# PYTHON (QISKIT) CODES
# FOR
# QUANTUM MACHINE LEARNING

Kumar Ghosh

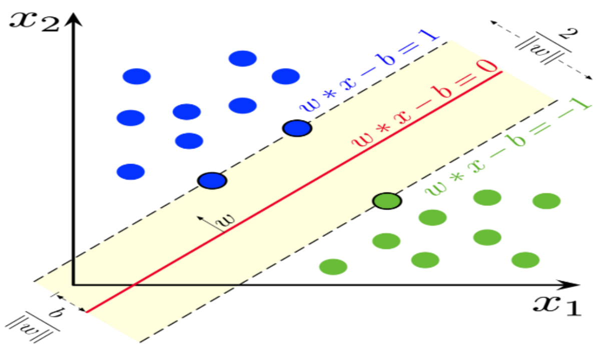
# Quantum machine learning

Quantum computation is a branch of physics and computer science which solves different computational problems by using quantum-mechanical phenomena, for e.g. entanglement and superposition. Quantum computation and quantum information theory is a centre of attention in last few decades because it can outperform classical computation and information processing, because the quantum algorithms can give rise to exponential speedups over their classical counterparts. Many known quantum algorithms have a diverse application, such as: integer factorization, search algorithm solving constraint satisfaction problems, and quantum machine learning. Quantum machine learning is an emerging interdisciplinary research area at the intersection of quantum physics and machine learning, where machine learning algorithms for the analysis of classical data is executed on a quantum computer, for obtaining quantum generalizations of classical machine learning algorithms, which will provide possible speed-ups and other improvements over the existing classical learning models. In the following I am describing the quantum version of a partiular kind of mahine learning algorithm, namely: quanutm support vector machine.

# Support vector machine (SVM):

Support vector machines (SVM) are a class of supervised machine learning algorithms for binary classifications. It can be used for both classification and regression challenges. However, it is mostly used in classification problems for e.g. image classification.

Consider a set of M data points $\{(\vec{x_j}, y_j)\text{: } j=1, 2 \ldots M\}$, where each data point $\vec{x_j}$ is an $N$ dimensional vector and $y_j$ is the label of the data, which is $+1$ or -1. SVM finds the hyper plane $\vec{\omega} \cdot \vec{x} + b = 0$, which separates the whole data sets into two categories. In the following there is a schematic diagram of SVM where two kind of data are described by the blue and red dots respectively.



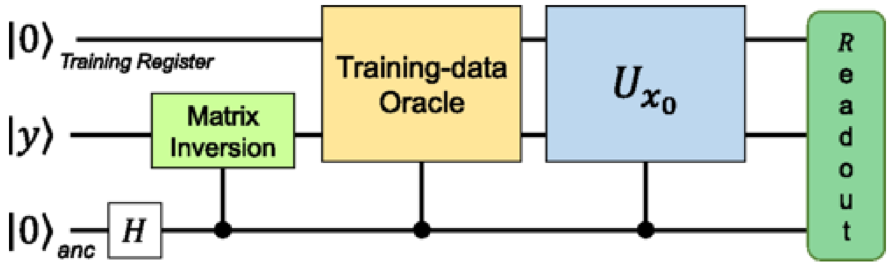

A version of SVM is called a Least squares SVM (LS-SVM) which approximates the hyper plane finding procedure of SVM by solving the following linear equation:

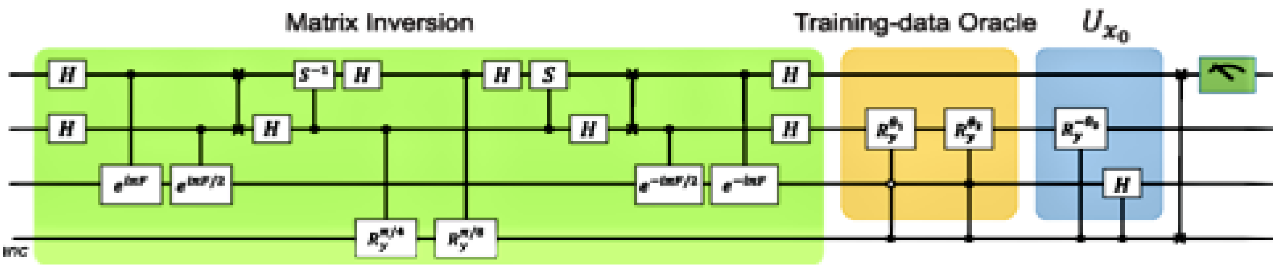
$$F \begin{bmatrix} b \\ \vec{\alpha} \end{bmatrix} \equiv \begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & K + \gamma.1 \end{bmatrix} \begin{bmatrix} b \\ \vec{\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{y} \end{bmatrix},$$

where $\boldsymbol{K_{ij}}$= $\vec{x}_i^{\,T} \cdot \vec{x_j}$ is the symmetric kernel matrix, $\vec{\boldsymbol{y}} = (y_1, \ldots, y_M)^T$, $\vec{\mathbf{1}} = (1, \ldots, 1)^T$, $\boldsymbol{\gamma}$ is the tuning parameter, and $(\boldsymbol{b}, \vec{\boldsymbol{\alpha}})$ are the parameters to determine the equation of the support vector plane.

## Quantum Support vector machines:

Quantum version of SVM performs the LS-SVM algorithm using quantum computers. It calculates the linear kernel-matrix using the quantum algorithm for inner product on quantum random access memory (QRAM), solves the linear equation using a quantum algorithm for solving linear equations, and perform the classification of a query data using the trained qubits with a quantum algorithm. Below we give a schematic diagram to implement QSVM and a matrix inversion circuit respectively.




In the above diagram the matrix inversion is employed to obtain the hyperplane parameters $(\boldsymbol{b}, \vec{\boldsymbol{\alpha}})$. The training data oracle can be implemented by preparing a desired quantum state through controlled rotation described before.

The overall complexity of the quantum SVM is $\mathrm{O}\,(\log\,(\mathrm{NM}))$, whereas classical complexity of the LS-SVM is $\mathrm{O}\,(M^2\,(\mathrm{M{+}N}))$. So we get exponential speed up in this procedure.

Although theoretically very efficient, the downside of the quantum SVM algoritm is involving Hamiltonian simulation, which itself is very complex. We can see the above quantum circuit involves lots of quantum gates just to calculate inverse of a simple (2x2) matrix. One of the open problem is to successfully construct a quanum Support Vector Machine for non-linear kernel to clasify more complex data.

The main computational part of this algorithm involves the matrix inversion. In the following we present the quantum circuit for computing $|X\rangle = A^{-1}\,|B\rangle$, with A= $\begin{bmatrix} \frac{5}{4} & -\frac{\sqrt{3}}{4} \\ -\frac{\sqrt{3}}{4} & \frac{7}{4} \end{bmatrix}$ and $|B\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. In the end of calculation, we obtain the Solution for the normalized vector $|x\rangle = \frac{|X\rangle}{|\,|X\rangle|}$.

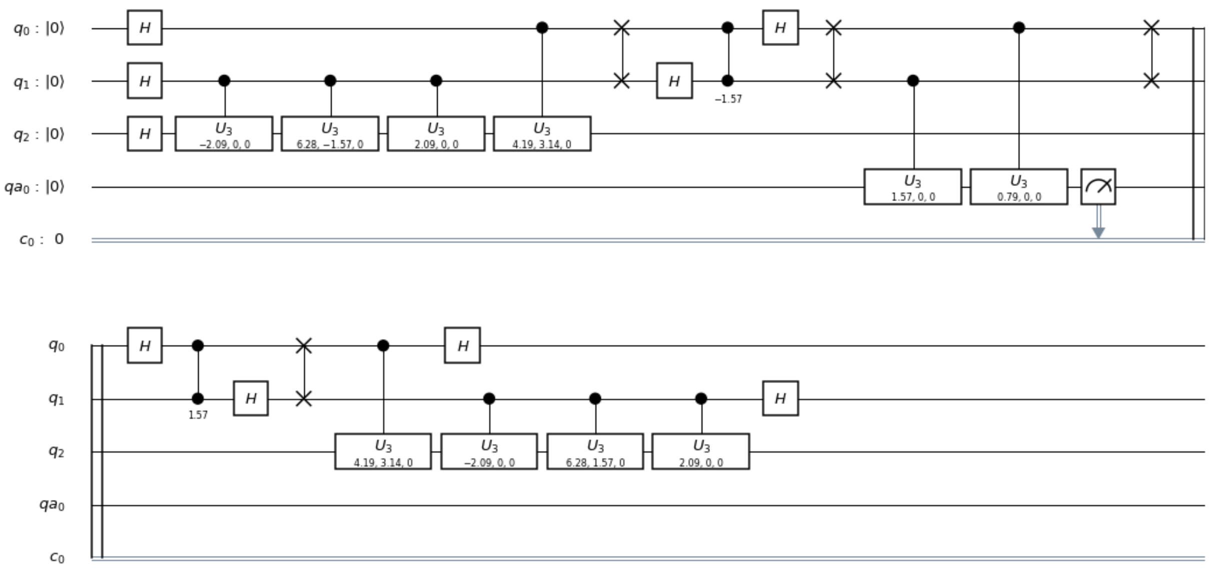
After measurement the theoretical result for obatining the states $|0\rangle$ and $|1\rangle$ are 0.62 and 0.38. respectively. Whereas, we simulate the quantum circuit through Qiskit from IBM-Q (qasm simulator) and obtained the results 0.58 and 0.42 respectively.

## ***Qiskit python code and quantum circuit:***

In the following I am attaching the Qiskit python code and quantum circuit

```
# Useful additional packages
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
from math import pi
from qiskit import QuantumCircuit, ClassicalRegister, QuantumRegister, exe
cute
from qiskit.tools.visualization import circuit_drawer
from qiskit.quantum_info import state_fidelity
from qiskit import BasicAer

backend = BasicAer.get_backend('unitary_simulator')

q= QuantumRegister(3, 'q')
q_a = QuantumRegister(1, name='qa')
c= ClassicalRegister(1, 'c')
circ= QuantumCircuit(q, q_a, c)
#circt= QuantumCircuit(q, c)
circ.h(q[0])
circ.h(q[1])
circ.h(q[2])
circ.cu3(-2*(pi/3), 0, 0, q[1], q[2])
circ.cu3(2*(pi), -(pi/2), 0, q[1], q[2])
circ.cu3(2*(pi/3), 0, 0, q[1], q[2])
circ.cu3(4*(pi/3), pi, 0, q[0], q[2])

circ.swap(q[0], q[1])
circ.h(q[1])
circ.cu1(-(pi/2), q[0], q[1])
circ.h(q[0])

circ.swap(q[0], q[1])
circ.cu3(pi/2,0,0,q[1],q_a[0])
circ.cu3(pi/4,0,0,q[0],q_a[0])
circ.swap(q[0], q[1])

circ.h(q[0])
circ.cu1(pi/2, q[1], q[0])
circ.h(q[1])
circ.swap(q[0], q[1])
```

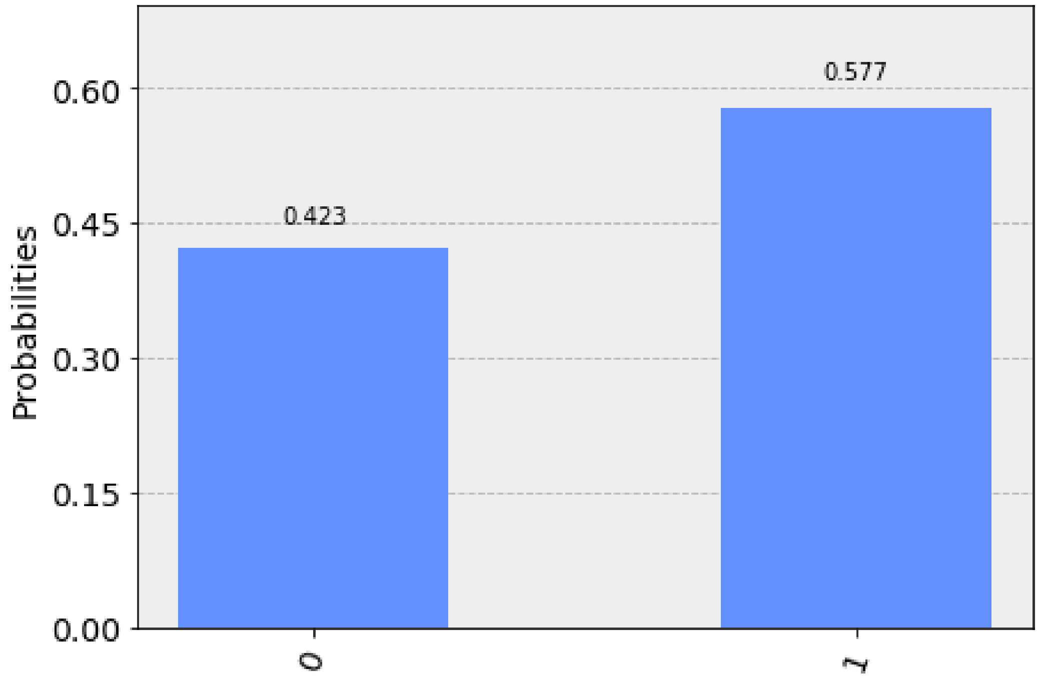
```
circ.cu3(4*(pi/3), pi, 0, q[0], q[2])
circ.cu3(-2*(pi/3), 0, 0, q[1], q[2])
circ.cu3(2*(pi), (pi/2), 0, q[1], q[2])
circ.cu3(2*(pi/3), 0, 0, q[1], q[2])

circ.h(q[0])
circ.h(q[1])

circuit_measure = circ.measure(q_a[0], c[0])

circ.draw(output='mpl')

##########################################################
```




```
#circuit_measure1 = circt.measure(q[0],  c[1])
#circuit_measure2 = circt.measure(q[1],  c[2])

backend_sim = BasicAer.get_backend('qasm_simulator')

# Execute the circuit on the qasm simulator.
# We've set the number of repeats of the circuit
# to be 1024, which is the default.
job_sim = execute(circ, backend_sim, shots=10000)

# Grab the results from the job.
result_sim = job_sim.result()

counts = result_sim.get_counts(circ)
print(counts)
#circ.draw(output='mpl')
```

```python
from qiskit.tools.visualization import plot_histogram
plot_histogram(counts)
```

```
{'0': 4228, '1': 5772}
```



# HHL algorithm for 8x8 matrix

In [1]:
```
from qiskit import Aer
from qiskit.circuit.library import QFT
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.quantum_info import state_fidelity
from qiskit.aqua.algorithms import HHL, NumPyLSsolver
from qiskit.aqua.components.eigs import EigsQPE
from qiskit.aqua.components.reciprocals import LookupRotation
from qiskit.aqua.operators import MatrixOperator
from qiskit.aqua.components.initial_states import Custom
import numpy as np
```

In [2]:
```
def create_eigs(matrix, num_ancillae, num_time_slices, negative_evals):
    ne_qfts = [None, None]
    if negative_evals:
        num_ancillae += 1
        ne_qfts = [QFT(num_ancillae - 1), QFT(num_ancillae - 1).inverse()]

    return EigsQPE(MatrixOperator(matrix=matrix),
                   QFT(num_ancillae).inverse(),
                   num_time_slices=num_time_slices,
                   num_ancillae=num_ancillae,
                   expansion_mode='suzuki',
                   expansion_order=2,
                   evo_time=None,
                   negative_evals=negative_evals,
                   ne_qfts=ne_qfts)

def fidelity(hhl, ref):
    solution_hhl_normed = hhl / np.linalg.norm(hhl)
    solution_ref_normed = ref / np.linalg.norm(ref)
    fidelity = state_fidelity(solution_hhl_normed, solution_ref_normed)
    print("fidelity %f" % fidelity)
```

In [3]:
```
matrix = [[3/4, 0,    8/9,    0,   0,   1/8,  -1/5,  1/3],
          [0,   -1/2, 0,    1/8, 1/4,   1,     0,     0],
          [8/9,   0,    1/8,  0,   0,   0,    1/7,    1/4],
          [0,   1/8,    0,   -1/3,  0,    2/3,    0,   1/6],
          [0,   1/4,    0,    0,   1/5,  0,      1,     1/9],
          [1/8, 1,      0,   2/3,  0,  -1/4,   0,      0],
          [-1/5,0,      1/7, 0,     1,    0,    -1/7,   0],
          [1/3, 0,    1/4,  1/6,   1/9,    0,    0,    1]]

vector = [1/np.sqrt(8), -1/np.sqrt(8), 1/np.sqrt(8), -1/np.sqrt(8), 1/np.sqrt(
8), -1/np.sqrt(8),
          1/np.sqrt(8), -1/np.sqrt(8)]
```

In [4]:
```
matrix = [[0.4,   -.65,  0,  -.1,    0,    0,    0,  -0.7],
          [-0.65,  0.3,-0.6,  0,      0,    0,    0,  -1.1],
          [0,     -.6,  1.2, -0.1,   0,    0,    0,  -0.5],
          [-0.1,    0, -0.1,  0.5, -0.8,   0,    0,  -0.3],
          [0,       0,   0,  -0.8, 0.25, -0.95,  0,   0],
          [0,       0,   0,   0,  -0.95,  0.9,  -0.2,   0],
          [0,       0,   0,   0,     0,  -0.2,   1,  -0.6],
          [-0.7, -1.1, -0.5, -0.3,   0,    0,  -0.6, 0.5]]

vector = [1.6, 0.8, -1.2, 0, 0, 0, -0.9, -1]
```

In [5]:
```
matrix = [[1.5,   -.6,  0,    0,     0,    0,    0,  -.7],
          [-0.6,  1.3, -0.6,  0,     0,    0,    0,  -.1],
          [0,     -.6,  1.2, -0.1,   0,    0,    0,  -0.5],
          [0,       0, -0.1,  1.1, -0.8,   0,    0,  -0.2],
          [0,       0,   0,  -0.8,  1.0, -0.2,   0,    0],
          [0,       0,   0,   0,   -0.2,  1.7,  -0.2,  0],
          [0,       0,   0,   0,     0,  -0.2,   0.5,  0],
          [-.7,   -.1,  -0.5, -0.2,   0,    0,    0,   1.5]]

vector = [1.6, 0.8, -1.2, 0, 0, 0, -0.9, -1]
```

In [6]:
```
matrix = [[13,      0,   0,   0,     0,   -13,    0,   0],
          [0,      11,   0,   0,     0,    0,    0,   -11],
          [0,       0,   22,  -8,    -8,    0,    0,   0],
          [0,       0,  -8,   13,    0,    -5,    0,   0],
          [0,       0,  -8,   0,     18,   0,    0,   -5],
          [-13,     0,   0,  -5,     0,    28,   -10,   0],
          [0,       0,   0,   0,     0,    -10,   15,   -5],
          [0,      -11,   0,   0,    -5,     0,    -5,   21]]

vector = [1.35, 0.8, 0, -1.2, -0.7, 0, -1, 0]
```

In [17]:
```
matrix = [[13,      0,   0,   0,     0,   -13,    0,   0],
          [0,      11,   0,   0,     0,    0,    0,   -11],
          [0,       0,   22,  -8,    -8,    0,    0,   0],
          [0,       0,  -8,   13,    0,    -5,    0,   0],
          [0,       0,  -8,   0,     13,   0,    0,   -5],
          [-13,     0,   0,  -5,     0,    28,   -10,   0],
          [0,       0,   0,   0,     0,    -10,   15,   -5],
          [0,      -11,   0,   0,    -5,     0,    -5,   21]]

vector = [1.4, 0.8, 0, -1.2, -0.7, 0, -1, 0]
```

In [7]:
```
aqua_globals.random_seed = 0
orig_size = len(vector)
matrix, vector, truncate_powerdim, truncate_hermitian = HHL.matrix_resize(matr
ix, vector)

# Initialize eigenvalue finding module
eigs = create_eigs(matrix, 3, 1, False)
num_q, num_a = eigs.get_register_sizes()

# Initialize initial state module
init_state = Custom(num_q, state_vector=vector)

# Initialize reciprocal rotation module
reci = LookupRotation(negative_evals=eigs._negative_evals, evo_time=eigs._evo_
time)

algo = HHL(matrix, vector, truncate_powerdim, truncate_hermitian, eigs,
           init_state, reci, num_q, num_a, orig_size)
result = algo.run(QuantumInstance(Aer.get_backend('statevector_simulator'),
                                  seed_simulator=aqua_globals.random_seed,
                                  seed_transpiler=aqua_globals.random_seed))
print("solution ", np.round(result['solution'], 5))

result_ref = NumPyLSsolver(matrix, vector).run()
print("classical solution ", np.round(result_ref['solution'], 5))

print("probability %f" % result['probability_result'])
fidelity(result['solution'], result_ref['solution'])
```

```
solution  [ 0.11256-0.j  0.05998-0.j -0.02998+0.j -0.08267+0.j -0.04583+0.j
  0.00475-0.j -0.08585+0.j -0.00406+0.j]
classical solution  [ 0.1157   0.0948  -0.0713  -0.13162 -0.06444  0.01186 -
0.0514   0.02208]
probability 0.537650
fidelity 0.872159
```

In [8]:
```
print("circuit_width", result['circuit_info']['width'])
print("circuit_depth", result['circuit_info']['depth'])
print("CNOT gates:\t", result['circuit_info']['operations']['cx'])
```

```
circuit_width 9
circuit_depth 129
CNOT gates:      70
```

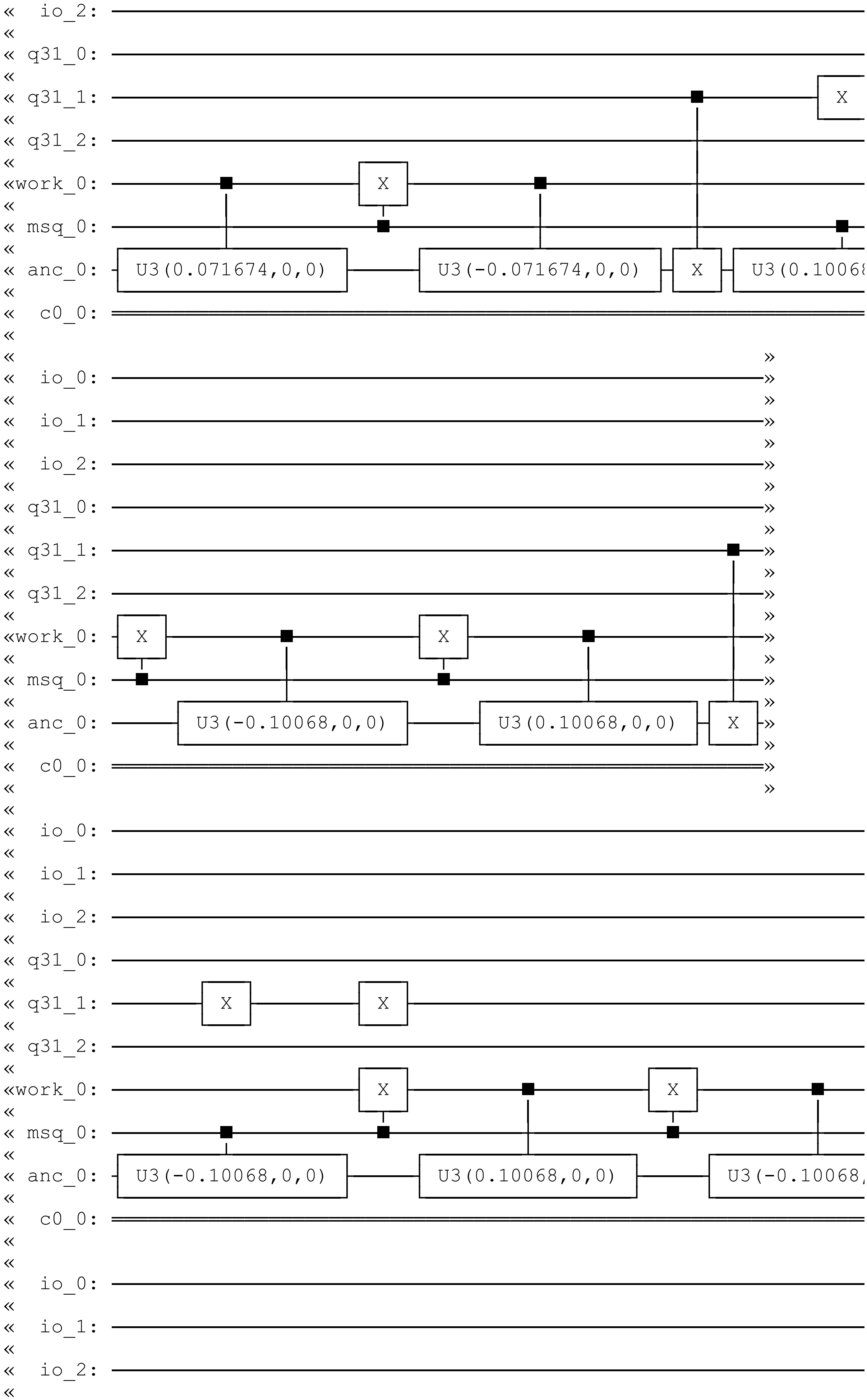
In [9]:
```
circuit= algo.construct_circuit(measurement=True)

circuit.draw(fold=100)
import matplotlib.pyplot as plt
%matplotlib inline
from qiskit import QuantumRegister, ClassicalRegister, QuantumCircuit
from qiskit.tools.visualization import circuit_drawer
circuit_drawer(circuit)
```

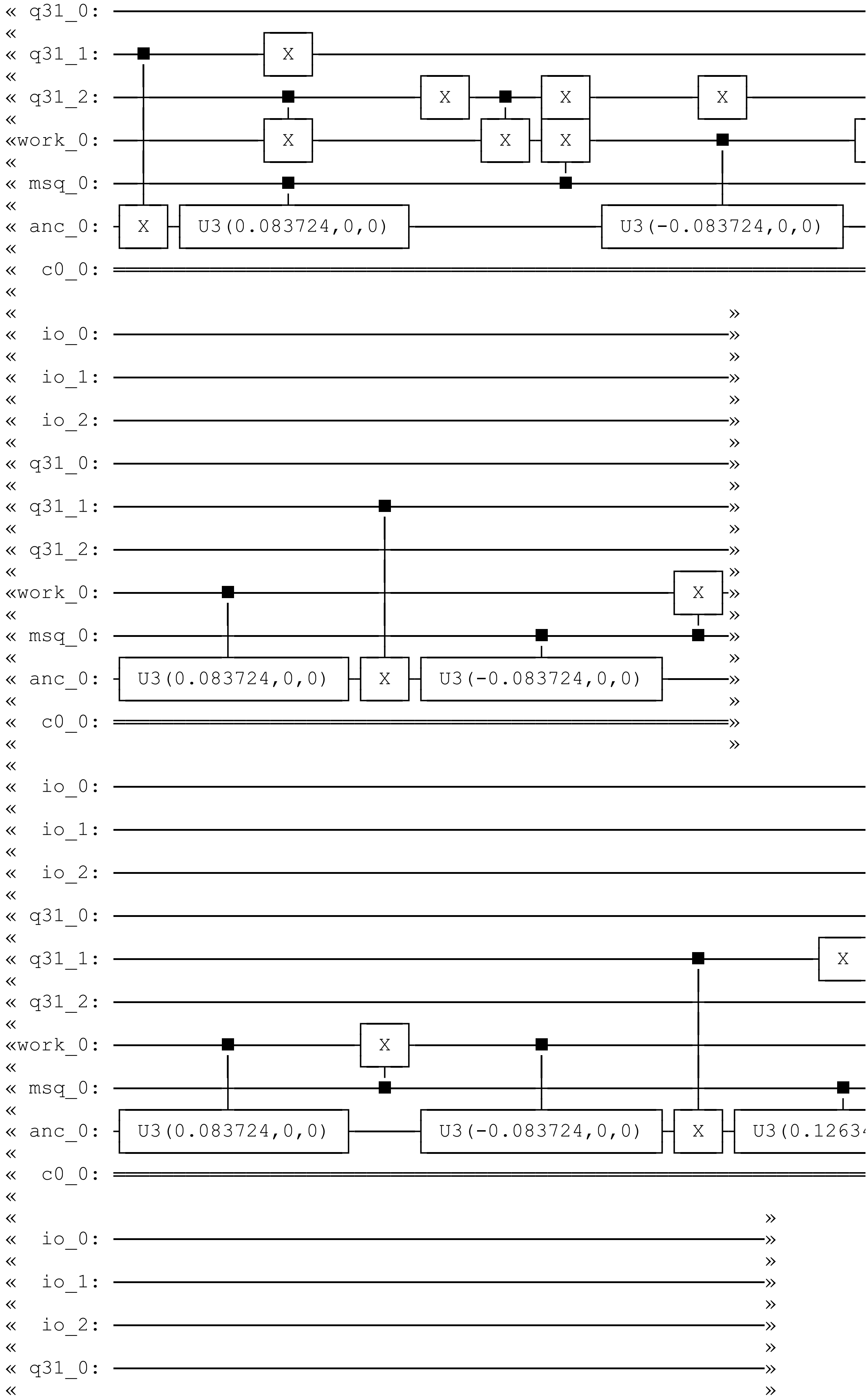
Out[9]:

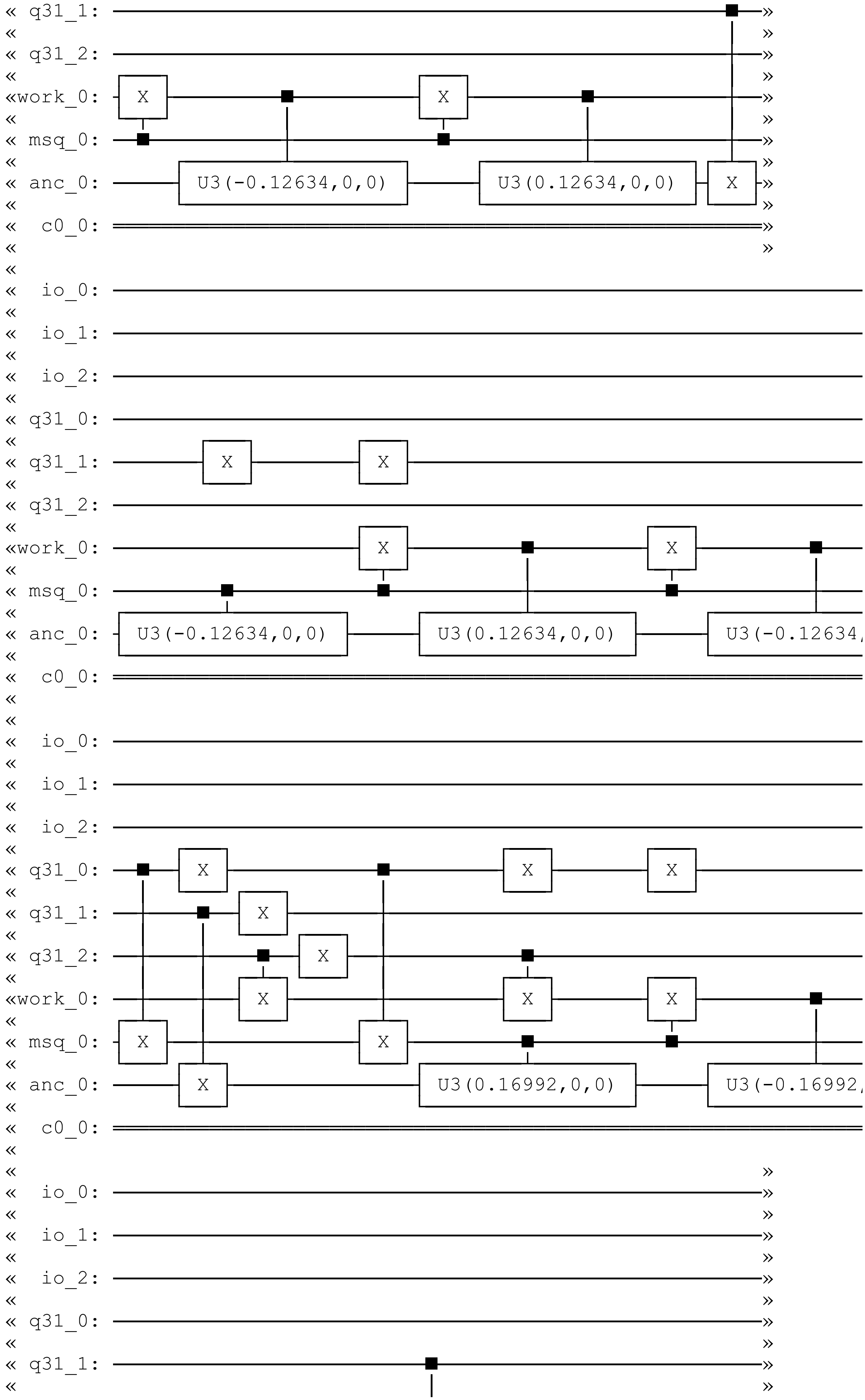
```
                                                                          »
   io_0: ─────────────────────────────────────────────────────────────────»
                                            ┌───┐┌──────────────────┐┌───┐»
   io_1: ───────────────────────────────────┤ X ├┤ U3(-0.30723,0,0) ├┤ X ├»
         ┌───────────────┐┌──────────┐      └─┬─┘└──────────────────┘└─┬─┘»
   io_2: ┤ U3(1.107,0,0) ├┤ U1(pi/4) ├────────■─────────────────────────■──»
         └──┬──────────┬─┘└──────────┘                                    »
  q31_0: ───┤ U2(0,pi) ├──────────────────────────────────────────────────»
            ├──────────┤                                                  »
  q31_1: ───┤ U2(0,pi) ├──────────────────────────────────────────────────»
            ├──────────┤                                                  »
  q31_2: ───┤ U2(0,pi) ├──────────────────────────────────────────────────»
            └──────────┘                                                  »
 work_0: ─────────────────────────────────────────────────────────────────»
                                                                          »
  msq_0: ─────────────────────────────────────────────────────────────────»
                                                                          »
  anc_0: ─────────────────────────────────────────────────────────────────»
                                                                          »
   c0_0: ═════════════════════════════════════════════════════════════════»
                                                                          »
«                                                                    ┌───┐»
«   io_0: ───────────────────────────────────────────────────────────┤ X ├»
«         ┌────────────────┐┌───┐┌──────────┐┌───┐┌──────────┐       └─┬─┘»
«   io_1: ┤ U3(1.6129,0,0) ├┤ X ├┤ U1(pi/4) ├┤ X ├┤ U1(pi/4) ├─────────┼──»
«         └────────────────┘└─┬─┘└──────────┘└─┬─┘└──────────┘         │  »
«   io_2: ────────────────────■────────────────■───────────────────────■──»
«                                                                         »
«  q31_0: ────────────────────────────────────────────────────────────────»
«                                                                         »
«  q31_1: ────────────────────────────────────────────────────────────────»
«                                                                         »
«  q31_2: ────────────────────────────────────────────────────────────────»
«                                                                         »
« work_0: ────────────────────────────────────────────────────────────────»
«                                                                         »
«  msq_0: ────────────────────────────────────────────────────────────────»
«                                                                         »
«  anc_0: ────────────────────────────────────────────────────────────────»
«                                                                         »
«   c0_0: ════════════════════════════════════════════════════════════════»
«                                                                         »
«         ┌────────────────┐┌───┐┌───────────────────┐┌───┐┌───┐┌───┐»
«   io_0: ┤ U3(1.0529,0,0) ├┤ X ├┤ U3(-0.51792,0,0) ├┤ X ├┤ X ├┤ X ├»
«         └────────────────┘└─┬─┘└───────────────────┘└─┬─┘└─┬─┘└─┬─┘»
«   io_1: ────────────────────■─────────────────────────■────┼────■──»
«                                                            │       »
«   io_2: ───────────────────────────────────────────────────■───────»
«                                                                    »
«  q31_0: ───────────────────────────────────────────────────────────»
«                                                                    »
«  q31_1: ───────────────────────────────────────────────────────────»
«                                                                    »
«  q31_2: ───────────────────────────────────────────────────────────»
«                                                                    »
« work_0: ───────────────────────────────────────────────────────────»
«                                                                    »
«  msq_0: ───────────────────────────────────────────────────────────»
«                                                                    »
«  anc_0: ───────────────────────────────────────────────────────────»
«                                                                    »
«   c0_0: ═══════════════════════════════════════════════════════════»
«                                                                    »
«         ┌──────────────────┐┌───┐┌──────────────────┐┌───┐┌───────────┐┌
```

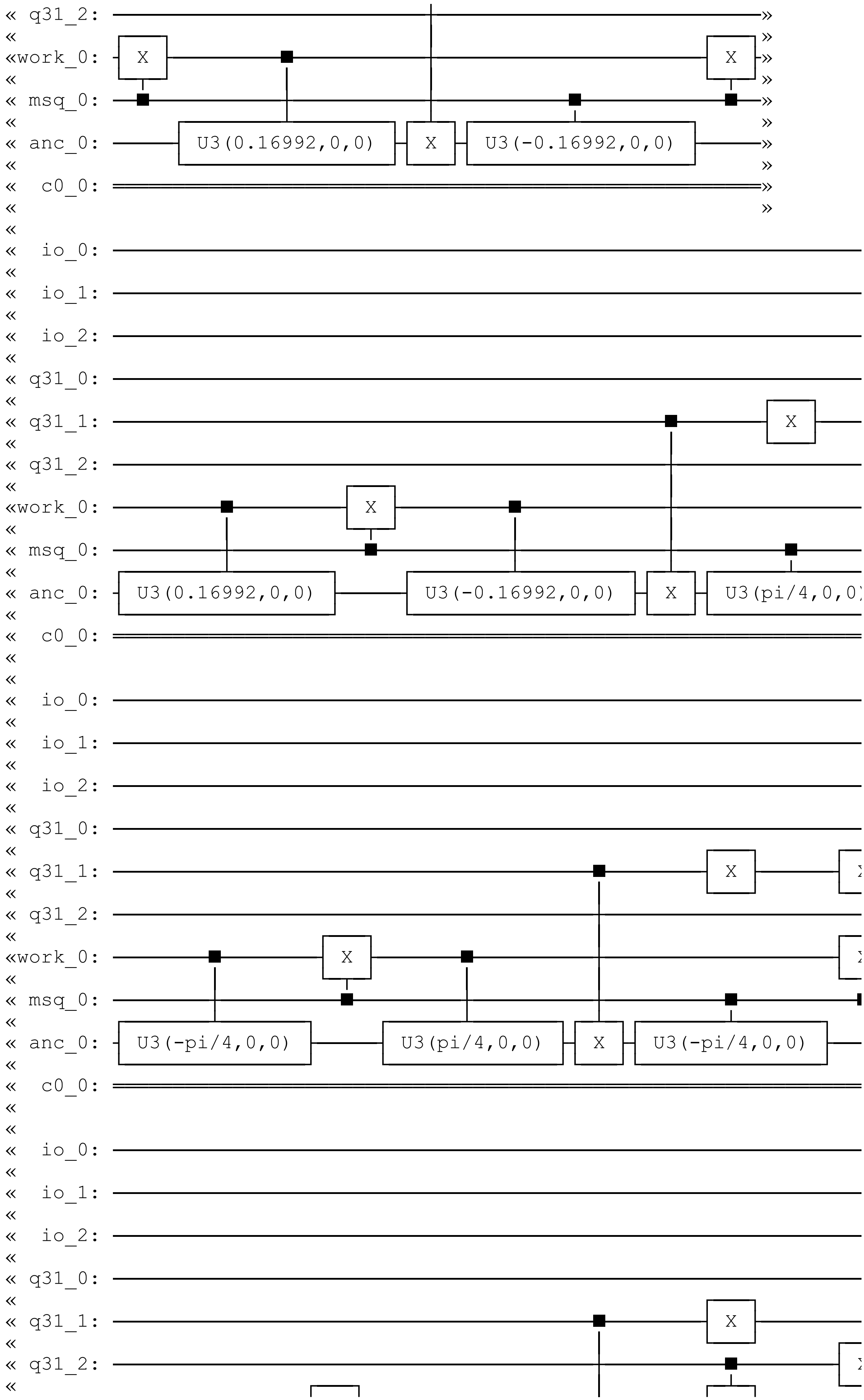
```
«  io_0: ┤ U3(-0.51792,0,0) ├┤ X ├┤ U3(1.0529,0,0) ├┤ X ├┤ U1(3pi/4) ├
«                              │                       │
«  io_1: ──────────────────────■───────────────────────┼──────────────
«                                                      │
«  io_2: ──────────────────────────────────────────────■──────────────
«
« q31_0: ─────────────────────────────────────────────────────────────
«
« q31_1: ─────────────────────────────────────────────────────────────
«
« q31_2: ─────────────────────────────────────────────────────────────
«
«work_0: ─────────────────────────────────────────────────────────────
«
« msq_0: ─────────────────────────────────────────────────────────────
«
« anc_0: ─────────────────────────────────────────────────────────────
«
«  c0_0: ═════════════════════════════════════════════════════════════
«
«                                                                     »
«  io_0: ┤ U1(-pi/4) ├┤ X ├┤ X ├┤ X ├┤ U1(-pi/4) ├┤ X ├┤ U1(-pi/4) ├»
«                       │    │    │                  │                 »
«  io_1: ───────────────■────┼────■──────────────────■─────────────────»
«                            │                                         »
«  io_2: ────────────────────■─────────────────────────────────────────»
«                                                                     »
« q31_0: ──────────────────────────────────────────────────────────────»
«                                                                     »
« q31_1: ──────────────────────────────────────────────────────────────»
«                                                                     »
« q31_2: ──────────────────────────────────────────────────────────────»
«                                                                     »
«work_0: ──────────────────────────────────────────────────────────────»
«                                                                     »
« msq_0: ──────────────────────────────────────────────────────────────»
«                                                                     »
« anc_0: ──────────────────────────────────────────────────────────────»
«                                                                     »
«  c0_0: ══════════════════════════════════════════════════════════════»
«                                                                     »
«
«  io_0: ┤0                         ├───────────────────┤0
«        │                          │                   │
«  io_1: ┤1                         ├───────────────────┤1
«        │  Controlled-Evolution^1  │                   │
«  io_2: ┤2                         ├───────────────────┤2 Controlled-Evoluti
«        │                          │ ┌──────────────┐  │
« q31_0: ┤3                         ├─┤ U1(1.3458)   ├──┤
«        └──────────────────────────┘ └──────────────┘  │
« q31_1: ───────────────────────────────────────────────┤3
«                                                       └──────────────
« q31_2: ──────────────────────────────────────────────────────────────
«
«work_0: ──────────────────────────────────────────────────────────────
«
« msq_0: ──────────────────────────────────────────────────────────────
«
« anc_0: ──────────────────────────────────────────────────────────────
«
«  c0_0: ══════════════════════════════════════════════════════════════
«
«                  ┌──────────────────────────┐
«  io_0: ──────────┤0                         ├────────────────────────
«                  │                          │
```

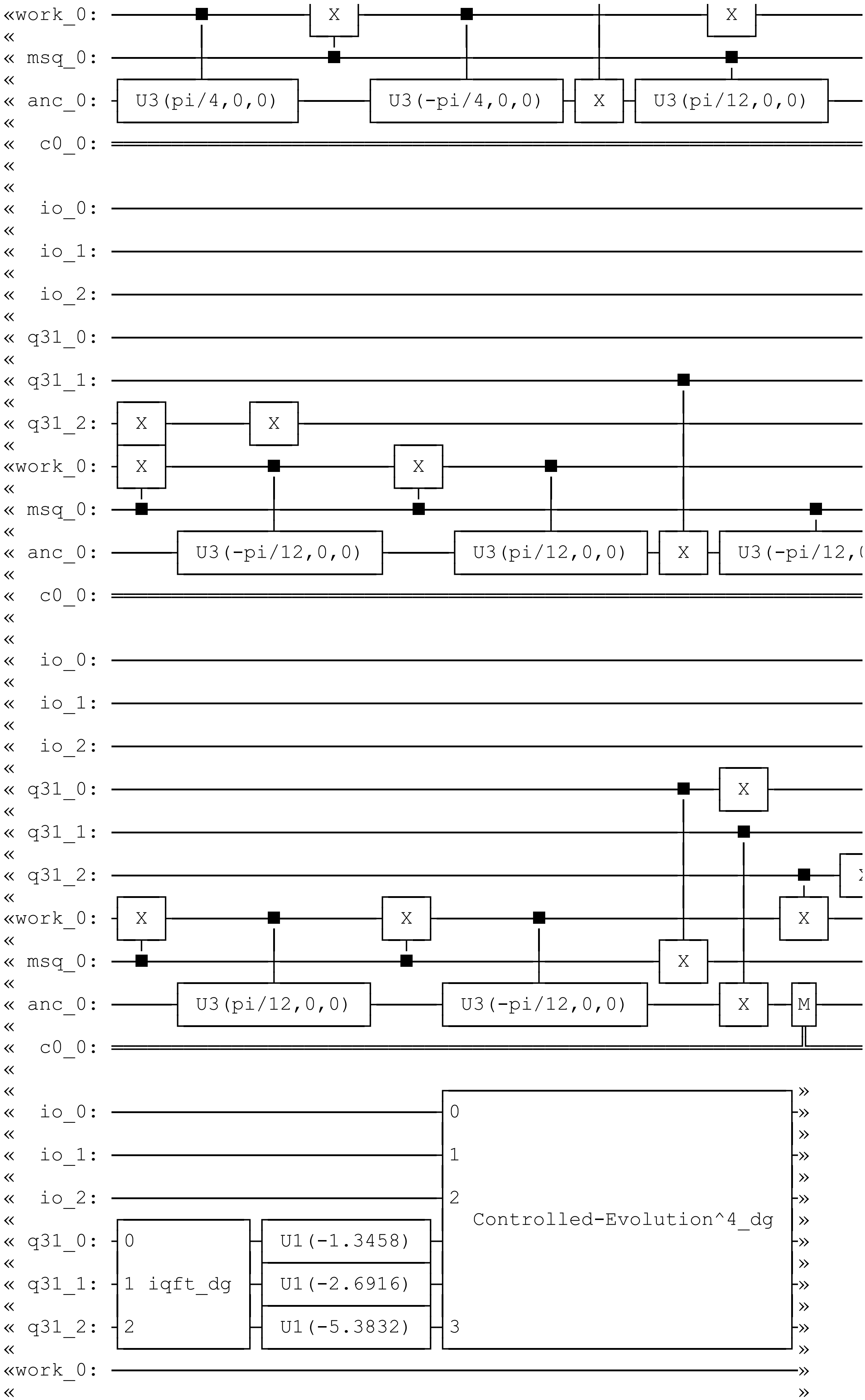
```
«  io_1: ─────────────────┤1                        ├─────────────────────────────
«                         │                         │
«  io_2: ─────────────────┤2                        ├─────────────────────────────
«                         │  Controlled-Evolution^4 │                 ┌──────────
« q31_0: ─────────────────┤                         ├─────────────────┤0
«        ┌──────────────┐ │                         │                 │
« q31_1: ┤ U1(2.6916)   ├─┤                         ├─────────────────┤1 iqft
«        └──────────────┘ │                         │ ┌─────────────┐ │
« q31_2: ─────────────────┤3                        ├─┤ U1(5.3832)  ├─┤2
«                         └─────────────────────────┘ └─────────────┘ └──────────
«work_0: ─────────────────────────────────────────────────────────────────────────
«
« msq_0: ─────────────────────────────────────────────────────────────────────────
«
« anc_0: ─────────────────────────────────────────────────────────────────────────
«
«  c0_0: ═════════════════════════════════════════════════════════════════════════
«
«                                                                   »
«  io_0: ───────────────────────────────────────────────────────────»
«                                                                   »
«  io_1: ───────────────────────────────────────────────────────────»
«                                                                   »
«  io_2: ───────────────────────────────────────────────────────────»
«                                                                   »
« q31_0: ───────────────────────────────────────────────────────────»
«                                                                   »
« q31_1: ───────────────────────────────────────────────────────────»
«                                                                   »
« q31_2: ─────────■─────────────────────────────────────────────────»
«               ┌─┴─┐          ┌───┐                         ┌───┐  »
«work_0: ───────┤ X ├──────────┤ X ├───────────■─────────────┤ X ├──»
«               └─┬─┘          └─┬─┘           │             └─┬─┘  »
« msq_0: ─────────■──────────────■─────────────┼───────────────■────»
«        ┌────────┴─────────┐     ┌────────────┴───────┐            »
« anc_0: ┤ U3(0.071674,0,0) ├─────┤ U3(-0.071674,0,0)  ├────────────»
«        └──────────────────┘     └────────────────────┘            »
«  c0_0: ═══════════════════════════════════════════════════════════»
«                                                                   »
«                                                                   »
«  io_0: ───────────────────────────────────────────────────────────»
«                                                                   »
«  io_1: ───────────────────────────────────────────────────────────»
«                                                                   »
«  io_2: ───────────────────────────────────────────────────────────»
«                                                                   »
« q31_0: ───────────────────────────────────────────────────────────»
«                                                                   »
« q31_1: ───────────────────────■───────────────────────────────────»
«                               │                                   »
« q31_2: ───────────────────────┼───────────────────────────────────»
«                               │                            ┌───┐  »
«work_0: ─────────■─────────────┼────────────────────────────┤ X ├──»
«                 │             │                            └─┬─┘  »
« msq_0: ─────────┼─────────────┼────────────■─────────────────■────»
«        ┌────────┴─────────┐ ┌─┴─┐ ┌────────┴───────────┐          »
« anc_0: ┤ U3(0.071674,0,0) ├─┤ X ├─┤ U3(-0.071674,0,0)  ├──────────»
«        └──────────────────┘ └───┘ └────────────────────┘          »
«  c0_0: ═══════════════════════════════════════════════════════════»
«                                                                   »
«
«  io_0: ─────────────────────────────────────────────────────────────────────────
«
«  io_1: ─────────────────────────────────────────────────────────────────────────
«
```


io_2:
q31_0:
q31_1:
q31_2:
work_0:
msq_0:
anc_0:
c0_0:
X
U3(0.071674,0,0)
U3(-0.071674,0,0)
U3(0.1006
io_0:
io_1:
U3(-0.10068,0,0)
U3(0.10068,0,0)
U3(-0.10068,


io_0:
io_1:
io_2:
q31_0:
q31_1:
q31_2:
work_0:
msq_0:
anc_0:
c0_0:
X
U3(0.083724,0,0)
U3(-0.083724,0,0)
U3(0.1263


« q31_1:
« q31_2:
«work_0:
« msq_0:
« anc_0:
« c0_0:
X
U3(-0.12634,0,0)
U3(0.12634,0,0)
« io_0:
« io_1:
« io_2:
« q31_0:
U3(-0.12634,
U3(0.16992,0,0)
U3(-0.16992,


q31_2:
work_0:
msq_0:
anc_0:
c0_0:
X
U3(0.16992,0,0)
X
U3(-0.16992,0,0)
X
io_0:
io_1:
io_2:
q31_0:
q31_1:
q31_2:
work_0:
msq_0:
anc_0:
c0_0:
X
U3(0.16992,0,0)
X
U3(-0.16992,0,0)
X
U3(pi/4,0,0
io_0:
io_1:
io_2:
q31_0:
q31_1:
q31_2:
work_0:
msq_0:
anc_0:
c0_0:
X
U3(-pi/4,0,0)
X
U3(pi/4,0,0)
X
U3(-pi/4,0,0)
io_0:
io_1:
io_2:
q31_0:
q31_1:
q31_2:
X


«work_0:
« msq_0:
« anc_0:
« c0_0:
« io_0:
« io_1:
« io_2:
« q31_0:
« q31_1:
« q31_2:
«work_0:
« msq_0:
« anc_0:
« c0_0:
« io_0:
« io_1:
« io_2:
« q31_0:
« q31_1:
« q31_2:
«work_0:
« msq_0:
« anc_0:
« c0_0:
« io_0:
« io_1:
« io_2:
« q31_0:
« q31_1:
« q31_2:
«work_0:
X
U3(pi/4,0,0)
U3(-pi/4,0,0)
U3(pi/12,0,0)
U3(-pi/12,0,0)
M
iqft_dg
U1(-1.3458)
U1(-2.6916)
U1(-5.3832)
Controlled-Evolution^4_dg

```
« msq_0: ──────────────────────────────────────────────────────»
«                                                              »
« anc_0: ──────────────────────────────────────────────────────»
«                                                              »
«  c0_0: ══════════════════════════════════════════════════════»
«                                                              »
«        ┌─────────────────────────────┐┌─────────────────────────────┐»
«  io_0: ┤0                            ├┤0                            ├»
«        │                             ││                             │»
«  io_1: ┤1                            ├┤1                            ├»
«        │                             ││  Controlled-Evolution^1_dg  │»
«  io_2: ┤2 Controlled-Evolution^2_dg  ├┤2                            ├»
«        │                             ││                             │»
« q31_0: ┤                             ├┤3                            ├»
«        │                             │└───────┬──────────────┬──────┘»
« q31_1: ┤3                            ├────────┤ U2(-2pi,pi)  ├───────»
«        └───────┬──────────────┬──────┘        └──────────────┘       »
« q31_2: ────────┤ U2(-2pi,pi)  ├──────────────────────────────────────»
«                └──────────────┘                                      »
«work_0: ──────────────────────────────────────────────────────────────»
«                                                                      »
« msq_0: ──────────────────────────────────────────────────────────────»
«                                                                      »
« anc_0: ──────────────────────────────────────────────────────────────»
«                                                                      »
«  c0_0: ══════════════════════════════════════════════════════════════»
«                                                                      »
«
«  io_0: ───────────────
«
«  io_1: ───────────────
«
«  io_2: ───────────────
«        ┌─────────────┐
« q31_0: ┤ U2(-2pi,pi) ├
«        └─────────────┘
« q31_1: ───────────────
«
« q31_2: ───────────────
«
«work_0: ───────────────
«
« msq_0: ───────────────
«
« anc_0: ───────────────
«
«  c0_0: ═══════════════
«
```

In [10]:
```python
an_array = [1.63, 0.85, 0, -1.25, -0.9, 0, -1, 0]
norm = np.linalg.norm(an_array)
normal_array = an_array/norm
print(normal_array)
```

```
[ 0.62729901  0.32711912  0.         -0.48105752 -0.34636142  0.
 -0.38484602  0.        ]
```

In [ ]:

# Quantum classifier with qiskit

In [5]:
```
from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import wine
from qiskit.circuit.library import TwoLocal

seed = 1376
aqua_globals.random_seed = seed

# Use Wine data set for training and test data
feature_dim = 4  # dimension of each data point
_, training_input, test_input, _ = wine(training_size=12,
                                        test_size=4,
                                        n=feature_dim)

feature_map = RawFeatureVector(feature_dimension=feature_dim)
vqc = VQC(COBYLA(maxiter=100),
          feature_map,
          TwoLocal(feature_map.num_qubits, ['ry', 'rz'], 'cz', reps=3),
          training_input,
          test_input)
result = vqc.run(QuantumInstance(BasicAer.get_backend('statevector_simulator'
),
                                 shots=1024, seed_simulator=seed, seed_transpi
ler=seed))

print('Testing accuracy: {:0.2f}'.format(result['testing_accuracy']))
```

```
Testing accuracy: 1.00
```

In [81]:
```
from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import ad_hoc_data
from qiskit.circuit.library import TwoLocal
from qiskit.aqua.components.feature_maps import SecondOrderExpansion

seed =  10598
#seed = 1376
aqua_globals.random_seed = seed

feature_dim = 2 # dimension of each data point
training_dataset_size = 20
testing_dataset_size = 10
#random_seed = 10598
#shots = 1024

_, training_input, test_input, _ = ad_hoc_data(training_size=training_dataset_
size,
                                               test_size=testing_dataset_size,
n=feature_dim, gap=0.3)


#feature_map = RawFeatureVector(feature_dimension=feature_dim)

feature_map = SecondOrderExpansion(feature_dimension=feature_dim, depth=2, ent
angler_map=None, entanglement='full')
vqc = VQC(COBYLA(maxiter=100),
          feature_map,
          TwoLocal(feature_map.num_qubits, ['ry', 'rz'], 'cz', reps=3),
          training_input,
          test_input)
#result = vqc.run(QuantumInstance(BasicAer.get_backend('qasm_simulator'),
#                                  shots=1024, seed_simulator=seed, seed_transp
iler=seed))
result = vqc.run(QuantumInstance(BasicAer.get_backend('qasm_simulator'),
                                 shots=1024, seed_simulator=seed, seed_transpi
ler=seed))

print('Testing accuracy: {:0.2f}'.format(result['testing_accuracy']))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:26: Deprecat
ionWarning: The qiskit.aqua.components.feature_maps.SecondOrderExpansion obje
ct is deprecated as of 0.7.0 and will be removed no sooner than 3 months afte
r the release. You should use qiskit.circuit.library.ZZFeatureMap instead.
C:\ProgramData\Anaconda3\lib\site-packages\qiskit\aqua\components\feature_map
s\second_order_expansion.py:59: DeprecationWarning: The qiskit.aqua.component
s.feature_maps.PauliZExpansion class is deprecated as of 0.7.0 and will be re
moved no sooner than 3 months after the release. You should use qiskit.circui
t.library.PauliFeatureMap instead.
  z_order=2, data_map_func=data_map_func)
C:\ProgramData\Anaconda3\lib\site-packages\qiskit\aqua\components\feature_map
s\pauli_z_expansion.py:71: DeprecationWarning: The qiskit.aqua.components.fea
ture_maps.PauliExpansion object is deprecated as of 0.7.0 and will be removed
no sooner than 3 months after the release. You should use qiskit.circuit.libr
ary.PauliFeatureMap instead.
  paulis=pauli_string, data_map_func=data_map_func)
C:\ProgramData\Anaconda3\lib\site-packages\qiskit\aqua\algorithms\classifiers
\vqc.py:138: DeprecationWarning: The qiskit.aqua.components.feature_maps.Feat
ureMap object is deprecated as of 0.7.0 and will be removed no earlier than 3
months after the release. You should pass a QuantumCircuit object instead. Se
e also qiskit.circuit.library.data_preparation for a collection of suitable c
ircuits.
  self.feature_map = feature_map

Testing accuracy: 1.00
```

In [82]:
```
from qiskit import Aer
from qiskit.aqua.utils import split_dataset_to_data_and_labels
```

In [83]:
```
datapoints, class_to_label = split_dataset_to_data_and_labels(test_input, clas
s_names=None)
print(datapoints[1])
```

```
[0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
```

In [84]:
```
predicted_probs, predicted_labels = vqc.predict(datapoints[0])
#predicted_classes = map_label_to_class_name(predicted_labels, vqc.label_to_cl
ass)
print("prediction:   {}".format(predicted_labels))
```

```
prediction:   [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
```

In [85]:
```
l=len(predicted_labels)
l
#datapoints[1][13]
```

Out[85]:
```
20
```

In [86]:
```
for i in range(0,l):
    print(datapoints[0][i], 'actual class', datapoints[1][i], 'predicted', pre
dicted_labels[i])
    #if datapoints[1][i] == predicted_labels[i]:
```

```
[3.58141563 2.26194671] actual class 0 predicted 0
[3.20442451 3.33008821] actual class 0 predicted 0
[1.13097336 1.63362818] actual class 0 predicted 0
[2.89026524 0.87964594] actual class 0 predicted 0
[6.1575216  1.13097336] actual class 0 predicted 0
[5.65486678 6.22035345] actual class 0 predicted 0
[4.58672527 3.83274304] actual class 0 predicted 0
[1.50796447 0.56548668] actual class 0 predicted 0
[6.09468975 0.62831853] actual class 0 predicted 0
[2.82743339 1.13097336] actual class 0 predicted 0
[1.38230077 4.52389342] actual class 1 predicted 1
[3.89557489 4.46106157] actual class 1 predicted 1
[5.46637122 3.0787608 ] actual class 1 predicted 1
[1.31946891 4.52389342] actual class 1 predicted 1
[3.89557489 5.02654825] actual class 1 predicted 1
[5.40353936 3.0787608 ] actual class 1 predicted 1
[5.65486678 5.90619419] actual class 1 predicted 1
[5.27787566 2.38761042] actual class 1 predicted 1
[0.31415927 2.45044227] actual class 1 predicted 1
[4.39822972 0.37699112] actual class 1 predicted 1
```

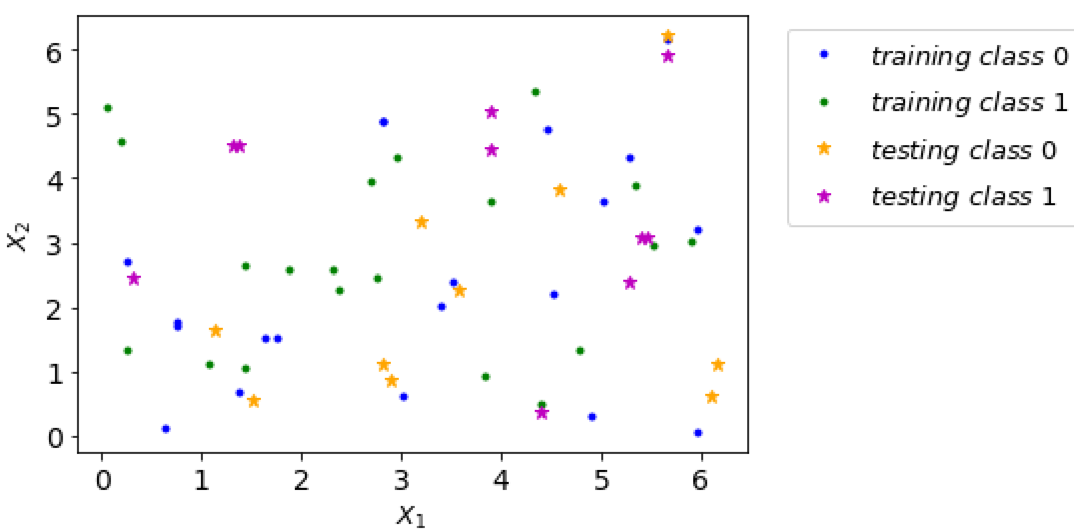
In [87]:
```
datapoints_train, class_to_label_train = split_dataset_to_data_and_labels(trai
ning_input, class_names=None)
print(datapoints_train[1])
print(datapoints_train[0])
print(len((datapoints_train[1])))
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1]
[[2.82743339 4.90088454]
 [0.25132741 2.70176968]
 [5.65486678 6.1575216 ]
 [5.27787566 4.33539786]
 [5.96902604 0.06283185]
 [3.39292007 2.0106193 ]
 [4.46106157 4.77522083]
 [5.02654825 3.64424748]
 [1.38230077 0.69115038]
 [4.90088454 0.31415927]
 [3.01592895 0.62831853]
 [2.82743339 4.90088454]
 [0.75398224 1.75929189]
 [5.96902604 3.20442451]
 [0.75398224 1.69646003]
 [0.62831853 0.12566371]
 [4.52389342 2.19911486]
 [1.75929189 1.50796447]
 [3.51858377 2.38761042]
 [1.63362818 1.50796447]
 [0.06283185 5.0893801 ]
 [1.44513262 1.0681415 ]
 [0.25132741 1.31946891]
 [4.39822972 0.50265482]
 [4.33539786 5.34070751]
 [2.95309709 4.33539786]
 [5.90619419 3.01592895]
 [1.44513262 2.63893783]
 [2.32477856 2.57610598]
 [3.83274304 0.9424778 ]
 [5.52920307 2.95309709]
 [2.70176968 3.95840674]
 [1.88495559 2.57610598]
 [3.89557489 3.64424748]
 [2.76460154 2.45044227]
 [1.0681415  1.13097336]
 [4.77522083 1.31946891]
 [2.38761042 2.26194671]
 [0.18849556 4.58672527]
 [5.34070751 3.89557489]]
40
```

In [89]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
%matplotlib inline

#plt.rcParams.update({'font.size': 14})
mpl.rc('xtick', labelsize=14)
mpl.rc('ytick', labelsize=14)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 13}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
rom~ curve$')



for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==0 :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
```

```
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                  #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)

plt.plot(x0_train, y0_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='b', label= r'$training ~class~ 0$')

plt.plot(x1_train, y1_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='green', label= r'$training ~class~ 1$')

plt.plot(x0_test, y0_test, linewidth = 3, linestyle='', marker='*', markersize
=7, color='orange', label= r'$testing ~class~ 0$')

plt.plot(x1_test, y1_test, linewidth = 3, linestyle='', marker='*', markersize
=7, color='m', label= r'$testing ~class~ 1$')

plt.ylabel(r'$X_2$')
plt.xlabel(r'$X_1$')
#plt.legend()
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left")
plt.show()
```

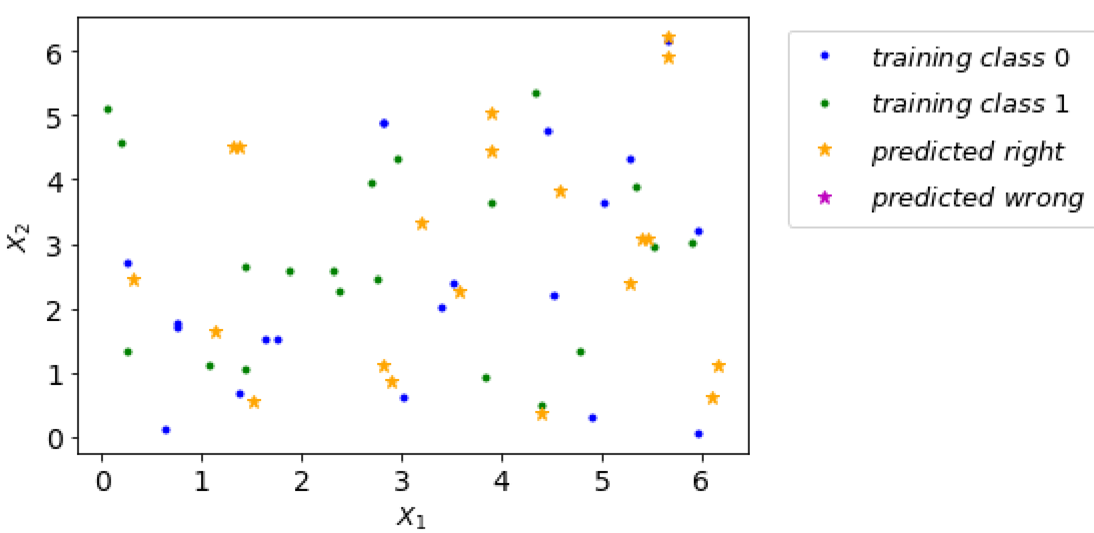
```
0
```



In [88]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
%matplotlib inline

#plt.rcParams.update({'font.size': 14})
mpl.rc('xtick', labelsize=14)
mpl.rc('ytick', labelsize=14)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 13}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
rom~ curve$')



for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==predicted_labels[i] :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
```

```
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                  #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)

plt.plot(x0_train, y0_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='b', label= r'$training ~class~ 0$')

plt.plot(x1_train, y1_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='green', label= r'$training ~class~ 1$')

plt.plot(x0_test, y0_test, linewidth = 3, linestyle='', marker='*', markersize
=7, color='orange', label= r'$predicted~ right$')

plt.plot(x1_test, y1_test, linewidth = 3, linestyle='', marker='*', markersize
=7, color='m', label= r'$predicted ~wrong$')

plt.ylabel(r'$X_2$')
plt.xlabel(r'$X_1$')
#plt.legend()
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left")
plt.show()
```

```
0
```



In [71]:
```
from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import ad_hoc_data
from qiskit.circuit.library import TwoLocal
from qiskit.aqua.components.feature_maps import SecondOrderExpansion
from qiskit.aqua.algorithms import QSVM

seed =  10598
aqua_globals.random_seed = seed

feature_dim = 2 # dimension of each data point
training_dataset_size = 20
testing_dataset_size = 10
#random_seed = 10598
#shots = 1024

#sample_Total, training_input, test_input, class_labels = ad_hoc_data(training
_size=training_dataset_size,
#                                                                      test_siz
e=testing_dataset_size,
#                                                                      n=featur
e_dim, gap=0.3, PLOT_DATA=True)

# Use Wine data set for training and test data
#feature_dim = 4  # dimension of each data point
_, training_input, test_input, _ = ad_hoc_data(training_size=training_dataset_
size,
                                               test_size=testing_dataset_size,
n=feature_dim, gap=0.3)


#feature_map = RawFeatureVector(feature_dimension=feature_dim)
#feature_map =SecondOrderExpansion(feature_dimension=feature_dim, depth=2)
feature_map = SecondOrderExpansion(feature_dimension=feature_dim, depth=2, ent
angler_map=None, entanglement='full')

qsvmm = QSVM(feature_map, training_dataset=training_input, test_dataset=test_i
nput,
           datapoints=None, multiclass_extension=None, quantum_instance=None)

resultqsvm = qsvmm.run(QuantumInstance(BasicAer.get_backend('qasm_simulator'),
                                 shots=1024, seed_simulator=seed, seed_transpi
ler=seed))

print('Testing accuracy: {:0.2f}'.format(resultqsvm['testing_accuracy']))
#print(resultresultqsvm)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:32: Deprecat
ionWarning: The qiskit.aqua.components.feature_maps.SecondOrderExpansion obje
ct is deprecated as of 0.7.0 and will be removed no sooner than 3 months afte
r the release. You should use qiskit.circuit.library.ZZFeatureMap instead.
C:\ProgramData\Anaconda3\lib\site-packages\qiskit\aqua\components\feature_map
s\second_order_expansion.py:59: DeprecationWarning: The qiskit.aqua.component
s.feature_maps.PauliZExpansion class is deprecated as of 0.7.0 and will be re
moved no sooner than 3 months after the release. You should use qiskit.circui
t.library.PauliFeatureMap instead.
  z_order=2, data_map_func=data_map_func)
C:\ProgramData\Anaconda3\lib\site-packages\qiskit\aqua\components\feature_map
s\pauli_z_expansion.py:71: DeprecationWarning: The qiskit.aqua.components.fea
ture_maps.PauliExpansion object is deprecated as of 0.7.0 and will be removed
no sooner than 3 months after the release. You should use qiskit.circuit.libr
ary.PauliFeatureMap instead.
  paulis=pauli_string, data_map_func=data_map_func)
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:35: Deprecat
ionWarning:
                The <class 'qiskit.aqua.components.feature_maps.second_order_
expansion.SecondOrderExpansion'> object as input for the QSVM is deprecated a
s of 0.7.0 and will
                be removed no earlier than 3 months after the release.
                You should pass a QuantumCircuit object instead.
                See also qiskit.circuit.library.data_preparation for a collec
tion
                of suitable circuits.

Testing accuracy: 1.00
```

In [3]:
```python
from qiskit import Aer
from qiskit.aqua.utils import split_dataset_to_data_and_labels
```

In [4]:
```python
datapoints, class_to_label = split_dataset_to_data_and_labels(test_input, clas
s_names=None)
print(datapoints[0][18:])
#datapoints[0][2]
```

```
[[0.31415927 2.45044227]
 [4.39822972 0.37699112]]
```

In [5]:
```python
import numpy as np
predicted_labels = qsvmm.predict(datapoints[0], quantum_instance=None)
#predicted_classes = map_label_to_class_name(predicted_labels, vqc.label_to_cl
ass)
print("prediction:   {}".format(predicted_labels))
```

```
prediction:   [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]
```

In [6]:
```
for i in range(0,len(predicted_labels)):
    print(datapoints[0][i], 'actual class', datapoints[1][i], 'predicted', pre
dicted_labels[i])
```

```
[3.58141563 2.26194671] actual class 0 predicted 0
[3.20442451 3.33008821] actual class 0 predicted 0
[1.13097336 1.63362818] actual class 0 predicted 0
[2.89026524 0.87964594] actual class 0 predicted 0
[6.1575216  1.13097336] actual class 0 predicted 0
[5.65486678 6.22035345] actual class 0 predicted 0
[4.58672527 3.83274304] actual class 0 predicted 0
[1.50796447 0.56548668] actual class 0 predicted 0
[6.09468975 0.62831853] actual class 0 predicted 0
[2.82743339 1.13097336] actual class 0 predicted 0
[1.38230077 4.52389342] actual class 1 predicted 1
[3.89557489 4.46106157] actual class 1 predicted 1
[5.46637122 3.0787608 ] actual class 1 predicted 1
[1.31946891 4.52389342] actual class 1 predicted 1
[3.89557489 5.02654825] actual class 1 predicted 1
[5.40353936 3.0787608 ] actual class 1 predicted 1
[5.65486678 5.90619419] actual class 1 predicted 1
[5.27787566 2.38761042] actual class 1 predicted 1
[0.31415927 2.45044227] actual class 1 predicted 1
[4.39822972 0.37699112] actual class 1 predicted 1
```

In [7]:
```
datapoints_train, class_to_label_train = split_dataset_to_data_and_labels(trai
ning_input, class_names=None)
print(datapoints_train[1])
print(datapoints_train[0])
print(len((datapoints_train[1])))
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1]
[[2.82743339 4.90088454]
 [0.25132741 2.70176968]
 [5.65486678 6.1575216 ]
 [5.27787566 4.33539786]
 [5.96902604 0.06283185]
 [3.39292007 2.0106193 ]
 [4.46106157 4.77522083]
 [5.02654825 3.64424748]
 [1.38230077 0.69115038]
 [4.90088454 0.31415927]
 [3.01592895 0.62831853]
 [2.82743339 4.90088454]
 [0.75398224 1.75929189]
 [5.96902604 3.20442451]
 [0.75398224 1.69646003]
 [0.62831853 0.12566371]
 [4.52389342 2.19911486]
 [1.75929189 1.50796447]
 [3.51858377 2.38761042]
 [1.63362818 1.50796447]
 [0.06283185 5.0893801 ]
 [1.44513262 1.0681415 ]
 [0.25132741 1.31946891]
 [4.39822972 0.50265482]
 [4.33539786 5.34070751]
 [2.95309709 4.33539786]
 [5.90619419 3.01592895]
 [1.44513262 2.63893783]
 [2.32477856 2.57610598]
 [3.83274304 0.9424778 ]
 [5.52920307 2.95309709]
 [2.70176968 3.95840674]
 [1.88495559 2.57610598]
 [3.89557489 3.64424748]
 [2.76460154 2.45044227]
 [1.0681415  1.13097336]
 [4.77522083 1.31946891]
 [2.38761042 2.26194671]
 [0.18849556 4.58672527]
 [5.34070751 3.89557489]]
40
```

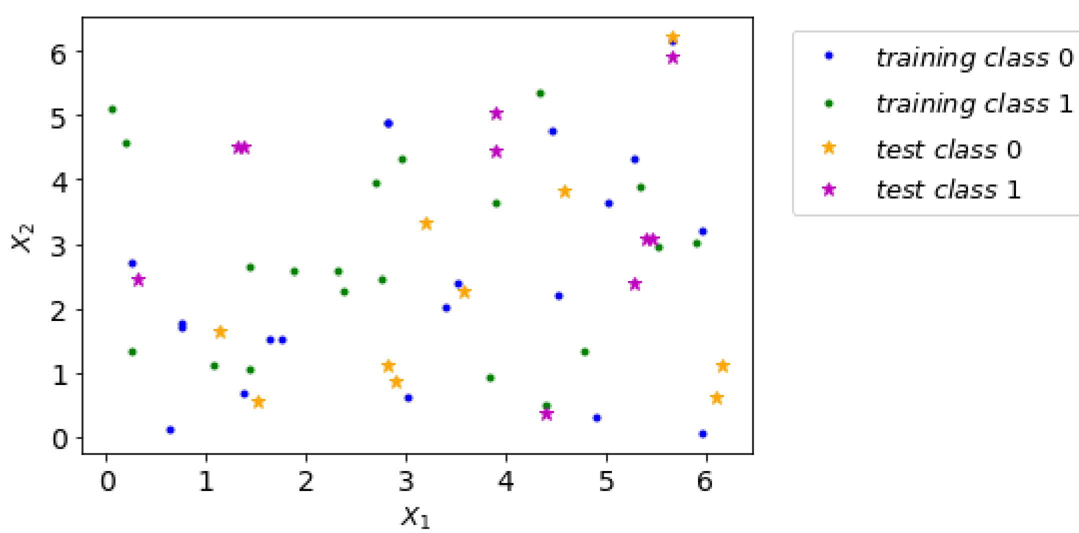
In [8]:
```
datapoints_train[0][1][0]
```

Out[8]:
```
0.25132741228718347
```

In [52]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
%matplotlib inline

#plt.rcParams.update({'font.size': 14})
mpl.rc('xtick', labelsize=14)
mpl.rc('ytick', labelsize=14)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 13}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
rom~ curve$')



for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==0 :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
```

```
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                 #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)

plt.plot(x0_train, y0_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='b', label= r'$training ~class~ 0$')

plt.plot(x1_train, y1_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='green', label= r'$training ~class~ 1$')

plt.plot(x0_test, y0_test, linewidth = 3, linestyle='', marker='*', markersize
=7, color='orange', label= r'$test ~class~ 0$')

plt.plot(x1_test, y1_test, linewidth = 3, linestyle='', marker='*', markersize
=7, color='m', label= r'$test ~class~ 1$')

plt.ylabel(r'$X_2$')
plt.xlabel(r'$X_1$')
#plt.legend()
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left")
plt.show()
```

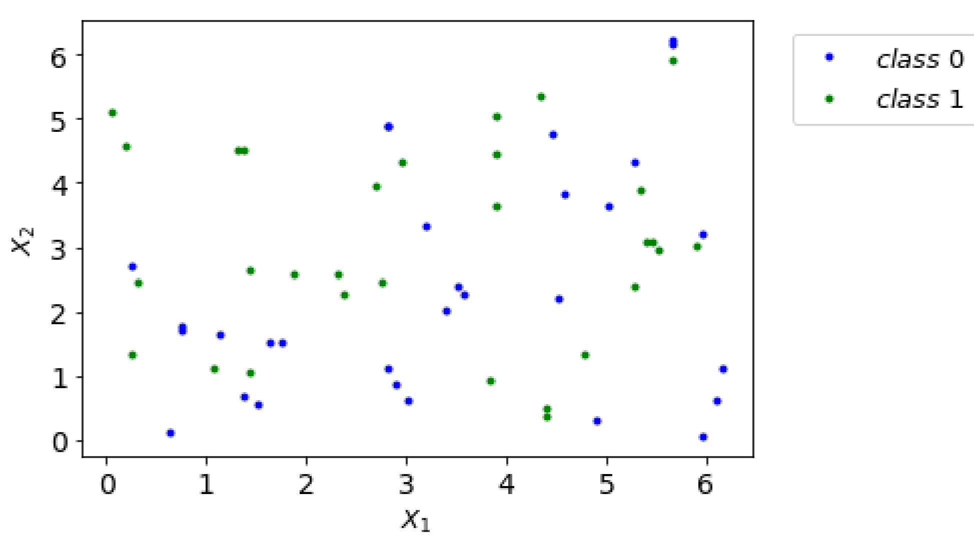
```
0
```



In [90]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
%matplotlib inline

#plt.rcParams.update({'font.size': 14})
mpl.rc('xtick', labelsize=14)
mpl.rc('ytick', labelsize=14)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 13}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
rom~ curve$')



for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==0 :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
```

```
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                  #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)

plt.plot(x0_train, y0_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='b', label= r'$class~ 0$')

plt.plot(x1_train, y1_train, linewidth = 3, linestyle='', marker='o', markersi
ze=3, color='green', label= r'$class~ 1$')

plt.plot(x0_test, y0_test, linewidth = 3, linestyle='', marker='o', markersize
=3, color='b')

plt.plot(x1_test, y1_test, linewidth = 3, linestyle='', marker='o', markersize
=3, color='g')

plt.ylabel(r'$X_2$')
plt.xlabel(r'$X_1$')
#plt.legend()
plt.legend(bbox_to_anchor=(1.04,1), loc="upper left")
plt.show()
```

```
0
```




In [91]:
```
print(len(datapoints[0]), len(datapoints_train[0]))
```

```
20 40
```

In [ ]:
```

```

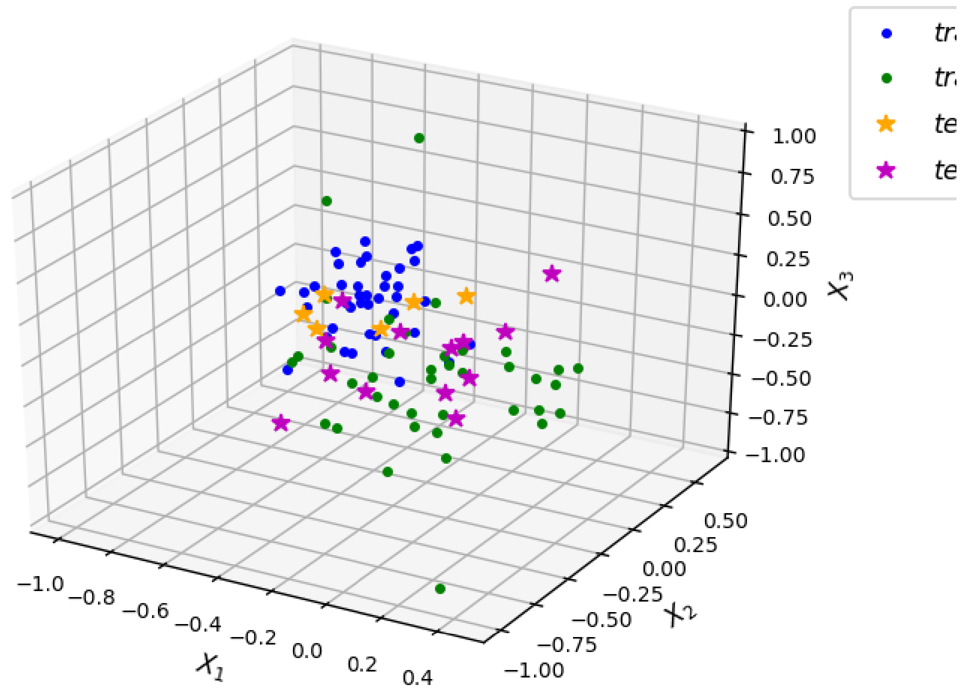
# dataset generator

In [1]:
```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA
```

In [2]:
```
# -*- coding: utf-8 -*-

# This code is part of Qiskit.
#
# (C) Copyright IBM 2018, 2020.
#
# This code is licensed under the Apache License, Version 2.0. You may
# obtain a copy of this license in the LICENSE.txt file in the root directory
# of this source tree or at http://www.apache.org/licenses/LICENSE-2.0.
#
# Any modifications or derivative works of this code must retain this
# copyright notice, and modified files need to carry a notice indicating
# that they have been altered from the originals.

"""
wine dataset
"""

import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA


def Wine(training_size, test_size, n, plot_data=False):
    """ returns wine dataset """
    class_labels = [r'A', r'B']

    data, target = datasets.load_wine(return_X_y=True)
    sample_train, sample_test, label_train, label_test = \
        train_test_split(data, target, test_size=test_size, random_state=7)

    # Now we standardize for gaussian around 0 with unit variance
    std_scale = StandardScaler().fit(sample_train)
    sample_train = std_scale.transform(sample_train)
    sample_test = std_scale.transform(sample_test)

    # Now reduce number of features to number of qubits
    pca = PCA(n_components=n).fit(sample_train)
    sample_train = pca.transform(sample_train)
    sample_test = pca.transform(sample_test)

    # Scale to the range (-1,+1)
    samples = np.append(sample_train, sample_test, axis=0)
    minmax_scale = MinMaxScaler((-1, 1)).fit(samples)
    sample_train = minmax_scale.transform(sample_train)
    sample_test = minmax_scale.transform(sample_test)
    # Pick training size number of samples from each distro
    training_input = {key: (sample_train[label_train == k, :])[:training_size]
                      for k, key in enumerate(class_labels)}
    test_input = {key: (sample_test[label_test == k, :])[:test_size]
                  for k, key in enumerate(class_labels)}

    if plot_data:
        try:
```

```
            import matplotlib.pyplot as plt
        except ImportError:
            raise NameError('Matplotlib not installed. Please install it befor
e plotting')
        for k in range(0, 2):
            plt.scatter(sample_train[label_train == k, 0][:training_size],
                        sample_train[label_train == k, 1][:training_size])

        plt.title("PCA dim. reduced Wine dataset")
        plt.show()

    return sample_train, training_input, test_input, class_labels
```

In [6]:
```
feature_dim = 3  # dimension of each data point
_, training_input, test_input, _ = Wine(training_size=20,
                                        test_size=20,
                                        n=feature_dim, plot_data=True)
print(training_input)
print(test_input)
```



tr
tr
te
te
$X_1$
$X_2$
$X_3$
−1.0 −0.8 −0.6 −0.4 −0.2 0.0 0.2 0.4
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

```
{'A': array([[ 0.79793355,  0.28187479, -0.33128971],
       [ 0.39343332, -0.08427555,  0.1119447 ],
       [ 0.40199003,  0.14214928, -0.32119205],
       [ 0.27183333, -0.03532283, -0.36099067],
       [ 0.24359139,  0.02226045, -0.086938  ],
       [ 0.49667755,  0.1744329 ,  0.20981907],
       [ 0.70254238,  0.54086689, -0.22133241],
       [ 0.43420364,  0.427876  ,  0.06947044],
       [ 0.6303877 ,  0.46178047, -0.22549449],
       [ 0.62597698,  0.31349661, -0.19044496],
       [ 0.58631141,  0.21263363, -0.11429097],
       [ 0.39582237,  0.1746587 , -0.10207793],
       [ 0.14338649,  0.06138962, -0.2394541 ],
       [ 0.80350035,  0.41282754, -0.19714028],
       [ 0.46880614,  0.40148459, -0.04646423],
       [ 0.57864998,  0.2269022 , -0.44634692],
       [ 0.51378791,  0.04245748, -0.30134363],
       [ 0.86872281,  0.71620827, -0.13805224],
       [ 0.57356045,  0.47232805, -0.15017466],
       [ 0.48668976,  0.25861296, -0.28062803]]), 'B': array([[ 0.39566165, -
0.33601802,  0.01623666],
       [ 0.50709816, -0.48024286, -0.4931085 ],
       [-0.29829571, -0.12223256,  0.53691483],
       [-0.15975575, -0.09061578,  0.19063522],
       [-0.21009715, -0.56658845,  0.08957511],
       [-0.13901377, -0.46758841,  0.18079584],
       [-0.0515843 , -0.51872856,  0.01250765],
       [ 0.21331498, -0.66560621, -0.2869653 ],
       [ 0.26349365, -0.2046635 , -0.32192802],
       [ 0.55954931,  0.03620748,  0.08309666],
       [-0.10805067, -0.54312622, -0.17086318],
       [ 0.45340538, -0.49200151, -0.07622572],
       [-0.13834845, -0.60927447,  0.20577842],
       [ 0.18667053, -0.18940068,  0.03506702],
       [ 0.19630018, -0.32194567,  0.22105999],
       [ 0.30233434, -0.54329773,  0.09000009],
       [-0.38262795, -0.36618072, -0.25154917],
       [-0.287466  , -0.44351986,  0.11928004],
       [-0.39623564, -0.31638676, -0.02940921],
       [ 0.49857052, -0.36137888, -0.12986118]])}
{'A': array([[ 0.30240709,  0.16176753,  0.01477704],
       [ 0.36476184,  0.00712627, -0.10879576],
       [ 0.57423018, -0.0377669 , -0.13044287],
       [ 0.65746851,  0.18259492, -0.07586862],
       [ 0.25693157,  0.4744081 , -0.07869493]]), 'B': array([[-0.01790808, -
0.28740798,  0.04495781],
       [-0.06635594, -0.2766249 ,  0.1262019 ],
       [ 0.13701037, -0.33814473,  0.14955472],
       [ 0.08185247, -1.        ,  0.22604611],
       [-0.15845642, -0.55249143, -0.14102704],
       [ 0.15849002, -0.53340888, -0.1145367 ],
       [-0.09236679, -0.47962149, -0.05856293],
       [ 0.33097935, -0.3636694 ,  0.28453884]])}
```

# qvc

In [7]:
```
from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import wine
from qiskit.circuit.library import TwoLocal

seed = 1376
aqua_globals.random_seed = seed

# Use Wine data set for training and test data
feature_dim = 3  # dimension of each data point
_, training_input, test_input, _ = Wine(training_size=40,
                                        test_size=30,
                                        n=feature_dim)
#print(training_input)
#print(test_input)
feature_map = RawFeatureVector(feature_dimension=feature_dim)
vqc = VQC(COBYLA(maxiter=100),
          feature_map,
          TwoLocal(feature_map.num_qubits, ['ry', 'rz'], 'cz', reps=3),
          training_input,
          test_input)
result = vqc.run(QuantumInstance(BasicAer.get_backend('statevector_simulator'
),
                                 shots=1024, seed_simulator=seed, seed_transpi
ler=seed))

print('Testing accuracy: {:0.2f}'.format(result['testing_accuracy']))
```

```
Testing accuracy: 0.84
```

In [8]:
```
from qiskit import Aer
from qiskit.aqua.utils import split_dataset_to_data_and_labels
```

In [9]:
```
datapoints, class_to_label = split_dataset_to_data_and_labels(test_input, clas
s_names=None)
print("actual class:", datapoints[1])
```

```
actual class: [0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [10]:
```
predicted_probs, predicted_labels = vqc.predict(datapoints[0])
#predicted_classes = map_label_to_class_name(predicted_labels, vqc.label_to_cl
ass)
print("prediction:   {}".format(predicted_labels))
```

```
prediction:   [0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1]
```

In [11]:
```
l=len(predicted_labels)
l
#datapoints[1][13]

for i in range(0,l):
    print(datapoints[0][i], 'actual class', datapoints[1][i], 'predicted', pre
dicted_labels[i])
    #if datapoints[1][i] == predicted_labels[i]:
```

```
[-0.3106672   0.12257661 -0.00159093] actual class 0 predicted 0
[-0.35676171 -0.03458755 -0.10368263] actual class 0 predicted 0
[-0.56922088 -0.08888029 -0.14532   ] actual class 0 predicted 0
[-0.65562359  0.12632826 -0.08121702] actual class 0 predicted 0
[-0.2715842   0.43525216 -0.12280559] actual class 0 predicted 1
[-0.78448856  0.21788693 -0.30188076] actual class 0 predicted 0
[ 0.0338409  -0.29331844  0.06459403] actual class 1 predicted 1
[ 0.0730454  -0.28451819  0.10873265] actual class 1 predicted 1
[-0.12415542 -0.35235832  0.13485989] actual class 1 predicted 1
[-0.04005161 -1.          0.27533628] actual class 1 predicted 1
[ 0.18616375 -0.56251029 -0.15607619] actual class 1 predicted 1
[-0.14351306 -0.55893384 -0.11592039] actual class 1 predicted 1
[ 0.10816375 -0.48786503 -0.07013531] actual class 1 predicted 1
[-0.31612966 -0.38608836  0.27987904] actual class 1 predicted 1
[-0.3891037  -0.36775187 -0.00605781] actual class 1 predicted 1
[-0.47708695 -0.51904108 -0.46247106] actual class 1 predicted 1
[ 0.30052158 -0.11142772  0.50769365] actual class 1 predicted 0
[ 0.13988346 -0.11331383  0.10356534] actual class 1 predicted 0
[ 0.23220208 -0.56880888  0.11121532] actual class 1 predicted 1
```

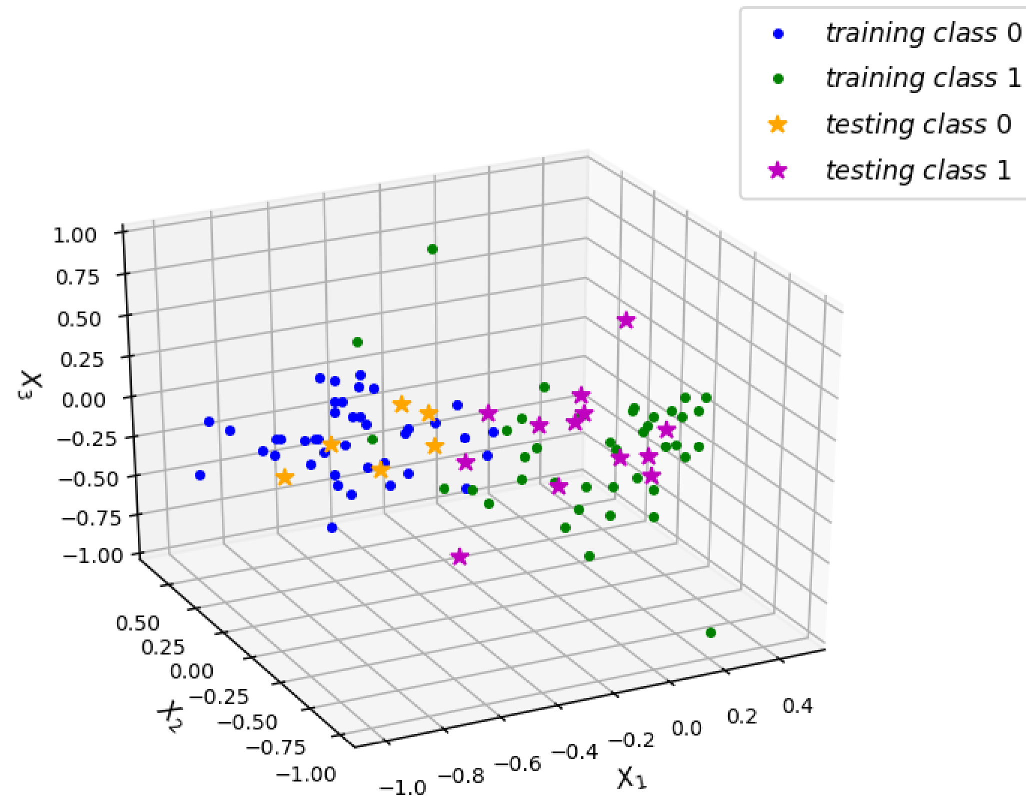
In [12]:
```
datapoints_train, class_to_label_train = split_dataset_to_data_and_labels(trai
ning_input, class_names=None)
print(datapoints_train[1])
#print(datapoints_train[0])
print(len((datapoints_train[1])))
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1]
80
```

In [18]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
#%matplotlib inline
%matplotlib notebook
from mpl_toolkits import mplot3d

plt.rcParams.update({'font.size': 6})
mpl.rc('xtick', labelsize=8)
mpl.rc('ytick', labelsize=6)
#mpl.rc('ztick', labelsize=8)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 10}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

z0_train=[]
z1_train=[]
z0_test=[]
z1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])
        z0_train.append(datapoints_train[0][i][2])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        z1_train.append(datapoints_train[0][i][2])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
rom~ curve$')
```

```
for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==0 :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        z0_test.append(datapoints[0][i][2])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        z1_test.append(datapoints[0][i][2])

        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                 #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)
fig=plt.figure(1)
ax = plt.axes(projection='3d')
#ax.scatter3D(x0_train, y0_train, z0_train, linewidth = 3, linestyle='', marke
r='o', markersize=3, color='b', label= r'$training ~class~ 0$')
plt.plot(x0_train, y0_train, z0_train, linewidth = 3, linestyle='', marker='o'
, markersize=3, color='b', label= r'$training ~class~ 0$')

plt.plot(x1_train, y1_train, z1_train, linewidth = 3, linestyle='', marker='o'
, markersize=3, color='green', label= r'$training ~class~ 1$')

plt.plot(x0_test, y0_test, z0_test, linewidth = 3, linestyle='', marker='*', m
arkersize=7, color='orange', label= r'$testing ~class~ 0$')

plt.plot(x1_test, y1_test,z1_test, linewidth = 3, linestyle='', marker='*', ma
rkersize=7, color='m', label= r'$testing ~class~ 1$')

ax.set_xlabel(r'$X_1$')
ax.set_ylabel(r'$X_2$')
ax.set_zlabel(r'$X_3$')
fig.legend()
#plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.show()
```

```
0
```

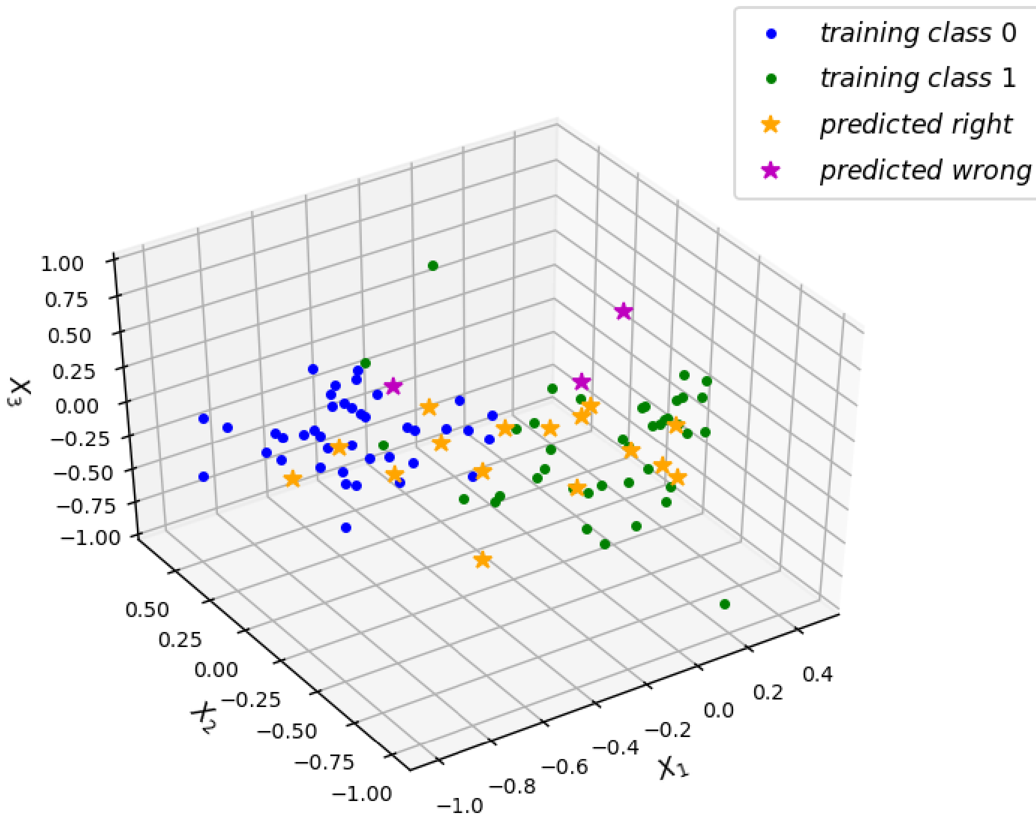


training class 0
training class 1
testing class 0
testing class 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
$X_3$
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
$X_2$
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
$X_1$

In [22]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
#%matplotlib inline
from mpl_toolkits import mplot3d
from mpl_toolkits.mplot3d import axes3d

%matplotlib notebook

mpl.rcParams.update({'font.size': 8})
mpl.rc('xtick', labelsize=8)
mpl.rc('ytick', labelsize=8)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 10}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

z0_train=[]
z1_train=[]
z0_test=[]
z1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])
        z0_train.append(datapoints_train[0][i][2])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        z1_train.append(datapoints_train[0][i][2])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
```

```
rom~ curve$')

for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==predicted_labels[i] :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        z0_test.append(datapoints[0][i][2])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        z1_test.append(datapoints[0][i][2])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                 #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)
fig=plt.figure(1)
#ax = fig.gca(projection='3d')
ax = fig.add_subplot(111, projection='3d')

ax.plot(x0_train, y0_train, z0_train, linewidth = 3, linestyle='', marker='o',
markersize=3, color='b', label= r'$training ~class~ 0$')

ax.plot(x1_train, y1_train,z1_train, linewidth = 3, linestyle='', marker='o',
markersize=3, color='green', label= r'$training ~class~ 1$')

ax.plot(x0_test, y0_test,z0_test, linewidth = 3, linestyle='', marker='*', mar
kersize=7, color='orange', label= r'$predicted~ right$')

plt.plot(x1_test, y1_test,z1_test, linewidth = 3, linestyle='', marker='*', ma
rkersize=7, color='m', label= r'$predicted ~wrong$')

ax.set_xlabel(r'$X_1$')
ax.set_ylabel(r'$X_2$')
ax.set_zlabel(r'$X_3$')
fig.legend()
#plt.legend(bbox_to_anchor=(1,1), loc="upper left")
#plt.legend(loc='best', bbox_to_anchor=(0.87, 0.5, 0.5, 0.5))
plt.show()
```

```
0
```



training class 0
training class 1
predicted right
predicted wrong
X1
X2
X3


In [ ]:

# qsvm

In [21]:
```
from qiskit import BasicAer
from qiskit.aqua import QuantumInstance, aqua_globals
from qiskit.aqua.algorithms import VQC
from qiskit.aqua.components.optimizers import COBYLA
from qiskit.aqua.components.feature_maps import RawFeatureVector
from qiskit.ml.datasets import ad_hoc_data
from qiskit.circuit.library import TwoLocal
from qiskit.aqua.components.feature_maps import SecondOrderExpansion
from qiskit.aqua.algorithms import QSVM

seed =  10598
aqua_globals.random_seed = seed

feature_dim = 2 # dimension of each data point
training_dataset_size = 20
testing_dataset_size = 10
#random_seed = 10598
#shots = 1024

#sample_Total, training_input, test_input, class_labels = ad_hoc_data(training
_size=training_dataset_size,
#                                                                      test_siz
e=testing_dataset_size,
#                                                                      n=featur
e_dim, gap=0.3, PLOT_DATA=True)

# Use Wine data set for training and test data
#feature_dim = 4  # dimension of each data point
# Use Wine data set for training and test data
feature_dim = 3  # dimension of each data point
_, training_input, test_input, _ = Wine(training_size=40,
                                        test_size=30,
                                        n=feature_dim)
#print(training_input)
#print(test_input)

#feature_map = RawFeatureVector(feature_dimension=feature_dim)
#feature_map =SecondOrderExpansion(feature_dimension=feature_dim, depth=2)
feature_map = SecondOrderExpansion(feature_dimension=feature_dim, depth=2, ent
angler_map=None, entanglement='full')

qsvmm = QSVM(feature_map, training_dataset=training_input, test_dataset=test_i
nput,
           datapoints=None, multiclass_extension=None, quantum_instance=None)

resultqsvm = qsvmm.run(QuantumInstance(BasicAer.get_backend('qasm_simulator'),
                                 shots=1024, seed_simulator=seed, seed_transpi
ler=seed))

print('Testing accuracy: {:0.2f}'.format(resultqsvm['testing_accuracy']))
#print(resultresultqsvm)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:36: Deprecat
ionWarning: The qiskit.aqua.components.feature_maps.SecondOrderExpansion obje
ct is deprecated as of 0.7.0 and will be removed no sooner than 3 months afte
r the release. You should use qiskit.circuit.library.ZZFeatureMap instead.
C:\ProgramData\Anaconda3\lib\site-packages\qiskit\aqua\components\feature_map
s\second_order_expansion.py:59: DeprecationWarning: The qiskit.aqua.component
s.feature_maps.PauliZExpansion class is deprecated as of 0.7.0 and will be re
moved no sooner than 3 months after the release. You should use qiskit.circui
t.library.PauliFeatureMap instead.
  z_order=2, data_map_func=data_map_func)
C:\ProgramData\Anaconda3\lib\site-packages\qiskit\aqua\components\feature_map
s\pauli_z_expansion.py:71: DeprecationWarning: The qiskit.aqua.components.fea
ture_maps.PauliExpansion object is deprecated as of 0.7.0 and will be removed
no sooner than 3 months after the release. You should use qiskit.circuit.libr
ary.PauliFeatureMap instead.
  paulis=pauli_string, data_map_func=data_map_func)
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:39: Deprecat
ionWarning:
                The <class 'qiskit.aqua.components.feature_maps.second_order_
expansion.SecondOrderExpansion'> object as input for the QSVM is deprecated a
s of 0.7.0 and will
                be removed no earlier than 3 months after the release.
                You should pass a QuantumCircuit object instead.
                See also qiskit.circuit.library.data_preparation for a collec
tion
                of suitable circuits.

Testing accuracy: 0.89
```

In [23]:

```
from qiskit import Aer
from qiskit.aqua.utils import split_dataset_to_data_and_labels
```

In [24]:

```
datapoints, class_to_label = split_dataset_to_data_and_labels(test_input, clas
s_names=None)
print(datapoints[0][18:])
#datapoints[0][2]
```

```
[[ 0.23220208 -0.56880888  0.11121532]]
```

In [25]:

```
import numpy as np
predicted_labels = qsvmm.predict(datapoints[0], quantum_instance=None)
#predicted_classes = map_label_to_class_name(predicted_labels, vqc.label_to_cl
ass)
print("prediction:   {}".format(predicted_labels))
```

```
prediction:   [0 0 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1]
```

In [26]:
```
for i in range(0,len(predicted_labels)):
    print(datapoints[0][i], 'actual class', datapoints[1][i], 'predicted', pre
dicted_labels[i])
```

```
[-0.3106672   0.12257661 -0.00159093] actual class 0 predicted 0
[-0.35676171 -0.03458755 -0.10368263] actual class 0 predicted 0
[-0.56922088 -0.08888029 -0.14532   ] actual class 0 predicted 0
[-0.65562359  0.12632826 -0.08121702] actual class 0 predicted 0
[-0.2715842   0.43525216 -0.12280559] actual class 0 predicted 0
[-0.78448856  0.21788693 -0.30188076] actual class 0 predicted 0
[ 0.0338409  -0.29331844  0.06459403] actual class 1 predicted 1
[ 0.0730454  -0.28451819  0.10873265] actual class 1 predicted 1
[-0.12415542 -0.35235832  0.13485989] actual class 1 predicted 1
[-0.04005161 -1.          0.27533628] actual class 1 predicted 0
[ 0.18616375 -0.56251029 -0.15607619] actual class 1 predicted 1
[-0.14351306 -0.55893384 -0.11592039] actual class 1 predicted 1
[ 0.10816375 -0.48786503 -0.07013531] actual class 1 predicted 1
[-0.31612966 -0.38608836  0.27987904] actual class 1 predicted 1
[-0.3891037  -0.36775187 -0.00605781] actual class 1 predicted 1
[-0.47708695 -0.51904108 -0.46247106] actual class 1 predicted 1
[ 0.30052158 -0.11142772  0.50769365] actual class 1 predicted 1
[ 0.13988346 -0.11331383  0.10356534] actual class 1 predicted 0
[ 0.23220208 -0.56880888  0.11121532] actual class 1 predicted 1
```

In [27]:
```
datapoints_train, class_to_label_train = split_dataset_to_data_and_labels(trai
ning_input, class_names=None)
print(datapoints_train[1])
#print(datapoints_train[0])
print(len((datapoints_train[1])))
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1]
80
```

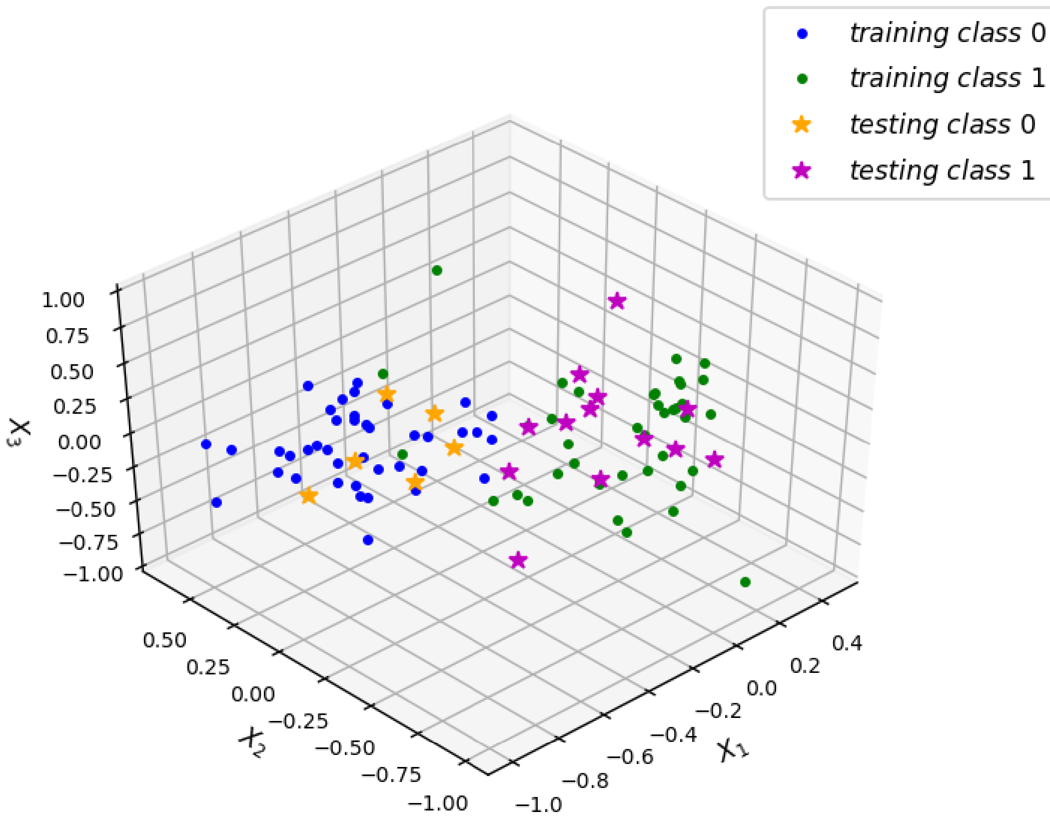
In [28]:
```
datapoints_train[0][1][0]
```

Out[28]:
```
-0.3991818439242424
```

In [30]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
#%matplotlib inline
%matplotlib notebook
from mpl_toolkits import mplot3d

plt.rcParams.update({'font.size': 6})
mpl.rc('xtick', labelsize=8)
mpl.rc('ytick', labelsize=6)
#mpl.rc('ztick', labelsize=8)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 10}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

z0_train=[]
z1_train=[]
z0_test=[]
z1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])
        z0_train.append(datapoints_train[0][i][2])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        z1_train.append(datapoints_train[0][i][2])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
rom~ curve$')
```

```
for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==0 :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        z0_test.append(datapoints[0][i][2])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        z1_test.append(datapoints[0][i][2])

        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                 #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)
fig=plt.figure(1)
ax = plt.axes(projection='3d')
#ax.scatter3D(x0_train, y0_train, z0_train, linewidth = 3, linestyle='', marke
r='o', markersize=3, color='b', label= r'$training ~class~ 0$')
plt.plot(x0_train, y0_train, z0_train, linewidth = 3, linestyle='', marker='o'
, markersize=3, color='b', label= r'$training ~class~ 0$')

plt.plot(x1_train, y1_train, z1_train, linewidth = 3, linestyle='', marker='o'
, markersize=3, color='green', label= r'$training ~class~ 1$')

plt.plot(x0_test, y0_test, z0_test, linewidth = 3, linestyle='', marker='*', m
arkersize=7, color='orange', label= r'$testing ~class~ 0$')

plt.plot(x1_test, y1_test,z1_test, linewidth = 3, linestyle='', marker='*', ma
rkersize=7, color='m', label= r'$testing ~class~ 1$')

ax.set_xlabel(r'$X_1$')
ax.set_ylabel(r'$X_2$')
ax.set_zlabel(r'$X_3$')
fig.legend()
#plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.show()
```

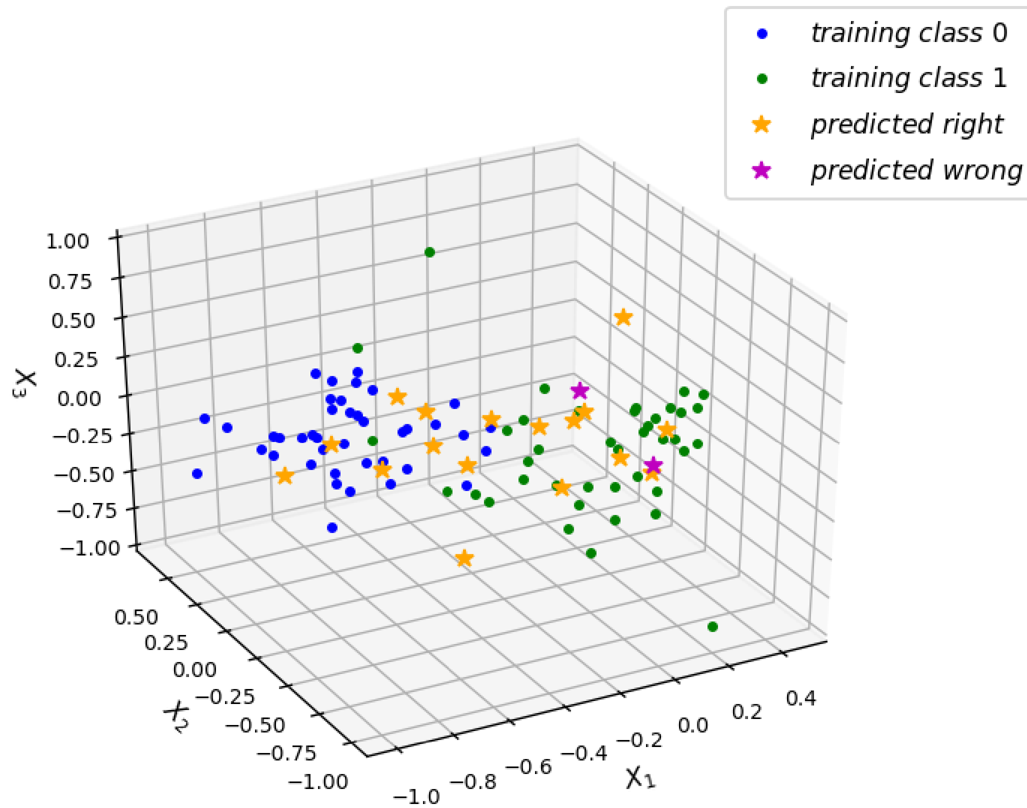
0


training class 0
training class 1
testing class 0
testing class 1
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
$X_3$
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
$X_2$
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
$X_1$

In [31]:
```
import numpy as np
import math
from scipy.optimize import minimize
import matplotlib as mpl
import matplotlib.pyplot as plt
from scipy.optimize import minimize, minimize_scalar
from scipy import integrate
#%matplotlib inline
%matplotlib notebook
from mpl_toolkits import mplot3d

plt.rcParams.update({'font.size': 8})
mpl.rc('xtick', labelsize=8)
mpl.rc('ytick', labelsize=8)
font = {'family' : 'normal',
        'weight' : 'normal',
        'size'   : 10}

mpl.rc('font', **font)

def example_inline():
    plt.clf()
x0_train=[]
y0_train=[]
x0_test=[]
y0_test=[]
x1_train=[]
y1_train=[]
x1_test=[]
y1_test=[]

z0_train=[]
z1_train=[]
z0_test=[]
z1_test=[]

#plt.figure(1)
for i in range(0, len(datapoints_train[0])):
    if datapoints_train[1][i]==0 :
        x0_train.append(datapoints_train[0][i][0])
        y0_train.append(datapoints_train[0][i][1])
        z0_train.append(datapoints_train[0][i][2])

        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
             #marker='o', markersize=3, color='blue', label= r'$Result ~from~
 curve~ fitting$')
    else:
        x1_train.append(datapoints_train[0][i][0])
        y1_train.append(datapoints_train[0][i][1])
        z1_train.append(datapoints_train[0][i][2])
        #plt.plot(datapoints_train[0][i][0], datapoints_train[0][i][1], linest
yle='', linewidth = 3,
                 #marker='o', markersize=3, color='green', label= r'$Result ~f
rom~ curve$')
```

```
for i in range(0, len(datapoints[0])):
    if datapoints[1][i]==predicted_labels[i] :
        x0_test.append(datapoints[0][i][0])
        y0_test.append(datapoints[0][i][1])
        z0_test.append(datapoints[0][i][2])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
             #marker='*', markersize=7, color='orange')
    else:
        x1_test.append(datapoints[0][i][0])
        y1_test.append(datapoints[0][i][1])
        z1_test.append(datapoints[0][i][2])
        #plt.plot(datapoints[0][i][0], datapoints[0][i][1], linestyle='', line
width = 3,
                 #marker='*', markersize=7, color='m')
    #plt.xlabel(r'$\mathbf{X_1}$')
    #plt.ylabel(r'$\mathbf{x_2}$')
#plt.legend()
    #plt.show()
print(len(x0_train)-len(y0_train))


plt.figure(1)
fig=plt.figure(1)
ax = plt.axes(projection='3d')

plt.plot(x0_train, y0_train, z0_train, linewidth = 3, linestyle='', marker='o'
, markersize=3, color='b', label= r'$training ~class~ 0$')

plt.plot(x1_train, y1_train,z1_train, linewidth = 3, linestyle='', marker='o',
markersize=3, color='green', label= r'$training ~class~ 1$')

plt.plot(x0_test, y0_test,z0_test, linewidth = 3, linestyle='', marker='*', ma
rkersize=7, color='orange', label= r'$predicted~ right$')

plt.plot(x1_test, y1_test,z1_test, linewidth = 3, linestyle='', marker='*', ma
rkersize=7, color='m', label= r'$predicted ~wrong$')

ax.set_xlabel(r'$X_1$')
ax.set_ylabel(r'$X_2$')
ax.set_zlabel(r'$X_3$')
fig.legend()
#plt.legend(bbox_to_anchor=(1,1), loc="upper left")
plt.show()
```

```
0
```




In [91]:
```
print(len(datapoints[0]), len(datapoints_train[0]))
```

```
20 40
```

In [ ]: