psr

November 11, 2022

[1]:
```
import qibo
import numpy as np
from qibo import hamiltonians, gates
from qibo.models import Circuit
from qibo.hamiltonians.abstract import AbstractHamiltonian
from qibo.config import raise_error
```

[5]:
```
# -*- coding: utf-8 -*-

def parameter_shift(
    circuit, hamiltonian, parameter_index, generator_eigenval,␣
 ↪initial_state=None
):
    """In this method the parameter shift rule (PSR) is implemented.
    Given a circuit U and an observable H, the PSR allows to calculate the␣
 ↪derivative
    of the expected value of H on the final state with respect to a variational␣
 ↪parameter of the circuit.

    Original references:
        `https://arxiv.org/abs/1811.11184`;
        `https://arxiv.org/abs/1803.00745`.

    Args:
        circuit (:class:`qibo.models.circuit.Circuit`): custom quantum circuit.
        observable (:class: `qibo.hamiltonians.Hamiltonian`): target observable.
        parameter_index (int): the index which identifies the target parameter␣
 ↪in the circuit.get_parameters() list
        gate_eigenv (float): abs(eigenvalue) of H. In case of Pauli 1/2{sigmas}␣
 ↪observable r = 0.5
        initial_state ((1, 2**nqubits) matrix): initial state on which we act␣
 ↪with the circuit.

    Returns:
        np.float value of the derivative of the expected value of H on the␣
 ↪final state obtained applying U
        to the initial state with respect to the target variational parameter.
```

```
Example:
    .. testcode::

        import qibo
        import numpy as np
        from qibo import hamiltonians, gates
        from qibo.models import Circuit

        # in order to see the difference with tf gradients
        import tensorflow as tf
        qibo.set_backend('tensorflow')

        # defining an observable
        def hamiltonian(nqubits = 1):
            m0 = (1/nqubits)*hamiltonians.Z(nqubits).matrix
            ham = hamiltonians.Hamiltonian(nqubits, m0)
            return ham

        # defining a dummy circuit
        def circuit(nqubits = 1):
            c = Circuit(nqubits = 1)
            c.add(gates.RY(q = 0, theta = 0))
            c.add(gates.RX(q = 0, theta = 0))
            c.add(gates.M(0))
            return c

        # using GradientTape to benchmark
        def gradient_tape(params):
            params = tf.Variable(params)

            with tf.GradientTape() as tape:
                c = circuit(nqubits = 1)
                c.set_parameters(params)
                h = hamiltonian()
                expected_value = h.expectation(c.execute().state())

            grads = tape.gradient(expected_value, [params])
            return grads

        # initializing the circuit
        c = circuit(nqubits = 1)

        # some parameters
        test_params = np.random.randn(2)
        c.set_parameters(test_params)
```

```
            # a test hamiltonian
            test_hamiltonian = hamiltonian()

            # running the psr with respect to the two parameters
            grad_0 = parameter_shift(circuit = c, observable =␣
↪test_hamiltonian, parameter_index = 0, gate_eigenv = 0.5)
            grad_1 = parameter_shift(circuit = c, observable =␣
↪test_hamiltonian, parameter_index = 1, gate_eigenv = 0.5)

            # evaluating the gradients using tensorflow
            tf_grads = gradient_tape(test_params)

            print('Test gradient with respect params[0] with PSR: ', grad_0.
↪numpy())
            print('Test gradient with respect params[0] with tf:  ',␣
↪tf_grads[0][0].numpy())
            print('Test gradient with respect params[1] with PSR: ', grad_1.
↪numpy())
            print('Test gradient with respect params[1] with tf:  ',␣
↪tf_grads[0][1].numpy())

    """

    if parameter_index > len(circuit.get_parameters()):
        raise_error(ValueError, """This index is out of bounds.""")

    if not isinstance(hamiltonian, AbstractHamiltonian):
        raise_error(TypeError, 'hamiltonian must be a qibo.hamiltonians.
↪Hamiltonian or qibo.hamiltonians.SymbolicHamiltonian object')

    # inheriting hamiltonian's backend
    backend = hamiltonian.backend

    # defining the shift according to the psr
    s = np.pi / (4 * generator_eigenval)

    # saving original parameters and making a copy
    original = np.asarray(circuit.get_parameters()).copy()
    shifted = original.copy()

    # forward shift and evaluation
    shifted[parameter_index] += s
    circuit.set_parameters(shifted)

    forward = hamiltonian.expectation(backend.execute_circuit(circuit=circuit,␣
↪initial_state=initial_state).state())
```

```
    # backward shift and evaluation
    shifted[parameter_index] -= 2 * s
    circuit.set_parameters(shifted)

    backward = hamiltonian.expectation(backend.execute_circuit(circuit=circuit,␣
 ↪initial_state=initial_state).state())

    # restoring the original circuit
    circuit.set_parameters(original)

    return generator_eigenval * (forward - backward)
```

[8]:
```
# in order to see the difference with tf gradients
import tensorflow as tf
qibo.set_backend('tensorflow')

# defining an observable
def hamiltonian(nqubits = 1):
    m0 = (1/nqubits)*hamiltonians.Z(nqubits).matrix
    ham = hamiltonians.Hamiltonian(nqubits, m0)
    return ham

# defining a dummy circuit
def circuit(nqubits = 1):
    c = Circuit(nqubits = 1)
    c.add(gates.RY(q = 0, theta = 0))
    c.add(gates.RX(q = 0, theta = 0))
    c.add(gates.M(0))
    return c

# using GradientTape to benchmark
def gradient_tape(params):
    params = tf.Variable(params)

    with tf.GradientTape() as tape:
        c = circuit(nqubits = 1)
        c.set_parameters(params)
        h = hamiltonian()
        expected_value = h.expectation(c.execute().state())

    grads = tape.gradient(expected_value, [params])
    return grads

# initializing the circuit
c = circuit(nqubits = 1)
```

```
# some parameters
test_params = np.random.randn(2)
c.set_parameters(test_params)

test_hamiltonian = hamiltonian()

# running the psr with respect to the two parameters
grad_0 = parameter_shift(circuit = c, hamiltonian = test_hamiltonian,␣
 ↪parameter_index = 0, generator_eigenval = 0.5)
grad_1 = parameter_shift(circuit = c, hamiltonian = test_hamiltonian,␣
 ↪parameter_index = 1, generator_eigenval = 0.5)

tf_grads = gradient_tape(test_params)

print('Test gradient with respect params[0] with PSR: ', grad_0.numpy())
print('Test gradient with respect params[0] with tf:  ', tf_grads[0][0].numpy())
print('Test gradient with respect params[0] with PSR: ', grad_1.numpy())
print('Test gradient with respect params[0] with tf:  ', tf_grads[0][1].numpy())
```

```
[Qibo 0.1.8|INFO|2022-10-24 14:46:15]: Using tensorflow backend on /device:CPU:0

Test gradient with respect params[0] with PSR:  0.5379036403446502
Test gradient with respect params[0] with tf:   0.5379036403446502
Test gradient with respect params[0] with PSR:  0.330061454113827
Test gradient with respect params[0] with tf:   0.33006145411382704
```

[ ]:

[ ]: