QHack

Quantum Coding Challenges

CHALLENGE COMPLETED View successful submissions

Jump to code — Collapse text

The Magic 5-Ball — 100 points

Backstory

Doc Trine has been kidnapped by a freelance security analyst, Ove, and held in some sort of 'hyperjail'. Zenda and Reece need to figure out where it is and how to get there, using only the janky equipment lying around the office. Once they've rescued her, maybe Doc Trine will finally reveal the secret of timbits!

At Trine's desk, they find her notebook, full of diagrams, calculations, and dense, messy handwriting. Leafing through them, Zenda and Reece discover a note helpfully explaining what this hyperjail is: "*A 5-dimensional hypercube, accessed from some unknown point in deep space. Should get those robots out there some time.*" Robots? They keep reading: "*Cell number is in the magic 5-ball.*" They root around in the games room and find a dusty old magic 5-ball, which outputs 'yes' and 'no' answers to a 5-bit input. Presumably (though who knows how) Trine has concealed her location in this oracle. They have another problem, though. The fault-tolerant office equipment has disappeared along with Trine, leaving only noisy old circuitry!

Noisy Bernstein–Vazirani

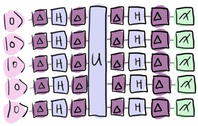
The oracle encodes the cell number $c \in \{0,1\}^5$, a 5-bit string. Consider the dot product

$$f(x) = x \cdot c = x_0c_0 + x_1c_1 + \cdots + x_5c_5 \mod 2,$$

where $x \in \{0,1\}^5$ is an arbitrary 5-bit string. The oracle is a unitary operator U_f which encodes the dot product f as a phase:

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle.$$

Zenda and Reece need to use this oracle to learn the starting positions c, but equipment in the old lab is noisy, with *depolarizing noise*, which with some probability λ replaces a qubit state with something random. Despite the noise, Zenda and Reece can attempt to learn Doc Trine's coordinates using the *Bernstein–Vazirani algorithm*. We picture the noisy circuit below:



Your goal: implement a noisy version of the Bernstein–Vazirani algorithm, using the noisy Hadamard gates provided. Will it work on the old computer?

Challenge code

In the code below, you are given various functions:

- `oracle_matrix`: which encodes Doc Trine's location in the hypercube.
- `noisy_Hadamard`: which applies a noisy Hadamard gate, with a probability `lmbda` of replacing an incoming or outgoing qubit state with something random.
- `noisy_BernsteinVazirani`: which implements the Bernstein-Vazirani algorithm using the oracle and the noisy Hadamard operation `noisy_Hadamard`. **You must complete this function**.

You may find this resource helpful:

- Bernstein-Vazirani algorithm

Inputs

The noisy Bernstein-Vazirani circuit `noisy_BernsteinVazirani` takes as input the probability `lmbda (float)` of replacing the state of a qubit.

Output

Your `noisy_BernsteinVazirani` circuit should correctly output the expectation value of the Pauli-Z operator on each qubit (`[float]`). The pattern of positive and negative expectations gives the cell number for any value of λ, showing that Bernstein–Vazirani is robust to noise.

If your solution matches the correct one within the given tolerance specified in `check` (in this case it's a `1e-4` relative error tolerance), the output will be `"Correct!"` Otherwise, you will receive a `"Wrong answer"` prompt.

Code Help

```
import json
import pennylane as qml
import pennylane.numpy as np
```

```
def noisy_Hadamard(lmbda, wire):
    """A Hadamard gate with depolarizing noise on either side.

    Args:
        lmbda (float): The parameter defining the depolarizing channel.
        wire (int): The wire the depolarizing channel acts on.
    """
    qml.DepolarizingChannel(lmbda, wires=wire)
    qml.Hadamard(wire)
    qml.DepolarizingChannel(lmbda, wires=wire)

# Oracle matrix for Doc Trine's cell number

flips = [1, 3, 5, 7, 8, 10, 12, 14, 16, 18, 20, 22, 25, 27, 29, 31]

oracle_matrix = np.eye(2**5)
for i in flips:
    oracle_matrix[i, i] = -1

# Implement the Bernstein-Vazirani algorithm with depolarizing noise

dev = qml.device("default.mixed", wires = 5)
@qml.qnode(dev)
def noisy_BernsteinVazirani(lmbda):
    """Runs the Bernstein-Vazirani algorithm with depolarizing noise.

    Args:
        lmbda (float): The probability of erasing the state of a qubit.

    Returns:
        (list(float)): Expectation values for PauliZ on all n wires.
    """
```

```
    # Put your code here #
```

```
    return [qml.expval(qml.PauliZ(i)) for i in range(5)]
```

```
# These functions are responsible for testing the solution.
def run(test_case_input: str) -> str:

    lmbda = json.loads(test_case_input)
    output = noisy_BernsteinVazirani(lmbda).tolist()

    return str(output)

def check(solution_output: str, expected_output: str) -> None:

    solution_output = json.loads(solution_output)
    expected_output = json.loads(expected_output)
    assert np.allclose(
        solution_output, expected_output, rtol=1e-4
    ), "Your noisy Bernstein-Vazirani algorithm isn't giving the right answers!"
```

```
test_cases = [['0.1', '[-0.5641679, -0.5641679,  0.5641679,  0.5641679, -0.5641679]']]
```

```
for i, (input_, expected_output) in enumerate(test_cases):
    print(f"Running test case {i} with input '{input_}'...")

    try:
        output = run(input_)

    except Exception as exc:
        print(f"Runtime Error. {exc}")

    else:
        if message := check(output, expected_output):
            print(f"Wrong Answer. Have: '{output}'. Want: '{expected_output}'.")

        else:
            print("Correct!")
```

Copy all — Reset — Open Notebook — Submit