q_fact

March 17, 2024

[47]:
```
from qiskit import *
from qiskit.circuit import Parameter
from qiskit.tools.visualization import plot_histogram
from qiskit import Aer, transpile

import numpy as np
import matplotlib.pyplot as plt
from scipy.special import gamma

simulator = Aer.get_backend('aer_simulator', device="GPU")
```

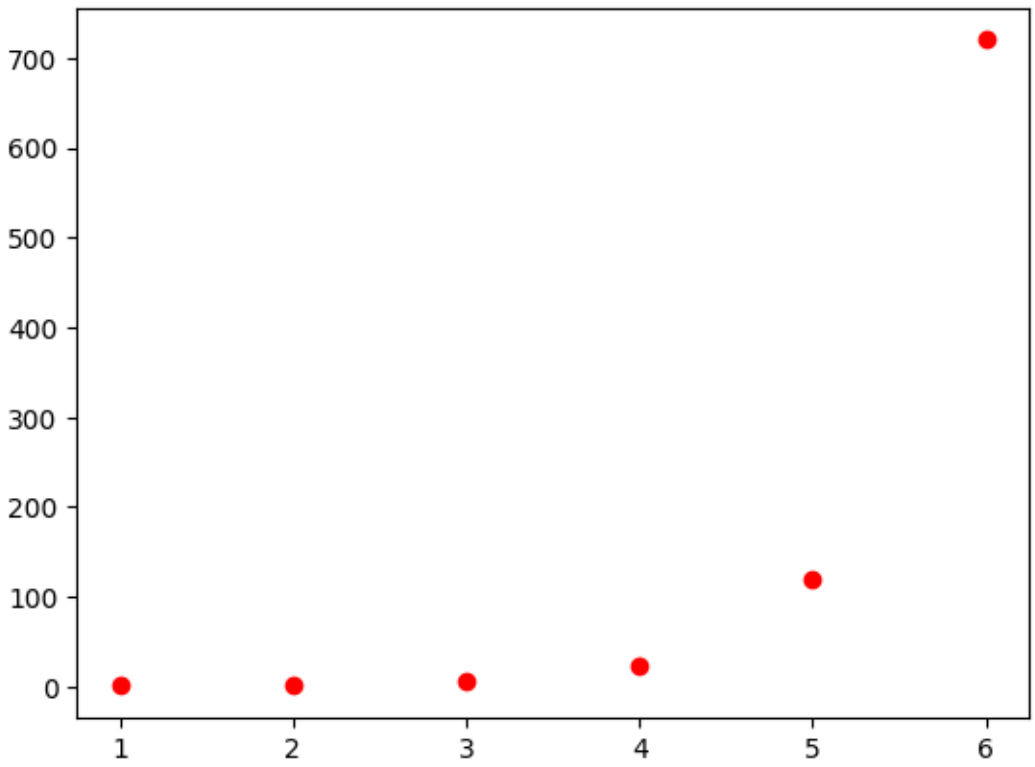
Mise en place des paramétres

[48]:
```
nqubits = 3
train_depth = 3
time_steps = 0.77
```

Points d'entrainements

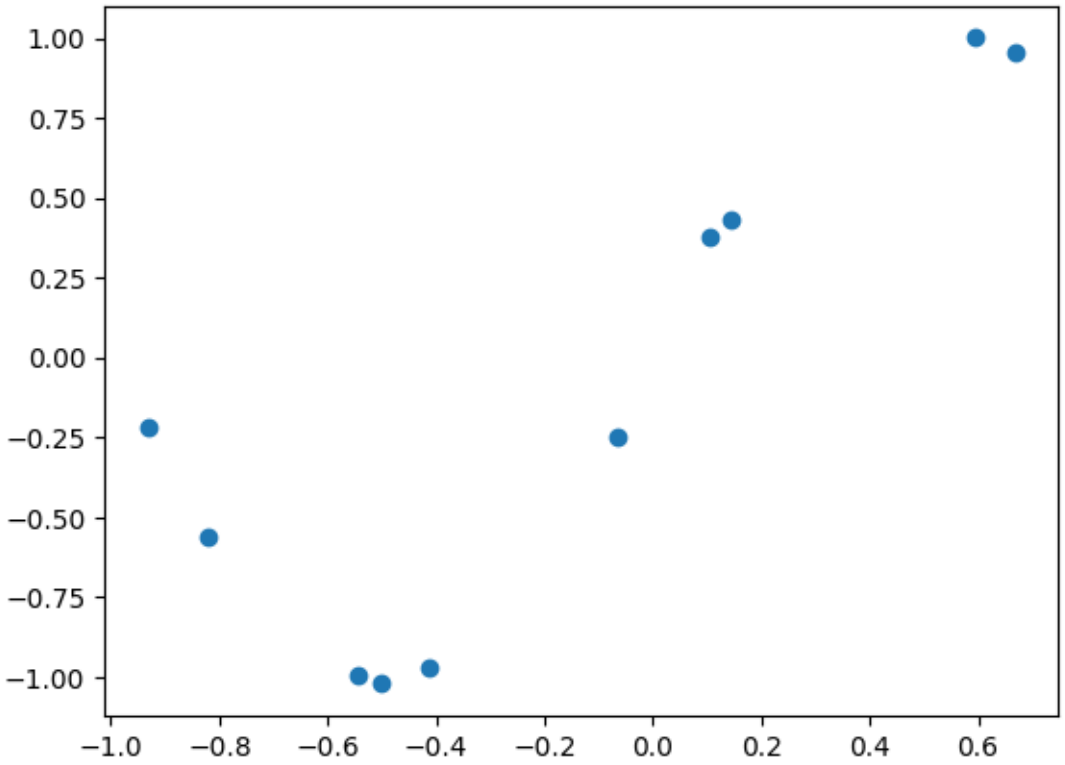
[49]:
```
def factorials(x):
    return gamma(x+1)

train_x = [n for n in range(1,7)]
train_y = [factorials(n) for n in train_x]

plt.plot(train_x, train_y, 'ro')
```

[49]:
```
[<matplotlib.lines.Line2D at 0x276ca7dd750>]
```



[50]:
```
## Take number_x_train points from [x_min, x_max] randomly and use them as the␣
↪training data.
x_min = - 1.; x_max = 1.
num_x_train = 10

## The 1-variable function we want to learn
func_to_learn = lambda x: np.sin(x*np.pi)
```

[51]:
```
#### Prepare training data

train_x = x_min + (x_max - x_min) * np.random.rand(num_x_train)
train_y = func_to_learn(train_x)

# Add noise to a clean sin function
mag_noise = 0.05
train_y = train_y + mag_noise * np.random.randn(num_x_train)

plt.plot(train_x, train_y, "o")
plt.show()
```



État de départ

[52]:
```
def U_in(x):
    qc = QuantumCircuit(nqubits)
    angle_y = np.arcsin(x)
    angle_z = np.arccos(x**2)
    for i in range(nqubits):
        qc.ry(angle_y, i)
        qc.rz(angle_z, i)
    return qc
```

État parmétrique

[53]:
```
# Import the matrix X, I, and Z gates from the qiskit library
from functools import reduce

X = np.array([[0, 1], [1, 0]])
Z = np.array([[1, 0], [0, -1]])
I = np.array([[1, 0], [0, 1]])

def make_fullgate(list_SiteAndOperator, nqubit):
    '''
    Receive list_SiteAndOperator = [ [i_0, O_0], [i_1, O_1], ...] ,
```

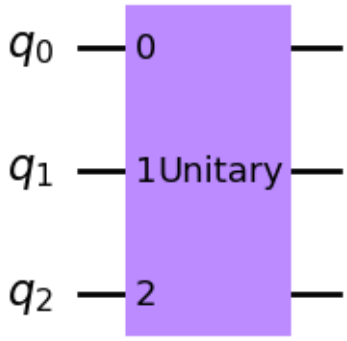
```
    insert Identity into irrelevent qubit, and create (2**nqubit, 2**nqubit)␣
 ↪size matrix
    I(0) * ... * O_0(i_0) * ... * O_1(i_1) ...
    '''
    list_Site = [SiteAndOperator[0] for SiteAndOperator in list_SiteAndOperator]
    list_SingleGates = []  ## list single 1-qubit gates and reduce them using␣
 ↪np.kron
    cnt = 0
    for i in range(nqubit):
        if (i in list_Site):
            list_SingleGates.append( list_SiteAndOperator[cnt][1] )
            cnt += 1
        else: ## insert identity if i is not included in list_Site
            list_SingleGates.append(I)

    return reduce(np.kron, list_SingleGates)

ham = np.zeros((2**nqubits,2**nqubits), dtype = complex)
for i in range(nqubits): ## i runs 0 to nqubit-1
    Jx = -1. + 2.*np.random.rand() ## random number between -1~1
    ham += Jx * make_fullgate( [ [i, X] ], nqubits)
    for j in range(i+1, nqubits):
        J_ij = -1. + 2.*np.random.rand()
        ham += J_ij * make_fullgate ([ [i, Z], [j, Z]], nqubits)

## Create a time evolution operator by diagonalizing.  H*P = P*D <-> H =␣
 ↪P*D*P^dagger
diag, eigen_vecs = np.linalg.eigh(ham)
time_evol_op = np.dot(np.dot(eigen_vecs, np.diag(np.exp(-1j*time_steps*diag))),␣
 ↪eigen_vecs.T.conj()) # e^-iHT
```

[54]:
```
time_evol_op.shape
```

[54]:
```
(8, 8)
```

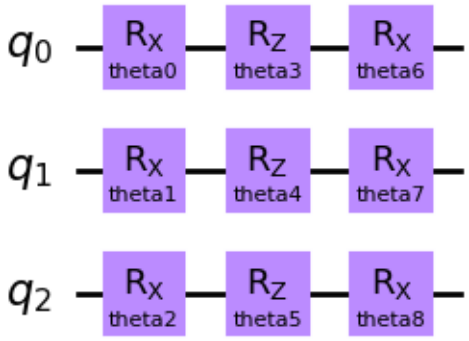
[55]:
```
# convert time_evol_op to a qiskit operator
from qiskit.quantum_info import Operator
time_evol_op = Operator(time_evol_op)
evolve = QuantumCircuit(nqubits)
evolve.unitary(time_evol_op, range(nqubits))
evolve.draw('mpl')
```

[55]:



[56]:
```
U_rot = QuantumCircuit(nqubits)
# Create 3*nqubits parameters for the rotation gates rx, rz, rx on each qubit␣
 ↪for U_rot
params = [Parameter('theta'+str(i)) for i in range(3*nqubits)]
for i in range(nqubits):
    U_rot.rx(params[i], i)
    U_rot.rz(params[i+nqubits], i)
    U_rot.rx(params[i+2*nqubits], i)
U_rot.draw('mpl')
```

[56]:



[57]:
```
def U_out(parmameters):
    qc = QuantumCircuit(nqubits)
    for c in range(train_depth):
        qc.compose(evolve, inplace=True)
```

```
        rot_gate = U_rot.bind_parameters({params[i]: parmameters[3*c*nqubits +␣
↪i] for i in range(3*nqubits)})
        qc.compose(rot_gate, inplace=True)
    return qc
```

Mesure

[58]:
```
def circ_meas(qc):
    qc.measure_all()
    qc = transpile(qc, simulator)
    counts = simulator.run(qc).result().get_counts()
    tot_counts = sum(counts.values())
    zero_state = 0
    for key in counts:
        if key[0] == '0':
            zero_state += counts[key]
    expected = zero_state/tot_counts
    return expected*2-1
```

Circuit complet

[59]:
```
def complete_circuit(x, *parameters):
    qc = QuantumCircuit(nqubits)
    qc.compose(U_in(x), inplace=True)
    qc.compose(U_out(*parameters), inplace=True)
    return circ_meas(qc)
```

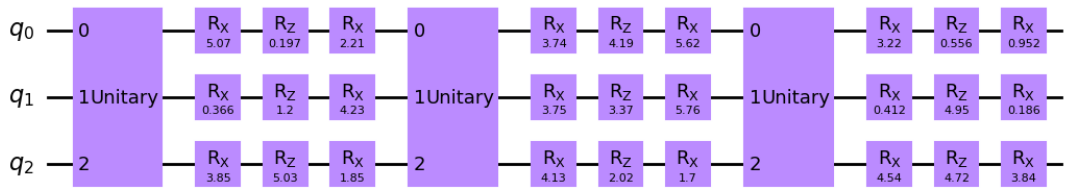
[60]:
```
# Create a set of initial parameters
initial_parameters = np.random.rand(train_depth, 3*nqubits)*2*np.pi
initial_parameters = initial_parameters.flatten()
initial_parameters.shape
```

[60]: (27,)

[61]:
```
U_out(initial_parameters).draw('mpl')
```

```
C:\Users\romai\AppData\Local\Temp\ipykernel_3548\869448860.py:5:
DeprecationWarning: The method
``qiskit.circuit.quantumcircuit.QuantumCircuit.bind_parameters()`` is deprecated
as of qiskit 0.45.0. It will be removed no earlier than 3 months after the
release date. Use assign_parameters() instead
  rot_gate = U_rot.bind_parameters({params[i]: parmameters[3*c*nqubits + i] for
i in range(3*nqubits)})
```

[61]:



[62]:
```
complete_circuit(1, initial_parameters)
```

```
C:\Users\romai\AppData\Local\Temp\ipykernel_3548\869448860.py:5:
DeprecationWarning: The method
``qiskit.circuit.quantumcircuit.QuantumCircuit.bind_parameters()`` is deprecated
as of qiskit 0.45.0. It will be removed no earlier than 3 months after the
release date. Use assign_parameters() instead
  rot_gate = U_rot.bind_parameters({params[i]: parmameters[3*c*nqubits + i] for
i in range(3*nqubits)})
```

[62]: 0.224609375

Cost

[63]:
```
def cost_function(parameters):
    tot = []
    x = train_x
    y = train_y
    for i in range(len(x)):
        expected = complete_circuit(x[i], parameters)
        cost = (expected - y[i])**2
        tot.append(cost)
    return np.mean(tot)
```

[64]:
```
cost_function(initial_parameters)
```

```
C:\Users\romai\AppData\Local\Temp\ipykernel_3548\869448860.py:5:
DeprecationWarning: The method
``qiskit.circuit.quantumcircuit.QuantumCircuit.bind_parameters()`` is deprecated
as of qiskit 0.45.0. It will be removed no earlier than 3 months after the
release date. Use assign_parameters() instead
  rot_gate = U_rot.bind_parameters({params[i]: parmameters[3*c*nqubits + i] for
i in range(3*nqubits)})
```

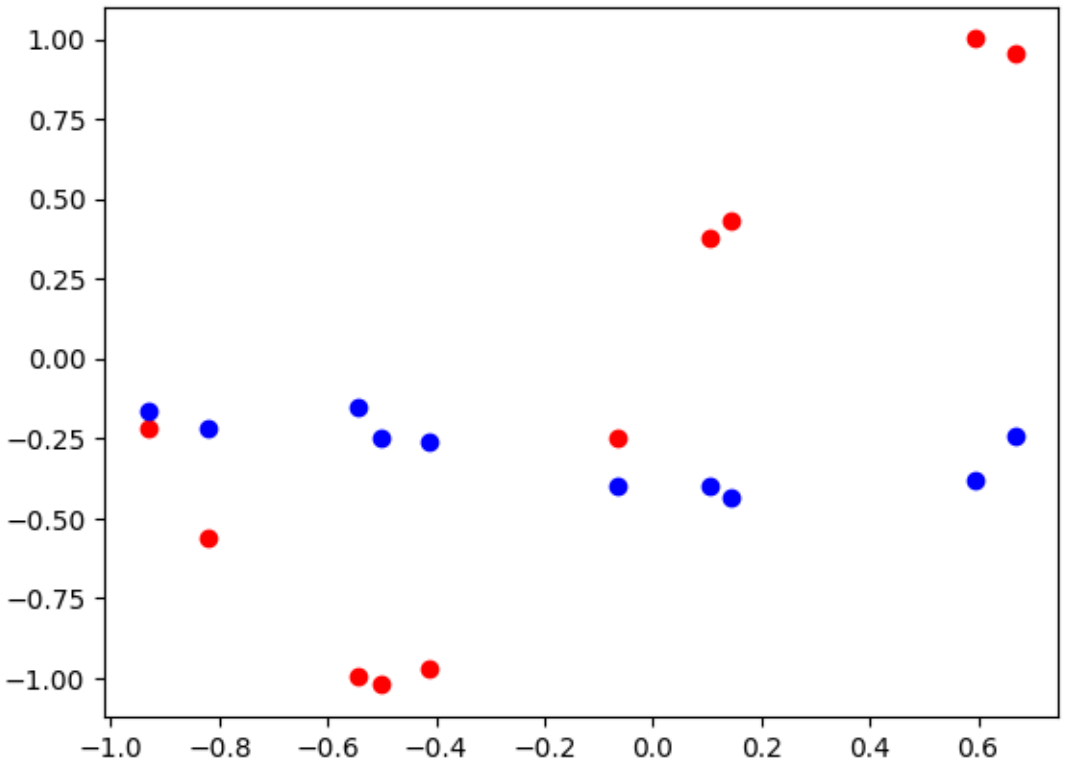
[64]: 0.6524846020707463

[65]:
```
plt.plot(train_x, train_y, 'ro')
```

```
plt.plot(train_x, [complete_circuit(x, initial_parameters) for x in train_x],␣
 ↪'bo')
```

```
C:\Users\romai\AppData\Local\Temp\ipykernel_3548\869448860.py:5:
DeprecationWarning: The method
``qiskit.circuit.quantumcircuit.QuantumCircuit.bind_parameters()`` is deprecated
as of qiskit 0.45.0. It will be removed no earlier than 3 months after the
release date. Use assign_parameters() instead
  rot_gate = U_rot.bind_parameters({params[i]: parmameters[3*c*nqubits + i] for
i in range(3*nqubits)})
```

[65]: `[<matplotlib.lines.Line2D at 0x276cad49a50>]`



Entrainement

[66]:
```
from scipy.optimize import minimize
result = minimize(cost_function, initial_parameters, method='Nelder-Mead')
```

```
C:\Users\romai\AppData\Local\Temp\ipykernel_3548\869448860.py:5:
DeprecationWarning: The method
``qiskit.circuit.quantumcircuit.QuantumCircuit.bind_parameters()`` is deprecated
as of qiskit 0.45.0. It will be removed no earlier than 3 months after the
release date. Use assign_parameters() instead
```

```
  rot_gate = U_rot.bind_parameters({params[i]: parmameters[3*c*nqubits + i] for
i in range(3*nqubits)})
```

[67]:
```
result.fun
```

[67]:
```
0.16782591724560741
```

[68]:
```
theta = result.x
theta
```

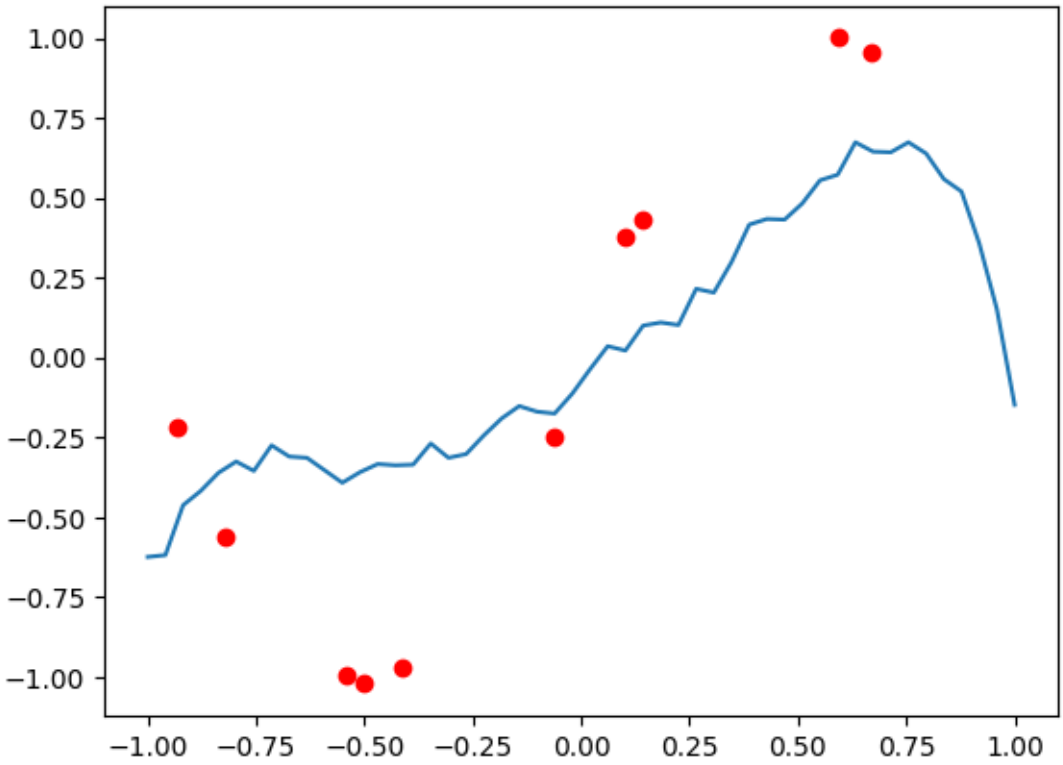
[68]:
```
array([6.25612847, 0.38326116, 3.59729313, 0.19109314, 1.34449733,
       5.96628712, 2.25685168, 3.75486093, 1.88045734, 4.20057105,
       3.52881964, 4.36222762, 4.14172856, 3.3431398 , 2.02170976,
       5.37981345, 6.24014735, 1.54331279, 2.83782114, 0.395741  ,
       4.35468793, 0.56290581, 4.93814148, 4.56451642, 0.81059456,
       0.18499445, 3.83653534])
```

Tests

[69]:
```
plt.plot(train_x, train_y, 'ro')
plt.plot(np.linspace(-1,1), [complete_circuit(x, theta) for x in np.
 ↪linspace(-1,1)], '-')
```

```
C:\Users\romai\AppData\Local\Temp\ipykernel_3548\869448860.py:5:
DeprecationWarning: The method
``qiskit.circuit.quantumcircuit.QuantumCircuit.bind_parameters()`` is deprecated
as of qiskit 0.45.0. It will be removed no earlier than 3 months after the
release date. Use assign_parameters() instead
  rot_gate = U_rot.bind_parameters({params[i]: parmameters[3*c*nqubits + i] for
i in range(3*nqubits)})
```

[69]:
```
[<matplotlib.lines.Line2D at 0x276cade0670>]
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00