Variational Quantum Circuits by Examples

Mahn-Soo Choi (Korea University)
choims@korea.ac.kr

Variational Principle

For any quantum state $|\psi\rangle \in \mathcal{V}$,

$$\langle\psi| H |\psi\rangle \geq E_g ,$$

where E_g is the ground state of Hamiltonian H.

Why does it matter?

Suppose that we have a family of quantum states $\left\{ |\psi(\theta_1, \theta_2, \ldots)\rangle : \theta_1, \theta_2, \ldots \in \mathbb{R} \right\}$, and ask which is the closest to the true ground state.

Find the parameter values $\theta_1^*, \theta_2^*, \ldots$ that minimizes

$$\langle\psi(\theta_1, \theta_2, \ldots)| H |\psi(\theta_1, \theta_2, \ldots)\rangle .$$

Then, $|\psi(\theta_1^*, \theta_2^*, \ldots)\rangle$ is the desired state.

In[•]:=
```
Let[Qubit, S]
Let[Real, x, y]
```

Example

Consider a one-parameter family of quantum states of the following form.

In[•]:=
```
try = Ket[S → 0] * Cos[Pi * x / 2] + Ket[S → 1] * Sin[Pi * x / 2]
```

Out[•]=

$$\text{Cos}\left[\frac{\pi\, x}{2}\right] |0_S\rangle + |1_S\rangle \, \text{Sin}\left[\frac{\pi\, x}{2}\right]$$

Calculate the expectation value of the Hamiltonian with respect to the trial wave function.

In[•]:=
```
avg = Dagger[try] ** S[1] ** try // ExpToTrig // Simplify
```
Out[•]=

$$\text{Sin}[\pi\, x]$$

Minimize the expectation value over the parameters.

In[•]:=
```
{val, opt} = Minimize[{avg, 0 ≤ x ≤ 1, 0 ≤ y ≤ 2}, {x, y}]
```
Out[•]=

$$\{0, \{x \to 0, y \to 0\}\}$$

This is the resulting state.

In[•]:=
```
gv = try /. opt
```
Out[•]=

$$\left|0_S\right\rangle$$

Compare it with the true ground state.

In[•]:=
```
ebs = KetNormalize /@ ProperStates[S[1]]
gs = First[ebs]
```
Out[•]=

$$\left\{-\frac{\left|0_S\right\rangle}{\sqrt{2}}+\frac{\left|1_S\right\rangle}{\sqrt{2}}, \frac{\left|0_S\right\rangle}{\sqrt{2}}+\frac{\left|1_S\right\rangle}{\sqrt{2}}\right\}$$

Out[•]=

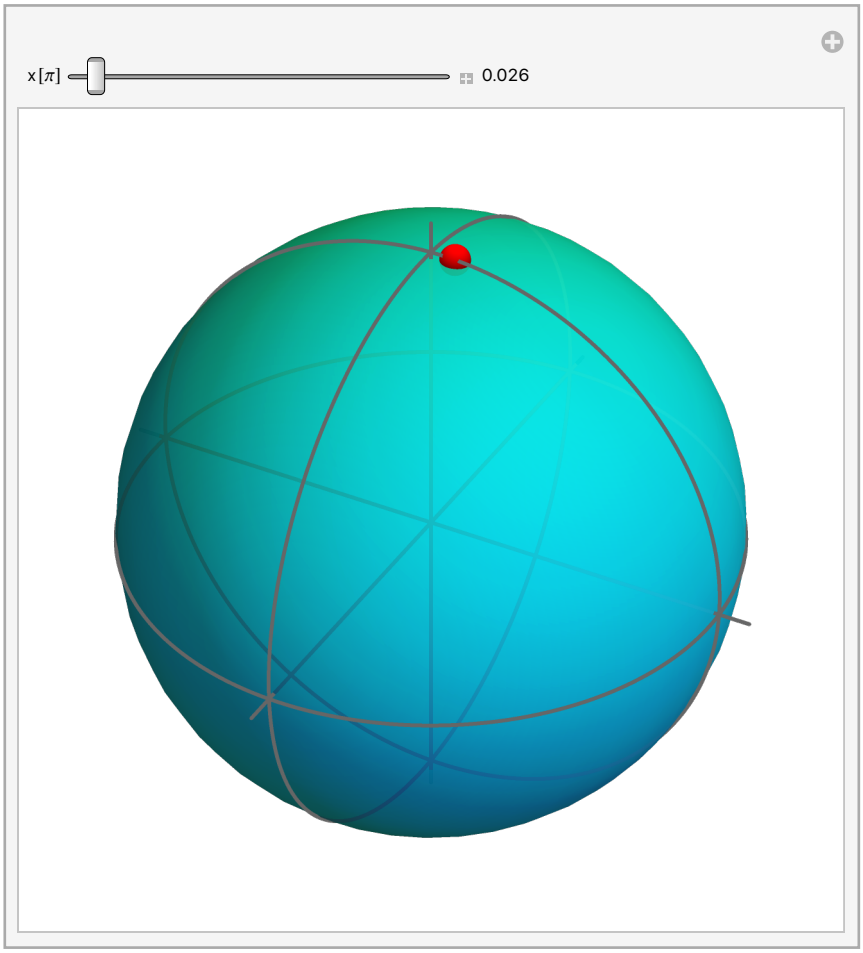
$$-\frac{\left|0_S\right\rangle}{\sqrt{2}}+\frac{\left|1_S\right\rangle}{\sqrt{2}}$$

In[•]:=
```
Fidelity[gv, ebs[[1]]]
```
Out[•]=

$$\frac{1}{\sqrt{2}}$$

In[•]:=
```
Manipulate[
 Evaluate@BlochSphere[{Red, Bead@BlochVector[try]}],
 {{x, 0, "x[π]"}, 0, 1, Appearance → "Labeled"}]
```

Out[•]=



Example

Consider a slightly different one-parameter family of quantum states of the following form.

In[•]:=
```
try = Ket[S → 0] * Cos[Pi * x / 2] - Ket[S → 1] * Sin[Pi * x / 2]
```

Out[•]=

$$\text{Cos}\left[\frac{\pi\, x}{2}\right] \left|0_S\right\rangle - \left|1_S\right\rangle \text{Sin}\left[\frac{\pi\, x}{2}\right]$$

Calculate the expectation value of the Hamiltonian with respect to the trial wave function.

In[•]:=
```
avg = Dagger[try] ** S[1] ** try // ExpToTrig // Simplify
```

Out[•]=

$$-\text{Sin}[\pi\, x]$$

Minimize the expectation value over the parameters.

In[•]:=
```
{val, opt} = Minimize[{avg, 0 ≤ x ≤ 1, 0 ≤ y ≤ 2}, {x, y}]
```

Out[•]=

$$\left\{-1, \left\{x \to \frac{1}{2}, y \to 0\right\}\right\}$$

This is the resulting state.

In[•]:=
```
gv = try /. opt
```

Out[•]=

$$\frac{\left|0_S\right\rangle}{\sqrt{2}} - \frac{\left|1_S\right\rangle}{\sqrt{2}}$$

Compare it with the true ground state.

In[•]:=
```
ebs = KetNormalize /@ ProperStates[S[1]]
gs = First[ebs]
```

Out[•]=

$$\left\{-\frac{\left|0_S\right\rangle}{\sqrt{2}} + \frac{\left|1_S\right\rangle}{\sqrt{2}}, \frac{\left|0_S\right\rangle}{\sqrt{2}} + \frac{\left|1_S\right\rangle}{\sqrt{2}}\right\}$$

Out[•]=

$$-\frac{\left|0_S\right\rangle}{\sqrt{2}} + \frac{\left|1_S\right\rangle}{\sqrt{2}}$$

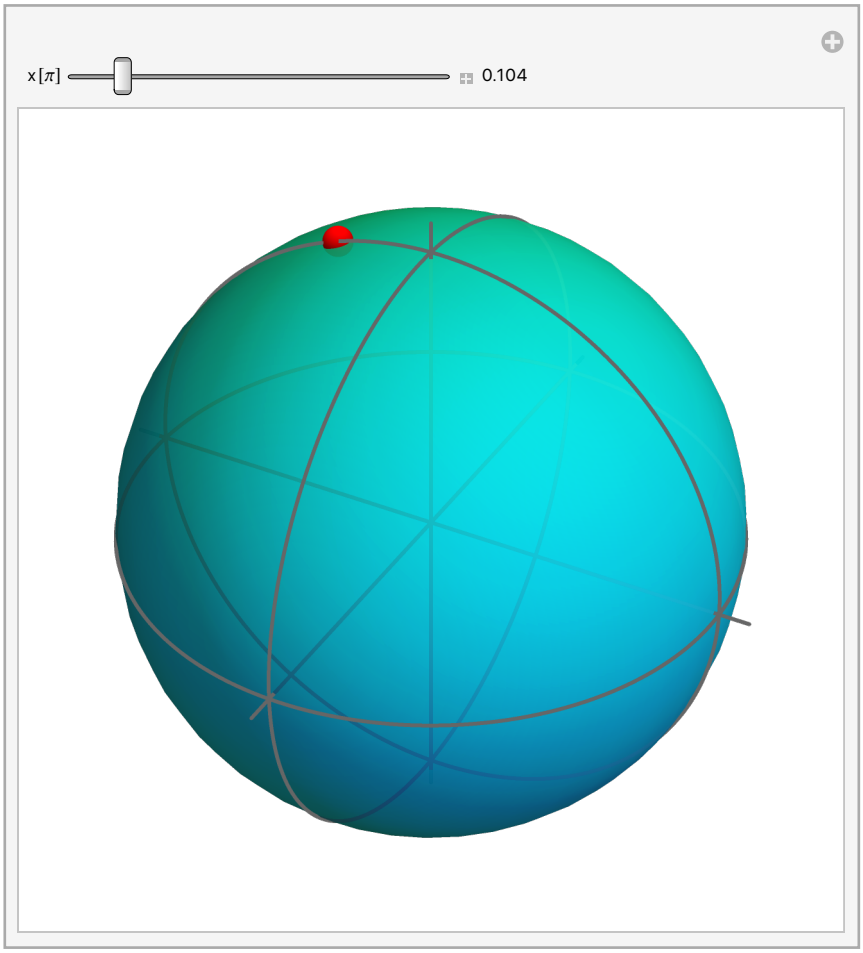
In[•]:=
```
Fidelity[gv, ebs[[1]]]
```

Out[•]=

1

In[ ]:=
```
Manipulate[
 Evaluate@BlochSphere[{Red, Bead@BlochVector[try]}],
 {{x, 0, "x[π]"}, 0, 1, Appearance → "Labeled"}]
```

Out[ ]=



Example

Now, let us consider a two-parameter family of quantum states of the following form.

In[ ]:=
```
try = Ket[S → 0] * Cos[Pi * x / 2] + Ket[S → 1] * Sin[Pi * x / 2] * Exp[I * Pi * y]
```

Out[ ]=

$$\text{Cos}\left[\frac{\pi\, x}{2}\right] \left|0_S\right\rangle + e^{i\,\pi\, y} \left|1_S\right\rangle \text{Sin}\left[\frac{\pi\, x}{2}\right]$$

In[ ]:=
```
avg = Dagger[try] ** S[1] ** try // ExpToTrig // Simplify
```

Out[ ]=

$$\text{Cos}[\pi\, y]\ \text{Sin}[\pi\, x]$$

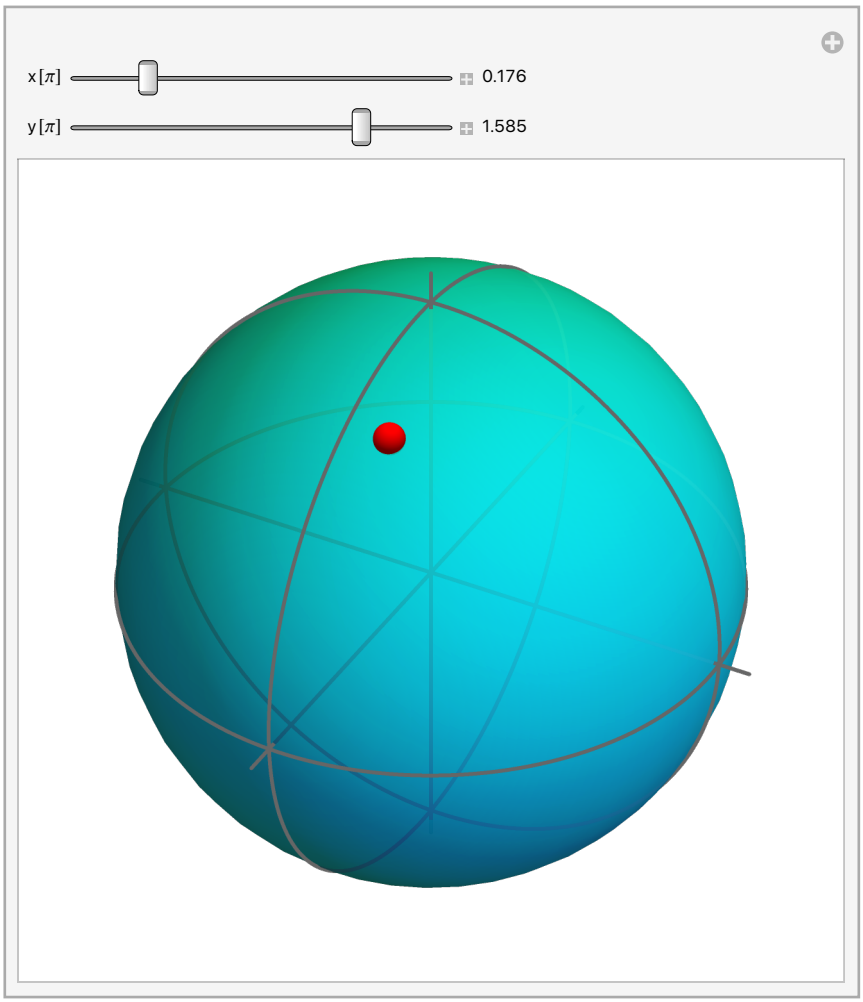
In[ ]:=
```
Minimize[{avg, 0 ≤ x ≤ 1, 0 ≤ y ≤ 2}, {x, y}]
```

Out[ ]=

$$\left\{-1, \left\{x \to \frac{1}{2}, y \to 1\right\}\right\}$$

In[ ]:=
```
Manipulate[
 Evaluate@BlochSphere[{Red, Bead@BlochVector[try]}],
 {{x, 0, "x[π]"}, 0, 1, Appearance → "Labeled"},
 {{y, 0, "y[π]"}, 0, 2, Appearance → "Labeled"}
]
```

Out[ ]=



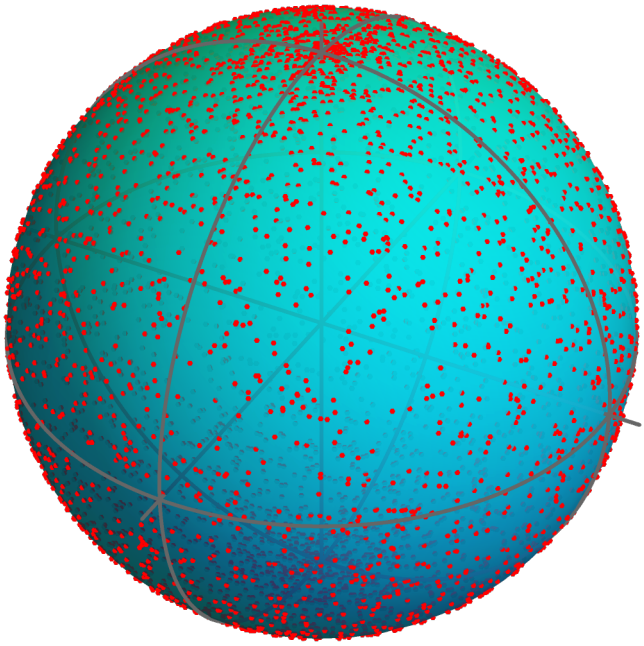
In[ ]:=
```
$N = 5000;
xy = Transpose@{RandomReal[{0, 1}, $N], RandomReal[{0, 2}, $N]};
TableForm[xy[[ ;; UpTo[5]]], TableHeadings → {None, {x, y}}]
```

Out[ ]//TableForm=

x	y
0.948785	0.086066
0.930615	1.95568
0.553553	1.38668
0.533337	1.13757
0.317418	1.80797

In[•]:=
```
bv = Map[Chop@BlochVector[try /. Thread[{x, y} → #]] &, xy];
BlochSphere[{Red, Point /@ bv}]
```

Out[•]=



Question

For an arbitrary single-qubit Hamiltonian *H*, would the above trial state be enough?

In[•]:=
```
try = Ket[S → 0] * Cos[Pi * x / 2] + Ket[S → 1] * Sin[Pi * x / 2] * Exp[I * Pi * y]
```

Out[•]=

$$\text{Cos}\left[\frac{\pi\, x}{2}\right] \left|0_S\right\rangle + e^{i\,\pi\, y} \left|1_S\right\rangle \text{Sin}\left[\frac{\pi\, x}{2}\right]$$

For example, the following state has two parameters.

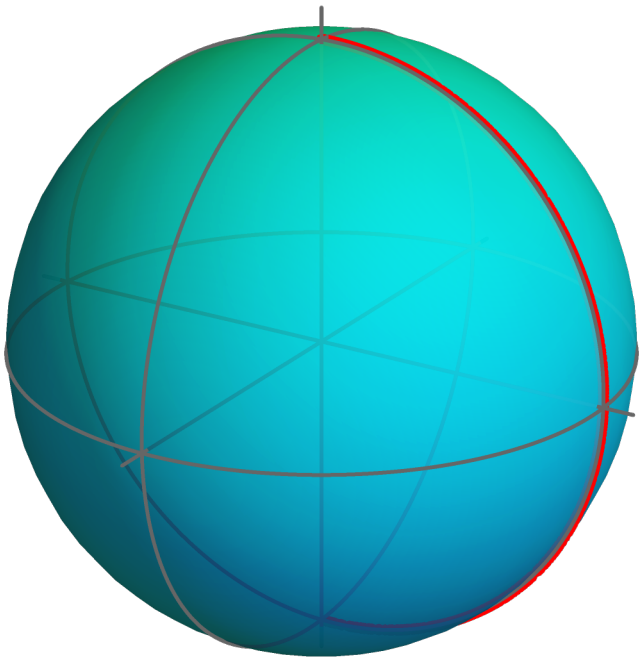
In[•]:=
```
new = Ket[S → 0] * Cos[Pi * x / 2] + Ket[S → 1] * Sin[Pi * x / 2] * Cos[Pi * y / 2]
```

Out[•]=

$$\text{Cos}\left[\frac{\pi\, x}{2}\right] \left|0_S\right\rangle + \text{Cos}\left[\frac{\pi\, y}{2}\right] \left|1_S\right\rangle \text{Sin}\left[\frac{\pi\, x}{2}\right]$$

In[ ]:=
```
$N = 1000;
xy = Transpose@{RandomReal[{0, 1}, $N], RandomReal[{0, 1}, $N]};
bv = Map[Chop@BlochVector@KetNormalize[new /. Thread[{x, y} → #]] &, xy];
BlochSphere[{Red, PointSize[0.0075], Point /@ bv}]
```

Out[ ]=



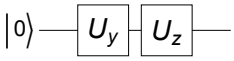
Variational Quantum Circuits for Single Qubits

Example

In[ ]:=
```
Let[Qubit, S]
Let[Real, x, y]
```

In[ ]:=
```
qc = QuantumCircuit[Ket[S@{1}],
  Rotation[Pi * x, S[1, 2]],
  Rotation[Pi * y, S[1, 3]]]
```

Out[ ]=



In[ ]:=
```
out = Elaborate[qc]
```

Out[ ]=

$$\text{Cos}\left[\frac{\pi\, x}{2}\right] \left|0_{S_1}\right\rangle \left(\text{Cos}\left[\frac{\pi\, y}{2}\right] - i\, \text{Sin}\left[\frac{\pi\, y}{2}\right]\right) + \left|1_{S_1}\right\rangle \text{Sin}\left[\frac{\pi\, x}{2}\right] \left(\text{Cos}\left[\frac{\pi\, y}{2}\right] + i\, \text{Sin}\left[\frac{\pi\, y}{2}\right]\right)$$

In[ ]:=
```
avg = Dagger[qc] ** S[1, 1] ** qc // Simplify
```

Out[ ]=

$$\text{Cos}[\pi\, y]\ \text{Sin}[\pi\, x]$$

```
In[•]:= Minimize[{avg, 0 ≤ x ≤ 1, 0 ≤ y ≤ 2}, {x, y}] // EchoTiming
```

1.12359

Out[•]=

$$\left\{-1, \left\{x \to \frac{1}{2}, y \to 1\right\}\right\}$$

Variational Quantum Circuits for Two Qubits

Issue: A two-qubit quantum state requires $2 \times 2^2 - 2 = 6$ parameters to fully specify. How can we actually parameterize two-qubit states?

Quantum Circuit Layers

```
In[•]:= Let[Qubit, S]
```

```
In[•]:= $n = 2;
        kk = Range[$n];
        SS = S[kk, $]
```

Out[•]=

$\{S_1, S_2\}$

The variational parameters are denoted by $a\,[\,layer\,,\ qubit\,,\ \mu\,]$.

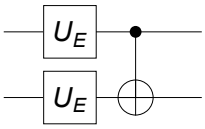
```
In[•]:= Let[Real, a]
```

Construct basic layers of quantum circuit.

```
In[•]:= Clear[layer]
        layer[k_] := QuantumCircuit[
          {EulerRotation[Pi * a[k, 1, {1, 2, 3}], S[1]],
           EulerRotation[Pi * a[k, 2, {1, 2, 3}], S[2]]},
          CNOT[S[1], S[2]]
         ]
```

```
In[•]:= layer[1]
```

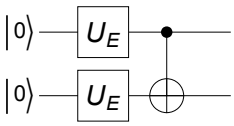
Out[•]=



Example

```
In[•]:= try = QuantumCircuit[Ket[SS], layer[1]]
```

Out[•]=



In[•]:=
```
Elaborate[try]
```

Out[•]=

$$\left|1_{S_1}0_{S_2}\right\rangle \text{Sin}\left[\frac{1}{2}\pi a_{1,1,2}\right] \text{Sin}\left[\frac{1}{2}\pi a_{1,2,2}\right]$$
$$\left(\text{Cos}\left[\frac{1}{2}\pi\left(a_{1,1,1}-a_{1,1,3}+a_{1,2,1}-a_{1,2,3}\right)\right]+\text{i}\,\text{Sin}\left[\frac{1}{2}\pi\left(a_{1,1,1}-a_{1,1,3}+a_{1,2,1}-a_{1,2,3}\right)\right]\right)+$$
$$\text{Cos}\left[\frac{1}{2}\pi a_{1,1,2}\right]\left|0_{S_1}1_{S_2}\right\rangle \text{Sin}\left[\frac{1}{2}\pi a_{1,2,2}\right]$$
$$\left(\text{Cos}\left[\frac{1}{2}\pi\left(a_{1,1,1}+a_{1,1,3}-a_{1,2,1}+a_{1,2,3}\right)\right]-\text{i}\,\text{Sin}\left[\frac{1}{2}\pi\left(a_{1,1,1}+a_{1,1,3}-a_{1,2,1}+a_{1,2,3}\right)\right]\right)+$$
$$\text{Cos}\left[\frac{1}{2}\pi a_{1,2,2}\right]\left|1_{S_1}1_{S_2}\right\rangle \text{Sin}\left[\frac{1}{2}\pi a_{1,1,2}\right]\left(\text{i}\,\text{Cos}\left[\frac{1}{2}\pi\left(1-a_{1,1,1}+a_{1,1,3}+a_{1,2,1}+a_{1,2,3}\right)\right]+\right.$$
$$\left.\text{Sin}\left[\frac{1}{2}\pi\left(1-a_{1,1,1}+a_{1,1,3}+a_{1,2,1}+a_{1,2,3}\right)\right]\right)+\text{Cos}\left[\frac{1}{2}\pi a_{1,1,2}\right]\text{Cos}\left[\frac{1}{2}\pi a_{1,2,2}\right]\left|0_{S_1}0_{S_2}\right\rangle$$
$$\left(\text{Cos}\left[\frac{1}{2}\pi\left(a_{1,1,1}+a_{1,1,3}+a_{1,2,1}+a_{1,2,3}\right)\right]-\text{i}\,\text{Sin}\left[\frac{1}{2}\pi\left(a_{1,1,1}+a_{1,1,3}+a_{1,2,1}+a_{1,2,3}\right)\right]\right)$$

In[•]:=
```
HH = S[1, 1] + S[2, 1] + 2 * Total[S[1, All] ** S[2, All]]
```

Out[•]=

$$2\left(S_1^X S_2^X + S_1^Y S_2^Y + S_1^Z S_2^Z\right) + S_1^X + S_2^X$$

In[•]:=
```
EchoTiming[avg = Dagger[try] ** HH ** try // Simplify]
```

1.15869

Out[•]=

$$\text{Cos}[\pi a_{1,2,2}]\,(2-2\,\text{Cos}[\pi a_{1,1,1}]\,\text{Sin}[\pi a_{1,1,2}])+\text{Cos}[\pi a_{1,2,1}]\,\text{Sin}[\pi a_{1,2,2}]+$$
$$\text{Cos}[\pi a_{1,1,1}]\,\text{Sin}[\pi a_{1,1,2}]\,(2+\text{Cos}[\pi a_{1,2,1}]\,\text{Sin}[\pi a_{1,2,2}])$$

In[•]:=
```
{val, opt} = NMinimize[avg, a[1, {1, 2}, {1, 2, 3}]] // Chop
```

Out[•]=

$$\{-6.,\ \{a_{1,1,1}\to -1.,\ a_{1,1,2}\to -5.5,$$
$$a_{1,1,3}\to 0.156333,\ a_{1,2,1}\to 0.633872,\ a_{1,2,2}\to 9.,\ a_{1,2,3}\to 0.206333\}\}$$

The ground-state energy seems to be **-6** (in the given unit of energy). This is in good agreement with the exact diagonalization.

In[•]:=
```
ProperValues[HH]
```

Out[•]=

$$\{-6,\ 4,\ 2,\ 0\}$$

Let us compare the result for the ground-state wave function to the exact wave function as well.

In[•]:=
```
{val, vec} = ProperSystem[HH]
```

Out[•]=

$$\left\{\{-6,\ 4,\ 2,\ 0\},\ \left\{-\left|0_{S_1}1_{S_2}\right\rangle+\left|1_{S_1}0_{S_2}\right\rangle,\ \left|0_{S_1}0_{S_2}\right\rangle+\left|0_{S_1}1_{S_2}\right\rangle+\left|1_{S_1}0_{S_2}\right\rangle+\left|1_{S_1}1_{S_2}\right\rangle,\right.\right.$$
$$\left.\left.-\left|0_{S_1}0_{S_2}\right\rangle+\left|1_{S_1}1_{S_2}\right\rangle,\ \left|0_{S_1}0_{S_2}\right\rangle-\left|0_{S_1}1_{S_2}\right\rangle-\left|1_{S_1}0_{S_2}\right\rangle+\left|1_{S_1}1_{S_2}\right\rangle\right\}\right\}$$

In[•]:=
```
gv = Elaborate[try /. opt] // Chop
```

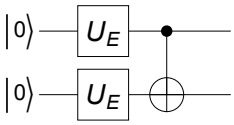
Out[•]=

$$(0.292204-0.643908\,\text{i})\left|0_{S_1}1_{S_2}\right\rangle-(0.292204-0.643908\,\text{i})\left|1_{S_1}0_{S_2}\right\rangle$$

```
In[•]:= Fidelity[gv, KetNormalize@vec[[1]]]
Out[•]=
1.
```

Example

```
In[•]:= try = QuantumCircuit[Ket[SS], layer[1]]
Out[•]=
```



Let us take a Hamiltonian of the form with the ground state chosen randomly. The ground-state energy is manifestly –1.

```
In[•]:= gs = ExpressionFor[Normalize@RandomVector[2^2], S@{1, 2}];
        HH = -Dyad[gs, gs, S@{1, 2}] // Elaborate // Chop
Out[•]=
-0.25 + 0.038322 S_1^X S_2^X + 0.128586 S_1^X S_2^Y - 0.183506 S_1^X S_2^Z + 0.196121 S_1^Y S_2^X - 0.0583226 S_1^Y S_2^Y -
 0.0632708 S_1^Y S_2^Z - 0.09357 S_1^Z S_2^X - 0.145073 S_1^Z S_2^Y - 0.0934351 S_1^Z S_2^Z + 0.122019 S_1^X +
 0.0952867 S_1^Y - 0.0417506 S_1^Z - 0.109081 S_2^X - 0.0647581 S_2^Y + 0.0980764 S_2^Z
```

```
In[•]:= EchoTiming[avg = Dagger[try] ** HH ** try // Simplify // Re;]
3.63586
```

Specify the constraints.

```
In[•]:= constraints = Thread[0 ≤ a[1, {1, 2}, {1, 2, 3}] < 2 Pi]
Out[•]=
{0 ≤ a_{1,1,1} < 2 π, 0 ≤ a_{1,1,2} < 2 π, 0 ≤ a_{1,1,3} < 2 π,
 0 ≤ a_{1,2,1} < 2 π, 0 ≤ a_{1,2,2} < 2 π, 0 ≤ a_{1,2,3} < 2 π}
```

Minimize the expectation value with respect to the parameters.

```
In[•]:= EchoTiming[
          {val, opt} = NMinimize[{avg, constraints}, a[1, {1, 2}, {1, 2, 3}]] // Chop]
0.078524
Out[•]=
{-0.90829, {a_{1,1,1} → 4.43858, a_{1,1,2} → 1.56556,
   a_{1,1,3} → 2.86602, a_{1,2,1} → 1.29478, a_{1,2,2} → 3.65132, a_{1,2,3} → 2.85118}}
```

Let us compare the result for the ground-state wave function to the exact wave function as well.

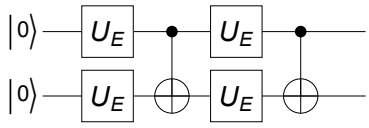
```
In[•]:= gv = Elaborate[try /. opt] // Chop
Out[•]=
(-0.430721 - 0.50342 i) |0_{S_1} 0_{S_2}⟩ + (0.0875366 - 0.394516 i) |0_{S_1} 1_{S_2}⟩ -
 (0.328299 + 0.00833342 i) |1_{S_1} 0_{S_2}⟩ - (0.33439 - 0.42199 i) |1_{S_1} 1_{S_2}⟩
```

```
In[•]:= Fidelity[gv, gs]
Out[•]=
0.953042
```

Example

```
In[ ]:= try = QuantumCircuit[Ket[SS], layer[1], layer[2]]
```

Out[ ]=



Let us take a Hamiltonian of the form with the ground state chosen randomly. The ground-state energy is manifestly –1.

```
In[ ]:= gs = ExpressionFor[Normalize@RandomVector[2^2], S@{1, 2}];
        HH = -Dyad[gs, gs, S@{1, 2}] // Elaborate // Chop
        mat = Matrix[HH, SS];
```

Out[ ]=

$$-0.25 - 0.228035\, S_1^X S_2^X + 0.00476709\, S_1^X S_2^Y + 0.0768043\, S_1^X S_2^Z + 0.0148305\, S_1^Y S_2^X - 0.0392316\, S_1^Y S_2^Y + 0.0261808\, S_1^Y S_2^Z + 0.0760681\, S_1^Z S_2^X + 0.0290551\, S_1^Z S_2^Y + 0.0174567\, S_1^Z S_2^Z - 0.235629\, S_1^X + 0.00661256\, S_1^Y + 0.0673388\, S_1^Z - 0.235809\, S_2^X - 0.00229539\, S_2^Y + 0.0669947\, S_2^Z$$

```
In[ ]:= vec = Matrix[try];
```

```
In[ ]:= EchoTiming[avg = Re[Conjugate[vec].mat.vec // N];]
```

0.026748

Specify the constraints.

```
In[ ]:= constraints = Thread[0 ≤ a[{1, 2}, {1, 2}, {1, 2, 3}] < 2 Pi]
```

Out[ ]=

$$\{0 \le a_{1,1,1} < 2\pi,\ 0 \le a_{1,1,2} < 2\pi,\ 0 \le a_{1,1,3} < 2\pi,\ 0 \le a_{1,2,1} < 2\pi,\ 0 \le a_{1,2,2} < 2\pi,\ 0 \le a_{1,2,3} < 2\pi,\ 0 \le a_{2,1,1} < 2\pi,\ 0 \le a_{2,1,2} < 2\pi,\ 0 \le a_{2,1,3} < 2\pi,\ 0 \le a_{2,2,1} < 2\pi,\ 0 \le a_{2,2,2} < 2\pi,\ 0 \le a_{2,2,3} < 2\pi\}$$

```
In[ ]:= EchoTiming[
          {val, opt} = NMinimize[{avg, constraints}, a[{1, 2}, {1, 2}, {1, 2, 3}]] // Chop]
```

0.604115

Out[ ]=

$$\{-1.,\ \{a_{1,1,1} \to 4.68537,\ a_{1,1,2} \to 2.9018,\ a_{1,1,3} \to 3.6085,\ a_{1,2,1} \to 4.51992,\ a_{1,2,2} \to 3.66463,\ a_{1,2,3} \to 2.28461,\ a_{2,1,1} \to 1.9761,\ a_{2,1,2} \to 1.59579,\ a_{2,1,3} \to 3.50414,\ a_{2,2,1} \to 3.2667,\ a_{2,2,2} \to 4.63122,\ a_{2,2,3} \to 1.91024\}\}$$

```
In[ ]:= gv = Elaborate[try /. opt] // Chop
```

Out[ ]=

$$(0.139994 + 0.280378\, i)\ |0_{S_1} 0_{S_2}\rangle + (0.304098 + 0.417896\, i)\ |0_{S_1} 1_{S_2}\rangle + (0.320018 + 0.40668\, i)\ |1_{S_1} 0_{S_2}\rangle + (0.32508 + 0.511077\, i)\ |1_{S_1} 1_{S_2}\rangle$$

```
In[ ]:= Fidelity[gv, gs]
```

Out[ ]=

```
1.
```

Questions

1. How can you visualize a two-qubit states (if at all)?
2. What is the role of the CNOT gate?
3. What is the minimal universal two-qubit quantum circuit?

- **See also**: V. V. Shende, I. L. Markov, and Stephen S. Bullock, Physical Review A 69, 062321 (2004).

Variational Classifier with 4 Qubits

- See Maria Schuld, PennyLane demonstration “Variational classifier”.

The Problem

We want to predict the parity of input data using variational quantum circuits.

```
In[•]:= bits = Tuples[{0, 1}, 4];
        parity = 2 * Mod[Count[#, 1] & /@ bits, 2] - 1;
        data = AssociationThread[bits → parity];
        Normal[data] // TableForm
```

```
Out[•]//TableForm=
{0, 0, 0, 0} → -1
{0, 0, 0, 1} → 1
{0, 0, 1, 0} → 1
{0, 0, 1, 1} → -1
{0, 1, 0, 0} → 1
{0, 1, 0, 1} → -1
{0, 1, 1, 0} → -1
{0, 1, 1, 1} → 1
{1, 0, 0, 0} → 1
{1, 0, 0, 1} → -1
{1, 0, 1, 0} → -1
{1, 0, 1, 1} → 1
{1, 1, 0, 0} → -1
{1, 1, 0, 1} → 1
{1, 1, 1, 0} → 1
{1, 1, 1, 1} → -1
```

Quantum circuit layers

The qubits to be modeled are `S[i,$]` for $i = 1, \ldots, n$.

```
In[•]:= Let[Qubit, S]
```

```
In[•]:= $n = 4;
        kk = Range[$n];
        SS = S[kk, $]
```

```
Out[•]=
{S1, S2, S3, S4}
```

The parameters to be optimized are `a[`*layer*, *i*, *j*`]` for $i = 1, \ldots, n$ and $j = 1, \ldots, 3$.

```
In[ ]:= Let[Real, a]
```

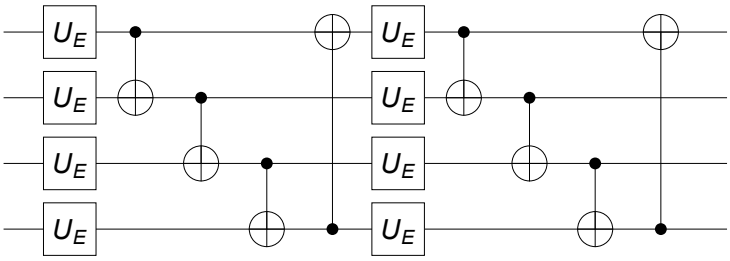
Define a quantum circuit layer.

```
In[ ]:= Clear[layer]
layer[aa_?MatrixQ, ss:{__?QubitQ}] := QuantumCircuit[
   Table[EulerRotation[aa[[k]], ss[[k]]], {k, kk}],
   Apply[Sequence, CNOT @@@ Transpose@{ss, RotateLeft@ss}]
  ] /; Length[aa] == Length[ss]
layer[aa_?MatrixQ] := layer[aa, SS]
layer[n_Integer] := QuantumCircuit @@ Table[layer[Array[a[k], {$n, 3}]], {k, n}]
```

For example, the following quantum circuit contains 24 real parameters, with three parameters $a[k, i, 1]$, $a[k, i, 2]$, $a[k, i, 3]$ associated with each qubit i on layer k.

```
In[ ]:= ll = layer[2]
```

```
Out[ ]=
```



Initialization

This is the simplest method to embed input data to quantum states.

```
In[ ]:= BasisEmbedding[vv:{__?BinaryQ}, ss:{__?QubitQ}] :=
  Ket[ss → PadRight[vv, Length@ss]]
BasisEmbedding[ss:{__?QubitQ}][vv:{__?BinaryQ}] := BasisEmbedding[vv, ss]
```

Embed the input data to quantum states.

```
In[ ]:= in = BasisEmbedding[SS] /@ Keys[data]
in = Matrix /@ in;
```

```
Out[ ]=
```

$$\left\{\left|0_{S_1}0_{S_2}0_{S_3}0_{S_4}\right\rangle,\ \left|0_{S_1}0_{S_2}0_{S_3}1_{S_4}\right\rangle,\ \left|0_{S_1}0_{S_2}1_{S_3}0_{S_4}\right\rangle,\ \left|0_{S_1}0_{S_2}1_{S_3}1_{S_4}\right\rangle,\right.$$
$$\left|0_{S_1}1_{S_2}0_{S_3}0_{S_4}\right\rangle,\ \left|0_{S_1}1_{S_2}0_{S_3}1_{S_4}\right\rangle,\ \left|0_{S_1}1_{S_2}1_{S_3}0_{S_4}\right\rangle,\ \left|0_{S_1}1_{S_2}1_{S_3}1_{S_4}\right\rangle,$$
$$\left|1_{S_1}0_{S_2}0_{S_3}0_{S_4}\right\rangle,\ \left|1_{S_1}0_{S_2}0_{S_3}1_{S_4}\right\rangle,\ \left|1_{S_1}0_{S_2}1_{S_3}0_{S_4}\right\rangle,\ \left|1_{S_1}0_{S_2}1_{S_3}1_{S_4}\right\rangle,$$
$$\left.\left|1_{S_1}1_{S_2}0_{S_3}0_{S_4}\right\rangle,\ \left|1_{S_1}1_{S_2}0_{S_3}1_{S_4}\right\rangle,\ \left|1_{S_1}1_{S_2}1_{S_3}0_{S_4}\right\rangle,\ \left|1_{S_1}1_{S_2}1_{S_3}1_{S_4}\right\rangle\right\}$$

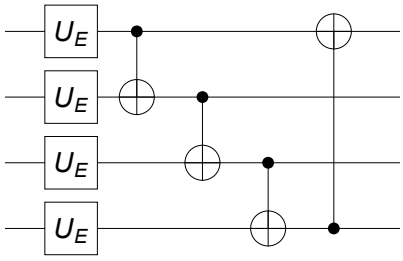
Learning with a single layer

```
In[ ]:= $L = 1;
```

Pick a quantum circuit layer to apply on the input states.

```
In[ ]:= op = layer[$L]
```

Out[ ]=




```
In[ ]:= mat = Normal@Matrix[op];

In[ ]:= Z1 = Matrix[S[1, 3], SS];

In[ ]:= w0 = Flatten@RandomReal[{0, 1} Pi, {$L, $n, 3}]
```

Out[ ]=

```
{1.14186, 0.461403, 2.24853, 2.01597, 1.25509, 0.261809,
 0.081487, 0.34783, 2.09139, 1.00905, 2.61312, 2.22469}
```

The set of parameters.

```
In[ ]:= aa = a[Range@$L, Range@$n, {1, 2, 3}];
```

Prepare the natural constraints for angles.

```
In[ ]:= constraints = Thread[0 ≤ aa < Pi];

In[ ]:= optimizer[ww_] := Module[
  {ii = RandomChoice[Range@Length@data, 4],
   yy, vv, avg, sol, cost},
  yy = Part[Values@data, ii];
  vv = Transpose@Part[in, ii];
  vv = Transpose[mat.vv];
  avg = Map[Conjugate[#].Z1.# &, vv];
  cost = Mean@Abs[avg - yy]^2;
  EchoTiming[{cost, sol} =
    FindMinimum[{cost, constraints}, Transpose@{aa, ww}, Method → "IPOPT"]];
  Echo[{Chop[avg /. sol], yy, cost}, "{avg, yy, cost}"];
  Return[aa /. sol]
 ]
```

Test the optimizer.

```
In[ ]:= optimizer[w0]
```

```
0.083338
```

» {avg, yy, cost}

$$\left\{\{-0.999738, -0.999738, -0.999738, -0.999738\}, \{-1, -1, -1, -1\}, 6.85243 \times 10^{-8}\right\}$$

Out[ ]=

```
{1.5708, 1.5708, 1.5708, 1.5708, 3.12838, 1.5708,
 1.5708, 0.0132097, 1.5708, 1.5708, 3.12838, 1.5708}
```

Find the optimal values of the parameters iteratively.

```
In[•]:= $R = 5;(* the number of iterations *)
        ww = Nest[optimizer, w0, $R]
```

```
0.040423
» {avg, yy, cost} {{-0.999738, -0.999738, 0.999738, 0.999738}, {-1, -1, 1, 1}, 6.85243 × 10^-8}
0.024793
» {avg, yy, cost} {{0.854586, 0.854586, -0.854586, 0.854586}, {-1, -1, -1, 1}, 1.}
0.042006
» {avg, yy, cost} {{-1., -1., -1., -1.}, {-1, -1, -1, 1}, 0.25}
0.0416
» {avg, yy, cost} {{1., 1., 1., -1.}, {1, 1, 1, 1}, 0.25}
0.030926
» {avg, yy, cost} {{-0.999999, 0.999999, 0.999999, 0.999999}, {-1, 1, 1, -1}, 0.250001}
```

```
Out[•]=
{1.57069, 1.57266, 1.57065, 1.57068, 0.000839754, 1.57143,
 1.57106, 0.000839754, 1.57038, 1.57158, 0.000839754, 1.57194}
```

From the optimized parameters, make predictions.

```
In[•]:= mm = mat /. Thread[aa → ww];
        vv = Transpose[mm.Transpose[in]];
        predicted = Map[Sign@Chop[Conjugate[#].Z1.#] &, vv]
```

```
Out[•]=
{1, -1, -1, 1, -1, 1, 1, -1, 1, -1, -1, 1, -1, 1, 1, -1}
```

```
In[•]:= predicted - Values[data]
```

```
Out[•]=
{2, -2, -2, 2, -2, 2, 2, -2, 0, 0, 0, 0, 0, 0, 0, 0}
```

Apparently, a single-layer is not sufficient to predict the parity values properly.

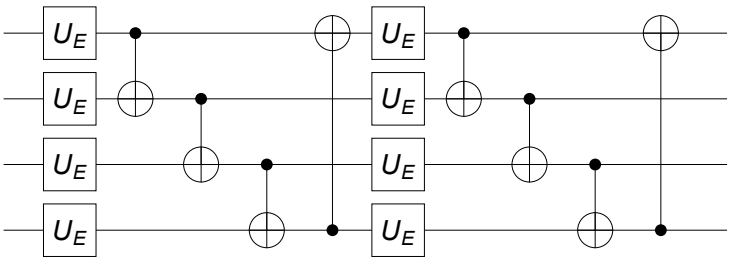
Learning with two layers

```
In[•]:= $L = 2;
```

Pick a quantum circuit layer to apply on the input states.

```
In[•]:= op = layer[$L]
```

```
Out[•]=
```



Check which parameters are actually involved in the above quantum circuit.

```
In[•]:= param = Cases[op, _a, Infinity]
        Length[param]
```

Out[•]=

$$\{a_{1,1,1}, a_{1,1,2}, a_{1,1,3}, a_{1,2,1}, a_{1,2,2}, a_{1,2,3}, a_{1,3,1}, a_{1,3,2}, a_{1,3,3}, a_{1,4,1}, a_{1,4,2}, a_{1,4,3}, a_{2,1,1}, a_{2,1,2}, a_{2,1,3}, a_{2,2,1}, a_{2,2,2}, a_{2,2,3}, a_{2,3,1}, a_{2,3,2}, a_{2,3,3}, a_{2,4,1}, a_{2,4,2}, a_{2,4,3}\}$$

Out[•]=

```
24
```

```
In[•]:= mat = Normal@N@Matrix[op];
        Dimensions[mat]
```

Out[•]=

```
{16, 16}
```

```
In[•]:= Z1 = Matrix[S[1, 3], SS];
        Dimensions[Z1]
```

Out[•]=

```
{16, 16}
```

```
In[•]:= w0 = Flatten@RandomReal[{0, 1} Pi, {$L, $n, 3}]
        Length[w0]
```

Out[•]=

```
{1.74817, 2.94478, 2.42736, 1.3261, 3.08923, 0.965756, 2.10151, 2.87453,
 0.220086, 0.690451, 0.786662, 0.84384, 1.00657, 0.987291, 0.646452, 3.09038,
 1.80856, 2.30169, 0.802831, 1.87877, 1.25505, 1.79362, 2.35276, 2.76367}
```

Out[•]=

```
24
```

The set of parameters.

```
In[•]:= aa = a[Range@$L, Range@$n, {1, 2, 3}];
```

Prepare the natural constraints for angles.

```
In[•]:= constraints = Thread[0 ≤ aa < Pi];
```

```
In[ ]:= Clear[optimizer];
        optimizer[ww_] := Module[
          {ii = RandomChoice[Range@Length@data, 4],
           yy, vv, avg, sol, cost},
          Echo[ii, "ii"];
          yy = Part[Values@data, ii];
          vv = Transpose@SparseArray@Part[in, ii];
          vv = Transpose[mat.vv];
          avg = Map[Conjugate[#].Z1.# &, vv];
          cost = Mean@Abs[avg - yy]^2;
          Echo[{avg, cost} /. Thread[aa → ww] // Chop, "{avg, cost}"];
          EchoTiming[
           {cost, sol} = FindMinimum[Evaluate@{cost, constraints}, Transpose@{aa, ww},
              Method → "IPOPT"];
          ];
          Echo[{Chop[avg /. sol], yy, cost}, "{avg, yy, cost}"];
          Return[aa /. sol]
         ]
```

Test the optimizer.

```
In[ ]:= optimizer[w0]
```

» ii {12, 7, 12, 12}

» {avg, cost} {{0.0544981, 0.0416415, 0.0544981, 0.0544981}, 0.940002}

0.499246

» {avg, yy, cost} $\{\{0.999681, -0.999805, 0.999681, 0.999681\}, \{1, -1, 1, 1\}, 8.30724 \times 10^{-8}\}$

```
Out[ ]=
{1.2749, 1.57083, 2.14843, 1.42713, 3.12727, 1.15761, 1.94307, 3.10658,
 0.71417, 0.937684, 0.0143384, 1.07684, 1.18499, 1.17204, 0.950608, 2.49884,
 1.57065, 1.75628, 1.05008, 1.57096, 1.38531, 1.72406, 1.57081, 1.86664}
```

Find the optimal values of the parameters iteratively.

```
In[ ]:= $R = 3;(* the number of iterations *)
        ww = Nest[optimizer, w0, $R]
```

```
» ii {10, 9, 1, 7}
» {avg, cost} {{-0.10771, -0.0703907, 0.0737407, 0.0416415}, 1.03941}
0.756236
» {avg, yy, cost} {{-0.999663, 0.999663, -0.999484, -0.999843}, {-1, 1, -1, -1}, 1.13349 × 10^-7}
» ii {12, 10, 11, 5}
» {avg, cost} {{0.999843, -0.999663, -0.999843, 0.999663}, 6.10621 × 10^-8}
0.516662
» {avg, yy, cost} {{0.999836, -0.999507, -0.999836, 0.999507}, {1, -1, -1, 1}, 1.07983 × 10^-7}
» ii {10, 7, 13, 5}
» {avg, cost} {{-0.999507, -0.999835, -0.450112, 0.999507}, 0.0189777}
0.554668
» {avg, yy, cost} {{-0.999662, -0.999842, -0.999482, 0.999662}, {-1, -1, -1, 1}, 1.14235 × 10^-7}
```

Out[ ]=

```
{1.5708, 3.12997, 1.5708, 1.59273, 3.1321, 1.5708, 3.12817, 1.5708,
 1.5708, 1.5708, 1.5708, 1.57076, 1.57078, 1.57081, 1.57077, 1.57079,
 3.1321, 1.55977, 1.57078, 1.5708, 1.56408, 1.57081, 3.12997, 1.57394}
```

From the optimized parameters, make predictions.

In[ ]:=

```
mm = mat /. Thread[aa → ww];
vv = Transpose[mm.Transpose[in]];
predicted = Map[Sign@Chop[Conjugate[#].Z1.#] &, vv]
```

Out[ ]=

```
{-1, 1, 1, -1, 1, -1, -1, 1, 1, -1, -1, 1, -1, 1, 1, -1}
```

In[ ]:=

```
predicted - Values[data]
```

Out[ ]=

```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

We can see that two layers is sufficient to properly predict the parity values.

Questions

1. Why does it work? Or, is there any better or alternative way to construct the variational quantum circuit for the problem (or in a problem-independent way)?
2. In this example, we have not used any quantum computer. As long as you can express the variational state and calculate the expectation value efficiently, no quantum computer is required. What is the advantage of using a quantum computer?

Summary

Keywords

- Variational principle
- Variational quantum circuit

- Parametrized quantum circuit

Related Links

- Q3 Tutorial: Single-Qubit Gates
- Q3 Tutorial: Two-Qubit Gates
- A Quantum Workbook (Springer, 2022), Chapter 2.
- Maria Schuld, PennyLane demonstration “Variational classifier”.