pgtio7foz

May 12, 2024

1 Variational Quantum Eigensolver VQE

Función objetivo $f(z_1, z_2, z_3) = 2z_1z_2 - 3z_2 + z_2z_3$

[2]:
```
import pennylane as qml
import math
from scipy.optimize  import minimize
```

Definimos el ansatz

[3]:
```
def ansatz(alpha):
    qml.RX(alpha[0], wires=1)
    qml.RX(alpha[1], wires=2)
    qml.RX(alpha[2], wires=3)
    qml.CNOT(wires=[1,2])
    qml.CNOT(wires=[2,3])
```

Creamos el circuito que representa nuesto Hamiltoniano

[4]:
```
dev = qml.device("default.qubit", wires = [1,2,3], shots =1000)

@qml.qnode(dev)
def circ1(alpha):
    ansatz(alpha)
    return qml.expval(qml.PauliZ(wires=1) @ qml.PauliZ(wires=2))

@qml.qnode(dev)
def circ2(alpha):
    ansatz(alpha)
    return qml.expval(qml.PauliZ(wires=2))

@qml.qnode(dev)
def circ3(alpha):
    ansatz(alpha)
    return qml.expval(qml.PauliZ(wires=2) @ qml.PauliZ(wires=3))
def hamiltoniano(alpha):
    return 2 * circ1(alpha) - 3 * circ2(alpha) + circ3(alpha)
```

Ahora vamos a utilzar un optimizar clásico para recalcular los valores del ansazt y repetir el proceso.

[5]:
```
pi = math.pi
alpha = [pi/2, pi/2, pi/2]
sol = minimize(hamiltoniano, alpha, method='COBYLA',options={'maxiter': 200} )
print("parametros optimos:", sol.x)
```

```
parametros optimos: [3.15587772 3.09996295 3.15681488]
```

Estos son los parametros optimos para nuestro Hamiltoniano, para obtener el resultado a partir de los parámatros no habrá más que aplicar estos parametros al ansazt y medir

[6]:
```
dev = qml.device("default.qubit", wires = [1, 2, 3], shots = 1)

@qml.qnode(dev)
def result(dev):
    ansatz(sol.x)
    return [qml.sample(qml.PauliZ(wires=i)) for i in [1, 2, 3]]

result(dev)
```

[6]:
```
[array(-1), array(1), array(-1)]
```

El resultado es [-1,1,-1]