CS 269Q Lecture 4

Programming a quantum algorithm with pyQuil

Peter Karalekas

peter@rigetti.com

April 11, 2019

We build out our quantum computing toolbox so that we can implement our very first quantum algorithm. We begin by working through a couple foundational concepts—the wavefunction, classical logic, and quantum parallelism—and then conclude with a walkthrough of the protocol for Deutsch's algorithm. Alongside these notes, there is a Jupyter notebook that uses Rigetti Computing's open-source software development kit, called Forest, to step through the algorithm and inspect the wavefunction.

1 The wavefunction and quantum circuits

We represent the state of our qubit using the "ket" $|\psi\rangle$, which is a superposition of the states $|0\rangle$ and $|1\rangle$ that form a basis for a two-dimensional complex vector space ($\mathbb{C}^2$). We additionally have a normalization condition on our complex amplitudes α and β such that their magnitudes squared must sum to 1.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \qquad |\alpha|^2 + |\beta|^2 = 1 \tag{1}$$

1.1 1Q states

For our quantum virtual machine (QVM), each qubit begins in the $|\psi\rangle = |0\rangle$ state. We then apply quantum gates to evolve the state of the QVM, and each gate U must be unitary, meaning that $U^\dagger U = I$. We can use an X gate to flip between $|0\rangle$ and $|1\rangle$.

$$X|0\rangle = |1\rangle \qquad X|1\rangle = |0\rangle \tag{2}$$

In addition to swapping between the 1Q computational basis states, we can use quantum gates to create superposition states. Here we create the 1Q superposition states $|+\rangle$ amd $|-\rangle$ using the H (Hadamard) gate.

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle \qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle \tag{3}$$

1.2 2Q states

For two-qubit states, we can create "product states" of computational basis states, which just look like two-bit bitstrings. Here we create the 2Q computational basis state $|10\rangle$.

$$(X \otimes I)|00\rangle = |10\rangle \tag{4}$$

We can also create product states of superposition states. Here we create the 2Q superposition state $|+,+\rangle$ using two Hadamard gates.

$$(H \otimes H)|00\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = |+,+\rangle \tag{5}$$

In addition to product states, we can create entangled states, which we can no longer factor into the tensor product of two individual qubit states. Here we create the Bell state $|\Phi^+\rangle$.

$$\text{CNOT}_{0,1}(I \otimes H)|00\rangle = \text{CNOT}_{0,1}|0\rangle \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = |\Phi^+\rangle \tag{6}$$

2 Classical logic and function evaluation

In Computer Science, we learn about Boolean logic gates like `NOT`, `AND`, `OR`, and `XOR`. In quantum computing, we can implement these classical logic gates, but we must do so in a way that respects the unitarity requirements of quantum logic gates.

2.1 Boolean functions of 1-bit domain

$$x \in \{0,1\} \qquad f(x) \to \{0,1\} \tag{7}$$

One-bit boolean functions represent the simplest classical logic we can implement on a quantum computer. There are four possible one-bit functions $f(x)$, and we will work through all of them.

2.1.1 Balanced functions

$$\text{Balanced-}I : (0 \to 0, 1 \to 1) \qquad \text{Balanced-}X : (0 \to 1, 1 \to 0)$$

For the balanced 1-bit functions, it's pretty easy to come up with a quantum circuit that works. If we use just an I gate for Balanced-I and just an X gate for Balanced-X, we can produce a quantum circuit U_f that maps $|x\rangle \to |f(x)\rangle$. Knowing the gate and and output, we can reproduce the input, which means our circuit satisfies our reversibility requirements. Below, we have the quantum circuit for Balanced-I on the left, and Balanced-X on the right.

(8)

2.1.2 Constant functions

$$\text{Constant-0} : (0 \to 0, 1 \to 0) \qquad \text{Constant-1} : (0 \to 1, 1 \to 1)$$

Coming up with the circuit for the constant functions seems less trivial. We can write down a matrix M_f that maps the 0 state to the 0 state and the 1 state to the 0 state.

$$M_f = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \tag{9}$$

However, this matrix has some problems. It is not invertible (determinant 0) and it is not length preserving (superposition state changes length), and therefore it is not unitary. We can also see that it is not reversible simply from the truth table—knowing the output and the gate isn't enough to get back to the input.

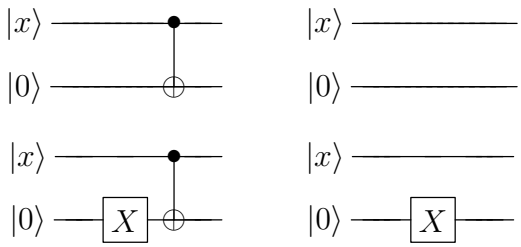
$$\det(M_f) = \det \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} = 1 \cdot 0 - 1 \cdot 0 = 0 \tag{10}$$

$$M_f|\psi\rangle = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0.6 \\ 0.8 \end{pmatrix} = \begin{pmatrix} 1.4 \\ 0.0 \end{pmatrix} \tag{11}$$

2.1.3 Ancilla qubits

In order to write this function as a quantum circuit, we need to introduce a new concept—the ancilla qubit. An ancilla qubit is an additional qubit used in a computation that we know the initial state of. Using an ancilla, we can produce a quantum circuit U_f that maps $|0, x\rangle \to |f(x), x\rangle$. Now, we can come up with a unitary matrix (albeit a trivial one) that allows us to evaluate constant functions. For the Constant-0, we just simply do nothing to the ancilla, and its state encodes $f(x)$. And for the Constant-1, all we have to do is flip the ancilla with an X gate, and we get $f(x)$ for all x. Below, we have the Balanced-I (top left), Balanced-X (bottom left), Constant-0 (top right), and Constant-1 (bottom right) functions implemented as quantum circuits with one ancilla qubit. We will continue to use this ordering for sections 2.3 and 2.4.

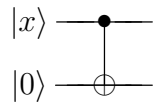
$$|0, x\rangle \to |f(x), x\rangle \tag{12}$$



2.2 The Quantum XOR gate

The Boolean function XOR (for "exclusive or") takes in two bits x and y and returns 1 if and only if the values of the bits are different from one another. Otherwise, it returns 0. The operation is written as $x \oplus y$, and although it is a two-bit function, we can implement it as a quantum circuit without an ancilla, by simply using the CNOT gate.

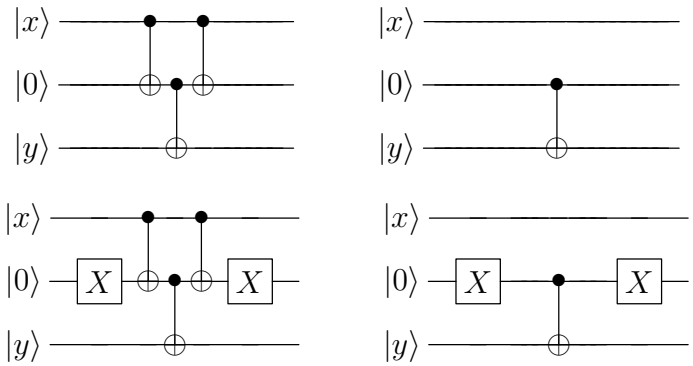
$$\text{CNOT}_{0,1}|y, x\rangle = |y \oplus x, x\rangle \tag{13}$$



2.3 Deutsch Oracle

In Deutsch's algorithm, we are given something called an oracle (referred to as U_f), which maps $|y, x\rangle \to |y \oplus f(x), x\rangle$, and the goal is to determine a global property of the function $f(x)$ with as few queries to the oracle as possible. We can combine the two concepts above (one-bit function evaluation with ancillas, and the XOR gate), to produce the four implementations of the Deutsch Oracle with one ancilla qubit.

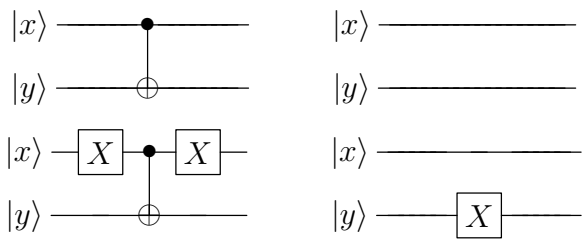
$$U_f : |y, 0, x\rangle \to |y \oplus f(x), 0, x\rangle \tag{14}$$



2.4 Optimized Deutsch Oracle

For pedagogical reasons, it is nice to separate out the three steps in the Deutsch Oracle—evaluate $f(x)$, calculate $y \oplus f(x)$, and then return the ancilla to $|0\rangle$. But, in practice we always want to implement our circuits in as few gates as possible (this is especially important when running on a real, noisy quantum computer!). Below, we show how we can rewrite each of the four Deutsch Oracle implementations (which we call U_f) without the need for an ancilla qubit.

$$U_f : |y, x\rangle \rightarrow |y \oplus f(x), x\rangle \tag{15}$$



3 Quantum parallelism

In the previous section, we showed that we could implement classical logic using quantum circuits. However, when using a computational basis state ($|0\rangle$ or $|1\rangle$), we don't do anything more interesting than a classical computer can do. If we instead feed a superposition state into one of these circuits, we can effectively evaluate a function $f(x)$ on multiple values of x at once!

$$U_f : |0, +\rangle \rightarrow \frac{|f(0), 0\rangle + |f(1), 1\rangle}{\sqrt{2}} \tag{16}$$

$$U_f : |0, -\rangle \rightarrow \frac{|f(0), 0\rangle - |f(1), 1\rangle}{\sqrt{2}} \tag{17}$$

It is important to note, that although this quantum parallelism concept is interesting, we are unable to learn about both $f(0)$ and $f(1)$ when the states above are in that form. This is due to the fact that we can only extract one classical bit of information from a quantum computer (of 1 qubit) when we measure it. But, as we will find in Deutsch's algorithm below, we can cleverly take advantage of quantum parallelism to do things that a classical computer cannot, even with the constraint that measurement yields only one classical bit.

3.1 Test with Balanced-I

To verify, we run the Balanced-I circuit for an input that is in a superposition state.

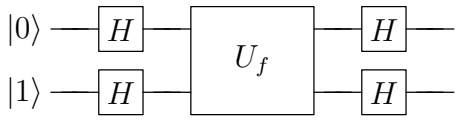
$$\text{CNOT}_{0,1}|0, +\rangle = \frac{\text{CNOT}_{0,1}|00\rangle + \text{CNOT}_{0,1}|01\rangle}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = |\Phi^+\rangle \tag{18}$$

$$\text{CNOT}_{0,1}|0, -\rangle = \frac{\text{CNOT}_{0,1}|00\rangle - \text{CNOT}_{0,1}|01\rangle}{\sqrt{2}} = \frac{|00\rangle - |11\rangle}{\sqrt{2}} = |\Phi^-\rangle \tag{19}$$

In both of our output states, we can see that if we take the state of qubit 0 in each ket to be x, the state of qubit 1 in the corresponding ket is equal to $f(x)$ (as $f(x) = x$ for Balanced-I).

4 Deutsch's algorithm

Goal: Determine if function $f(x)$ is *constant* ($f(0) = f(1)$) or *balanced* ($f(0) \neq f(1)$).



(20)

As part of the algorithm, we are given a Deutsch Oracle and are unaware of which one-bit Boolean function $f(x)$ it implements. We show that we can do this with only one query to the Deutsch Oracle, which is impossible on a classical computer, which would require two queries to the Deutsch Oracle to determine this global property of $f(x)$.

4.1 Initial state

We begin our algorithm in the computational basis state $|10\rangle$. The fact that the states for qubit 0 and qubit 1 are different proves to be important.

$$|\psi_0\rangle = |10\rangle \tag{21}$$

4.2 Prepare superpositions

We cannot do anything interesting with computational basis states, so to take advantage of quantum parallelism we put our qubits in superposition states.

$$|\psi_1\rangle = (H \otimes H)|\psi_0\rangle = (H \otimes H)|10\rangle = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = |-,+\rangle \tag{22}$$

4.3 Apply the Deutsch Oracle

We learned earlier that the action of the Deutsch Oracle on input state $|y, x\rangle$ is $U_f|y, x\rangle \rightarrow |y \oplus f(x), x\rangle$. So, what happens if we apply the Deutsch Oracle to the input state $|-, x\rangle$?

$$\begin{aligned} U_f|-,x\rangle &= \frac{U_f|0,x\rangle - U_f|1,x\rangle}{\sqrt{2}} \\ &= \frac{|0 \oplus f(x), x\rangle - |1 \oplus f(x), x\rangle}{\sqrt{2}} \\ &= \begin{cases} \dfrac{|0,x\rangle - |1,x\rangle}{\sqrt{2}} = (+1)|-,x\rangle & \text{if } f(x) = 0; \\ \dfrac{|1,x\rangle - |0,x\rangle}{\sqrt{2}} = (-1)|-,x\rangle & \text{if } f(x) = 1. \end{cases} \end{aligned} \tag{23}$$

These two branches can be unified, as we see in the following equation.

$$U_f|-,x\rangle = (-1)^{f(x)}|-,x\rangle \tag{24}$$

Thus, we get a negative sign if $f(x) = 1$, and the state is unchanged if $f(x) = 0$. However, something interesting happens when we apply U_f to the state $|-,+\rangle$, which is $|\psi_1\rangle$.

$$\begin{aligned}
|\psi_2\rangle = U_f|\psi_1\rangle &= U_f|-,+\rangle \\
&= \frac{U_f|-,0\rangle + U_f|-,1\rangle}{\sqrt{2}} \\
&= \frac{(-1)^{f(0)}|-,0\rangle + (-1)^{f(1)}|-,1\rangle}{\sqrt{2}} \\
&= |-\rangle \otimes \left(\frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}}\right) \\
&= \begin{cases} \pm|-\rangle \otimes \left(\dfrac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \pm|-,+\rangle & \text{if constant;} \\ \pm|-\rangle \otimes \left(\dfrac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \pm|-,-\rangle & \text{if balanced.} \end{cases}
\end{aligned} \tag{25}$$

If $f(x)$ is balanced, this has the effect of changing the relative phase between the $|0\rangle$ and $|1\rangle$ components of qubit 0's state, which flips it from $|+\rangle$ to $|-\rangle$. This is interesting, because the action of our oracle on the computational basis state $|y,x\rangle$ is to change the state of qubit 1 and leave qubit 0 alone. But, when our qubits are in superposition states, the balanced oracle actually changes the state of qubit 0 (which is the control qubit in the oracle), and leaves alone the state of qubit 1 (which is the target qubit in the oracle).

4.4 Return to 2Q computational basis states

We know that the outcome of the previous step is to produce one of two superposition product states, $|-,+\rangle$ or $|-,-\rangle$. However, our goal is to query the Deutsch Oracle as few times as possible, and so although we can see that these states are different, we cannot distinguish them with a single measurement (as we only get a 0 or a 1). Therefore, we use the Hadamard gate to return to 2Q computational basis states that can be distinguished in one measurement.

$$|\psi_3\rangle = (H \otimes H)|\psi_2\rangle = \begin{cases} \pm(H \otimes H)|-,+\rangle = \pm|10\rangle & \text{if constant;} \\ \\ \pm(H \otimes H)|-,-\rangle = \pm|11\rangle & \text{if balanced.} \end{cases} \tag{26}$$

Thus, we are in two distinct 2Q computational basis states, dependent on the nature of $f(x)$. We could then measure the state of qubit 0 one time, and we would immediately know the answer to whether $f(x)$ is constant or balanced.

4.5 Conclusions

So, we were able to learn about a *global property* of the function $f(x)$ in just one query to the Deutsch Oracle, which is impossible on a classical computer. Although the problem statement for Deutsch's algorithm is a bit contrived, if you can suspend your judgment, you can imagine that we could take some of the non-classical concepts of this algorithm and apply them to a more complex scenario to actually produce an interesting quantum speedup. And, later in the course, you will do exactly this!