves an exponential speedup over any classical

algorithm. Such a result would be interesting because Kerenidis and Prakash's algorithm is a quantum machine learning (QML) algorithm.

QML spawned from Harrow, Hassidim, and Lloyd's 2008 quantum algorithm for solving linear systems [11]. 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, in part because of caveats exposited by Aaronson [1], it is not clear whether any known QML algorithm gives a new exponential speedup over classical algorithms, for practically relevant instances of a machine learning problem. Kerenidis and Prakash's work was notable for addressing all of these caveats, giving a complete quantum algorithm that can be compared directly to classical algorithms [19].

When Kerenidis and Prakash's 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. We give a classical algorithm whose runtime is only polynomially greater than that of Kerenidis and Prakash's (up to small loss in approximation), thus answering this question. Although this removes one of the most convincing examples we have of exponential speedups for machine learning problems, our algorithm can also be seen as a success for QML, because the framing and analysis required to apply QML techniques also served to inform and inspire the new algorithm we present here.

1.3 Algorithm Sketch

Outside of the recommendation systems context, the quantum algorithm just computes low-rank matrix approximations: its output is a quantum state representing a row of a low-rank approximation of the input matrix A. Measuring this state samples an entry from that row proportional to its magnitude. Kerenidis and Prakash then bring in the recommendation systems model in the analysis to show that this sampled entry is usually a good recommendation. We will follow the same strategy for our algorithm.

We first state the main result, our classical algorithm, informally. The formal statement can be found in Section 7.1.

Theorem (1, informal)**.** *Suppose we are given as input a matrix A supporting query and ℓ^2-norm sampling operations, a row/user $i \in [m]$, a singular value threshold σ, and sufficiently small $\varepsilon > 0$. There is a classical algorithm whose output distribution is $O(\varepsilon)$-close in total variation distance to the distribution given by ℓ^2-norm sampling from the ith row of a low-rank approximation D of A in query and time complexity*

$$O\left(\text{poly}\left(\frac{\|A\|_F}{\sigma}, \frac{1}{\varepsilon}, \frac{\|A_i\|}{\|D_i\|}, \log\frac{1}{\delta}\right)\right)$$

where δ is the probability of failure.

This runtime is *independent* of m and n. To implement the needed sampling operations, we can use the data structure described in Section 4, which adds an additional $O(\log^2(mn))$

factor of overhead. This gives a time complexity of

$$\tilde{O}\left(\max\left\{\frac{\|A\|^{33}}{\sigma^{33}\varepsilon^{18}}, \frac{\|A\|^{24}}{\sigma^{24}\varepsilon^{24}}\right\}\log^2(mn)\frac{\|A_i\|^2}{\|D_i\|^2}\right).$$

This is a large slowdown versus the quantum algorithm in the $\|A\|/\sigma$ and $1/\varepsilon$ exponent. However, we suspect that these exponents can be improved significantly through standard techniques.

The only difference between Theorem 1 and its quantum equivalent in [13] is that the quantum algorithm has no ε approximation factors (so $\varepsilon = 0$ and $1/\varepsilon$ does not appear in the runtime). Thus, we can say that our algorithm performs just as well, up to polynomial slowdown and ε approximation factors. Further, these ε's don't affect the classical algorithm in practice:

Theorem (2)**.** *Applying Theorem 1 to the recommendation systems model (described in Section 5) with the quantum state preparation data structure (Section 4) achieves the same bounds of quality of recommendations as the quantum algorithm in [13], up to constant factors and for sufficiently small ε.*

To prove Theorem 1 (see Section 7.1), we give a classical algorithm (Algorithm 4) that can sample a high-value entry from a given row of a low-rank approximation of a given matrix. We do this by combining a variety of techniques, all relying on sampling access to relevant input vectors. The main restriction to keep in mind is that we need to perform linear algebra operations without incurring the cost of reading a full row or column.

We begin with our input matrix A in the same data structure that Kerenidis and Prakash use (see Section 4); this lightweight data structure supports powerful ℓ^2-norm sampling operations. Using these operations, we apply a subsampling strategy (called ModFKV, see Section 6.3) based on Frieze, Kannan, and Vempala's 2004 algorithm [10] to find a low-rank approximation of A. The algorithm doesn't have enough time to output the matrix in full; instead, it outputs a succinct description of the matrix. Essentially, this description is a set of orthonormal approximate singular vectors $\hat{V}$, where the corresponding low-rank approximation D is $A\hat{V}\hat{V}^T$, the projection of the rows of the input matrix onto the low-dimensional subspace spanned by $\hat{V}$. By Proposition 6.14, we (surprisingly) can both sample from and query to these singular vectors quickly. While showing fast querying is straightforward, fast sampling is nonobvious. Sampling from a vector in $\hat{V}$ reduces to the problem of sampling from a linear combination of vectors, given the ability to sample and query to each individual vector. Rejection sampling is a natural solution to this problem (Proposition 6.4).

So, we have the desired low-rank approximation D given in the implicit form $A\hat{V}\hat{V}^T$. To sample from a row of D given its description (that is, sample from $A_i\hat{V}\hat{V}^T$), we first estimate $A_i\hat{V}$. This amounts to estimating dot products and can be done with sampling access to A_i by Proposition 6.2. Then, we use this estimate to give an approximate sample from $A_i\hat{V}\hat{V}^T$, which is a linear combination of vectors from $\hat{V}$. Since we can sample from these vectors, we can produce this sample from $A_i\hat{V}\hat{V}^T$ with the same rejection sampling technique (Lemma 6.8). This sample is the desired output.

This description omits one major subtlety: $\hat{V}$ is not orthonormal, but approximately orthonormal. Though we take care to deal with this caveat, it has no major effect on the algorithm; Section 6.2 details the techniques we use to show this. This completes the broad overview of the algorithm. The only sizable step in the analysis is showing that ModFKV gives the kind of bounds that we want for our recommendation system (Theorem 6.12).

To prove Theorem 2 and show that the quality bounds on the recommendations are the same (see Section 7.2), we just follow Kerenidis and Prakash's steps to apply the model assumptions and theorems (Section 5) in a straightforward manner.

2 Discussion and Further Questions

2.1 Discussion

This section is devoted to the high-level implications of our work in the area of quantum machine learning. It is not essential to understanding the classical algorithm.

How the quantum recommendation systems algorithm works. The main technique used in Kerenidis and Prakash's quantum recommendation system algorithm [13] is quantum phase estimation. This procedure is used to implicitly estimate singular values and locate singular vectors of the given preference matrix. A quantum projection procedure then uses this implicit information to project a quantum state corresponding to a user's known preferences to a state with a user's potential recommendations. Measuring this resulting state then yields a possible recommendation.

One might expect this algorithm to perform exponentially faster than any classical algorithm. In Section 1.2 we discussed reasons to believe such a claim for this algorithm compared to other QML algorithms. We can find such reasons just from considering the algorithm in isolation, though, too. The intuition behind believing such claims is not that the steps (singular value estimation, computing projections) are particularly difficult computationally. Rather, it's simply hard to believe that any of these steps can be done without reading the full input. After all, a significant portion of the theory of low-rank matrix completion and approximation only asks for time complexity linear in input-sparsity or sublinear with some number of passes through the data. By comparison to these types of results, what the quantum algorithm achieves (query complexity polylogarithmic in the input size) is impressive.

State preparation: the quantum algorithm's assumption. To see why the classical algorithm we present is possible, we need to consider the technique Kerenidis and Prakash use to construct their relevant quantum states.

Kerenidis and Prakash's algorithm is one of many QML algorithms [11, 17, 18, 12] that require quantum state preparation assumptions, which state that given an input vector v, one can quickly form a corresponding quantum state $|v\rangle$. To achieve the desired runtime in practice, an implementation would replace this assumption with either a procedure to prepare

a state from an arbitrary input vector (where the cost of preparation could be amortized over multiple runs of the algorithm) or a specification of input vectors for which quantum state preparation is easy. Usually QML algorithms abstract away these implementation details, assuming a number of the desired quantum states are already prepared. The quantum recommendation systems algorithm is unique in that it explicitly comes with a data structure to prepare its states (introduced in Section 4).

These state preparation assumptions are nontrivial: even given ability to query entries of a vector in superposition, preparing states corresponding to arbitrary n-length input vectors is known to take $\Omega(\sqrt{n})$ time (a corollary of quantum search lower bounds by Bennett, Bernstein, Brassard, and Vazirani [6]). Thus, the data structure to quickly prepare quantum states is essential for the recommendation systems algorithm to achieve query complexity polylogarithmic in input size.

How a classical algorithm can perform as well as the quantum algorithm. The classical algorithm we present cannot be compared to classical low-rank matrix completion and approximation results requiring linear time. Rather, the key insight is that the data structure used to satisfy state preparation assumptions can also satisfy ℓ^2-norm sampling assumptions (see Section 4).

So, a classical algorithm whose goal is to "match" the quantum algorithm can exploit these assumptions, which are known in the classical ML community to be quite strong (as seen in Frieze, Kannan, and Vempala's paper [10]). From there, performing an implicit projection of a vector onto a subspace to get a sample recommendation is not outside the realm of feasibility. Our algorithm just puts the two pieces together.

The importance of ℓ^2-norm sampling. In an imprecise sense, our algorithm has replaced state preparation assumptions with ℓ^2-norm sampling assumptions. In this particular case, while quantum superpositions served to represent data implicitly that would take linear time to write out, this need can be served just as well with probability distributions and subsamples of larger pieces of data.

The correspondence between ℓ^2-norm sampling assumptions and state preparation assumptions makes sense. While the former sidesteps the obvious search problems inherent in linear algebra tasks by pinpointing portions of vectors or matrices with the most weight, the latter sidesteps such search problems by allowing for quantum states that are implicitly aware of such weight distributions.

In a wealth of ways, the ℓ^2-norm sampling assumptions are more reasonable: they can be satisfied classically and they are already well-studied. Further, as evidenced by this classical algorithm, such a model is likely more appropriate to reveal speedups (or the lack thereof) given by QML algorithms. We suspect that this connection revealed by the state preparation data structure is somewhat deep, and cannot be fixed by simply finding a state preparation data structure without sampling power.

Thus, we argue for the following guideline: *when QML algorithms are compared to classical ML algorithms in the context of finding speedups, any state preparation assumptions in the QML model should be matched with ℓ^2-norm sampling assumptions in the classical ML model.*

2.2 Further Questions

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

First, we can continue to ask whether any quantum machine learning algorithms have provably exponential speedups over classical algorithms. We believe that a potentially enlightening approach is to investigate how state preparation assumptions can be satisfied, and whether they are in some way comparable to classical sampling assumptions. After all, we find it unlikely that a quantum exponential speedup can be reinstated just with a better state preparation data structure. However, we are unaware of any research in this area in particular, which could formalize a possible connection between QML algorithms with state preparation assumptions and classical ML algorithms with sampling assumptions.

Second, while the recommendation system algorithm we give is asymptotically exponentially faster than previous algorithms, there are several aspects of this algorithm that make direct application infeasible in practice. First, the model assumptions are somewhat constrictive. It is unclear whether the algorithm still performs well when such assumptions are not satisfied. Second, the exponents and constant factors are large (mostly as a result of using Frieze, Kannan, and Vempala's algorithm [10]). We believe that the "true" exponents are much smaller, but our technique is fairly roundabout and likely compounds exponents unnecessarily. These issues could be addressed with more straightforward analysis combined with more sophisticated techniques.

3 Definitions

Throughout, we use the following notation. $[n] := \{1, \ldots, n\}$. $f \lesssim g$ denotes the ordering $f = O(g)$ (and correspondingly for $\gtrsim$ and $\eqsim$). For a matrix A, A_i and $A^{(i)}$ will refer to the ith row and column, respectively. $\|A\|_F$ and $\|A\|_2$ will refer to Frobenius and spectral norm, respectively. Norm of a vector v, denoted $\|v\|$, will always refer to ℓ^2-norm. The absolute value of $x \in \mathbb{R}$ will be denoted $|x|$. Occasionally, matrix and vector 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 always refer to some form of perturbation or 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 matrices with columns $\{u_i\}_{i \in [m]}$ and $\{v_i\}_{i \in [n]}$, the left and right singular vectors, respectively. $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal with $\sigma_i := \Sigma_{ii}$ and the σ_i nonincreasing and nonnegative.

We will use the function ℓ to indicate splitting the singular vectors along a singular value:

$$\ell(\lambda) := \max\{i \mid \sigma_i \geq \lambda\}.$$

For example, σ_1 through $\sigma_{\ell(\lambda)}$ gives all of the singular values that are at least λ. This notation suppresses ℓ's dependence on σ_i, but it will always be clear from context.

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

3.1 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.

$$\begin{aligned} A_k &:= A\Pi_k & \operatorname{im}\Pi_k &:= \operatorname{span}\{v_i \mid i \in [k]\} \\ A_{\geq\sigma} &:= A\Pi_{\geq\sigma} & \operatorname{im}\Pi_{\geq\sigma} &:= \operatorname{span}\{v_i \mid i \in [\ell(\sigma)]\} \end{aligned}$$

A_k and $A_{\geq\sigma}$ correspond to the standard notions of low-rank approximations of A. Thus, $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$ and is a rank k matrix minimizing the Frobenius norm distance from A. Similarly, $A_{\geq\sigma}$ is just A_t for $t = \ell(\sigma)$. Notice that $\operatorname{rank} A_{\geq\frac{\|A\|_F}{\sqrt{\lambda}}} \leq \lambda$.

We will need to relax this notion for our purposes, and introduce error $\kappa \in [0, 1]$:

$$\begin{aligned} A_{\geq\sigma,\kappa} := A\Pi_{\geq\sigma,\kappa} = A(\Pi_{\geq\sigma} + \Pi_E) \text{ for some } \Pi_E \text{ satisfying} \\ \operatorname{im}\Pi_E \text{ is a subspace of } \operatorname{span}\{v_i \mid \ell(\sigma) < i \leq \ell(\sigma(1-\kappa))\}. \end{aligned}$$

In words, $A_{\geq\sigma,\kappa}$ is some projection of A onto its singular vectors whose values are at least σ, perhaps along with some vectors in the subspace spanned by the singular vectors associated to singular values between $\sigma(1-\kappa)$ and σ.[1] Such a form of error could arise from having some κ-like error in estimating the singular values used to compute a low-rank matrix approximation. Thus, we think of Π_E as an error term, and $A_{\geq\sigma,\kappa}$ as a class of matrices with sufficiently small error.

κ should be thought of as constant (1/3 will be the eventual value), and σ should be thought of as very large (say, a constant multiple of $\|A\|_F$), so $A_{\geq\sigma,\kappa}$ always has low rank.

It will also be useful to consider a version with two-sided error; we can do this with a simple transform of parameters:

$$\begin{aligned} A_{\sigma,\kappa} := A_{\geq\sigma(1+\kappa),\frac{2\kappa}{1+\kappa}} = A(\Pi_{\geq\sigma(1+\kappa)} + \Pi_E) \text{ for some } \Pi_E \text{ satisfying} \\ \operatorname{im}\Pi_E \leq \operatorname{span}\{v_i \mid \ell(\sigma(1+\kappa)) < i \leq \ell(\sigma(1-\kappa))\}. \end{aligned}$$

Since $0 < \kappa \leq 1$, this only perturbs parameters by constant factors.

[1] This error is essentially equivalent to the more familiar property $\Pi_{\geq\sigma} \preceq \Pi_{\geq\sigma,\kappa} \preceq \Pi_{\geq\sigma(1-\kappa)}$, where $\preceq$ is the Loewner order, since that error is contained in the convex hull of the error we define. However, this is the type of error that Kerenidis and Prakash achieve, so we will stick with this more unorthodox choice for consistency.

3.2 Sampling

For a nonzero vector $x \in \mathbb{R}^n$, we denote by $\mathcal{D}_x$ the distribution over $[n]$ whose probability density function is

$$\mathcal{D}_x(i) = \frac{x_i^2}{\|x\|^2}$$

We will call a sample from $\mathcal{D}_x$ a sample from x.

We make two basic observations. First, $\mathcal{D}_x$ is the distribution resulting from measuring the quantum state $|x\rangle$ in the computational basis (see [13] for details). Second, sampling access to $\mathcal{D}_x$ makes easy some tasks that are hard given just query access to x. For example, while finding a hidden large entry of $x \in \mathbb{R}^n$ takes $\Omega(n)$ queries with just query access, it takes a constant number of samples with query and sample access.

In all situations, sampling access will be present in addition to query access, and accordingly, we will conflate samples $i \sim \mathcal{D}_x$ with the corresponding entries x_i. Note that knowledge of $\|x\|$ is also relevant and useful in this sampling context, since it allows for computing probabilities from $\mathcal{D}_x$ and yet is hard to compute even with query and sampling access to x.

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

$$\|P, Q\|_{TV} := \frac{1}{2} \sum_{x \in X} \Big| P(x) - Q(x) \Big|.$$

For a set S, we denote pulling an $s \in S$ uniformly at random by $s \sim_u S$. We will continue to conflate a distribution with its density function.

4 Data Structure

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

In the recommendation systems context, entries correspond to user-product interactions, so we might expect that the input matrix $A \in \mathbb{R}^{m \times n}$ is given as an unordered stream of entries (i, j, A_{ij}). However, if the entries are given in such an unprocessed format, then clearly linear time is required even to parse the input 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 because of the time needed to locate any nonzero entries.

To avoid these issues, we will instead consider our input stored in a low-overhead data structure, the same one used by Kerenidis and Prakash [13].

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 $\mathcal{D}_v$ in $O(\log^2 n)$ time.

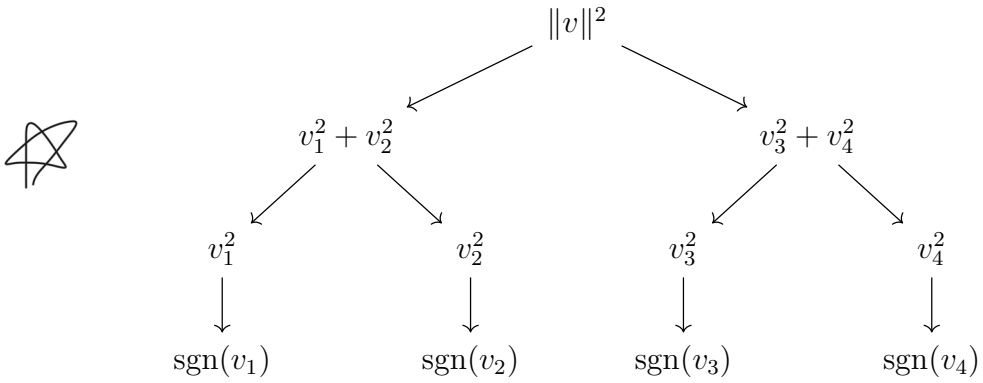

Figure 1: Binary search tree data structure for $v \in \mathbb{R}^4$. The leaf nodes store v_i (or equivalently, a weight v_i^2 and the sign of v_i), and the weight of an interior node is just the sum of the weights of its children. Updates can be done quickly by updating all of the nodes above a particular leaf. Sampling from $\mathcal{D}_v$ can be done by starting from the top of the tree and randomly recursing on a child, with probability proportional to its weight. 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 ith row of A. Let $\tilde{A} \in \mathbb{R}^m$ be a vector whose ith entry is $\|A_i\|$.*

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 $\mathcal{D}_{\tilde{A}}$ and $\mathcal{D}_{A_i}$ 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 structures are the entries of $\tilde{A}$, and a BST is constructed on top, using these as inputs. This has all of the desired properties, and in fact, is the data structure Kerenidis and Prakash use to prepare arbitrary quantum states (Theorem A.1 in [13]). Thus, our algorithm can operate on the same input, although any data structure supporting the operations detailed in Proposition 4.2 will also suffice.

This data structure and its operations are not as ad hoc as they might appear. The operations

listed above appear in other work as a standard way to endow a matrix with ℓ^2-norm sampling assumptions (see [9] and [10]).

5 Model Assumptions

We now go through the relevant assumptions necessary for our recommendation system model. Although these don't directly impact the algorithm for Theorem 1, they motivate the goals of that algorithm and come into play later for the recommendation system bounds of Theorem 2. As mentioned above, these are the same assumptions as those in [13]: an exposition of these assumptions is also given there.

5.1 Preference Matrix

Recall that given m users and n products, the preference matrix $T \in \mathbb{R}^{m \times n}$ contains the complete information on whether user i likes product j in T_{ij}.

For ease of exposition, we will assume the input data is binary:

Definition. If user i likes product j, then $T_{ij} = 1$. If not, $T_{ij} = 0$.

We can form such a preference matrix from generic data about recommendations, simply by condensing information down to the binary question of whether a product is a good recommendation or not.[2]

***T* is close to a low-rank matrix.** 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 standard 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). See Drineas et al. [9] and Kerenidis and Prakash [13] for this assumption in recommendation systems, Azar et al. [4] for more general contexts, and Koren et al. [15] for its manifestations in practice.

The low-rank assumption gives T structure that is relatively robust; that is, only given a small number of entries, T can be reconstructed fairly well.

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 [13]). However, this bound considers "matrix-wide" recommendations. We would like to give a bound on the probability that an output is a good recommendation *for a particular user*.

[2]This algorithm makes no distinction between binary matrices and matrices with values in the interval $[0, 1]$, and the corresponding analysis is straightforward upon defining a metric for success when data is nonbinary.

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 like every 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 who desire very few products or who 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 (γ, ζ)-*typical* (where $\zeta, \gamma > 0$) if $|S| \geq (1-\zeta)m$ and, for all $i \in S$,

$$\frac{1}{1+\gamma}\frac{\|T\|_F^2}{m} \leq \|T_i\|^2 \leq (1+\gamma)\frac{\|T\|_F^2}{m}.$$

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 good recommendations; 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 that come from an approximate reconstruction of T.

Theorem 5.1. *For $T \in \mathbb{R}^{m\times n}$, S a (γ, ζ)-typical set of users, and a matrix $\tilde{T}$ satisfying $\|T - \tilde{T}\|_F \leq \varepsilon\|T\|_F$,*

$$\mathrm{E}_{i\sim_u S}\left[\|\mathcal{D}_{T_i}, \mathcal{D}_{\tilde{T}_i}\|_{TV}\right] \leq \frac{2\varepsilon\sqrt{1+\gamma}}{1-\zeta},$$

and for a chosen parameter $\psi > 0$ there exists some $S' \subset S$ of size at least $(1-\psi-\zeta)m$ such that, for $i \in S'$,

$$\|\mathcal{D}_{T_i}, \mathcal{D}_{\tilde{T}_i}\|_{TV} \leq 2\varepsilon\sqrt{\frac{1+\gamma}{\psi}}.$$

The first bound is an average-case bound on typical users and the second is a strengthening of the resulting Markov bound. Both bound total variation distance from $\mathcal{D}_{T_i}$, which we deem a good goal distribution to sample from for recommendations. We defer the proof of this theorem to the appendix.

When we don't aim for a particular distribution and only want to bound the probability of giving a bad recommendation, we can prove a stronger average-case bound on the failure probability.

Theorem 5.2 (Theorem 3.3 of [13])**.** *For $T \in \mathbb{R}^{m\times n}$ a* binary *preference matrix, 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* $(1-\psi-\zeta)m$ *such that*

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

For intuition, if ε is sufficiently small compared to the other parameters, this bound becomes

$$O\left(\frac{\varepsilon^2(1+\gamma)}{1-\psi-\zeta}\right).$$

The total variation bound from Theorem 5.1 is not strong enough to get this: the failure probability we get is $2\varepsilon\sqrt{1+\gamma}/(1-\psi-\zeta)$. A more careful analysis is necessary to gain the extra ε factor.

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

In summary, we have reduced the problem of "find a good recommendation for a user" to "given some entries from a close-to-low-rank matrix T, sample from $\tilde{T}_i$ for some $\tilde{T}$ satisfying $\|T - \tilde{T}\|_F \leq \varepsilon\|T\|_F$ for small ε."

5.2 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 [2]. In this model, the entries we are given are chosen uniformly over all entries. This model has seen use previously in the theoretical recommendation systems literature [9].

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

$$\hat{T}_{ij} = \begin{cases} \frac{T_{ij}}{p} & \text{with probability } p \\ 0 & \text{with probability } 1-p \end{cases}. \qquad (\star)$$

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

When the entries of T are bounded, $\hat{T}$ is T perturbed by a random matrix E whose entries are independent and bounded random variables. Standard concentration inequalities imply that such random matrices don't have large singular values (the largest singular value is,

say, $O(\sqrt{n/p})$). Thus, for some vector v, if $\|Tv\|/\|v\|$ is large (say, $O(\sqrt{mn/k})$), then $\|(T+E)v\|/\|v\|$ will still be large, despite E having large Frobenius norm.

The above intuition suggests that when T has large singular values, its low-rank approximation T_k is not perturbed much by E, and thus, low-rank approximations of $\hat{T}$ are good reconstructions of T. A series of theorems by Achlioptas and McSherry [2] and Kerenidis and Prakash [13] formalizes this intuition. For brevity, we only describe a simplified form of the last theorem in this series, which is the version they (and we) use for analysis. It states that, under appropriate circumstances, it's enough to compute $\hat{T}_{\geq\sigma,\kappa}$ for appropriate σ and κ.

Theorem 5.3 (4.3 of [13])**.** *Let $T \in \mathbb{R}^{m\times n}$ and let $\hat{T}$ be the random matrix defined in ($\star$), with $p \geq \frac{3\sqrt{nk}}{2^{9/2}\varepsilon^3\|T\|_F}$ and $\max_{ij}|T_{ij}| = 1$. Let $\sigma = \sqrt{\frac{\varepsilon^2 p}{8k}}\|\hat{T}\|_F$, let $\kappa = 1/3$, and assume that $\|T\|_F \geq \frac{9}{\sqrt{2}\varepsilon^3}\sqrt{nk}$. Then with probability at least $1 - \exp(-19(\log n)^4)$,*

$$\|T - \hat{T}_{\geq\sigma,\kappa}\|_F \leq 3\|T - T_k\|_F + 3\varepsilon\|T\|_F.$$

With this theorem, we have a formal goal for a recommendation systems algorithm. We are given some subsample $A = \hat{T}$ of the preference matrix, along with knowledge of the size of the subsample p, the rank of the preference matrix k, and an error parameter ε. Given that the input satisfies the premises for Theorem 5.3, for some user i, we can provide a recommendation by sampling from $(A_{\geq\sigma,\kappa})_i$ with σ, κ specified as described. Using the result of this theorem, $A_{\geq\sigma,\kappa}$ is close to T, and thus we can use the results of Section 5.1 to conclude that such a sample is likely to be a good recommendation for typical users.

Now, all we need is an algorithm that can sample from $(A_{\geq\sigma,\kappa})_i$.

6 Tools and Subroutines

We present the algorithm and analysis for Theorem 1 nonlinearly, starting with simple techniques and building up more complex subroutines.

First, we discuss vector sampling and give algorithms that use sampling access to their input vectors to perform basic tasks. Second, we introduce the notion of approximate orthonormality, a property of some of the vectors in our main algorithm, and present lemmas that allow us to treat approximately orthonormal vectors as orthonormal. Third, we present ModFKV, an algorithm to find low-rank matrix approximations, and use the techniques built up in the first two sections to prove nice properties about its output.

6.1 Vector Sampling

Recall how we defined sampling from a vector.

Definition. For a vector $x \in \mathbb{R}^n$, we denote by $\mathcal{D}_x$ the distribution over $[n]$ with density function $\mathcal{D}_x(i) = x_i^2/\|x\|^2$. We call a sample from $\mathcal{D}_x$ a sample from x.

We will need that closeness of vectors in ℓ^2-norm implies closeness of their respective distributions in TV distance:

Lemma 6.1. For $x, y \in \mathbb{R}^n$ satisfying $\|x - y\| \le \varepsilon$, the corresponding distributions $\mathcal{D}_x$, $\mathcal{D}_y$ satisfy $\|\mathcal{D}_x, \mathcal{D}_y\|_{TV} \le 2\varepsilon/\|x\|$.

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

$$\begin{aligned}\|\mathcal{D}_x, \mathcal{D}_y\|_{TV} &= \frac{1}{2}\sum_{i=1}^{n} \left|\hat{x}_i^2 - \hat{y}_i^2\right| = \frac{1}{2}\big\langle |\hat{x} - \hat{y}|, |\hat{x} + \hat{y}| \big\rangle \le \frac{1}{2}\|\hat{x} - \hat{y}\|\|\hat{x} + \hat{y}\| \le \|\hat{x} - \hat{y}\| \\ &= \frac{1}{\|x\|}\Big\|x - y - (\|x\| - \|y\|)\hat{y}\Big\| \le \frac{1}{\|x\|}\Big(\|x - y\| + \big|\|x\| - \|y\|\big|\Big) \le \frac{2\varepsilon}{\|x\|}\end{aligned}$$

The first inequality follows from Cauchy-Schwarz, and the rest follow from triangle inequality. □

Now, we will give two subroutines that can be performed, assuming some vector sampling access. First, we show that we can estimate the dot product of two vectors well (used later in Algorithm 4).

Proposition 6.2. *Given query access to $x, y \in \mathbb{R}^n$, sample access to $\mathcal{D}_x$, and knowledge of $\|x\|$, $\langle x, y\rangle$ can be estimated to additive error $\|x\|\|y\|\varepsilon$ with at least $1 - \delta$ probability using $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ queries and samples (and the same time complexity).*

Proof. Perform samples in the following way: for each i, let the random variable Z be y_i/x_i with probability $x_i^2/\|x\|^2$ (select the index by sampling from $\mathcal{D}_x$). We then have:

$$\begin{aligned}\mathrm{E}[Z] &= \sum \frac{y_i}{x_i}\frac{x_i^2}{\|x\|^2} = \frac{\sum x_i y_i}{\|x\|^2} = \frac{\langle x, y\rangle}{\|x\|^2}, \\ \mathrm{Var}[Z] &= \sum \left(\frac{y_i}{x_i}\right)^2 \frac{x_i^2}{\|x\|^2} = \frac{\sum y_i^2}{\|x\|^2} = \frac{\|y\|^2}{\|x\|^2}.\end{aligned}$$

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 $\frac{9}{2\varepsilon^2}$ copies of Z to get within $\varepsilon\sigma = \varepsilon\|x\|\|y\|$ of $\langle x, y\rangle$ with probability at least $1 - \delta$ in $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$ accesses. All of the techniques used here take linear time. □

Second, we show that, given sample access to some vectors, we can sample from a linear combination of them. This uses rejection sampling: given sampling access to a distribution P,

Algorithm 1: Rejection Sampling

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

rejection sampling allows for sampling from a "close" distribution Q, provided we can compute some information about their corresponding distributions. We describe the procedure below.[3]

The standard rejection sampling guarantee is as follows:

Lemma 6.3. 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.[4]

We now use rejection sampling to prove the following proposition.

Proposition 6.4. *Suppose we are given query access, sample access, and knowledge of the norms of the columns of $V \in \mathbb{R}^{n\times k}$. Then given $w \in \mathbb{R}^k$ (as input), we can output a sample from Vw in $O(k^2 C(V,w))$ expected query complexity and expected time complexity, where*

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

C measures the amount of cancellation for Vw. For example, when the columns of V are orthogonal, $C = 1$ for all nonzero w, since there is no cancellation. Conversely, when the columns of V are linearly dependent, there is a choice of w such that $\|Vw\| = 0$, maximizing cancellation. In this context, C is undefined, matching the fact that sampling from the zero vector is undefined. By perturbing w we can find vectors requiring arbitrarily large values of C.

Proof. We use rejection sampling: see Algorithm 1 and Lemma 6.3.

In our case, P is the distribution formed by sampling from $\mathcal{D}_{V^{(i)}}$ with probability proportional to $\|w_i V^{(i)}\|^2$, and Q is the target $\mathcal{D}_{Vw}$. We have

$$r_i = \frac{(Vw)_i^2}{k\sum_{j=1}^k (w_j V_{ij})^2},$$

which we can compute in k queries. This expression is written in a way the algorithm can directly compute, but it can be put in the form of the rejection sampling procedures stated

[3] We leave the input implicit. In some of our applications, we cannot compute probabilities of the input random variables, but we can still literally perform the procedure, so rejection sampling works as intended.

[4] The number of iterations is a geometric random variable, so this can be converted into a bound guaranteeing a sample in $M \log 1/\delta$ iterations with failure probability $1-\delta$, provided the algorithm knows M. All expected complexity bounds we deal with can be converted to high probability bounds in the manner described.

above:

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

M is independent of i, so it is a constant as desired. To prove correctness, all we need to show is that our choice of r_i is always at most 1. This follows from Cauchy-Schwarz:

$$r_i = \frac{(Vw)_i^2}{k \sum_{j=1}^{k} (w_j V_{ij})^2} = \frac{(\sum_{j=1}^{k} w_j V_{ij})^2}{k \sum_{j=1}^{k} (w_j V_{ij})^2} \leq 1.$$

Each iteration of the procedure takes $O(k)$ queries, leading to a query complexity of $O(k^2 C(V, w))$. Time complexity is linear in the number of queries. □

While the above proposition will work for one use of rejection sampling in the full algorithm (Proposition 6.14), we will need a technical lemma to allow us to perform rejection sampling when conditions are relaxed slightly: we can only compute approximate probabilities $p(i) \approx P(i)$. This lemma will be used in Lemma 6.8. The procedure is as follows:

Algorithm 2: Relaxed Rejection Sampling

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

In this situation, we can approximately sample from the target distribution.

Lemma 6.5. Suppose $r_i \leq 1$ and $a \leq P(i)/p(i) \leq A$ for all i in the support of X. (Note that p need not be a probability density function.) Then Algorithm 2 is well-defined and outputs samples from a distribution Q' satisfying $\|Q, Q'\|_{TV} \leq (A - a)/2a$ in at most M/a iterations in expectation.

Proof. We accept with probability

$$\sum_i \frac{Q(i)}{Mp(i)} P(i) \geq \frac{a}{M}$$

and our output distribution is

$$Q'(i) = \frac{\Pr[\text{accept on } i]}{\Pr[\text{accept}]} = \frac{\frac{Q(i)P(i)}{Mp(i)}}{\sum_j \frac{Q(j)P(j)}{Mp(j)}} = \frac{\frac{Q(i)P(i)}{p(i)}}{\sum_j \frac{Q(j)P(j)}{p(j)}}.$$

The TV distance is bounded as follows:

$$\begin{aligned}\|Q, Q'\|_{TV} &= \frac{1}{2}\sum_i \left| Q(i) - \frac{\frac{Q(i)P(i)}{p(i)}}{\sum_j \frac{Q(j)P(j)}{p(j)}} \right| \\ &= \frac{1}{2}\sum_i Q(i) \left| 1 - \frac{\frac{P(i)}{p(i)}}{\sum_j \frac{P(j)Q(j)}{p(j)}} \right| \\ &\leq \frac{1}{2}\sum_i Q(i)\frac{A-a}{a} = \frac{A-a}{2a},\end{aligned}$$

where the last step results from the inequality

$$\frac{a}{A} \leq \frac{\frac{P(i)}{p(i)}}{\sum \frac{Q(x)P(x)}{p(x)}} \leq \frac{A}{a}.$$

□

6.2 Approximate Orthonormality

As described in Section 1.3, our main algorithm will have estimated singular vectors that are not orthonormal, but close to orthonormal. We will define this notion of approximate orthonormality, and prove that such vectors have exactly the properties one would expect from the name.

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

$$I - \frac{\alpha}{k}\vec{1}\vec{1}^T \preceq V^TV \preceq I + \frac{\alpha}{k}\vec{1}\vec{1}^T.$$

Here, $\preceq$ denotes entry-wise inequality and $\vec{1}$ denotes the vector of ones. This definition is equivalent to saying that any dot product is at most α/k off from the expected result.

We will assume that $\alpha \leq 1$.

Lemma 6.6. If $V \in \mathbb{R}^{n\times k}$ is α-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

$$I - \frac{\alpha}{k}\vec{1}\vec{1}^T \preceq V^TV = R^TQ^TQR = R^TR \preceq I + \frac{\alpha}{k}\vec{1}\vec{1}^T.$$

Since $R \in \mathbb{R}^{n\times k}$ is upper triangular, we can think about R as $k \times k$. Thus, R^TR is an approximate Cholesky factorization of I, with error $\frac{\alpha}{k}\vec{1}\vec{1}^T$; by Theorem 1 in Chang, Paige, and Stewart's perturbation analyses [8], $\|R - I\|_F \leq \frac{\alpha}{\sqrt{2}} + O(\alpha^2)$.

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 $\|U - V\|_F = \|Q(I - R)\|_F = \|I - R\|_F \leq \alpha/\sqrt{2} + O(\alpha^2)$. □

The above lemma makes the notion of approximate orthonormality very versatile: we can treat these vectors as orthonormal with only α change in Frobenius norm. We give a basic corollary to illustrate.

Corollary 6.7. *For V α-approximately orthonormal, $\|VV^T - \Pi_V\|_F \lesssim \alpha$, where Π_V is the orthogonal projector on the image of V.*

Essentially, we can treat VV^T as an orthonormal projector. This follows from Lemma 6.6 by a simple calculation:

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

We use the above lemma and corollary throughout the algorithm analyses to follow.

Next, we observe that sampling from a linear combination of approximately orthonormal vectors can be done quickly *without* knowledge of the norms of these vectors. This will be used in the last step of Algorithm 4.

Lemma 6.8. Given sample and query access to $V \in \mathbb{R}^{n \times k}$, a set of α-approximately orthonormal vectors, and an input vector $w \in \mathbb{R}^k$, we can output a sample from a distribution $(\alpha + O(\alpha^2))$-close to $\mathcal{D}_{Vw}$ in $O(k^2(1 + O(\alpha)))$ expected query complexity and expected time complexity.

Proof. We use our relaxed rejection sampling procedure: see Algorithm 2 and Lemma 6.5.

Here, P is the distribution formed by sampling from $\mathcal{D}_{V^{(i)}}$ with probability proportional to w_i^2, and $Q = \mathcal{D}_{Vw}$. Further, we choose

$$p(s) = \frac{\sum_i (w_i V_s^{(i)})^2}{\|w\|^2}.$$

By approximate orthonormality of V, we can choose $a = 1/(1 + \alpha)$ and $A = 1/(1 - \alpha)$:

$$\frac{P(s)}{p(s)} = \frac{\sum_i \left(w_i \frac{V_s^{(i)}}{\|V^{(i)}\|}\right)^2 / \|w\|^2}{\sum_i (w_i V_s^{(i)})^2 / \|w\|^2} = \frac{\sum (w_i V_s^{(i)})^2 / \|V^{(i)}\|^2}{\sum_i (w_i V_s^{(i)})^2}.$$

In k queries to V, we can compute

$$r_i = \frac{(Vw)_i^2}{k \sum_j (w_j V_i^{(j)})^2} = \frac{Q(s)}{\frac{k\|w\|^2}{\|Vw\|^2} p(s)},$$

which means that

$$M = \frac{k\|w\|^2}{\|Vw\|^2} \leq \frac{k}{1 - O(\alpha)} = k(1 + O(\alpha)).$$

Finally,

$$r_s = \frac{(Vw)_s^2}{k\sum_i (w_i V_s^{(i)})^2} \leq 1.$$

With this, we have verified the premises for Lemma 6.5. The corresponding result is the desired result here. The operations here take time linear in the number of queries. □

6.3 Finding a Low-Rank Approximation

Now, we describe the low-rank approximation algorithm that we use at the start of the main algorithm.

Theorem 6.9. *Given a matrix $A \in \mathbb{R}^{m\times n}$ supporting the query operations described in Proposition 4.2, along with parameters $\sigma \in (0, \|A\|_F], \varepsilon \in (0, \sqrt{\sigma/\|A\|_F}/4], \kappa \in [\varepsilon^2, 1]$, there is an algorithm that outputs a succinct description (of the form described below) of some D satisfying $\|D - A_{\sigma,\kappa}\|_F \leq \varepsilon\|A\|_F$ with probability at least $1-\delta$ and*

$$O\left(\text{poly}\left(\frac{\|A\|_F^2}{\sigma^2}, \frac{1}{\varepsilon}, \frac{1}{\kappa}, \log\frac{1}{\delta}\right)\right)$$

query and time complexity.

To prove this theorem, we will modify the algorithm given in [10] and show that it satisfies the desired properties. We present the algorithm as Frieze, Kannan, and Vempala do, not accounting for the δ failure probability, instead aiming for a constant failure probability to be amplified by standard techniques.

We can get the output matrix D from its description by projecting A onto a subspace defined by vectors $\hat{V} \in \mathbb{R}^{n\times k}$. That is,

$$D = A\hat{V}\hat{V}^T = A\sum_{t=1}^{k} (\hat{V}^{(t)})(\hat{V}^{(t)})^T.$$

The columns of $\hat{V}$ are described implicitly as a linear combination of rows of A. Let S be the submatrix given by restricting the rows to $i_1, \ldots, i_p$ and renormalizing row r by $1/\sqrt{p\mathcal{D}_{\tilde{A}}(r)}$ (so they all have the same norm). Then $\hat{V}^{(i)} := S^T u^{(i)}/\sigma^{(i)}$.

To summarize, ModFKV subsamples a matrix, computes the large singular vectors of this matrix, and outputs those singular vectors, with the promise that the singular vectors of the subsample give a good description of the singular vectors of the full matrix. More of the underpinnings are explained in Frieze, Kannan, and Vempala's paper [10].

ModFKV differs from FKV only in that σ 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 σ 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.

Algorithm 3: ModFKV

Input: Matrix $A \in \mathbb{R}^{m \times n}$ supporting operations in Proposition 4.2, threshold σ, error parameters ε, κ

Output: A description of an output matrix D

Set $K = \|A\|_F^2/\sigma^2$;

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

Set $p = 10^7 \max\{\frac{K^4}{\bar{\varepsilon}^3}, \frac{K^2}{\bar{\varepsilon}^4}\}$;

Sample rows $i_1, \ldots, i_p$ from $\mathcal{D}_{\tilde{A}}$;

Let $\mathcal{F}$ denote the distribution given by choosing an $s \sim_u [p]$, and choosing a column from $\mathcal{D}_{\tilde{A}_{i_s}}$;

Sample columns $j_1, \ldots, j_p$ from $\mathcal{F}$;

Let the resulting $p \times p$ submatrix, with row r normalized by $\frac{1}{\sqrt{p\mathcal{D}_{\tilde{A}}(r)}}$ and column c normalized by $\frac{1}{\sqrt{p\mathcal{F}(c)}}$, be denoted W;

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

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

To analyze the complexity of the algorithm, notice that query complexity is dominated by querying all of the entries of W, which is $O(p^2)$, and time complexity is dominated by computing W's SVD, which is $O(p^3)$. We can convert this to the input parameters using that

$$p = O\left(\max\left\{\frac{\|A\|^{11}}{\sigma^{11}\varepsilon^6\kappa^3}, \frac{\|A\|^8}{\sigma^8\varepsilon^8\kappa^4}\right\}\right).$$

We will first note two useful observations: the first is the bound that shows that the output matrix a good low-rank approximation; the second will be useful when discussing the description in Proposition 6.14.

Lemma 6.10. The following are properties of the modified algorithm:

(i) The following bound holds for the output matrix D (here, k is the width of $\hat{V}$, and thus a bound on $\operatorname{rank} D$):

$$\|A - D\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon\|A\|_F^2 \qquad \ell((1 + \bar{\varepsilon}\sqrt{K})\sigma) \leq k \leq \ell((1 - \bar{\varepsilon}\sqrt{K})\sigma) \qquad (\diamond)$$

(ii) In the description, $u^{(i)}$'s have unit norm and $\|A\|_F^2/2 \leq \|S\|_F^2 \leq 3\|A\|_F^2/2$.

The following property will be important both in ModFKV's analysis and when the output description is used later in Algorithm 4. The estimated singular vectors in $\hat{V}$ behave like singular vectors, in that they are approximately orthonormal.

Proposition 6.11. *The output vectors $\hat{V}$ are $\bar{\varepsilon}^2 K$-approximately orthonormal.*

The proofs of the above lemma and proposition delve into FKV's analysis, so we defer them to the appendix.

The guarantee on our output matrix D is ($\diamond$), but for our recommendation system, we want that D is close to some $A_{\geq\sigma,\kappa}$. Now, we present the core theorem showing that the former kind of error implies the latter.

Theorem 6.12. *If Π a k-dimensional orthogonal projector satisfies*

$$\|A_k\|_F^2 \leq \|A\Pi\|_F^2 + \varepsilon\sigma_k^2,$$

then

$$\|A\Pi - A_{\sigma_k,\kappa}\|_F^2 \lesssim \varepsilon\sigma_k^2/\kappa,$$

where $\varepsilon \leq \kappa \leq 1$.[5]

The proof is somewhat involved, so we defer it to the appendix. To our knowledge, this is a novel translation of a typical FKV-type bound as in ($\diamond$) to a new, useful type of bound, so we believe this theorem may find use elsewhere.

Now, we use this theorem to show that D is close to some $A_{\sigma,\kappa}$.

Corollary 6.13. $\|D - A_{\sigma,\kappa}\|_F \lesssim \varepsilon\|A\|_F/\sqrt{\kappa}$.

Proof. Throughout the course of this proof, we simplify and apply theorems based on the restrictions on the parameters in Theorem 6.9.

First, notice that the bound in ($\diamond$) can be translated to the type of bound in the premise of Theorem 6.12, using Proposition 6.11 and Corollary 6.7.

$$\begin{aligned}
\|A - D\|_F^2 &\leq \|A - A_k\|_F^2 + \bar{\varepsilon}\|A\|_F^2 \\
\|A - A(\Pi + E)\|_F^2 &\leq \|A - A_k\|_F^2 + \bar{\varepsilon}\|A\|_F^2 \\
(\|A - A\Pi\|_F - \bar{\varepsilon}^2 K\|A\|_F)^2 &\lesssim \|A - A_k\|_F^2 + \bar{\varepsilon}\|A\|_F^2 \\
\|A - A\Pi\|_F^2 &\lesssim \|A - A_k\|_F^2 + \max\{\bar{\varepsilon}, \bar{\varepsilon}^2 K\}\|A\|_F^2 \\
\|A\| - \|A\Pi\|_F^2 &\lesssim \|A\| - \|A_k\|_F^2 + \max\{\bar{\varepsilon}, \bar{\varepsilon}^2 K\}\|A\|_F^2 \\
\|A_k\|_F^2 &\lesssim \|A\Pi\|_F^2 + (\max\{\bar{\varepsilon}, \bar{\varepsilon}^2 K\}\|A\|_F^2/\sigma_k^2)\sigma_k^2
\end{aligned}$$

The result of the theorem is that

$$\|A\Pi - A_{\sigma_k, \frac{\kappa - \bar{\varepsilon}\sqrt{K}}{1 - \bar{\varepsilon}\sqrt{K}}}\|_F^2 \lesssim \frac{(\max\{\bar{\varepsilon}, \bar{\varepsilon}^2 K\}\|A\|_F^2/\sigma_k^2)\sigma_k^2}{\frac{\kappa - \bar{\varepsilon}\sqrt{K}}{1 - \bar{\varepsilon}\sqrt{K}}} \lesssim \max\left\{\frac{\bar{\varepsilon}}{\kappa}, \frac{\bar{\varepsilon}^2 K}{\kappa}\right\}\|A\|_F^2.$$

The second part of ($\diamond$), bounding k, implies that any $A_{\sigma_k, \frac{\kappa - \bar{\varepsilon}\sqrt{K}}{1 - \bar{\varepsilon}\sqrt{K}}}$ is also an $A_{\sigma,\kappa}$ (the error of

[5] An analogous proof gives the more general bound $\|\Pi - \Pi_{\sigma,\kappa}\|_F^2 \lesssim \varepsilon/\kappa$.

the former is contained in the latter), so we can conclude that

$$\begin{aligned}\|D - A_{\sigma,\kappa}\|_F &= \|A(\Pi + E) - A_{\sigma,\kappa}\|_F \\ &\leq \|A\Pi - A_{\sigma,\kappa}\|_F + \|A\|_F\|E\|_F \\ &\lesssim \|A\Pi - A_{\sigma,\kappa}\|_F + \bar{\varepsilon}^2 K\|A\|_F \\ &\lesssim \max\left\{\sqrt{\frac{\bar{\varepsilon}}{\kappa}}, \sqrt{\frac{\bar{\varepsilon}^2 K}{\kappa}}, \bar{\varepsilon}^2 K\right\}\|A\|_F.\end{aligned}$$

$\bar{\varepsilon}$ was chosen so that the final term is bounded by $\varepsilon\|A\|_F$. □

This completes the proof of Theorem 6.9.

We now have an algorithm that can quickly compute low-rank matrix approximations and output them as succinct descriptions. However, 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 to $\hat{V}$, and that will be enough.

Proposition 6.14. *Given queries to the operations described in Proposition 4.2, along with the description of $\hat{V}$, we can sample from any $\hat{V}^{(i)}$ in $O(Kp^2)$ expected queries and query 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 normalized 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. There are k such vectors, making $\hat{V}$ an $n \times k$ matrix.

We can query to any particular entry easily: $\hat{V}_j^{(i)} = \frac{1}{\sigma^{(i)}}\sum_{x=1}^{p} A_{i_x j}u_x^{(i)}$, 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 6.4. In this case,

$$C(S^T, u^{(i)}/\sigma^{(i)}) = \frac{\sum_{j=1}^{p}\|u_j^{(i)}S_j/\sigma^{(i)}\|^2}{\|S^T u^{(i)}/\sigma^{(i)}\|^2} \leq \frac{\|u^{(i)}/\sigma^{(i)}\|^2\left(\sum_{j=1}^{p}\|S_j\|^2\right)}{(1-\varepsilon^4)^2} \leq \frac{3\|A\|_F^2}{2\sigma^2(1-\varepsilon^4)^2} \lesssim K.$$

using Cauchy-Schwarz, Proposition 6.11, and Lemma 6.10 part (ii). □

As it turns out, these subroutines are all we need from the description of D. Since $D = A\hat{V}\hat{V}^T$, and we have query and sample access to the columns of $\hat{V}$, we can black-box the rest of the description. Outside this section, we treat D as $A\hat{V}\hat{V}^T$, where $\hat{V}$ are approximately orthonormal vectors with query and sampling access.

7 Main Algorithm

Here, we give the proofs for Theorem 1 and Theorem 2. Theorem 1 contains the low-rank approximation sampling algorithm and its corresponding analysis. Theorem 2 contains the analysis that shows that such an algorithm produces good recommendations in our model.

7.1 Proof of Theorem 1

Theorem 1. *There is a classical algorithm that, given a matrix A with query and sampling assumptions as described in Proposition 4.2, along with a row $i \in [m]$, threshold σ, $\kappa \in [0,1]$, and sufficiently small $\varepsilon > 0$, has an output distribution $O(\varepsilon)$-close in total variation distance to $\mathcal{D}_{D_i}$ where $D \in \mathbb{R}^{m \times n}$ satisfies $\|D - A_{\geq \sigma,\kappa}\|_F \leq \varepsilon \|A\|_F$ for some $A_{\geq\sigma,\kappa}$, in query and time complexity*

$$O\left(\text{poly}\left(\frac{\|A\|_F}{\sigma}, \frac{1}{\varepsilon}, \frac{1}{\kappa}, \frac{\|A_i\|}{\|D_i\|}, \log\frac{1}{\delta}\right)\right)$$

where δ is the probability of failure.

Proof. We will give an algorithm (Algorithm 4) where the error in the output distribution is $O(\varepsilon\|A_i\|/\|D_i\|)$-close to $\mathcal{D}_{D_i}$, and there is no dependence on $\|A_i\|/\|D_i\|$ in the runtime.

Algorithm 4: Low-rank approximation sampling

Input: Matrix $A \in \mathbb{R}^{m\times n}$ supporting the operations in 4.2, user $i \in [m]$, threshold σ, $\varepsilon > 0$, $\kappa > 0$

Output: Sample $s \in [n]$

Run ModFKV (3) with $(\sigma, \varepsilon, \kappa)$ parameters as $(\sigma(1-\kappa/2), \varepsilon, 2\kappa/(2-\kappa))$ to get a description of $D = A\hat{V}\hat{V}^T$;

For the following, simulate $\hat{V} \in \mathbb{R}^{n\times k}$ from the description as described in Proposition 6.14;

Use Proposition 6.2 with parameters $\frac{\varepsilon}{\sqrt{k}}$ to estimate $\langle A_i, \hat{V}^{(t)}\rangle$ for all $t \in [k]$;

Let est be the $1 \times k$ vector of estimates: $\text{est} = \{\langle A_i, \hat{V}^{(t)}\rangle\}_{t\in[k]}$;

Use Lemma 6.8 to get a sample s from $\text{est}\hat{V}^T$;

Output s;

Correctness: By Theorem 6.9, for sufficiently small[6] ε, the output matrix D satisfies

$$\|D - A_{\geq\sigma,\kappa}\|_F \leq \varepsilon\|A\|_F,$$

using that $A_{\sigma(1-\frac{\kappa}{2}),\frac{2\kappa}{2-\kappa}}$ and $A_{\geq\sigma,\kappa}$ are the exact same form of error. So, all we need is to approximately sample from the ith row of D, given its description.

Since $A_i\hat{V}$ is such a small vector ($1 \times k$), we can give an estimate for it in ℓ^2-norm easily. The guarantee from Proposition 6.2 states that each estimate of an entry is at most $\varepsilon\|\hat{V}^{(i)}\|\|A_i\|/\sqrt{k}$ off from the correct value, meaning that $\|\text{est} - A_i\hat{V}\| \leq \varepsilon\|A_i\|(1 + O(\alpha))$.

We (approximately) sample from $\text{est}\hat{V}^T$, so now we have to determine how far the output distribution is from the desired distribution. Using approximate orthonormality of $\hat{V}$,

[6]This is not a strong restriction: $\varepsilon \lesssim \min\{\sqrt{\kappa}, \sqrt{\sigma/\|A\|_F}\}$ works. This makes sense: for ε any larger, the error can encompass addition or omission of full singular vectors.

$\|\text{est}\hat{V}^T - A_i\hat{V}\hat{V}^T\| \leq \varepsilon\|A_i\|(1+O(\alpha))$. Let $\mathcal{O}$ be the output distribution. Then, combining the error from sampling in Lemma 6.8 with the ℓ^2 error from est (using Lemma 6.1),

$$\|\mathcal{O}, \mathcal{D}_{A_i\hat{V}\hat{V}^T}\|_{TV} \leq \|\mathcal{O}, \mathcal{D}_{\text{est}\hat{V}^T}\|_{TV} + \|\mathcal{D}_{\text{est}\hat{V}^T}, \mathcal{D}_{A_i\hat{V}\hat{V}^T}\|_{TV} \lesssim \alpha + \frac{\varepsilon\|A_i\|}{\|A_i\hat{V}\hat{V}^T\|},$$

which is the desired correctness bound.

Runtime: The query complexity is dominated by the use of Proposition 6.2, since it repeatedly uses the subroutine in Proposition 6.14, resulting in

$$\tilde{O}\left(\frac{\|A\|^2}{\sigma^2}\Big(\max\left\{\frac{\|A\|^{11}}{\sigma^{11}\varepsilon^6\kappa^3}, \frac{\|A\|^8}{\sigma^8\varepsilon^8\kappa^4}\right\}\Big)^2\Big(\frac{\|A\|_F^6}{\sigma^6\varepsilon^2}\Big)\right) = \tilde{O}\left(\max\left\{\frac{\|A\|^{30}}{\sigma^{30}\varepsilon^{14}\kappa^6}, \frac{\|A\|^{24}}{\sigma^{24}\varepsilon^{18}\kappa^8}\right\}\right),$$

where the $\tilde{O}$ hides the log factors incurred by amplifying the failure probability to δ.[7] The time complexity is dominated by computing the SVD of the $p \times p$ matrix in ModFKV in $O(p^3)$ time, resulting in

$$\tilde{O}\left(\max\left\{\frac{\|A\|^{33}}{\sigma^{33}\varepsilon^{18}\kappa^9}, \frac{\|A\|^{24}}{\sigma^{24}\varepsilon^{24}\kappa^{12}}\right\}\right).$$

Finally, we give some brief comments about variants of this algorithm.

- By repeatedly estimating $A_i\hat{V}$ (creating est_1, est_2, etc.) with progressively smaller ε, one can eventually get within $O(\varepsilon\|A_i\hat{V}\|/\|A_i\|)$ of the correct vector. Doing this will decrease the total variation error to $O(\varepsilon)$, and increase the runtime by $\tilde{O}(\|A_i\|^2/\|VV^TA_i\|^2)$, giving the promised bound in the theorem statement.
- While the input is a user $i \in [m]$ (and thus supports query and sampling access), it need not be. Most generally, given query and sample access to orthonormal (or approximately orthonormal) vectors $V \in \mathbb{R}^{n\times k}$, and query access to $x \in \mathbb{R}$, one can approximately sample from a distribution $O(\varepsilon\|x\|/\|VV^Tx\|)$-close to $\mathcal{D}_{VV^Tx}$, the projection of x onto the span of V, in $O(k^2 \log k/\varepsilon^2 \log 1/\delta)$ time.
- While the SVD dominates the time complexity of Algorithm 4, the same description output by ModFKV can be used for multiple recommendations, amortizing the cost down to the query complexity (since the rest of the algorithm is linear in the number of queries).

□

7.2 Proof of Theorem 2

Theorem 2. *Applying Theorem 1 to the recommendation systems model (described in Section 5) with the quantum state preparation data structure (Section 4) achieves the same bounds of quality of recommendations as the quantum algorithm in [13], up to constant factors and for sufficiently small ε.*

[7] The full computation has factors of $1 + O(\text{poly}(\varepsilon))$ that we ignore in this abbreviated version.

Proof. To do this, we just run Algorithm 4 with parameters as described in Proposition 5.3: $\sigma = \sqrt{\frac{\varepsilon^2 p}{8k}}\|A\|_F$, ε, and $\kappa = 1/3$. Provided $\varepsilon \lesssim \sqrt{p/k}$, we get the desired output. We can perform the algorithm because A is in the data structure given by Proposition 4.2 (inflating the runtime by a factor of $\log^2(mn)$).

Just like Kerenidis and Prakash, we need to assume that the following two conditions hold:

- The subsample probability p is constant; if p is subconstant, then the runtime becomes larger. This is typical for recommendation systems [9, 4, 13].
- $\|A_i\|/\|D_i\|$ is constant. This holds for a constant fraction of (γ, ζ)-typical users (as defined in Section 5.1), provided ζ is constant. We defer the details to Kerenidis and Prakash's proof in [13], and only present a broad sketch of the proof in this context. The main idea is that, when $\|T - D\|_F \leq O(\varepsilon)\|T\|_F$, for a $(1-\psi)$-fraction of typical users,
$$E_{i\sim_u S'}\left[\frac{\|A_i\|^2}{\|(A_{\geq\sigma,\kappa})_i\|^2}\right] \lesssim \frac{(1+\varepsilon)^2}{(1-\psi-\zeta)\left(\frac{1}{\sqrt{1+\gamma}} + \frac{\varepsilon}{\sqrt{\psi}}\right)^2}.$$
For appropriate parameters this is a constant, and so the statement follows from Markov's inequality.

  We will see below that $\|T - D\|_F \leq O(\varepsilon + \rho)\|T\|_F$.

Correctness: Using Theorem 5.3, for D satisfying Theorem 1,

$$\|T - D\|_F \leq \|T - A_{\geq\sigma,\kappa}\|_F + \|A_{\geq\sigma,\kappa} - D\|_F \leq 3\|T - T_k\|_F + 3\varepsilon\|T\|_F + O(\varepsilon)\|A\|_F.$$

Applying a Chernoff bound to $\|A\|_F^2$ (a sum of independent random variables), we get that with high probability $1 - e^{-\|T\|_F^2 p/3}$, $\|A\|_F \leq \sqrt{2/p}\|T\|_F$. Since p is constant, and $\|T - T_k\|_F \leq \rho\|T\|_F$, we get that $\|T - D\|_F = O(\rho + \varepsilon)\|T\|_F$.

Then, we can apply Theorem 5.1 to get that, for S a (γ, ζ)-typical set of users of T, and $\mathcal{O}_i$ the output distribution for user i,

$$\begin{aligned}\mathrm{E}_{i\sim_u S}[\|\mathcal{O}_i, \mathcal{D}_{T_i}\|_{TV}] &\leq \mathrm{E}_{i\sim_u S}[\|\mathcal{O}_i, \mathcal{D}_{D_i}\|_{TV} + \|\mathcal{D}_{D_i}, \mathcal{D}_{T_i}\|_{TV}] \\ &\lesssim \varepsilon + \frac{(\varepsilon+\rho)(\sqrt{1+\gamma})}{1-\zeta} \lesssim \frac{(\varepsilon+\rho)(\sqrt{1+\gamma})}{1-\zeta},\end{aligned}$$

which is the same bound that Kerenidis and Prakash achieve.

We can also get the same bound when applying Theorem 5.2: although our total variation error is ε, we can still achieve the same desired $O(\varepsilon^2)$ failure probability as in the theorem. One needs to notice that the total variation error from error in estimating est translates to an ε^2 probability of sampling a bad recommendation, since it comes from ℓ^2-norm sampling of a vector.[8] From there, everything else follows similarly.

[8] As rough intuition: if $\|u - v\| \leq \varepsilon$ then the probability that a sample from u corresponds to a zero-entry in v is $O(\varepsilon^2/\|v\|^2)$, just because probabilities are entries squared.

In summary, the classical algorithm has two forms of error that the quantum algorithm does not. However, the error in estimating the low-rank approximation folds into the error between T and $A_{\geq\sigma,\kappa}$, and the error in total variation distance folds into the error from sampling from an inexact reconstruction of T. Thus, we can achieve the same bounds.

Runtime: Our algorithm runs in the desired time and query complexity of

$$O(\text{poly}(k, 1/\varepsilon, p, \|A_i\|/\|D_i\|)\,\text{polylog}(mn, 1/\delta)),$$

which is the same runtime as Kerenidis and Prakash's algorithm up to polynomial slowdown. Under the assumptions that p and $\|A_i\|/\|D_i\|$ are constant, this can be simplified to $O(\text{poly}(k, 1/\varepsilon)\,\text{polylog}(mn, 1/\delta))$. □

Acknowledgments

Thanks to Scott Aaronson for introducing me to this problem, advising me during the research process, and rooting for me every step of the way. His mentorship and help were integral to this work, as well as my growth and abilities as a CS researcher, and for this I am deeply grateful. 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 for the continuing help throughout the research process and the particularly incisive editing feedback. Thanks to everybody who attended my informal presentations and gave me helpful insight at Simons, including Andras Gilyen, Iordanis Kerenidis, Anupam Prakash, Mario Szegedy, and Ronald de Wolf. Thanks to Sujit Rao and anybody else that I had enlightening conversations with over the course of the project. Thanks to Prabhat Nagarajan for the continuing support.

References

[1] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291, 2015.

[2] Dimitris Achlioptas and Frank McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9, 2007.

[3] Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Mark Tuttle. Improved recommendation systems. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 1174–1183, 2005.

[4] Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 619–626. ACM, 2001.

[5] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[6] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997.

[7] James Bennett, Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[8] Xiao-Wen Chang, Christopher C Paige, and GW Stewart. New perturbation analyses for the Cholesky factorization. *IMA journal of numerical analysis*, 16(4):457–484, 1996.

[9] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 82–90. ACM, 2002.

[10] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *Journal of the ACM (JACM)*, 51(6):1025–1041, 2004.

[11] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[12] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *arXiv preprint arXiv:1704.04992*, 2017.

[13] Iordanis Kerenidis and Anupam Prakash. Quantum Recommendation Systems. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67, pages 49:1–49:21, Dagstuhl, Germany, 2017.

[14] Jon Kleinberg and Mark Sandler. Using mixture models for collaborative filtering. *Journal of Computer and System Sciences*, 74(1):49–69, 2008.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[16] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Recommendation systems: A probabilistic analysis. *Journal of Computer and System Sciences*, 63(1):42 – 61, 2001.

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

[18] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631, 2014.

[19] John Preskill. Quantum computing in the NISQ era and beyond. *arXiv preprint arXiv:1801.00862*, 2018.

A Deferred Proofs

Proof of Theorem 5.1. The following shows the first, average-case bound (note the use of Lemma 6.1 and Cauchy-Schwarz).

$$\begin{aligned}
\mathrm{E}_{i\sim_u S}\left[\|\mathcal{D}_{T_i}, \mathcal{D}_{\tilde{T}_i}\|_{TV}\right] &= \frac{1}{|S|}\sum_{i\in S}\|\mathcal{D}_{T_i}, \mathcal{D}_{\tilde{T}_i}\|_{TV} \\
&\leq \frac{1}{(1-\zeta)m}\sum_{i\in S}\frac{2\|T_i - \tilde{T}_i\|}{\|T_i\|} \\
&\leq \frac{2(1+\gamma)}{(1-\zeta)\sqrt{m}\|T\|_F}\sum_{i\in S}\|T_i - \tilde{T}_i\| \\
&\leq 2\frac{1+\gamma}{1-\zeta}\Big(\frac{\sum_{i\in[m]}\|T_i - \tilde{T}_i\|}{\sqrt{m}\|T\|_F}\Big) \\
&\leq 2\frac{1+\gamma}{1-\zeta}\Big(\frac{\sqrt{m}\|T - \tilde{T}\|_F}{\sqrt{m}\|T\|_F}\Big) \\
&\leq \frac{2\varepsilon(1+\gamma)}{(1-\zeta)}
\end{aligned}$$

Using that $\|T - \tilde{T}\|_F \leq \varepsilon\|T\|_F$ in combination with a pigeonhole-like argument, we know that at least a $(1-\psi)$-fraction of users $i \in [m]$ satisfy

$$\|T_i - \tilde{T}_i\|^2 \leq \frac{\varepsilon^2\|A\|_F^2}{\psi m}.$$

Thus, there is a $S' \subset S$ of size at least $(1-\psi-\zeta)m$ satisfying the above. For such an $i \in S'$, we can argue from Lemma 6.1 and the definition of a (γ,ζ)-typical user that

$$\|\mathcal{D}_{T_i}, \mathcal{D}_{\tilde{T}_i}\|_{TV} \leq \frac{2\|T_i - \tilde{T}_i\|}{\|T_i\|} \leq \frac{2\varepsilon\|T\|_F(1+\gamma)\sqrt{m}}{\sqrt{\psi m}\|T\|_F} = \frac{2\varepsilon(1+\gamma)}{\sqrt{\psi}}.$$

□

Proof of Lemma 6.10.

(i) ModFKV only differs from FKV run on K in that ModFKV chooses fewer singular vectors from the subsampled SVD. Thus, because a larger K leads to a larger sample, the correctness bound for FKV run on k holds for the output of ModFKV.

As for bounding k, ModFKV can compute the first k singular values to within a cumulative additive error of $\bar{\varepsilon}\|A\|_F$. This is stated below Theorem 2 in [10].

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

(ii) That $u^{(i)}$ have unit norm follows from these vectors being singular vectors of some SVD.

The second half follows directly from Lemma 1 of [10].

□

Proof of Proposition 6.11. For $i \neq j$, we have as follows (using notation from [10]):

$$\left|\hat{v}_i^T \hat{v}_j\right| = \frac{|u_i S S^T u_j^T|}{\|W^T u_i\| \|W^T u_j\|} \leq \frac{|u_i S S^T u_j^T|}{\gamma \|W\|_F^2} \leq \frac{\theta \|S\|_F^2}{\gamma \|W\|_F^2} \lesssim \frac{\theta}{\gamma}.$$

For $i = j$, we also get a similar bound of $|\|\hat{v}_i\|^2 - 1| \lesssim \frac{\theta}{\gamma}$.

In the original algorithm's filter step, γ is chosen to be $\bar{\varepsilon}/8k$. However, in ModFKV, the bound for γ still holds for values as large as $1/K$ without altering the behavior of the algorithm. Thus, the bound θ/γ is actually much better than its normal value of $O(\bar{\varepsilon})$, it is $O(\bar{\varepsilon}^2)$. From the definition of approximate orthonormality, this means that the vectors are $O(\bar{\varepsilon}^2 K)$-approximately orthonormal, as desired. □

Proof of Theorem 6.12. For simplicity we denote σ_k by σ and $\min m, n$ by N.

$$\begin{aligned}
\|A\Pi - A_{\sigma,\kappa}\|_F^2 &= \|U\Sigma V^T(\Pi - \Pi_{\sigma,\kappa})\|_F^2 \\
&= \|\Sigma V^T(\Pi - \Pi_{\sigma,\kappa})\|_F^2 \\
&= \sum_{i=1}^{N} \sigma_i^2 \|v_i^T \Pi - v_i^T \Pi_{\sigma,\kappa}\|^2
\end{aligned}$$

That is, $A\Pi$ and $A_{\sigma,\kappa'}$ are close when their corresponding projectors behave in the same way. Let $a_i = v_i^T \Pi$, and $b_i = v_i^T \Pi_{\sigma,\kappa'}$. Note that

$$b_i = \begin{cases} v_i^T & \sigma_i \geq (1+\kappa)\sigma \\ v_i^T \Pi_E & (1+\kappa)\sigma > \sigma_i \geq (1-\kappa)\sigma \\ 0 & (1-\kappa)\sigma > \sigma_i \end{cases},$$

where Π_E is as defined in Section 3.1. Using the first and third case, and the fact that orthogonal projectors Π satisfy $\|v - \Pi v\|^2 = \|v\|^2 - \|\Pi v\|^2$, the formula becomes

$$\|A\Pi - A_{\sigma,\kappa}\|_F^2 = \sum_{1}^{\ell(\sigma(1+\kappa))} \sigma_i^2(1 - \|a_i\|^2) + \sum_{\ell(\sigma(1+\kappa))+1}^{\ell(\sigma(1-\kappa))} \sigma_i^2 \|a_i - b_i\|^2 + \sum_{\ell(\sigma(1-\kappa))+1}^{N} \sigma_i^2(\|a_i\|^2).$$

* * *

Now, we consider the assumption equation. We reformulate the assumption into the following system of equations:

$$\sum_{i=1}^{k} \sigma_i^2 \leq \sum_{i=1}^{N} \sigma_i^2 \|a_i\|^2 + \varepsilon\sigma^2 \qquad \sigma_i^2 \text{ are nonincreasing}$$

$$\|a_i\|^2 \in [0,1] \qquad \sum \|a_i\|^2 = k$$

The first line comes from the equation. The second line follows from Π being an orthogonal projector on a k-dimensional subspace.

It turns out that this system of equations is enough to show that the $\|a_i\|^2$ behave the way we want them to. We defer the details to Lemma A.2; the results are as follows.

$$\sum_{1}^{\ell(\sigma(1+\kappa))} \sigma_i^2(1-\|a_i\|^2) \le \varepsilon\Big(1+\frac{1}{\kappa}\Big)\sigma^2 \qquad \sum_{\ell(\sigma(1-\kappa))+1}^{N} \sigma_i^2\|a_i\|^2 \le \varepsilon\Big(\frac{1}{\kappa}-1\Big)\sigma^2$$

$$\sum_{1}^{\ell(\sigma(1+\kappa))} (1-\|a_i\|^2) \le \frac{\varepsilon}{\kappa} \qquad \sum_{\ell(\sigma(1-\kappa))+1}^{N} \|a_i\|^2 \le \frac{\varepsilon}{\kappa}$$

Now, applying the top two inequalities:

$$\|A\Pi - A_{\sigma,\kappa}\|_F^2 \le \frac{2\varepsilon\sigma^2}{\kappa} + \sum_{\ell(\sigma(1+\kappa))+1}^{\ell(\sigma(1-\kappa))} \sigma_i^2\|a_i-b_i\|^2.$$

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

$$\sum_{\ell(\sigma(1+\kappa))+1}^{\ell(\sigma(1-\kappa))} \sigma_i^2\|a_i-b_i\|^2 \le \sigma^2(1+\kappa)^2\|U^T(\Pi-\Pi_E)\|_F^2,$$

where U is the set of vectors $v_{\ell(\sigma(1+\kappa))+1}$ through $v_{\ell(\sigma(1-\kappa))}$.

* * *

Notice that Π_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 Π_E making the norm as small as possible. If $UU^T\Pi$ were an orthogonal projection, this would be easy:

$$\|U^T(\Pi - UU^T\Pi)\|_F^2 = 0.$$

However, this is likely not the case. $UU^T\Pi$ is *close* to an orthogonal projector, though, through the following reasoning:

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 the rest of the singular vectors. We are concerned with $P_2\Pi$.

Notice that $P_1+P_2+P_3 = I$. Further, from Lemma A.2, $\|(I-\Pi)P_1\|_F^2 \le \varepsilon/\kappa$ and $\|\Pi P_3\|_F^2 \le \varepsilon/\kappa$. Then

$$P_2\Pi = (I-P_1-P_3)\Pi = \Pi - P_1 + P_1(I-\Pi) - P_3\Pi$$
$$\|P_2\Pi - (\Pi - P_1)\|_F = \|P_1(I-\Pi) - P_3\Pi\|_F \le 2\sqrt{\varepsilon/\kappa}$$

So now it is sufficient to show that $\Pi - P_1$ is close to a projector matrix. This follows from Lemma A.1, since it satisfies the premise:

$$\begin{aligned}(\Pi-P_1)^2 - (\Pi-P_1) &= \Pi - \Pi P_1 - P_1\Pi + P_1 - \Pi + P_1 \\ &= (I-\Pi)P_1 + P_1(I-\Pi)\end{aligned}$$
$$\|(\Pi-P_1)^2 - (\Pi-P_1)\|_F \le 2\sqrt{\varepsilon/\kappa}$$

Thus, $UU^T\Pi$ is $(2\sqrt{\varepsilon/\kappa} + (2\sqrt{\varepsilon/\kappa} + 16\varepsilon/\kappa))$-close to an orthogonal projector in Frobenius norm.

* * *

We can choose Π_E to be M, and plug this into the original equation. We use the assumptions that $\varepsilon/\kappa < 1$ and $\kappa < 1$ to bound.

$$\begin{aligned}
\sum_{\ell(\sigma(1+\kappa))+1}^{\ell(\sigma(1-\kappa))} \sigma_i^2 \|a_i - b_i\|^2 &\le \sigma^2(1+\kappa)^2 \|U^T(\Pi - M)\|_F^2 \\
&\le \sigma^2(1+\kappa)^2 \|U^T(\Pi - (UU^T\Pi + E))\|_F^2 \\
&\le \sigma^2(1+\kappa)^2 \|U^T E\|_F^2 \\
&\lesssim \sigma^2(1+\kappa)^2 \varepsilon/\kappa \\
\|A\Pi - A_{\sigma,\kappa}\|_F^2 &\lesssim \frac{2\varepsilon\sigma^2}{\kappa} + \sigma^2(1+\kappa)^2\frac{\varepsilon}{\kappa} \\
&\lesssim \varepsilon\sigma^2/\kappa
\end{aligned}$$

This concludes the proof. (The constant factor is 1602.) □

Lemma A.1. If a Hermitian A satisfies $\|A^2 - A\|_F \le \varepsilon$, then $\|A - P\|_F \le \varepsilon + 4\varepsilon^2$ for some orthogonal projector P.

Proof. Use the fact that Hermitian matrices are normal, so $A = U\Gamma U^T$ for unitary U and diagonal matrix Γ, and

$$A^2 - A = U(\Gamma^2 - \Gamma)U^T \implies \|\Gamma^2 - \Gamma\|_F \le \varepsilon.$$

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

If Γ had only 0's and 1's on the diagonal, the resulting $U\Gamma U^T$ would be an orthogonal projector. □

Lemma A.2. The system of equations:

$$\sum_{i=1}^{k} \sigma_i^2 \le \sum_{i=1}^{N} \sigma_i^2 \|a_i\|^2 + \varepsilon\sigma_k^2 \qquad \sigma_i^2 \text{ are nonincreasing}$$
$$\|a_i\|^2 \in [0,1] \qquad \sum \|a_i\|^2 = k$$

imply the following, for $0 < \kappa \le 1$:

$$\sum_{1}^{\ell(\sigma_k(1+\kappa))} \sigma_i^2(1 - \|a_i\|^2) \le \varepsilon\Big(1 + \frac{1}{\kappa}\Big)\sigma_k^2 \qquad \sum_{\ell(\sigma_k(1-\kappa))+1}^{N} \sigma_i^2\|a_i\|^2 \le \varepsilon\Big(\frac{1}{\kappa} - 1\Big)\sigma_k^2$$

$$\sum_{1}^{\ell(\sigma_k(1+\kappa))} (1 - \|a_i\|^2) \le \frac{\varepsilon}{\kappa} \qquad \sum_{\ell(\sigma_k(1-\kappa))+1}^{N} \|a_i\|^2 \le \frac{\varepsilon}{\kappa}$$

Proof. We are just proving straightforward bounds on a linear system. We will continue to denote σ_k by σ. Thus, $k = \ell(\sigma)$.

The slack in the inequality is always maximized when the weight of the $\|a_i\|^2$ is concentrated on the large-value (small-index) entries. For example, the choice of $\|a_i\|^2$ maximizing slack in the given system of equations is the vector $\{\|a_i\|\}_{i\in[N]} = \mathbb{1}_{\leq k}$. Here, $\mathbb{1}_{\leq x}$ denotes the vector where

$$(\mathbb{1}_{\leq x})_i := \begin{cases} 1 & i \leq x \\ 0 & \text{otherwise} \end{cases}.$$

For brevity, we only give the details for the first bound; the others follow similarly. Consider adding the constraint $C = \sum_1^{\ell(\sigma(1+\kappa))} \sigma_i^2(1 - \|a_i\|^2)$ to the system of equations. We want to determine for which values of C the modified system is still feasible; we can do this by trying the values that maximize slack.

This occurs when weight is on the smallest possible indices: when $\|a_{\ell(\sigma(1+\kappa))}\|^2 = 1 - C/\sigma^2_{\ell(\sigma(1+\kappa))}$, $\|a_{\ell(\sigma)+1}\|^2 = C/\sigma^2_{\ell(\sigma(1+\kappa))}$, and all other $\|a_i\|^2$ are $\mathbb{1}_{\geq k}$. Notice that $\|a_{\ell(\sigma(1+\kappa))}\|^2$ could be negative and $\|a_{\ell(\sigma)+1}\|$ could be larger than one, breaking constraints. However, 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):

$$\begin{aligned} \sum_{i=1}^{k} \sigma_i^2 &\leq \sum_{i=1}^{k} \sigma_i^2 - C + C\frac{\sigma^2_{\ell(\sigma)+1}}{\sigma_{\ell(\sigma(1+\kappa))}} + \varepsilon\sigma^2 \\ C\Big(1 - \frac{\sigma^2_{\ell(\sigma)+1}}{\sigma_{\ell(\sigma(1+\kappa))}}\Big) &\leq \varepsilon\sigma^2 \\ C\Big(1 - \frac{1}{(1+\kappa)^2}\Big) &\leq \varepsilon\sigma^2 \end{aligned}$$

This gives the bound on C. Repeating for all four cases, we get the following bounds:

$$\sum_{1}^{\ell(\sigma(1+\kappa))} \sigma_i^2(1 - \|a_i\|^2) \leq \frac{\varepsilon(1+\kappa)^2\sigma^2}{2\kappa+\kappa^2} \qquad \sum_{\ell(\sigma(1-\kappa))+1}^{N} \sigma_i^2\|a_i\|^2 \leq \frac{\varepsilon(1-\kappa)^2\sigma^2}{2\kappa-\kappa^2}$$

$$\sum_{1}^{\ell(\sigma(1+\kappa))} (1 - \|a_i\|^2) \leq \frac{\varepsilon}{2\kappa+\kappa^2} \qquad \sum_{\ell(\sigma(1-\kappa))+1}^{N} \|a_i\|^2 \leq \frac{\varepsilon}{2\kappa-\kappa^2}$$

We get the bounds in the statement by simplifying the above (using that $\kappa \leq 1$). ∎

B Variant for an Alternative Model

In this section, we describe a variant of our recommendation systems algorithm for the competitive recommendations model, seen in Drineas, Kerenidis, and Raghavan's 2002 paper giving two algorithms for competitive recommendations [9]. The idea is to output good recommendations with as little knowledge about the preference matrix T as possible (hence the name). Our algorithm is similar to Drineas et al's second algorithm, which has weak assumptions on the form of T, but strong assumptions on how we can gain knowledge about it.

We use a similar model, as follows:

- We begin with no knowledge of our preference matrix T apart from the promise that $\|T - T_k\|_F \leq \rho\|T\|_F$;
- We can request the value of any particular entry T_{ij} for some cost;
- For some constant $0 < c \leq 1$, we can sample from and compute probabilities from a distribution p over $[m]$ satisfying

$$p(i) \geq c\frac{\|T_i\|^2}{\|T\|_F^2}.$$

  Further, we can sample from and compute probabilities from distributions q_i over $[n]$, for $i \in [m]$, satisfying

$$q_i(j) \geq c\frac{T_{ij}^2}{\|T_i\|^2}.$$

We discuss the first assumption in Section 5.1. The second assumption is very strong, but we will only need to use it sparingly, for some small set of users and products. In practice, this assumption could be satisfied through paid user surveys.

The last assumption states that the way that we learn about users naturally, via normal user-site interaction, follows the described distributions. For example, consider when T is binary (as in Section 5.1). The assumption about p states that we can sample for users proportional to the number of products they like (with possible error via c). Even though we don't know the exact number of products a user likes, it is certainly correlated with the amount of purchases/interactions the user has with the site. With this data we can form p. The assumption about q_i's states that, for a user, we can sample uniformly from the products that user likes. We can certainly assume the ability to sample from the products that a user likes, since such positive interactions are common, intended, and implicit in the user's use of the website. It is not clear whether uniformity is a reasonable assumption, but this can be made more reasonable by making T non-binary and more descriptive of the utility of products to users.

Under these assumptions, our goal is, given a user i, to recommend products to that user *that were not previously known to be good* and are likely to be good recommendations.

To do this, we run Algorithm 4 with T, k, ε as input, the main change being that we use Frieze, Kannan, and Vempala's standard algorithm instead of ModFKV. As samples and requests are necessary, we can provide them using the assumptions above.

For the FKV portion of the algorithm, this leads to $O(p^2)$ requests to p users about p products, where $p = O(\max\{\frac{k^4}{c^3\varepsilon^6}, \frac{k^2}{c^3\varepsilon^8}\})$. This gives the description of a D such that

$$\|T - D\|_F \leq \sqrt{\|T - T_k\|_F^2 + \varepsilon^2\|T\|_F^2} \leq (\rho + \varepsilon)\|T\|_F.$$

Thus, immediately we can use theorems from Section 5.1 to show that samples from D will give good recommendations.

From here, the next part of Algorithm 4 can output the desired approximate sample from D_i. A similar analysis will show that this approximate sample is likely to be a good recommendation, all while requesting and sampling a number of entries independent of m and n. Such requests and samples will only be needed for the p users chosen by FKV for its subsample, along with the input user. Further, for more recommendations, this process can be iterated with unused information about the p users chosen by FKV. Alternatively, if we can ask the p users for all of their recommendations, we only need $O(k^2 \log k/\varepsilon^2 \log 1/\delta)$ samples from the input user to provide that user with an unlimited number of recommendations (we can store and update the value of $\hat{V}$est to use when sampling).

This gives good recommendations, only requiring knowledge of $O(\text{poly}(k, 1/\varepsilon, \log 1/\delta))$ entries of T, and with time complexity polynomial in the number of known entries.