Quantum Computing 3/29/2022

Lecture 16: Input/Output Access Mechanisms and Block Encoding

Instructor: Dieter van Melkebeek Scribe: Chenhao Ye

Through quantum, people came up with algorithms with exponential speedup, some of which, however, need the input to be represented in certain ways. In this lecture, we look at the input/output mechanisms, with a case study about the recommendation problem. We then discuss QRAM access to vectors and matrices which allow us to perform operations in sub-polynomial time, even classically! QRAM access and several other I/O regimes for matrices in the literature are each encapsulated by block encoding methods, which we will discuss in detail.

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 ? above 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 will be the rank of T. Two users with similar preferences are

[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.

assumed to have similar combinations of those dimensions. It is equivalent to say that entries in the matrix are highly correlated. 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.

- *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 of the sampling on another entry's sampling. 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 of 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. Since the null space of B has dimension $N - k$, it contains a nonzero vector x whose only nonzero components 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. □

Remark 1. 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:

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

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:

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

It is the projection of the i-th row of A onto the top k singular value of A with a scaling factor for normalization.

1.1 Quantum algorithm for recommendation systems

Given that we know the optimal (in 2-norm) prediction (under the two model assumptions stated earlier), a quantum algorithm for the problem just needs to compute $|P_{i*}\rangle$, a superposition of entries in the row. Note that here we don't need the coefficients of the superposition but only the index of an entry that is large, i.e., corresponding to a product that customer i likes a lot. This can be done by simply measuring the superposition.

The computation of $|P_{i*}\rangle$ can be done efficiently within an error bound ε in time

$$O\left(\text{poly}\log\left(MN/\varepsilon\right)\right).$$

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

2 Input/output access issues

The input/output access issues arise naturally when a quantum algorithm has input and/or output that are not classical. Sometimes we can also have problems classically solvable in polynomial time but need superposition representation to achieve super-polynomial speedups, e.g., the recommendation problems above.

2.1 Input considerations

First, we may need to prepare quantum inputs from classical inputs. These could be vectors, superpositions, matrices, or some black box data structure.

Second, 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. For example, perhaps the required input arises from some fast unitary generating process. Perhaps we only need a partial reconstruction which we can feasibly generate. Or, if we are lucky, maybe our algorithm is non-destructive to the state so we can re-run it without worrying about recreating the input state. One example scenario is in phase estimation. If the state is an eigenstate, then the algorithm is non-destructive.

Finally, we may need to reflect around a state, not just generate states. For example, Grover's algorithm uses two reflections: one about the bad states, and one about the initial state which is

the uniform superposition of states. The latter can be done using the Hadamard transform and a reflection over $|0^n\rangle$, the former is handled using phase kickback based on a blackbox for the goodness predicate. 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 $R_{|0^n\rangle}$ denotes the reflection about the state $|0^n\rangle$.

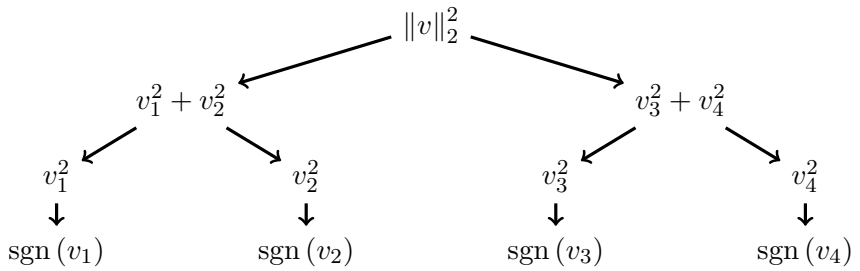
2.2 Output considerations

It deserves special attention for using a quantum output in a classical manner. For example, in the case of the recommendation system, the quantum superposition is a desired output as we can simply measure it to extract what we want, while in the case of solving linear equations, the quantum superposition may not be as helpful to produce the entire solution.

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.

One representation of a vector $v \in \mathbb{R}^N$ is given by 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.

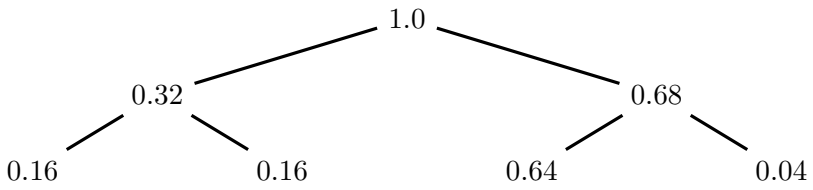


A nice property of this structure is that if we read nonzero components one at a time in an online fashion, then for each component we can update our tree by just traversing up from the corresponding leaf to the root in time $O(\log N)$. If we assume our vector v is s-sparse (it only has s nonzero entries),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

3.1 QRAM state generation

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:

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

4 Matrix encoding

There are multiple ways to encode a matrix. The first is QRAM access, which uses the QRAM vector access we just discussed.

4.1 Matrix QRAM access

QRAM access to an matrix M with rows M_{i*} indexed by i and columns M_{*j} indexed by j is defined by having each of the following:

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

4.2 Sparse matrix access

An alternative efficient accessing method is sparse access. Sparse access gives us three oracle functions that recover a given matrix entry or indices corresponding to nonzero entries.

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. This can be done for any matrix and we don't exploit the spareness here.

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.

Next, we discuss a more general encoding regime called block encoding. We will see that both QRAM and sparse encodings are special cases of block encodings.

5 Dequantization

Briefly, before we get to block encodings, there are a few interesting results about dequantization. Once we have QRAM we have some assumptions about how input is formatted as well as the assumption that we have random access to this special data structure. The structure may take time to build in the first place, but once it is done, we can access it at will. This assumption can also be exploited by classical algorithms and can even be leveraged classically to get algorithms that run, even classically, in poly-logarithmic time. 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 coin flips to get a sample according to some distribution. This is interesting because it can allow 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).

6 Block encoding

The rest of the lecture will be about the block encoding mechanism. First, recall the definition of a block encoding.

Definition 1. *A block encoding of a matrix M acting on m qubits with l ancilla qubits is a unitary circuit A acting on $l + m$ 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.

Next, we review a few properties of block encodings that can help build such an efficient unitary circuit. We have seen all of these previously, but here is a summary.

1. Given $|v\rangle$ we can compute $|Mv\rangle$. This is done by applying A to $|0^l\rangle|v\rangle$ and postselecting $|0^l\rangle$ on the first register. Note that we may need to use amplitude amplification in order to boost the probability of measuring the desired $|0^l\rangle$.

2. 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.

3. Given encodings for M_1 and M_2, we have an efficient encoding for $M_1 M_2$ which uses $l_1 + l_2$ ancillas, where l_i is the number of ancillas for the block encoding of M_i.

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

Remark 2. We can use properties 2 and 4 to approximate arbitrary polynomials of M using $\log d$ ancillas where d is the degree of the polynomial.

We will look at three settings in which we want to create a block encoding of a matrix. The first two will use a common strategy and the third will use a variation on that strategy. This strategy is to write A as a Gram matrix. Informally, a Gram matrix is made of inner products of states. More specifically:

Definition 2 (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 $U_L : |0^n\rangle|i\rangle \mapsto |\psi_i\rangle$ and $U_R : |0^n\rangle|j\rangle \mapsto |\phi_j\rangle$. Then the first few columns of U_L will be the $|\psi_i\rangle$s 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 $l = n$ 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.

6.1 Sparse Access

Recall from above that in this setting, we have access to the matrix via the following three oracles

1. $O_{row} : |i\rangle|k\rangle \mapsto |i\rangle|c_{ik}\rangle$, where c_{ik} is the column index of the k-th nonzero entry in row i

2. $O_{column} : |\ell\rangle|j\rangle \mapsto |r_{\ell j}\rangle|j\rangle$, where $r_{\ell j}$ is the row index of the ℓ-th nonzero entry in column j

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

In this setting we will assume that M is s-sparse. Additionally, we will 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 actual construction goes as follows.

First, we can use O_{row} to create $U_L : |0^n\rangle |i\rangle \mapsto \frac{1}{\sqrt{s}} \sum_{k=1}^{s} |i\rangle |c_{ik}\rangle$. Then we use O_{column} to create $U_R : |0^n\rangle |j\rangle \mapsto \frac{1}{\sqrt{s}} \sum_{\ell=1}^{s} |r_{\ell j}\rangle |j\rangle$. 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 $l = n + 1$ ancillas.

6.2 QRAM access

In this setting we have access to the matrix via the following two methods

1. QRAM access to row vectors M_{i*} and column vectors M_{*j}.
2. QRAM access to the 2-norms of these vectors: r with $r_i \doteq \|M_{i*}\|_2$ and c with $c_j \doteq \|M_{*j}\|_2$.

As we will see, it will actually only be necessary to use the row accesses. For the construction we again use the Gram matrix approach as follows.

First, we use access to M_{i*} to create

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

Then we use access to r to create

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

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 $(U_L^* U_R)_{ij} = (M_{ij}/\|M_{i*}\|_2)(\|M_{i*}\|_2/\|M\|_F) = M_{ij}/\|M\|_F$. So this time we get a block encoding for $M/\|M\|_F$ with $m = n$ ancillas.

6.3 Density operator from purification

For the last setting, we will get the block encoding of a density operator from its purification. To do this, we will use the Schmidt decomposition, which we have used before. Consider an n-qubit density operator ρ with an $(a + n)$-qubit purification $|\psi\rangle$ (using a additional ancilla qubits). We will view this as a two party system where Alice has the first a qubits and Bob has the last n. Recall that there exist $\lambda_k \in \mathbb{R}$ and two orthonormal bases $|\phi_{A,k}\rangle$ for Alice's register and $|\phi_{B,k}\rangle$ for Bob's register such that $|\psi\rangle = \sum_k \lambda_k |\phi_{A,k}\rangle |\phi_{B,k}\rangle$ and $\rho = \sum_k \lambda_k^2 |\phi_{B,k}\rangle \langle\phi_{B,k}|$. The construction of the block encoding is as follows.

First, we assume that we can generate the purification using a unitary $U : |0^{a+n}\rangle \mapsto |\psi\rangle$. Then consider $B = (U^* \otimes I_n)(I_a \otimes S)(U \otimes I_n)$ where $S : |i\rangle |j\rangle \mapsto |j\rangle |i\rangle$ is the swap operator. Then we

do the dot product

$$\begin{aligned}
\langle 0^{a+n}| \langle \phi_{B,i}| B |0\rangle^{a+n} |\phi_{B,j}\rangle &= \langle \psi| \langle \phi_{B,i}| (I_a \otimes S) |\psi\rangle |\phi_{B,j}\rangle \\
&= \left(\sum_k \lambda_k \langle \phi_{A,k}| \langle \phi_{B,k}| \langle \phi_{B,i}| \right) (I_a \otimes S) \left(\sum_\ell \lambda_\ell |\phi_{A,\ell}\rangle |\phi_{B,\ell}\rangle |\phi_{B,j}\rangle \right) \\
&= \left(\sum_k \lambda_k \langle \phi_{A,k}| \langle \phi_{B,k}| \langle \phi_{B,i}| \right) \left(\sum_\ell \lambda_\ell |\phi_{A,\ell}\rangle |\phi_{B,j}\rangle |\phi_{B,\ell}\rangle \right) \\
&= \lambda_i \lambda_j \delta_{ij} = \langle \phi_{B,i}| \rho |\phi_{B,j}\rangle
\end{aligned}$$

Since the $|\phi_{B,*}\rangle$s are a basis, this is telling us that for every $i, j \in [N]$, $\langle 0^{a+n}| \langle i| B |0^{a+n}\rangle |j\rangle = \langle i| \rho |j\rangle$, so B is a block encoding of ρ with $m = a + n$ ancillas.