Quantum Algorithms 3/23/2023

Lecture 16: Input/Output Access Mechanisms & Block Encodings

Instructor: Dieter van Melkebeek

Some of the quantum algorithms that exhibit quantum advantage or supremacy need the input to be represented in certain ways, and produce the output in certain ways. In this lecture, we look at the input/output mechanisms, with a case study about the recommendation problem. We discuss the QRAM access to vectors and matrices, and the sparse access to matrices, and see how they facilitate the creation of efficient block encodings for matrices.

1 Recommendation problem

To begin, we discuss a famous motivating problem called the *recommendation problem*, which is also known as the *Netflix prize problem*[1]. The setup is as follows:

1. There are M customers indexed by $i \in [M]$.
2. There are N products indexed by $j \in [N]$.
3. There exists an unseen true preference matrix $T \in \mathbb{R}^{M \times N}$, whose (i, j)-th entry corresponds to the preference of customer i for product j. The preference is often binary or constrained to $[0, 1]$.
4. We observe A, a collection of samples from T. An example A may look like

	P_1	P_2	P_3	P_4	$\cdots$	P_{N-1}	P_N
C_1	0.1	0.4				0.85	
C_2			0.6			0.85	
C_3			0.8	0.9			
$\vdots$					$\ddots$		
C_M		0.75					0.2

   where the empty entries correspond to unknown entries of T that are not observed in A.

Given this data, the input is a row index $i \in [M]$ corresponding to a customer, and the expected output is a recommendation $j \in [N]$ which is believed to be preferred for customer i.

The problem has further two associated assumptions:

- *Low rank hypothesis*: T is (close to) a matrix of small rank.

  The low rank hypothesis means that there are a small number of underlying dimensions that matter, and the customer preferences are some linear combination of those few dimensions. The number of dimensions is the rank of T. Two users with similar preferences have similar combinations of those dimensions. One can feasibly predict a missing entry based on seen entries. Without this assumption, the missing entries could be anything. This hypothesis turns out to be fairly reasonable.

[1] From 2006-2009, Netflix held a contest with a $1,000,000 dollar grand prize to attempt to find an effective solution to this problem for their movie recommendation system.

- *Sampling hypothesis*: A is obtained by picking each entry from T independently with some probability p.

  The sampling hypothesis suggests that the sampling of one entry from T has no impact on the sampling of another entry. This assumption is more debatable because one could imagine a customer who gave a rating for one product may be more willing to rate other closely related products. Here for simplicity, we assume there is no such correlation.

Given these two assumptions, the best guess for T is a low rank approximation of A/p , where p is the probability of a given entry being observed and unknown entries of A are replaced with zeros. Let $k \ll \min\{N, M\}$ be the assumed low rank of T.

The best rank k approximation to A in 2-norm can be computed from the singular value decomposition (SVD) by setting all but the top k singular values equal to zero. This can be seen as follows. Recall that the SVD writes A as $A = U\Sigma V^*$, where $U \in \mathbb{C}^{M\times M}$ and $V \in \mathbb{C}^{N\times N}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{M\times N} = \mathrm{diag}(\sigma_1, \sigma_2, \dots)$ is a diagonal matrix with $\sigma_1 \geq \sigma_2 \cdots \geq 0$. Since multiplication with a unitary matrix on the left or the right does not affect the 2-norm nor the rank of a matrix, it suffices to argue the property for the diagonal matrix Σ.

Claim. In the case of Σ, the property states that the best rank k approximation to Σ in 2-norm is $\Sigma_k \doteq \mathrm{diag}(\sigma_1, \dots, \sigma_k)$.

Proof. It suffices to consider the setting with $N = M$. Let $B \in \mathbb{C}^{N\times N}$ have rank at most k. The null space of B, i.e., the set of $x \in \mathbb{C}^N$ such that $Bx = 0$, has dimension $N - k$. The set of $x \in \mathbb{C}^N$ where the last $N - k - 1$ components are zero has dimension $k + 1$. It follows that both subspaces have a nontrivial intersection. In other words, there exists a nonzero $x^* \in \mathbb{C}^N$ such that $Bx^* = 0$ and the only nonzero components of x^* lie in the first $k + 1$ dimensions. It follows that

$$\|(\Sigma - B)x^*\|_2 = \|\Sigma x\|_2 \geq \sigma_{k+1}\|x\|_2,$$

and therefore that $\|\Sigma - B\|_2 \geq \sigma_{k+1}$. On the other hand, Σ_k is of rank k and satisfies $\|\Sigma - \Sigma_k\|_2 = \sigma_{k+1}$. It follows that $B = \Sigma_k$ is a best rank k approximation to Σ. □

Note that the claim also holds for the Frobenius norm instead of the 2-norm. It again suffices to consider the case of diagonal matrices. In the case of the Frobenius norm the case of diagonal matrices is straightforward.

Using braket notation, we can write the SVD of A as

$$A = U\Sigma V^* = \sum_{j=1}^{\min(M,N)} \sigma_j \left|u_j\right\rangle \left\langle v_j\right|,$$

where $U = [|u_1\rangle\, |u_2\rangle \dots |u_M\rangle]$ and $V = [|v_1\rangle\, |v_2\rangle \dots |v_N\rangle]$. For a given customer with index i, we can write the i-th row of A as:

$$A_{i,\cdot} = \langle e_i| A = \sum_{j=1}^{\min(M,N)} \sigma_j \langle e_i|u_j\rangle \langle v_j|.$$

We obtain the i-th row of the best approximation of A of rank k by cancelling the singular values σ_j for $j \geq k+1$ in the above expression. Thus, our approximation to the i-th row of T is:

$$T_{i,\cdot} \sim P_{i,\cdot} \doteq \frac{1}{p} \sum_{j=1}^{k} \sigma_j \langle e_i | u_j \rangle \langle v_j |.$$

Up to a scale factor, it is the projection of the i-th row of A onto the top k row singular value of A.

Quantum algorithm for recommendation systems. Given a block encoding of A, we can efficiently compute a block encoding for a close approximation to a scaled version of A_k, where A_k denotes the matrix A with all but the top k singular values set to zero. This can be done using similar techniques as the ones we developed for obtaining a block encoding of A^{-1} from a block encoding of A. One we have such a block encoding, we can use it to compute a close approximation to the normalization of $A_k^* |e_i\rangle$, i.e., $|P_{i,\cdot}\rangle$. The details of how this is done are not the focus of this lecture (see [KP17] if you are interested), but the result is that we can compute $|P_{i,\cdot}\rangle$ up to an error of ϵ in 2-norm in time $O(\text{poly}\log(MN/\varepsilon))$.

Recall that $|P_{i,\cdot}\rangle$ represents $P_{i,\cdot}$ in a very compact format and does not provide easy access to individual components of $P_{i,\cdot}$. However, we are only interested in finding an index j for which $P_{i,j}$ is large. This is something that we can easily obtain given the compact representation as the superposition $|P_{i,\cdot}\rangle$: Measuring $|P_{i,\cdot}\rangle$ yields j with probability proportional to $|P_{i,j}|^2$. It follows that, with high probability and under the model assumptions, measuring $|P_{i,\cdot}\rangle$ yields a product (movie) j that is well-liked by customer i. Thus, the recommendation problem is a setting where the compact representation in the form of a superposition works great.

Notice that the classical input size of a matrix of size $M \times N$ is already MN. To just read it in would take $\Theta(MN)$, which is already exponentially larger than the runtime of $O(\text{poly}\log(MN/\varepsilon))$ mentioned above. Therefore in order to perform such quick quantum computations, we need some compressed or compact way of representing the input, which we discuss next.

2 Input/output access issues

Input/output access issues arise naturally when a quantum algorithm has inputs and/or outputs that are not classical. The quantum inputs need to be constructed from classical input, and classical output needs to be extracted from the quantum output. Such transformations are inherently needed to achieve superpolynomial speedups for problems that are classically solvable in time polynomial in the size of the classical input. In particular, this is the case for the linear algebra problems we considered: solving systems of linear equations and the recommendation problem above.

Input considerations. Typical quantum structures that need to be created are vectors (superpositions), matrices, and black-boxes. The first two arise in the linear algebra problems mentioned. The last one occur in black-box algorithms like Grover search.

Many quantum algorithms require multiple trials run on the same quantum input. Due to the no cloning theorem, we cannot just copy the input and rerun an algorithm on the copy. In order to efficiently do these trials, we would need some extra constraints.

- If the input is obtained from some unitary generating process, then we can run the process from scratch as often as we need. This is one reason why we often assume the existence of a unitary that, when applied to $|0^n\rangle$, yields the input superposition.
- Sometimes we cannot regenerate the input, but we can reconstruct it partially from the output of one run, and the partial reconstruction may suffice to run the process again. This what happened in oblivious amplitude amplification in the purified setting with independent initial weight.
- Sometimes the quantum algorithm, even though it involves measurements, is non-destructive, so we can run it multiple times on the same input without issue. This happens in eigenvalue estimation when an eigenvector is given and in some error-correction procedures.

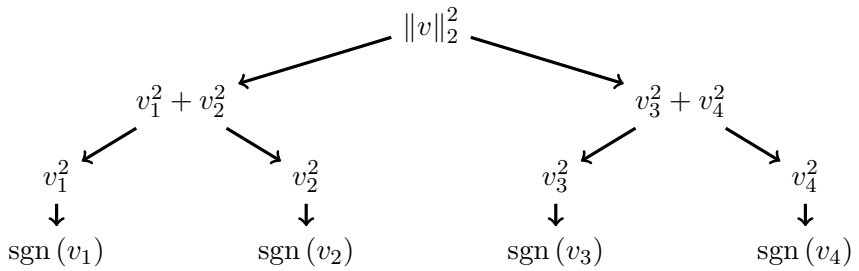
Another input consideration is that we may need to reflect about the input state. For example, Grover search and uses two reflections: one about the bad states, and one about the initial state, which is the uniform superposition. The latter reflection can be done using the Hadamard transform (and its inverse) and a reflection about $|0^n\rangle$. In amplitude amplification, the initial state may be more complicated, so reflecting over it is more of a challenge, and requires access to a unitary that produces the initial state. In general, given a unitary generating a state, a reflection R can be realized as $R = UR_{|0^n\rangle}U^*$, where U denotes a unitary that maps $|0^n\rangle$ to the axis of relection and $R_{|0^n\rangle}$ denotes the reflection about the state $|0^n\rangle$.

Output considerations. Extracting classical output from quantum output requires attention in cases where the quantum output is more succinct than the classical one. This happens in the linear algebra problems we discussed. In the generic setting of solving systems of linear equations, the quantum output $|x\rangle$ of the solution to the system $Ax = b$ may not be very helpful. However, in the setting of the recommendation problem, the quantum superposition is good enough as we can simply measure it to extract what we want.

3 Vector encoding using QRAM access

Quantum Random Access Memory (QRAM) refers to a classical read/write data structure to which we have quantum read access in the sense that we can query the data structure on a superposition.

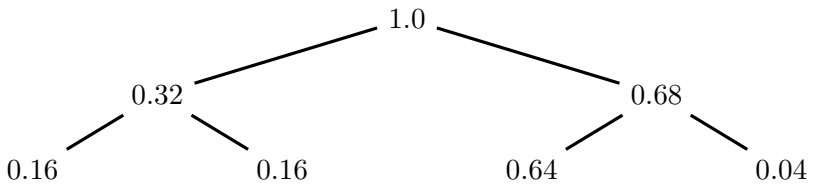
To encode a vector $v \in \mathbb{R}^N$ we make use of a classical binary tree, where each node is labeled by the squared 2-norm of the subtree below it. The leaves are doubled up with sign information since that is lost in the squaring of the 2-norm. More generally, if $v \in \mathbb{C}^N$, the leaves contain the phase instead of the sign. The depth of the tree is given by $n \doteq \log N$. An example of the structure for a vector $v = (v_1, v_2, v_3, v_4) \in \mathbb{R}^4$ is given below.



A nice property of this structure is that if we receive nonzero components one at a time in an online fashion, then for each component we can update our tree by just traversing up from the corresponding leaf to the root in time $O(\log N)$. If we assume our vector v is s-sparse (it only has s nonzero entries), then the entire tree representation of our vector can be filled classically in time $O(s \log N)$.

QRAM state generation. Provided that we can access the tree in superposition (QRAM), we can also generate $|v\rangle$ in time $O(\log N)$. The idea here is that each qubit can be computed using a rotation conditioned on the previous qubit. We demonstrate it through the following example.

Suppose $|v\rangle = 0.4\,|00\rangle + 0.8\,|01\rangle + 0.8\,|10\rangle + 0.2\,|11\rangle$. Then the corresponding tree is



In order to prepare the state $|v\rangle$, we first perform a unitary mapping corresponding to the superposition of the first row's values.

$$|0\rangle\,|0\rangle \mapsto (\sqrt{0.32}\,|0\rangle + \sqrt{0.68}\,|1\rangle)\,|0\rangle\,.$$

That is, we perform a rotation to get a superposition of the state of the first qubit. Next, we perform a rotation on the second qubit conditioning on the first qubit:

$$\begin{aligned}
(\sqrt{0.32}\,|0\rangle + \sqrt{0.68}\,|1\rangle)|0\rangle \mapsto\ & \sqrt{0.32}\,|0\rangle\,\frac{1}{\sqrt{0.32}}\left(\sqrt{0.16}\,|0\rangle + \sqrt{0.16}\,|1\rangle\right) \\
& + \sqrt{0.68}\,|1\rangle\,\frac{1}{\sqrt{0.68}}\left(\sqrt{0.64}\,|0\rangle + \sqrt{0.04}\,|1\rangle\right) \\
=\ & \sqrt{0.32}\,|0\rangle\,\frac{1}{\sqrt{0.32}}\,(0.4\,|0\rangle + 0.4\,|1\rangle) + \sqrt{0.68}\,|1\rangle\,\frac{1}{\sqrt{0.68}}\,(0.8\,|0\rangle + 0.2\,|1\rangle) \\
=\ & (0.4\,|0\rangle\,|0\rangle + 0.4\,|0\rangle\,|1\rangle) + (0.8\,|1\rangle\,|0\rangle + 0.2\,|1\rangle\,|1\rangle) \\
=\ & 0.4\,|00\rangle + 0.4\,|01\rangle + 0.8\,|10\rangle + 0.2\,|11\rangle = |v\rangle
\end{aligned}$$

4 Matrix encoding

There are multiple ways to encode a matrix. Two common ones are QRAM access and sparse access.

Matrix QRAM access. This consists of QRAM access to the rows and columns, and to a vector with the row norms and one with the column norms. More precisely, QRAM access to an matrix M with rows $M_{i,\cdot}$ indexed by i and columns $M_{\cdot,j}$ indexed by j is defined by having each of the following:

- vector QRAM access to each row: $M_{i,\cdot}$,

- vector QRAM access to each column: $M_{\cdot,j}$,
- vector QRAM access to row norms: $r_i \doteq \|M_{i,\cdot}\|_2$
- vector QRAM access to column norms: $c_j \doteq \|M_{\cdot,j}\|_2$

Sparse matrix access. An alternative efficient accessing method that is particularly suitable for matrices with few nonzero entries is sparse access. Sparse access gives us three oracle functions that recover a given matrix entry or indices corresponding to nonzero entries: One that allows us to determine the value of an entry given its row and column index, and two that allow us to locate the nonzero elements in a given row or column, respectively.

First, we have

$$O_M : |i\rangle |j\rangle |0^b\rangle \mapsto |i\rangle |j\rangle |M_{ij}\rangle ,$$

which takes the input of a row index i, a column index j, and a register consisting of b qubits, and XORs in the last register the value of the corresponding matrix entry up to b bits of accuracy.

Second, we have

$$O_{row} : |i\rangle |k\rangle \mapsto |i\rangle |c_{ik}\rangle ,$$

which replaces the value k of the second register by $c_{i,k}$, the column index of the k-th nonzero entry in queried row i. Note that the mapping from k to $c_{i,k}$ is injective for any fixed i, and therefore invertible and a valid quantum operation.

Finally, we have

$$O_{column} : |\ell\rangle |j\rangle \mapsto |r_{\ell j}\rangle |j\rangle ,$$

which replaces the value ℓ of the first register by $r_{\ell j}$, the row index of the ℓ-th nonzero entry in queried column j.

5 Dequantization

Having classical random access to the trees underlying the QRAM access to a matrix A enables us to do certain operations classically in polylogaritmic time, even though this is impossible with merely the standard component-wise access to A. In particular, the QRAM access data structure enables classical sampling in time $O(\log N)$ by flipping a weighted coin at each level. Since the depth of the tree is $O(\log N)$, it only takes that many steps to get a sample according to some distribution. This is interesting because it allows sublinear-time classical algorithms to be developed based on sampling. This has led to the dequantization of several of the machine learning and linear algebra problems that require QRAM access.

The recommendation problem can be solved this way by subsampling the matrix A, and computing the best rank k approximation for the subsample; with high probability this gives us a good approximation to the best rank k approximation for A (after rescaling). In some more detail, the dequantized algorithm works as follows:

1. Get good approximations to the top k singular row vectors of A expressed as linear combinations of rows of A by running the singular value decomposition (SVD) on a $q \times q$ subsample of A for small q:
   (a) $I \leftarrow q$ samples from r.

(b) $J \leftarrow q$ samples from $A_{i,\cdot}$ where $i \in_u I$ for each sample.

(c) $S \leftarrow$ row-and-column normalized $A_{I \times J}$.

(d) Find the top k row singular vectors of S using SVD. Express them as linear combinations of the rows of S.

(e) Take the same linear combinations of the corresponding rows of A as approximations for the top k row singular vectors of A.

2. Find the projection of $A_{i,\cdot}$ onto the approximate row singular vectors.

All of the required operations can be performed in time $n^{O(1)} = (\log N)^{O(1)}$ given classical random access to entries of A and sample access to r and $A_{i,\cdot}$ for each $i \in [N]$. The resulting running time is polynomially bounded in the running time of the quantum algorithm. We refer to [Tan19] for more details.

6 Block encodings

As we have seen in prior lecture, a block encoding is often convenient to represent a matrix in quantum computing. In this section we show how to efficiently obtain a block encoding for a matrix M given its sparse or QRAM encoding.

Let us first recall the definition and the properties of block encodings we have seen so far. A block encoding of a matrix M acting on m qubits with ℓ ancilla qubits is a unitary A acting on $\ell + m$ qubits such that

$$A = \begin{bmatrix} M & * \\ * & * \end{bmatrix}$$

When we talk about computing an efficient block encoding for a matrix M, this usually means we can efficiently construct a small unitary circuit that realizes A.

Here is a summary of the properties we have seen previously.

- Given $|v\rangle$ we can compute $|Mv\rangle$. This is done by applying A to $|0^\ell\rangle |v\rangle$ and postselecting $|0^\ell\rangle$ on the first register. We can use oblivious amplitude amplification in order to boost the probability of measuring the desired $|0^\ell\rangle$.

- Given block encodings for M_j for $j \in [k]$, we have an efficient encoding of $\sum_j c_j M_j / (\sum_j |c_j|)$. This is the linear combination of unitaries (LCU) method and uses an additional $\log k$ ancillas.

- Given encodings for M_1 and M_2, we have an efficient encoding for $M_1 M_2$ which uses $\ell_1 + \ell_2$ ancillas, where ℓ_i is the number of ancillas for the block encoding of M_i.

- Given an encoding of M, we have efficient encodings of the Chebyshev polynomials $T_d(M)$, which we constructed using quantum walks. This uses a constant number of additional ancillas.

The second and fourth property can be used to approximate arbitrary polynomials of M using $\log d$ ancillas where d is the degree of the polynomial.

We now show how to efficiently create a block encoding from either sparse access or QRAM access. Both constructions follow a common pattern. The strategy is to write M as a Gram matrix. Informally, a Gram matrix is made of inner products of states. More specifically:

Definition 1 (Gram matrix). *Given collections states $|\psi_i\rangle$ and $|\phi_j\rangle$, their* Gram matrix G *is given by* $G_{ij} \doteq \langle\psi_i|\phi_j\rangle$

Our strategy for producing an encoding of M is to specify how to generate the $|\psi_i\rangle$ and $|\phi_j\rangle$. In particular we will define maps $U_L : |0^p\rangle|i\rangle \mapsto |\psi_i\rangle$ and $U_R : |0^p\rangle|j\rangle \mapsto |\phi_j\rangle$. Then the first few columns of U_L will be the $|\psi_i\rangle$ and the first few columns of U_R will be the $|\phi_j\rangle$ and the rest will be garbage so

$$U \doteq U_L^* U_R \begin{bmatrix} \langle\psi_1| \\ \langle\psi_2| \\ \langle\psi_3| \\ \vdots \end{bmatrix} \begin{bmatrix} |\phi_1\rangle & |\phi_2\rangle & |\phi_3\rangle & \cdots \end{bmatrix} = \begin{bmatrix} G & * \\ * & * \end{bmatrix}$$

is a block encoding of Gram matrix G with p ancillas. All we need to do in each setting is determine how to get appropriate U_L and U_R to get a block encoding for the desired matrix.

From sparse access. In this setting we assume that M is s-sparse. Additionally, we require that $\|M\|_{\max} \doteq \max_{i,j} |M_{ij}| \leq 1$. This is not too extreme of a requirement because in order to even be able to have a block encoding M must have 2-norm at most 1. The construction goes as follows.

First, we can use O_{row} to create $U_L : |0^p\rangle|i\rangle \mapsto \frac{1}{\sqrt{s}} \sum_{k=1}^{s} |i\rangle|c_{ik}\rangle$. Then we use O_{column} to create $U_R : |0^p\rangle|j\rangle \mapsto \frac{1}{\sqrt{s}} \sum_{\ell=1}^{s} |r_{\ell j}\rangle|j\rangle$. If we consider the Gram matrix $U_L^* U_R$, notice that the only way for an entry $(U_L^* U_R)_{ij}$ to be nonzero is if $c_{ik} = j$ and $r_{\ell j} = i$. In this case $(U_L^* U_R)_{ij} = 1/s$, and 0 otherwise. Note that the only nonzero entries of M appear in positions (i, j) with $(U_L^* U_R)_{ij} = 1/s$. Finally, we can can use O_M to apply quantum rejection sampling on extra ancilla: $|0\rangle \mapsto M_{ij}|0\rangle + \sqrt{1 - |M_{ij}|^2}|1\rangle$. This yields a block encoding of M/s using $p + 1 = (\log s) + 1$ ancillas.

From QRAM access. In this setting, we have QRAM access to the rows and columns, and to a vector with the row norms and one with the column norms. As we will see, it will actually only be necessary to use the row accesses.

For the construction we again follow the Gram matrix approach. First, we use access to $M_{i,\cdot}$ to create

$$U_L : |0^n\rangle|i\rangle \mapsto \frac{1}{\|M_{i,\cdot}\|_2} \sum_{\ell} M_{i\ell}|i\rangle|\ell\rangle$$

Then we use access to r to create

$$U_R : |0^n\rangle|j\rangle \to \frac{1}{\|M\|_F} \sum_{k} \|M_{k,\cdot}\|_2 |k\rangle|j\rangle$$

where $\|\cdot\|_F$ is the Frobenius norm. To compute $U_L^* U_R$, we only need to consider $\ell = j$ and $k = i$. We then have $(U_L^* U_R)_{ij} = (M_{ij}/\|M_{i,\cdot}\|_2)(\|M_{i,\cdot}\|_2/\|M\|_F) = M_{ij}/\|M\|_F$. So this time we get a block encoding for $M/\|M\|_F$ with n ancillas.

References

[KP17] Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 49:1–49:21, 2017.

[Tan19] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. In *Proceedings of the 51st Annual Symposium on Theory of Computing (STOC)*, pages 217–228, 2019.