QOSF Task 4

Sagnik Bhattacharya

September 19, 2021

1 Task

Prepare 4 random 4-qubit quantum states of your choice. Create and train a variational circuit that transforms input states into predefined output states. Namely

- if random state 1 is provided, it returns state $|0011\rangle$
- if random state 2 is provided, it returns state $|0101\rangle$
- if random state 3 is provided, it returns state $|1010\rangle$
- if random state 4 is provided, it returns state $|1100\rangle$

What would happen if you provided a different state? Analyze and discuss the results.

2 Solution

2.1 Random state preparation

We create the state vector for a random n-qubit state $|\psi\rangle$ by generating 2^n random numbers between 0 and 1 using `np.random.rand()` and then dividing by the square root of the sum of the numbers generated to get a normalized state vector (that is, such that $\langle\psi|\psi\rangle = 1$). This is fed into the circuit by using `qml.QubitStateVector` after tensoring with the state $|0\rangle$ for an ancilla qubit.

2.2 Variational quantum circuit architecture

Each layer of the variation quantum circuit consists of the following gates: three single qubit rotations $R_X(\theta)$ and $R_Z(\phi)$ on each wire, followed by IsingXX(ζ) coupling gates between wires i and $i+1$, where i ranges over the number of wires, with the last wire connected to the first one. Since we are using one ancilla qubit, the system has five wires and so there are fifteen parameters for each layer. We use five layers.

2.3 Cost function

The cost function is given by the ℓ_2-norm of the ℓ_1-distance between the target states and the expectation values of measuring the four qubits in the computational basis.

2.4 Optimization

We use the Adam optimizer in PyTorch for optimization. We start with a learning rate of 1 and decay the learning rate by $\gamma = 0.95$ after every optimization step.

3 Results obtained

For each of the initial states, the target states and the actually obtained outputs are given below.

```
[0, 0, 1, 1]
[0.06412698920493662, -0.0005711535994421262, 0.646179517681348, 0.6588933428089019]
[0, 1, 0, 1]
[0.013939253786773293, 0.5585991566978238, 0.009863141048212603, 0.6579721926683691]
[1, 0, 1, 0]
[0.5513982122590539, 0.006552224289317921, 0.6273626029571284, 0.01166190918869095]
[1, 1, 0, 0]
[0.7354939131142594, 0.7246896045519693, -0.0006598254170192934, 0.014436462942792622]
```

Clearly the obtained value is close to zero when the target value is zero, but the error is large when the target value is one. This continues to be the case when using IsingZZ coupling gates instead of IsingXX ones.

```
[0, 0, 1, 1]
[0.041273330146426875, 0.00405723672037404, 0.644412850523908, 0.5523505730183145]
[0, 1, 0, 1]
[0.017369737149882603, 0.5805192273842145, 0.0011082420043118844, 0.5570452314365447]
[1, 0, 1, 0]
[0.4120025065496564, -0.0030229787276920914, 0.7110613332434057, 0.005602383597934079]
[1, 1, 0, 0]
[0.8124113113989183, 0.6731637778238683, 0.005326141865556461, 0.15757592399570175]
```

We do not have a good explanation for this an this merits further investigation.

4 Discussion

4.1 Impact of different learning architectures

We tested several different architectures with different gates in each layer, different number of ancilla qubits and different number of layers. We discuss the impact of these choices in this section.

4.1.1 Layer architecture

We used the following layer architectures.

1. R_X and R_Z on each wire, followed by CNOT gates between wires i and $i+1$;

2. Ising XX coupling gates between wires i and $i+1$;

3. R_X and R_Z on each wire, followed by Ising XX coupling gates between wires i and $i+1$;

4. R_Z, R_X, R_Z on each wire, followed by Ising XX coupling gates between wires i and $i+1$ (suggested by the one used in [1], which used two-qubit rotations instead of Ising coupling gates).

After 100 iterations with five layers, the costs of the above layer architectures are as follows.

Layer Architecture	Cost
R_X, R_Y, CNOT	1.7024
IsingXX coupling gate	3.7423
R_X, R_Z, IsingXX	1.5016
R_Z, R_X, R_Z, IsingXX	1.5470

Clearly the third choice performs the best, and this is the one we choose for the layer architecture.

4.2 Cost function

We tried a few different cost functions - ℓ_1-norm of the costs from each of the four states, ℓ_2-norm of the individual costs, and the product of the four costs. The two cost functions with norms performed approximately the same, but the ℓ_2-norm converged the fastest and that is what we finally used. We also tried the ℓ_∞-norm, but the convergence of that was very slow.

4.3 Number of ancilla qubits

Using one ancilla qubit outperforms using none, and while using more ancilla qubits does lead to lower eventual cost, the training time is also higher and convergence is slower, so we use just one ancilla qubit.

4.4 Choice of learning rate

Using a flat learning rate of 1 in the Adam optimizer in PyTorch leads to a fast initial improvement in the cost but slower convergence. Using a smaller learning rate leads to substantially slower convergence overall because the initial improvement is very slow. We therefore decay the learning rate by a parameter γ after every iteration using `torch.optim.lr_scheduler.ExponentialLR`. Both $\gamma = 0.9$ and $\gamma = 0.99$ need more iterations before convergence.

4.5 Number of layers

Increasing the number of layers does lead to a lower cost, but also substantially increases the training time, both because each training step takes longer and because it needs more steps before the cost stabilizes. We found five layers to perform well in practice - using ten layers only reduced the cost by around 0.3 while requiring nearly four times the training time.

4.6 Number of steps

With five layers and the above layer architecture, the cost stabilizes after around 100 iterations. Even though more iterations does lead to a slightly more optimal solution (in this case, we obtain a cost of 1.5283 after 200 iterations), the increase is not enough to justify the longer training time.

4.7 Providing a different starting state

The space of 4-qubit quantum states has sixteen dimensions, and we have trained the circuit to transform four random states in that space to four target states. It is highly probable that (a) the four chosen initial states are linearly independent and therefore span a subspace of dimension four and (b) a randomly chosen fifth quantum state will lie outside this linear space. If the fifth state does lie in the space spanned by the four initial states, by linearity they will be taken to a linear combination of the target states (at least just before measurement of the non-ancilla qubits). For states outside of that subspace, we have no control.

References

[1] M. Benedetti, D. Garcia-Pintos, O. Perdomo, V. Leyton-Ortega, Y. Nam, and A. Perdomo-Ortiz, "A generative modeling approach for benchmarking and training shallow quantum circuits," *npj Quantum Information*, vol. 5, no. 1, May 2019.