High Energy Physics and Quantum Computing

2024 IonQ 3rd week meeting

Weekly goal

[Week 1] Review the reference [1] thoroughly and understand basic idea.

https://github.com/HeechanYi/Data-Reuploading

[Week 2] Implement the idea using Qiskit and/or PennyLane, and reproduce their results for two examples in the paper.

[Week 3] Then consider a more complex and more realistic example, taking electron-positron production at the Large Hadron Collider (pp → e+e–). Effectively, this problem involves four-dimensional integration.

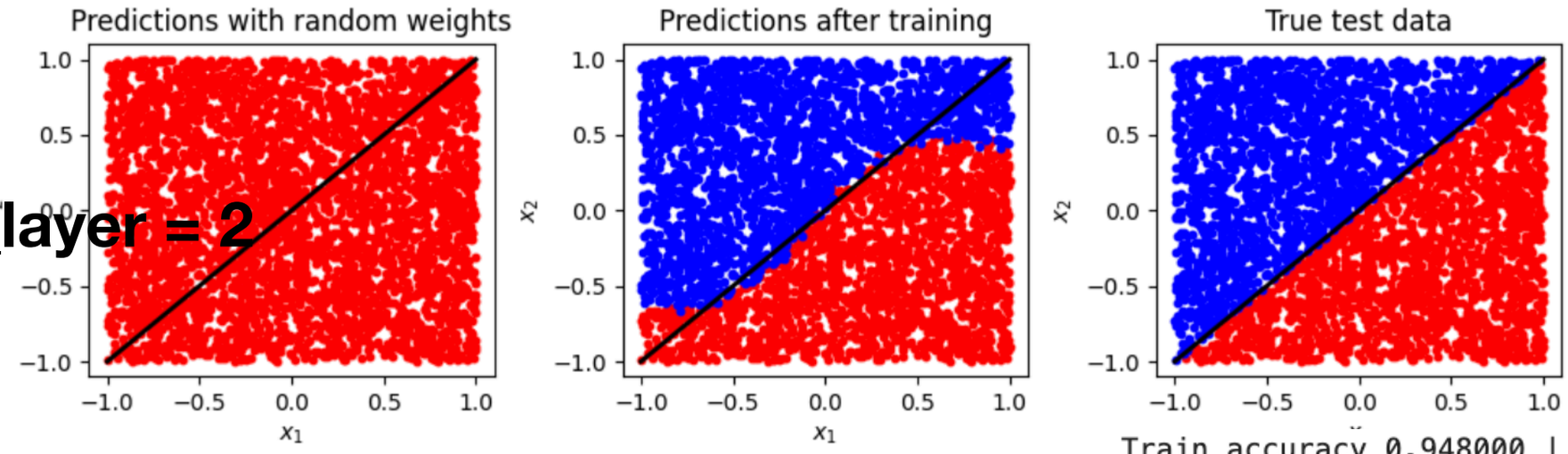
[Week 4] Try to optimize the circuits with different choice of the cost function or variations of quantum circuits.

[Week 5] Study Monte-Carlo sampling with above integration method, and how to implement the importance sampling into a variational quantum circuit. (Check Ref. [2])

Single Qubit Classifier Result Linear problem

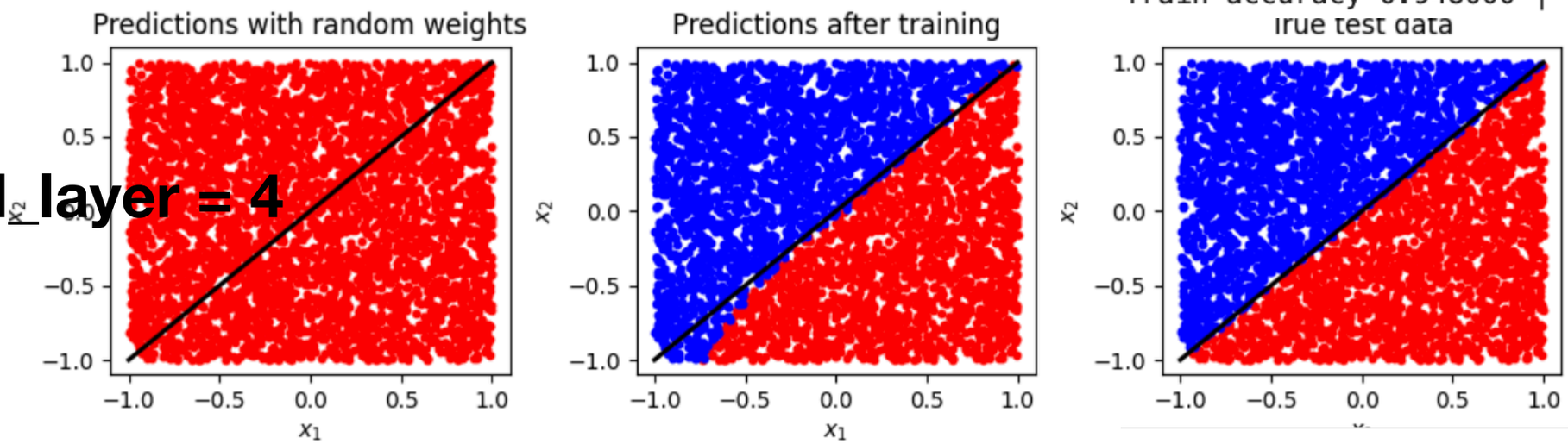
Number of Layers

N_layer = 2



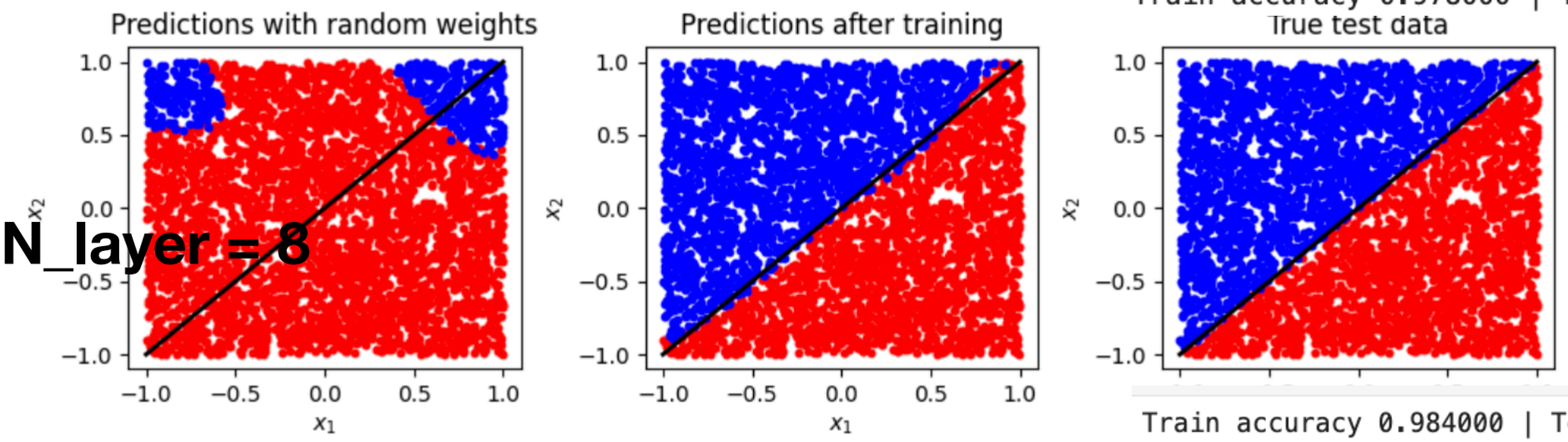

```
Train accuracy 0.948000 | Test Accuracy : 0.931000
```

N_layer = 4




```
Train accuracy 0.978000 | Test Accuracy : 0.972500
```

N_layer = 8



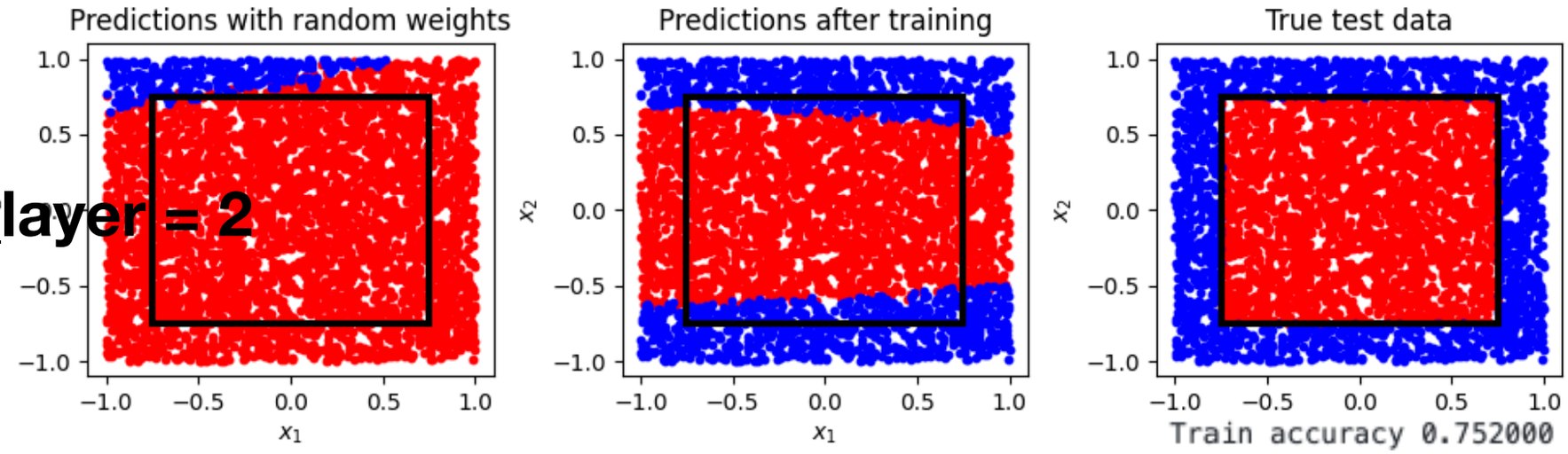

```
Train accuracy 0.984000 | Test Accuracy : 0.982000
```

Train data : 500, Test data : 4000, Learning rate : 0.1

Single Qubit Classifier Result Square problem

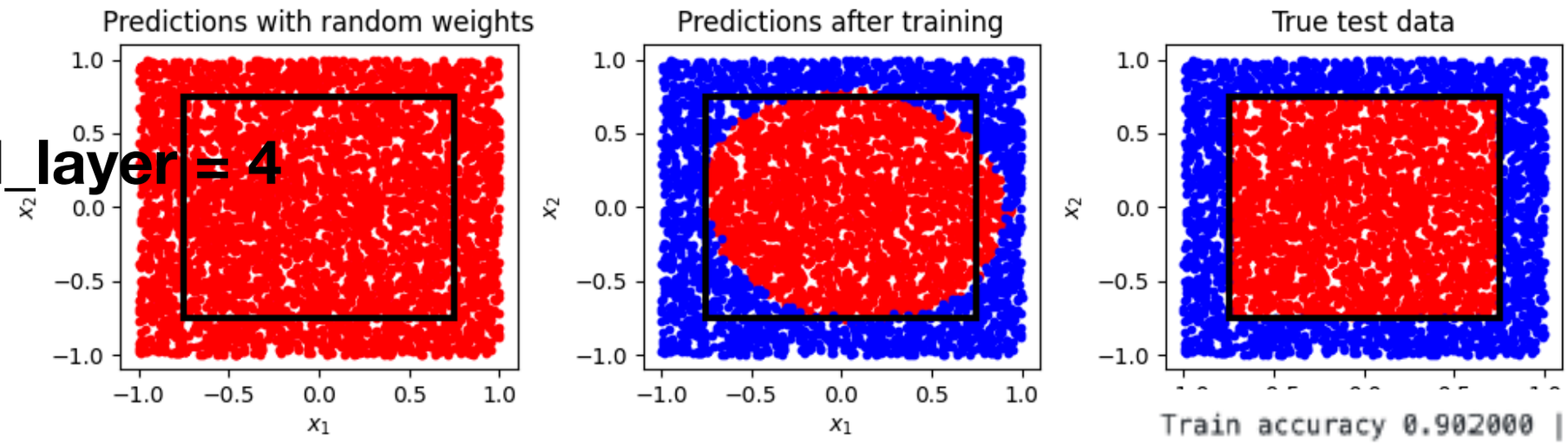
Number of Layers

N_layer = 2



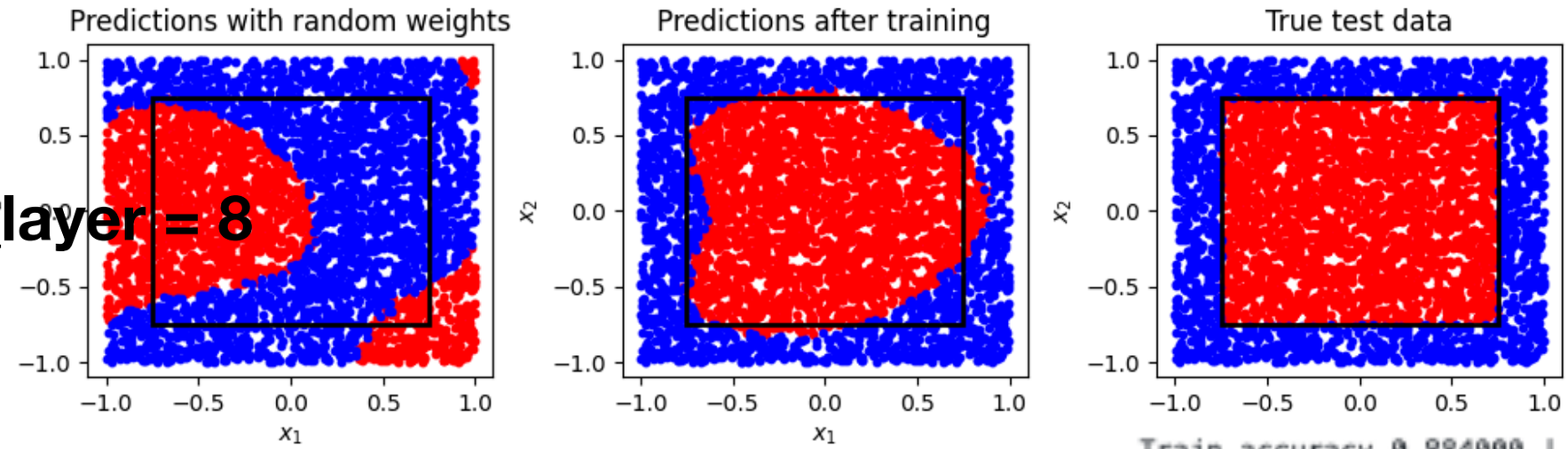
Train accuracy 0.752000 | Test Accuracy : 0.752250

N_layer = 4



Train accuracy 0.902000 | Test Accuracy : 0.902750

N_layer = 8



Train accuracy 0.884000 | Test Accuracy : 0.900250

Train data : 500, Test data : 4000, Learning rate : 0.1

Weekly goal

[Week 1] Review the reference [1] thoroughly and understand basic idea.

[Week 2] Implement the idea using Qiskit and/or **PennyLane**, and reproduce their results for two examples in the paper.

[Week 3] Then consider a more complex and more realistic example, taking electron-positron production at the Large Hadron Collider (pp → e+e–). Effectively, this problem involves four-dimensional integration.

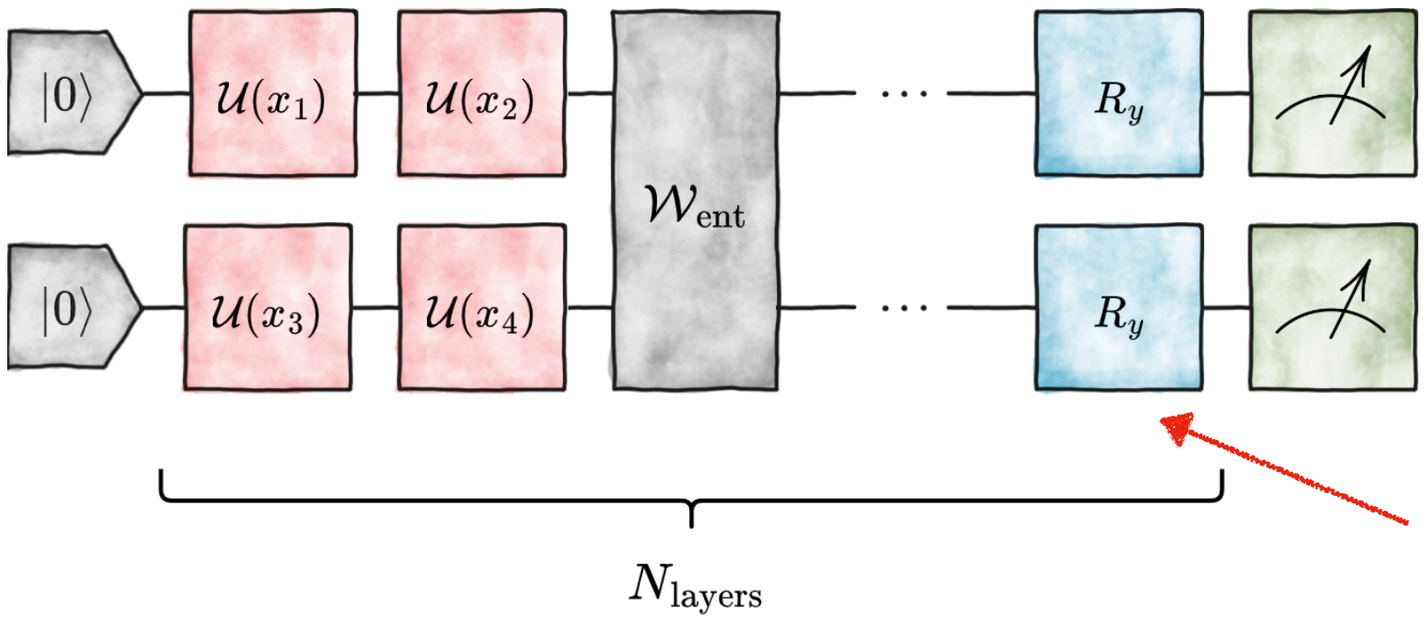
[Week 4] Try to optimize the circuits with different choice of the cost function or variations of quantum circuits.

[Week 5] Study Monte-Carlo sampling with above integration method, and how to implement the importance sampling into a variational quantum circuit. (Check Ref. [2])

Methodology

- Data reuploading model

$$\mathcal{U}(x|\theta) = R_z(\theta_1)R_y(\theta_2)R_z(\theta_3)R_z(\theta_4\, x)R_y(\theta_5)$$



Question? Why R_y?

The $U(x_i)$ quantum channel corresponds to the fundamental Fourier Gate, while the entangling channel W_{ent} is built with a combination of CZ gates.

```
GoodScaling (dims=4, nqubits=2, nlayers=2) running in process id: 1235043
Circuit drawing:

q0: -RY-RZ-RZ-RY-RZ-RY-RZ-RZ-RY-RZ-o-RY-RZ-RZ-RY-RZ-RY-RZ-RZ-RY-RZ-o-RY-M-
q1: -RY-RZ-RZ-RY-RZ-RY-RZ-RZ-RY-RZ-Z-RY-RZ-RZ-RY-RZ-RY-RZ-RZ-RY-RZ-Z-RY-M-
Circuit summary:
    Circuit depth = 24
Total number of gates = 46
Number of qubits = 2
Most common gates:
rz: 24
ry: 18
cz: 2
measure: 2
```

Methodology - training

$$I(\vec{\alpha};\vec{x}) = \int g(\vec{\alpha};\vec{x})\, d\vec{x}$$

Input data : $[x_1, x_2, x_3, \ldots, \alpha_0]$ ==(train the parameters θ)== Target data : $g = g(\vec{\alpha};\vec{x})$

After training

Get the integration values I

- **Training**: First, learn the parameters of the Variational Quantum Circuit (VQC) for the given objective function $g(\alpha; x)$.

The goal is to make the circuit approximate the function g.

In this process, we aim to minimize the Mean-Squared Error loss function, which finds the parameters of the circuit that provide the optimal prediction for each training data in g.

$$g_{j,\text{est}}(\boldsymbol{\alpha};\boldsymbol{x}_j|\boldsymbol{\theta}) = \left.\frac{\partial G(\boldsymbol{\alpha}, x_1, ..., x_n|\boldsymbol{\theta})}{\partial x_1 \ldots \partial x_n}\right|_{\boldsymbol{x}_j}$$

$$J_{\text{mse}} = \frac{1}{N_{\text{train}}} \sum_{j=1}^{N_{\text{train}}} \left[g_{j,\text{meas}} - g_{j,\text{est}}(\boldsymbol{\alpha};\boldsymbol{x}_j|\boldsymbol{\theta}) \right]^2$$

Methodology - test (Integration)

Evaluating analytic gradients on quantum hardware

Maria Schuld,* Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran
Xanadu Inc., 372 Richmond St W, Toronto, M5V 1X6, Canada
(Dated: November 29, 2018)

$$\partial_\mu f = r\big(\langle\psi|\mathcal{G}^\dagger(\mu+s)\hat{Q}\mathcal{G}(\mu+s)|\psi\rangle - \langle\psi|\mathcal{G}^\dagger(\mu-s)\hat{Q}\mathcal{G}(\mu-s)|\psi\rangle\big) = r\left(f(\mu+s)-f(\mu-s)\right).$$

a parameter $\mu \in \theta$

$$g(\mu) = \partial_\mu G(\theta) = r(G(\mu^+) - G(\mu^-))$$

$$G(x;\alpha) = \int g(\alpha;x)dx\,.$$

- **Integration**: Using the learned model, calculate the integral of g. Using Parameter Shift Rule (**PSR**) to calculate the derivative of a circuit, which allows us to calculate the indefinite integral of g ($G(\alpha;x)$).
That is, evaluate G as a circuit, calculate the derivative of g as PSR, and obtain the integral value.

$$I(\alpha) = G(x_b;\alpha) - G(x_a;\alpha) \qquad g_{i,est}(\alpha;x_j|\theta) = \left.\frac{\partial G(\alpha,x_1,\ldots,x_n|\theta)}{\partial x_1\ldots\partial x_n}\right|_{x_j}$$

How to derive?

$$I_{ab}(\ldots,x_{k-1},x_{k+1},\ldots) = \int_{x_{k,a}}^{x_{k,b}} g(x)dx_k = G(x_{k,b};\alpha) - G(x_{k,a};\alpha) \approx g_{est}(x_{k,b}|\theta_{best}) - g_{est}(x_{k,a}|\theta_{best})\,.$$

Target toy example

$$g(\vec{x}) = \cos(\vec{\alpha} \cdot \vec{x} + \alpha_0)$$

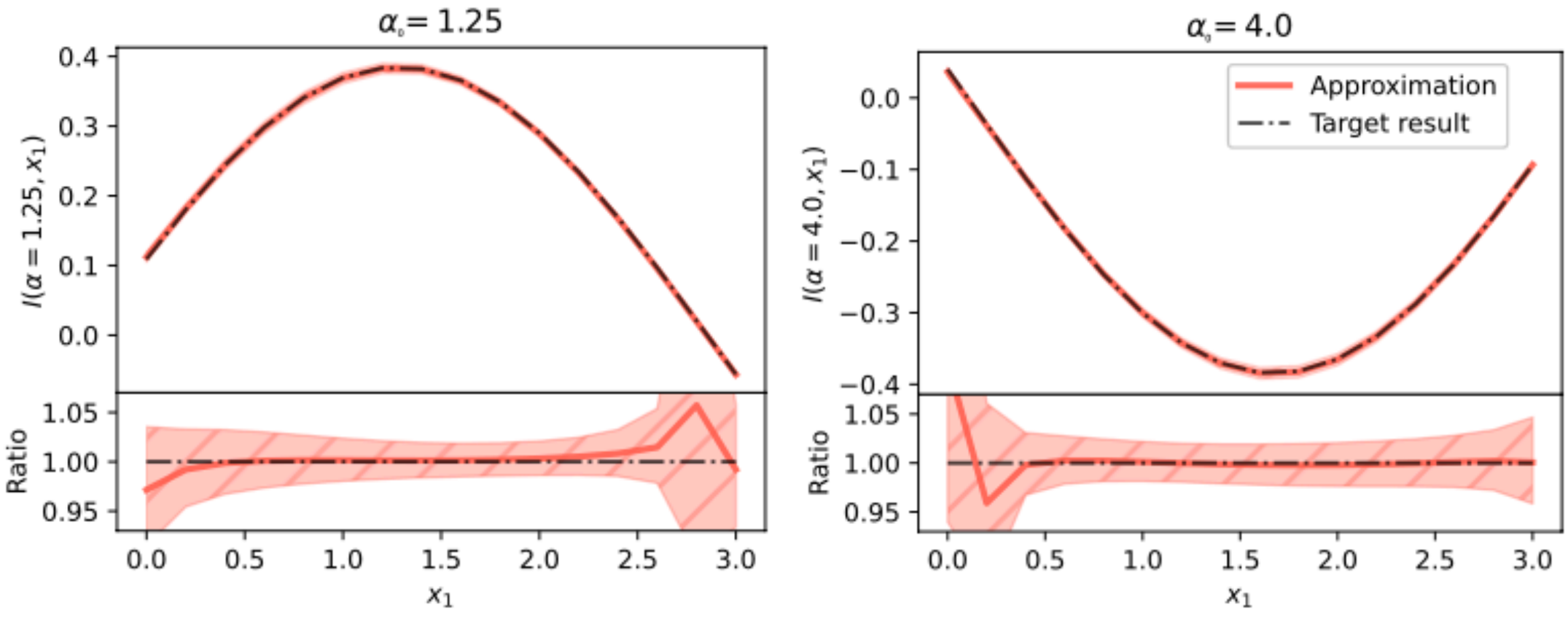
$$I(\vec{\alpha}; \vec{x}) = \int g(\vec{\alpha}; \vec{x})\, d\vec{x} = \int \cos(x_1 + 2x_2 + 0.5x_3 + \alpha_0) dx_1 dx_2 dx_3$$

Input data : $[x_1, x_2, x_3, \alpha_0]$

Target data : $g = \cos(x_1 + 2x_2 + 0.5x_3 + \alpha_0)$

Parameter	Value
$N_{x,\text{train}}$	100
α	$\{1, 2, 0.5\}$
N_{α_0}	10
N_{layers}	2
N_{params}	20
$\lvert I - \tilde{I} \rvert$	$4.4 \cdot 10^{-3}$
N_{shots}	Exact simulation
Optimizer	L-BFGS

$$I(\alpha_0; x_1)_{target} = [-\cos(\alpha_0 + x_1) + \cos(\alpha_0 + x_1 + 1.5) + \cos(\alpha_0 + x_1 + 6) - \cos(\alpha_0 + x_1 + 7.5)]_{x_1=0}^{x_1=3}$$

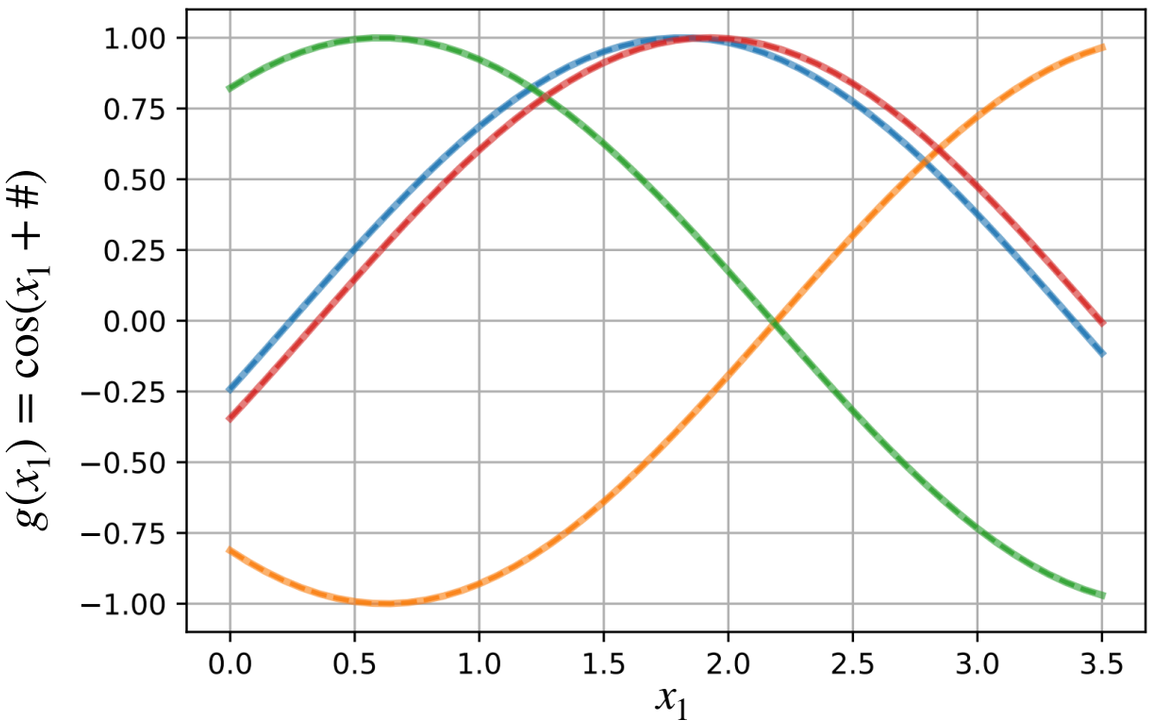


Target toy example

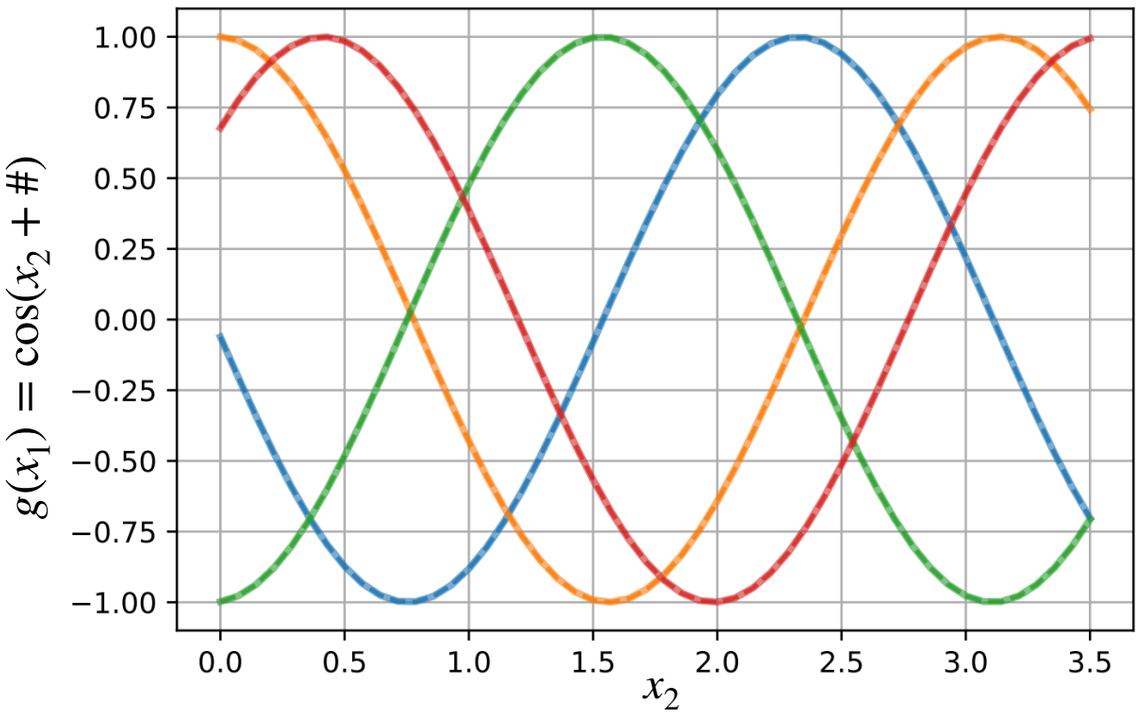
$$g(\vec{x}) = \cos(\vec{\alpha} \cdot \vec{x} + \alpha_0)$$

QIBO reproduce results

$\min = 0.0,\ \max = 3.0$



Input data is $[x_1,\ \text{rand}(\min, \max, \text{shape} = 3)]$
where $x_1 = \text{linspace}[\min, \max]$



Input data is $[x_2,\ \text{rand}(\min, \max, \text{shape} = 3)]$
where $x_2 = \text{linspace}[\min, \max]$

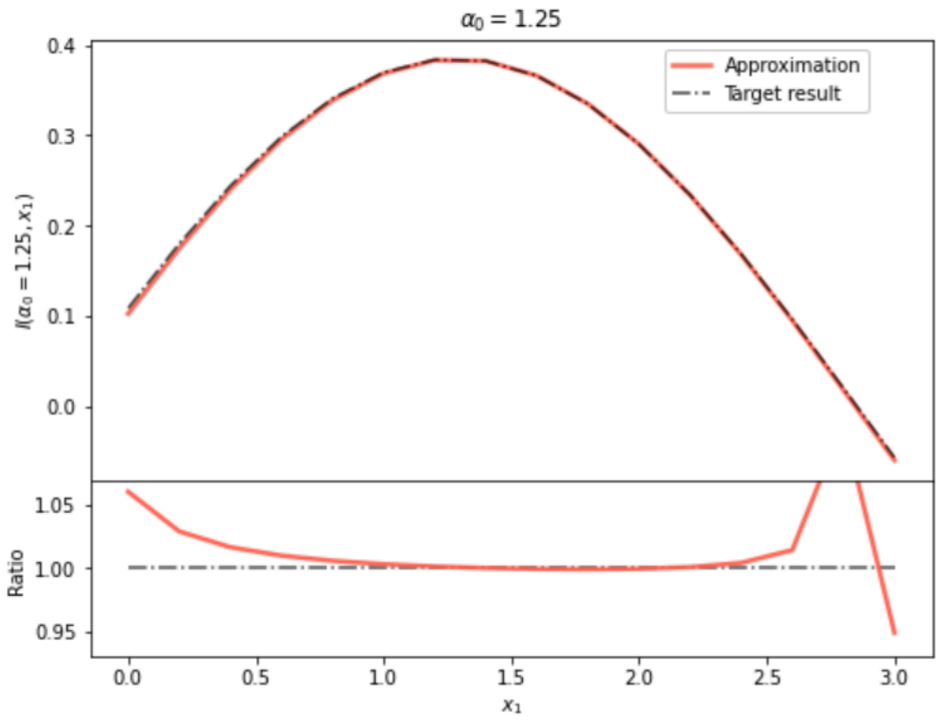
Same as x_1, x_2 we can draw it for x_3, α_0

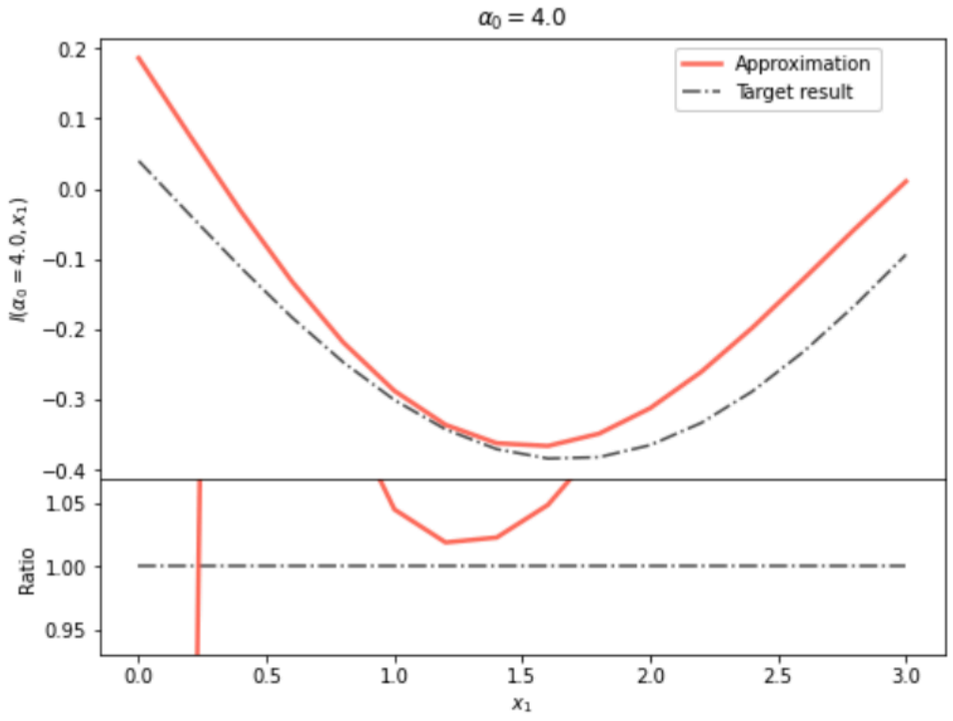
Target toy example

$$g(\vec{x}) = \cos(\vec{\alpha} \cdot \vec{x} + \alpha_0)$$

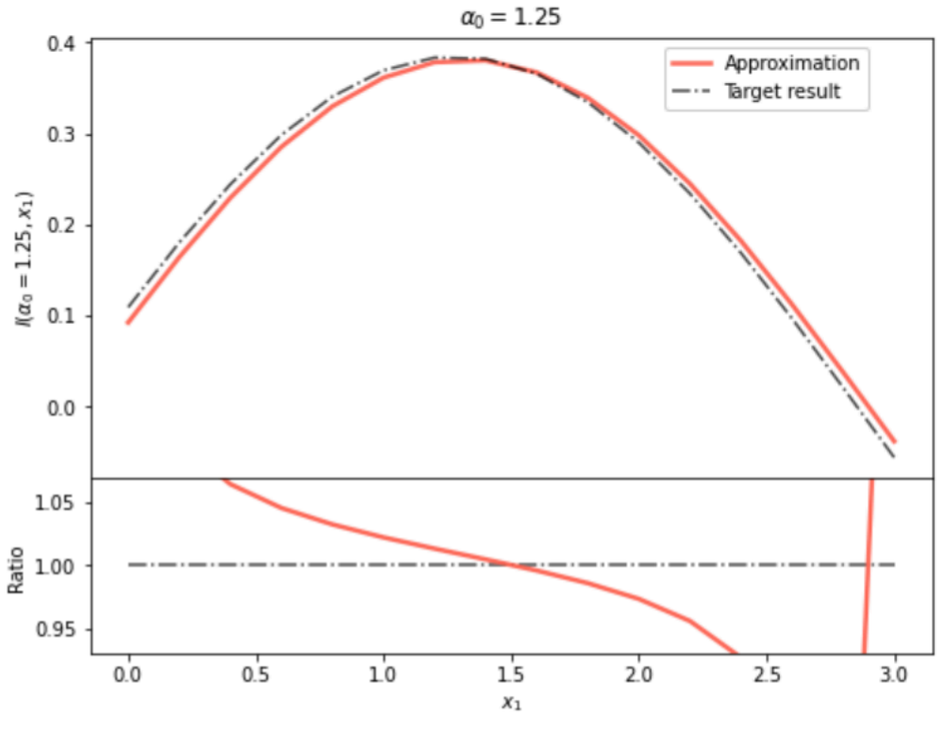
QIBO reproduce results

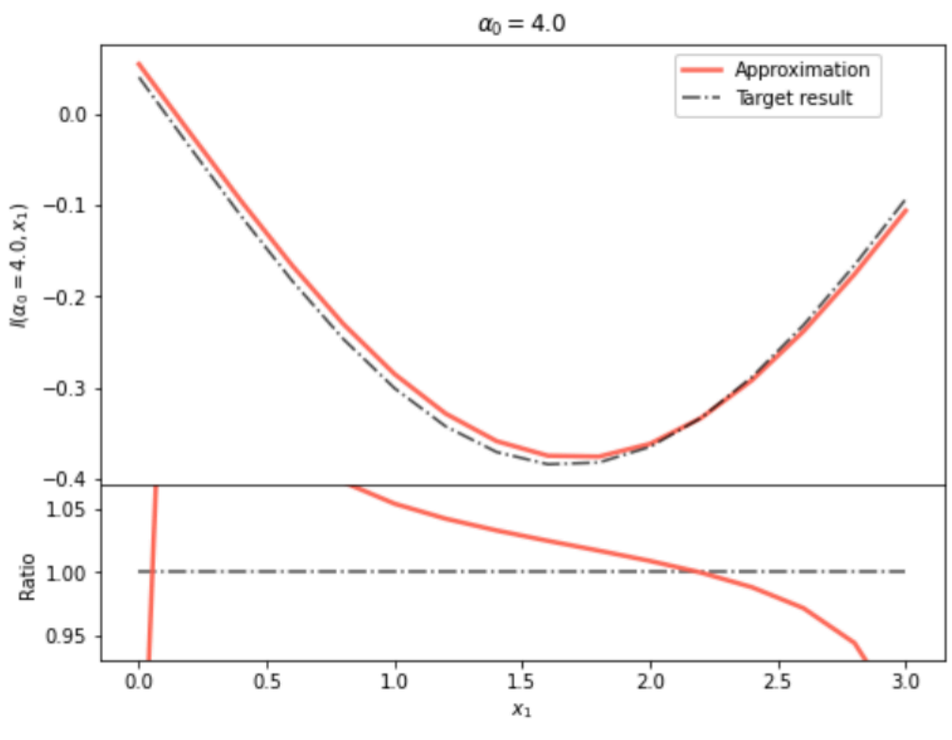
```
Plot saved to cos_on_x1_alpha1.25.pdf, average ratio: 1.0131339688248422
```






```
Plot saved to cos_on_x1_alpha1.25.pdf, average ratio: 1.0113928557656395
```





Pennylane implements

Data Generation

[2]:
```
data = np.load('toy_data_10e4.npz')
```

[3]:
```
Xdata = data['xdata']
print(Xdata)
```

```
[[2.55258737 0.41757623 2.5810235  0.        ]
 [1.05945455 2.73902844 2.59982768 0.01002004]
 [2.71202945 3.38179615 1.0272925  0.02004008]
 ...
 [2.26058831 2.62355914 0.18093676 4.97995992]
 [2.91426381 3.41772047 2.93088925 4.98997996]
 [0.1022243  2.0529486  1.72946409 5.        ]]
```

[4]:
```
Ydata = data['ydata']
print(Ydata)
```

```
[-0.03413077  0.00653628 -0.83397146 ...  0.9999308  -0.8789377
 -0.79724696]
```

Circuit Model

[7]:
```
dev_stochastic = qml.device("lightning.qubit", wires=10)

@qml.qnode(dev_stochastic, diff_method="parameter-shift")
def single_data_point_ansatz(params, phis, x, num_layers):
    """
    Quantum Circuit Model for a single data point x.

    INPUT
    params : array of theta parameters
    phis : array of phi for the last rotation gate
    x : single data point for reuploading

    OUTPUT
    Expectation value with PauliZ measure
    """

    num_qubits = math.ceil(len(x) / 2)
    param_index = 0

    for _ in range(num_layers):
        for i in range(len(x)):
            qubit = i // 2
            qml.RY(params[param_index], wires=qubit)
            qml.RZ(params[param_index + 1], wires=qubit)
            qml.RZ(x[i], wires=qubit)  # Re-uploading data here
            qml.RY(params[param_index + 2], wires=qubit)
            qml.RZ(params[param_index + 3], wires=qubit)
            param_index += 4

        if num_qubits > 1:
            for q in range(0, num_qubits - 1, 1):
                qml.CZ(wires=[q, q + 1])
            if num_qubits > 2:
                qml.CZ(wires=[num_qubits - 1, 0])

    for i in range(num_qubits):
        qml.RY(phis[i], wires=i)

    obs = qml.PauliZ(0)
    for i in range(num_qubits-1):
        obs = obs @ qml.PauliZ(i+1)

    return qml.expval(obs)
```

Still need more time...

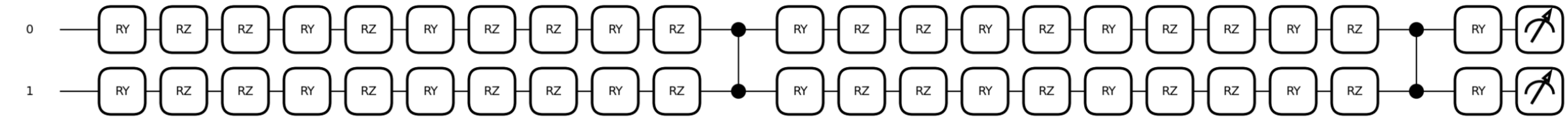
[8]:
```
xdim = 4
num_qubits = math.ceil(xdim / 2)
num_layers = 2

xdata = Xdata[:100] #np.random.uniform(size = [100,xdim], requires_grad=False)
thetas = np.random.uniform(size = xdim*num_layers*5,requires_grad = True)
phis = np.random.uniform(size=(num_qubits,1), requires_grad =True)
target = Ydata[:100]
print(thetas)

params = 0
params += thetas
param_index = 1
for _ in range(num_layers):
    for j in range(xdim):
        params[param_index] = thetas[param_index] * xdata[0][j]
        param_index += 5

fig, ax = qml.draw_mpl(single_data_point_ansatz)(params, phis, xdata[0], num_layers)
plt.show()
```

```
[0.17827215 0.57666515 0.67513005 0.81846239 0.93932721 0.26357393
 0.20275885 0.8592312  0.26100854 0.95916566 0.73823606 0.98867133
 0.76168891 0.55937236 0.12678064 0.04602907 0.97936243 0.6393217
 0.28326717 0.56462284 0.10534565 0.62418791 0.02774595 0.26683968
 0.78912964 0.00125882 0.32445339 0.46856635 0.12562493 0.47928415
 0.82211142 0.70636646 0.26782975 0.17742842 0.47312994 0.9278913
 0.11108655 0.2551302  0.12350515 0.27610254]
```



Weekly goal

[Week 1] Review the reference [1] thoroughly and understand basic idea.

[Week 2~3] Implement the idea using Qiskit and/or **PennyLane**, and reproduce their results for two examples in the paper.

[Week 4] Then consider a more complex and more realistic example, taking electron-positron production at the Large Hadron Collider (pp → e+e–). Effectively, this problem involves four-dimensional integration.

[Week 5] Try to optimize the circuits with different choice of the cost function or variations of quantum circuits.

[Week ?] Study Monte-Carlo sampling with above integration method, and how to implement the importance sampling into a variational quantum circuit. (Check Ref. [2])