吉布斯态的制备

(GIBBS STATE PREPARATION)

概览

- 在本案例中，我们将展示如何通过Paddle Quantum训练量子神经网络来制备量子吉布斯态。
- 让我们通过下面几行代码引入必要的library和package。

```
import scipy
from numpy import array, concatenate, zeros
from numpy import pi as PI
from numpy import trace as np_trace
from paddle import fluid
from paddle.complex import matmul, trace
from paddle_quantum.circuit import UAnsatz
from paddle_quantum.state import density_op
from paddle_quantum.utils import state_fidelity, partial_trace,
hermitian, pauli_str_to_matrix
```

背景

量子计算中的前沿方向包含量子机器学习和量子优化，在这两个方向中，特定量子态的制备是非常重要的问题。特别的，吉布斯态（Gibbs state）的制备是实现诸多量子算法所必须的步骤并且广泛应用于：

- 量子机器学习中受限波尔兹曼机的学习 (1)
- 解决凸优化和半正定规划等优化问题 (2)
- 组合优化问题 (3)

具体的吉布斯态定义如下：给定一个 n 量子位的哈密顿量 H（一般来说这是一个$2^n \times 2^n$的厄米矩阵），其在温度 T 下的吉布斯态为

$$\rho_G = \frac{e^{-\beta H}}{\mathrm{tr}(e^{-\beta H})}$$

其中 $e^{-\beta H}$ 是矩阵 $-\beta H$ 的矩阵指数，$\beta = \frac{1}{kT}$ 是系统的逆温度参数，其中 T 是温度参数，k 是玻尔兹曼常数 (这个例子中我们取 $k = 1$)。作为一个上手的例子，这里我们首先考虑一个3量子比特的哈密顿量及其吉布斯态。

$$H = -Z \otimes Z \otimes I - I \otimes Z \otimes Z - Z \otimes I \otimes Z, \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

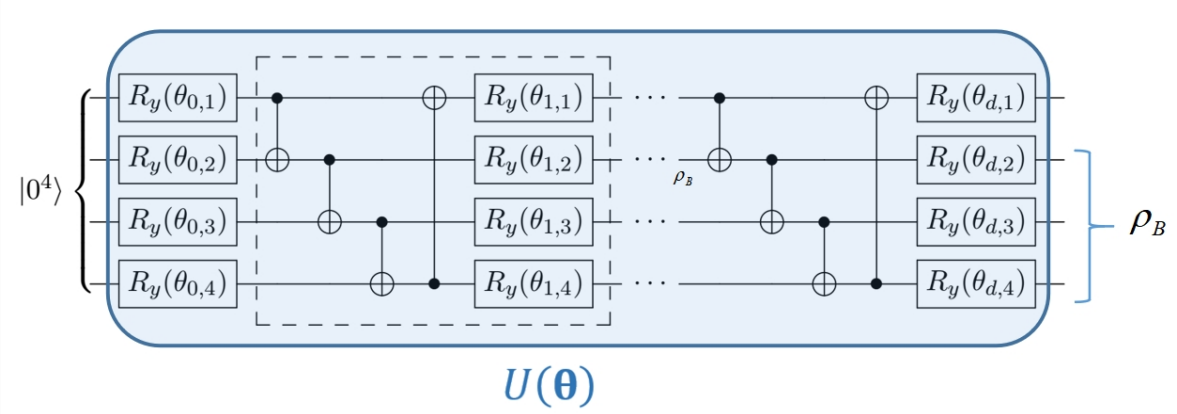
这个例子中，我们将逆温度参数设置为 $\beta = 1.5$。此外，为了方便测试结果，我们按照定义提前生成好了理想情况的吉布斯态 ρ_G。

```
N = 4          # 量子神经网络的宽度
N_SYS_B = 3    # 用于生成吉布斯态的子系统B的量子比特数
SEED = 14       # 固定随机种子
```

```
beta = 1.5    # 设置逆温度参数 beta

# 生成用泡利字符串表示的特定的哈密顿量
H = [[-1.0, 'z0,z1'], [-1.0, 'z1,z2'], [-1.0, 'z0,z2']]

# 生成哈密顿量的矩阵信息
hamiltonian = pauli_str_to_matrix(H, N_SYS_B)

# 生成理想情况下的目标吉布斯态 rho
rho_G = scipy.linalg.expm(-1 * beta * hamiltonian) /
np_trace(scipy.linalg.expm(-1 * beta * hamiltonian))

# 设置成 Paddle quantum 所支持的数据类型
hamiltonian = hamiltonian.astype("complex128")
rho_G = rho_G.astype("complex128")
```

搭建量子神经网络

- 在这个案例中，我们将通过训练量子神经网络QNN（也可以理解为参数化量子电路）来训练吉布斯态。这里，我们提供一个简单的4量子位的量子电路如下：



- 我们需要预设一些电路的参数，比如电路有4量子比特，其中第1个量子位是辅助系统，第2-4个量子位是用以产生吉布斯态的子系统。

- 初始化其中的变量参数，θ 代表我们量子神经网络中的参数组成的向量。

接下来我们根据上图中的电路设计，通过 Paddle Quantum 的 `UAnsatz` 函数和内置的 `real_entangled_layer(theta, D)` 电路模板来高效搭建量子神经网络。

```
def U_theta(initial_state, theta, N, D):
    """
    Quantum Neural Network
    """

    # 按照量子比特数量/网络宽度初始化量子神经网络
    cir = UAnsatz(N)

    # 内置的 {R_y + CNOT} 电路模板
    cir.real_entangled_layer(theta[:D], D)

    # 铺上最后一列 R_y 旋转门
    for i in range(N):
        cir.ry(theta=theta[D][i][0], which_qubit=i)

    # 量子神经网络作用在给定的初始态上
    final_state = cir.run_density_matrix(initial_state)

    return final_state
```

配置训练模型 - 损失函数

- 现在我们已经有了数据和量子神经网络的架构，我们将进一步定义合适的训练参数、模型和损失函数来达到我们的目标。
- 具体的我们参考的是论文[4]中的方法，核心思想是**利用吉布斯态达到了最小自由能**的性质。
- 通过作用量子神经网络 $U(\theta)$ 在初始态上，我们可以得到输出态 $|\psi(\theta)\rangle$，其在第2-4个量子位的态记为 $\rho_B(\theta)$。
- 设置训练模型中的的损失函数。在吉布斯态学习中，我们利用冯诺依曼熵函数的截断来进行自由能的估计，相应的损失函数参考[4]可以设为 $loss = L_1 + L_2 + L_3$，其中

$$L_1 = \mathrm{tr}(H\rho_B), \quad L_2 = 2\beta^{-1}\mathrm{tr}(\rho_B^2), \quad L_3 = -\beta^{-1}\big(\mathrm{tr}(\rho_B^3) + 3\big)/2$$

```
class Net(fluid.dygraph.Layer):
    """
    Construct the model net
    """

    def __init__(self, shape, param_attr=fluid.initializer.Uniform(
        low=0.0, high=2*PI, seed=SEED), dtype='float64'):
        super(Net, self).__init__()

        # 初始化 theta 参数列表，并用 [0, 2*pi] 的均匀分布来填充初始值
        self.theta = self.create_parameter(shape=shape,
                     attr=param_attr, dtype=dtype, is_bias=False)

        # 初始化 rho = |0..0><0..0| 的密度矩阵
        self.initial_state=fluid.dygraph.to_variable(density_op(N))

    # 定义损失函数和前向传播机制
    def forward(self, H, N, N_SYS_B, D):

        # 施加量子神经网络
        rho_AB = U_theta(self.initial_state, self.theta, N, D)

        # 计算偏迹 partial trace 来获得子系统B所处的量子态 rho_B
        rho_B = partial_trace(rho_AB,
                2 ** (N - N_SYS_B), 2 ** (N_SYS_B), 1)

        # 计算三个子损失函数
        rho_B_squre = matmul(rho_B, rho_B)
        loss1 = (trace(matmul(rho_B, H))).real
        loss2 = (trace(rho_B_squre)).real * 2 / beta
        loss3 = - ((trace(matmul(rho_B_squre, rho_B))).real + 3)
                                                    / (2 * beta)

        # 最终的损失函数
        loss = loss1 + loss2 + loss3

        return loss, rho_B
```

配置训练模型 - 模型参数

在进行量子神经网络的训练之前，我们还需要进行一些训练的超参数设置，主要是学习速率 (LR, learning rate)、迭代次数(ITR, iteration)和量子神经网络计算模块的深度 (D, Depth)。这里我们设定学习速率为0.5, 迭代次数为50次。读者不妨自行调整来直观感受下超参数调整对训练效果的影响。

```
ITR = 50 # 设置训练的总迭代次数
LR = 0.5 # 设置学习速率
D = 1    # 设置量子神经网络中重复计算模块的深度 Depth
```

进行训练

- 当训练模型的各项参数都设置完成后，我们将数据转化为 Paddle 动态图中的变量，进而进行量子神经网络的训练。
- 训练过程中我们用的是 Adam Optimizer，也可以调用 Paddle 中提供的其他优化器。
- 我们将训练过程中的结果依次输出。
- 特别的我们依次输出了我们学习到的量子态 $\rho_B(\theta)$ 与吉布斯态 ρ_G 的保真度，保真度越高说明QNN输出的态越接近于吉布斯态。

```
# 初始化paddle动态图机制
with fluid.dygraph.guard():

    # 我们需要将 Numpy array 转换成 Paddle 动态图模式中支持的 variable
    H = fluid.dygraph.to_variable(hamiltonian)

    # 确定网络的参数维度
    net = Net(shape=[D + 1, N, 1])

    # 一般来说，我们利用Adam优化器来获得相对好的收敛，
    # 当然你可以改成SGD或者是RMS prop.
    opt = fluid.optimizer.AdamOptimizer(learning_rate=LR,
                          parameter_list=net.parameters())

     # 优化循环
    for itr in range(1, ITR + 1):

        # 前向传播计算损失函数并返回生成的量子态 rho_B
        loss, rho_B = net(H, N, N_SYS_B, D)

        # 在动态图机制下，反向传播极小化损失函数
        loss.backward()
        opt.minimize(loss)
        net.clear_gradients()

        # 转换成 Numpy array 用以计算量子态的保真度 F(rho_B, rho_G)
        rho_B = rho_B.numpy()
        fid = state_fidelity(rho_B, rho_G)

        # 打印训练结果
        if itr % 10 == 0:
            print('iter:', itr, 'loss:', '%.4f' % loss.numpy(),
                                  'fid:', '%.4f' % fid)
```

```
iter: 10 loss: -3.1189 fid: 0.9504
iter: 20 loss: -3.3502 fid: 0.9846
iter: 30 loss: -3.3630 fid: 0.9873
iter: 40 loss: -3.4087 fid: 0.9948
iter: 50 loss: -3.4110 fid: 0.9953
```

总结

根据上面训练得到的结果，通过大概50次迭代，我们就能达到高于99.5%保真度的高精度吉布斯态，高效并精确地完成了吉布斯态的制备。我们可以通过print函数来输出学习到的量子神经网络的参数和它的输出态。

参考文献

(1) Kieferová, M. & Wiebe, N. Tomography and generative training with quantum Boltzmann machines. Phys. Rev. A 96, 062327 (2017).

(2) Brandao, F. G. S. L. & Svore, K. M. Quantum Speed-Ups for Solving Semidefinite Programs. in 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS) 415–426 (IEEE, 2017).

(3) Somma, R. D., Boixo, S., Barnum, H. & Knill, E. Quantum Simulations of Classical Annealing Processes. Phys. Rev. Lett. 101, 130504 (2008).

(4) Wang, Y., Li, G. & Wang, X. Variational quantum Gibbs state preparation with a truncated Taylor series. arXiv:2005.08797 (2020).