Quantum and Quantum-inspired Recommendation System Algorithms

Siqi Shen
Department of Electrical Engineering
University of Michigan
Ann Arbor, MI 48105
shensq@umich.edu

Do June Min
Department of Computer Science
University of Michigan
Ann Arbor, MI 48105
dojmin@umich.edu

***Abstract*—This report is a case study of how development of quantum algorithms can not only lead to quantum methods that achieve speedup over classical algorithms but also provide insights for improving classical solutions. Specifically, we focus on the case of collaborative recommendation systems by matrix sampling and describe the process of how the potential quantum speedup was identified and implemented. Then, we look at an improved classical algorithm that builds on the quantum algorithm and achieves equivalent computational complexity and introduce a few guidelines a quantum machine learning researchers may employ.**

***Index Terms*—quantum computation, quantum complexity, recommendation systems, low-rank approximation**

I. Introduction

In this report, we examine three quantum computation-related papers and provide a synopsis of how development of quantum algorithms to a given problem can not only lead to quantum methods that achieve speedup over classical algorithms but also provide insights for improving classical solutions. Specifically, we focus on the case of collaborative recommendation systems by matrix reconstruction and illustrate the process of how possible quantum speedup was identified and implemented in the context of recommendation system, which in turn led to a classical algorithm that achieves equivalent computational complexity. To this end, we read and summarize the following three papers:

1) On the Power of Quantum Computation by Daniel R. Simon [2]. Using this paper, we introduce the model of quantum computation and discuss the promise and limitation of quantum computation.
2) Quantum Recommendation Systems by Iordanis Kerenidis and Anupam Prakash [11] implements a quantum recommendation algorithm which achieves exponential speedup over conventional classical methods.
3) A Quantum-inspired Classical Algorithm for Recommendation Systems by Ewin Tang [12] takes hint from [11] and proposes a classical algorithm with equivalent speedup. Moreover, the paper generalizes from the observation that the dequantization of the quantum recommendation system led to a classical recommendation algorithm with equivalent complexity and provides a useful guideline for quantum machine learning researchers.

The report is organized as follows: In Section II, we briefly introduce relevant literature in quantum computation and recommendation systems other than the ones covered in this report. Through Section III, IV, and V, we summarize each paper while highlighting how the quantum algorithm informed an improved classical algorithm.

II. Related Works

Formal models of quantum computation has been pioneered by Deutsch in his influential work [1]. Fortnow and Aaronson also made important contributions about the relationship of BQP and other complexity classes [3] [9]. The approach to recommendation system covered in this project, collaborative filtering, follows the definition and assumptions made in the seminal work of Drineas et al [4]. Finally, the classical recommendation algorithm by Tang builds upon the low-rank approximation algorithm written by Frieze et al [6].

III. Quantum Computation and Quantum Speedup

Often popular media's conception of quantum supremacy amounts to the idea that quantum computation will bring no-questions-asked-type exponential speedup over classical modes of computation [10]. However, this is not the case and effective quantum computation requires programmers to "choreograph" the steps of computation so that the amplitudes of the "undesirable" paths cancel out and only the desirable configurations would remain or at least have high amplitude so that after measurement one can obtain the desired classical output with high probability. In this section, we review Simon's paper [2] and briefly describe the promise and limitation of quantum computation, which in turn informs us about what type of care and attention should be paid to claim quantum speedup over classical computation.

A formal model of quantum computation, quantum Turing machine (QTM), can be succinctly described informally as an analogue of probailistic Turing machine (PTM), with modifications that conform to the laws of quantum mechanics. Whereas each edge of the computation path of a PTM corresponded to a probability and thus the probability of a path is calculated as the product of the probabilities of the edges along the path, each edge in the computation path of a QTM is is instead associated with an *amplitude*. Thus the amplitude

of a path is similarly computed as the sum of the constituent edges' amplitudes. More importantly, now the probability of a configuration at each step is computed as the *square* of the amplitude of the leaf nodes in the configuration. This endows QTM with the ability to "cancel out" some configurations, such as in $(-\alpha+\alpha)^2 = 0$. An important caveat is that at each computation step the computation tree must obey the property that the sum of all configuration's probabilities should sum to 1. It can be shown, but omitted here, that transition functions (or matrices) that satisfy this are *unitary*.

With QTMs, we can define and work with a complexity class analogous to bounded-error probabilistic polynomial time (BPP) by replacing the PTM in the formal definition of BPP with a QTM. This new complexity class is called bounded-error quantum polynomial time (BQP). Just as BPP can be used as a proxy for the class of problems efficiently solvable by classical (including probabilistic) computation, BQP is representative of problems efficiently solvable by quantum computation. Naturally, complexity theorists and quantum scientists are interested in the relationship between BQP and other classes. For instance, what are some problems not in P or BPP but are in BQP? What about in NP (or NP-hard) and BQP?

Unfortunately, there are only few theoretical results on the standing of BQP in relation to other complexity classes, and the following inclusion relationship only provides a partial picture:

$$\text{P} \subseteq \text{BPP} \subseteq \text{BQP} \subseteq \text{AWPP} \subseteq \text{PP} \subseteq \text{PP}$$

It does not tell us whether BPP $\subsetneq$ BQP or even P $\subsetneq$ BQP. Still, there are several pieces of evidence that quantum computation does provide. Shor's integer factorization algorithm and Grover's database search are some examples. However, the bottom line is that any claim of quantum supremacy or speedup should be carefully examined, especially given that it is hard to show that there can be no classical algorithm that can match a quantum algorithm's performance.

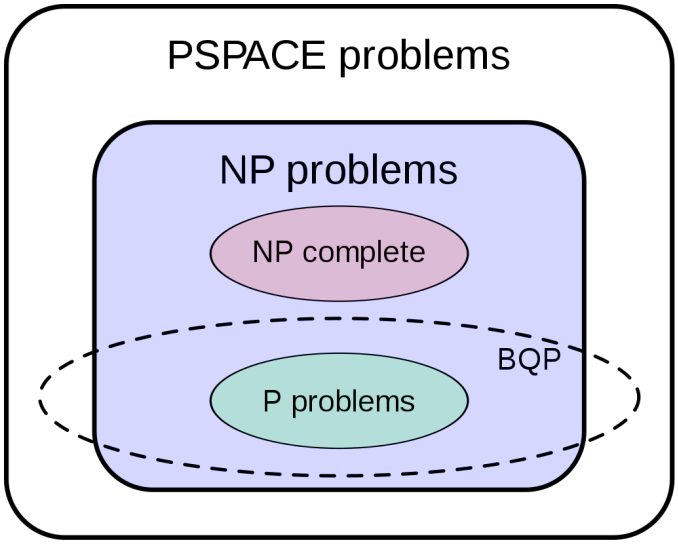

Figure 1. BQP and Other Complexity Classes

IV. Quantum Recommendation System

The algorithm proposed by Harrow, Hassidim, Lloyd[8] was a breakthrough in quantum algorithm development and provides a powerful tool for performing linear algebra and machine learning operations using quantum computers. However, the HHL algorithm and HHL-based algorithms cannot output the classical solution of the problem, and they assume the input is both sparse and well-conditioned in order to get a polylogarithmic performance.

On the other hand, it is natural to ask how well can quantum algorithms perform compared to classical algorithms, under the same problem setting and assumptions. In the work of Kerenidis et al. [11], the recommendation system is used as the problem setting to answer the previous question.

A. Recommendation System

A recommendation system uses known preference information of m users on n products and provides a personalized recommendation for each individual users. For example, the preference information can be the number of stars a user give to a product in the review on Amazon, or whether the user chose to skip a song on Spotify.

The preference information is usually model described by an $m \times n$ matrix P, which is called the preference matrix. There are two key characteristics of matrix P worth noting. First, for a recommendation system used in an industrial setting, the number of users and products can be more than a million, which makes it prohibitive to store the whole preference matrix. Second, matrix P is not based on prior information, since an individual user won't specify his preference on all the products. Thus, typically there will be a lot of unknown entries in the preference matrix. Besides, the preference matrix can always be cast into a binary form by using some threshold or ranking.

Most classical recommendation algorithms uses a two-stage approach. The first stage reconstructs the low-rank approximation of the preference matrix, and restore the intermediate result such as singular values and singular vectors. And this stage is only performed when there is a need to update the reconstruction to get a more accurate result with the new data. The second stage is an online computation step using the intermediate result each time a user sends a query. In general, the first stage takes $O(poly(mn))$, and the second stage takes $O(nk)$, where k is rank of low-rank approximation.

B. Recommendations by Matrix Sampling

The classical algorithms which outputs the whole user vector of the low-rank approximation are $O(n)$. Note that to produce a good recommendation, it is not always necessary to know the exact preference value for every product. Instead, all we need is a sample from the products with a high preference value. It can be shown that sampling the low-rank approximation of subsample matrix $\hat{T}_k$ is equivalent to producing a good recommendation with high probability. Here

	P_1	P_2	P_3	P_4	$\cdots$	$\cdots$	P_{n-1}	P_n
U_1	0	0	?	?	$\cdots$	$\cdots$	?	1
U_2	0	?	0	?	$\cdots$	$\cdots$	1	?
U_3	?	?	1	1	$\cdots$	$\cdots$	?	0
$\vdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
U_m	?	1	?	?	$\cdots$	$\cdots$	?	0

Figure 2. Illustration of preference matrix

the subsample matrix refers to the matrix with all known entries and 0 for the unknown entries.

When we consider how similar two matrices are, it is always helpful to check their distance according to some distance measure. The intuition is that if a matrix is close enough to the underlying preference matrix, sampling from that matrix will behave similarly as sampling from the preference matrix. This is formally stated as follows:

Theorem 1. *Let $\tilde{T}$ be an approximation of the matrix T such that $\left\|T-\tilde{T}\right\|_F \leq \epsilon\|T\|_F$. Then, the probability a sample according to $\tilde{T}$ is a bad recommendation is*

$$Pr[(i,j)bad] \leq (\epsilon/(1-\epsilon))^2 \tag{1}$$

This theorem guarantees that the probability of sampling a bad recommendation will decrease exponentially as we required that the approximation is close to the preference matrix. This result can also be generalized to typical users, whose known entries is close to the average.

The subsampling of preference matrix is assumed to be according to uniform distribution, which means each entry has a equal probability p to be sampled. And the result is normalized to $\hat{A}_{ij} = A_{ij}/p$. Following that, the subsample matrix has desirable properties.

Theorem 2. *Let $A \in \mathbb{R}^{m*n}$ be a matrix and $b = \max_{ij} A_{ij}$. Define the matrix $\hat{A}$ to be a random matrix obtained by subsampling with probability $p = 16nb^2/(\eta\|A\|_F)^2$ and rescaling, that is $\hat{A}_{ij} = A_{ij}/p$ with probability p and 0 otherwise. With probability at least $1 - \exp\{-19(\log n)^4\}$ we have for any k*

$$\left\|A-\hat{A}_k\right\|_F \leq \|A-A_k\|_F + 3\sqrt{\eta}k^{1/4}\|A\|_F \tag{2}$$

Since the term $\|A-A_k\|_F$ is small according to the low-rank assumption, the distance from $\hat{A}_k$ to A is upper bounded by a small value with high probability.Thus, the problem is now reduced to how to sampling from $\hat{A}_k$.

C. Singular Value Estimation

To project a vector onto $\hat{T}_k$, it is clear that we need to eliminate the components corresponding singular vector with singular value out of the top-k. Since this operation directly depends on the singular value of each component, it is desirable that we can separate every component in the singular vector bases, and have access to the singular value. This is achieved by the singular value estimation algorithm. The task is to find an algorithm within time complexity polylog(mn). Thus any one step should not exceed this limit, including preparing the quantum states and quantum operations.

1) Quantum State preparation: For the state preparation stage, we need a data structure that can be used by a quantum algorithm to do the mapping $\tilde{U} : |i\rangle|0\rangle \to |i\rangle|A_i\rangle$, for $i \in [m]$, corresponding to a row of the matrix, and $\tilde{V} : |0\rangle|j\rangle \to |\tilde{A}\rangle|j\rangle$, for $j \in [n]$, corresponding to the norm for each row. It turns out that binary search trees can satisfy these requirements. Each row is represented by a binary search tree, with each leaf to be the square of each entry, and the value of the root node is the sum of two child nodes. With this data structure, there is an established algorithm by Grover et al [5] that can create the corresponding state by conditioned unitary transformation.

2) Phase Estimation: Phase estimation can be used to extract eigenvalue information of a unitary transformation. There will be an additive error to the estimated phase and the time complexity is inversely proportional to the precision parameter ϵ.

Theorem 3 (Phase Estimation)**.** *Let U be a unitary operator, with eigenvectors $|v_j\rangle$ and eigenvalues $e^{i\theta_j}$ for $\theta_j \in [-\pi,\pi]$. For a precision parameter $\epsilon > 0$, there exists a quantum algorithm that runs in time $O(T(U))\log n/\epsilon$ and with probability $1 - 1/poly(n)$ maps a state $|\phi\rangle = \sum_{j\in[n]} \alpha_j|v_j\rangle$ to the state $|\phi\rangle = \sum_{j\in[n]} \alpha_j|v_j\rangle|\bar{\theta}_j\rangle$ such that $\bar{\theta}_j \in \theta_j \pm \epsilon$ for all $j \in [n]$*

It turns out that by constructing a unitary matrix $W = U \cdot V$, where $U = 2PP^t - I_{mn}$ and $V = 2QQ^t - I_{mn}$, the isometry $Q : \mathbb{R}^n \to \mathbb{R}^{mn}$ maps a row singular vector of A with singular value σ_i to an eigenvector Qv_i with eigenvalue $e^{i\theta_i}$ such that $\cos(\theta_i/2) = \sigma_i/\|A\|_F$. With this relationship, once the eigenvalues are known the singular values are also known. The whole singular value estimation process is shown in Algorithm 1.

Algorithm 1: Singular Value Estimation

Data: $A \in \mathbb{R}^{m*n}$, $x \in \mathbb{R}^n$, precision parameter $\epsilon > 0$.
Result: A quantum state with the second register being an estimation of singular values

1. Create $|x\rangle = \sum_i \alpha_i|v_i\rangle$.
2. Append a first register $|0^{\lceil \log m \rceil}\rangle$ and create the state $|Qx\rangle = \sum_i \alpha_i|Qv_i\rangle$
3. Perform phase estimation with precision parameter $2\epsilon > 0$ on the input $|Qx\rangle$for the unitary $W = U \cdot V$ and obtain $\sum_i \alpha_i|Qv_i, \bar{\theta}_i\rangle$
4. Compute $\bar{\sigma}_i = \cos(\bar{\theta}_i/2)\|A\|_F$ where $\bar{\theta}_i$ is the estimate from phase estimation, and uncompute the output of the phase estimation.
5. Apply the inverse of the transformation in step 2 to obtain $\sum_i \alpha_i|v_i\rangle|\bar{\sigma}_i\rangle$.

D. Quantum Projection

After we extract the information of the singular value, what remains then straight forward. Note that the information of singular values and singular vectors are all contained in the quantum state. Since we want to distinguish components with singular values higher than the threshold and the other components, a second register is appended to the original state, which is used to contain the information whether this component in the superposition belongs to the part we are interested in.

The value of the second register can be calculated through a unitary transformation on the former state. This idea is general and can be compared to the classical way. It is equivalent to doing an "if" query and appending the Boolean result to the end of the data. The only difference is that in the classical setting the condition is checked one by one, while in the quantum setting the condition is checked simultaneously, and the workload is moved to building the corresponding unitary transformation in physical model.

With the last register acting as an identifier, we will be able to tell which subspace the collapsed state comes from, by measuring the state and check the classical result of the last register. It is worth noting that the algorithm does not guarantee a valid output in a single iteration. Thus the time complexity directly depends on the probability to get valid output. If that probability is inversely exponential to the problem size, then the expected time complexity will always be exponential. Luckily, in our case, the expected time complexity can be proved to be $O(\text{polylog}(mn))$

Algorithm 2: Quantum Projection with Threshold

Data: $A \in \mathbb{R}^{m*n}, x \in \mathbb{R}^n$, parameters $\theta, \kappa > 0$.
Result: A sample from the row rank approximation $\hat{A}_k$.

1 Create $|x\rangle = \sum_i \alpha_i |v_i\rangle$
2 Apply the singular value estimation on $|x\rangle$ with precision $\epsilon = \frac{\kappa}{2}\frac{\sigma}{\|A\|_F}$ to obtain the state $\sum_i \alpha_i |v_i\rangle|\bar{\sigma}_i\rangle$
3 Apply on a second new register the unitary V that maps $|t\rangle|0\rangle \to |t\rangle|1\rangle$ if $t < \sigma - (\kappa/2)\sigma$ and $|t\rangle|0\rangle \to |t\rangle|0\rangle$ otherwise, to get the state $\sum_{i\in S} \alpha_i |v_i\rangle|\bar{\sigma}_i\rangle|0\rangle + \sum_{i\in\bar{S}} \alpha_i |v_i\rangle|\bar{\sigma}_i\rangle|1\rangle$
4 Apply the singular value estimation on the above state to erase the second register $\sum_{i\in S} \alpha_i |v_i\rangle|0\rangle + \sum_{i\in\bar{S}} \alpha_i |v_i\rangle|1\rangle$
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.

E. Analysis

For the algorithm state above, it is necessary to check its performance in correctness and expected running time. Just as any probabilistic algorithm, the result is not always correct. If the probability of correct is very small, the result will be of little value if any. And as mentioned before, each iteration of algorithm does not guarantee a valid output, so the expected time becomes what we care about. And it can be showed that

$$Pr_{i\ U_{S'},j\ (\hat{T}_{\geq\sigma,\kappa})_i}[(i,j)bad] \leq \frac{(\frac{9\epsilon(1+9\epsilon)}{(1-9\epsilon)})^2}{((\frac{1}{\sqrt{1+\gamma}} - \frac{9\epsilon}{\sqrt{\delta}})^2(1-\sigma-\zeta))}$$

For the expected running time, with at least $(1-\xi)(1-\theta-\zeta)m$ users in the typical set, the expected running time is $O(\text{polylog}(mn)\text{poly}(k))$, where ξ, θ, ζ have small values.

V. Classical Recommendation System

In the previous section, we have seen that a quantum matrix sampling method supported by a relatively simple data structure resulted in an exponential speedup over previous classical algorithms. Thus, it is tempting to claim that akin to Shor's integer factorization algorithm and Grover's search algorithm, Kerenidis et al's quatun recommendation system serves as an indirect evidence of the advantange of quantum computation over classical computation. However, this is not the case since there is a classical algorithm that achieves equivalent speedup.

A. $\mathcal{L}^2$-norm sampling

The classical algorithms adopts essentially the same approach as the quantum algorithm to the recommendation problem, using matrix sampling instead of matrix reconstruction to avoid polynomial complexity while computing a low-rank approximation of the preference matrix. However, instead of relying on quantum measurement to get a recommendation sample, the classical algorithm employs the BST-like data structure to satisfy $\mathcal{L}^2$-norm sampling assumption required by Frieze et al's algorithm for finding low-rank approximations. Specifically, the $\mathcal{L}^2$-norm sampling assumption requires that given a nonzero vector $x \in \mathcal{R}^n$, we can sample an entry x_i with probability

$$P(X = x_i) = \frac{x_i^2}{||x||^2}$$

With this sampling operation provided, we can use Frieze et al's low-rank approximation algorithm to get a description of an approximation of the preference matrix, which in turn is used to sample a good recommendation.

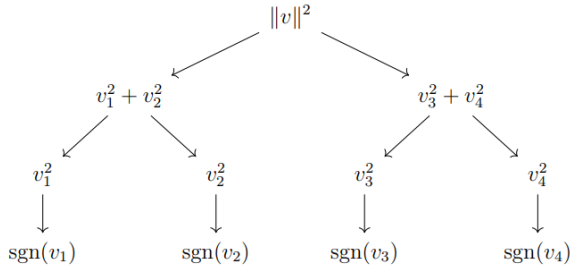

Figure 3. An Example BST

Note that with the classical BST structure assumed, we can perform $\mathcal{L}^2$-norm sampling in the following way:

1) Start from the root node.
2) Let the norms assigned to the left and right child node be a and b respectively. Choose the left child with probability $\frac{a}{a+b}$. Otherwise, choose the right child.
3) Go to the chosen child node.
4) If the child node is a leaf node, terminate. Otherwise, go to Step 2.

B. ModFKV and Recommendation Sampling Algorithm

Here, for completeness, we provide the core algorithms used in the improved classical recommendation system, ModFKV algorithm and the low-rank sampling algorithm.

Algorithm 3: ModFKV

Data: A Data Structure Encoding Matrix $A \in \mathbb{R}^{mn}$ that supports the $\mathcal{L}^2$-norm operations, threshold σ, error parameters ϵ, η

Result: A description of an output matrix D

Set $K = ||A||_F^2/\sigma^2$ and $\bar{\epsilon} = \eta\epsilon^2$;

Set $q = \Theta(\frac{K^4}{\bar{\epsilon}^2})$;

Sample rows $i_1, \cdots, i_q$ from $D_{\tilde{A}}$;

Let $\mathcal{F}$ denote the distribution given by choosing an $s \sim_u [q]$, and choosing a column from $D_{A_{i_s}}$;

Sample columns $j_1, \cdots, j_q$ from $\mathcal{F}$;

Let W be the resulting qq row-and-column-normalized submatrix $W_{rc} := \frac{A_{i_r j_c}}{q\sqrt{D_{\tilde{A}(i_r)}\mathcal{F}(j_c)}}$;

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

Output $i_1, ..., i_q$, $U \in \mathbb{R}^{q\times k}$ the matrix whose ith column is $u^{(i)}$, and $\hat{\Sigma} \in \mathbb{R}^{k\times k}$ the diagonal matrix whose ith entry on the diagonal is $\sigma^{(i)}$. This is the description of the output matrix D

ModFKV algorithm adapts and slightly modifies from Frieze et al's fast algorithm to find a rank k approximation of a matrix [6]. The only difference between the original FKV algorithm and ModFKV algorithm is that the latter includes additioinal error parameters to get a low-rank approximation bound that matches that of the quantum algorithm. Using ModFKV as a subroutine, now we can reconstruct a low-rank reconstruction of the preference matrix and sample a recommendation s.

Finally, we note that this algorithm runs in $O(\text{poly}(k)\log(mn))$. Without delving into a complete analysis of the complexity, this runtime can be broken down into the $O(\text{poly}(\frac{||A||_F}{\sigma}, \frac{1}{\epsilon}, \frac{1}{\eta}\frac{||A_i||}{||D_i||}))$, which is independent of the dimensions m, n of the preference matrix, and $O(\log mn)$ required for sampling operations. While it is polynomially slower than the quantum algorithm, Tang's classical algorithm still achieves an exponential speedup over previous classical methods. For a more detailed discussion and analysis of the algorithm, we refer the reader to [6] and [12].

Algorithm 4: Low-rank approximation sampling

Data: A Data Structure Encoding Matrix $A \in \mathbb{R}^{mn}$ that supports the $\mathcal{L}^2$-norm operations, user $i \in [m]$, threshold σ, $\epsilon > 0$, $\eta \in (0, 1]$

Result: Sample $s \in [n]$

Run ModFKV with parameters (σ, ϵ, η) to get a description of $D = A\hat{V}\hat{V}^T = AS^T\hat{U}\hat{\Sigma}^2\hat{U}^TS$;

Estimate A_iS^T entrywise with parameter $\frac{\sqrt{\epsilon}}{K}$ to estimate $\langle A_i, S_t^T\rangle$ for all $t \in [q]$. Let est be the resulting $1 \times q$ vector of estimates;

Compute est$\hat{U}\hat{\Sigma}^2\hat{U}T$ with matrix-vector multiplication;

Sample s from (est$\hat{U}\hat{\Sigma}^2\hat{U}T$)S ;

Output s

C. Guideline for Quantum Machine Learning Researchers

Motivated by the observation that the dequantization of a quantum algorithm led to an equivalent speedup, we can derive the following rules of thumbs for quantum machine learning algorithm developers:

- Match and compare quantum machine learning algorithms with state preparation assumptions/requirement to classical algorithms with that uses sampling operations.
- If a quantum machine learning algorithm uses state preparation assumptions, then it should be able to surpass the capabilities of classical algorithms with $\mathcal{L}^2$- sampling. Not only is quantum data preparation assumption often non-trivial to satisfy, but also in many cases we already know how to perform fast classical sampling on existing computers.
- When comparing quantum machine learning algorithms to classical machine learning algorithms in the context of finding speedups, quantum state preparation assumptions should be matched with $\mathcal{L}^2$-norm sampling assumptions in the classical ML model.

VI. Conclusion

According to Aaronson [10], writers of quantum machine learning algorithms must "read the fine print", meaning that special attention should be given to the whole range of different assumptions and conditions that need to be met before the algorithm can be considered implementable. This is not only to ensure that the proposed algorithm is realizable, but also to see if the necessary assumptions allow for an equivalent or better classical algorithm. Our report illustrates that by reading the "fine print", one can not only gain an improved understanding of quantum algorithms but also be gifted with valuable insights that may lead to improved classical algorithms.

References

[1] David Deutsch. "Quantum theory, the Church-Turing principle and the universal quantum computer". In: 400 (1985), pp. 97–117.

[2] Daniel R. Simon. “On the Power of Quantum Computation”. In: *SIAM J. Comput.* 26.5 (Oct. 1997), pp. 1474–1483. ISSN: 0097-5397. DOI: 10.1137/S0097539796298637. URL: http://dx.doi.org/10.1137/S0097539796298637.

[3] Lance Fortnow and John D. Rogers. “Complexity limitations on quantum computation”. In: *CoRR* cs.CC/9811023 (1998). URL: http://arxiv.org/abs/cs.CC/9811023.

[4] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. “Competitive Recommendation Systems”. In: *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*. STOC ’02. Montreal, Quebec, Canada: ACM, 2002, pp. 82–90. ISBN: 1-58113-495-9. DOI: 10.1145/509907.509922. URL: http://doi.acm.org/10.1145/509907.509922.

[5] Lov Grover and Terry Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions”. In: *arXiv preprint quant-ph/0208112* (2002).

[6] Alan Frieze, Ravi Kannan, and Santosh Vempala. “Fast Monte-carlo Algorithms for Finding Low-rank Approximations”. In: *J. ACM* 51.6 (Nov. 2004), pp. 1025–1041. ISSN: 0004-5411. DOI: 10.1145/1039488.1039494. URL: http://doi.acm.org/10.1145/1039488.1039494.

[7] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum Algorithm for Linear Systems of Equations”. In: *Phys. Rev. Lett.* 103 (15 Oct. 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[8] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. In: *Physical review letters* 103.15 (2009), p. 150502.

[9] Scott Aaronson. “BQP and the Polynomial Hierarchy”. In: *Proceedings of the Forty-second ACM Symposium on Theory of Computing*. STOC ’10. Cambridge, Massachusetts, USA: ACM, 2010, pp. 141–150. ISBN: 978-1-4503-0050-6. DOI: 10.1145/1806689.1806711. URL: http://doi.acm.org/10.1145/1806689.1806711.

[10] Scott Aaronson. “Quantum Machine Learning Algorithms : Read the Fine Print”. In: 2015.

[11] Iordanis Kerenidis and Anupam Prakash. “Quantum Recommendation Systems”. In: (Mar. 2016).

[12] Ewin Tang. “A quantum-inspired classical algorithm for recommendation systems”. In: *CoRR* abs/1807.04271 (2018). arXiv: 1807.04271. URL: http://arxiv.org/abs/1807.04271.