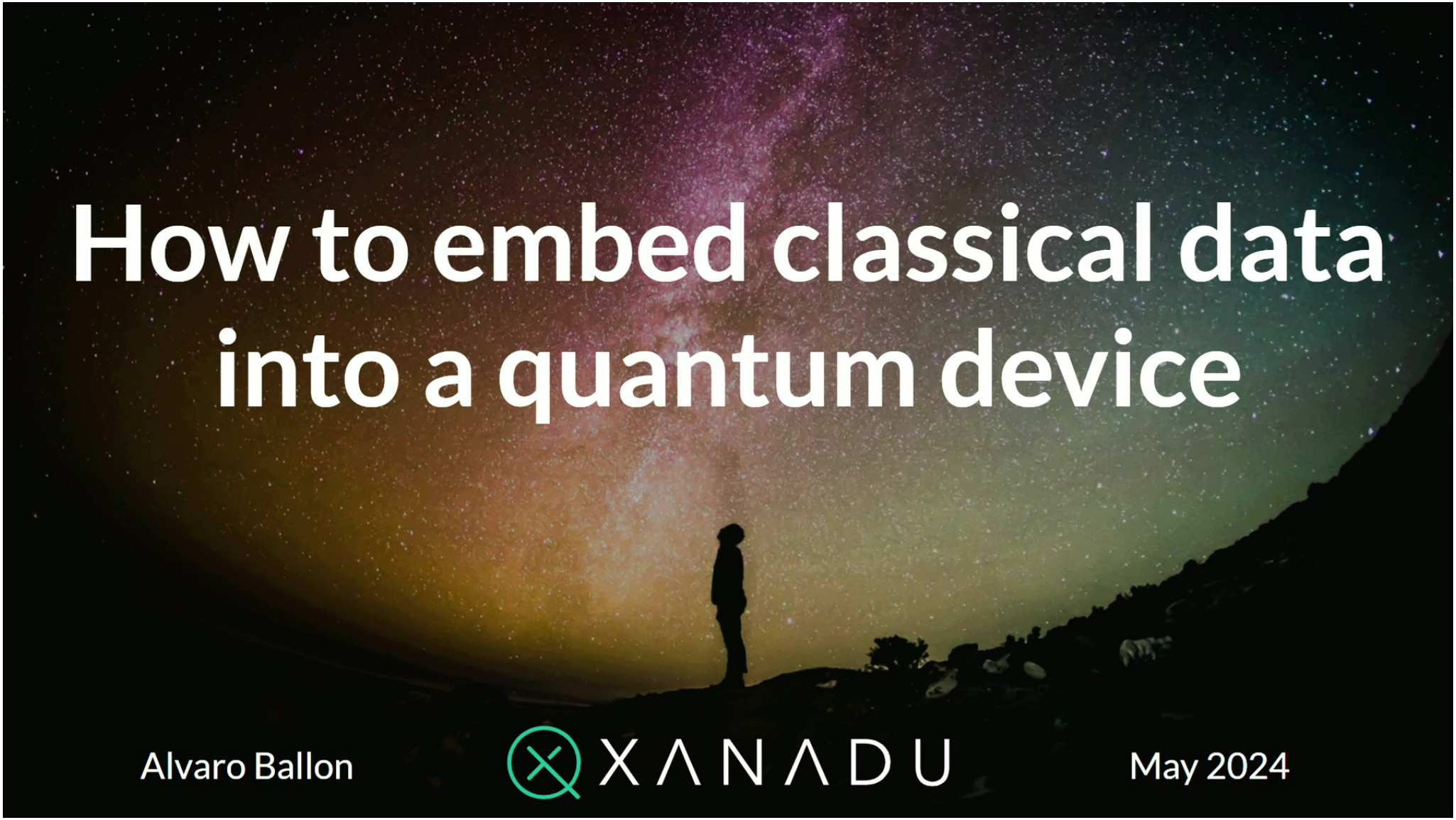
Title: Quantum Machine Learning

Speakers: Alvaro Ballon Bordo

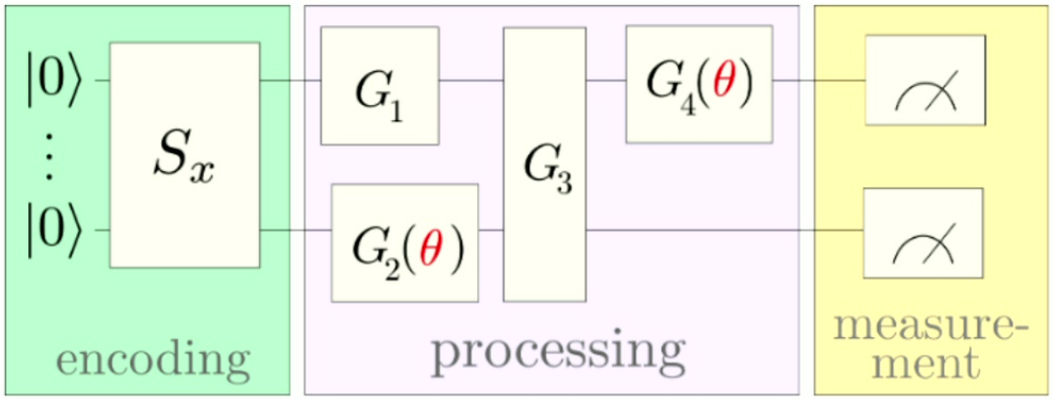
Collection: Navigating Quantum and AI Career Trajectories: A Beginnerâ€™s Mini-Course on Computational Methods and their Applications

Date: May 23, 2024 - 9:30 AM

URL: https://pirsa.org/24050085


How to embed classical data into a quantum device
Alvaro Ballon
XANADU
May 2024

Data embedding



Basis Embedding

Used for discrete data, encoded as binary strings

```
N = 3
wires = range(N)
dev = qml.device("default.qubit", wires)

@qml.qnode(dev)
def circuit(b):
    qml.BasisEmbedding(b, wires)
    return qml.state()
>>> circuit([1, 1, 1])
[0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 0.+0.j 1.+0.j]
```

Angle Embedding

Used for real-valued data. Normalize, then embed as rotation angles.

$$x \mapsto R_k(x)|0\rangle = e^{-ix\sigma_k/2}|0\rangle,$$

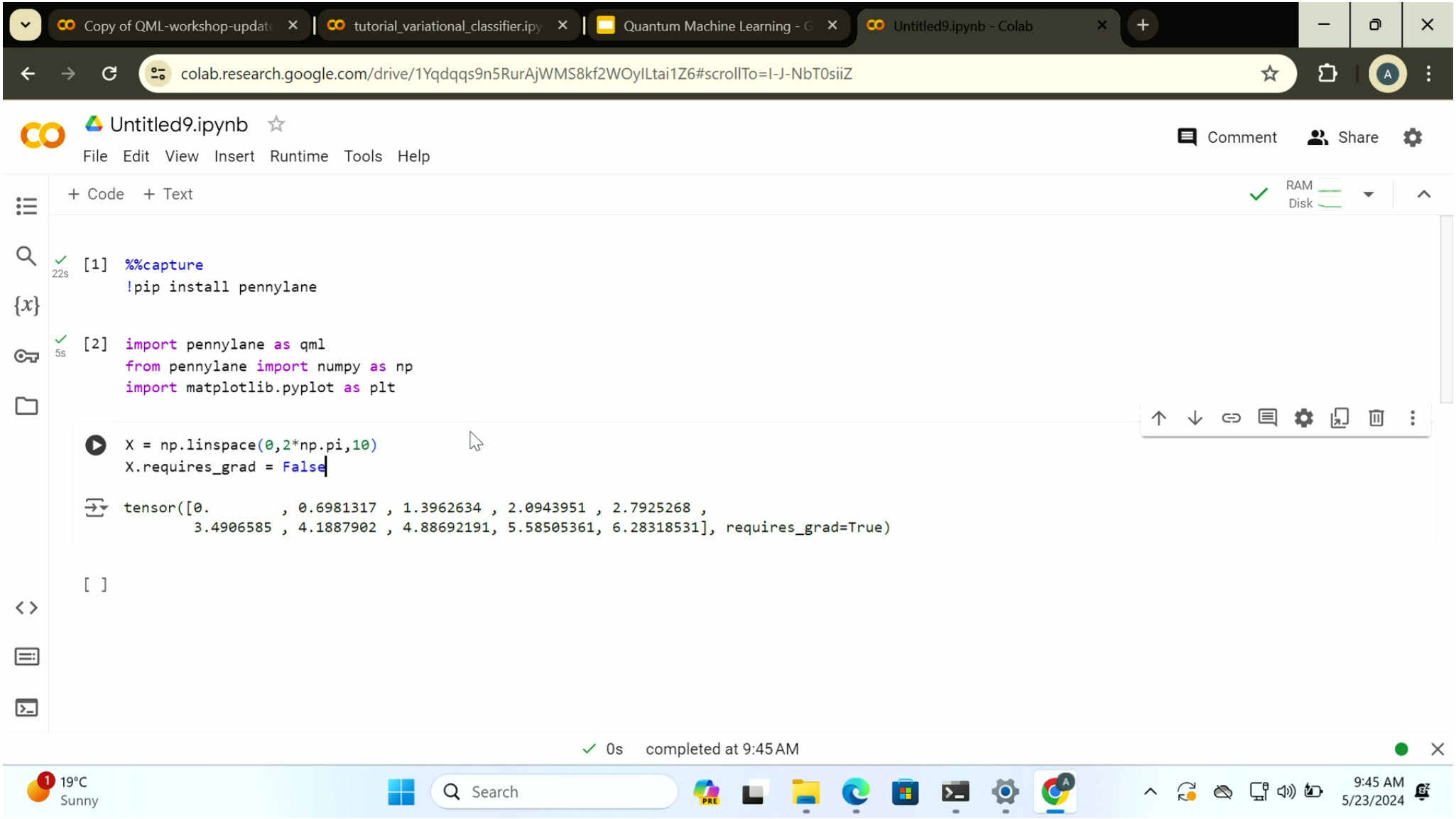
```
@qml.qnode(dev)
def circuit(val_list):
    qml.AngleEmbedding(val_list, wires)
    return [qml.expval(qml.PauliZ(w)) for w in wires]
>>> circuit([0.0, np.pi / 2, np.pi])
tensor([ 1.0,  0.0, -1.0], requires_grad=True)
```

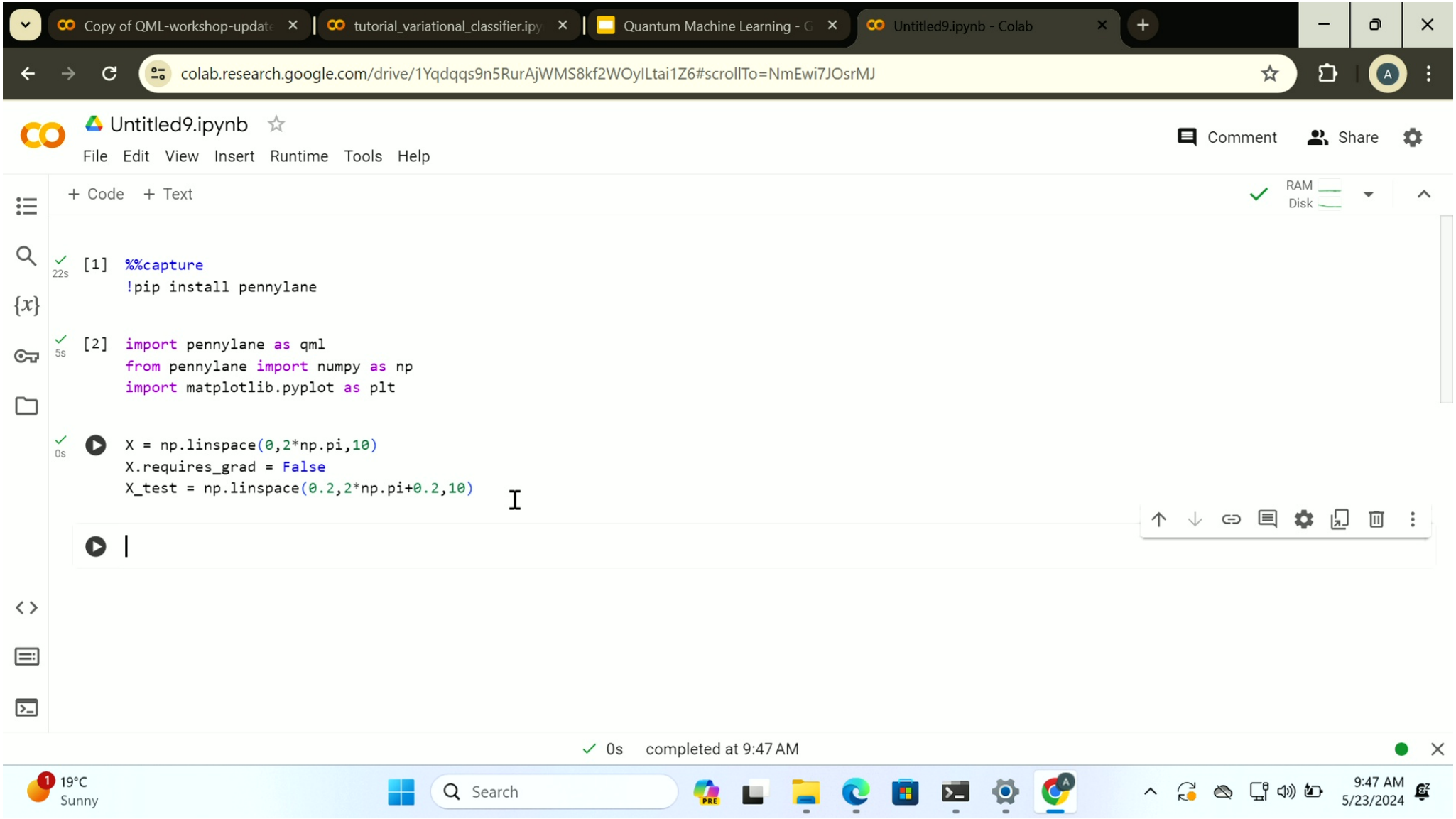
Amplitude Embedding

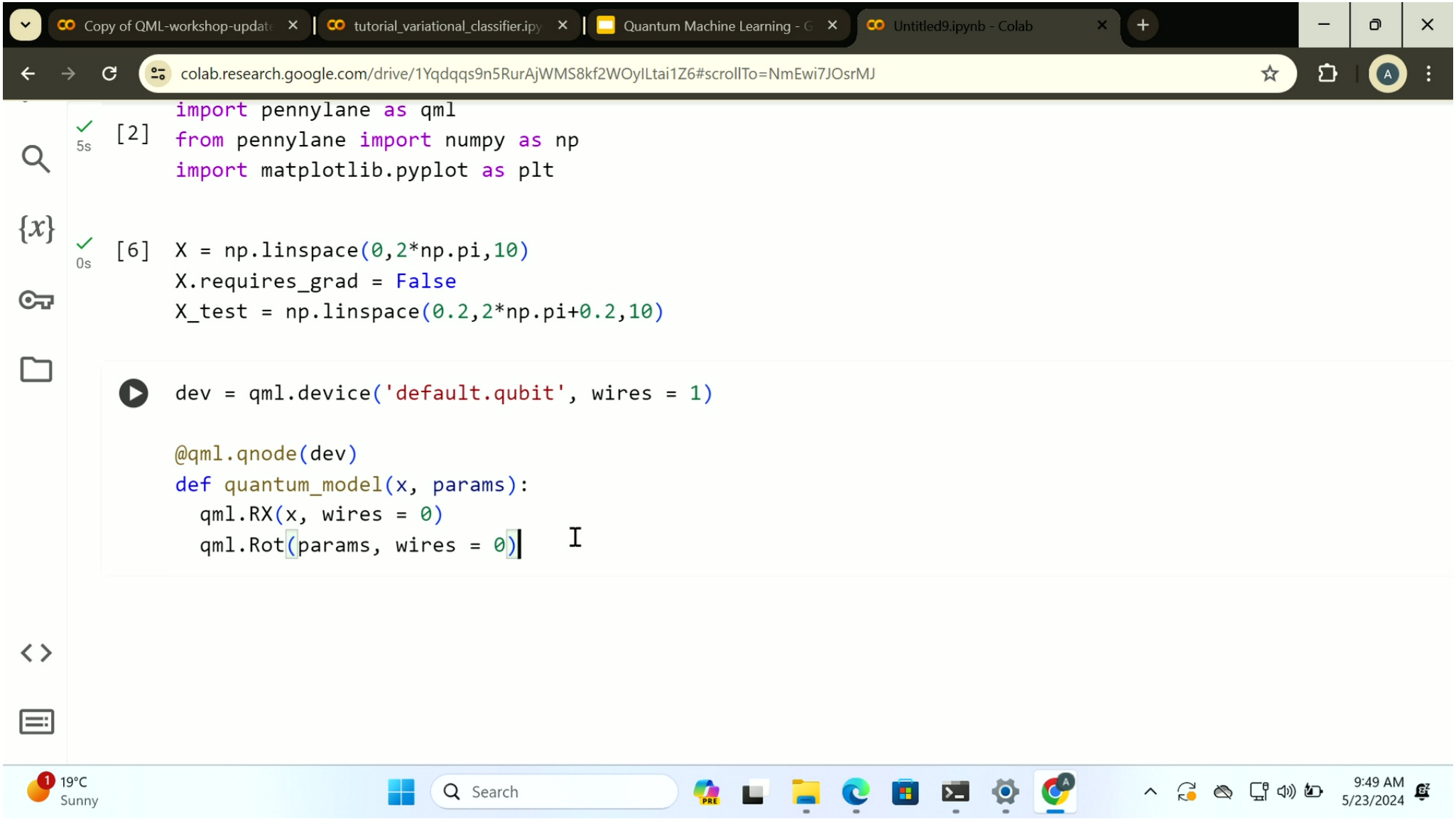
Used for complex- valued data.

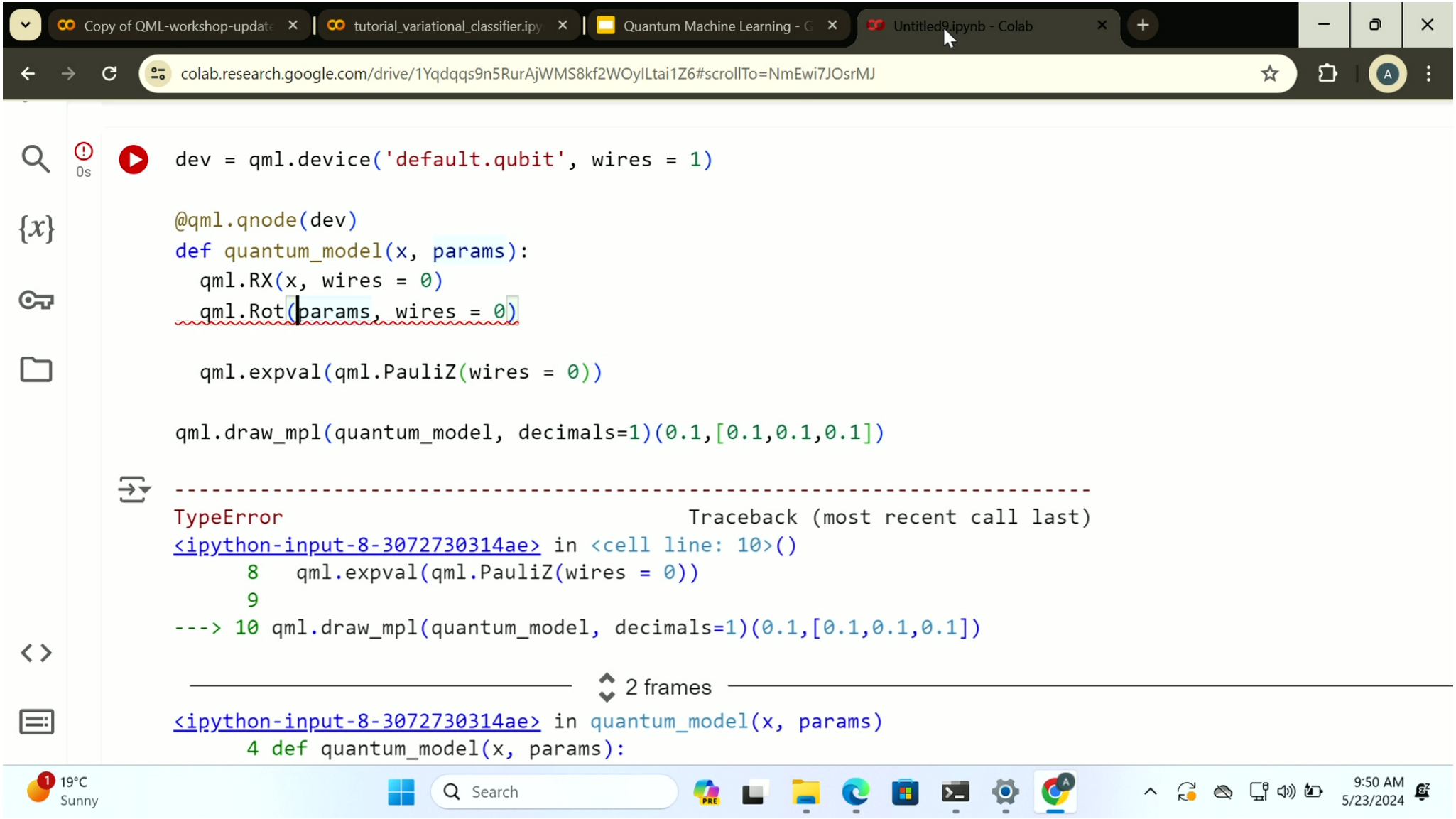
$$\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_{2^N-1}) \mapsto \sum_{k=0}^{2^N-1} \alpha_k |k\rangle$$

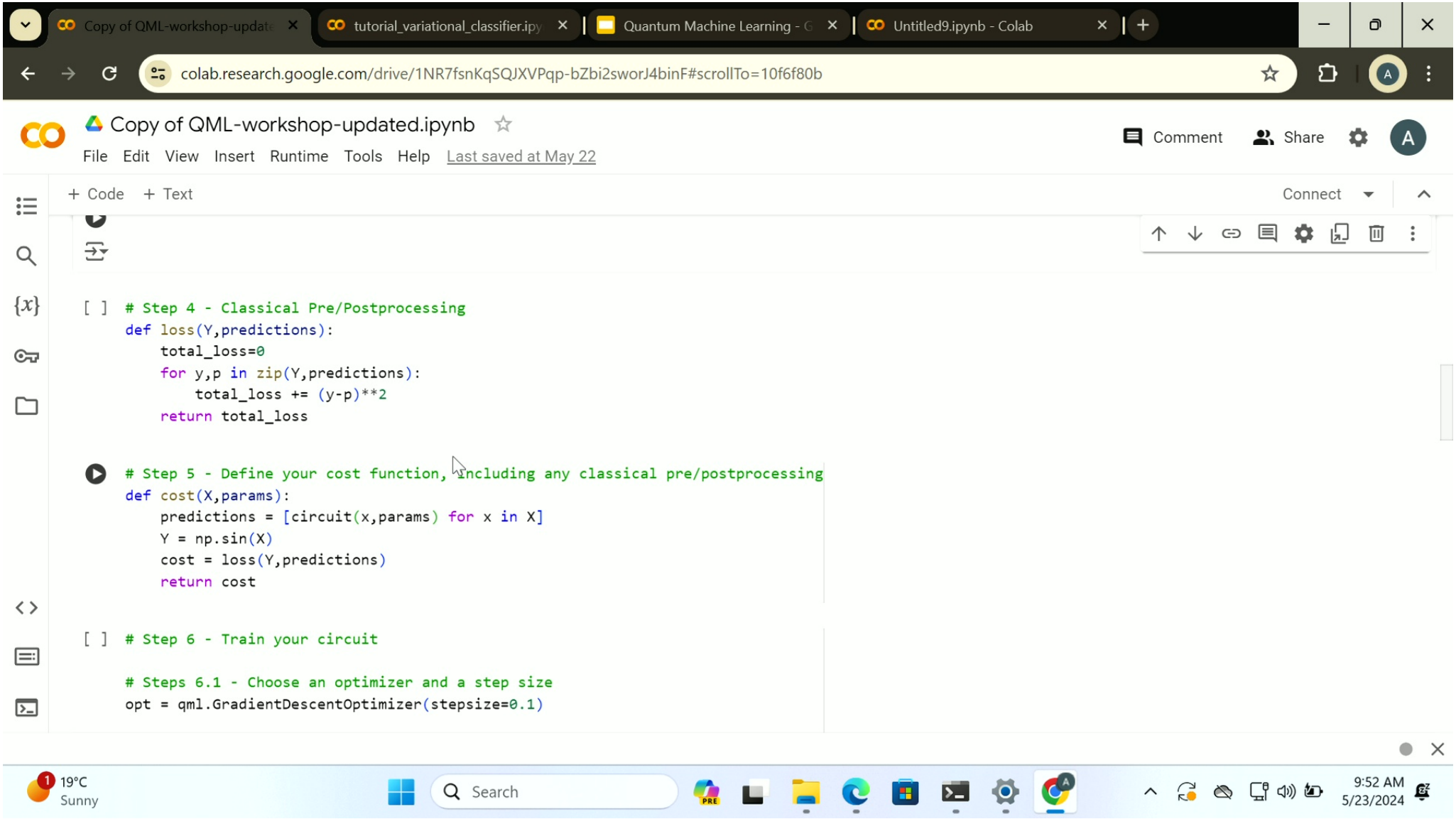
```
N = 3
wires = range(N)
dev = qml.device("default.qubit", wires)

@qml.qnode(dev)
def circuit(features):
    qml.AmplitudeEmbedding(features, wires)
    return qml.state()
>>> circuit([0.625, 0.0, 0.0, 0.0, 0.625j, 0.375, 0.25, 0.125])
tensor([0.625+0.j   , 0.   +0.j   , 0.   +0.j   , 0.   +0.j   ,
        0.   +0.625j, 0.375+0.j   , 0.25 +0.j   , 0.125+0.j   ], requires_grad=True)
```


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=I-J-NbT0siiZ
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
22s
[1] %%capture
!pip install pennylane
5s
[2] import pennylane as qml
from pennylane import numpy as np
import matplotlib.pyplot as plt
X = np.linspace(0,2*np.pi,10)
X.requires_grad = False
tensor([0. , 0.6981317 , 1.3962634 , 2.0943951 , 2.7925268 ,
3.4906585 , 4.1887902 , 4.88692191, 5.58505361, 6.28318531], requires_grad=True)
[ ]
0s completed at 9:45 AM
19°C
Sunny
Search
9:45 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=NmEwi7JOsrMJ
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
22s [1] %%capture
!pip install pennylane
5s [2] import pennylane as qml
from pennylane import numpy as np
import matplotlib.pyplot as plt
0s X = np.linspace(0,2*np.pi,10)
X.requires_grad = False
X_test = np.linspace(0.2,2*np.pi+0.2,10)
0s completed at 9:47 AM
19°C
Sunny
Search
9:47 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=NmEwi7JOsrMJ
import pennylane as qml
[2]
5s
from pennylane import numpy as np
import matplotlib.pyplot as plt
[6]
0s
X = np.linspace(0,2*np.pi,10)
X.requires_grad = False
X_test = np.linspace(0.2,2*np.pi+0.2,10)
dev = qml.device('default.qubit', wires = 1)
@qml.qnode(dev)
def quantum_model(x, params):
qml.RX(x, wires = 0)
qml.Rot(params, wires = 0)
19°C
Sunny
Search
9:49 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled0.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=NmEwi7JOsrMJ
0s
dev = qml.device('default.qubit', wires = 1)
@qml.qnode(dev)
def quantum_model(x, params):
qml.RX(x, wires = 0)
qml.Rot(params, wires = 0)
qml.expval(qml.PauliZ(wires = 0))
qml.draw_mpl(quantum_model, decimals=1)(0.1,[0.1,0.1,0.1])
TypeError Traceback (most recent call last)
<ipython-input-8-3072730314ae> in <cell line: 10>()
8 qml.expval(qml.PauliZ(wires = 0))
9
---> 10 qml.draw_mpl(quantum_model, decimals=1)(0.1,[0.1,0.1,0.1])
2 frames
<ipython-input-8-3072730314ae> in quantum_model(x, params)
4 def quantum_model(x, params):
19°C
Sunny
Search
9:50 AM
5/23/2024

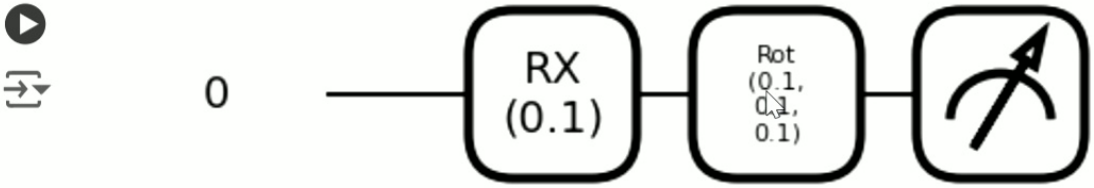

Copy of QML-workshop-updat
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1NR7fsnKqSQJXVPqp-bZbi2sworJ4binF#scrollTo=10f6f80b
Copy of QML-workshop-updated.ipynb
File Edit View Insert Runtime Tools Help Last saved at May 22
Comment
Share
+ Code + Text
Connect
[ ] # Step 4 - Classical Pre/Postprocessing
def loss(Y,predictions):
total_loss=0
for y,p in zip(Y,predictions):
total_loss += (y-p)**2
return total_loss
# Step 5 - Define your cost function, including any classical pre/postprocessing
def cost(X,params):
predictions = [circuit(x,params) for x in X]
Y = np.sin(X)
cost = loss(Y,predictions)
return cost
[ ] # Step 6 - Train your circuit
# Steps 6.1 - Choose an optimizer and a step size
opt = qml.GradientDescentOptimizer(stepsize=0.1)
19°C
Sunny
Search
9:52 AM
5/23/2024

Copy of QML-workshop-update | tutorial_variational_classifier.ipy | Quantum Machine Learning - G | Untitled9.ipynb - Colab

colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=nZGs-EpYt1-G

+ Code + Text

0s

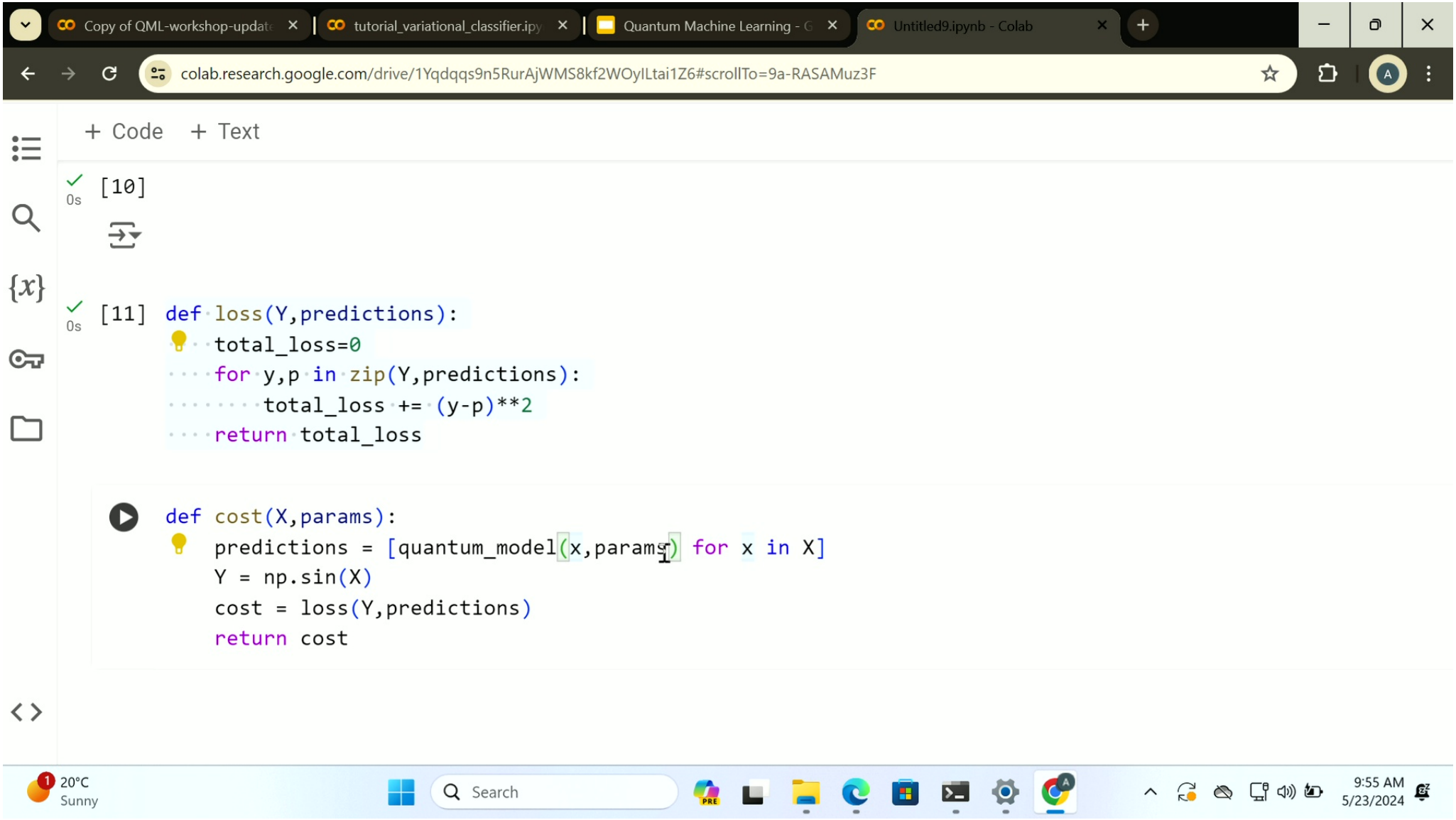


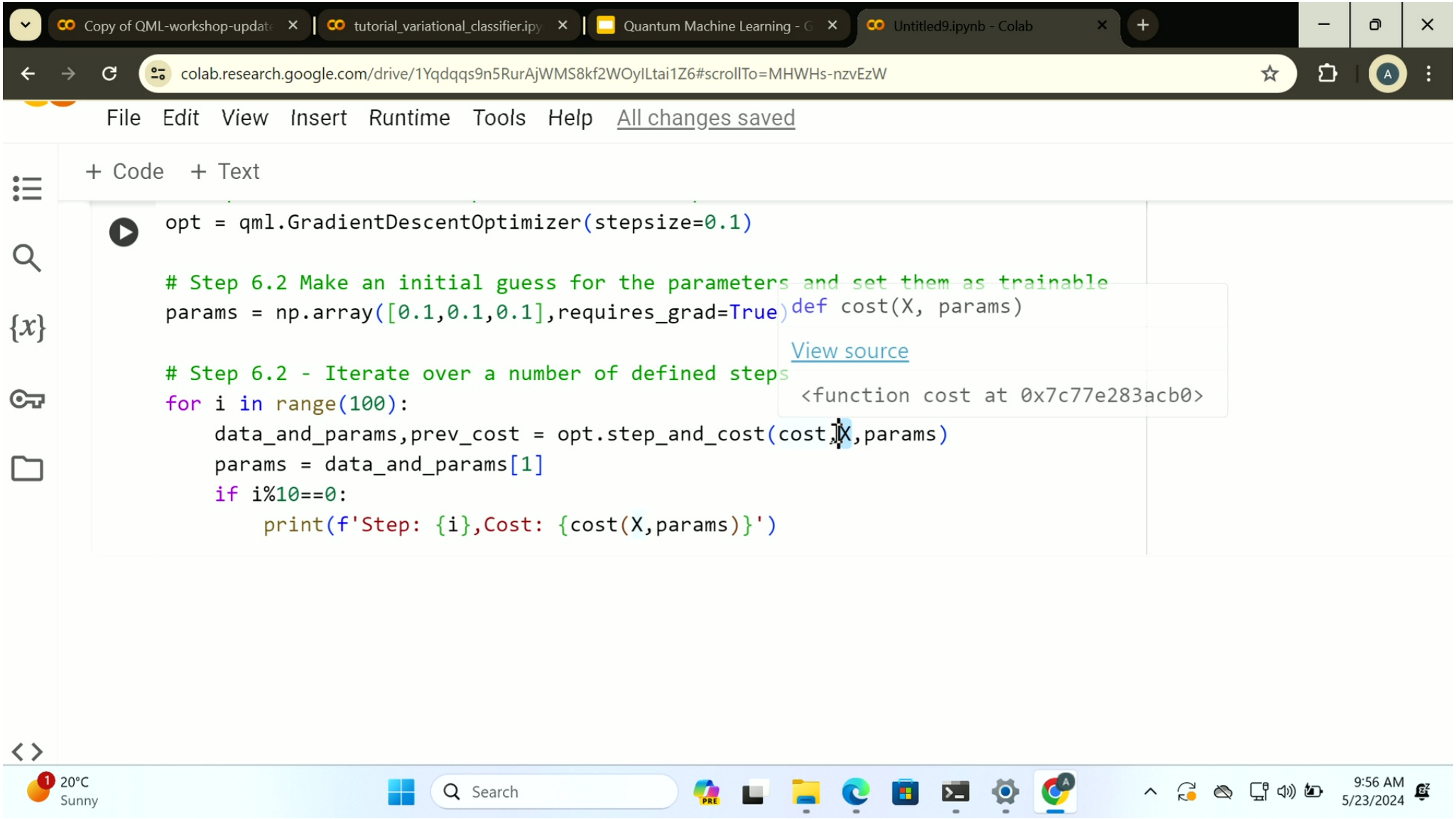

```
def loss(Y,predictions):
    total_loss=0
    for y,p in zip(Y,predictions):
        total_loss += (y-p)**2
    return total_loss
```

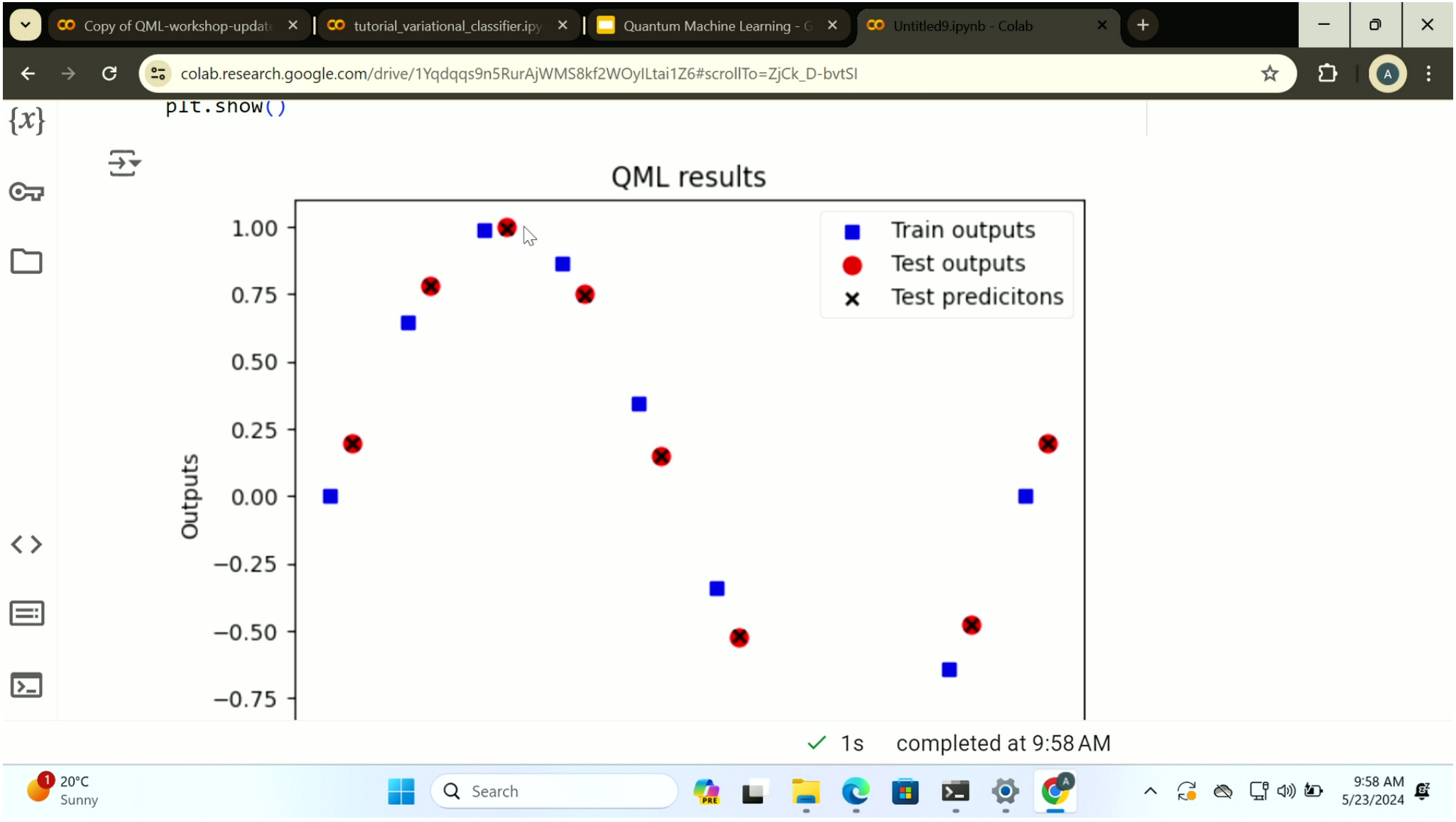
DAL - MIN
Video highlight

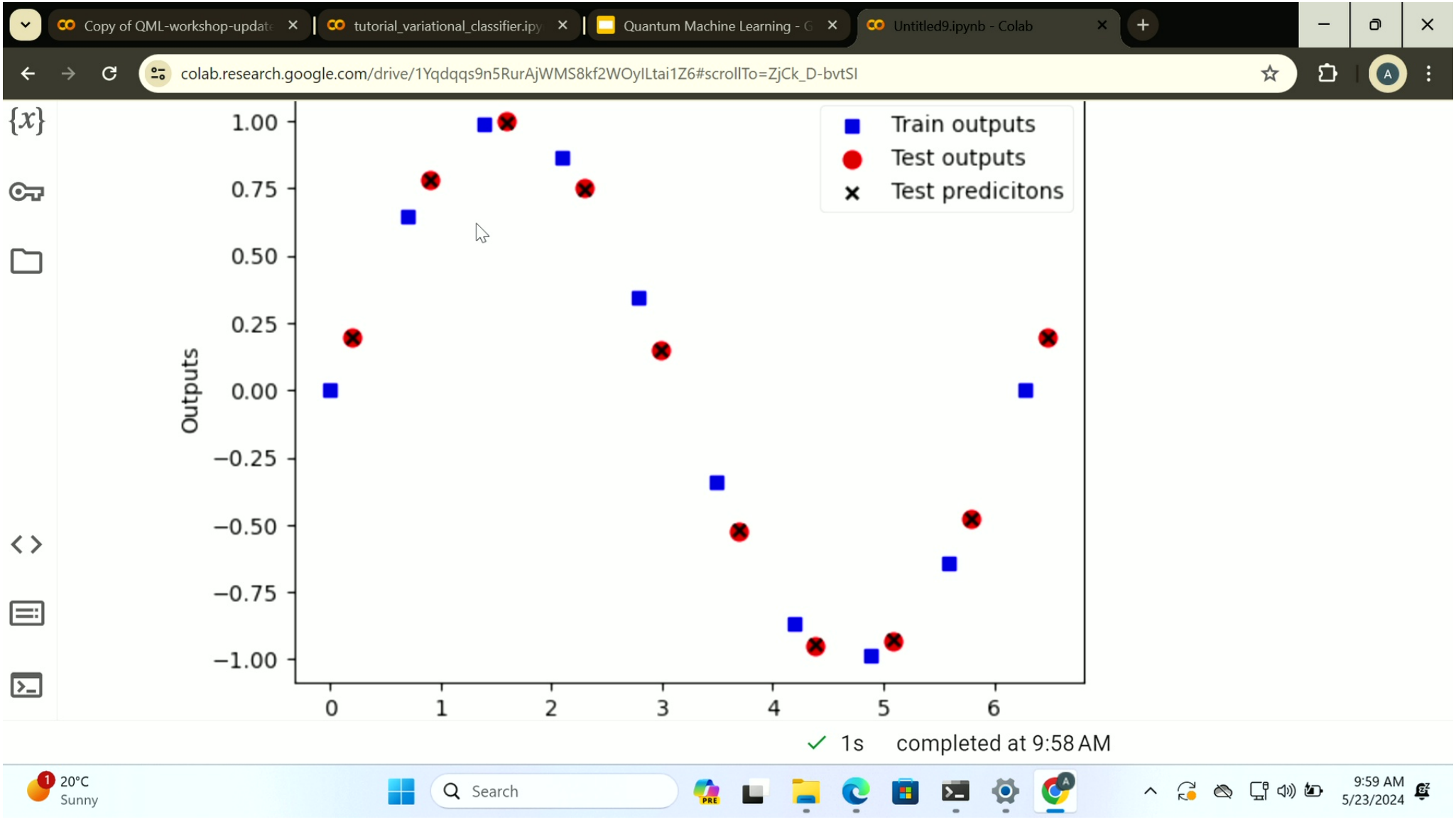
Search

9:53 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=9a-RASAMuz3F
+ Code
+ Text
[10]
0s
[11]
0s
def loss(Y,predictions):
total_loss=0
for y,p in zip(Y,predictions):
total_loss += (y-p)**2
return total_loss
def cost(X,params):
predictions = [quantum_model(x,params) for x in X]
Y = np.sin(X)
cost = loss(Y,predictions)
return cost
20°C
Sunny
Search
9:55 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=MHWHs-nzvEzW
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
opt = qml.GradientDescentOptimizer(stepsize=0.1)
# Step 6.2 Make an initial guess for the parameters and set them as trainable
params = np.array([0.1,0.1,0.1],requires_grad=True)
# Step 6.2 - Iterate over a number of defined steps
for i in range(100):
data_and_params,prev_cost = opt.step_and_cost(cost,X,params)
params = data_and_params[1]
if i%10==0:
print(f'Step: {i},Cost: {cost(X,params)}')
def cost(X, params)
View source
<function cost at 0x7c77e283acb0>
20°C
Sunny
Search
9:56 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=ZjCk_D-bvtSI
plt.show()
QML results
Train outputs
Test outputs
Test predicitons
Outputs
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
1s completed at 9:58 AM
20°C
Sunny
Search
9:58 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=ZjCk_D-bvtSI
Train outputs
Test outputs
Test predicitons
Outputs
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4
5
6
1s completed at 9:58 AM
20°C
Sunny
Search
9:59 AM
5/23/2024

Copy of QML-workshop-update | tutorial_variational_classifier.ipy | Quantum Machine Learning - G | Untitled9.ipynb - Colab

docs.google.com/presentation/d/18nRyOMHS8DR-xNEuw4vYfheGCr5yPD2D2_-k82oEfLg/edit#slide=id.g29e30b7e761_0_53

Quantum Machine Learning

File Edit View Insert Format Slide Arrange Tools Extensions Help

Slideshow | Share

Fit | Background | Layout | Theme | Transition | Rec

4 Angle embedding

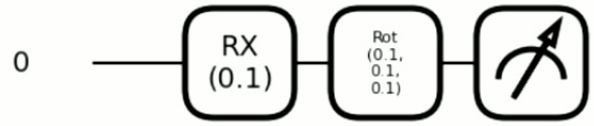
5 Amplitude Embedding

6 A simple QML model

7 Variational Quantum Classifier

A simple QML model

Let's do something super simple to warm up



RX here is the angle embedding. Rot is the model

XANADU

Click to add speaker notes

20°C Sunny | Search | 9:59 AM 5/23/2024

Variational Quantum Classifier (VQC)

Aim: Classify **discrete** data.

As per the Machine Learning supervised learning paradigm, we train with labelled data, and then extend to unknown data

In this example, we classify bit strings according to their parity.

$$f : x \in \{0,1\}^{\otimes n} \rightarrow y = \begin{cases} 1 \text{ if uneven number of 1's in } x \\ 0 \text{ else.} \end{cases}$$

Copy of QML-workshop-update | tutorial_variational_classifier.ipy | Quantum Machine Learning - G | Untitled9.ipynb - Colab

colab.research.google.com/drive/19noaiTFs0WoKOGZH9g5nizCyPtwdW55F

tutorial_variational_classifier.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:19 AM

Comment Share

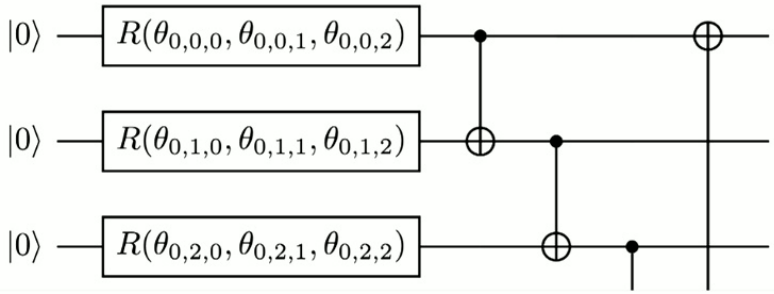
+ Code + Text Connect

We then create a quantum device that will run our circuits.

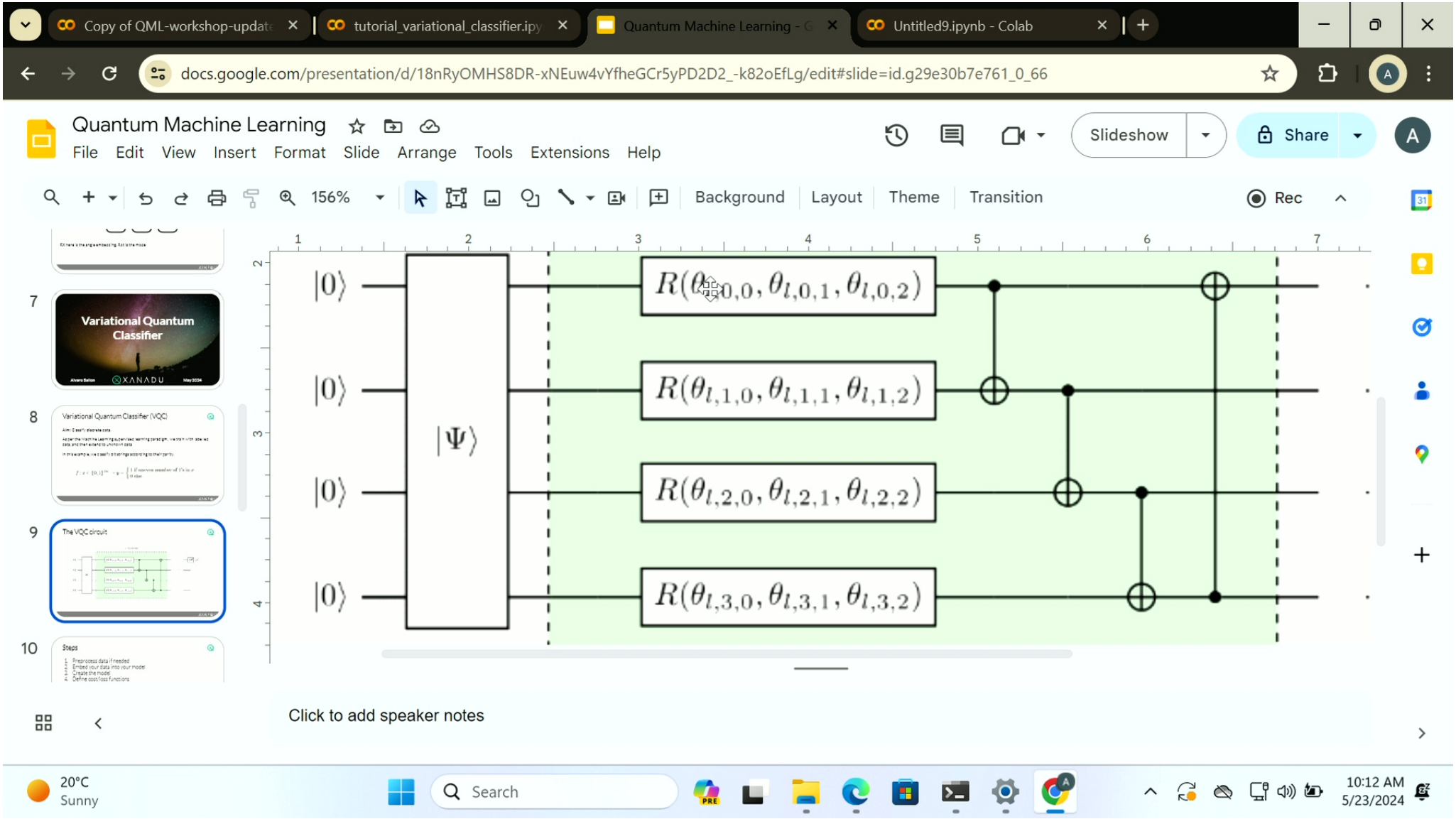
```
[ ]  dev = qml.device("default.qubit")
```

Variational classifiers usually define a "layer" or "block", which is an elementary circuit architecture that gets repeated to build the full variational circuit.

Our circuit layer will use four qubits, or wires, and consists of an arbitrary rotation on every qubit, as well as a ring of CNOTs that entangles each qubit with its neighbour. Borrowing from machine learning, we call the parameters of the layer `weights`. Let's take a look on how this quantum circuit looks like:



20°C Sunny | Search | 10:10 AM 5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
docs.google.com/presentation/d/18nRyOMHS8DR-xNEuw4vYfheGCr5yPD2D2_-k82oEfLg/edit#slide=id.g29e30b7e761_0_66
Quantum Machine Learning
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Share
156%
Background
Layout
Theme
Transition
Rec
7
Variational Quantum Classifier
8
Variational Quantum Classifier (VQC)
9
The VQC circuit
10
Steps
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|\Psi\rangle$
$R(\theta_{l,0,0}, \theta_{l,0,1}, \theta_{l,0,2})$
$R(\theta_{l,1,0}, \theta_{l,1,1}, \theta_{l,1,2})$
$R(\theta_{l,2,0}, \theta_{l,2,1}, \theta_{l,2,2})$
$R(\theta_{l,3,0}, \theta_{l,3,1}, \theta_{l,3,2})$
Click to add speaker notes
20°C
Sunny
Search
10:12 AM
5/23/2024

Copy of QML-workshop-update | tutorial_variational_classifier.ipy | Quantum Machine Learning - G | Untitled9.ipynb - Colab

colab.research.google.com/drive/19noaiTFs0WoKOGZH9g5nizCyPtwdW55F#scrollTo=UDCZeExGUN6-

tutorial_variational_classifier.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:19 AM

Comment Share

+ Code + Text Connect

```
[ ]
```

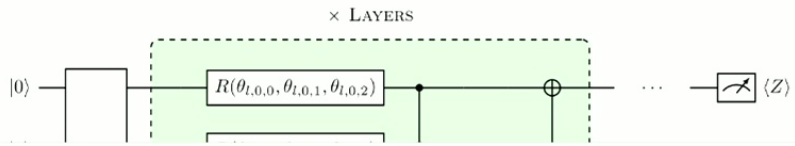
We also need a way to encode data inputs x into the circuit, so that the measured output depends on the inputs. In this first example, the inputs are bitstrings, which we encode into the state of the qubits. The quantum state ψ after state preparation is a computational basis state that has 1s where x has 1s, for example

$$x = 0101 \rightarrow |\psi\rangle = |0101\rangle.$$

The `pennylane.BasisState` function provided by PennyLane is made to do just this. It expects `x` to be a list of zeros and ones, i.e. `[0,1,0,1]`.

```
def state_preparation(x):
    qml.BasisState(x, wires=[0, 1, 2, 3])
```

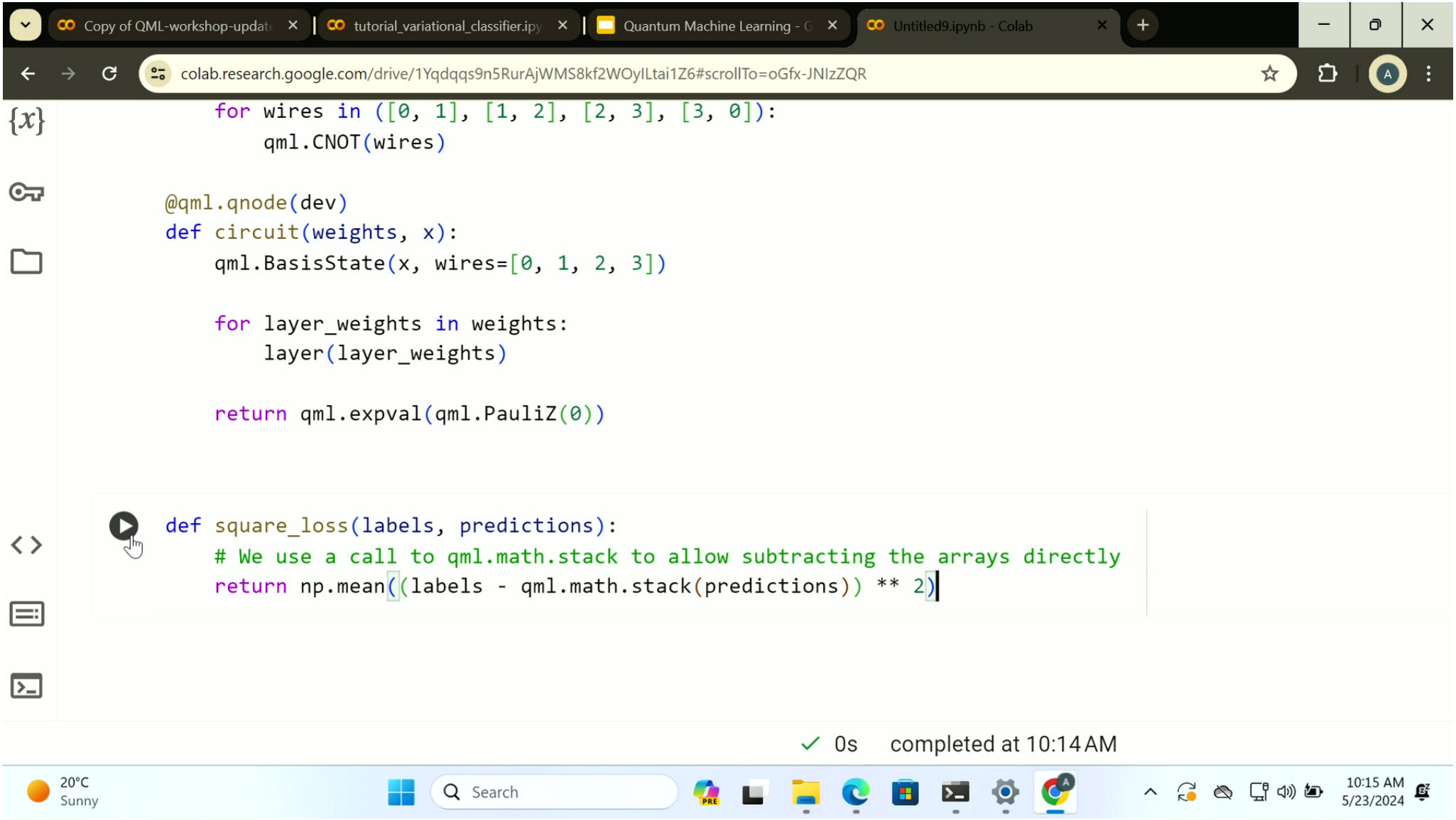
Now we define the variational quantum circuit as this state preparation routine, followed by a repetition of the layer structure. This circuit should look like this:

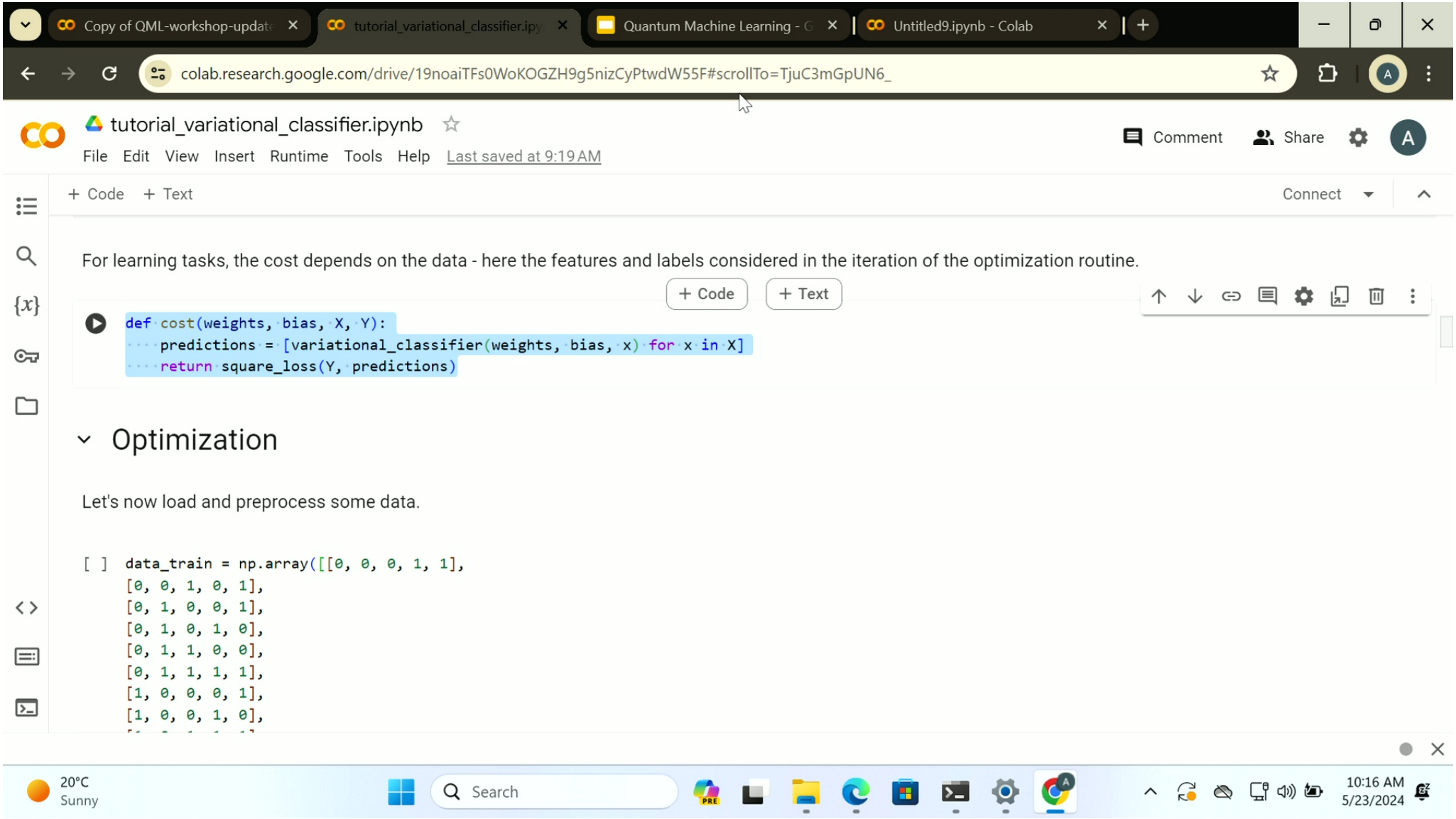


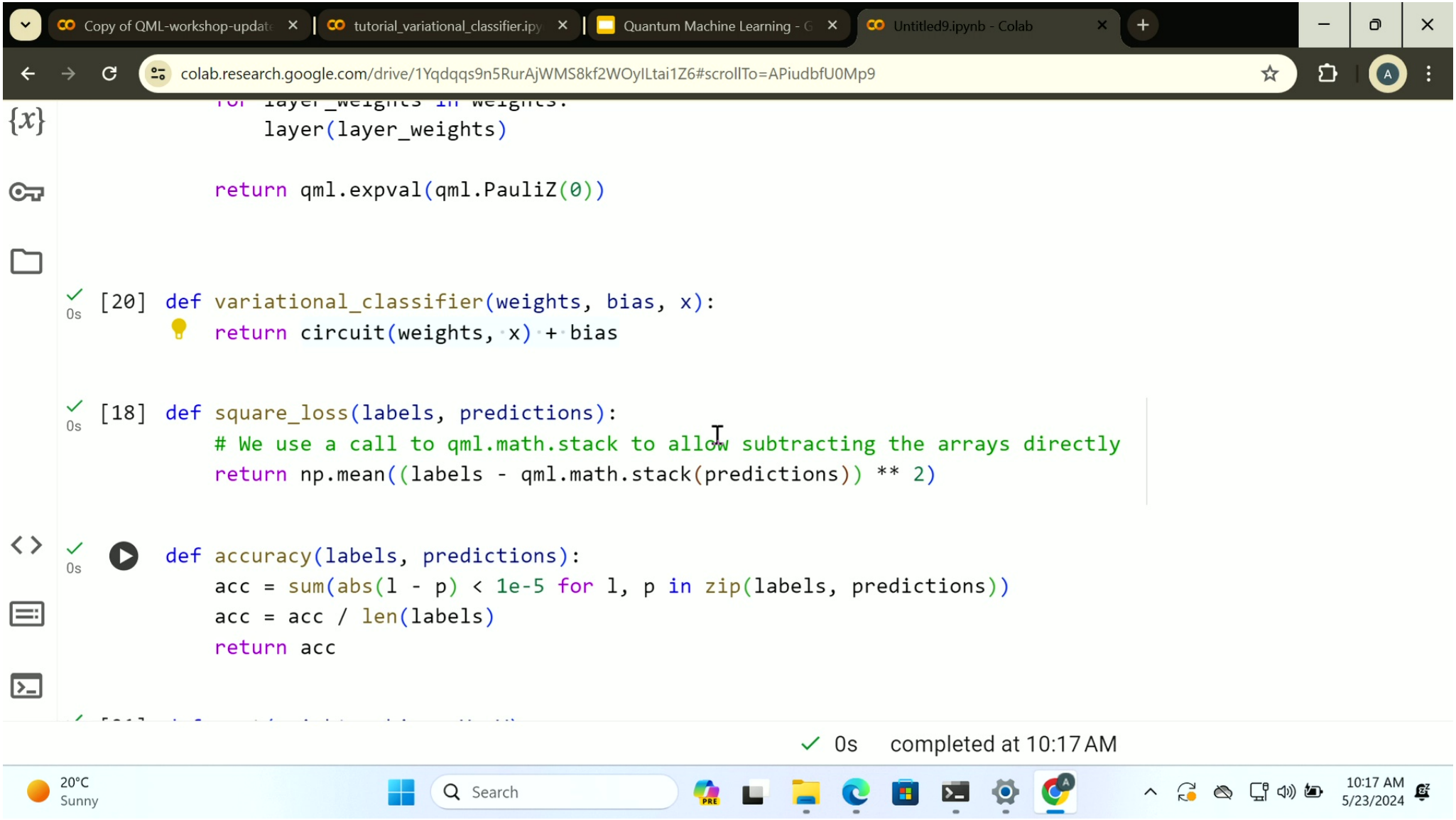
20°C Sunny

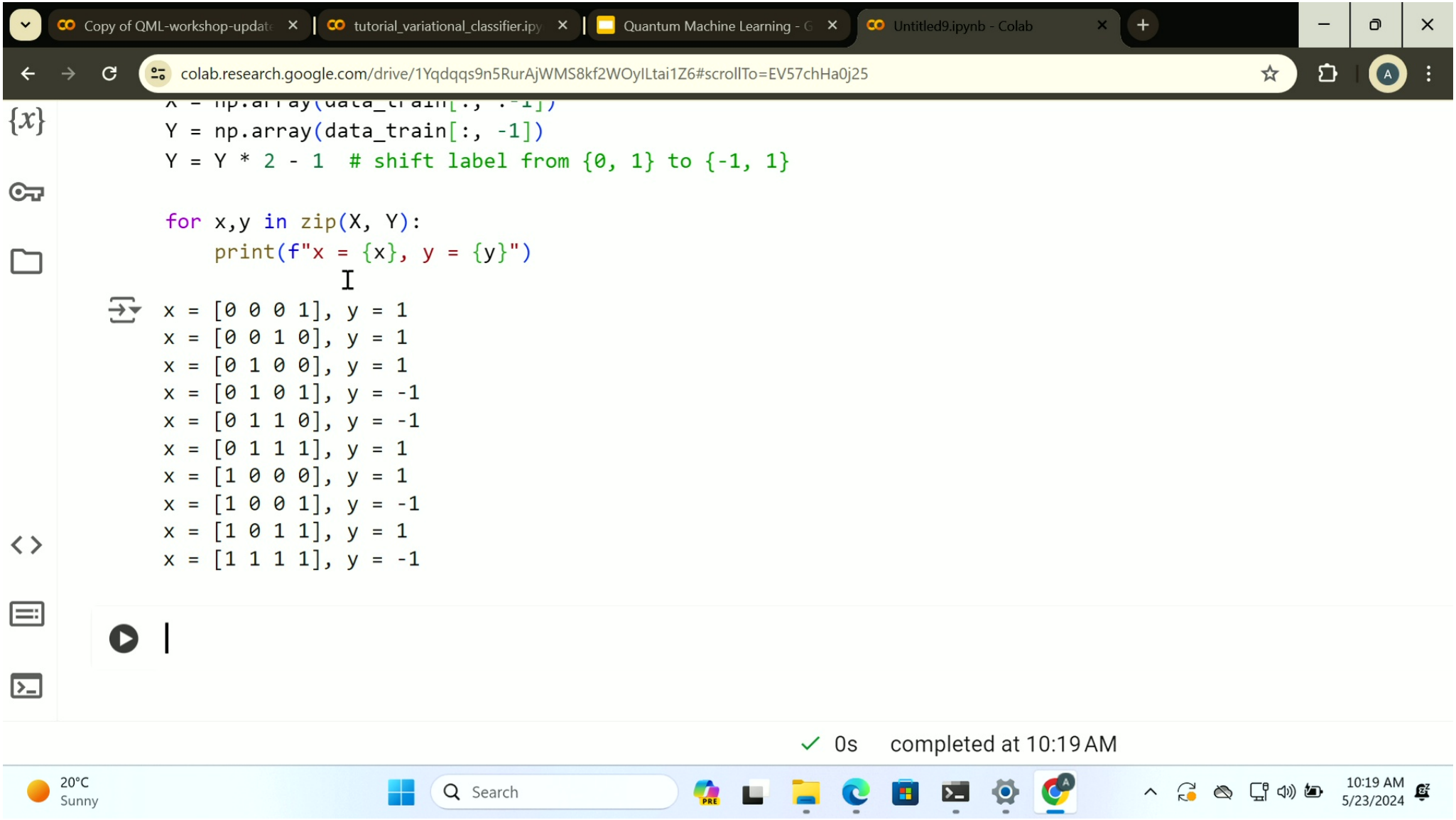
Search

10:13 AM 5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=oGfx-JNIzZQR
for wires in ([0, 1], [1, 2], [2, 3], [3, 0]):
qml.CNOT(wires)
@qml.qnode(dev)
def circuit(weights, x):
qml.BasisState(x, wires=[0, 1, 2, 3])
for layer_weights in weights:
layer(layer_weights)
return qml.expval(qml.PauliZ(0))
def square_loss(labels, predictions):
# We use a call to qml.math.stack to allow subtracting the arrays directly
return np.mean((labels - qml.math.stack(predictions)) ** 2)
0s completed at 10:14 AM
20°C
Sunny
Search
10:15 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/19noaiTFs0WoKOGZH9g5nizCyPtwdW55F#scrollTo=TjuC3mGpUN6_
tutorial_variational_classifier.ipynb
File Edit View Insert Runtime Tools Help Last saved at 9:19 AM
Comment
Share
+ Code + Text
Connect
For learning tasks, the cost depends on the data - here the features and labels considered in the iteration of the optimization routine.
+ Code + Text
def cost(weights, bias, X, Y):
predictions = [variational_classifier(weights, bias, x) for x in X]
return square_loss(Y, predictions)
Optimization
Let's now load and preprocess some data.
[ ] data_train = np.array([[0, 0, 0, 1, 1],
[0, 0, 1, 0, 1],
[0, 1, 0, 0, 1],
[0, 1, 0, 1, 0],
[0, 1, 1, 0, 0],
[0, 1, 1, 1, 1],
[1, 0, 0, 0, 1],
[1, 0, 0, 1, 0],
20°C
Sunny
Search
10:16 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=APiudbfU0Mp9
layer(layer_weights)
return qml.expval(qml.PauliZ(0))
[20] def variational_classifier(weights, bias, x):
return circuit(weights, x) + bias
[18] def square_loss(labels, predictions):
We use a call to qml.math.stack to allow subtracting the arrays directly
return np.mean((labels - qml.math.stack(predictions)) ** 2)
def accuracy(labels, predictions):
acc = sum(abs(l - p) < 1e-5 for l, p in zip(labels, predictions))
acc = acc / len(labels)
return acc
0s completed at 10:17 AM
20°C
Sunny
Search
10:17 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=EV57chHa0j25
Y = np.array(data_train[:, -1])
Y = Y * 2 - 1 # shift label from {0, 1} to {-1, 1}
for x,y in zip(X, Y):
print(f"x = {x}, y = {y}")
x = [0 0 0 1], y = 1
x = [0 0 1 0], y = 1
x = [0 1 0 0], y = 1
x = [0 1 0 1], y = -1
x = [0 1 1 0], y = -1
x = [0 1 1 1], y = 1
x = [1 0 0 0], y = 1
x = [1 0 0 1], y = -1
x = [1 0 1 1], y = 1
x = [1 1 1 1], y = -1
0s completed at 10:19 AM
20°C
Sunny
Search
10:19 AM
5/23/2024

Copy of QML-workshop-update | tutorial_variational_classifier.ipy | Quantum Machine Learning - G | Untitled9.ipynb - Colab

colab.research.google.com/drive/19noaiTFs0WoKOGZH9g5nizCyPtwdW55F#scrollTo=_Z2suPXdUN69

tutorial_variational_classifier.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:19 AM

Comment Share A

+ Code + Text Connect

```
import pennylane as qml
from pennylane import numpy as np
from pennylane.optimize import NesterovMomentumOptimizer
```

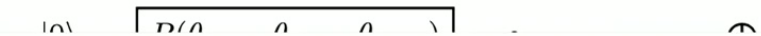
Quantum and classical nodes

We then create a quantum device that will run our circuits.

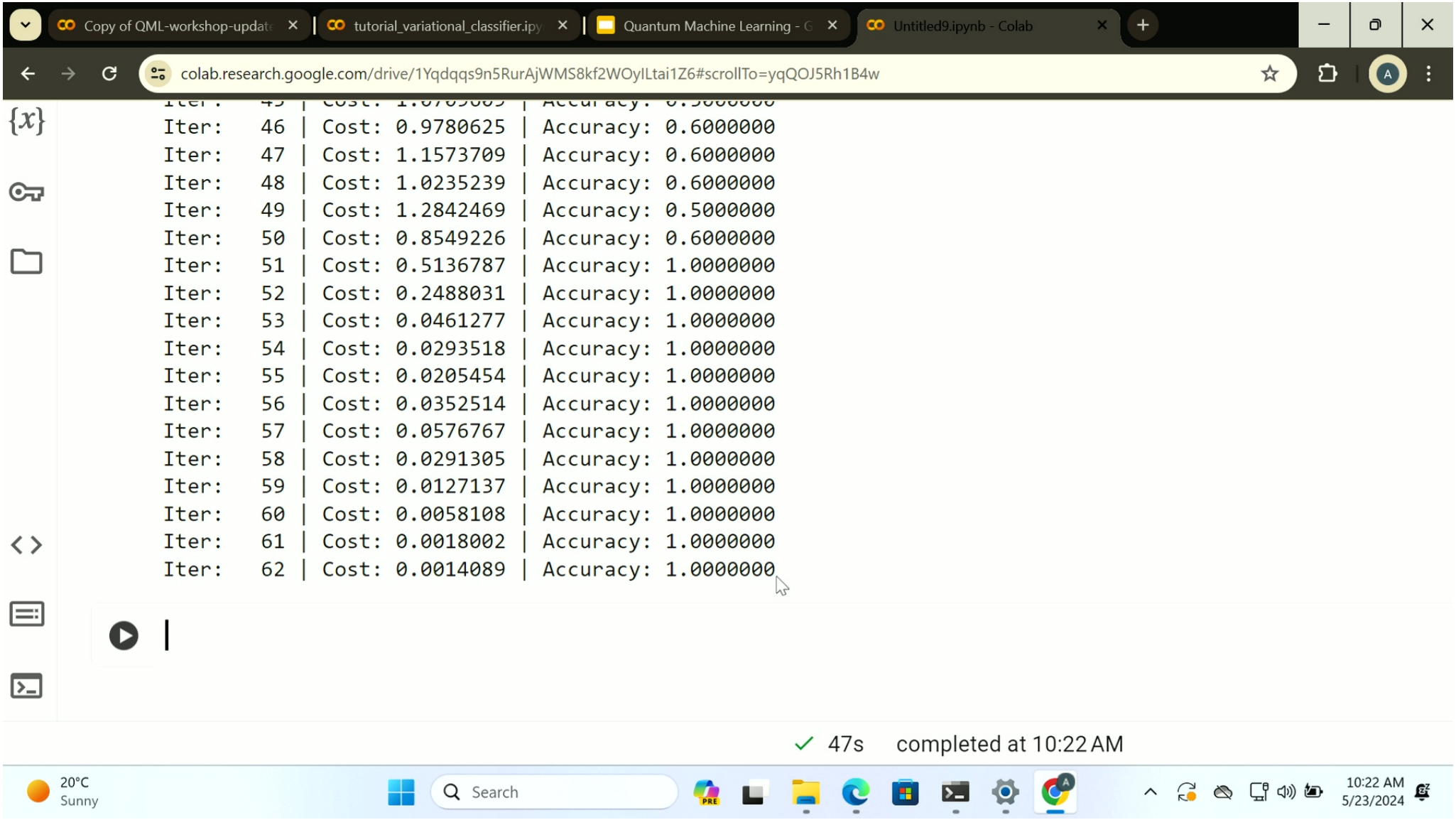
```
[ ]  dev = qml.device("default.qubit")
```

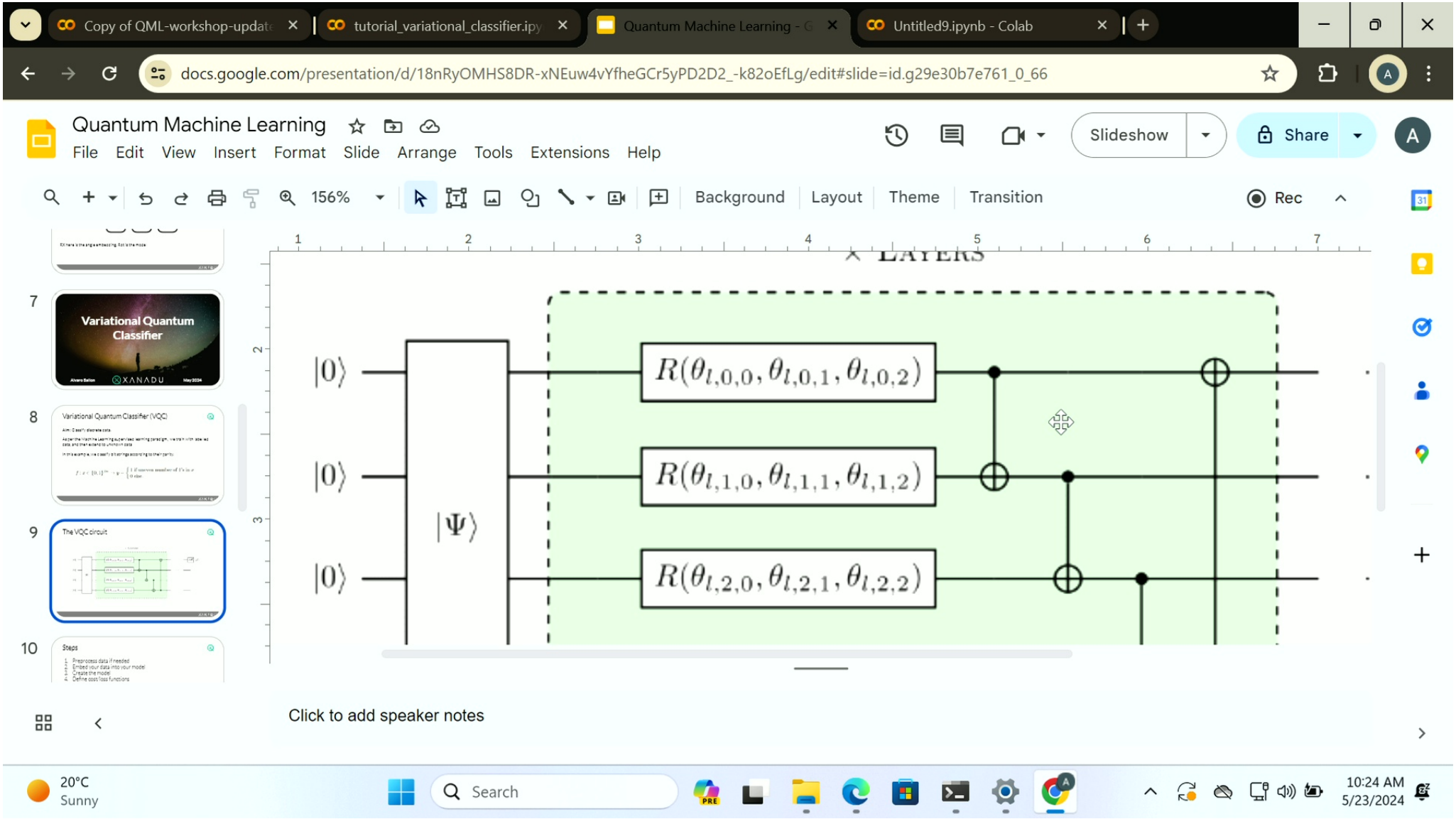
Variational classifiers usually define a "layer" or "block", which is an elementary circuit architecture that gets repeated to build the full variational circuit.

Our circuit layer will use four qubits, or wires, and consists of an arbitrary rotation on every qubit, as well as a ring of CNOTs that entangles each qubit with its neighbour. Borrowing from machine learning, we call the parameters of the layer `weights`. Let's take a look on how this quantum circuit looks like:



20°C Sunny | Search | 10:21 AM 5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=yqQOJ5Rh1B4w
Iter: 46 | Cost: 0.9780625 | Accuracy: 0.6000000
Iter: 47 | Cost: 1.1573709 | Accuracy: 0.6000000
Iter: 48 | Cost: 1.0235239 | Accuracy: 0.6000000
Iter: 49 | Cost: 1.2842469 | Accuracy: 0.5000000
Iter: 50 | Cost: 0.8549226 | Accuracy: 0.6000000
Iter: 51 | Cost: 0.5136787 | Accuracy: 1.0000000
Iter: 52 | Cost: 0.2488031 | Accuracy: 1.0000000
Iter: 53 | Cost: 0.0461277 | Accuracy: 1.0000000
Iter: 54 | Cost: 0.0293518 | Accuracy: 1.0000000
Iter: 55 | Cost: 0.0205454 | Accuracy: 1.0000000
Iter: 56 | Cost: 0.0352514 | Accuracy: 1.0000000
Iter: 57 | Cost: 0.0576767 | Accuracy: 1.0000000
Iter: 58 | Cost: 0.0291305 | Accuracy: 1.0000000
Iter: 59 | Cost: 0.0127137 | Accuracy: 1.0000000
Iter: 60 | Cost: 0.0058108 | Accuracy: 1.0000000
Iter: 61 | Cost: 0.0018002 | Accuracy: 1.0000000
Iter: 62 | Cost: 0.0014089 | Accuracy: 1.0000000
47s completed at 10:22 AM
20°C
Sunny
Search
10:22 AM
5/23/2024


Copy of QML-workshop-update
tutorial_variational_classifier.ipy
Quantum Machine Learning - G
Untitled9.ipynb - Colab
docs.google.com/presentation/d/18nRyOMHS8DR-xNEuw4vYfheGCr5yPD2D2_-k82oEfLg/edit#slide=id.g29e30b7e761_0_66
Quantum Machine Learning
File Edit View Insert Format Slide Arrange Tools Extensions Help
Slideshow
Share
Background Layout Theme Transition
Rec
7
Variational Quantum Classifier
8
Variational Quantum Classifier (VQC)
9
The VQC circuit
10
Steps
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|\Psi\rangle$
$R(\theta_{l,0,0}, \theta_{l,0,1}, \theta_{l,0,2})$
$R(\theta_{l,1,0}, \theta_{l,1,1}, \theta_{l,1,2})$
$R(\theta_{l,2,0}, \theta_{l,2,1}, \theta_{l,2,2})$
Click to add speaker notes
20°C
Sunny
Search
10:24 AM
5/23/2024

Copy of QML-workshop-update | tutorial_variational_classifier.ipy | Quantum Machine Learning - G | Untitled9.ipynb - Colab

colab.research.google.com/drive/1Yqdqqs9n5RurAjWMS8kf2WOyILtai1Z6#scrollTo=fOE4chnW1tPj

Untitled9.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

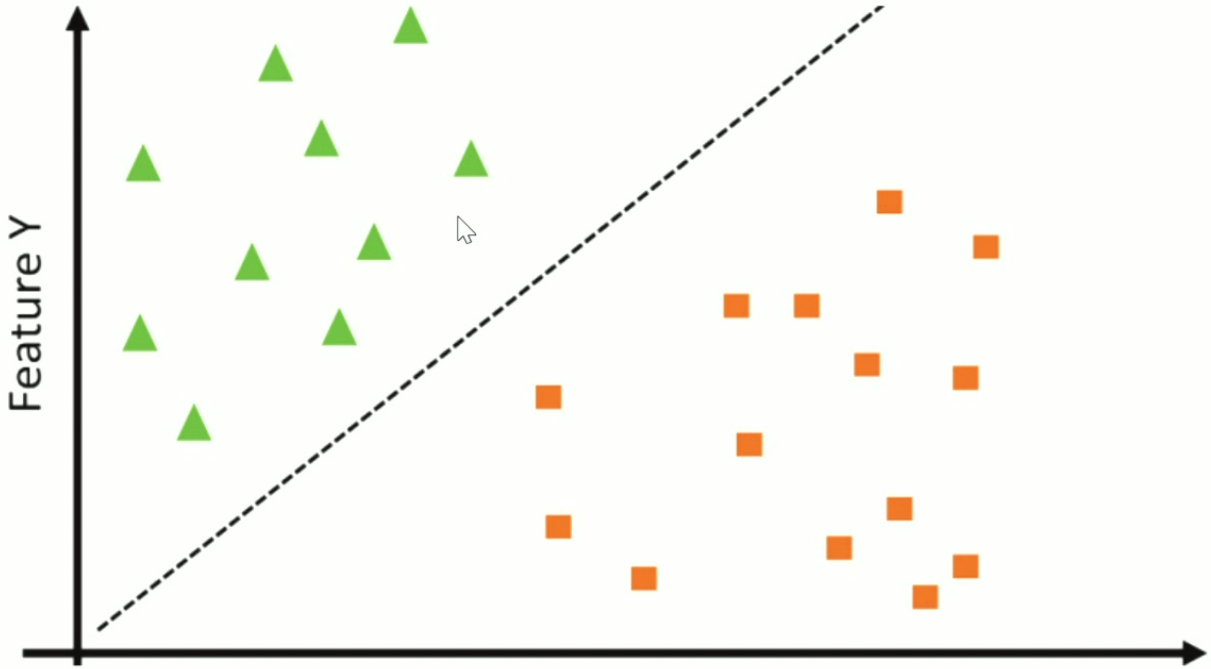
0s

```
for x,y,p in zip(X_test, Y_test, predictions_test):
    print(f"x = {x}, y = {y}, pred={p}")

acc_test = accuracy(Y_test, predictions_test)
print("Accuracy on unseen data:", acc_test)
```

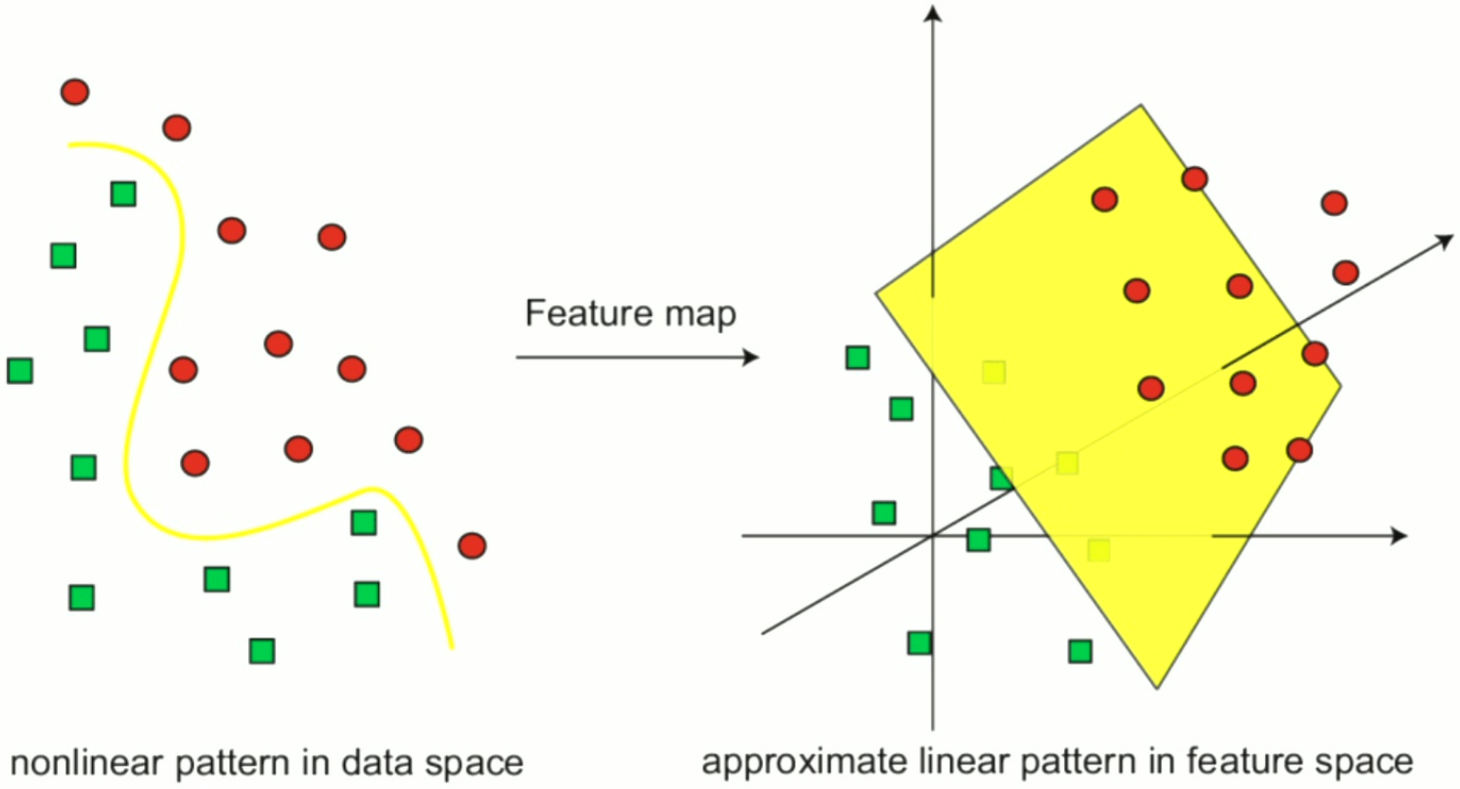
```
x = [0 0 0 0], y = -1, pred=-1.0
x = [0 0 1 1], y = -1, pred=-1.0
x = [1 0 1 0], y = -1, pred=-1.0
x = [1 1 1 0], y = 1, pred=1.0
x = [1 1 0 0], y = -1, pred=-1.0
x = [1 1 0 1], y = 1, pred=1.0
Accuracy on unseen data: 1.0
```

20°C Sunny | Search | 10:25 AM 5/23/2024

Support vector machine



Feature map


Feature map
nonlinear pattern in data space
approximate linear pattern in feature space

Copy of QML-workshop-update | PI_QML_Tutorial_2.ipynb - Colab | Quantum Machine Learning - G | Untitled9.ipynb - Colab

colab.research.google.com/drive/17ZHxPBToA8SrBVbE0eEv4kAwKfjDHvNa

PI_QML_Tutorial_2.ipynb

File Edit View Insert Runtime Tools Help Last saved at 9:19 AM

Comment Share

+ Code + Text Connect

(iv) The way an SVM with a custom kernel is implemented in scikit-learn requires us to pass a function that computes a matrix of kernel evaluations for samples in two different datasets A, B. If A=B, this is the Gram matrix. **Complete the `kernel_matrix` function below to build this matrix of kernel evaluations.**

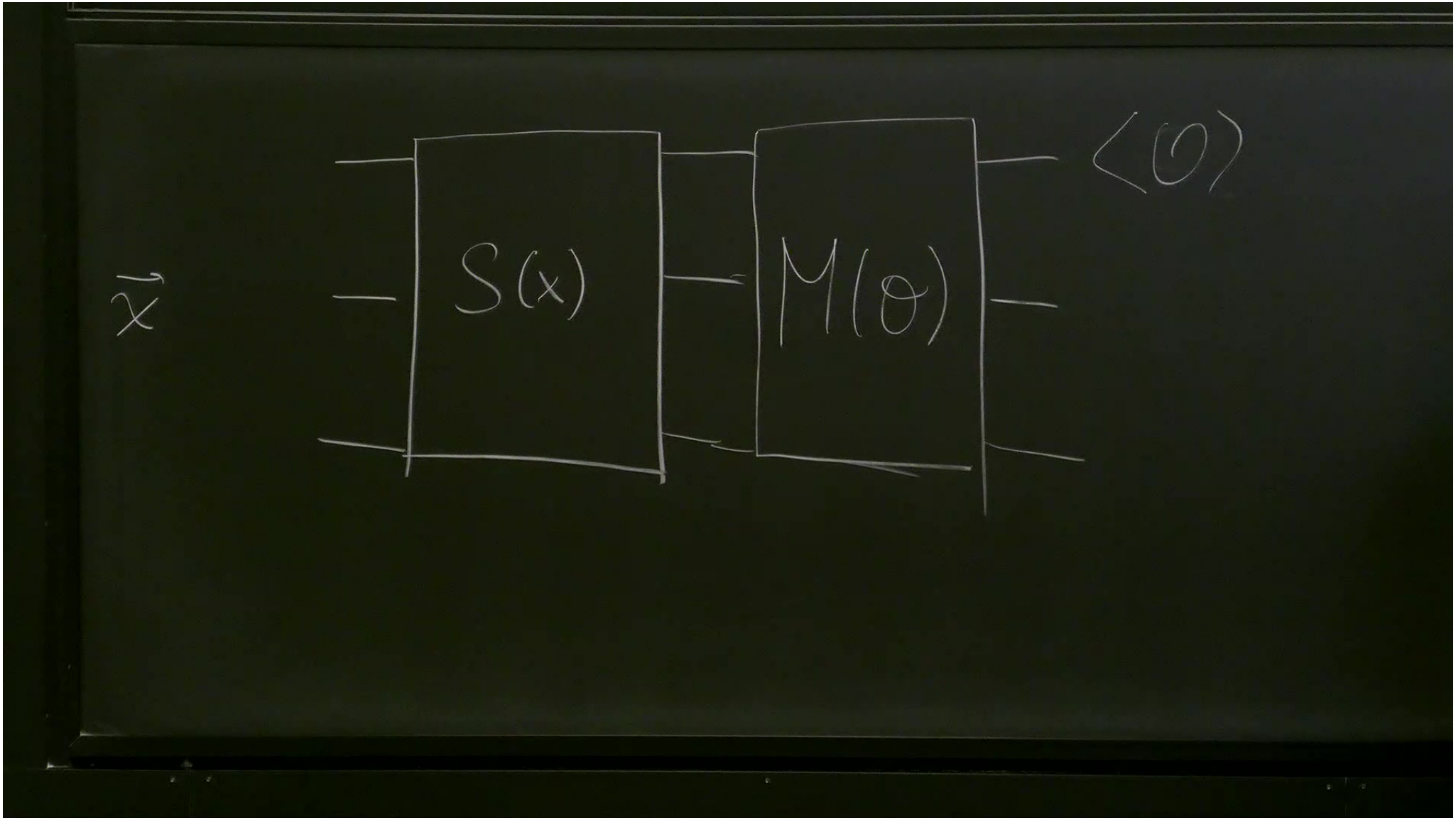
```
def kernel_matrix(A, B):
    """Compute the matrix whose entries are the kernel
       evaluated on pairwise data from sets A and B."""
    return # Return the kernel matrix
```

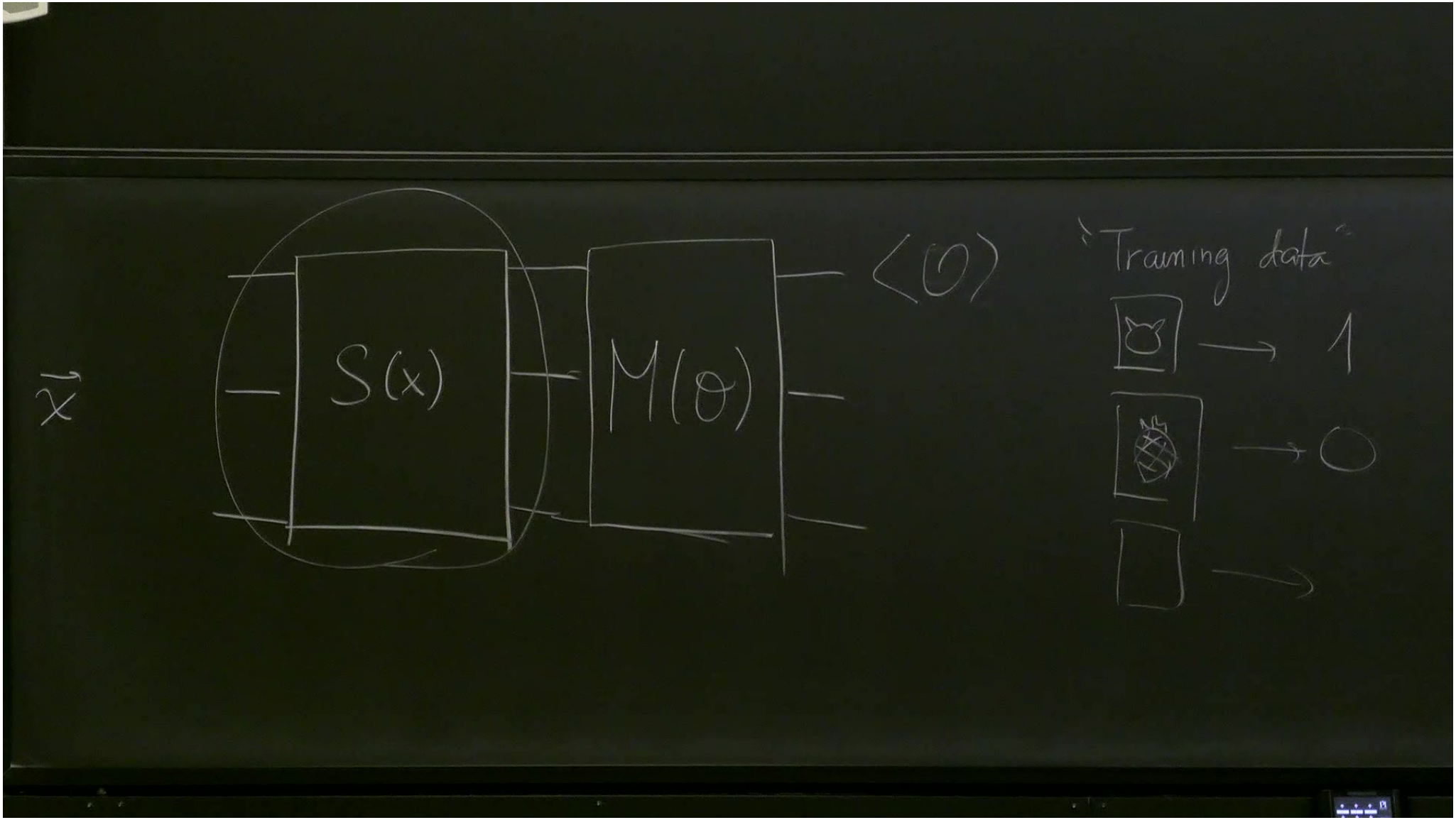
(v) Training the SVM optimizes internal parameters that basically weigh kernel functions. It is a breeze in scikit-learn, which is designed as a high-level machine learning library. **Run the cell below to fit the Support Vector Machine**. (I'm aware this is kind of black-boxy, but if you're interested in how this work, do read up on *normal* Machine Learning. You will find that one is very black-boxy too though).

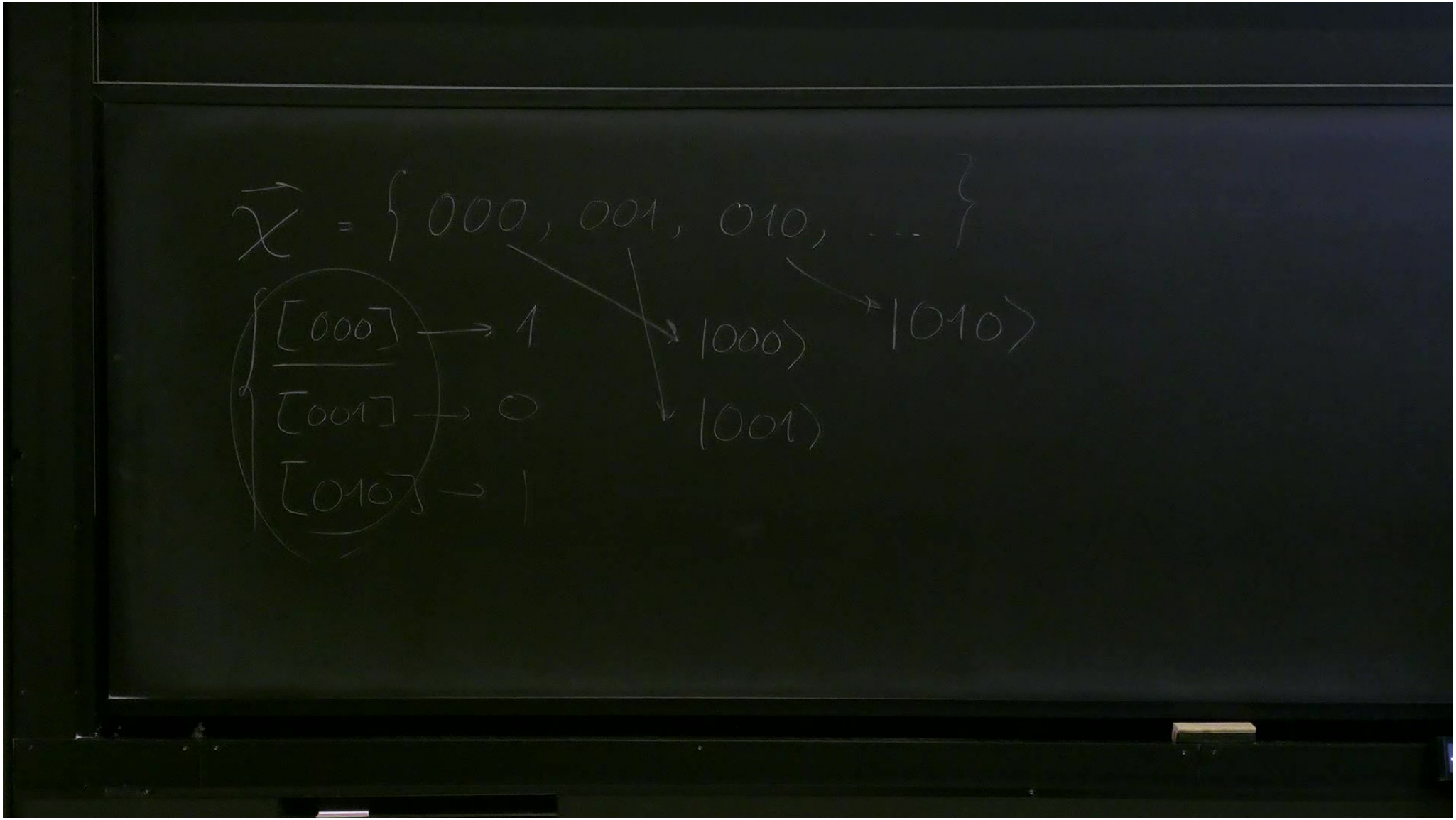
```
[ ]  svm = SVC(kernel=kernel_matrix).fit(X_train, y_train)
```

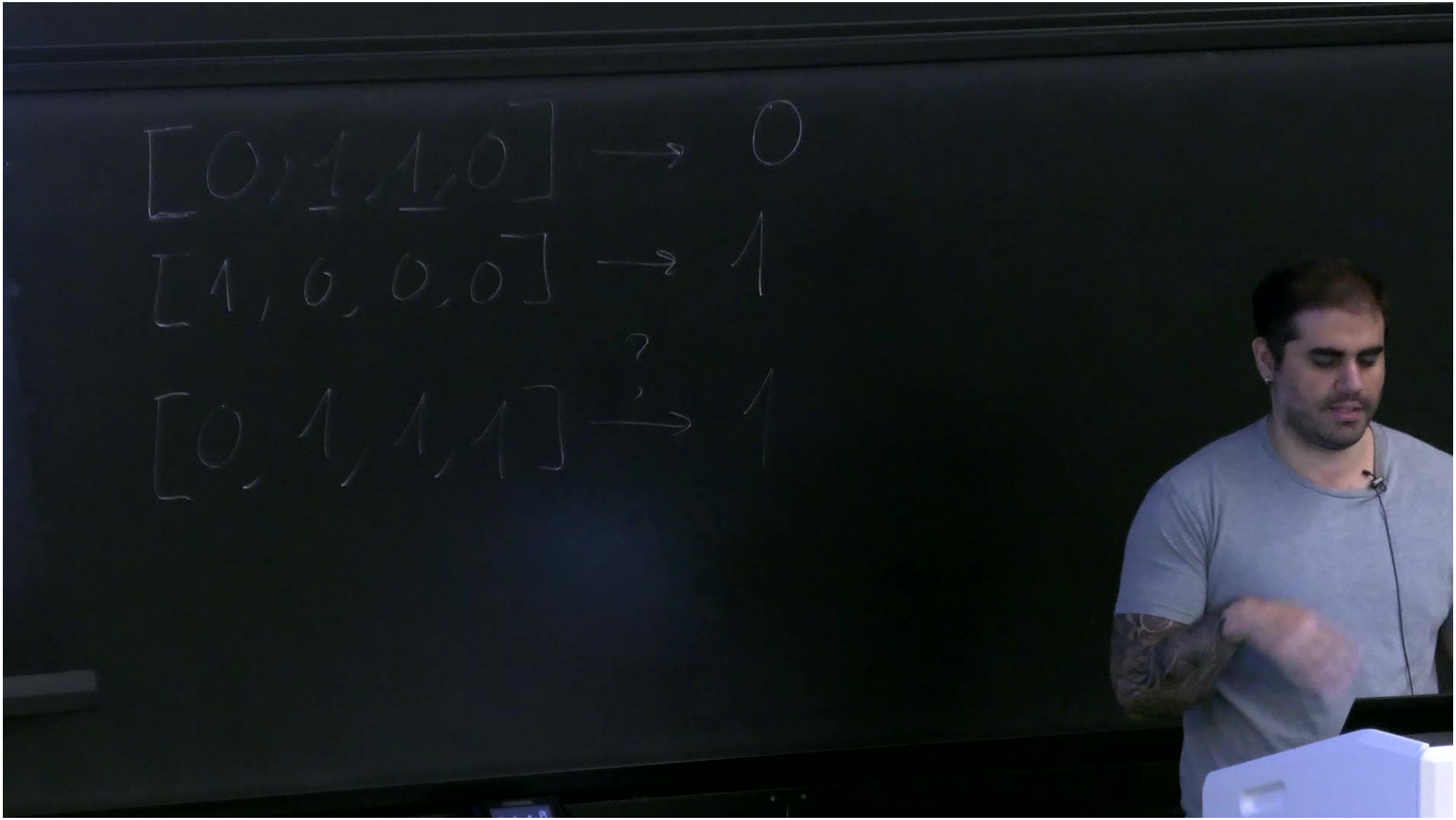
Now run the cell below to compute the accuracy on the test set

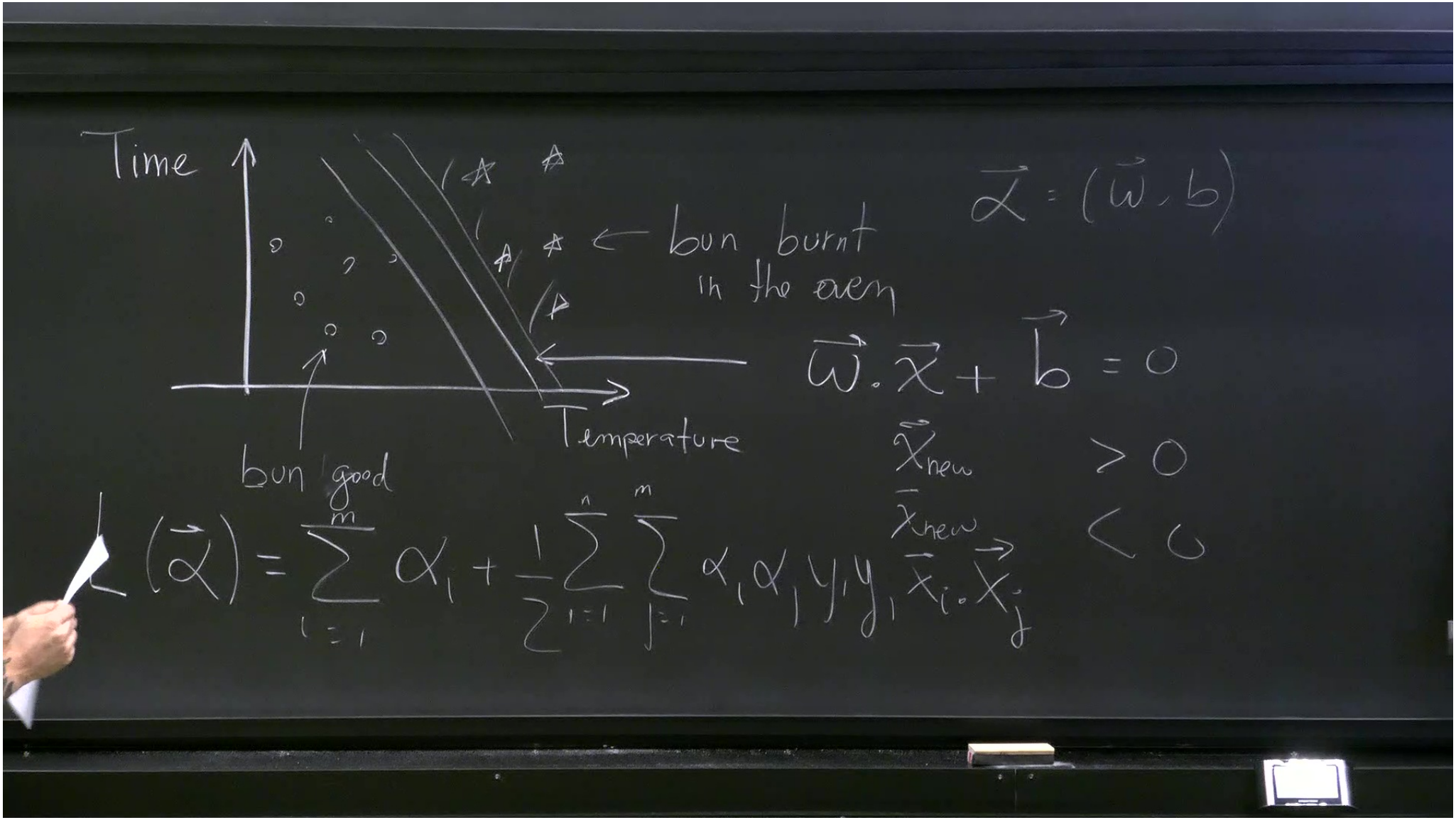
20°C Sunny | Search | 10:51 AM 5/23/2024

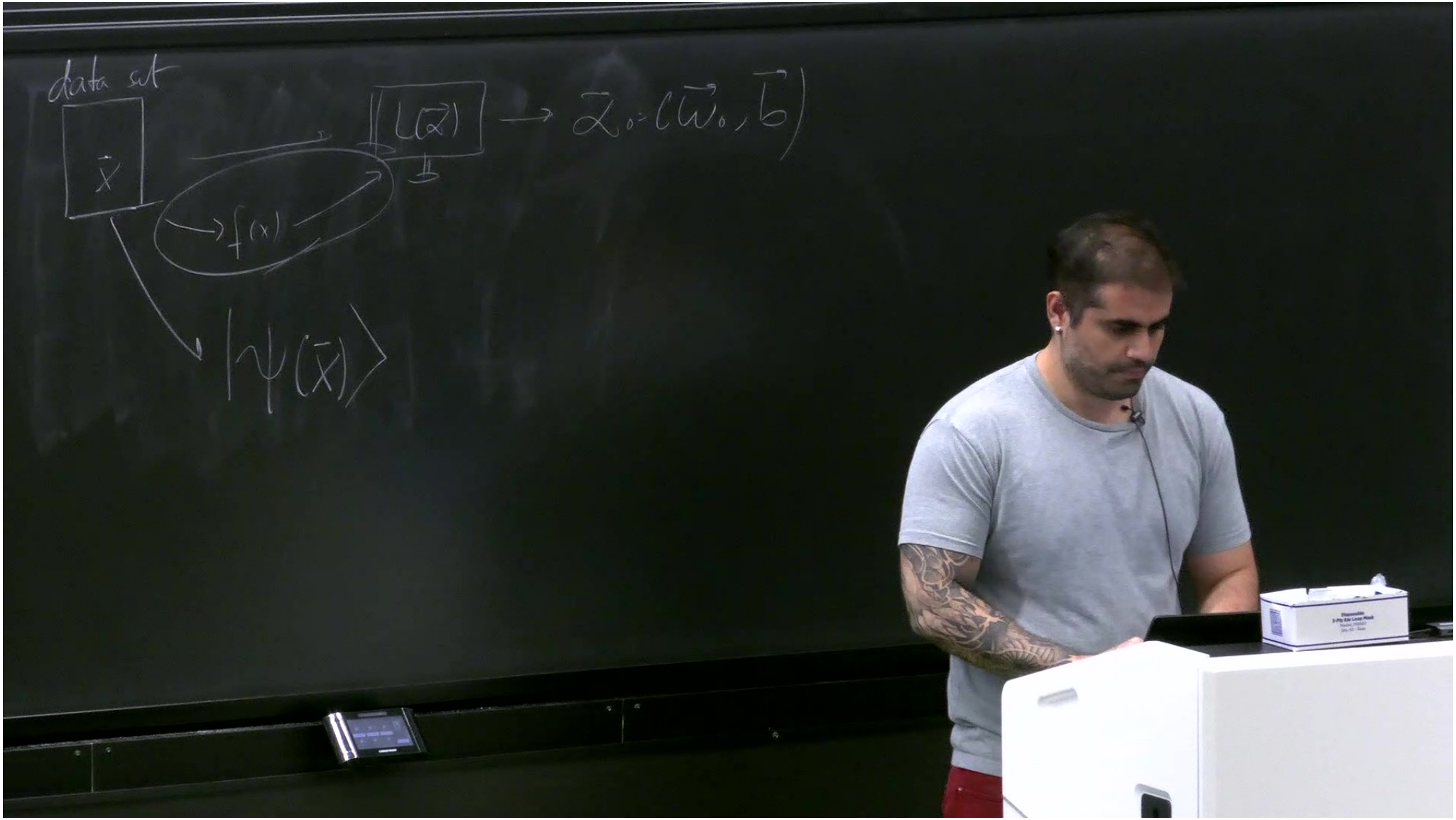

$\vec{x}$
S(x)
M(θ)
⟨O⟩


$\vec{x}$
S(x)
M(θ)
<O>
"Traning data"
1
0


$\vec{x} = \{000, 001, 010, \ldots\}$
[000] → 1
[001] → 0
[010] → 1
|000⟩
|001⟩
|010⟩


[0,1,1,0] → 0
[1,0,0,0] → 1
[0,1,1,1] → ? 1


Time
bun burnt in the oven
Temperature
bun good
$\vec{\alpha} = (\vec{w}, b)$
$\vec{w} \cdot \vec{x} + b = 0$
$\vec{x}_{new} > 0$
$\vec{x}_{new} < 0$
$L(\vec{\alpha}) = \sum_{i=1}^{m} \alpha_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j$


data set
$\vec{x}$
$L(\vec{x})$
$\vec{z}_0 = (\vec{w}_0, \vec{b})$
$f(x)$
$|\psi(\vec{x})\rangle$