CS C191 Final Project Report

Aditya Ramabadran aditya.ramabadran@
Jonny Pei jonnypei@
Yuki Ito itoyuki@

May 8, 2023

Abstract

In this project, we investigate quantum recommendation systems that harness the power of quantum computing and matrix sampling to enhance the performance of personalized product recommendations (e.g. for users on platforms such as Amazon and Netflix). We dive into the mathematics behind the recommendation systems algorithm, exploring linear algebraic concepts such as Singular Value Decomposition (SVD), low-rank approximation, and quantum algorithms like phase estimation, quantum singular value estimation, and quantum projection; we also explore an interesting data structure to allow quantum access to our product data. We analyze how, by employing a low-rank assumption on preference matrices, Kerenedis and Prakash [11] develop an efficient quantum algorithm that achieves polylogarithmic runtime complexity in the matrix dimensions, providing an exponential improvement over classical recommendation systems at the time it was published. Additionally, we delve into and implement the related HHL algorithm, a quantum algorithm for solving linear systems, which also offers exponential speedup compared to classical algorithms but returns an expectation value rather than directly providing the solution.

1 Contribution Statement

Aditya Ramabadran: Abstract, Main Results, Techniques section 4.1 (Assumptions and Overview), Techniques section 4.4 (Quantum algorithms)

Jonny Pei: Main Results, Discussion, HHL Code Implementation, Appendix, and Formatting/Revisions.

Yuki Ito: Introduction, Techniques sections 4.2 (Approximation Bounds), Techniques sections 4.3 (Matrix Sampling), Techniques sections 4.5 (Correctness and Runtime Derivation), and corresponding Appendix proofs to include implicit steps glossed over in the paper.

2 Introduction

2.1 Motivation and Background

In the classical space, there has been extensive research on recommendation systems outside of matrix reconstruction because of its theoretical and practical aspect. Companies like Netflix and Amazon heavily rely on having a good recommendation system to have users continuously use its service. We provide a short survey of state-of-the-art classical recommendation systems in Appendix A.1.

2.2 Classical Recommendation System Problem

2.2.1 Problem Setup

The classical problem of recommendation systems goes like this: given m users with ratings of n products, and a user i to whom we want to recommend a product, the recommendation algorithm outputs a product that user i will most likely enjoy, even if they haven't used it before. Using this framework, we can organize such data as a matrix $P \in \mathbb{R}^{m \times n}$ called the preference matrix, where each $P_{i,j}$ indicates how much a user i prefers a product j. We define the preference for each element of the matrix to be $P_{i,j} \in [-10, 10] \cup [\text{"?"}]$, where a positive value corresponds to a positive rating, negative value to a negative rating, zero value to a neutral rating, and the "?" represents an unknown rating (i.e. user i haven't tried product j).

One of the most common ways to provide competitive recommendation systems is by re-framing this problem as a matrix completion problem (in which the paper [11] refers to as matrix reconstruction). This problem [4] is described as follows:

$$\min_{X \in \mathbb{R}^{m \times n}} \text{rank}(X) \text{ subject to } \forall (i,j) \in S : X_{i,j} = P_{i,j} \tag{1}$$

where we define $S = \{(i,j) \mid P_{i,j} \neq \text{"?"}\}$ (i.e. we want matrix X to have the same elements as P in places where there is a rating). In other words, the matrix completion problem tries to impute the missing values (i.e. non-rated product j by user i) of the preference matrix P by constructing a matrix X constrained by its rank. The reason why we can re-frame the recommendation problem as a matrix completion comes from a practical assumption we make about P, which we describe later in Section 2.2.3.

An important point to be made about re-framing the recommendation problem into matrix completion is that *matrix completion solves more than just recommendation.* Recommendation systems just require a high-value (i.e. high-rating) element from the optimal matrix $X^\star_{i,j}$ for index $\{i,j\}$ where $P_{i,j} =$ "?" for its output. On the other hand, matrix completion imputes all "?" elements in P that is, thus outputting the entire optimal matrix $X^\star$. Thus, matrix completion can be a bit "overkill" for this problem.

Furthermore an issue to note about matrix completion problem is that it is NP-hard. Therefore, the direct re-framing the recommendation problem into the matrix completion problem is impractical. Hence, for practical applications, we generally introduce relaxations to the matrix completion problem in Eq. 1.

2.2.2 Relaxation of Matrix Completion Problem:

There are usually two popular and effective ways to relax the matrix completion problem. In this section, we only explain one of the ways because it is the one that [11] references; for sake of space the other way is left to the appendix.

Non-convex Reformulation: one productive way to reformulate the matrix completion problem is as follows:

$$\min_{X \in \mathbb{R}^{m \times n}} \sum_{(i,j) \in S} (X_{i,j} - P_{i,j})^2 \text{ s.t. } \text{rank}(X) \leq k \tag{2}$$

Now, we introduce the observation operator $\mathcal{P}_S : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$:

$$(\mathcal{P}_S(M))_{i,j} = \begin{cases} (M)_{i,j}, & \text{if } (i,j) \in S \\ \text{"?"}, & \text{else} \end{cases} \tag{3}$$

where $M \in \mathbb{R}^{m \times n}$. The intuition behind this observation operator comes from the idea that there is a *true* preference matrix M that represents all the ratings for all users (including the ones they technically haven't tried / wrote a review for). Then, we perform low-rank approximation to obtain the true preference matrix using the preference matrix $P = \mathcal{P}_S(M)$. This intuition allows us to equivalently reformulate Eq. 2 as:

$$\min_{X \in \mathbb{R}^{m \times n}} ||\mathcal{P}_S(X - M)||_F^2 \text{ s.t. } \text{rank}(X) \leq k \tag{4}$$

Note that $\mathcal{P}$ is linear under the same set S (i.e. $\mathcal{P}_S(X - M) = \mathcal{P}_S(X) - \mathcal{P}_S(M)$).

$$\min_{X \in \mathbb{R}^{m \times n}} ||\mathcal{P}_S(X) - P||_F^2 \text{ s.t. } \text{rank}(X) \leq k \tag{5}$$

Here, we make an argument that even without the $\mathcal{P}_S$ operator on X, it would generally achieve a similar optimal $X^\star$. Thus,

$$\min_{X \in \mathbb{R}^{m \times n}} ||X - P||_F^2 \text{ s.t. } \text{rank}(X) \leq k \tag{6}$$

is our final relaxation.

With this reformulation, we note that the solution to this optimization problem can be solved via the Eckart-Young-Mirsky Theorem (Eq. 15).

2.2.3 The low-rank assumption

User preferences can generally be categorized as types. The assumption that all users fall under the small pool of categories, albeit a noisy one, allows us to say that our preference matrix has a small rank. This guarantees performance to use low-rank approximated matrices as our recommendation system. As this paper [11] cites, preference matrices coming from real data tend to have rank asymptotically smaller than the size of the matrix ($r \ll \min(m, n)$). Therefore, thinking about finding the smallest number of "types" (i.e. rank of matrix X) to encapsulate the data representative in P is a good assumption to make in competitive recommendation systems.

2.3 Runtime of Classical Recommendation Systems

Computing and storing the low-rank approximation of the Preference matrix P takes about $\text{poly}(mn)$ and $O(nk)$ respectively. When a specific user requires its recommendation on a product, such computation would take $O(nk)$ via projection to the top-k right singular vectors.

3 Main results

3.1 Main Algorithm: Quantum Recommendation System

In this section, we'll briefly cover the main algorithm, its input and output, and its runtime. In later sections, we'll further analyze the algorithm and its practical use, and compare it to classical recommendation systems algorithms.

The pseudocode describing the entire quantum recommendation algorithm is provided below in Algorithm 3.1.

Algorithm 3.1 Quantum Recommendation Algorithm

Require: Preference matrix $T \in \mathbb{R}^{m \times n}$ stored in a special online data structure (Section 3.1.1); and a user index i.

Ensure: Recommended product j for user i satisfying the bound in Theorem 4.2.

1: Generate the sub-sample matrix $\tilde{T} \in \mathbb{R}^{m \times n}$ by sampling from T with $p = \frac{16n}{\eta ||T||_F^2}$ where $\eta > 0$ is a bounded constant. We store $\tilde{T}$ in the special data structure.

2: Apply quantum projection (Section 4.4) on the sub-sample matrix $\tilde{T}$ to yield a correct quantum representation of the i-th row $\tilde{T}_i$ with $\sigma = \sqrt{\frac{\epsilon^2 p}{2k}} \|\tilde{T}\|_F$ and $\kappa = 1/3$

$$\left|\tilde{T}^{\dagger}_{\geq \sigma,\kappa} \tilde{T}_{\geq \sigma,\kappa} \tilde{T}_i\right\rangle = \left|(\tilde{T}_{\geq \sigma,\kappa})_i\right\rangle \approx \left|\tilde{T}_i\right\rangle$$

with probability $1 - 1/\text{poly}(n)$. We define the notation for $\tilde{T}_{\geq \sigma,\kappa}$ in Appendix A.2.

3: We measure $\left|(\tilde{T}_{\geq \sigma,\kappa})_i\right\rangle$ in the computation basis to yield a product j.

The input to the algorithm is the true preference matrix $T \in \mathbb{R}^{m \times n}$ where $T_{ij} \in \{0, 1\}$, and a user index i (to whom the algorithm recommends a product).

The output of the algorithm is the product j that is to be recommended to user i. This is obtained by sampling (i.e. measuring) the estimated state $\left|\tilde{T}^{\dagger}_{\geq \sigma,\kappa} \tilde{T}_{\geq \sigma,\kappa} \tilde{T}_i\right\rangle$, which is the projection of $\tilde{T}_i$ (the ith row of the subsampled matrix $\tilde{T}$) onto the subspace spanned by the row singular vectors with singular values greater than σ and some subset of row singular vectors with singular values in the interval $[(1 - \kappa)\sigma, \sigma)$. In particular, the quantum projection algorithm outputs the correct state with probability $1 - 1/\text{poly}(n)$.

The expected runtime of this algorithm is $O\left(\frac{\text{polylog}(mn)\sqrt{k}\|\tilde{T}_i\|^2}{\sqrt{p}\|\tilde{T}^{+}_{\geq \sigma} \tilde{T}_{\geq \sigma} \tilde{T}_i\|^2}\right)$, and the correctness probability is at least $1 - 1/\text{poly}(n)$. We define the notation for $\tilde{T}_{\geq \sigma}$ in Appendix A.2.

Note that this algorithm relies on subroutines (quantum projection, quantum singular value estimation). We will cover these subroutines in detail in Section 4.

3.1.1 Data Structure for Efficient Quantum Access

In this subsection, we highlight the key features and complexities of the data structure used to allow the previously described quantum algorithm to access the data.

Our inputs include a matrix $A \in \mathbb{R}^{m \times n}$ and a vector $x \in \mathbb{R}^n$, which are stored in an online classical data structure such that our quantum algorithm can create the state $|x\rangle$ and $|A\rangle_i$ (row i of A) in polylogarithmic time polylog(mn). In particular, our data structure is a memory model that contains information arriving into the system in the form of entries (i, j, A_{ij})—note that these matrix entries can arrive in any arbitrary order.

If w entries have arrived thus far, then our data structure is able to store these entries with the following properties:

- The size of the data structure is $O(w \cdot \log^2(mn))$.
- The time to store a new entry (i, j, A_{ij}) is $O(\log^2(mn))$.
- Quantum algorithms can "generate" the following structures each in time polylog(mn):
  - (i) A_i, a row of the matrix A.

(ii) A_{norms}, a vector containing the norms of each row A_i

Note that "generation" refers to being able to perform the unitary mapping $|i\rangle|0\rangle \to |i\rangle|A_i\rangle$ and the mapping $|0\rangle|j\rangle \to |A_{\text{norms}}\rangle|j\rangle$, where A_{norms} is a vector with entries $||A_i||$.

We summarize the technical details of the data structure in Section 4.4.1.

3.2 Reach Goal: Quantum Linear Systems Solver Code Implementation

In addition to digesting the quantum recommendation system algorithm, we also explore the quantum linear systems solver described in paper [10]. The paper presents the HHL algorithm, characterized by the following inputs, outputs, and runtime:

- Input: Sparse Hermitian $A \in \mathbb{R}^{N\times N}$ with condition umber κ, unit vector $\vec{b} \in \mathbb{R}^N$, and an observable $M \in \mathbb{R}^{N\times N}$.
- Output: $\vec{x}^\dagger M\vec{x} \pm \epsilon$, where $A\vec{x} = \vec{b}$ and $\epsilon > 0$.
- Runtime: $O(\text{poly}(\log N, \kappa, 1/\epsilon))$.

We provide a brief summary of technical details in (Appendix) and provide a naive code implementation in our github repository: `https://github.com/jonnypei/c191-final-project`. Note that we do not implement the most optimized version of HHL, which operates on special matrices[1] using trotter approximation. Our code is relatively well-documented, so feel free to look through the codebase for our design choices and implementation details.

4 Techniques

4.1 Assumptions and Overview of Techniques

In our main paper [11], the authors propose a novel quantum recommendation system leveraging a low-rank assumption on preference matrices to provide personalized recommendations in poly-logarithmic time concerning the matrix dimensions and polynomial in its rank. We model the preferences of m users on n products with an $m \times n$ matrix P, with P_{ij} representing the strength of preference of user i on product j. This matrix is not known a priori, and we gather information about it when users buy, review, or rate products. To make good recommendations with incomplete information, we assume that the preference matrix has a low-rank structure.

The problem is reframed as a matrix completion problem, with the best approximation derived from the Singular Value Decomposition (SVD) by the Eckart-Young theorem. Classical algorithms for matrix reconstruction require polynomial time in the matrix dimensions, but recommendation systems only need to sample from the resulting matrix. The proposed quantum recommendation systems algorithm demonstrated an exponential complexity improvement over known classical algorithms at the time, using a hidden preference matrix (with assumed low rank) T, where T_{ij} is boolean, indicating if the product is suitable for the user.

By demonstrating that if $\tilde{T}$ is a good approximation of T by Frobenius norm, the probability of a bad recommendation using $\tilde{T}$ is upper bounded, we can provide guarantees for average users. To account for variations in user preferences, we define parameters for typical users and the number of good recommendations for them. The problem is then reduced to sampling from a good approximation of the preference matrix T, using a subsampled matrix of T, called $\hat{T}$. The

[1] Tridiagonal Toeplitz symmetric real matrices

approximation error between the preference matrix and the subsampled matrix is small enough to use an approximation of the subsampled matrix. We formalize these bounds in the next section (Lemma 4.1).

The quantum part of the algorithm focuses on quantum projections, aiming to construct a data structure for storing matrix entries, perform quantum singular value estimation, and project the matrix onto the space spanned by its singular vectors with singular values higher than a threshold. The algorithm achieves time complexity $O(\text{polylog}(mn)\cdot\text{poly}(k))$ with a lower bound on the number of typical users, representing an exponential speedup over known classical recommendation system algorithms. However, this speedup was later matched by dequantized classical algorithms.

4.2 Approximation Bounds

Here we provide the bounds for our approximation that guarantees providing generally a *good* recommendation to analyze the correctness of the quantum recommendation algorithm.

Lemma 4.1. *Let $\tilde{T}_k$ be the sampled, rank k approximation of the hidden preference matrix T (i.e. the true preference matrix) such that $||T - \tilde{T}_k||_F \leq \epsilon||T||_F$. Then,*

$$\Pr[\tilde{T}_k \textit{ gives bad recommendation}] \leq \left(\frac{\epsilon}{1-\epsilon}\right)^2 \tag{7}$$

Proof. The proof is provided in Appendix C.2.1. □

This bound makes sense as the problem of providing good recommendation system boils down to constructing the optimal approximation that best represents the true preferences of all users.

Theorem 4.2. *Let S be a subset of the rows of T where $|S| \geq (1-\zeta)m$ for $\zeta > 0$. $\forall i \in S$,*

$$\frac{1}{1+\gamma}\frac{||T||_F^2}{m} \leq ||T_i||^2 \leq (1+\gamma)\frac{||T||_F^2}{m} \tag{8}$$

for some $\gamma > 0$ (i.e. the ℓ_2-norm of user i's preference is bounded by some expectation of the entire preference matrix per user. This describes the bound for typical-ness of a user for parameter γ). Again, let $\tilde{T}_k$ be an approximation such that $||T - \tilde{T}_k||_F \leq \epsilon||T||_F$.

Given above, there exist a subset $S' \subseteq S$ where $|S'| \geq (1-\delta-\zeta)m$ for $\delta > 0$ such that on average over the users in S', the probability that a sample from row i of $\tilde{T}_k$ is a bad recommendation is:

$$\Pr[\tilde{T}_k \textit{ gives bad recommendation for typical user } i \in S'] \leq \frac{\left(\frac{\epsilon}{1-\epsilon}\right)^2 \cdot (1+\epsilon)^2}{\left(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}}\right)^2 (1-\delta-\zeta)} \tag{9}$$

Proof. The proof is provided in Appendix C.2.2. □

The typical bound 4.2 allows us to construct a tighter bound depending on the average number of 'good' ratings users would give (i.e. typical-ness). For example, users with little to no 'good' ratings makes it hard for the recommendation system to output meaningful recommendations. Therefore, the more typical the user population is with a good low rank approximation $\tilde{T}_k$, the better the recommendation system would perform.

4.3 Matrix Sampling

We need a sampling requisite such that, when performing low-rank approximation on the sub-sampled matrix, would be as close to the low-rank approximation $\tilde{T}_k$ of the true preference matrix T, as possible. Since quantum algorithms can only construct $\tilde{T}_{\geq\sigma,\kappa}$ instead of the true $\tilde{T}_k$, we want to derive a bound in respect to $\tilde{T}_{\geq\sigma,\kappa}$.

Theorem 4.3 (Quantum-adapted bound for sub-sampled matrix)**.** *We know* $\max_{i,j} T_{ij} = 1$ *and define* $\tilde{T}$ *to be a random matrix obtained by sub-sampling w.p.* $p = \frac{16n}{\eta||T||_F^2} \geq \frac{36\sqrt{2}(nk)^{\frac{1}{2}}}{||T||_F\epsilon^3}$ *(*n *is the number of products, assume* $\eta > 0$*, and* $||T-T_k||_F \leq \epsilon||T||_F$ *where* T_k *is the* k*-th rank approximation of* T*). That is:*

$$\tilde{T}_{i,j} = \begin{cases} \frac{(T)_{i,j}}{p}, & w.p.\ p = \frac{16n}{(\eta||T||_F)^2} \\ 0, & else \end{cases} \tag{10}$$

Let $\mu > 0$ *be a threshold parameter and denote* $\sigma = \sqrt{\frac{\mu}{k}}||\tilde{T}||_F$*. Let* $\kappa > 0$ *be a precision parameter.*

Given above, for quantum-calculated k*-rank approximation,* $\tilde{T}_{\geq\sigma,\kappa}$*, we have that*

$$||T - \tilde{T}_{\geq\sigma,\kappa}||_F^2 \leq 3||T - T_k||_F + (3\sqrt{\eta}k^{\frac{1}{4}}\mu^{-\frac{1}{4}}[2 + (1-\kappa)^{-\frac{1}{2}}] + (3-\kappa)\sqrt{\frac{2\mu}{p}})||T||_F \tag{11}$$

with probability at least $1 - e^{-19(\log n)^4}$*,*

For concreteness, suppose $\kappa = \frac{1}{3}, \mu = \frac{\epsilon^2 p}{2}$*, and* $\eta \in \left(0, \frac{2n^{1/4}\epsilon^{3/2}}{3(2k)^{1/4}||T||_F^{1/2}}\right]$*. Then, we have:*

$$||T - \tilde{T}_{\geq\sigma,\kappa}||_F \leq \cdots \leq 9\epsilon||T||_F \tag{12}$$

(i.e. a linear bound similar to the one in Lemma 4.1 (say $\epsilon' \coloneqq 9\epsilon$*), suggesting* $\tilde{T}_{\geq\sigma,\kappa}$ *is a good approximation given correct parameterization).*

Proof. The proof is provided in Appendix C.3.2. □

Given sufficient parameterization on our sub-sampling procedure, we can essentially construct $\tilde{T}_{\geq\sigma,\kappa}$ instead of k-th rank approximation of T (i.e. $\tilde{T}_k$ or even T_k) for our recommendation system to remain competitive. The next section will discuss how to construct the state $|\tilde{T}_{\geq\sigma,\kappa}\rangle$ that has previously been glossed over as the "quantum algorithm."

4.4 Quantum Algorithms

Now that we've gone over the preliminaries and matrix sampling, we move to the key quantum components of the algorithm.

4.4.1 Data Structure

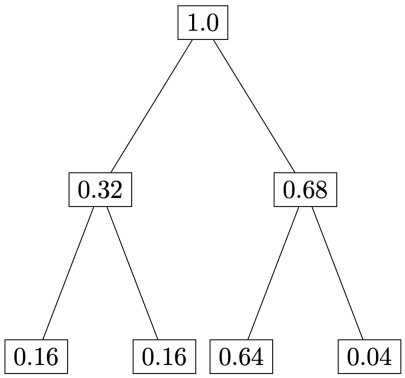
But first, let's revisit the data structure we require for quantum projection to have polylogarithmic time complexity. Earlier in Section 3.1.1, we briefly described the requirements and capabilities of our data structure. Here, we summarize how this data structure is designed to meet those needs.

First, each row of the matrix (a vector in $\mathbb{R}^n$) corresponds to a $2n$-length array stored as a complete binary tree (with n leaves and depth at most $\log n$). The leaves hold the individual amplitudes of the vector and their sign (e.g. A_{ij}^2 and the sign of A_{ij}). Then, the internal nodes above hold the sum of the squares of the amplitudes of its child leaves. Therefore, the root node

will store the sum of the squares of all the amplitudes of the vector, which is $||A_i||^2$. We can also use this same data structure in the same way for vector $|x\rangle$.

Note that this follows the time and space complexity we required earlier of our data structure, for processing and storing new entries. However, we also need to be able to perform the mapping $|i\rangle|0\rangle \to |i\rangle|A_i\rangle$ if our algorithm has quantum access to this data structure (the mapping for the norms also reduces to being able to do this mapping). This state preparation procedure involves applying a series of $\lceil \log n \rceil$ conditional rotations to the initial state $|0\rangle^{\lceil \log n \rceil}$. Each rotation requires two quantum queries to the data structure (querying the two children of each node in the superposition).

An example is as follows: Suppose our state is $|\phi\rangle = 0.4|00\rangle + 0.4|01\rangle + 0.8|10\rangle + 0.2|11\rangle$. Then our corresponding binary tree would look like this:



Now, starting from $|0\rangle|0\rangle$, we want to generate $|\phi\rangle$. To do this, we will start by applying a rotation on qubit 1:

$$|0\rangle|0\rangle \to (\sqrt{0.32}|0\rangle + \sqrt{0.68}|1\rangle)|0\rangle$$

Note that the amplitudes in the children of the root of the binary tree are used in this unitary rotation. Now, we need to apply a conditional rotation; specifically, a rotation on qubit 2 conditioned on qubit 1:

$$(\sqrt{0.32}|0\rangle + \sqrt{0.68}|1\rangle)|0\rangle \to \sqrt{0.32}|0\rangle \frac{1}{\sqrt{0.32}}(0.4|0\rangle + 0.4|1\rangle) + \sqrt{0.68}|1\rangle \frac{1}{\sqrt{0.68}}(0.8|0\rangle + 0.2|1\rangle)$$

$$= 0.4|00\rangle + 0.4|01\rangle + 0.8|10\rangle + 0.2|11\rangle = |\phi\rangle$$

Note that we again use the square roots of the two children of the node in each term. Doing this, we are able to obtain $|\phi\rangle$ as needed. The general formula for this algorithm is as follows: conditioned on the first register being $|i\rangle$ and the first t qubits being in state $|k\rangle$, our rotation for the $(t+1)$th qubit is:

$$|i\rangle|k\rangle|0\rangle \to |i\rangle|k\rangle \frac{1}{\sqrt{B_{i,k}}}(\sqrt{B_{i,k0}}|0\rangle + \sqrt{B_{i,k1}}|1\rangle)$$

where $B_{i,k}$ is the amplitude stored at an internal node of B_i at depth t, corresponding to $k \in \{0,1\}^t$.

4.4.2 Quantum Singular Value Estimation

The second tool required for the projection algorithm is an efficient quantum algorithm for singular value estimation.

The paper presents an algorithm that, given a matrix $A \in R^{m \times n}$ with its singular value decomposition $A = \sum_i \sigma_i u_i v_i^t$ stored in the data structure described earlier, can estimate the singular values corresponding to each singular vector in coherent superposition. The algorithm has a running time of $O(\text{polylog}(mn)/\epsilon)$, where $\epsilon > 0$ is the precision parameter. The algorithm maps the state $|x\rangle = \sum_i \alpha_i |v_i\rangle$ to $\sum_i \alpha_i |v_i\rangle|\sigma_i\rangle$, with $\sigma_i \in \sigma_i \pm \epsilon |A|_F$ for all i, achieving a success probability of at least $1 - 1/\text{poly}(n)$.

The algorithm finds isometries $P \in R^{mn \times m}$ and $Q \in R^{mn \times n}$ that can be efficiently applied, such that $A/|A|F = P^t Q$. We then define a unitary matrix W acting on R^{mn}, which is also efficiently implementable, and maps the row singular vector v_i of A with singular value σ_i to an eigenvector Qv_i of W with eigenvalue $e^{i\theta_i}$, where $\cos(\theta_i/2) = \sigma_i/|A|F$. The unitary matrix W is given by the product of reflections U and V, with $U = 2PP^t - I_{mn}$ and $V = 2QQ^t - I_{mn}$ being reflections (implementable in time $O(\text{polylog}(mn))$).

The algorithm consists of the following steps:

Algorithm 4.2 Quantum singular value estimation

Require: $A \in R^{m \times n}$, $x \in R^n$ in the data structure described earlier, and precision parameter $\epsilon > 0$.

1: Create state $|x\rangle = \sum_i \alpha_i |v_i\rangle$, where v_i are the singular vectors.
2: Append a first register $|0^{\lceil \log m \rceil}\rangle$.
3: Create the state $|Qx\rangle = \sum_i \alpha_i |Qv_i\rangle$.
4: Perform quantum phase estimation (with precision parameter $2\epsilon > 0$) on the input $|Qx\rangle$ for the unitary $W = U \cdot V$ where U, V are the reflection unitaries defined earlier. This yields $\sum_i \alpha_i |Qv_i, \theta_i\rangle$.
5: Compute $\overline{\sigma}_i = \cos(\overline{\theta}_i/2)||A||_F$ where $\overline{\theta}_i$ is the estimate from phase estimation. Uncompute the output of the phase estimation.
6: Invert step 3's transformation. This yields $\sum_i \alpha_i |v_i\rangle|\sigma_i\rangle$.

Further analyzing the algorithm, we note that the reflection matrices U and V, derived from P and Q, are used to create the unitary matrix W efficiently. The isometry Q maps row singular vectors of A to eigenvectors of W with specific eigenvalues, resulting in a crucial property: $\sigma_i = \cos(\theta_i/2)||A||_F$, where θ_i are the eigenvalues of W and σ_i are the singular values of A. This important relationship between the eigenvalues of W and the singular values of A enables the estimation of singular values. By performing quantum phase estimation on the input $|Qx\rangle$ for the unitary W, the algorithm obtains the eigenvalue estimates, which are then used to compute the singular values. This approach allows for efficient and precise estimation of singular values.

4.4.3 Quantum Projection with Threshold

The quantum projection algorithm estimates the projection of a vector x onto the subspace spanned by the union of the row singular vectors of a matrix A whose corresponding singular values are larger than a threshold σ (as well as a subset of row singular vectors with singular values in the interval $[(1 - \kappa)\sigma, \sigma)$.

Given a matrix A and parameters σ, $\kappa > 0$, the algorithm works as follows:

Algorithm 4.3 Quantum Projection with Threshold

Require: $A \in \mathbb{R}^{m\times n}, x \in \mathbb{R}^n$ in the data structure described earliers, and parameters $\sigma, \kappa > 0$.

1: Create $|x\rangle = \sum_i \alpha_i |v_i\rangle$, where v_i are the singular vectors.

2: Apply the singular value estimation algorithm described earlier on $|x\rangle$ (with precision $\epsilon = \frac{\kappa\sigma}{2\|A\|_F}$). This yields the state $\sum_i \alpha_i |v_i\rangle|\overline{\sigma}_i\rangle$.

3: Create a new second register. Apply unitary V mapping $|t\rangle|0\rangle \to |t\rangle|1\rangle$ if $t < \sigma - \frac{\kappa^2\sigma}{2}$, and $|t\rangle|0\rangle \to |t\rangle|0\rangle$ (the identity) otherwise, to get the state $\sum_{i\in S} \alpha_i|v_i\rangle|\overline{\sigma}_i\rangle|0\rangle + \sum_{i\in\overline{S}} \alpha_i|v_i\rangle|\overline{\sigma}_i\rangle|1\rangle$, where S is the union i's such that $\sigma_i \geq \sigma$ (as well as some i's with $\sigma_i \in [(1-\kappa)\sigma, \sigma)$)

4: Apply the singular value estimation algorithm on this state to erase the second register:

$$\sum_{i\in S} \alpha_i|v_i\rangle|0\rangle + \sum_{i\in\overline{S}} \alpha_i|v_i\rangle|1\rangle = \beta|A^+_{\geq\sigma,\kappa}A_{\geq\sigma,\kappa}x\rangle|0\rangle + \sqrt{1-|\beta|^2}|A^+_{\geq\sigma,\kappa}A_{\geq\sigma,\kappa}x\rangle^\perp|1\rangle$$

, with $\beta = \frac{\|A^+_{\geq\sigma,\kappa}A_{\geq\sigma,\kappa}x\|}{\|x\|}$

5: Measure the second register in the standard basis. If the outcome is $|0\rangle$, output the first register and exit. Otherwise, repeat step 1. Note that upon termination, the output of the first register is $|A^+_{\geq\sigma,\kappa}A_{\geq\sigma,\kappa}x\rangle$ as desired.

The algorithm outputs $|A^+_{\geq\sigma,\kappa}A_{\geq\sigma,\kappa}x\rangle$ with probability at least $1 - \frac{1}{\text{poly}(n)}$ and in expected time $O(\text{polylog}(mn)\frac{\|A\|_F\|x\|^2}{\sigma|A^+_{\geq\sigma,\kappa}A_{\geq\sigma,\kappa}x|^2})$. Note that this is the projection of x onto the space spanned by a set of high singular value singular vectors of the (subsampled) matrix A. In the context of recommendation systems, for most users, the ratio $\frac{|A^+\geq\sigma A\geq\sigma x|^2}{|x|^2}$ is constant, leading to a running time that is polynomial in k and polylogarithmic in the matrix dimensions. Also, measuring the resulting state here would then yield a product j we can recommend to a user i, if we use matrix $\hat{T}$ and vector $\hat{T}_i$ in this algorithm.

4.5 Correctness and Runtime Derivation

4.5.1 Correctness

The output of the Quantum Recommendation Algorithm, w.p. at least $1-\frac{1}{\text{poly}(n)}$, is $|\tilde{T}^\dagger_{\geq\sigma,\kappa}\tilde{T}_{\geq\sigma,\kappa}\tilde{T}_i\rangle = |(\tilde{T}_{\geq\sigma,\kappa})_i\rangle$ (i.e. i-th row of $\tilde{T}_{\geq\sigma,\kappa}$). As mentioned in 4.3, setting appropriate parameters gives us a bound $||T - \tilde{T}_{\geq\sigma,\kappa}||_F \leq 9\epsilon||T||_F$ w.p. at least $1 - e^{-19(logn)^4} = 1 - \text{poly}(n)$. Then, either using the approximate bounds 4.1 or 4.2, by setting $\epsilon' \coloneqq 9\epsilon$, we can observe the overall correctness of our quantum recommendation system to be sufficient for providing good recommendations.

4.5.2 Runtime

Theorem 4.4. *For at least $(1-\xi)(1-\delta-\zeta)m$ users in the subset S' (for some $\xi > 0$), we have that the expected running time of our quantum recommendation system 3.1 is $O(polylog(mn)poly(k))$.*

This is the result of 4.2 and the runtime of 4.3 in addition to computing the expectation of providing a good recommendation. ξ is introduced as a general bound of confidence through Markov's Inequality. Worst case, we will run this experiment for $\log(n)$ times more to converging to "ensure" a good recommendation.

5 Discussion

In this section, we discuss the strengths and weaknesses of the previously described quantum recommendation system and its involved techniques. We also highlight current extensions upon this machine learning model, and suggest ideas for future work in the field.

5.1 Strengths

The quantum recommendation system described in Algorithm 3.1 provides several substantial improvements over similar classical recommendation system algorithms.

In terms of time complexity, this quantum version provides an exponential speedup over the classical version with respect to matrix reconstruction methods. As mentioned in Appendix A.1, there are deep learning methods (involving compressed/efficient neural networks) that have competitive time complexity during inference, but training yields a super-exponential slowdown compared to the quantum algorithm. Also by solving the matrix reconstruction problem quantumly, it allows for interpretability of its output state $|\tilde{T}_{\geq\sigma,\kappa}\rangle$, which a lot of deep learning methods lack.

With respect to space complexity, the quantum algorithm uses the same amount of space as classical methods, ignoring polylogarithmic factors. However, the quantum version also allows efficient online updating of the preference matrix due to its unique storage method (Section 4.4.1).

5.2 Weaknesses

Despite its strengths, Algorithm 3.1 has some limitations, mainly due to its correctness guarantees.

We can only prove that the quantum algorithm produces good recommendations with high probability for the typical user, and not necessarily any user (Lemma 4.1, Theorem 4.2). However, this issue can be mitigated by recommending a small number of products to a given user, instead of just a single product.

The main underlying assumption that enables the correctness of this algorithm is that the preference matrix can be well approximated by a low-rank matrix. In practice, while in many cases most users can agree that there are some well-defined "high quality" products, there are also many instances where users may have very diverse preferences and there may be more than a few categories. In the latter scenarios, the quantum algorithm (as well as classical recommendation algorithms that heavily involve matrix completion) may perform poorly, and are outclassed by more powerful models in the deep learning field.

5.3 Another Quantum Recommendation Algorithm

In this section, we highlight Gao and Yang's quantum recommender system [7] and Wang et. al's context-aware recommendation system. They also aim to efficiently solve the (relaxed) matrix reconstruction problem—just with a slightly different objective.

They aim to solve the following problem: given preference matrix $P \in \mathbb{R}^{m\times n}$, we want to find two low-rank regularized factor matrices $X \in \mathbb{R}^{k\times m}$ and $Y \in \mathbb{R}^{k\times n}$ such that $X^\dagger Y = \tilde{P} \approx P$. Here, X and Y serve to expose the underlying latent structure of P. In particular, they aim to solve the following optimization problem called Alternating Least Squares (ALS):

$$\min_{X,Y} \|P - X^\dagger Y\|_F^2 + \gamma(\|X\|_F^2 + \|Y\|_F^2) \tag{13}$$

where γ is a regularization constant.

They provide a direct quantum analog to the iterative classical ALS algorithm, yielding an exponential speedup under certain assumptions on the condition numbers of X, Y.

Overall, we liked the way they relaxed the matrix completion problem and their (relatively) direct approach to quantum-izing the classical algorithm.

5.4 Dequantization

Back in 2018, Ewin Tang published developed a classical algorithm that delivers an equivalent runtime to the QML algorithm we explored previously [11]. The main result of her paper [14] is as follows:

Theorem 5.1 (Tang [14])**.** *Suppose we are given as input a matrix $A \in \mathbb{R}^{m \times n}$ supporting query and ℓ^2-norm sampling operations, a row (i.e. user) $i \in [m]$, a singular value threshold σ, an error parameter $\eta > 0$, and a sufficiently small $\epsilon > 0$. There exists a classical algorithm whose output distribution is ϵ-close in total variation distance to the distribution given by ℓ^2-norm sampling from the i-th row of a low-rank approximation D or A in query and time complexity*

$$O\left(\text{poly}\left(\frac{\|A\|_F}{\sigma}, \frac{1}{\epsilon}, \frac{1}{\eta}, \frac{\|A_i\|_2}{\|D_i\|_2}\right)\right)$$

where the quality of D depends on η, ϵ.

Note that this runtime is independent of m, n. σ Since $(\|A\|_F/\sigma)^2$ is a bound on k, we can think of σ as approximately $\|A\|_F/\sqrt{k}$. Then, due to Tang's use of a special data structure (which is very similar to the one in [11]), we must consider an extra $O(\log(mn))$ overhead factor. This yields a time complexity of

$$\tilde{O}\left(\frac{\|A\|_F^{24}}{\sigma^{24}\epsilon^{12}\eta^6}\log(mn)\frac{\|A_i\|^2}{\|D_i\|^2}\right) = O(\text{poly}(k)\log(mn))$$

which is asymptotically equivalent to Kerenidis and Prakash's algorithm, but does have much larger exponent factors. The main idea of Tang's algorithm is to replace the state preparation assumptions in the QML with ℓ^2-norm sampling assumptions in the classical algorithm. This way, the algorithm is able to efficiently manipulate and approximate matrices. The

Tang, along with many other researchers, have since then harnessed the power of classical sampling assumptions to dequantize other quantum linear algebra and machine learning tasks in [5]. One thing to note, though, is that sometimes these dequantized machine learning algorithms can perform quite poorly when used on practical datasets, mainly due to their stringent assumptions and high constant factors.

5.5 Future Directions

In this section, we discuss a couple direct extensions of the QML algorithm and suggest ideas for future research.

5.5.1 Quantum context-aware recommendation systems based on tensor singular value decomposition

The context-aware quantum algorithm for recommendation systems that Wang et al. extends [11] from 2D matrices to third-order tensors. They argue that the (2D) matrix factorization/reconstruction

approach fails to model context information (e.g. time) and their method of third-order tensor reconstruction provides a robust context-aware recommendation system. We briefly describe their overall method below:

Wang et al. aim to approximate the third-order preference tensor $\mathcal{T}$ with dimension N by using truncated tensor singular value decomposition (t-svd) of the subsample tensor. Their method is relatively similar structure to [11] in that Quantum Singular Value Estimation (QSVE) is the main component of their approximation/reconstruction method—most notably, they take advantage of quantum parallelism to optimize QSVE.

Their quantum algorithm achieves time complexity $O(\sqrt{k}N\text{polylog}(N))$, substantially faster than the classical counterpart with complexity $O(kN^3)$, where k is the truncated tubal rank of $\mathcal{T}$.

We think this paper is a super interesting extension of [11], and wonder if there are other ways to represent "context" beyond just considering an additional dimension.

5.5.2 Exponential Advantages in Machine Learning through Feature Mapping

In their survey paper [13], Nader et al. describe how we can extend the idea of preprocessing and data assumptions to obtain exponential speedups in QML algorithms. In particular, they highlight a couple quantum algorithms that use quantum enhanced feature maps to obtain a provably exponential advantage over classical counterparts.

5.6 Our Thoughts

Based off of reading all these papers, a two common themes we find are:

1. Many quantum algorithms that use some special state preparation or data representation can be reasonably replicated in classical algorithms via some form of sampling. Also, many quantum algorithms make lots of assumptions or probabilistic assumptions about the data or outputs in general, which make them not very applicable in practice if there are numerous confounding variables and priors to consider experimentally.

2. A good way of achieving provably exponential improvements over classical algorithms is by exploiting the hardness of classically intractable problems (e.g. the discrete logarithm problem). The main example we saw of this the feature/kernel mappings in [13].

References

[1] D. Achlioptas and F. McSherry. Fast computation of low rank matrix approximations in proceedings of the thirty-third annual acm symposium on theory of computing. *ACM*, pp. 611–618., 2001.

[2] Ziheng Chen, Fabrizio Silvestri, Jia Wang, Yongfeng Zhang, and Gabriele Tolomei. The dark side of explanations: Poisoning recommender systems with counterfactual examples. 2023.

[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. *CoRR*, abs/1606.07792, 2016.

[4] Yuejie Chi. Low-rank matrix completion. *IEEE Signal Processing Magazine*, 2018.

[5] Nai-Hui Chia, András Gilyén, Tongyang Li, Han-Hsuan Lin, Ewin Tang, and Chunhao Wang. Sampling-based sublinear low-rank matrix arithmetic framework for dequantizing quantum machine learning. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. ACM, jun 2020.

[6] Fuchang Gao. ae^2i: A double autoencoder for imputation of missing values. 2023.

[7] Shang Gao and Yu-Guang Yang. A novel quantum recommender system. *Physica Scripta*, 98, 12 2022.

[8] Lov Grover and Terry Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions, 2002.

[9] Pegah Sagheb Haghighi, Olurotimi Seton, and Olfa Nasraoui. An explainable autoencoder for collaborative filtering recommendation. 2019.

[10] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15), oct 2009.

[11] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. *ArXiv*, abs/1603.08675, 2016.

[12] Chen Li, Yang Cao, Ye Zhu, Debo Cheng, Chengyuan Li, and Yasuhiko Morimoto. Ripple knowledge graph convolutional networks for recommendation systems. 2023.

[13] Andrew Nader, Kyle Oppenheimer, G. Tom, and December. Exponential advantages in quantum machine learning through feature mapping. 2020.

[14] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. ACM, jun 2019.

A Introduction

A.1 State-of-the-Art Classical Recommendation Systems

State-of-the-art recommendation systems lean on deep-learning methods which is way beyond what quantum recommendation system [11] aims to perform (i.e. by replicating the matrix reconstruction algorithm). This Google research paper [3], published the same year as [11], encapsulates statistical theory and generalized linear models with wide & deep networks to construct recommendation systems that significantly increased acquisition in their products offered in Google Play. All of these methods are examples of collaborative filtering (CF) - meaning they utilize cross-user data to recommend a product to unrated users.

More recently, many graph convolutional networks (specifically, knowledge graph convolutional networks) [12], and autoencoders [6] are common implementations of recommendation systems. Something to note is that many of these neural network implementations, given its black box nature, lack interpretability and can be susceptible to the idea of counterfactual examples - which may be exploited by malicious actors [2]. Though there have been works to address interpretability in various implementations like autoencoders [9], it is a challenge state-of-the-art recommendation systems have in exchange for its remarkable performance.

A.2 Linear Algebra Preliminaries

- Let $A \in \mathbb{R}^{m\times n}$ have rank r. A *singular value decomposition (SVD)* of A is the form:

$$A = U\Sigma V^T = \begin{bmatrix} U_r & U_{m-r} \end{bmatrix} \begin{bmatrix} \Sigma_r & 0_{r\times(n-r)} \\ 0_{(m-r)\times r} & 0_{(m-r)\times(n-r)} \end{bmatrix} \begin{bmatrix} V_r^T \\ V_{n-r}^T \end{bmatrix} = U_r\Sigma_r V_r^T = \sum_{i=1}^{r} \sigma_i \vec{u}_i \vec{v}_i^T \quad (14)$$

  where $U = \begin{bmatrix} \vec{u}_1 & ... & \vec{u}_m \end{bmatrix} \in \mathbb{R}^{m\times m}$ where $\vec{u}_i \in \mathbb{R}^m, U_r \in \mathbb{R}^{m\times r}, U_{m-r} \in \mathbb{R}^{m\times(m-r)}, V = \begin{bmatrix} \vec{v}_1 & ... & \vec{v}_n \end{bmatrix} \in \mathbb{R}^{m\times m}$ where $\vec{v}_i \in \mathbb{R}^n \in \mathbb{R}^{n\times n}, V_r \in \mathbb{R}^{n\times r}$, and $V_{n-r} \in \mathbb{R}^{n\times(n-r)}$ are orthonormal matrices. $\vec{u}_i$ is defined as *left singular vectors* and $\vec{v}_i$ is defined as *right singular vectors.* $\Sigma \in \mathbb{R}^{m\times n}$ is defined as a composite of $\Sigma_r = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$ and zero matrices where σ_i is defined as the *singular values* ($\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > 0$ for convention).

- Let $A \in \mathbb{R}^{m\times n}$ and have rank $r \leq \min(m, n)$. For $\ell \leq r$, define $A_\ell := \sum_{i=1}^{\ell} \sigma_i \vec{u}_i \vec{v}_i^T$. *Eckart-Young-Mirsky Theorem* states:

$$A_\ell \in \underset{B\in\mathbb{R}^{m\times n}}{\arg\min} ||A - B||_F^2 \text{ s.t. } \text{rank}(B) \leq \ell \quad (15)$$

  where $||C||_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(C)_{i,j}^2 = \sum_{i=1}^{\text{rank}(C)} \sigma_i^2(C)$ is defined as the Frobenius norm squared.

- Let $A = \sum_i \sigma_i u_i v_i^\dagger$. We define the following notation:

$$A_{\geq\sigma} = \sum_{\sigma_i\geq\sigma} \sigma_i u_i v_i^\dagger \quad (16)$$

$$A_{\geq\sigma,\kappa} = \sum_{\sigma_i\geq\sigma \text{ and some } \sigma_i\in[(1-\kappa)\sigma,\sigma)} \sigma_i u_i v_i^\dagger \quad (17)$$

A.3 Convex Relaxation of the Matrix Completion Problem

This is a method motivated by the success of of ℓ_1 norm minimization for sparse recovery ℓ_0 norm minimization in compressed sensing (Note that $||\vec{v}||_1 = \sum_{i=1}^{n} |\vec{v}_i|$ and $||\vec{v}||_0 =$ number of non-zero elements in $\vec{v}$ given $\vec{v} \in \mathbb{R}^n$). Think of our current matrix completion objective function, rank(X) as the equivalent ℓ_0 norm. Then, suppose the ℓ_1 norm equivalent of the matrix completion problem as the nuclear norm of X defined as $||X||_\star \coloneqq \sum_{i=1}^{\min(m,n)} \sigma_i(X)$. Nuclear norm is considered as the tightest convex relaxation of the rank constraint and the optimization problem would be:

$$\min_{X \in \mathbb{R}^{m \times n}} ||X||_\star \text{ s.t. } \forall (i,j) \in S : X_{i,j} = P_{i,j} \tag{18}$$

As an aside, the nuclear norm, as mentioned, can be represented as a solution to a semidefinite program (i.e. convex optimization problem):

$$||X||_\star = \min_{W_1 \in \mathbb{R}^{m \times m}, W_2 \in \mathbb{R}^{n \times n}} \frac{1}{2}(\text{Tr}(W_1) + \text{Tr}(W_2)) \text{ s.t. } \begin{bmatrix} W_1 & X \\ X^T & W_2 \end{bmatrix} \succeq 0 \tag{19}$$

where $\succeq 0$ denotes positive semi-definiteness of the matrix.

B HHL Algorithm

We provide an overview of the HHL linear systems solver in Algorithm 2.4

Algorithm 2.4 HHL Algorithm [10]

Require: s-sparse[2] Hermitian $A \in \mathbb{R}^{N \times N}$ with condition number κ, unit vector $\vec{b} \in \mathbb{R}^N$, and an observable $M \in \mathbb{R}^{N \times N}$

Ensure: $\vec{x}^\dagger M \vec{x} \pm \epsilon$, where $A\vec{x} = \vec{b}$ and $\epsilon > 0$ in expected time $\tilde{O}(\log(N)s^2\kappa^2/\epsilon)$.

1: Efficiently prepare $|b\rangle \in \mathbb{C}^N$ (perform $|0\rangle \mapsto |b\rangle$) using e.g. [8].

2: Apply Quantum Phase Estimation (QPE) with $U = e^{iAt} \coloneqq \sum_{j=1}^{N} e^{i\lambda_j t} |u_j\rangle \langle u_j|$ to decompose $|b\rangle$ in the eigenbasis of A, yielding an approximation of

$$\sum_{j=1}^{N} b_j |\lambda_j\rangle |u_j\rangle$$

where $(\lambda_j, |u_j\rangle)$ are eigenvalue-eigenvector pairs of A.

3: Add an auxiliary qubit and a rotation condition on the eigenvectors $|\lambda_j\rangle$, yielding

$$\sum_{j=1}^{N} b_j |\lambda_j\rangle |u_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right)$$

where C is a normalization constant s.t. $|C| < \lambda_{\min}$.

4: Apply Inverse QPE to yield an approximation of

$$\sum_{j=0}^{N-1} b_j |0\rangle |u_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle \right)$$

5: Measure the auxiliary qubit in the computational basis. If the outcome is 1, the post-measurement state is approximately

$$\frac{1}{\sqrt{\sum_{j=1}^{N} |b_j|^2/|\lambda_j|^2}} \sum_{j=1}^{N} \frac{b_j}{\lambda_j} |u_j\rangle$$

which corresponds to $|x\rangle = \sum_{j=1}^{N} b_j \lambda_j^{-1} |u_j\rangle$ up to normalization.

6: Apply observable M to yield $\langle x| M |x\rangle \pm \epsilon$.

Note: the error ϵ comes from performing conditional Hamiltonian evolution and choosing approximation scalings during QPE.

C Techniques

C.1 Quantum Information Preliminaries

- The vector state $|x\rangle$ for $\vec{x} \in \mathbb{R}^n$ is defined as $\frac{1}{||x||_2}\sum_{i\in[n]} x_i|i\rangle$

  Note: You would observe outcome i w.p. $\frac{x_i^2}{||x||_2^2}$

- The matrix state $|A\rangle$ for $A \in \mathbb{R}^{m\times n}$ is defined as $\frac{1}{||A||_F}\sum_{i\in[m],j\in[n]} A_{i,j}(|i\rangle \otimes |j\rangle)$

- Phase Estimation: Let U be a unitary operator such that $U|v_j\rangle = e^{i\theta_j}|v_j\rangle$ for $\theta_j \in [-\pi, \pi]$ (definition of eigenvalue, eigenvectors). For a precision parameter $\epsilon > 0$, there exist a quantum algorithm that runs in time $O(T(U)\log(n/\epsilon))$ (where $T(U)$ is the time to implement the unitary U) and w.p. $1 - \frac{1}{\text{poly}(n)}$ such that it maps the state $\sum_{j\in[n]}\alpha_j|v_j\rangle \to \sum_{j\in[n]}\alpha_j(|v_j\rangle \otimes |\bar{\theta_j}\rangle)$ for $\bar{\theta_j} \in [\theta_j - \epsilon, \theta_j + \epsilon]$

  This means if you repeat the process $O(logn)$ times and choosing the most frequent estimate for the eigenvalue gives us an additive error of ϵ w.p. $1 - \frac{1}{\text{poly}(n)}$

C.2 Approximation Bound

C.2.1 Bad Recommendation Bound for All Users

We provide a proof of Lemma 4.1 here.

Proof. From triangle inequality, we can say $||T||_F - ||\tilde{T_k}||_F \le ||T - \tilde{T_k}||_F \le \epsilon||T||_F$. Looking at the ends of the inequality, we can say $||\tilde{T_k}||_F \ge (1-\epsilon)||T||_F$. Separately, we can also say

$$
\begin{aligned}
(\epsilon||T||_F)^2 &\ge ||T - \tilde{T_k}||_F^2 && (20)\\
&= \sum_{(i,j)}(T_{i,j} - (\tilde{T_k})_{i,j})^2 && (21)\\
&= \sum_{(i,j):T_{i,j}=1}(1 - (\tilde{T_k})_{i,j})^2 + \sum_{(i,j):T_{i,j}=0}(-(\tilde{T_k})_{i,j})^2 && (22)\\
&\ge \sum_{(i,j):T_{i,j}=0}(-(\tilde{T_k})_{i,j})^2 && (23)\\
&= \sum_{(i,j):\text{bad recommendation}}(\tilde{T_k})_{i,j}^2 && (24)
\end{aligned}
$$

Therefore, we have that

$$
\begin{aligned}
\Pr[\tilde{T_k} \text{ gives bad recommendation}] &= \frac{\sum_{(i,j):\text{bad recommendation}}(\tilde{T_k})_{i,j}^2}{||\tilde{T_k}||_F^2} && (25)\\
&\le \frac{\sum_{(i,j):\text{bad recommendation}}(\tilde{T_k})_{i,j}^2}{((1-\epsilon)||T||_F)^2} && (26)\\
&\le \frac{(\epsilon||T||_F)^2}{((1-\epsilon)||T||_F)^2} = \left(\frac{\epsilon}{1-\epsilon}\right)^2 && (27)
\end{aligned}
$$

□

C.2.2 Bad Recommendation Bound for Typical Users

We provide a proof of Theorem 4.2 here:

Theorem C.1. *Let S be a subset of the rows of T where $|S| \geq (1-\zeta)m$ for $\zeta > 0$. $\forall i \in S$,*

$$\frac{1}{1+\gamma}\frac{||T||_F^2}{m} \leq ||T_i||^2 \leq (1+\gamma)\frac{||T||_F^2}{m} \tag{28}$$

for some $\gamma > 0$ (i.e. the ℓ_2-norm of user i's preference is bounded by some expectation of the entire preference matrix per user. This describes the bound for typical-ness of a user for parameter γ). Again, let $\tilde{T}_k$ be an approximation such that $||T - \tilde{T}_k||_F \leq \epsilon||T||_F$.

Given above, there exist a subset $S' \subseteq S$ where $|S'| \geq (1-\delta-\zeta)m$ for $\delta > 0$ such that on average over the users in S', the probability that a sample from row i of $\tilde{T}_k$ is a bad recommendation is:

$$\Pr[\tilde{T}_k \textit{ gives bad recommendation for typical user } i \in S'] \leq \frac{\left(\frac{\epsilon}{1-\epsilon}\right)^2 \cdot (1+\epsilon)^2}{\left(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}}\right)^2 (1-\delta-\zeta)} \tag{29}$$

Proof. We first claim that at least $(1-\delta)m$ rows (i.e. users) of T and $\tilde{T}_k$ (denoted as T_i and $(\tilde{T}_k)_i$ - ith row) are close enough in the bound:

$$||T_i - (\tilde{T}_k)_i||_2^2 \leq \frac{\epsilon^2||T||_F^2}{\delta \cdot m} \tag{30}$$

We can show this by contradiction. Suppose, at least $\delta \cdot m$ rows (i.e. users) satisfy

$$||T_i - (\tilde{T}_k)_i||_2^2 > \frac{\epsilon^2||T||_F^2}{\delta \cdot m} \tag{31}$$

and these $\delta \cdot m$ users are all contained in the set S''. This means, by summing all the users in the set S'':

$$||T - \tilde{T}_k||_F^2 = \sum_{i=1}^{m} ||T_i - (\tilde{T}_k)_i||_2^2 > \sum_{i \in S''} ||T_i - (\tilde{T}_k)_i||_2^2 > (\delta \cdot m)\frac{\epsilon^2||T||_F^2}{\delta \cdot m} = \epsilon^2||T||_F^2 \tag{32}$$

in which this is a contradiction as we have assumed $(||T - \tilde{T}_k||_F)^2 \leq (\epsilon||T||_F)^2$

Now, suppose for some set $S' \subseteq S$ containing at least $(1-\delta-\zeta)m$ users that *also* is in the bound we proved 30 (as $(1-\delta)m > (1-\delta-\zeta)m$), we construct a new bound:

$$\begin{aligned}
&\forall i \in S' : ||(\tilde{T}_k)_i||_F^2 && (33)\\
&\geq (||T_i||_F - ||T_i - (\tilde{T}_k)_i||_F)^2 && (34)\\
&\geq ([\frac{1}{\sqrt{1+\gamma}}\frac{||T||_F}{\sqrt{m}}] - [\frac{\epsilon||T||_F}{\sqrt{\delta \cdot m}}])^2 = \frac{||T||_F^2}{m}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2 && (35)\\
&\geq \frac{||\tilde{T}_k||_F^2}{(1+\epsilon)^2 m}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2 && (36)
\end{aligned}$$

where going from the second line to the third line and third line to the fourth line ($(1+\epsilon)||T||_F \geq ||\tilde{T}_k||_F$) comes from triangle inequality and rest from derived assumptions. Using our results from 4.1, we have:

$$\left(\frac{\epsilon}{1-\epsilon}\right)^2 \tag{37}$$

$$\geq \Pr[\tilde{T}_k \text{ gives bad recommendation}] \tag{38}$$

$$= \sum_{i=1}^{m} \frac{||(\tilde{T}_k)_i||_F^2}{||\tilde{T}_k||_F^2} \cdot \frac{\sum_{j:(i,j)\text{bad recommendation}} (\tilde{T}_k)_{i,j}^2}{||(\tilde{T}_k)_i||_F^2} \tag{39}$$

$$\geq \sum_{i=S'} \frac{||(\tilde{T}_k)_i||_F^2}{||\tilde{T}_k||_F^2} \cdot \frac{\sum_{j:(i,j)\text{bad recommendation}} (\tilde{T}_k)_{i,j}^2}{||(\tilde{T}_k)_i||_F^2} \tag{40}$$

$$\geq \sum_{i=S'} \frac{[\frac{||\tilde{T}_k||_F^2}{(1+\epsilon)^2 m}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2]}{||\tilde{T}_k||_F^2} \cdot \frac{\sum_{j:(i,j)\text{bad recommendation}} (\tilde{T}_k)_{i,j}^2}{||(\tilde{T}_k)_i||_F^2} \tag{41}$$

$$= |S'| \cdot \frac{[\frac{||\tilde{T}_k||_F^2}{(1+\epsilon)^2 m}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2]}{||\tilde{T}_k||_F^2} \cdot \frac{\sum_{j:(i,j)\text{bad recommendation}} (\tilde{T}_k)_{i,j}^2}{||(\tilde{T}_k)_i||_F^2} \tag{42}$$

$$\geq |S'| \cdot \frac{[\frac{||\tilde{T}_k||_F^2}{(1+\epsilon)^2 m}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2]}{||\tilde{T}_k||_F^2} \cdot (\Pr[\tilde{T}_k \text{ gives bad recommendation for typical user } i \in S']) \tag{43}$$

$$\geq (1-\delta-\zeta)m \cdot \frac{[\frac{||\tilde{T}_k||_F^2}{(1+\epsilon)^2 m}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2]}{||\tilde{T}_k||_F^2} \cdot (\Pr[\tilde{T}_k \text{ gives bad recommendation for typical user } i \in S']) \tag{44}$$

$$\geq (1-\delta-\zeta)m \cdot [\frac{1}{(1+\epsilon)^2 m}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2] \cdot (\Pr[\tilde{T}_k \text{ gives bad recommendation for typical user } i \in S']) \tag{45}$$

Finally, solving for $\Pr[\tilde{T}_k$ gives bad recommendation for typical user $i \in S']$ gives us:

$$\Pr[\tilde{T}_k \text{ gives bad recommendation for typical user } i \in S']) \leq \frac{(\frac{\epsilon}{1-\epsilon})^2}{(1-\delta-\zeta)\frac{1}{(1+\epsilon)^2}(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}})^2} \tag{46}$$

$$= \frac{\left(\frac{\epsilon}{1-\epsilon}\right)^2 \cdot (1+\epsilon)^2}{\left(\frac{1}{\sqrt{1+\gamma}} - \frac{\epsilon}{\sqrt{\delta}}\right)^2 (1-\delta-\zeta)} \tag{47}$$

□

C.3 Matrix Sampling

C.3.1 Classical Bound for Sub-Sampled Matrix

Theorem C.2 (Classical bound for sub-sampled matrix)**.** *Let $A \in \mathbb{R}^{m\times n}$ and $b = \max_{i,j} A_{ij}$ and define $\tilde{A}$ to be a random matrix obtained by sub-sampling w.p. $p = \frac{16nb^2}{\eta||A||_F^2}$ (assume $\eta > 0$). That*

is:

$$\tilde{A}_{i,j} = \begin{cases} \frac{(A)_{i,j}}{p}, & w.p.\ p = \frac{16n}{(\eta||A||_F)^2} \\ 0, & else \end{cases} \tag{48}$$

Then, we have that for any k-th rank approximation of $\tilde{A}$ defined as $\tilde{A}_k$, the following bound holds

$$||A - \tilde{A}_k||_F^2 \leq ||A - A_k||_F + 3\sqrt{\eta}k^{\frac{1}{4}}||A||_F \tag{49}$$

with probability at least $1 - e^{-19(\log n)^4}$.

Proof. Full-proof covered in this paper [1]. Won't be covered here as this is the extensive proof for the survey of classical matrix-sampling. We will be using the results here for our quantum case. But the general idea is knowing each sampling is independent, constructing a bound with its the variance (dependent on the max element of the matrix) and the expectation of the random matrix - nothing too relevant to our discussion of recommendation systems. □

C.3.2 Quantum Bound for Sub-Sampled Matrix

Theorem C.3 (Quantum-adapted bound for $\tilde{T}_{\geq\sigma}$)**.** *We know $\max_{i,j} T_{i,j} = 1$ and define $\tilde{T}$ to be a random matrix obtained by sub-sampling w.p. $p = \frac{16n \cdot 1^2}{\eta||T||_F^2}$ (assume $\eta > 0$). That is:*

$$\tilde{T}_{i,j} = \begin{cases} \frac{(T)_{i,j}}{p}, & w.p.\ p = \frac{16n}{(\eta||T||_F)^2} \\ 0, & else \end{cases} \tag{50}$$

Let $\mu > 0$ be a threshold parameter and denote $\sigma = \sqrt{\frac{\mu}{k}}||\tilde{T}||_F$. Define T_k as the k-th rank approximation of T such that $||T - T_k||_F \leq \epsilon||T||_F$. Then, we have for $\tilde{T}_{\geq\sigma}$ that:

$$||T - \tilde{T}_{\geq\sigma}||_F \leq ||T - T_k||_F + (3\sqrt{\eta}k^{\frac{1}{4}}\mu^{-\frac{1}{4}} + \sqrt{\frac{2\mu}{p}})||T||_F \tag{51}$$

with probability at least $1 - e^{-19(\log n)^4}$.

Proof. Utilizing the theorem in C.2 and encapulating $\tilde{T}_{\geq\sigma}$ in terms of the l-th rank approximation $\tilde{T}_l$ by saying $\sigma_l \geq \sigma = \sqrt{\frac{\mu}{k}}||\tilde{T}||_F$, we can bound:

$$||T - \tilde{T}_{\geq\sigma}||_F = ||T - \tilde{T}_l||_F \leq ||T - T_l||_F + 3\sqrt{\eta}l^{\frac{1}{4}}||T||_F \tag{52}$$

Now to relate T_k, we need to consider two cases ($l \geq k$ and $k > l$).

- If $l \geq k$, we can see that $||T - T_l||_F \leq ||T - T_k||_F$ by the definition of low-rank approximation.
- If $k > l$, consider the triangle inequality: $||T - T_l||_F \leq ||T - T_k||_F + ||T_k - T_l||_F$.

We see that taking the tightest bound for both cases, we want to evaluate $||T - T_l||_F \leq ||T - T_k||_F + ||T_k - T_l||_F$. Specifically, we want to find out the upper-bound for the error $||T_k - T_l||_F$. We have for $k > l$:

$$||T_k - T_l||_F^2 = \sum_{i=l+1}^{k} \sigma_i^2(T) \leq k\frac{\mu}{k}||\tilde{T}||_F^2 \tag{53}$$

Knowing that the sampling of each element $\tilde{T}_{i,j}$ is independent, we obtain the Chernoff bounds $\Pr[||\tilde{T}_F^2|| > \frac{2||T||_F^2}{p}] \leq e^{-\frac{||T||_F^2}{3p}}$. Note the probability that $||\tilde{T}||_F^2 > \frac{2||T||_F^2}{p}$ is exponentially small - thus we'll generalize that with high confidence, $||\tilde{T}||_F^2 \leq \frac{2||T||_F^2}{p}$ holds. This allows one to say:

$$||T_k - T_l||_F^2 = \sum_{i=l+1}^{k} \sigma_i^2(T) \leq k\frac{\mu}{k}||\tilde{T}||_F^2 \leq \frac{2\mu}{p}||T||_F^2 \tag{54}$$

Therefore, combining the derived bound we obtain that

$$||T - \tilde{T}_{\geq\sigma}||_F \leq ||T - T_k||_F + (3\sqrt{\eta}k^{\frac{1}{4}}\mu^{-\frac{1}{4}} + \sqrt{\frac{\mu}{p}})||T||_F \tag{55}$$

with probability at least $1 - e^{-19(\log n)^4}$ where $e^{-19(\log n)^4}$ is the derived Chernoff bound for our choice of $p = \frac{16n}{(\eta||T||_F)^2}$, $\mu = \frac{\epsilon^2 p}{2}$, and $\eta \leq \frac{2n^{1/4}\epsilon^{3/2}}{3(2k)^{1/4}||T||_F^{1/2}}$.

□

We provide a proof of Theorem 4.3 here, theorem copied below for convenience:

Theorem C.4 (Quantum-adapted bound for $\tilde{T}_{\geq\sigma,\kappa}$)**.** *We know* $\max_{i,j} T_{ij} = 1$ *and define* $\tilde{T}$ *to be a random matrix obtained by sub-sampling w.p.* $p = \frac{16n}{\eta||T||_F^2} \geq \frac{36\sqrt{2}(nk)^{\frac{1}{2}}}{||T||_F\epsilon^3}$ *(n is the number of products, assume* $\eta > 0$*, and* $||T - T_k||_F \leq \epsilon||T||_F$ *where* T_k *is the k-th rank approximation of T). That is:*

$$\tilde{T}_{i,j} = \begin{cases} \frac{(T)_{i,j}}{p}, & w.p.\ p = \frac{16n}{(\eta||T||_F)^2} \\ 0, & else \end{cases} \tag{56}$$

Let $\mu > 0$ *be a threshold parameter and denote* $\sigma = \sqrt{\frac{\mu}{k}}||\tilde{T}||_F$*. Let* $\kappa > 0$ *be a precision parameter.*

Given above, for quantum-calculated k-rank approximation, $\tilde{T}_{\geq\sigma,\kappa}$*, we have that*

$$||T - \tilde{T}_{\geq\sigma,\kappa}||_F^2 \leq 3||T - T_k||_F + (3\sqrt{\eta}k^{\frac{1}{4}}\mu^{-\frac{1}{4}}[2 + (1-\kappa)^{-\frac{1}{2}}] + (3-\kappa)\sqrt{\frac{2\mu}{p}})||T||_F \tag{57}$$

with probability at least $1 - e^{-19(\log n)^4}$*.*

For concreteness, suppose $\kappa = \frac{1}{3}, \mu = \frac{\epsilon^2 p}{2}$*, and* $\eta \in \left(0, \frac{2n^{1/4}\epsilon^{3/2}}{3(2k)^{1/4}||T||_F^{1/2}}\right]$*. Then, we have:*

$$||T - \tilde{T}_{\geq\sigma,\kappa}||_F \leq \cdots \leq 9\epsilon||T||_F \tag{58}$$

(i.e. a linear bound similar to the one in Lemma 4.1 (say $\epsilon' \coloneqq 9\epsilon$*), suggesting* $\tilde{T}_{\geq\sigma,\kappa}$ *is a good approximation given correct parameterization).*

Proof. We first have to derive the bound for $\tilde{T}_{\geq\sigma}$. Then extend the bound for $\tilde{T}_{\geq\sigma,\kappa}$. The theorem for $\tilde{T}_{\geq\sigma}$ is provided C.3.

From our bound of $||T - \tilde{T}_{\geq\sigma}||$, we now relate this to $||T - \tilde{T}_{\geq\sigma,\kappa}||$ by noting:

$$\begin{aligned} &||T - \tilde{T}_{\geq\sigma,\kappa}||_F && (59)\\ &\leq ||T - \tilde{T}_{\geq\sigma}||_F + ||\tilde{T}_{\geq\sigma} - \tilde{T}_{\geq\sigma,\kappa}||_F && (60)\\ &\leq ||T - \tilde{T}_{\geq\sigma}||_F + ||\tilde{T}_{\geq\sigma} - \tilde{T}_{\geq(1-\kappa)\sigma}||_F && (61)\\ &\leq ||T - \tilde{T}_{\geq\sigma}||_F + ||T - \tilde{T}_{\geq\sigma}||_F + ||T - \tilde{T}_{\geq(1-\kappa)\sigma}|| && (62)\\ &= 2||T - \tilde{T}_{\geq\sigma}||_F + ||T - \tilde{T}_{\geq(1-\kappa)\sigma}||_F && (63) \end{aligned}$$

through definitions mentioned above. For the first term in the RHS inequality ($||T - \tilde{T}_{\geq\sigma}||_F$), we can directly substitute C.3. As for $||T - \tilde{T}_{\geq(1-\kappa)\sigma}||_F$, we can also substitute C.3 with setting $\mu' = (1-\kappa)^2\mu$:

$$||T - \tilde{T}_{\geq(1-\kappa)\sigma}||_F \leq ||T - T_k||_F + (3\sqrt{\eta}k^{1/4}\mu^{-1/4} + (1-\kappa)\sqrt{\frac{2\mu'}{p}}) \tag{64}$$

Substituting and bounding the entire term give us:

$$||T - \tilde{T}_{\geq\sigma,\kappa}||_F^2 \leq 3||T - T_k||_F + (3\sqrt{\eta}k^{\frac{1}{4}}\mu^{-\frac{1}{4}}[2 + (1-\kappa)^{-\frac{1}{2}}] + (3-\kappa)\sqrt{\frac{2\mu}{p}})||T||_F \tag{65}$$

□