Lecture 1 Notes: Bits and Qubits

Quantum Computing 101

1 Bits as vectors

Before talking about qubits, we must first become comfortable with treating bits (and sequences of bits) as vectors. We'll build up an understanding of how classical circuits like **OR**, **AND**, and **NOT** can be thought of as matrix operations on those vectors.

1.1 0, 1 and more

A single bit can be in two possible states (either 0 or 1), so let's try representing that bit as a length-2 vector, with a 1 in the position that the bit has, and a 0 otherwise. The vector

$$v^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

represents the bit 0, and the vector

$$v^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

represents the bit 1. (This may be a bit hard to follow at first, but the index of the **1** element in the vector tells you which bit is on). How would we implement the **NOT** gate, such that $\mathbf{NOT}v^{(0)} = v^{(1)}$ and $\mathbf{NOT}v^{(1)} = v^{(0)}$? We can do this with the matrix

$$\mathbf{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

You can quickly verify that this has the desired properties. Before moving on, note that the vector

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

doesn't really make in this context; it's essentially saying that the bit is *both* 0 and 1 at the same time.

1.2 Combining bits

How would we represent, say, two bits at once in a vector? Since two bits can be in four total states (00, 01, 10, 11), a natural way would be to have a length-4 vector that is all zeroes, except 1 in the

position that the two bits take. Ordering the possible states that way (00, 01, 10, 11, we can index the vector. For example,

$$v^{(10)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

represents the two-bit state 01. *Note that this is different than 01*, which has a 1 in the previous position. One way to think of combining bits like this is by treating the possible states as binary numbers, so the state 10 (which is just 2 in binary) has the (zero-indexed) second element set to 1.

Most of the interesting gates in classical computing involve at least two bits, so let's look at the **AND** gate. **AND** takes two bits and returns a single bit: 1 if they're both set to 1, and 0 otherwise. Explicitly, $\mathbf{AND}v^{(00)} = v^{(0)}$, $\mathbf{AND}v^{(01)} = v^{(0)}$, $\mathbf{AND}v^{(10)} = v^{(0)}$, and $\mathbf{AND}v^{(11)} = v^{(1)}$.

Represented as a matrix,

$$\mathbf{AND} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Think about how you might implement **OR** as a matrix: it would look very similar.

One thing to note about **AND** is that it's a *lossy operation*: given that the output of some **AND**-gate is 0, you don't really know what the input was. Contrast this to **NOT**, where the computation was *reversible*, meaning you could tell what the input was to the gate, based on the output you got. This will be important later.

2 Enter probability...

So far we've dealt with bits that were *always* either 0 or 1. Consider a bit that has some randomness to it: 30% of the time it's 0, and 70% of the time it's 1. A natural way to represent this bit in our vector format is

$$v = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}$$

where each element is the probability that the bit takes on that value. Think about any such vector representing a probabilistic bit. What has to be true? Well, we can't have any negative probabilities ($v_i \geq 0$ for all v_i). Also, the probabilities need to sum to 1 ($\sum_i v_i = 1$). In other words, the bit needs to take on *some* value. **These are the two axioms of discrete probability distributions**.

2.1 Combinations

The process for combining probabilistic bits is essentially the same as when we had no randomness, except now it involves a bit(!) of multiplication. For example, consider bits a and b. a is 0 half the time and 1 half the time, while b is 0 30% of the time and 1 70% of the time (as in the previous example). Assuming a and b are independent, what's the distribution of values of ab?

Well,

$$a = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$b = \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix}$$

and the "product" ab takes on the value 00 with probability 0.5×0.3, 01 with probability 0.5×0.7, 10 with probability 0.5×0.3 and 11 with probability 0.5×0.7. Putting it together (using the same ideas as before for indexing),

$$ab = \begin{bmatrix} 0.15 \\ 0.35 \\ 0.15 \\ 0.35 \end{bmatrix}$$

This may seem like a lot of math, but it works exactly how you'd expect it to.

2.2 Transitions

Consider the following "probabilistic gate" **T**:

1. if the bit is 0, keep it as 0
2. if the bit is 1, keep it as 1 with probability 0.5, and flip it to 0 with probability 0.5

For the 30%/70% bit that we've been using throughout, the result of putting it through **T** is a new bit that is 0 65% of the time, and 1 35% of the time. We can think of this gate more generally as (you guessed it!) a matrix operating on the bit vector. A bit of thought shows that

$$\mathbf{T} = \begin{bmatrix} 1 & 0.5 \\ 0 & 0.5 \end{bmatrix}$$

Which will always perform that gate's operation. You can double-check that

$$\mathbf{T} \begin{bmatrix} 0.3 \\ 0.7 \end{bmatrix} = \begin{bmatrix} 0.65 \\ 0.35 \end{bmatrix}$$

What sort of matrices give valid transitions? By valid, we mean that given a valid (according to the axioms above) probability vector as input, it always returns a valid probability vector as output.

It turns out that the necessary and sufficient condition for a valid transition matrix is that *all of the columns need to sum to 1.* (There's the other condition that none of the entries can be negative, but that's more obvious). Matrices that satisfy this property are called *stochastic matrices.*

2.3 Measurement

Randomness is great, but eventually we'll probably want to know what configuration our bit system is actually in. This is called *measurement.* This also works the way you'd expect: once you decide to

measure the system, the whole thing "collapses" into a deterministic, non-random state, according to the probabilities specified by the system. For example, our 2-bit state

$$ab = \begin{bmatrix} 0.15 \\ 0.35 \\ 0.15 \\ 0.35 \end{bmatrix}$$

once it is measured, will be in the state $[1, 0, 0, 1]^T$ with probability 0.15, $[1, 0, 0, 1]^T$ with probability 0.35, et cetera.

3 A diversion: complex numbers

One **final** thing before we get to qubits: a quick refresher on complex numbers. You can skip this if you're comfortable with these.

Complex numbers are an extension of the real numbers along another axis. Any complex number can be written as $c = a + bi$, where a and b are real numbers, and $i = \sqrt{-1}$. a is often called the "real part", and bi is often called the "imaginary part".

While this whole system might seem a bit arbitrary, complex numbers are extremely important in both math and science, and it's the number system that underpins quantum mechanics (and by extension, the universe apparently.)

There's two important things you should know about complex numbers right now, and they both relate to the **complex conjugate**. The complex conjugate of a number (denoted c^*) is obtained by simply negating its imaginary part: $(a + bi)^* = a - bi$. Note that the real part stays the same.

One is that the **magnitude** of a complex number, $|a + bi|$ is the square root of the product of the number with its complex conjugate. In other words, $|a+bi| = \sqrt{(a + bi)(a - bi)}$, which simplifies to $\sqrt{a^2 + b^2}$. Note that this is always a real number! We often care about the *square* of the magnitude, $|a + bi|^2 = a^2 + b^2$.

The other is the idea of the **conjugate transpose** of a matrix. A matrix with complex entries, $\mathbf{A}$, has a conjugate transpose (denoted $\mathbf{A}^\dagger$) which is obtained by transposing the matrix and taking the complex conjugate of all of the entries. For example, if

$$\mathbf{A} = \begin{bmatrix} 1 & -2 - i \\ 1 + i & i \end{bmatrix}$$

then

$$\mathbf{A}^\dagger = \begin{bmatrix} 1 & 1 - i \\ -2 + i & -i \end{bmatrix}$$

4 Qubits

We're finally here! Yahoo.

At their core, qubits are what you'd get if you took the ideas from section 2 about probabilistic bits, and let the probabilities have complex values. Sound crazy? Read on.

We had a few different rules about how probabilistic bits worked: what constraints there were on the vectors, how you combined bits together, how they transitioned, and how they were measured. We're going to draw some direct analogies between these ideas and quantum computing.

In general, a quantum state is represented as a vector of values (here not called probabilities, but *amplitudes*), that can be negative, positive, imaginary, complex... really whatever you'd like them to be. The only constraint we have is that the 2-norm of the vector needs to equal 1. What does that mean? Well, for real numbers, the 2-norm is the square root of the sum of the squares of the values in the vector:

$$\sqrt{\sum_i v_i^2}$$

For complex numbers, it's the same thing, except we use the magnitude of the complex numbers, instead of just the values:

$$\sqrt{\sum_i |v_i|^2}$$

Since we want this to be 1, to make it easier to think about, we can just square both sides and say that the necessary and sufficient condition for a complex state vector to be valid is that

$$\sum_i |v_i|^2 = 1$$

For example, the vector

$$\begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}$$

Has a 2-norm equal to 1 (try it out yourself!), so it's a valid state representation, and could be thought of as the "state" of a single qubit.

This may all sound quite bizarre: why are we using complex numbers instead of probabilities? To be totally honest, I don't think anyone really knows, but we do know that nature seems to work this way, and can't quite fully be explained by real numbers. (The history of quantum mechanics is worth reading about).

Now that we have a handle on these vectors, let's go through the analogies to the probabilistic bit vectors.

4.1 Measurement

To measure a complex state, we look at the squares of the magnitudes of the complex numbers in the vector, and pick one of the configurations by treating the square of the probability as the

magnitude. Perhaps an example will help:

$$v = \begin{bmatrix} \frac{-1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{bmatrix}$$

The squared magnitude of the first entry $|\frac{-1}{\sqrt{2}}|^2$, is 0.5 because $(-1/\sqrt{2})^2 = 0.5$. The same is true of the second entry. So if we had a complex system in this state and we wanted to measure it, half the time it would be 0, and half the time it would be 1.

Convince yourself that if we have a vector with a 2-norm equal to 1, then this process gives a valid probability distribution! In particular, note that the squared magnitude is never negative (because it's $a^2 + b^2$ for some real numbers a, b), and we know that the sum of the squared magnitudes is 1.

One confusing thing about measurement of a quantum state: once it's measured, that configuration is deterministic: it "collapses" into whatever state you measured it as. In other words, measurement is a physical process that changes the state. There's a lot more to be said about this topic, but now's not really the time. Just be aware that measuring a quantum state is a crucial (perhaps *the* crucial) part of quantum computing.

4.2 Combinations

Combining two complex states works the same way as probabilistic bits: you simply multiply the amplitudes of each state pair together to get a new one. Note that it's complex multiplication instead of real, but there's nothing else to say here.

4.3 Transitions

Just as stochastic matrices can specify "valid" transitions from one probabilistic bit vector to another (where the property we wanted to preserve was that the vector summed to 1), we would like to find what sort of matrices preserve the property that the 2-norm is 1.

I'll spare you the math and say that the type of matrices we're looking for are **unitary matrices**: these are the matrices $\mathbf{U}$ such that $\mathbf{U}$ multiplied with its conjugate transpose $\mathbf{U}^\dagger$ gives the identity matrix I. This is the most general type of matrix that preserves the 2-norm for all input vectors.

Here's an example:

$$\mathbf{U} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{bmatrix}$$

Using the conjugate transpose (convince yourself this works!)

$$\mathbf{U}^\dagger = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-i}{\sqrt{2}} \end{bmatrix}$$

The product $\mathbf{U}\mathbf{U}^\dagger$, if you do the math, is the identity matrix. So $\mathbf{U}$ is indeed unitary.

Let's see how we'd use this to transition a qubit state. Assume the qubit is currently represented by

$$v = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

This means that if we were to measure it, there would be a 50% chance of getting a 0, and a 50% chance of getting a 1. The product

$$\mathbf{U}v = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-i}{\sqrt{2}} & \frac{i}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

In other words, applying this transformation to our qubit makes it completely deterministic: it's 100% likely to be a 0 now.