FIT_DP_NODP_CIRCUIT

January 3, 2025

[4]:

```
!pip install qiskit_ibm_provider
```

```
Collecting qiskit_ibm_provider
  Downloading qiskit_ibm_provider-0.11.0-py3-none-any.whl.metadata (7.6 kB)
Collecting qiskit>=0.45.0 (from qiskit_ibm_provider)
  Downloading
qiskit-1.3.1-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(12 kB)
Requirement already satisfied: requests>=2.19 in /usr/local/lib/python3.10/dist-
packages (from qiskit_ibm_provider) (2.32.3)
Collecting requests-ntlm>=1.1.0 (from qiskit_ibm_provider)
  Downloading requests_ntlm-1.3.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: numpy>=1.13 in /usr/local/lib/python3.10/dist-
packages (from qiskit_ibm_provider) (1.26.4)
Requirement already satisfied: urllib3>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from qiskit_ibm_provider) (2.2.3)
Requirement already satisfied: python-dateutil>=2.8.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit_ibm_provider) (2.8.2)
Requirement already satisfied: websocket-client>=1.5.1 in
/usr/local/lib/python3.10/dist-packages (from qiskit_ibm_provider) (1.8.0)
Requirement already satisfied: websockets>=10.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit_ibm_provider) (14.1)
Requirement already satisfied: typing-extensions>=4.3 in
/usr/local/lib/python3.10/dist-packages (from qiskit_ibm_provider) (4.12.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.0->qiskit_ibm_provider) (1.17.0)
Collecting rustworkx>=0.15.0 (from qiskit>=0.45.0->qiskit_ibm_provider)
  Downloading rustworkx-0.15.1-cp38-abi3-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.9 kB)
Requirement already satisfied: scipy>=1.5 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=0.45.0->qiskit_ibm_provider) (1.13.1)
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=0.45.0->qiskit_ibm_provider) (1.13.1)
Collecting dill>=0.3 (from qiskit>=0.45.0->qiskit_ibm_provider)
  Downloading dill-0.3.9-py3-none-any.whl.metadata (10 kB)
Collecting stevedore>=3.0.0 (from qiskit>=0.45.0->qiskit_ibm_provider)
  Downloading stevedore-5.4.0-py3-none-any.whl.metadata (2.3 kB)
Collecting symengine<0.14,>=0.11 (from qiskit>=0.45.0->qiskit_ibm_provider)
```

```
  Downloading symengine-0.13.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.2 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.19->qiskit_ibm_provider) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.19->qiskit_ibm_provider) (3.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.19->qiskit_ibm_provider) (2024.12.14)
Requirement already satisfied: cryptography>=1.3 in
/usr/local/lib/python3.10/dist-packages (from requests-
ntlm>=1.1.0->qiskit_ibm_provider) (43.0.3)
Collecting pyspnego>=0.4.0 (from requests-ntlm>=1.1.0->qiskit_ibm_provider)
  Downloading pyspnego-0.11.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-
packages (from cryptography>=1.3->requests-ntlm>=1.1.0->qiskit_ibm_provider)
(1.17.1)
Collecting pbr>=2.0.0 (from
stevedore>=3.0.0->qiskit>=0.45.0->qiskit_ibm_provider)
  Downloading pbr-6.1.0-py2.py3-none-any.whl.metadata (3.4 kB)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from
sympy>=1.3->qiskit>=0.45.0->qiskit_ibm_provider) (1.3.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-
packages (from cffi>=1.12->cryptography>=1.3->requests-
ntlm>=1.1.0->qiskit_ibm_provider) (2.22)
Downloading qiskit_ibm_provider-0.11.0-py3-none-any.whl (249 kB)
                         249.9/249.9 kB
7.7 MB/s eta 0:00:00
Downloading
qiskit-1.3.1-cp39-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (6.7 MB)
                         6.7/6.7 MB
69.7 MB/s eta 0:00:00
Downloading requests_ntlm-1.3.0-py3-none-any.whl (6.6 kB)
Downloading dill-0.3.9-py3-none-any.whl (119 kB)
                         119.4/119.4 kB
9.6 MB/s eta 0:00:00
Downloading pyspnego-0.11.2-py3-none-any.whl (130 kB)
                         130.5/130.5 kB
9.9 MB/s eta 0:00:00
Downloading
rustworkx-0.15.1-cp38-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.0
MB)
                         2.0/2.0 MB
58.3 MB/s eta 0:00:00
Downloading stevedore-5.4.0-py3-none-any.whl (49 kB)
                         49.5/49.5 kB
```

```
3.8 MB/s eta 0:00:00
Downloading
symengine-0.13.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(49.7 MB)
                         49.7/49.7 MB
18.9 MB/s eta 0:00:00
Downloading pbr-6.1.0-py2.py3-none-any.whl (108 kB)
                         108.5/108.5 kB
8.9 MB/s eta 0:00:00
Installing collected packages: symengine, rustworkx, pbr, dill, stevedore,
qiskit, pyspnego, requests-ntlm, qiskit_ibm_provider
Successfully installed dill-0.3.9 pbr-6.1.0 pyspnego-0.11.2 qiskit-1.3.1
qiskit_ibm_provider-0.11.0 requests-ntlm-1.3.0 rustworkx-0.15.1 stevedore-5.4.0
symengine-0.13.0
```

[ ]:

```python
# Create circuit to test transpiler on
from qiskit import QuantumCircuit, transpile
from qiskit.circuit.library import GroverOperator, Diagonal

# Use Statevector object to calculate the ideal output
from qiskit.quantum_info import Statevector
from qiskit.visualization import plot_histogram

# Qiskit Runtime
```

[ ]:

```python
oracle = Diagonal([1] * 15 + [-1])
print(oracle)
qc = QuantumCircuit(4)
qc.h([0, 1,2,3])
qc = qc.compose(GroverOperator(oracle))

qc.draw()
```

```
q_0: 0

q_1: 1
       Diagonal(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1)
q_2: 2

q_3: 3
```

[ ]:

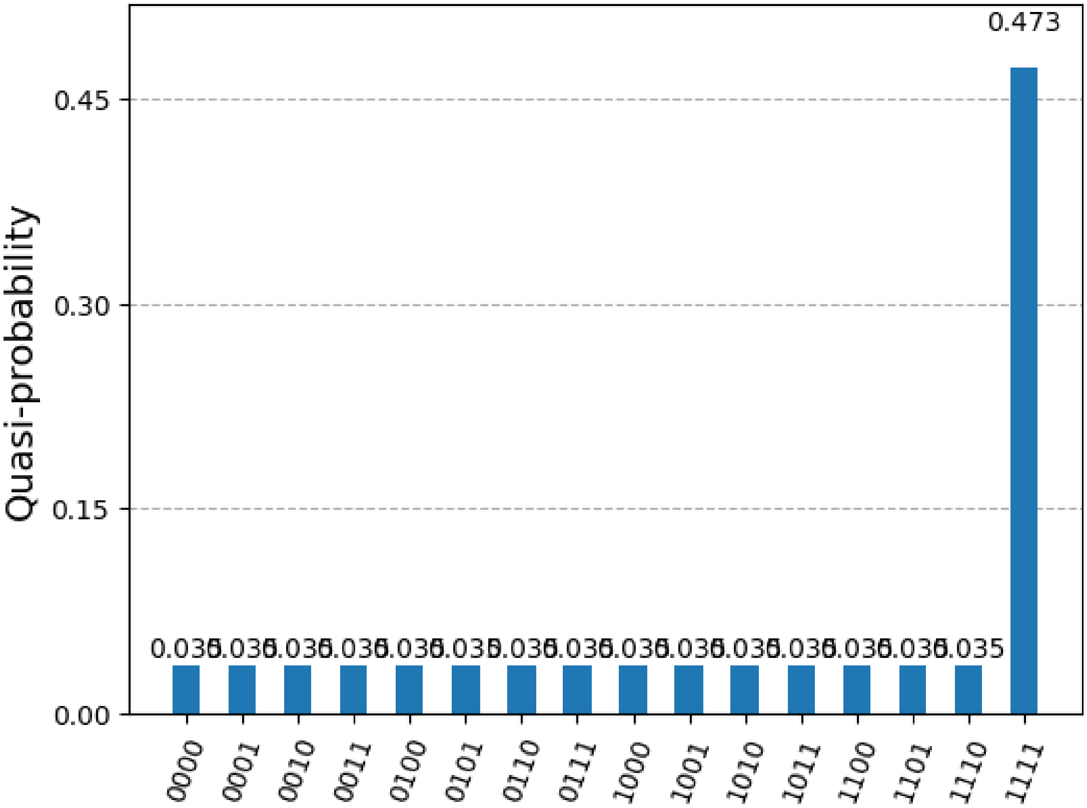
```
q_0:  H  0

q_1:  H  1
```

```
          Q
q_2:  H  2

q_3:  H  3
```

[ ]:

```
ideal_distribution = Statevector.from_instruction(qc).probabilities_dict()

plot_histogram(ideal_distribution)
```

[ ]:



[6]:

```
!pip install qiskit
```

```
Requirement already satisfied: qiskit in /usr/local/lib/python3.10/dist-packages
(1.3.1)
Requirement already satisfied: rustworkx>=0.15.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit) (0.15.1)
Requirement already satisfied: numpy<3,>=1.17 in /usr/local/lib/python3.10/dist-
packages (from qiskit) (1.26.4)
Requirement already satisfied: scipy>=1.5 in /usr/local/lib/python3.10/dist-
packages (from qiskit) (1.13.1)
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.10/dist-
```

```
packages (from qiskit) (1.13.1)
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit) (0.3.9)
Requirement already satisfied: python-dateutil>=2.8.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit) (2.8.2)
Requirement already satisfied: stevedore>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit) (5.4.0)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from qiskit) (4.12.2)
Requirement already satisfied: symengine<0.14,>=0.11 in
/usr/local/lib/python3.10/dist-packages (from qiskit) (0.13.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.0->qiskit) (1.17.0)
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from stevedore>=3.0.0->qiskit) (6.1.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy>=1.3->qiskit) (1.3.0)
```

[7]:
```
!pip install qiskit_algorithms
```

```
Collecting qiskit_algorithms
  Downloading qiskit_algorithms-0.3.1-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: qiskit>=0.44 in /usr/local/lib/python3.10/dist-
packages (from qiskit_algorithms) (1.3.1)
Requirement already satisfied: scipy>=1.4 in /usr/local/lib/python3.10/dist-
packages (from qiskit_algorithms) (1.13.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from qiskit_algorithms) (1.26.4)
Requirement already satisfied: rustworkx>=0.15.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=0.44->qiskit_algorithms)
(0.15.1)
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=0.44->qiskit_algorithms) (1.13.1)
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=0.44->qiskit_algorithms) (0.3.9)
Requirement already satisfied: python-dateutil>=2.8.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=0.44->qiskit_algorithms)
(2.8.2)
Requirement already satisfied: stevedore>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=0.44->qiskit_algorithms)
(5.4.0)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from qiskit>=0.44->qiskit_algorithms)
(4.12.2)
Requirement already satisfied: symengine<0.14,>=0.11 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=0.44->qiskit_algorithms)
(0.13.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
```

```
packages (from python-dateutil>=2.8.0->qiskit>=0.44->qiskit_algorithms) (1.17.0)
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from stevedore>=3.0.0->qiskit>=0.44->qiskit_algorithms) (6.1.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from
sympy>=1.3->qiskit>=0.44->qiskit_algorithms) (1.3.0)
Downloading qiskit_algorithms-0.3.1-py3-none-any.whl (310 kB)
                         310.5/310.5 kB
5.7 MB/s eta 0:00:00
Installing collected packages: qiskit_algorithms
Successfully installed qiskit_algorithms-0.3.1
```

[8]:

```
!pip install qiskit-ibm-runtime
```

```
Collecting qiskit-ibm-runtime
  Downloading qiskit_ibm_runtime-0.34.0-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: requests>=2.19 in /usr/local/lib/python3.10/dist-
packages (from qiskit-ibm-runtime) (2.32.3)
Requirement already satisfied: requests-ntlm>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit-ibm-runtime) (1.3.0)
Requirement already satisfied: numpy>=1.13 in /usr/local/lib/python3.10/dist-
packages (from qiskit-ibm-runtime) (1.26.4)
Requirement already satisfied: urllib3>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from qiskit-ibm-runtime) (2.2.3)
Requirement already satisfied: python-dateutil>=2.8.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit-ibm-runtime) (2.8.2)
Requirement already satisfied: websocket-client>=1.5.1 in
/usr/local/lib/python3.10/dist-packages (from qiskit-ibm-runtime) (1.8.0)
Collecting ibm-platform-services>=0.22.6 (from qiskit-ibm-runtime)
  Downloading ibm_platform_services-0.59.0-py3-none-any.whl.metadata (9.0 kB)
Collecting pydantic<2.10,>=2.5.0 (from qiskit-ibm-runtime)
  Downloading pydantic-2.9.2-py3-none-any.whl.metadata (149 kB)
                           149.4/149.4

kB 3.7 MB/s eta 0:00:00
Requirement already satisfied: qiskit>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit-ibm-runtime) (1.3.1)
Collecting ibm-cloud-sdk-core<4.0.0,>=3.22.0 (from ibm-platform-
services>=0.22.6->qiskit-ibm-runtime)
  Downloading ibm_cloud_sdk_core-3.22.0-py3-none-any.whl.metadata (8.6 kB)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic<2.10,>=2.5.0->qiskit-ibm-
runtime) (0.7.0)
Collecting pydantic-core==2.23.4 (from pydantic<2.10,>=2.5.0->qiskit-ibm-
runtime)
  Downloading pydantic_core-2.23.4-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.6 kB)
Requirement already satisfied: typing-extensions>=4.6.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from pydantic<2.10,>=2.5.0->qiskit-ibm-
runtime) (4.12.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.0->qiskit-ibm-runtime) (1.17.0)
Requirement already satisfied: rustworkx>=0.15.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-ibm-runtime)
(0.15.1)
Requirement already satisfied: scipy>=1.5 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=1.1.0->qiskit-ibm-runtime) (1.13.1)
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=1.1.0->qiskit-ibm-runtime) (1.13.1)
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=1.1.0->qiskit-ibm-runtime) (0.3.9)
Requirement already satisfied: stevedore>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-ibm-runtime)
(5.4.0)
Requirement already satisfied: symengine<0.14,>=0.11 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-ibm-runtime)
(0.13.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.19->qiskit-ibm-
runtime) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.19->qiskit-ibm-runtime) (3.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.19->qiskit-ibm-
runtime) (2024.12.14)
Requirement already satisfied: cryptography>=1.3 in
/usr/local/lib/python3.10/dist-packages (from requests-ntlm>=1.1.0->qiskit-ibm-
runtime) (43.0.3)
Requirement already satisfied: pyspnego>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from requests-ntlm>=1.1.0->qiskit-ibm-
runtime) (0.11.2)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.10/dist-
packages (from cryptography>=1.3->requests-ntlm>=1.1.0->qiskit-ibm-runtime)
(1.17.1)
Requirement already satisfied: PyJWT<3.0.0,>=2.8.0 in
/usr/local/lib/python3.10/dist-packages (from ibm-cloud-sdk-
core<4.0.0,>=3.22.0->ibm-platform-services>=0.22.6->qiskit-ibm-runtime) (2.10.1)
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from stevedore>=3.0.0->qiskit>=1.1.0->qiskit-ibm-runtime) (6.1.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy>=1.3->qiskit>=1.1.0->qiskit-
ibm-runtime) (1.3.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-
packages (from cffi>=1.12->cryptography>=1.3->requests-ntlm>=1.1.0->qiskit-ibm-
runtime) (2.22)
Downloading qiskit_ibm_runtime-0.34.0-py3-none-any.whl (3.0 MB)
```

```
                         3.0/3.0 MB
29.1 MB/s eta 0:00:00
Downloading ibm_platform_services-0.59.0-py3-none-any.whl (340 kB)
                         340.8/340.8 kB
18.8 MB/s eta 0:00:00
Downloading pydantic-2.9.2-py3-none-any.whl (434 kB)
                         434.9/434.9 kB
15.2 MB/s eta 0:00:00
Downloading
pydantic_core-2.23.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(2.1 MB)
                         2.1/2.1 MB
42.9 MB/s eta 0:00:00
Downloading ibm_cloud_sdk_core-3.22.0-py3-none-any.whl (69 kB)
                         69.4/69.4 kB
5.7 MB/s eta 0:00:00
Installing collected packages: pydantic-core, pydantic, ibm-cloud-sdk-
core, ibm-platform-services, qiskit-ibm-runtime
  Attempting uninstall: pydantic-core
    Found existing installation: pydantic_core 2.27.1
    Uninstalling pydantic_core-2.27.1:
      Successfully uninstalled pydantic_core-2.27.1
  Attempting uninstall: pydantic
    Found existing installation: pydantic 2.10.3
    Uninstalling pydantic-2.10.3:
      Successfully uninstalled pydantic-2.10.3
Successfully installed ibm-cloud-sdk-core-3.22.0 ibm-platform-services-0.59.0
pydantic-2.9.2 pydantic-core-2.23.4 qiskit-ibm-runtime-0.34.0
```

```
[ ]: !pip install pylatexenc
```

```
Collecting pylatexenc
  Downloading pylatexenc-2.10.tar.gz (162 kB)
                           0.0/162.6

kB ? eta -:--:--

153.6/162.6 kB 5.7 MB/s eta 0:00:01
                      162.6/162.6 kB
4.0 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Building wheels for collected packages: pylatexenc
  Building wheel for pylatexenc (setup.py) … done
  Created wheel for pylatexenc: filename=pylatexenc-2.10-py3-none-any.whl
size=136816
sha256=b07b6159efedeb81ec72a943bb9dc9e398ffae690902c3df752229bb333d1a72
  Stored in directory: /root/.cache/pip/wheels/d3/31/8b/e09b0386afd80cfc556c0040
8c9aeea5c35c4d484a9c762fd5
```

```
Successfully built pylatexenc
Installing collected packages: pylatexenc
Successfully installed pylatexenc-2.10
```

[ ]:

```
!pip install 'qiskit-machine-learning[sparse]'
```

```
Collecting qiskit-machine-learning[sparse]
  Downloading qiskit_machine_learning-0.8.2-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: qiskit>=1.0 in /usr/local/lib/python3.10/dist-
packages (from qiskit-machine-learning[sparse]) (1.3.1)
Requirement already satisfied: scipy>=1.4 in /usr/local/lib/python3.10/dist-
packages (from qiskit-machine-learning[sparse]) (1.13.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from qiskit-machine-learning[sparse]) (1.26.4)
Requirement already satisfied: psutil>=5 in /usr/local/lib/python3.10/dist-
packages (from qiskit-machine-learning[sparse]) (5.9.5)
Requirement already satisfied: scikit-learn>=1.2 in
/usr/local/lib/python3.10/dist-packages (from qiskit-machine-learning[sparse])
(1.6.0)
Requirement already satisfied: setuptools>=40.1 in
/usr/local/lib/python3.10/dist-packages (from qiskit-machine-learning[sparse])
(75.1.0)
Requirement already satisfied: dill>=0.3.4 in /usr/local/lib/python3.10/dist-
packages (from qiskit-machine-learning[sparse]) (0.3.9)
Collecting sparse (from qiskit-machine-learning[sparse])
  Downloading sparse-0.15.4-py2.py3-none-any.whl.metadata (4.5 kB)
Requirement already satisfied: rustworkx>=0.15.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.0->qiskit-machine-
learning[sparse]) (0.15.1)
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=1.0->qiskit-machine-learning[sparse]) (1.13.1)
Requirement already satisfied: python-dateutil>=2.8.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.0->qiskit-machine-
learning[sparse]) (2.8.2)
Requirement already satisfied: stevedore>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.0->qiskit-machine-
learning[sparse]) (5.4.0)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.0->qiskit-machine-
learning[sparse]) (4.12.2)
Requirement already satisfied: symengine<0.14,>=0.11 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.0->qiskit-machine-
learning[sparse]) (0.13.0)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=1.2->qiskit-machine-learning[sparse]) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.2->qiskit-machine-
learning[sparse]) (3.5.0)
```

```
Requirement already satisfied: numba>=0.49 in /usr/local/lib/python3.10/dist-
packages (from sparse->qiskit-machine-learning[sparse]) (0.60.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in
/usr/local/lib/python3.10/dist-packages (from numba>=0.49->sparse->qiskit-
machine-learning[sparse]) (0.43.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.0->qiskit>=1.0->qiskit-machine-
learning[sparse]) (1.17.0)
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from stevedore>=3.0.0->qiskit>=1.0->qiskit-machine-learning[sparse])
(6.1.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy>=1.3->qiskit>=1.0->qiskit-
machine-learning[sparse]) (1.3.0)
Downloading qiskit_machine_learning-0.8.2-py3-none-any.whl (231 kB)
                         231.6/231.6 kB
6.1 MB/s eta 0:00:00
Downloading sparse-0.15.4-py2.py3-none-any.whl (237 kB)
                         237.3/237.3 kB
17.6 MB/s eta 0:00:00
Installing collected packages: sparse, qiskit-machine-learning
Successfully installed qiskit-machine-learning-0.8.2 sparse-0.15.4
```

```
[ ]: !pip install qiskit-aer
```

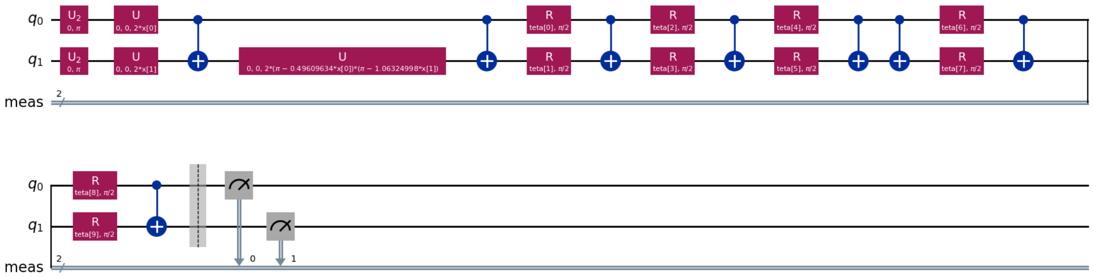
```
Collecting qiskit-aer
  Downloading qiskit_aer-0.15.1-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.0 kB)
Requirement already satisfied: qiskit>=1.1.0 in /usr/local/lib/python3.10/dist-
packages (from qiskit-aer) (1.3.1)
Requirement already satisfied: numpy>=1.16.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit-aer) (1.26.4)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.10/dist-
packages (from qiskit-aer) (1.13.1)
Requirement already satisfied: psutil>=5 in /usr/local/lib/python3.10/dist-
packages (from qiskit-aer) (5.9.5)
Requirement already satisfied: rustworkx>=0.15.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-aer)
(0.15.1)
Requirement already satisfied: sympy>=1.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=1.1.0->qiskit-aer) (1.13.1)
Requirement already satisfied: dill>=0.3 in /usr/local/lib/python3.10/dist-
packages (from qiskit>=1.1.0->qiskit-aer) (0.3.9)
Requirement already satisfied: python-dateutil>=2.8.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-aer) (2.8.2)
Requirement already satisfied: stevedore>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-aer) (5.4.0)
Requirement already satisfied: typing-extensions in
```

```
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-aer)
(4.12.2)
Requirement already satisfied: symengine<0.14,>=0.11 in
/usr/local/lib/python3.10/dist-packages (from qiskit>=1.1.0->qiskit-aer)
(0.13.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.0->qiskit>=1.1.0->qiskit-aer) (1.17.0)
Requirement already satisfied: pbr>=2.0.0 in /usr/local/lib/python3.10/dist-
packages (from stevedore>=3.0.0->qiskit>=1.1.0->qiskit-aer) (6.1.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy>=1.3->qiskit>=1.1.0->qiskit-
aer) (1.3.0)
Downloading
qiskit_aer-0.15.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(12.3 MB)
                         12.3/12.3 MB
77.9 MB/s eta 0:00:00
Installing collected packages: qiskit-aer
Successfully installed qiskit-aer-0.15.1
```

```
[ ]: import numpy as np
     from qiskit.circuit import ParameterVector
     from qiskit import QuantumCircuit
     import numpy as np

     from qiskit import QuantumCircuit

     # 10-parameter variational quantum model
     # define your parameters
     x = ParameterVector('x', 2)
     teta=ParameterVector('teta',10)
     # define your parameters
     def ZZFeatureMap_10_parametros(feature_dimension=2, reps=1, theta_param=np.pi/
      ↪2, x=x, teta=teta):
         circuit = QuantumCircuit(feature_dimension)
     #1.06324998,  0.49609634
         for i in range(reps):
             if i == 0:
                 circuit.h(range(feature_dimension))

             for j in range(feature_dimension):
                 circuit.p(theta_param * x[j] * 1, j)

             circuit.cx(0, 1)

             circuit.p(theta_param * (np.pi - x[0] *0.49609634) * (np.pi - x[1] *1.
      ↪06324998), 1)
```

```
        circuit.cx(0, 1)

        # Ensure that the indices are within the range of feature_dimension
        if feature_dimension > 1:
            circuit.ry(teta[0], 0)
            circuit.ry(teta[1], 1)

        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[2], 0)
            circuit.ry(teta[3], 1)
        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[4], 0)
            circuit.ry(teta[5], 1)
            circuit.cx(0, 1)
        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[6], 0)
            circuit.ry(teta[7], 1)
            circuit.cx(0, 1)
            circuit.ry(teta[8], 0)
            circuit.ry(teta[9], 1)
            circuit.cx(0, 1)
            circuit.measure_all()

    return circuit

feature_map = ZZFeatureMap_10_parametros(feature_dimension=2, reps=1,␣
 ↪theta_param=2, x=x,teta=teta)
#print(feature_map)


feature_map.decompose().draw("mpl")
```

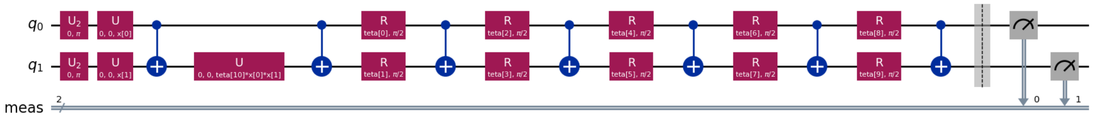
[ ]:



[ ]:

```
#FakeProviderForBackendV2()        Fake provider containing fake V2 backends.
#FakeProvider()        Fake provider containing fake V1 backends.
# https://docs.quantum.ibm.com/api/qiskit-ibm-runtime/fake_provider

from qiskit import QuantumCircuit
from qiskit import transpile
from qiskit.visualization import plot_histogram
from qiskit_ibm_runtime.fake_provider import␣
 ↪FakeManilaV2,FakeBrisbane,FakeKyoto,FakeOsaka

#FakeKyoto()        A fake 127 qubit backend.
  #FakeMontrealV2()        A fake 27 qubit backend.
# FakeMumbaiV2()        A fake 27 qubit backend.
#FakeNairobiV2()        A fake 7 qubit backend.
#FakeOsaka()        A fake 127 qubit backend.
# FakeOslo()        A fake 7 qubit backend.
import numpy as np
from qiskit.circuit import ParameterVector

import numpy as np
# 11-parameter variational quantum model
# define your parameters
x = ParameterVector('x', 2)
teta=ParameterVector('teta',11)
# define your parameters
def ZZFeatureMap_11_parametros_MODI(feature_dimension=2, reps=1, x=x,␣
 ↪teta=teta):
    circuit = QuantumCircuit(feature_dimension)
#1.06324998,  0.49609634
#0.6810406 ,  0.93371965
#opt_var7=np.array([0.68958857, 0.93659037])#0.765
    for i in range(reps):
        if i == 0:
            circuit.h(range(feature_dimension))

        for j in range(feature_dimension):
            circuit.p( x[j] * 1, j)

        circuit.cx(0, 1)

        circuit.p( (x[0]*teta[10]) * (x[1]), 1)

        circuit.cx(0, 1)

        # Ensure that the indices are within the range of feature_dimension
        if feature_dimension > 1:
            circuit.ry(teta[0], 0)
            circuit.ry(teta[1], 1)

        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[2], 0)
            circuit.ry(teta[3], 1)
        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[4], 0)
            circuit.ry(teta[5], 1)
            circuit.cx(0, 1)
        if feature_dimension > 1:

            circuit.ry(teta[6], 0)
            circuit.ry(teta[7], 1)
            circuit.cx(0, 1)
            circuit.ry(teta[8], 0)
            circuit.ry(teta[9], 1)
            circuit.cx(0, 1)
            circuit.measure_all()

    return circuit

feature_map = ZZFeatureMap_11_parametros_MODI(feature_dimension=2, reps=1,␣
 ↪x=x,teta=teta)
#print(feature_map)
feature_map.decompose().draw("mpl")
```

[ ]:



[10]:

```
import qiskit_algorithms
from qiskit_algorithms.optimizers import SPSA,COBYLA

from qiskit import QuantumCircuit

from qiskit.circuit.library import ZZFeatureMap, RealAmplitudes
from qiskit.quantum_info import Statevector

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle

import warnings
warnings.filterwarnings("ignore")




import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle

import warnings
warnings.filterwarnings("ignore")
# constants
n = 2
RANDOM_STATE = 42
LR = 1e-3
class_labels = ['0', '1']


# Normalizing data from 0 to 2pi

def normalizeData(DATA_PATH = "./FEATURE_RESULTS/FEATURE_resultante_DP_NODP.
 ↪csv"):
    """
    Normalizes the data
    """
    # Reads the data mean_coords_x,  mean_coords_y  centroid_y_roi,␣
 ↪mean_coords_x
    data = pd.read_csv(DATA_PATH)
    data = shuffle(data, random_state=RANDOM_STATE)
    X, Y = data[['area_pixels', ' mean_coords_x']].values, data[' class'].values
    #X, Y = data[[' mean_coords_x', ' centroid_y_roi']].values, data[' class'].
 ↪values
    #X, Y = data[[' mean_coords_x', ' mean_coords_y']].values, data[' class'].
 ↪values
    # normalize the data
    scaler = MinMaxScaler(feature_range=(-0 * np.pi, 2 * np.pi))
    X = scaler.fit_transform(X)
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=RANDOM_STATE)
    return X_train, X_test, Y_train, Y_test





import qiskit_algorithms
from qiskit_algorithms.optimizers import SPSA

from qiskit import QuantumCircuit

from qiskit.circuit.library import ZZFeatureMap, RealAmplitudes
from qiskit.quantum_info import Statevector

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.utils import shuffle

import warnings
warnings.filterwarnings("ignore")

TRAIN_DATA, TEST_DATA, TRAIN_LABELS, TEST_LABELS = normalizeData()
# Replace all occurrences of 2 with 1
TRAIN_LABELS = np.where(TRAIN_LABELS == 2, 0, TRAIN_LABELS)
TEST_LABELS = np.where(TEST_LABELS == 2, 0, TEST_LABELS)
print(TRAIN_LABELS)
#
# Count the number of elements that are 0
num_zeros = np.count_nonzero(TRAIN_LABELS == 0)

# Count the number of elements that are  1
num_unos = np.count_nonzero(TRAIN_LABELS == 1)



print("Count the number of elements that are 0 :", num_zeros)
print(" Count the number of elements that are 1:", num_unos)
```

```
[1 0 1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0
 1 0 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 1 1 0 1 0 1 1 0 0 1 0 0 1 1 0 1 0 1 1 0
 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 0 1 1 0 1 0
 0 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 1 1 0 0 0 1 0 1 0 1
 1 0 1 0 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1 1
 0 0 1 0 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1 0 0
 0 1 0 1 0 0 0 0 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 1
 1 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 0 1 1 0 0 1 1 1 1 1
 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0
 0 0 1 0 0 1 1 0 1 0 0 1 1 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0
 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1 1 0 1 1 0 0 0 1 0 1 0 0 0 0 0 1 0 1
 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0
 1 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 1 0 0 1
 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1
 1 0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 1 0
 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 0 0 1 0
 1 0 0 0 0 0 1 1 0 1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 0 1 0
 1 1 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0
 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0]
Count the number of elements that are 0 : 403
 Count the number of elements that are 1: 290
```

[ ]:

```
TEST_LABELS
```

[ ]:

```
array([1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1])
```

[2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:

```
import numpy as np
from qiskit.circuit import ParameterVector
from qiskit import QuantumCircuit
import numpy as np
#from qiskit import BasicAer, execute

from qiskit_ibm_runtime import QiskitRuntimeService

# define your parameters
x = ParameterVector('x', 2)
teta=ParameterVector('teta',11)
```

```
teta10=1.52596629


#Functions used in the optimization process of circuit parameters

def ZZFeatureMap_11_parametros_MODI(feature_dimension=2, reps=1, x=x,␣
 ↪teta=teta):
    circuit = QuantumCircuit(feature_dimension)
#1.06324998,  0.49609634
#0.6810406 ,  0.93371965
#opt_var7=np.array([0.68958857, 0.93659037])#0.765
    for i in range(reps):
        if i == 0:
            circuit.h(range(feature_dimension))

        for j in range(feature_dimension):
            circuit.p( x[j] * 1, j)

        circuit.cx(0, 1)

        circuit.p( (x[0]*teta[10]) * (x[1]), 1)

        circuit.cx(0, 1)

        # Ensure that the indices are within the range of feature_dimension
        if feature_dimension > 1:
            circuit.ry(teta[0], 0)
            circuit.ry(teta[1], 1)

        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[2], 0)
            circuit.ry(teta[3], 1)
        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[4], 0)
            circuit.ry(teta[5], 1)
            circuit.cx(0, 1)
        if feature_dimension > 1:

            circuit.ry(teta[6], 0)
            circuit.ry(teta[7], 1)
            circuit.cx(0, 1)
            circuit.ry(teta[8], 0)
            circuit.ry(teta[9], 1)
            circuit.cx(0, 1)
            circuit.measure_all()
    return circuit
feature_map = ZZFeatureMap_11_parametros_MODI(feature_dimension=2, reps=1,␣
 ↪x=x,teta=teta)
print(feature_map)
feature_map.draw('mpl')



def classification_probability(data, variational):
    """Classify data points using given parameters.
    Args:
        data (list): Set of data points to classify
        variational (list): Parameters for `VAR_FORM`
    Returns:
    #results = execute(circuits, backend, shots=shots_per_execution).result()
    job = backend.run(circuits,shots=4000)
    #counts = job.result().get_counts()
    #######33
    results = job.result()
        list[dict]: Probability of circuit classifying
                    each data point as 0 or 1.
    shots_per_execution = 100"""
    shots_per_execution = 50
    circuits = [circuit_instance_MIO(d, variational) for d in data]
    #backend = FakeManilaV2()
    backend = FakeOsaka()
    #results = execute(circuits, backend, shots=shots_per_execution).result()
    #results = execute(circuits, backend, shots=shots_per_execution).result()
    job = backend.run(circuits,shots=100)
#counts = job.result().get_counts()
    classification = [
        label_probability(job.result().get_counts(c)) for c in circuits]
    return classification

def parity(bitstring):
    """Returns 1 if parity of `bitstring` is even, otherwise 0."""
    hamming_weight = sum(int(k) for k in list(bitstring))
    return (hamming_weight+1) % 2
def label_probability(results):
    """Converts a dict of bitstrings and their counts,
    to parities and their counts"""
    shots = sum(results.values())
    probabilities = {0: 0, 1: 0}
    for bitstring, counts in results.items():
        label = parity(bitstring)
        probabilities[label] += counts / shots
    return probabilities

def cross_entropy_loss(classification, expected):

    p = classification.get(expected)  # Prob. of correct classification
    return -np.log(p + 1e-10)



def cost_function(data, labels, variational):
    """Evaluates performance of our circuit with `variational`
    parameters on `data`.

    Args:
        data (list): List of data points to classify
        labels (list): List of correct labels for each data point
        variational (list): Parameters to use in circuit

    Returns:
        float: Cost (metric of performance)
    """
    classifications = classification_probability(data, variational)
    cost = 0
    for i, classification in enumerate(classifications):
        cost += cross_entropy_loss(classification, labels[i])
    cost /= len(data)
    return cost

import time








class OptimizerLog:
    """Log to store optimizer's intermediate results"""
    def __init__(self):
        self.evaluations = []
        self.parameters = []
        self.costs = []

    def update(self, parameter):
        """Save intermediate results."""
```

```python
        evaluation = len(self.evaluations) + 1  # Assuming evaluations are sequential
        cost = objective_function(parameter)
        self.evaluations.append(evaluation)
        self.parameters.append(parameter)
        self.costs.append(cost)




def marcar_probabilidades_altas(probabilidades, umbral=0.91):
    INDICE_CLASE_0 = []
    INDICE_CLASE_1 = []

    for idx, prob_dict in enumerate(probabilidades):
        # Obtener la clase con la probabilidad más alta
        clase_elegida = max(prob_dict, key=prob_dict.get)

        # Marcar el elemento si la probabilidad es mayor que el umbral
        if prob_dict[clase_elegida] > umbral:
            if clase_elegida == 0:
                INDICE_CLASE_0.append(idx)
            elif clase_elegida == 1:
                INDICE_CLASE_1.append(idx)

    return INDICE_CLASE_0, INDICE_CLASE_1
```

```
                                                                 »
   q_0:   H   P(x[0])                      Ry(teta[0])    »
                                               »
   q_1:   H   P(x[1])   X   P(teta[10]*x[0]*x[1])   X   Ry(teta[1])   X  »
                                               »
meas: 2/                                       »
                                                                                »
«                                                »
«   q_0:   Ry(teta[2])     Ry(teta[4])      Ry(teta[6])     »
«                                        »
«   q_1:   Ry(teta[3])   X   Ry(teta[5])   X   Ry(teta[7])   X  »
«                                        »
«meas: 2/                                »
«                                                                     »
«
«   q_0:   Ry(teta[8])      M
«
«   q_1:   Ry(teta[9])   X     M
«
```

```
«meas: 2/
«                                 0  1
```

[ ]:

```
# Set up the optimizationfrom qiskit_algorithms.optimizers import SPSA ␣
 ↪optimizer = COBYLA(maxiter=40)
#from qiskit.algorithms.optimizers import
import qiskit_algorithms
from qiskit_algorithms.optimizers import SPSA,COBYLA,ADAM
log = OptimizerLog()
#optimizer = SPSA(maxiter=200,callback=callback_graph)

bounds = [(0, 3), (0, 2)]
optimizer = SPSA(maxiter=100)
#optimizer = SPSA(maxiter=50, learning_rate=0.05, perturbation=0.05)

#optimizer=COBYLA(maxiter=40 rhobeg=2.0,callback=log.update)
optimizer = COBYLA(maxiter=100,callback=log.update)
#optimizer =ADAM(maxiter=5)
#optimizer = SPSA(maxiter=50)
def circuit_instance_MIO(data, variational):
    """Assigns parameter values to `AD_HOC_CIRCUIT`.
    Args:
        data (list): Data values for the feature map
        variational (list): Parameter values for `VAR_FORM`
    Returns:
        QuantumCircuit: `AD_HOC_CIRCUIT` with parameters assigned
    """

    return feature_map.assign_parameters({x:data,teta:variational})


# It is uncommented according to the circuit used 10 or 11 parameters

#feature_map =ZZFeatureMap_11_parametros_MODI(feature_dimension=2, reps=1,␣
 ↪x=x,teta=teta)

feature_map = ZZFeatureMap_10_parametros(feature_dimension=2, reps=1,␣
 ↪theta_param=2, x=x,teta=teta)
```

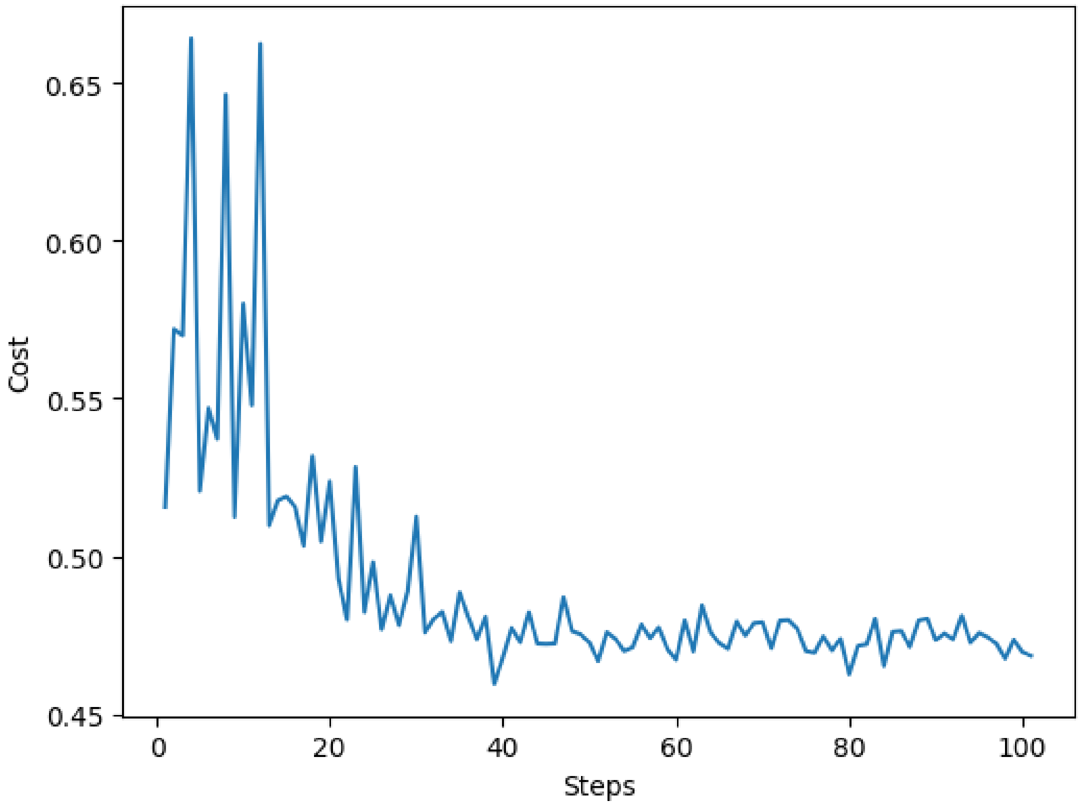
```
# optimal value obtained WITH A MEAN P OF 0.9 for DP_NODP
opt_var=np.array([-1.36465941,  0.72901008,  0.46274449, -0.22550087, 0.
 ↪71628267, -5.0369175 ,  0.25267942,  3.20192607,  2.22427876,  2.87675972])




def objective_function(variational):
    """Cost function of circuit parameters on training data.
    The optimizer will attempt to minimize this."""
    return cost_function(TRAIN_DATA# clear objective value history

objective_func_vals = []
# Run the optimization
bounds = [(0, 1.1), (0, 1.1)]


# Record the start time


inicio_tiempo = time.time()

# PARAA 10 PARA
opt_var=np.array([-0.28500921,  3.49039855,  1.85757059,  0.09167011, -3.
 ↪19778968,
       -6.05658208,  2.21860378,  5.5063756 ,  4.54796396,  2.70080547,
        1.55475791])

initial_point_FIN=np.array([-1.36465941,  0.72901008,  0.46274449, -0.22550087,␣
 ↪0.71628267, -5.0369175 ,  0.25267942,  3.20192607,  2.22427876,  2.87675972])


# for 10 PARAM. intial value
opt_var=np.array([-0.28500921,  3.49039855,  1.85757059,  0.09167011, -3.
 ↪19778968,
       -6.05658208,  2.21860378,  5.5063756 ,  4.54796396,  2.70080547,
        1.55475791])

initial_point=opt_var
result = optimizer.minimize(objective_function, initial_point)

# Record the completion time
fin_tiempo = time.time()

# Calculate the time difference
tiempo_ejecucion = fin_tiempo - inicio_tiempo
```

```
# Print the execution time
print(f"Time of the execution: {tiempo_ejecucion} s")



#final param for DP_NODP
feature_map = ZZFeatureMap_10_parametros(feature_dimension=2, reps=1,␣
 ↪theta_param=2, x=x,teta=teta)
opt_var_10parametros=np.array([-1.36465941,  0.72901008,  0.46274449, -0.
 ↪22550087, 0.71628267, -5.0369175 ,  0.25267942,  3.20192607,  2.22427876,  2.
 ↪87675972])


print(result)
opt_var=result.x
```

```
Tiempo de ejecución: 1679.867520570755 segundos
{   'fun': 0.4633378087513245,
    'jac': None,
    'nfev': 100,
    'nit': None,
    'njev': None,
    'x': array([ 0.88107158,  5.34857287, -2.68008328, -3.0875298 , -6.47556594,
       -0.99679796, -0.98433294,  8.79611478,  4.15260632,  7.95524071,
        1.46057183])}
```

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np

# Suppose you have a 'log' object with attributes 'evaluations' and 'costs'
# log = ... 'evaluations' y 'costs'
# log = ...

# Crear la figura
fig = plt.figure()

# Graficar los datos
plt.plot(log.evaluations, log.costs)
plt.xlabel('Steps')
plt.ylabel('Cost')
```

[ ]:
```
Text(0, 0.5, 'Cost')
```



[11]:
```
import numpy as np
from qiskit.circuit import ParameterVector
from qiskit import QuantumCircuit
import numpy as np

from qiskit import QuantumCircuit
# define your parameters
x = ParameterVector('x', 2)
teta=ParameterVector('teta',10)
# define your parameters
def ZZFeatureMap_10_parametros(feature_dimension=2, reps=1, theta_param=np.pi/
 ↪2, x=x, teta=teta):
    circuit = QuantumCircuit(feature_dimension)
#1.06324998,  0.49609634
    for i in range(reps):
        if i == 0:
            circuit.h(range(feature_dimension))

        for j in range(feature_dimension):
            circuit.p(theta_param * x[j] * 1, j)
        circuit.cx(0, 1)

        circuit.p(theta_param * (np.pi - x[0] *0.49609634) * (np.pi - x[1] *1.
 ↪06324998), 1)

        circuit.cx(0, 1)

        # Ensure that the indices are within the range of feature_dimension
        if feature_dimension > 1:
            circuit.ry(teta[0], 0)
            circuit.ry(teta[1], 1)

        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[2], 0)
            circuit.ry(teta[3], 1)
        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[4], 0)
            circuit.ry(teta[5], 1)
            circuit.cx(0, 1)
        if feature_dimension > 1:
            circuit.cx(0, 1)
            circuit.ry(teta[6], 0)
            circuit.ry(teta[7], 1)
            circuit.cx(0, 1)
            circuit.ry(teta[8], 0)
            circuit.ry(teta[9], 1)
            circuit.cx(0, 1)
            circuit.measure_all()

    return circuit

feature_map = ZZFeatureMap_10_parametros(feature_dimension=2, reps=1,␣
 ↪theta_param=2, x=x,teta=teta)
#print(feature_map)

feature_map.decompose().draw()

from qiskit import QuantumCircuit
from qiskit import transpile
from qiskit.visualization import plot_histogram
from qiskit_ibm_runtime.fake_provider import␣
 ↪FakeManilaV2,FakeBrisbane,FakeKyoto,FakeOsaka
backend = FakeOsaka()
```

```python
def circuit_instance_MIO(data, variational):
    """Assigns parameter values to `AD_HOC_CIRCUIT`.
    Args:
        data (list): Data values for the feature map
        variational (list): Parameter values for `VAR_FORM`
    Returns:
        QuantumCircuit: `AD_HOC_CIRCUIT` with parameters assigned
    """

    return feature_map.assign_parameters({x:data,teta:variational})





def classification_probability(data, variational):
    """Classify data points using given parameters.
    Args:
        data (list): Set of data points to classify
        variational (list): Parameters for `VAR_FORM`
    Returns:
    #results = execute(circuits, backend, shots=shots_per_execution).result()
    job = backend.run(circuits,shots=4000)
    #counts = job.result().get_counts()
    #######33
    results = job.result()
        list[dict]: Probability of circuit classifying
                    each data point as 0 or 1.
    shots_per_execution = 100"""
    shots = 2000
    circuits = [circuit_instance_MIO(d, variational) for d in data]
    #backend = FakeManilaV2()
    backend = FakeOsaka()
    #results = execute(circuits, backend, shots=shots_per_execution).result()
    #results = execute(circuits, backend, shots=shots_per_execution).result()
    job = backend.run(circuits,shots=shots)
#counts = job.result().get_counts()
    classification = [
        label_probability(job.result().get_counts(c)) for c in circuits]
    return classification

def parity(bitstring):
    """Returns 1 if parity of `bitstring` is even, otherwise 0."""
    hamming_weight = sum(int(k) for k in list(bitstring))
    return (hamming_weight+1) % 2
def label_probability(results):
    """Converts a dict of bitstrings and their counts,
    to parities and their counts"""
    shots = sum(results.values())
    probabilities = {0: 0, 1: 0}
    for bitstring, counts in results.items():
        label = parity(bitstring)
        probabilities[label] += counts / shots
    return probabilities

def cross_entropy_loss(classification, expected):
    """Calculate accuracy of predictions using cross entropy loss.
    Args:
        classification (dict): Dict where keys are possible classes,
                               and values are the probability our
                               circuit chooses that class.
        expected (int): Correct classification of the data point.

    Returns:
        float: Cross entropy loss
    """
    p = classification.get(expected)  # Prob. of correct classification
    return -np.log(p + 1e-10)







def comparar_probabilidades(probabilidades, labels):
    if len(probabilidades) != len(labels):
        raise ValueError("Lists of probabilities and labels must be the same length.")

    coincidencias = 0

    for prob_dict, label in zip(probabilidades, labels):
        # Get the class with the highest probability
        clase_predicha = max(prob_dict, key=prob_dict.get)

        # Check if the predicted class matches the actual label
        if clase_predicha == label:
            coincidencias += 1

    probabilidad_media = coincidencias / len(labels)
    return probabilidad_media
```

```
feature_map = ZZFeatureMap_10_parametros(feature_dimension=2, reps=1,␣
 ↪theta_param=2, x=x,teta=teta)

opt_var=np.array([-1.36465941,  0.72901008,  0.46274449, -0.22550087, 0.
 ↪71628267, -5.0369175 ,  0.25267942,  3.20192607,  2.22427876,  2.87675972])

""" In this final part we can calculate the accuracy with this set of␣
 ↪parameters after the optimization process.
"""
resultado =␣
 ↪comparar_probabilidades(classification_probability(TRAIN_DATA,opt_var),␣
 ↪TRAIN_LABELS)

print(f"The average probability of matching is for train : {resultado}")

resultado =␣
 ↪comparar_probabilidades(classification_probability(TEST_DATA,opt_var),␣
 ↪TEST_LABELS)
print(f"The average probability of matching is for test : {resultado}")
```

```
The average probability of matching is for train : 0.8903318903318903
The average probability of matching is for test : 0.9195402298850575
```