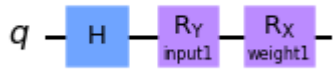
In [1]:
```
from qiskit import QuantumCircuit
from qiskit.circuit import Parameter
from qiskit.circuit.library import RealAmplitudes
from qiskit.utils import algorithm_globals

algorithm_globals.random_seed = 42
```

In [2]:
```
from qiskit_machine_learning.neural_networks import EstimatorQNN
```

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

In [3]:
```
# construct parametrized circuit
params1 = [Parameter("input1"), Parameter("weight1")]
qc1 = QuantumCircuit(1)
qc1.h(0)
qc1.ry(params1[0], 0)
qc1.rx(params1[1], 0)
qc1.draw("mpl")
```

Out[3]:



In [4]:
```
from qiskit.quantum_info import SparsePauliOp

observable1 = SparsePauliOp.from_list([("Y" * qc1.num_qubits, 1)])
```

In [5]:
```
qnn1 = EstimatorQNN(
    circuit=qc1, observables=observable1, input_params=[params1[0]], weight_params=
)
```

In [6]:
```
# define (random) input and weights
input1 = algorithm_globals.random.random(qnn1.num_inputs)
weights1 = algorithm_globals.random.random(qnn1.num_weights)
```

In [7]:
```
# QNN forward pass
qnn1.forward(input1, weights1)
```

Out[7]:
```
array([[0.2970094]])
```

In [8]:
```
# QNN batched forward pass
qnn1.forward([input1, input1], weights1)
```

Out[8]:
```
array([[0.2970094],
       [0.2970094]])
```

In [9]:
```
# QNN backward pass
qnn1.backward(input1, weights1)
```

Out[9]:
```
(None, array([[[0.63272767]]]))
```

In [10]:
```
# QNN batched backward pass
qnn1.backward([input1, input1], weights1)
```

Out[10]:
```
(None,
 array([[[0.63272767]],

        [[0.63272767]]]))
```

In [11]:
```
observable2 = SparsePauliOp.from_list([("Z" * qc1.num_qubits, 1)])

qnn2 = EstimatorQNN(
    circuit=qc1,
    observables=[observable1, observable2],
    input_params=[params1[0]],
    weight_params=[params1[1]],
)
```

In [12]:
```
# QNN forward pass
qnn2.forward(input1, weights1)
```

Out[12]:
```
array([[ 0.2970094 , -0.63272767]])
```

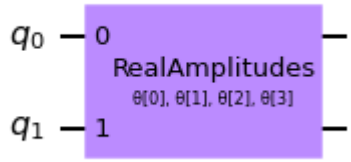
In [13]:
```
# QNN backward pass
qnn2.backward(input1, weights1)
```

Out[13]:
```
(None,
 array([[[0.63272767],
         [0.2970094 ]]]))
```

In [14]:
```
from qiskit.primitives import Sampler
from qiskit_machine_learning.neural_networks import SamplerQNN
```

In [15]:
```
qc = RealAmplitudes(2, entanglement="linear", reps=1)
qc.draw(output="mpl")
```

Out[15]:



In [16]:
```
# specify sampler-based QNN
qnn4 = SamplerQNN(circuit=qc, input_params=[], weight_params=qc.parameters)
```

In [17]:
```
# define (random) input and weights
input4 = algorithm_globals.random.random(qnn4.num_inputs)
weights4 = algorithm_globals.random.random(qnn4.num_weights)
```

In [18]:
```
# QNN forward pass
qnn4.forward(input4, weights4)
```

Out[18]:
```
array([[0.37369597, 0.00083983, 0.42874976, 0.19671444]])
```

In [19]:
```
# QNN backward pass, returns a tuple of matrices, None for the gradients with respe
qnn4.backward(input4, weights4)
```

Out[19]:
```
(None,
 array([[[-0.16667913, -0.42400024,  0.0177156 , -0.40027747],
         [ 0.00403062, -0.0110119 , -0.0177156 ,  0.0128533 ],
         [-0.22984019,  0.39671924, -0.29041568,  0.40027747],
         [ 0.3924887 ,  0.0382929 ,  0.29041568, -0.0128533 ]]]))
```

In [ ]: