Task 1

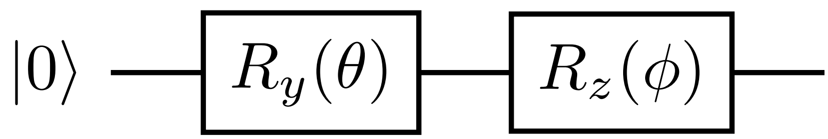
Trent Fridey

February 15, 2021

Contents

1 Part 1

A variational quantum circuit that is able to generate the most general 1-qubit state, starting from the initial state $|0\rangle$ is the following:



Proof:
A general state on the Bloch sphere can be parameterized by two angles, $\theta \in [0, \pi)$ and $\phi \in [0, 2\pi)$, as:

$$|\theta, \phi\rangle = \begin{bmatrix} \cos(\theta/2) \\ e^{i\phi}\sin(\theta/2) \end{bmatrix}$$

The matrix representation of our circuit is:

$$\begin{aligned} R_z(\phi)R_y(\theta) &= \begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix} \begin{bmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta/2)e^{-i\phi/2} & -\sin(\theta/2)e^{i\phi/2} \\ \sin(\theta/2)e^{-i\phi/2} & \cos(\theta/2)e^{i\phi/2} \end{bmatrix} \end{aligned}$$

Our initial state is $|0\rangle$. The resulting state after the circuit is:

$$\begin{aligned} R_z(\phi)R_y(\theta)|0\rangle &= \begin{bmatrix} \cos(\theta/2)e^{-i\phi/2} & -\sin(\theta/2)e^{i\phi/2} \\ \sin(\theta/2)e^{-i\phi/2} & \cos(\theta/2)e^{i\phi/2} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= e^{-i\phi/2} \begin{bmatrix} \cos(\theta/2) \\ e^{i\phi}\sin(\theta/2) \end{bmatrix} = |\theta, \phi\rangle \end{aligned}$$

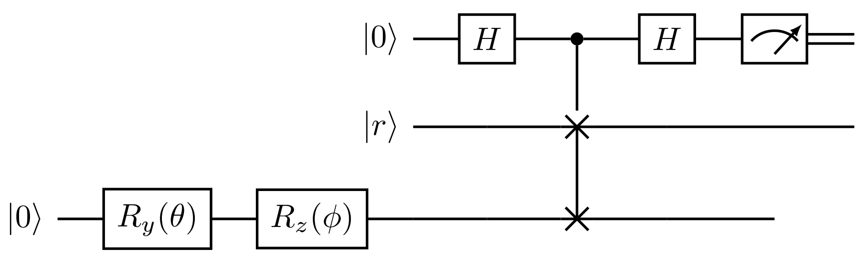
which is equivalent to $|\theta, \phi\rangle$ modulo a global phase. $\blacksquare$

2 Part 2

Let $|r\rangle$ be a random 1-qubit state. From above, we know that it can be written as $|r\rangle = |\theta_0, \phi_0\rangle$ for some angles θ_0, ϕ_0. Since we don't know the angles ahead of time, we can't find the random state by just tuning our gates to $R_z(\phi_0)$ and $R_y(\theta_0)$. Instead, we can use the SWAP test and machine learning to discover the angles. The principle behind this approach is minimizing a cost function in terms of the overlap between the random state $|r\rangle$ and the controlled state $|\theta, \phi\rangle$

$$\text{cost} \propto 1 - |\langle r|\theta, \phi\rangle|^2$$

We can implement this cost function using the SWAP test as in the following circuit:



Proof:
The wavefunction of the system immediately before measurement is:

$$\frac{1}{2}|0\rangle\left(|\theta, \phi\rangle|r\rangle + |r\rangle|\theta, \phi\rangle\right) + \frac{1}{2}|1\rangle\left(|\theta, \phi\rangle|r\rangle - |r\rangle|\theta, \phi\rangle\right)$$

Let M be the value measured of the ancilla qubit, in the z - basis, $\{|0\rangle, |1\rangle\}$, returning the value 0 or 1 respectively. Suppose we run the circuit n times, so that we obtain a set of measurements $\{M_i\}_{i=1}^{n}$. Then in the limit $n \to \infty$, we can compute the overlap:

$$\lim_{n\to\infty}\left(1 - \frac{2}{n}\sum_{i=1}^{n} M_i\right) = |\langle r|\theta, \phi\rangle|^2$$

The extension to the cost function is straightforward $\blacksquare$.
We then run this circuit many times ($n \gg 1$) so that we can approach the limiting value as mentioned in the proof. Once we have the resulting

measurements, the optimal angles θ_0, ϕ_0 are the solution to the minimization problem:

$$(\theta_0, \phi_0) = \arg\min_{(\theta,\phi)} \frac{2}{n} \sum_{i=1}^{n} M_i$$

We can solve this via an appropriate minimization algorithm.

2.1 Implementation

We can implement the circuit and the optimization using the Pennylane Python library:

```
import pennylane as qml
from pennylane import numpy as np
```

For the purposes of illustration, in this implementation we let the random state be $|r\rangle = |\theta_0 = \pi/8, \phi_0 = \pi/4\rangle$. In reality, in order to be actually random, we would not know this ahead of time.

```
t_0 = 0.125*np.pi # ~ 0.3927
p_0 = 0.25*np.pi  # ~ 0.7854
```

Now we are ready to implement our circuit. We use `QubitStateVector` to define the random state:

```
num_shots = 1000
dev = qml.device('default.qubit', wires=3, shots=num_shots)

@qml.qnode(dev)
def var_swap(params):
  # random state preparation
  qml.QubitStateVector([np.cos(t_0/2.), np.sqrt(0.5)*(1+1j)*np.sin(t_0/2.)], wires=1)
  # actual circuit follows:
  qml.RY(params[0], wires=2)
  qml.RZ(params[1], wires=2)
  qml.Hadamard(wires=0)
  qml.CSWAP(wires=[0,1,2])
  qml.Hadamard(wires=0)
  return qml.sample(qml.PauliZ(0))
```

Now we set up the optimizer. Define the cost function:

```
def cost(params):
    return 1-(1./num_shots)*np.sum(var_swap(params))
```

Define the initial parameters:

```
params = np.array([0.,0.])
```

Now we define the optimization algorithm. Because our cost function has a stochastic output (via the call to `qml.sample`), we are limited to using *gradient-free* methods of optimization. In this case we can use Rotosolve via the `RotosolveOptimizer`.[1]

```
opt = qml.RotosolveOptimizer()
```

So with that we are ready to run the optimization:

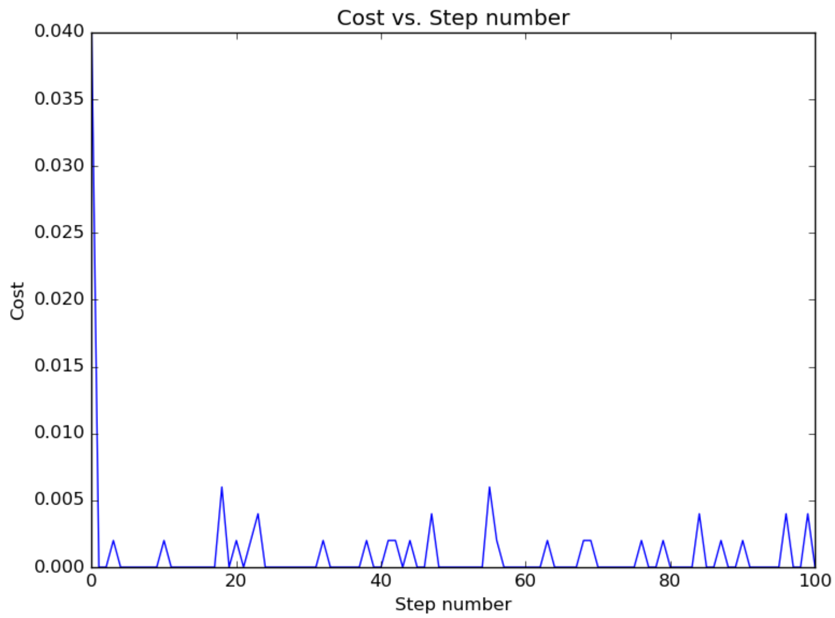
```
steps = 100
best_cost = [cost(params)]
best_params = params
for i in range(steps):
  params = opt.step(cost, params)
  new_cost = cost(params)
  if(new_cost < best_cost[-1]):
    best_params = params
  best_cost.append(new_cost)

print(best_params)
```

```
[0.3565424  0.64350111]
```

The optimization gives $\theta =$ `0.3565423996331807` and $\phi =$ `0.6435011087932843`, which differs from the actual best parameters by:
$\theta - \pi/8 = -0.0362, \phi - \pi/4 = -0.142$

[1]Pennylane also has an API for the RotoselectOptimizer, which not only can find the best θ, ϕ minimizing the cost function, but can also find the best rotation gates for us! However, we already know the best set of rotation gates – they are the gates we used to prepare the "random" state.

We visually check the progression of our optimizer with a plot of cost-vs-steps:



3 Part 3

In this part we extend the analysis from the first two parts to perform a SWAP test on a N - qubit product state.
Now the random state $|r\rangle$ has each qubit in either the $|0\rangle$ or $|1\rangle$ state:

$$|r\rangle = \bigotimes_{i=1}^{N} |k_i\rangle, \qquad k_i \in \{0, 1\}$$

4 References

Wikipedia - SWAP test
Quantum Fingerprinting
Learning the quantum algorithm for state overlap