Task2 (2)

August 23, 2022

Task 2: Input encoding

[1]:
```
import numpy as np
from qiskit.algorithms.linear_solvers.numpy_linear_solver import␣
↪NumPyLinearSolver
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning:
scipy._lib.messagestream.MessageStream size changed, may indicate binary
incompatibility. Expected 56 from C header, got 64 from PyObject
```

Make a quantum circuit that prepares the input state

[2]:
```
#
# Function that randomly pick a 2 dimentional state to simulate position of␣
↪satellite
#

from random import randrange
def random_quantum_state():
    # quantum state
    quantum_state=[0,0]
    random_number1 = randrange(-100,101)
    #print("the first number is",random_number1)
    random_number2 = randrange(-100,101)
    #print("the second number is",random_number2)
    #probability=random_number1**2+random_number2**2
   # print("probability is",probability)
    #amplitude1=random_number1/(probability**0.5)
   # print("amplitude 1 is",amplitude1)
    #amplitude2=random_number2/(probability**0.5)
   # print("amplitude 2 is",amplitude2)
    quantum_state[0]=random_number1
    quantum_state[1]=random_number2
    #sum=amplitude1**2+amplitude2**2
    #print("sum=",sum)
    return quantum_state
```

We make a function that normalizes the state we made in the function above

[3]:
```
#
# Function that normalizes a 2 dimentional state to simulate quantum state of
↪the position of satellite
#
def normal_quantum_state(quantum_state):
    probability=quantum_state[0]**2+quantum_state[1]**2
    amplitude1=quantum_state[0]/(probability**0.5)
    amplitude2=quantum_state[1]/(probability**0.5)
    quantum_state[0]=amplitude1
    quantum_state[1]=amplitude2
    return quantum_state
```

We can make sure our functions work propertly with the following 2 cells.

[4]:
```
#
# check randomly pick function
#
rs = random_quantum_state()
rs
```

[4]:
```
[38, -85]
```

[5]:
```
#check function that normilizes the state
n_quantum_state = normal_quantum_state(rs)
n_quantum_state
```

[5]:
```
[0.4081305412501237, -0.9129235791121187]
```

We define a fucntion that checks whether the picked quantum state is valid

[6]:
```
# Function that checks whether the picked quantum state is valid
from random import randrange
def check_random_quantum_state(quantum_state):
    value=0
    for i in range(len(quantum_state)):
        value+=quantum_state[i]**2
    if (value - 1)**2 < 0.00000001:
        return True
    else:
        return False
```

We can make sure our function work propertly with the following cell

[7]:
```
#
# check function that states if the quantum state is valid
#
check_random_quantum_state(n_quantum_state)
```

[7]: True

Finally we make a quantum circuit that prepares the input state, here we cosnider 10 satellites.

[8]:
```
#Function to prepare the quantum state of N satellites
quantum_satellites=[] #this will be our quantum state for the N satellites
N=10 #Number of satellites
for i in range(N):
    picked_quantum_state=random_quantum_state() #call function that randomly␣
 ↪pick a 2 dimentional state to simulate position of satellite
    n_state = normal_quantum_state(picked_quantum_state)  # Function that␣
 ↪normalizes a 2 dimentional state to simulate quantum state of the position␣
 ↪of satellite
    print(n_state,"this is randomly picked quantum state for satellite",i)
    print("Is it valid?",check_random_quantum_state(n_state)) #check if it is a␣
 ↪valid quantum state
    if check_random_quantum_state(n_state)==True: #if it is a quantum valid␣
 ↪quantum state then save it to our quantum state for N satellites
        x= n_state[0]  #define x component
        y = n_state[1]  #define y component
        quantum_satellites.append(x)  #save x position of satellite on our␣
 ↪quantum state
        quantum_satellites.append(y)   #save y position of satellite on our␣
 ↪quantum state
    print() # print an empty line
print("The imput quantum state for the",N,"satellites is given by the following␣
 ↪state")
print()
print("x=",quantum_satellites)  #print the quantum state for the N satellites
```

```
[0.9449860734402582, 0.3271105638831663] this is randomly picked quantum state
for satellite 0
Is it valid? True

[0.8372705045624257, -0.546788900938727] this is randomly picked quantum state
for satellite 1
Is it valid? True

[-0.5985256885300932, 0.8011036138787402] this is randomly picked quantum state
for satellite 2
Is it valid? True

[-0.9496887635303413, -0.3131952305259636] this is randomly picked quantum state
for satellite 3
Is it valid? True
```

```
[-0.696785684263249, 0.7172793808592269] this is randomly picked quantum state
for satellite 4
Is it valid? True

[0.7022134834058589, 0.7119664484531625] this is randomly picked quantum state
for satellite 5
Is it valid? True

[-0.973417168333576, -0.2290393337255473] this is randomly picked quantum state
for satellite 6
Is it valid? True

[-0.4948386187009886, -0.8689848913773458] this is randomly picked quantum state
for satellite 7
Is it valid? True

[-0.9989685402102997, 0.045407660918649985] this is randomly picked quantum
state for satellite 8
Is it valid? True

[0.7196931822512745, 0.6942922464071118] this is randomly picked quantum state
for satellite 9
Is it valid? True

The imput quantum state for the 10 satellites is given by the following state

x= [0.9449860734402582, 0.3271105638831663, 0.8372705045624257,
-0.546788900938727, -0.5985256885300932, 0.8011036138787402,
-0.9496887635303413, -0.3131952305259636, -0.696785684263249,
0.7172793808592269, 0.7022134834058589, 0.7119664484531625, -0.973417168333576,
-0.2290393337255473, -0.4948386187009886, -0.8689848913773458,
-0.9989685402102997, 0.045407660918649985, 0.7196931822512745,
0.6942922464071118]
```

We see that our quantum state can make a quantum circuit that prepares the input state for N satellites. We choose that it made smaller quantum states in each itaration, rather than performing the measurment of positions of the N satellites and then preparing the quantum state, the introduction of another for cicle would increase the complexity of the system.

[ ]:

```
```