```
!pip install qiskit
```

```
Collecting qiskit
  Downloading qiskit-0.30.0.tar.gz (12 kB)
Collecting qiskit-terra==0.18.2
  Downloading qiskit_terra-0.18.2-cp37-cp37m-manylinux2010_x86_64.whl (6.1 MB)
     |████████████████████████████████| 6.1 MB 4.3 MB/s
Collecting qiskit-aer==0.9.0
  Downloading qiskit_aer-0.9.0-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (17.9 MB)
     |████████████████████████████████| 17.9 MB 179 kB/s
Collecting qiskit-ibmq-provider==0.16.0
  Downloading qiskit_ibmq_provider-0.16.0-py3-none-any.whl (235 kB)
     |████████████████████████████████| 235 kB 61.5 MB/s
Collecting qiskit-ignis==0.6.0
  Downloading qiskit_ignis-0.6.0-py3-none-any.whl (207 kB)
     |████████████████████████████████| 207 kB 45.1 MB/s
Collecting qiskit-aqua==0.9.5
  Downloading qiskit_aqua-0.9.5-py3-none-any.whl (2.1 MB)
     |████████████████████████████████| 2.1 MB 70.0 MB/s
Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.7/dist-packages (from qiskit-aer==0.9.0->qiskit) (1.19.5)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/dist-packages (from qiskit-aer==0.9.0->qiskit) (1.4.1)
Collecting pybind11>=2.6
  Downloading pybind11-2.7.1-py2.py3-none-any.whl (200 kB)
     |████████████████████████████████| 200 kB 54.1 MB/s
Collecting retworkx>=0.8.0
  Downloading retworkx-0.10.2-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_
     |████████████████████████████████| 1.4 MB 46.7 MB/s
Requirement already satisfied: fastdtw<=0.3.4 in /usr/local/lib/python3.7/dist-packages (from qiskit-aqua==0.9.5->qiskit) (0.3.4)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.7/dist-packages (from qiskit-aqua==0.9.5->qiskit) (0.22.2.post
Requirement already satisfied: setuptools>=40.1.0 in /usr/local/lib/python3.7/dist-packages (from qiskit-aqua==0.9.5->qiskit) (57.4.0)
Collecting yfinance>=0.1.62
  Downloading yfinance-0.1.63.tar.gz (26 kB)
Collecting docplex>=2.21.207
  Downloading docplex-2.22.213.tar.gz (634 kB)
     |████████████████████████████████| 634 kB 68.3 MB/s
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from qiskit-aqua==0.9.5->qiskit) (1.1.5)
Requirement already satisfied: psutil>=5 in /usr/local/lib/python3.7/dist-packages (from qiskit-aqua==0.9.5->qiskit) (5.4.8)
Collecting quandl
  Downloading Quandl-3.6.1-py2.py3-none-any.whl (26 kB)
Requirement already satisfied: h5py<3.3.0 in /usr/local/lib/python3.7/dist-packages (from qiskit-aqua==0.9.5->qiskit) (3.1.0)
Collecting dlx<=1.0.4
  Downloading dlx-1.0.4.tar.gz (5.5 kB)
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.7/dist-packages (from qiskit-aqua==0.9.5->qiskit) (1.7.1)
Requirement already satisfied: requests>=2.19 in /usr/local/lib/python3.7/dist-packages (from qiskit-ibmq-provider==0.16.0->qiskit) (2.23.
Collecting websocket-client>=1.0.1
  Downloading websocket_client-1.2.1-py2.py3-none-any.whl (52 kB)
     |████████████████████████████████| 52 kB 1.4 MB/s
Collecting requests-ntlm>=1.1.0
  Downloading requests_ntlm-1.1.0-py2.py3-none-any.whl (5.7 kB)
Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/python3.7/dist-packages (from qiskit-ibmq-provider==0.16.0->qiskit)
Requirement already satisfied: urllib3>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from qiskit-ibmq-provider==0.16.0->qiskit) (1.24.3
Collecting tweedledum<2.0,>=1.1
  Downloading tweedledum-1.1.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (943 kB)
     |████████████████████████████████| 943 kB 58.2 MB/s
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra==0.18.2->qiskit) (2.6.0)
Collecting python-constraint>=1.4
  Downloading python-constraint-1.4.0.tar.bz2 (18 kB)
Collecting ply>=3.10
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 6.4 MB/s
Collecting symengine>0.7
  Downloading symengine-0.8.1-cp37-cp37m-manylinux2010_x86_64.whl (38.2 MB)
```

```
!pip install qiskit_machine_learning
```

```
Collecting qiskit_machine_learning
  Downloading qiskit_machine_learning-0.2.1-py3-none-any.whl (96 kB)
     |████████████████████████████████| 96 kB 2.8 MB/s
```

```
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from qiskit_machine_learning) (1.19.5)
Requirement already satisfied: fastdtw in /usr/local/lib/python3.7/dist-packages (from qiskit_machine_learning) (0.3.4)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.7/dist-packages (from qiskit_machine_learning) (0.22.2.post1)
Requirement already satisfied: qiskit-terra>=0.18.0 in /usr/local/lib/python3.7/dist-packages (from qiskit_machine_learning) (0.18.2)
Requirement already satisfied: psutil>=5 in /usr/local/lib/python3.7/dist-packages (from qiskit_machine_learning) (5.4.8)
Requirement already satisfied: setuptools>=40.1.0 in /usr/local/lib/python3.7/dist-packages (from qiskit_machine_learning) (57.4.0)
Requirement already satisfied: scipy>=1.4 in /usr/local/lib/python3.7/dist-packages (from qiskit_machine_learning) (1.4.1)
Requirement already satisfied: tweedledum<2.0,>=1.1 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine
Requirement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine_lear
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine_learning)
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine_learning) (0.3
Requirement already satisfied: fastjsonschema>=2.10 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine
Requirement already satisfied: retworkx>=0.9.0 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine_learni
Requirement already satisfied: python-dateutil>=2.8.0 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine_
Requirement already satisfied: python-constraint>=1.4 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine
Requirement already satisfied: symengine>0.7 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine_learnir
Requirement already satisfied: ply>=3.10 in /usr/local/lib/python3.7/dist-packages (from qiskit-terra>=0.18.0->qiskit_machine_learning) (3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.8.0->qiskit-terra>=0.18.0->qiski
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn>=0.20.0->qiskit_machine_learning)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.7/dist-packages (from sympy>=1.3->qiskit-terra>=0.18.0->qiskit_m
Installing collected packages: qiskit-machine-learning
Successfully installed qiskit-machine-learning-0.2.1
```

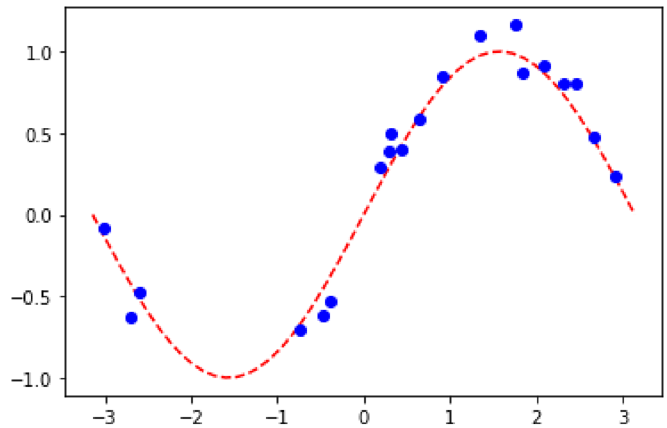
Superposition, **quantum measurement**, and **entanglement** are three phenomena that are central to quantum computing.if you are a quantum particle, then you can have a certain probability of facing left AND a certain probability of facing right due to a phenomenon known as superposition (also known as coherence).

A quantum particle such as an electron has its own “facing left or facing right” properties, for example spin, referred to as either up or down, or to make it more relatable to classical binary computing, let’s just say 1 or 0. When a quantum particle is in a superposition state, it’s a linear combination of an infinite number of states between 1 and 0, but you don’t know which one it will be until you actually look at it, which brings up our next phenomenon, quantum measurement.

```
# Necessary imports

import·numpy·as·np
import·matplotlib.pyplot·as·plt

from·torch·import·Tensor
from·torch.nn·import·Linear,·CrossEntropyLoss,·MSELoss
from·torch.optim·import·LBFGS

from·qiskit··import·Aer,·QuantumCircuit
from·qiskit.utils·import·QuantumInstance
from qiskit.opflow import AerPauliExpectation
from qiskit.circuit import Parameter
from qiskit.circuit.library import RealAmplitudes, ZZFeatureMap
from qiskit_machine_learning.neural_networks import CircuitQNN, TwoLayerQNN
from qiskit_machine_learning.connectors import TorchConnector

# declare quantum instance
qi = QuantumInstance(Aer.get_backend('aer_simulator_statevector'))
```

```
# Generate random dataset

np.random.seed(0)
num_samples = 20
eps = 0.2
lb, ub = -np.pi, np.pi
f = lambda x: np.sin(x)

X = (ub - lb)*np.random.rand(num_samples, 1) + lb
```

```
y = f(X) + eps*(2*np.random.rand(num_samples, 1)-1)
plt.plot(np.linspace(lb, ub), f(np.linspace(lb, ub)), 'r--')
plt.plot(X, y, 'bo')
plt.show()
```



We create the circuit. The fundamental element of quantum computing is the quantum circuit. A quantum circuit is a computational routine consisting of coherent quantum operations on quantum data, such as qubits. It is an ordered sequence of quantum gates, measurements and resets, which may be conditioned on real-time classical computation. A set of quantum gates is said to be universal if any unitary transformation of the quantum data can be efficiently approximated arbitrarily well as as sequence of gates in the set. Any quantum program can be represented by a sequence of quantum circuits and classical near-time computation. A quantum circuit is a computational routine consisting of coherent quantum operations on quantum data, such as qubits, and concurrent real-time classical computation. It is an ordered sequence of quantum gates, measurements and resets, all of which may be conditioned on and use data from the real-time classical computation.

A set of quantum gates is said to be universal if any unitary transformation of the quantum data can be efficiently approximated arbitrarily well as a sequence of gates in the set. Any quantum program can be represented by a sequence of quantum circuits and non-concurrent classical computation.

The quantum circuit uses three qubits and two classical bits. There are four main components in this quantum circuit.

Initialization and reset First, we need to start our quantum computation with a well-defined quantum state. This is achieved using the initialization and reset operations. The resets can be performed by a combination of single-qubit gates and concurrent real-time classical computation that monitors whether we have successfully created the desired state through measurements. The initialization of q into a desired state $|\psi\rangle$ can then follow by applying single-qubit gates.

Quantum gates Second, we apply a sequence of quantum gates that manipulate the three qubits as required by the teleportation algorithm. In this case, we only need to apply single-qubit Hadamard (H) and two-qubit Controlled-X (⊕) gates.

Measurements Third, we measure two of the three qubits. A classical computer interprets the measurements of each qubit as classical outcomes (0 and 1) and stores them in the two classical bits.

Classically conditioned quantum gates Fourth, we apply single-qubit Z and X quantum gates on the third qubit. These gates are conditioned on the results of the measurements that are stored in the two classical bits. In this case, we are using the results of the classical computation concurrently in real-time within the same quantum circuit.

The act of observing or measuring a quantum particle collapses the superposition state (also known as decoherence) and the particle takes on a classical binary state of either 1 or 0.

```
# Construct simple feature map
param_x = Parameter('x')
feature_map = QuantumCircuit(1, name='fm')
feature_map.ry(param_x,·0)
```

```

#·Construct·simple·feature·map
param_y·=·Parameter('y')
ansatz·=·QuantumCircuit(1,·name='vf')
ansatz.ry(param_y,·0)

#·Construct·QNN
qnn3·=·TwoLayerQNN(1,·feature_map,·ansatz,·quantum_instance=qi)
print(qnn3.operator)

#·Set·up·PyTorch·module
#·Reminder:·If·we·don't·explicitly·declare·the·initial·weights
#·they·are·chosen·uniformly·at·random·from·[-1,·1].
#·Set·seed·for·random·dataset
np.random.seed(7)
initial_weights·=·0.1*(2*np.random.rand(qnn3.num_weights)·-·1)
model3·=·TorchConnector(qnn3,·initial_weights)
```

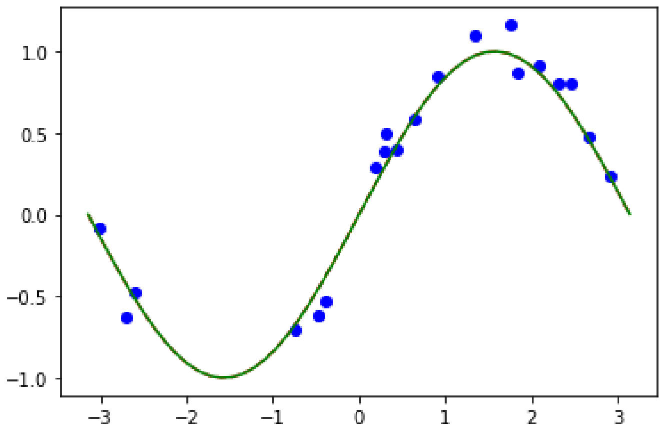
```
ComposedOp([
  OperatorMeasurement(1.0 * Z),
  CircuitStateFn(
       ┌───────┐ ┌───────┐
  q_0: ┤ fm(x) ├─┤ vf(y) ├
       └───────┘ └───────┘
  )
])
```

```
# Define optimizer and loss function
optimizer = LBFGS(model3.parameters())
f_loss = MSELoss(reduction='sum')

# Start training
model3.train()   # set model to training mode

# Define objective function
def closure():
    optimizer.zero_grad(set_to_none=True)        # Initialize gradient
    loss = f_loss(model3(Tensor(X)), Tensor(y))  # Compute batch loss
    loss.backward()                              # Backward pass
    print(loss.item())                           # Print loss
    return loss

# Run optimizer
optimizer.step(closure)
```

```
21.262964248657227
3.3382768630981445
20.620962142944336
2.447436809539795
25.977062225341797
3.206045627593994
13.661500930786133
25.774295806884766
21.07257080078125
5.010274410247803
1.3200629949569702
0.25994038581848145
0.24564915895462036
0.24564766883850098
tensor(21.2630, grad_fn=<MseLossBackward>)
```

```
# Plot target function
plt.plot(np.linspace(lb, ub), f(np.linspace(lb, ub)), 'r--')

```

```
# Plot data
plt.plot(X, y, 'bo')

# Plot fitted line
y_ = []
for x in np.linspace(lb, ub):
    output = model3(Tensor([x]))
    y_ += [output.detach().numpy()[0]]
plt.plot(np.linspace(lb, ub), y_, 'g-')
plt.show()
```



0s completed at 12:35 PM