CS M238 Lecture Notes

Kyle Chui

2022-09-22

Contents

1 Lecture 1

1.1 History

Quantum computers only really came into existence in 2016, and modern quantum computers have 127 qubits. We hope that the number of qubits explodes in the next few years, with Google targeting 1 million qubits by 2029.

We will *never* be able to simulate 100+ qubit quantum computers, since the state space just gets too large. Modern supercomputers can simulate a maximum of 70 qubits (Summit).

Normal computers typically have an error rate of once a year, which is mitigated by checksums and other mechanisms. To contrast, the error rate for a quantum computer is around 1%, which is a lot higher. Furthermore, we need around 1000 qubits in order to "validate" that a single qubit is correct. We call these validated qubits "perfect qubits".

Definition. *Neven's Law*
Quantum computers are gaining computational power at a doubly exponential rate.

Quantum computers are *very good* at solving linear algebra problems, e.g. machine learning or physics simulations, not *all* problems. According to experts, roughly 6000 perfect qubits are needed to be better than a classical computer.

We can't perform that many computations per qubit because of their decoherence time.

1.2 Basic Mathematics

A qubit can be represented as a unit vector in $\mathbb{C}^2$. Every computation step takes a unit vector to a unit vector, i.e. they can be represented by unitary matrices. Quantum computing uses complex numbers to allow amplitudes (read: probabilities) to be "negative", while still having positive norm.

To convert amplitudes to probabilities, we use Born's law and take the square of the norm, i.e. the probability of an event with amplitude $-\frac{1}{\sqrt{2}}$ is

$$\left(-\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}.$$

1.3 Abstractions

Classical computing languages mostly differ in how they deal with abstractions; different choices lead to different languages. At the moment, quantum computing has little to no abstractions, and the languages are all more similar (this course uses Qiskit).

While in classical computing we can choose any two registers to perform some multi-register computation, for quantum computers there are more restrictions. Operations can only be performed on neighboring qubits, as opposed to any two arbitrary registers.

2 Lecture 2

	Classical Computing	Quantum Computing
Software	Boolean Algebra	Linear Algebra
Hardware	Classical Mechanics e.g. semiconductors	Quantum Mechanics e.g. superconductors

2.1 The Four Postulates of Quantum Mechanics

2.1.1 State Space Rule

The state space is a unit vector with dimension 2^n, where n is the number of qubits. We use the following abbreviations:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The state space rule says that we are working with superpositions (or linear combinations).

2.1.2 Composition Rule

We use the tensor product to compose objects; for each additional qubit the *dimension* of our state space doubles.

2.1.3 Step Rule

We use unitary matrices to get from one state to the next state. This is because unitary matrices send unit vectors to unit vectors. For example,

$$U|\psi\rangle = |\varphi\rangle.$$

Note that this means that the size of U is $2^n \times 2^n$.

2.1.4 Measurement Rule

There is a way to get information out of a quantum computer. When we measure a computation performed with n qubits, we get n bits out of the system.

2.2 Computations

For probabilistic programming, our state is no longer a bit vector, but rather a probability vector. The matrix used to update this state is called a stochastic matrix. We can take the tensor product of two vectors to get the next state.

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}.$$

	Probabilistic Computing	Quantum Computing
Numbers	Real	Complex
State	Vector of Probabilities $\sum p_i = 1$	Vector of Amplitudes $\sum \lvert a_i \rvert^2 = 1$
Step	Stochastic Matrix	Unitary Matrix

The Hadamard matrix is given by

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Observe that

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} \\ &= \begin{pmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{pmatrix} \\ &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle. \end{aligned}$$

We also have

$$\begin{aligned} H|1\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} \\ &= \begin{pmatrix}\frac{1}{\sqrt{2}}\\ -\frac{1}{\sqrt{2}}\end{pmatrix} \\ &= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle. \end{aligned}$$

This is analogous to the "fair flip" matrix for probabilistic computing.

2.3 Quantum Computing Trick #1

One of the problems is that many of our operations are non-unitary/irreversible, so how can we use them? With the help of a helper bit b, we may define $U_f\colon \{0,1\}^2 \to \{0,1\}^2$ by

$$U_f(x,b) = (x, b \oplus f(x)),$$

where $\oplus$ is the XOR operator. Via some computation, we find that $(U_f \circ U_f)(x,b) = (x,b)$, so we have successfully made our operation reversible.

This allows us to run the quantum computation for a while. We can then stop it and reverse a few steps to help debug a program.

3 Lecture 3

3.1 Math Definitions

For the purposes of this class, a Hilbert space is a complex vector space with an inner product. We define our inner product to be

$$\left\langle \begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \middle| \begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix} \right\rangle = \alpha_1^*\beta_1 + \alpha_2^*\beta_2 = \begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix}^\dagger \begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix},$$

where $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{C}$. We are also going to use Dirac notation:

$$|\psi\rangle = \begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \quad \text{and} \quad |\varphi\rangle = \begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix}.$$

Furthermore, $|\psi\rangle^\dagger = \langle\psi|$. We also use the following notations:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Then we have $H|0\rangle = |+\rangle$ and $H|1\rangle = |-\rangle$.

> **Note.** Unitary matrices preserve inner products.

We also have a notion of an outer product, defined by

$$\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix}\begin{pmatrix}\beta_1^* & \beta_2^*\end{pmatrix} = \begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix}\begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix}^\dagger = \begin{pmatrix}\alpha_1\beta_1^* & \alpha_1\beta_2^*\\ \alpha_2\beta_1^* & \alpha_2\beta_2^*\end{pmatrix} = |\psi\rangle\langle\varphi|.$$

3.2 Change of Basis

Suppose we have two bases: $\{|0\rangle, |1\rangle\}$ and $\{|\psi\rangle, |\varphi\rangle\}$. Suppose we want to change a vector $|v\rangle$ from the former basis to the latter basis. Then we have

$$\begin{aligned}|v\rangle &= \langle\psi|v\rangle \cdot |\psi\rangle + \langle\varphi|v\rangle \cdot |\varphi\rangle\\ &= |\langle\psi|v\rangle|^2 + |\langle\varphi|v\rangle|^2.\end{aligned}$$

3.3 Multiple Qubits

Suppose we have the start state

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle.$$

Suppose we measured the first qubit to be zero. Then the new state is

$$\frac{1}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}(\alpha_{00}|00\rangle + \alpha_{01}|01\rangle) = \frac{|0\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \otimes (\alpha_{00}|0\rangle + \alpha_{01}|1\rangle).$$

3.4 Tensor Product

> **Definition.** *Tensor Product*
> We define the *tensor product* (for two vectors) to be
>
> $$\begin{pmatrix}\alpha_1\\ \alpha_2\end{pmatrix} \otimes \begin{pmatrix}\beta_1^* & \beta_2^*\end{pmatrix} = \begin{pmatrix}\alpha_1\beta_1^* & \alpha_1\beta_2^*\\ \alpha_2\beta_1^* & \alpha_2\beta_2^*\end{pmatrix},$$

which is the same as the outer product. In general, we have

$$\begin{pmatrix} \alpha_{00} & \alpha_{01} \\ \alpha_{10} & \alpha_{11} \end{pmatrix} \otimes B = \begin{pmatrix} \alpha_{00}B & \alpha_{01}B \\ \alpha_{10}B & \alpha_{11}B \end{pmatrix}.$$

Using this definition, we have

$$\begin{aligned} |00\rangle &= |0\rangle \otimes |0\rangle \\ &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \cdot 1 \\ 1 \cdot 0 \\ 0 \cdot 1 \\ 0 \cdot 0 \end{pmatrix} \\ &= \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \end{aligned}$$

Note. If we have $|\psi\rangle$ where ψ is some binary string of length n, then we have that $|\psi\rangle$ is a 2^n-dimensional vector where the ψ^{th} element (in decimal, zero-indexed) is a 1 and all the rest are 0.

3.4.1 General Rules

- The tensor product is *not* commutative, but is associative
- $A \otimes (B + C) = A \otimes B + A \otimes C$
- $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$
- $(\alpha A) \otimes B = A \otimes (\alpha B) = \alpha(A \otimes B)$

4 Lecture 4

4.1 Quantum Circuits

A quantum circuit consists of a series of "wires", and time passes from left to right. We say that at each time step or "moment", some action is performed (which takes the form of a unitary matrix). At each moment, the unitary matrix being applied is the tensor product of all of the unitary matrices on each wire. In the case that there is no matrix applied on a given wire, we substitute in the identity matrix.

> **Note.** From a physical perspective, it is actually *more* error prone to do nothing, than it is to apply some transformation. Thus real-world quantum compilers may substitute the identity for pairs of operations that cancel out, e.g. $I = Z^2$.

The CNOT gate has the control bit on top (filled circle), and the target bit on the bottom (empty circle). For example,

$$\begin{aligned} CNOT \cdot (H \otimes I) \cdot |00\rangle &= CNOT \cdot \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \end{aligned}$$

4.2 Superdense Coding

> **Theorem** — *Holevo's Theorem*
> We must use n qubits to encode n bits of information.

Consider the problem where Alice wants to share 2 bits of information (ab) with Bob using "fewer" qubits.

- Bob first creates two qubits entangled in the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, and sends one to Alice
- Alice does the following:
  - If $a = 1$, then apply Z to qubit A
  - If $b = 1$, then apply X to qubit A
  - Send qubit A to Bob
- Bob then performs the following:
  - Apply $CNOT(A, B)$
  - Apply H to A
  - Measure both A and B to get ab

Through some math, we can show that when Bob measures A, B, that the qubits collapse to ab.

4.3 No Cloning

Suppose Alice has an unknown qubit that she wants to send to Bob. We start with an Bell pair: $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, and suppose Alice's qubit is in the state $\alpha|0\rangle + \beta|1\rangle$. Then we may apply Alice's qubit to the Bell pair,

$$(\alpha|0\rangle + \beta|1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle).$$

Alice first applies the $CNOT$ gate to A, B, then the Hadamard gate to A. After the $CNOT$ transformation, our state is

$$\frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle).$$

Then after the Hadamard transformation, we have

$$\frac{1}{2}(\alpha\left|000\right\rangle + \alpha\left|100\right\rangle + \alpha\left|011\right\rangle + \alpha\left|111\right\rangle + \beta\left|010\right\rangle - \beta\left|110\right\rangle + \beta\left|001\right\rangle - \beta\left|101\right\rangle).$$

Simplifying yields

$$\frac{1}{2}(\left|00\right\rangle(\alpha\left|0\right\rangle + \beta\left|1\right\rangle) + \left|01\right\rangle(\alpha\left|1\right\rangle + \beta\left|0\right\rangle) + \left|10\right\rangle(\alpha\left|0\right\rangle - \beta\left|1\right\rangle) + \left|11\right\rangle(\alpha\left|1\right\rangle - \beta\left|0\right\rangle)).$$

Now when Alice measures her two qubits and sends them over to Bob, he can apply the following protocol:

- If $b = 1$, then Bob applies X to his qubit
- If $a = 1$, then Bob applies Z to his qubit

He then ends up with $\alpha\left|0\right\rangle + \beta\left|1\right\rangle$, the original qubit.

Notice that in the previous example, when we teleported a qubit from Alice to Bob, we *destroyed* Alice's qubit.

Theorem — *No Cloning Theorem*
It is *impossible* to clone an unknown qubit. In other words, no operation can map $\left|\psi\right\rangle\left|0\right\rangle$ to $\left|\psi\right\rangle\left|\psi\right\rangle$.

Proof. Suppose towards a contradiction that there exists some operation U such that for all $\left|\psi\right\rangle$, we have

$$U\left|\psi\right\rangle\left|0\right\rangle = \left|\psi\right\rangle\left|\psi\right\rangle.$$

We pick some $\left|\psi_1\right\rangle, \left|\psi_2\right\rangle$ that are non-orthogonal and non-proportional, i.e.

$$\left\langle\psi_1\right|\left.\psi_2\right\rangle \neq 0 \quad \text{and} \quad \left\langle\psi_1\right|\left.\psi_2\right\rangle \neq 1.$$

We compute

$$\begin{aligned}
\left\langle\psi_1\right|\left.\psi_2\right\rangle &= \left\langle\psi_1\right|\left.\psi_2\right\rangle\left\langle 0\right|\left. 0\right\rangle \\
&= \left\langle\psi_1 \otimes \left|0\right\rangle\right|\left.\psi_2 \otimes \left|0\right\rangle\right\rangle \\
&= \left\langle U(\psi_1 \otimes \left|0\right\rangle)\right|\left. U(\psi_2 \otimes \left|0\right\rangle)\right\rangle \\
&= \left\langle\left|\psi_1\right\rangle\left|\psi_1\right\rangle\right|\left.\left|\psi_2\right\rangle\left|\psi_2\right\rangle\right\rangle \\
&= \left\langle\psi_1\right|\left.\psi_2\right\rangle^2,
\end{aligned}$$

so $\left\langle\psi_1\right|\left.\psi_2\right\rangle = 0$ or $\left\langle\psi_1\right|\left.\psi_2\right\rangle = 1$. Thus we have arrived at a contradiction, so we cannot clone an unknown qubit. □

5 Lecture 5

5.1 Deutsch–Jozsa

We're given a function $f\colon \{0,1\}^n \to \{0,1\}$, which is either *constant* ($f(x)$ is always 0 or always 1), or *balanced* ($f(x)$ is 0 half the time and 1 the other half). How can we determine if f is constant or balanced?

Let's first consider the case where $n = 1$, so $f\colon \{0,1\} \to \{0,1\}$. We use the same encoding trick from previous lectures to define U_f by

$$U_f(|x\rangle \otimes |b\rangle) = |x\rangle \otimes |b \oplus f(x)\rangle\,.$$

In this case we have four cases for f, where two are constant and two are balanced:

Input	f_0	f_1	f_2	f_3
0	0	0	1	1
1	0	1	0	1

Thus we have

$$U_{f_0}(|0\rangle \otimes |b\rangle) = |0\rangle \otimes |b\rangle$$
$$U_{f_0}(|1\rangle \otimes |b\rangle) = |1\rangle \otimes |b\rangle\,.$$

Since U_{f_0} maps things to themselves, f_0 must be the identity map. Note that if f is constant, then $f(0) \oplus f(1) = 0$, and if f is balanced, then $f(0) \oplus f(1) = 1$.

Algorithm 1: Deutsch–Jozsa Algorithm

```
Take two qubits and initialize them to |0⟩ and |1⟩
Send each qubit through a Hadamard gate
Send both qubits through U_f
Send the first qubit through another Hadamard gate
Measure the first qubit
if the measurement is zero then
    return f is constant
else
    return f is balanced
```

For the generalized case, we use $n + 1$ qubits, with all but the last initially set to $|0\rangle$. We then pass the n qubits through n Hadamard gates, then all of the qubits through U_f. Finally, we pass the first n qubits through another n Hadamard gates, and measure the n qubits. If the n measured bits are all zero, then we output "constant", otherwise "balanced".

6 Lecture 6

6.1 Bernstein–Vazirani's Algorithm

Suppose we have a function $f\colon \{0,1\}^n \to \{0,1\}$. We assume that $f(x) = a \cdot x \oplus b$ for some n-bit string a, and singular bit b. We would like to find a, b.

> **Note.** If $a = 0$, then $f(x) = x \cdot 0 \oplus b = b$, and so f is constant. If $a \neq 0$, then each bit string x has an inverse, and so f is balanced.

6.2 Big Ideas

- Superposition (apply f to everything in one shot)
- We are again going to use U_f to have the operations be reversible
  - This will move "something" into the exponent (of -1)
- The Hadamard gate will move the exponent (a) back down

Our goal is to find the final state $|a\rangle$.

Algorithm 2: Bernstein–Vazirani's Algorithm

1. We initially have all n qubits in state $|0\rangle$, and a helper qubit in state $|1\rangle$
2. Apply the Hadamard gate to all $n+1$ qubits
3. Apply U_f to all qubits
4. Apply the Hadamard gate to the first n qubits
5. Measure the end state to get a

By doing some math, we can show that the final state of the circuit is $(-1)^b |a\rangle$ with 100% probability, as desired.

7 Lecture 7

7.1 Simon's Problem

Suppose we have some function $f\colon \{0,1\}^n \to \{0,1\}^n$. We assume that there exists some $s \in \{0,1\}^n$ such that for all $x, y \in \{0,1\}^n$, we have $f(x) = f(y)$ if and only if $x + y \in \{0^n, s\}$. We wish to find s.

7.2 Simon's Algorithm

The high-level idea here is that we use a quantum computer to map f to a series of equations, and then use classical computers to find s. Since this is a probabilistic computation, we repeat this process until we're "fairly certain" that we have found s.

> **Note.** It is very easy to check if a given guess s is a valid solution to the problem. We can just choose an arbitrary x and compare $f(x)$ with $f(x \oplus s)$.

After running the quantum computer, we will end up with $n-1$ equations of the form

$$\begin{aligned} y_1 \cdot s &= 0 \\ y_2 \cdot s &= 0 \\ &\vdots \\ y_{n-1} \cdot s &= 0. \end{aligned}$$

Ideally we wish all y_i to be linearly independent, so s is orthogonal to the $n-1$ bit strings. The probability that all of the y_i's are linearly independent is greater than $\frac{1}{4}$.

8 Lecture 8

8.1 Grover's Algorithm

The main ideas of this algorithm are *amplitude amplification* and *quadratic speedup*.

We wish to search in an unstructured database. We may restructure this problem as: we are given a function $f\colon \{0,1\}^n \to \{0,1\}$, and wish to find x such that $f(x) = 1$. We assume that there exists exactly one such x.

We use the notation:

- $z_f \left|x\right\rangle = (-1)^{f(x)} \left|x\right\rangle$
- $z_0 \left|x\right\rangle = \begin{cases} -\left|x\right\rangle & \text{if } x = 0^n \\ \left|x\right\rangle & \text{otherwise} \end{cases}$

> **Note.** Both of these can be computed using basic quantum gates, but they get quite large/involved, so we assume that they can be constructed.

Let

$$G \left|x\right\rangle = -H^{\otimes n} z_0 H^{\otimes n} z_f \left|x\right\rangle .$$

Algorithm 3: Grover's Algorithm

```
Set x to be n qubits, all in the zero state
Apply n Hadamard gates to X
for  do
    Apply G to x (O(√(2^n)) times)
Measure x and get the result
```

9 Lecture 9