On the Implementation of a Quantum-Inspired Classical Algorithm to Build Recommender Systems

By: Ary Swaminathan, Savvy Raghuvanshi, Raahul Acharya

December 2018

1 Introduction

A recommendation system suggests products to users based on data preferences. User-user and product-product recommendation systems are usually modeled by completing the entries of an $m \times n$ matrix M where M_{ij} represents user i's preferences towards product j. A significant desirable property of such a matrix is having a small rank k – the ability to be approximated as a low-rank matrix – which is essential when performing matrix factorization of M into smaller matrices of a minimum dimension k. This operation is used extensively in recommendation system algorithms.

In this project, we offer (to our knowledge) the first implementation of the first classical algorithm to produce a recommendation in $O(poly(k)polylog(m,n))$ time, which is an exponential improvement over previous classical algorithms that have runtime linear in m and n. This project is based on Ewin Tang's paper "A Quantum-Inspired Classical Algorithm for Recommendation Systems" (2018) which constructed a classical algorithm inspired by the Quantum Machine Learning algorithm of Kerenidis and Prakash (2016). In addition to providing a concrete implementation of this algorithm in C++, we discuss Tang's result that Kerenidis and Prakash's algorithm, which previously displayed one of the most significant examples of quantum speedup, does not in fact provide an exponential speedup over classical algorithms.

An essential characteristic of these algorithms is to seek only a randomized sample from the user's preferences instead of reconstructing all of them, as the

latter would, of course, require large amounts of data computation. Tang's algorithm samples the high-value entries from a low-rank approximation of an input matrix, which is nothing more than an approximated factorization of this input matrix into smaller, more tractable matrices in time independent of m and n, the number of users and number of items, respectively.

To complete the entries of a large user preference matrix, we work based on the standard assumptions that users tend to fall in a small number of classes based upon their preferences, and that using these classes we can build future recommendations for products that the users have not interacted with. This assumption corresponds to the desired property of a low rank in the input matrix. Current algorithms tend to reconstruct full rows of user's preferences, taking $\Omega(n)$ time. Tang's new algorithm employs probabilistic methods to sample a given user's data – favoring high-weight entries – and uses these to build future recommendations.

2 Quantum Inspiration

2.1 Context

Tang's algorithm is classical, and thus can be explained without any reference to quantum computers. Still, to convey some of the quantum inspiration behind Tang's algorithm before discussing it in further detail, we will give a brief introduction to quantum information. In his book *Quantum Computing Since Democritus*, Professor Scott Aaronson, advisor to Tang during the discovery of this quantum-inspired algorithm, states that "[q]uantum mechanics is a beautiful generalization of the laws of probability: a generalization based on the 2-norm rather than the 1-norm, and on complex numbers rather than nonnegative real numbers." Abstracting away from the details of particle dynamics and physical devices, some basic assumptions allow us to reason about the complexity of quantum information, particularly in the context of recommendation algorithms and the underlying data structures of interest to this project.

2.2 Preliminaries

In quantum computation, the analog to a classical bit is a quantum bit, or "qubit". Unlike classical bits, which have a discrete value of 0 or 1, single qubits are complex-weighted superpositions of 0 and 1. It is convenient to represent

these superpositions as normalized unit vectors in Hilbert space, with 0 and 1 as a basis. By extension, a classical n-bit state can be represented as an element of $\{0,1\}^n$. Thus, there are a total of 2^n possible n-bit states. Considering this set of classical states as a basis for a Hilbert space, we generalize to the quantum case by representing an n-qubit state as a unit vector in $\mathbb{C}^{2^n}$.

Bra-ket notation is commonly used to represent quantum states. The underlying vector of an n-qubit state ψ and its conjugate transpose can be written as the ket $|\psi\rangle_n$ and the bra $\langle\psi|_n$, respectively. Keeping in mind the formula for the inner product of complex vectors, this notation gives a natural representation of the inner product of states ϕ and ψ as $\langle\phi|\psi\rangle = \langle\psi|\phi\rangle$. Notice that $\langle\psi|\psi\rangle = 1$ for any normalized state ψ. In keeping with convention, we often suppress the number of qubits when writing the bra or ket of an arbitrary quantum state.

For clarity, consider the example of a 2-qubit state $|\psi\rangle_2$, which is a weighted sum of $2^2 = 4$ basis states. Taking the ith term of the (zero-indexed) vector representation of $|\psi\rangle_2$ to be the complex amplitude $\alpha_i \in \mathbb{C}$, we can write $|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$, such that the binary representation of indices corresponds to the classical basis states. Note that requiring $|\psi\rangle$ to be normalized, representing a unit vector in $\mathbb{C}^4$, implies that

$$\langle\psi|\psi\rangle = \sum_{i=0}^{3} \alpha_i \overline{\alpha}_i = 1$$

, where $\overline{\alpha}$ is the complex conjugate of α.

The essential connection between quantum information states and classical probabilities (and a key point of interest to this project) arises from the concept of measurement. Although we can imagine a quantum state as a superposition of classical states, the act of measurement collapses a quantum state, yielding a single classical state. We can think of measurement as probabilistic projection onto the subspace generated by a measured basis state.

Letting b_i denote the binary representation of index i, the Born measurement rule states that for an arbitrary n-qubit state

$$|\psi\rangle_n = \sum_{i=0}^{2^n-1} \alpha_i |b_i\rangle$$

the probability of measuring the state $|b_i\rangle$ is given by $\alpha_i\overline{\alpha}_i$. As alluded to in our previous example of a 2-qubit state, it is easily verified that normalizing an arbitrary n-qubit state ensures that the probabilities of measuring each of the 2^n classical basis states sum to 1.

2.3 Relevance of the No-cloning Theorem

Kerenedis and Prakash exploit the structure of quantum measurement to encode the operation of l_2 sampling from a vector. For a normalized vector v of dimension 2^n, we can prepare an n-qubit state that has amplitudes $\alpha_i = v_i$, such that measurement yields i with probability v_i^2. In turn, the classical data structure proposed by Tang for the purpose of l_2 sampling relies on the difficulty of quantum state preparation to achieve a similar complexity bound.

An aspect of classical information that we take for granted is that it can be copied and distributed. Given that measurement collapses a quantum state, one might naturally wonder whether it is possible to learn the superposition of a state by copying it many times and then measuring repeatedly as a means of sampling the underlying distribution. This turns out to impossible, however, as a result of the the No-cloning theorem. For intuition, without developing the additional concepts required for a complete proof, we present a non-rigorous derivation toward this result. A disclaimer aimed toward readers with a background in physics: we gloss over the formalization of unitary operators and entirely ignore the notion of phase factors in the following discussion, as it suffices to consider quantum states in terms of their projective components.

Based on our assumption that valid quantum states represent vectors of norm 1, we can further assume that valid operations on quantum states are norm-preserving and respect linearity. Assume there exists an cloning operator

$$f : \mathbb{C}^{2^{2n}} \to \mathbb{C}^{2^{2n}}$$

such that, for any arbitrary quantum state $|\psi\rangle_n$, we have

$$f(|\psi\rangle_n|0\rangle_n) = |\psi\rangle_n|x\rangle_n$$

Then by the definition of f, for any quantum state $|\alpha x\rangle$, we have

$$f(|\alpha x\rangle|0\rangle) = |\alpha x\rangle|\alpha x\rangle = \alpha^2|x\rangle|x\rangle$$

But f must also preserve linearity, and so we have

$$f(|\alpha x\rangle|0\rangle) = f(\alpha|x\rangle|0\rangle) = \alpha f(|x\rangle|0\rangle) = \alpha|x\rangle|x\rangle$$

This implies that

$$\alpha^2|x\rangle|x\rangle = \alpha|x\rangle|x\rangle$$

The above is clearly false for any $\alpha \neq 1$. Although we have not proven the No-cloning theorem, we emphasize the main takeaway, that there cannot exist a function that copies arbitrary quantum states.

Due to the No-cloning theorem, a data structure encoding the necessary matrix information in quantum states cannot be copied, and so the Quantum Machine Learning algorithm relies on the computationally expensive process of preparing identical states for the purpose of repeated measurement. This is an important insight that Tang exploits, bringing us back to the world of classical information. While we focus the remainder of our discussion on Tang's classical algorithm, it is worth remembering the idea of quantum states as complex amplitudes on a computational basis, and the role of quantum measurement in drawing the resulting connection to classical probability distributions. In particular, while both quantum and classical computations result in classical outputs, the notion of complex amplitudes allows for a different set of manipulations to be performed on qubits in the intermediate steps of a calculation, offering a different (and sometimes faster) path to the desired solution.

3 Algorithm

3.1 Sketch

Our algorithm is based off the following two theorems produced in Tang's paper.

Theorem 1: Suppose we are given as an input a matrix A supporting query and ℓ^2-norm sampling operations, a row/user $i \in [m]$, a singular threshold σ, and a sufficiently small $\epsilon > 0$. There is a classical algorithm whose output distribution is O(ϵ)-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, where the run time of this algorithm is independent of m and n, as shown below:

$O(poly(\frac{||A||_F}{\sigma}, \frac{1}{\epsilon}, \frac{||A_i||}{||D_i||}, log(\frac{1}{\delta})))$

where δ is the probability of failure, $||A||_F$ is the Frobenius norm of A, $||A_i||$ is the ℓ^2 norm of row i in A, and similarly for $||D_i||$.

This theorem is essentially stating that, given a user i, we can output a list of preferences whose values are ϵ-bounded in size to the given user i preferences in A, and therefore this implies that the values of this fully-completed list of approximated recommendations are sufficiently close to what the user would actually rate the products he has not seen before.

Theorem 2: Applying theorem 1 to the recommendation systems model with a quantum state preparation data structure achieves the same bounds of quality recommendations as the quantum-based algorithm (that this classical

algorithm was based off) up to constant factors and for a sufficiently small ϵ.

This theorem essentially explains how the classical algorithm we will be implementing can achieve the same bounds of accuracy as the quantum-version of this algorithm proposed, and combined with Theorem One, can do so in the same order of time. Note that it requires the usage of the same quantum state preparation structure, but this can be averted by implementing the lightweight Binary Search Tree (BST) we will discuss.

The algorithm works as follows:

1. Begins with an input matrix A stored in the BST data structure that suuports powerful ℓ^2-norm sampling operations in constant time.

2. Apply a subsampling strategy, called ModFKV, to find a low-rank approximation of A. This algorithm does not output the completed recommendation matrix in full, and rather outputs a succinct description of the matrix, which is a set of approximately orthonormal singular vectors $\hat{V}$, where the low-rank approximation D of A then becomes $A\hat{V}\hat{V}^T$, which is the projection of the rows of A onto the low-dimensional subspace spanned by $\hat{V}$. Interestingly, we are able to sample from and query these singular vectors quickly.

3. Now given the desired low-rank approximation D, we look to sample from a particular row of it- representing extracting high-value entries from the a given user i's preferences. This is equivalent to sampling from $A_i\hat{V}\hat{V}^T$. To perform this sampling, we first estimate $A\hat{V}$, estimating dot products that can be done with the sampling access to A allowed through our data structure. We then use this output to perform the same operations and give an approximate sample from $A\hat{V}\hat{V}^T$. This sample is the desired output.

In the following sections we will discuss in more details the significant areas of this algorithm.

3.2 Data Structure

Since we are interested in achieving sublinear bounds for this classical algorithm, we must expedite the process by concerning ourselves with how the input is given.

If the input matrix $A \in \mathbb{R}^{mxn}$ is given in a classical unordered list of entries (i,j,A_{ij}), clearly linear time is required to parse through the input to alter it into a usable form. Even if this input is relatively structured (for example, given the entries of A sorted by row and column) then we still cannot sample the low-rank approximation of a generic matrix in sublinear time because of the time needed to locate non-zero entries.

Therefore, for our algorithm, we implemented the following data structure with low overhead cost.

- We built a binary search tree for a given vector $v \in \mathbb{R}^n$, which is equivalent to the row of user i's preferences towards n products.
- The n leaf nodes store, instead of v_i for $i \in [n]$, v_i^2 along with $sgn(v_i)$
- Building from this, each interior node stores the sum of the values of its two children. Ex: The parent node of v_0^2 and v_1^2 holds a value equivalent to $v_0^2 + v_1^2$. This allows for updates to be done quickly by updating all of the nodes above a particular leaf.
- Following this process, root node of a particular vector v simply holds $||v||^2$
- A significant characteristic of this data structure is its sampling efficiency. Sampling a high-value entry from D_v, which is the distribution on vector v, can simply be done by starting from the top node of the tree $||v||^2$ and randomly recursing on a child with the probability proportional to the child's weight. Note: We define $D_v(i) = \frac{v_i^2}{||v||^2}$
- This data structure allows for:
  1. Reading an updating an entry of the vector in O($log^2(n)$) time, where n is the number of components of the vector. This, as described above, is equivalent to updating all parent nodes above a particular leaf.
  2. Finding $||v||^2$ in O(1) time- simply pull the root node of the tree.
  3. Sampling from D_v in O($log^2(n)$) time again, by randomly recursing through all levels of a tree until you reach a leaf node.
- Now, when given a full m x n matrix A, we can extend this data structure to allow for the same operations in the same time constraints by building

n BST's with the norms of each row squared as their roots, and then extending this BST by taking binary sums until the root node of this new BST is simply $||A||_F^2$.

3.3 ModFKV

ModFKV is the essentially the low-rank approximation algorithm that produces the matrix D from which we will sample finally, making it central to this writeup. Recall that ModFKV produces the D from its description by projecting input matrix A onto a subspace defined by vectors $\hat{V} \in \mathbb{R}^{nxk}$ where k is the small rank we mentioned earlier in this paper. That is,

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

where $\hat{V}^{(t)}$ represents the t'th column vector in $\hat{V}$.

This description implies the columns of $\hat{V}$ to be a linear combination of rows of A. Let S be the submatrix of A by restricting the rows to $i_0...i_p$ for some p and then renormalizing row r by $\frac{1}{\sqrt{pD_{\tilde{A}(r)}}}$ where $\tilde{A}$ is equivalent to A restricted to the first p rows.

Then $\hat{V}^{(i)} := \frac{S^T u^{(i)}}{\sigma^{(i)}}$ where $u^{(i)}$ is the i'th left singular column vector of U given the singular value decomposition(SVD) $A = U\Sigma V$ (which can be produced trivially by already known algorithms.

In summary, what ModFKV does is subsamples a matrix, computes the large singular vectors of the matrix, and then outputs these, under the assumption that these singular vectors give a good description of the matrix.

ModFKV works, in full, as follows (Note: this is taken fully from Ewin Tang's Paper):

1. **Input:** Matrix $A \in \mathbb{R}^{mxn}$ supporting quick norm sampling operations outlined in Data Structures section, with a threshold σ (which described lower-bound value for entries of the sampled matrix), and error parameters ϵ and κ.

2. **Output** Description of D, low-rank approximation of A.

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

4. Set $\hat{\epsilon} = \frac{\kappa\epsilon^2}{\sqrt{K}}$

5. Set $p = 10^7 max\{\frac{K^4}{\hat{\epsilon}^3}, \frac{K^2}{\hat{\epsilon}^4}\}$

6. Sample rows $i_1...i_p$ from $D_{\tilde{A}}$

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

8. Sample columns $j_1...j_p$ from F

9. Let the resulting p x p submatrix, with row r normalized by $\frac{1}{\sqrt{pD_{\tilde{A}(r)}}}$ and column c be normalized by $\frac{1}{\sqrt{pD_{\tilde{A}(c)}}}$, be denoted W.

10. Compute the left singular vectors of W $u^{(1)}...u^{(k)}$ that correspond to singular values $\sigma^{(1)}...\sigma^{(k)}$ larger than σ (again produced by SVD of A)

11. Output $i_1...i_p$, $u^{(1)}...u^{(k)}$, and $\sigma^{(1)}...\sigma^{(k)}$ as a description of output matrix D.

4 Testing and Drawbacks

A large drawback about this algorithm's applicability are its run-time constants. While for a significantly large input matrix A (say more than 10 million users and/or products), it can outperform most current singular value decomposition algorithms, for any test sets that must run in < 1 hour, this algorithm is not so practical.

Earlier in this writeup. We mentioned that the run time of this classical algorithm that produced a low-ranked sampling was:

O($poly(\frac{||A||_F}{\sigma}, \frac{1}{\epsilon}, \frac{||A_i||}{||D_i||}, log(\frac{1}{\delta}))$)

The poly() function applied here omits one major subtlety- the actual degree of this polynomial. The true run time, presented below, is more informative of the drawbacks of this algorithm.

O($max\{\frac{||A||^{30}}{\sigma^{30}\epsilon^{18}\kappa^6}, \frac{||A||^{24}}{\sigma^{24}\epsilon^{18}\kappa^8}\}$)

where κ is sampling error.

As can be seen, with a max degree of 30, this algorithm poses a large barrier of computational time for numerous entries. Therefore, we were not able to display the outputs of our implementation on the Netflix Dataset as we intended, and, in fact, were not able to implement it on any meaningful dataset of user preferences. We did, however, attempt to test this algorithm's accuracy on small, self-constructed matrices and even here were obstructed with a run-time of a length on the order of the Earth's history.

5 Conclusion

We were initially inspired to take on this project after reading a blog post by Tang's advisor, Professor Scott Aaronson. Here is a short excerpt from Professor Aaronson's blog, *Shtetl-Optimized*:

"This is an abstraction of the problem that's famously faced by Amazon and Netflix, every time they tell you which books or movies you "might enjoy." What's striking about Ewin's algorithm is that it uses only polylogarithmic time: that is, time polynomial in log(m), log(n), the matrix rank, and the inverses of the relevant error parameters. Admittedly, the polynomial involves exponents of 33 and 24: so, not exactly "practical"! But it seems likely to me that the algorithm will run much, much faster in practice than it can be guaranteed to run in theory. Indeed, if any readers would like to implement the thing and test it out, please let us know in the comments section! [...] "At the same time, Ewin's result yields a new algorithm that can be run on today's computers, that could conceivably be useful to those who need to recommend products to customers, and that was only discovered by exploiting intuition that came from quantum computing."

Based on our initial implementation of the algorithm as formulated in Tang's paper, we offer a tentative response. Although we cannot state this with certainty, based on back-of-the-envelope calculations it is our suspicion that a fully optimized implementation of this algorithm making use of multiple state-of-the-art GPUs would nonetheless be unable to run within the order of days even on very small matrices. This is because of its reliance on rejection sampling, which causes samples to be accepted with extremely low probability (this is the step which results in the above referenced exponent of 33). Thus we are led to believe that, while this algorithm can theoretically run on today's computers, it cannot yet be of use to those who need to recommend products to customers. We are grateful to Professor Aaronson for introducing us to the significance of this result, and of course to Ewin Tang for sharing his impressive discovery. Though the immediate usefulness of this algorithm is unclear (certainly our implementation offers no practical benefit) we are excited by the possibility that quantum inspiration could yield further advances in classical computing.

6 References

1. *Aaronson, Scott. Quantum Computing since Democritus. Cambridge University Press, 2015.*

2. *Aaronson, Scott. "Shtetl-Optimized." ShtetlOptimized RSS, Wordpress, www.scottaaronson.com/blog/.*

3. *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.*

4. *Tang, Ewin. "A Quantum-Inspired Classical Algorithm for Recommendation Systems." 13 July 2018, pp. 1–36.*