Quantum Generative Adversarial Network with Noise

Project Name: Quantum Generative Adversarial Network with Noise

Project member:

XXX
XXX
XXX

Dodument Type: Report

Project Start Time: 26/04/2020

Sourcecode Version: 0.0.1

Keywords: Variational Quantum Circuit, Machine Learning

Modify April 26, 2020

Submitted by:

YIXUAN ZHU

Contents

1 Experiment

Run the code then draw the picture
maxcut gradient
maxcut adam
maxcut adagrad
maxcut COBYLA
maxcut momentum

2 Next Plan

P: run the next code
GAN

3 Appendix

A Source Code

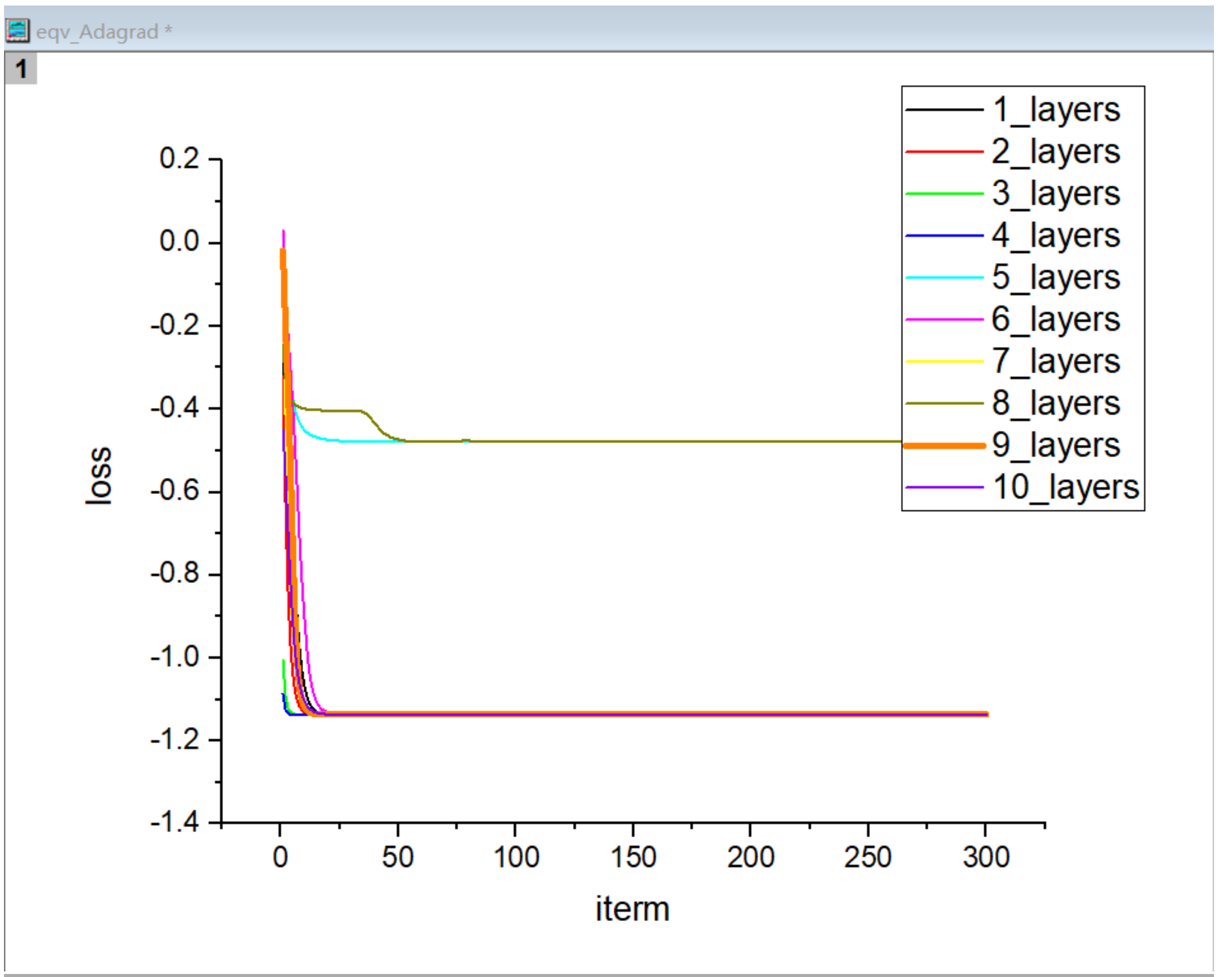
```

import pennylane as qml
from pennylane import numpy as np
geometry = 'h2.xyz'

import pickle


def load_model(file_path):
    with open(file_path, 'rb') as qc:
        model = pickle.load(qc)
    return model


h = load_model('./hamiltonian/h_test.xyz')
nr_qubits = 4

print('Number of qubits = ', nr_qubits)
print('Hamiltonian is ', h)

##############################################################################
# That's it! From here on, we can use PennyLane as usual, employing its entire stack
    of
# algorithms and optimizers.
#
# Implementing the VQE algorithm
# ------------------------------
#
# PennyLane contains the :class:`~.VQECost` class, specifically
# built to implement the VQE algorithm. We begin by defining the device, in this case
    a simple
```

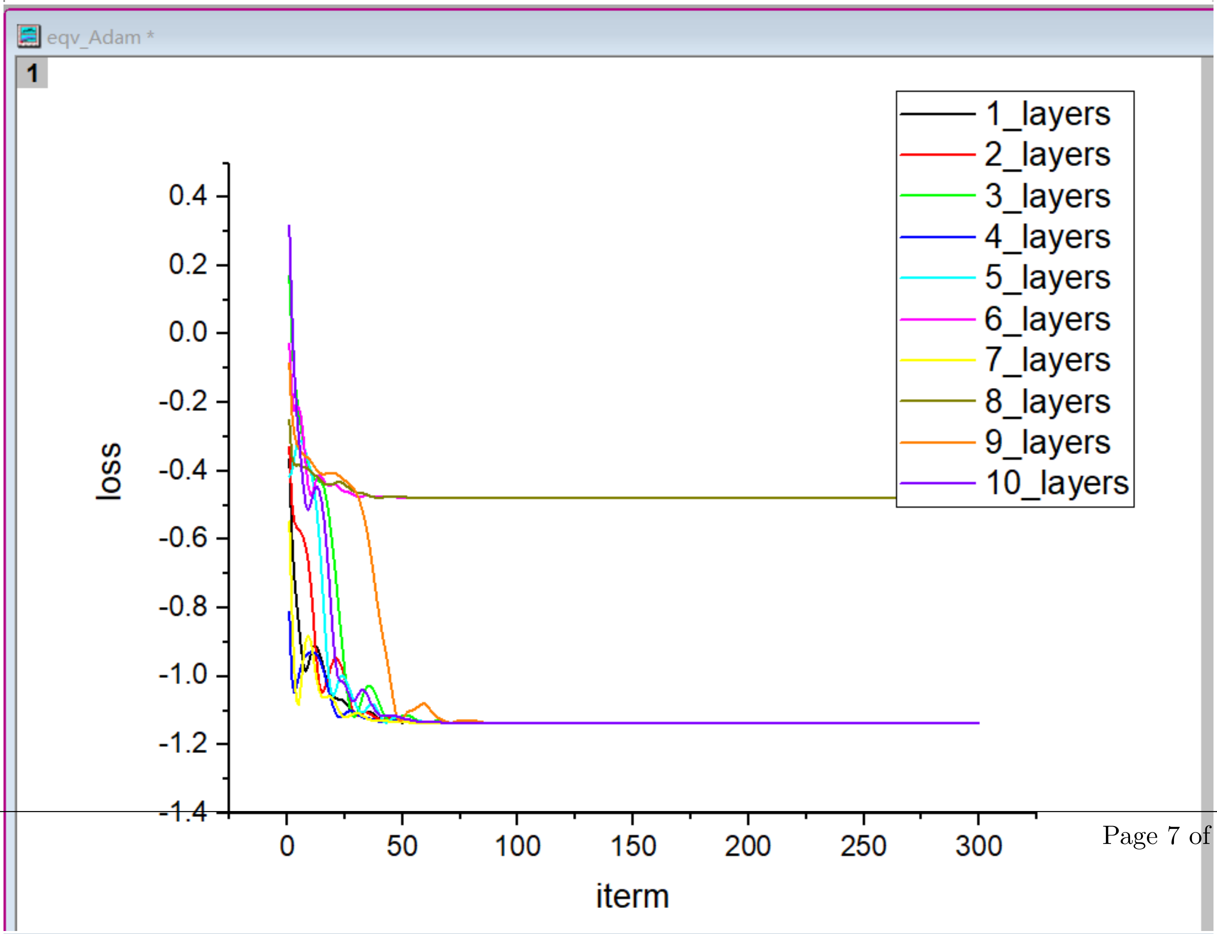
```
# qubit simulator:

dev = qml.device('default.qubit', wires=nr_qubits)


##############################################################################
# In VQE, the goal is to train a quantum circuit to prepare the ground state of the
    input
# Hamiltonian. This requires a clever choice of circuit, which should be complex
    enough to
# prepare the ground state, but also sufficiently easy to optimize. In this example,
    we employ a
# variational circuit that is capable of preparing the normalized states of the form
# :math:`\alpha|1100\rangle + \beta|0011\rangle` which encode the ground state wave
    function of
# the hydrogen molecule described with a minimal basis set. The circuit consists of
    single-qubit
# rotations on all wires, followed by three entangling CNOT gates, as shown in the
    figure below:
#
# |
#
# .. figure:: /demonstrations/variational_quantum_eigensolver/sketch_circuit.png
#     :width: 50%
#     :align: center
#
# |
#

##############################################################################
# In the circuit, we apply single-qubit rotations, followed by CNOT gates:

'''
def circuit(params, wires):
    qml.BasisState(np.array([1, 1, 0, 0]), wires=wires)
    for i in wires:
        qml.Rot(*params[i], wires=i)
    qml.CNOT(wires=[2, 3])
    qml.CNOT(wires=[2, 0])
    qml.CNOT(wires=[3, 1])
'''

def circuit(params, wires, n_layers=1):
    qml.BasisState(np.array([1, 1, 0, 0]), wires=wires)

    for i in range(n_layers):
        for j in wires:
            qml.Rot(params[i,j,0], params[i,j,1],params[i,j,2],wires=j)
        qml.CNOT(wires=[2, 3])
        qml.CNOT(wires=[2, 0])
        qml.CNOT(wires=[3, 1])


```

```python
##############################################################################
# .. note::
#
#     The qubit register has been initialized to :math:`|1100\rangle` which encodes the
#     Hartree-Fock state of the hydrogen molecule described with a `minimal basis
#     <https://en.wikipedia.org/wiki/Basis_set_(chemistry)#Minimal_basis_sets>`__.
#
# The cost function for optimizing the circuit can be created using the :class:`~.VQECost`
# class, which is tailored for VQE optimization. It requires specifying the
# circuit, target Hamiltonian, and the device, and returns a cost function that can
# be evaluated with the circuit parameters:


cost_fn = qml.VQECost(circuit, h, dev)

##############################################################################
# Wrapping up, we fix an optimizer and randomly initialize circuit parameters. For reliable
# results, we fix the seed of the random number generator, since in practice it may be necessary
# to re-initialize the circuit several times before convergence occurs.

opt = qml.AdamOptimizer(stepsize=0.4)
np.random.seed()
num_layers = 1
params = np.random.normal(0, np.pi, (num_layers, nr_qubits, 3))

print(params)

##############################################################################
# We carry out the optimization over a maximum of 200 steps, aiming to reach a convergence
# tolerance (difference in cost function for subsequent optimization steps) of :math:`\sim 10^{
# -6}`.

import xlrd

from xlutils.copy import copy as xl_copy

#V
'''
rb = xlrd.open_workbook("./DATA/vqe_Adam_DATA.xls",formatting_info=True)
workbook=xl_copy(rb)
print(workbook)
sheet = rb.sheets()[0]
col =sheet.ncols
sheet = workbook.get_sheet(0)
'''

max_iterations = 300
```

```
conv_tol = 1e-06

prev_energy = cost_fn(params,n_layers=num_layers)
for n in range(max_iterations):
    params = opt.step(cost_fn, params)
    energy = cost_fn(params)
    conv = np.abs(energy - prev_energy)

    print('Iteration = {:},  Ground-state energy = {:.8f} Ha,  Convergence parameter = {'
          ':.8f} Ha'.format(n, energy, conv))

    #sheet.write(n, col, "{:0.7f}".format(energy))
    '''
    if conv <= conv_tol:
        break
    '''
    prev_energy = energy

'''
workbook.save('./DATA/vqe_Adam_DATA.xls')
'''

print()
print('Final convergence parameter = {:.8f} Ha'.format(conv))
print('Final value of the ground-state energy = {:.8f} Ha'.format(energy))
print('Accuracy with respect to the FCI energy: {:.8f} Ha ({:.8f} kcal/mol)'.
    format(np.abs(energy - (-1.136189454088)), np.abs(energy - (-1.136189454088)) * 627.503))
print()
print('Final circuit parameters = \n', params)

##############################################################################
# Success! ??? The ground-state energy of the hydrogen molecule has been estimated with chemical
# accuracy (< 1 kcal/mol) with respect to the exact value of -1.136189454088 Hartree (Ha) obtained
# from a full configuration-interaction (FCI) calculation. This is because, for the optimized
# values of the single-qubit rotation angles, the state prepared by the VQE ansatz is precisely
# the FCI ground-state of the :math:`H_2` molecule :math:`|H_2\rangle_{gs} = 0.99 |1100\rangle - 0.10
# |0011\rangle`.
#
# What other molecules would you like to study using PennyLane?
#
# .. _vqe_references:
#
# References
# ----------
#
```

```
# 1. Alberto Peruzzo, Jarrod McClean *et al.*, "A variational eigenvalue solver on a photonic
#    quantum processor". `Nature Communications 5, 4213 (2014).
#    <https://www.nature.com/articles/ncomms5213?origin=ppub>`__
#
# 2. Yudong Cao, Jonathan Romero, *et al.*, "Quantum Chemistry in the Age of Quantum Computing".
#    `Chem. Rev. 2019, 119, 19, 10856-10915.
#    <https://pubs.acs.org/doi/10.1021/acs.chemrev.8b00803>`__
```


eqv_Adagrad *
1
loss
iterm
1_layers
2_layers
3_layers
4_layers
5_layers
6_layers
7_layers
8_layers
9_layers
10_layers
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.2
-1.4
0
50
100
150
200
250
300


eqv_Adam *
1
loss
iterm
1_layers
2_layers
3_layers
4_layers
5_layers
6_layers
7_layers
8_layers
9_layers
10_layers
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.2
-1.4
0
50
100
150
200
250
300