QHack

Quantum Coding Challenges

🚀 CHALLENGE COMPLETED

View successful submissions

Jump to code — Collapse text

Ising Uprising

500 points

Backstory

Zenda and Reece model Sqynet as a spin chain, and they come up with a strategy. What if, in addition to using plasma bombs and missiles to increase the temperature of the device, they use a strong magnetic field? After all, magnetic fields might pass through Sqynet's outer shell more easily. The scientists proceed to simulate the effect of a magnetic field on a closed spin chain to quantify the effects.

Ground state of an Ising spin chain

A simple way to model Sqynet is by considering it as a closed spin chain of length N. A spin chain contains particles of spin $1/2$ in each of its N sites. The spins may be pointing in the positive or negative z direction, and we consider that there may be an external magnetic field acting on the system.

Such a quantum system is described by the *Transverse Ising Hamiltonian*. For closed spin chain with a transverse magnetic field of intensity h, the Transverse Ising Hamiltonian reads

$$H = -\sum_{i=1}^{N} Z_i \otimes Z_{i+1} - h\sum_{i}^{N} X_i.$$

The subindices i indicate the spin site where the operators act. In a closed spin chain, we identify site $N+1$ with the first site.

A possible plan for Zenda and Reece is to use a strong magnetic field that changes the ground energy of Sqynet, causing it to malfunction.

Your task is to help Zenda and Reece calculate the effect of external magnetic forces on the ground energy. Using the Variational Quantum Eigensolver (VQE) algorithm, you will compute the ground energy of a closed spin chain of length $N = 4$.

▶ Epilogue

Challenge code

In this challenge you will be given the following functions:

- `create_Hamiltonian`: In which you build the Transverse Ising Hamiltonian for $N = 4$ and a magnetic field intensity `h`. **You must complete this function.**
- `model`: This QNode builds a general enough ansatz for the ground state. This circuit must depend on some parameters `params`, which you will later optimize. It returns the expectation value of the Hamiltonian for the output state of the circuit. **You must complete this function.**
- `train`: This function returns the parameters that minimize the output of `model`. **You must complete this function.**

Input

As input to this problem, you are given:

- `h` (`float`): Magnetic field intensity applied to the spin chain.

Output

This code will output a `float` corresponding to the energy of the ground state.

If your solution matches the correct one within the given tolerance specified in `check` (in this case it's an relative tolerance of `0.1`), the output will be `"Correct!"` Otherwise, you will receive a `"Wrong answer"` prompt.

Good luck!

Code

Help

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

```python
def create_Hamiltonian(h):
    """
    Function in charge of generating the Hamiltonian of the statement.

    Args:
        h (float): magnetic field strength

    Returns:
        (qml.Hamiltonian): Hamiltonian of the statement associated to h
    """
```

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

```python
dev = qml.device("default.qubit", wires=4)

@qml.qnode(dev)
def model(params, H):
    """
    To implement VQE you need an ansatz for the candidate ground state!
    Define here the VQE ansatz in terms of some parameters (params) that
    create the candidate ground state. These parameters will
    be optimized later.

    Args:
        params (numpy.array): parameters to be used in the variational circuit
        H (qml.Hamiltonian): Hamiltonian used to calculate the expected value

    Returns:
        (float): Expected value with respect to the Hamiltonian H
    """
```

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

```python
def train(h):
    """
    In this function you must design a subroutine that returns the
    parameters that best approximate the ground state.

    Args:
        h (float): magnetic field strength

    Returns:
        (numpy.array): parameters that best approximate the ground state.
    """
```

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

```python
# These functions are responsible for testing the solution.
def run(test_case_input: str) -> str:
    ins = json.loads(test_case_input)
    params = train(ins)
    return str(model(params, create_Hamiltonian(ins)))


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-1
    ), "The expected value is not correct."
```

```python
test_cases = [['1.0', '-5.226251859505506'], ['2.3', '-9.66382463698038']]
```

```python
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