量子物理计算方法选讲实验报告

李昊恩　2021010312

Density Matrix Renormalization Group(DMRG), Task 1

目录

1　引言

密度矩阵重整化群（DMRG）算法是求解较大规模量子多体物理问题的一种数值方法。在DMRG 算法的传统版本（不基于矩阵乘积态（MPS））的架构中，我们考虑一个具有 N 个自旋的量子态 $|\psi\rangle$，它在全组态基底 $\{|i_1 i_2 \cdots i_N\rangle\}$ 下可以展开成一个 N 阶张量 $C_{i_1 \cdots i_N}$：

$$|\psi\rangle = \sum_{i_1 i_2 \cdots i_N} C_{i_1 i_2 \cdots i_N} |i_1 i_2 \cdots i_N\rangle , \tag{1}$$

该张量 α 可以用图1表示，其中每条边表示一个指标：我们考虑所谓的 “二分量子系统”，具体来

图 1: N 阶张量

说，将指标集分为两个部分，前 p 个指标合并成一个新的指标 i，后 $N-p$ 个指标合并成一个新的指标 j，这相当于对张量进行一个 reshape 操作。如图2所示。

此时，我们事实上得到了张量 C 的一个 “矩阵化”：

$$|\psi\rangle = \sum_{i_1 \cdots i_N} C_{i_1 \cdots i_N} |i_1 \cdots i_N\rangle = \sum_{(i_1 \cdots i_p),(i_{p+1} \cdots i_N)} C_{(i_1 \cdots i_p),(i_{p+1} \cdots i_N)} |i_1 \cdots i_p\rangle_A \otimes |i_{p+1} \cdots i_N\rangle_B \tag{2}$$

$$= \sum_{i,j} C_{i,j} |i\rangle_A |j\rangle_B , \tag{3}$$

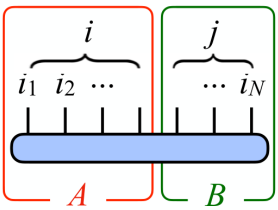

图 2: 二分量子系统

显然，$|i\rangle_A$ 和 $|j\rangle_B$ 构成了一族标准正交基。我们对矩阵 $C_{i,j}$ 做奇异值分解可得：

$$C_{i,j} = \sum_{i,\alpha,j} u_{i,\alpha} s_\alpha v^*_{j,\alpha}.$$

其中，$U = [u_{i,\alpha}]$，$V = [v_{\alpha,j}]$，都是等距矩阵，也就是满足：

$$U^\dagger U = \mathrm{Id}, \quad VV^\dagger = \mathrm{Id}. \tag{4}$$

由此可以定义一族新的基底：

$$|u_\alpha\rangle_A = \sum_{i,\alpha} u_{i,\alpha} |i\rangle_A, \quad |v_\alpha\rangle_B = \sum_{j,\alpha} v^*_{j,\alpha} |j\rangle_B. \tag{5}$$

在这组新的基底下，$|\psi\rangle$ 可以写成：

$$|\psi\rangle = \sum_{i,j} \left[\sum_\alpha s_\alpha u_{i,\alpha} v^*_{j,\alpha}\right] |i\rangle_A |j\rangle_B = \sum_\alpha s_\alpha |u_\alpha\rangle_A |v_\alpha\rangle_B, \tag{6}$$

该过程称为对二分量子系统进行 Schmidt 分解。Schmidt 分解以及等距矩阵 U 和 V 的左、右正交条件可以用图3和图4表示。

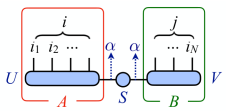

图 3: Schmidt 分解

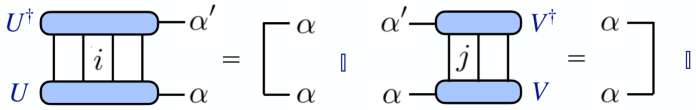

图 4: 左、右正交条件

有了量子系统的 Schmidt 分解，我们可以讨论体系的二分纠缠熵（bipartite entanglement entropy），其定义是：

$$S = -\operatorname{Tr}[\rho_A \ln \rho_A] = -\operatorname{Tr}[\rho_B \ln \rho_B]. \tag{7}$$

其中，ρ_A 和 ρ_B 分别是子系统 A 和 B 的约化密度矩阵：

$$\rho_A = \operatorname{Tr}_B \rho = \sum_\beta \langle v_\beta|_B \left[\sum_{\alpha,\alpha'} s_\alpha s_{\alpha'} |u_\alpha\rangle_A \langle u_{\alpha'}|_A \otimes |v_\alpha\rangle_B \langle v_{\alpha'}|_B\right] |v_\beta\rangle_B = \sum_\beta s_\beta^2 |u_\beta\rangle_A \langle u_\beta|_A. \tag{8}$$

同样，有 $\rho_B = \sum_\alpha s_\alpha^2 |v_\alpha\rangle_B \langle v_\alpha|_B$.

由此可得，二分纠缠熵实际上与矩阵 C_{ij} 的谱有关，根据 $\ln \rho_A = \sum_\alpha \ln s_\alpha \, |u_\alpha\rangle\langle u_\alpha|$ 可知：

$$S = -\operatorname{Tr}\left(\sum_{\alpha,\alpha'} s_\alpha^2 \ln s_{\alpha'}^2 \, |u_\alpha\rangle\langle u_\alpha| \, |u_{\alpha'}\rangle\langle u_{\alpha'}|\right) = -\operatorname{Tr}\left(\sum_{\alpha,\alpha'} \delta_{\alpha\alpha'} s_\alpha^2 \ln s_{\alpha'}^2 \, |u_\alpha\rangle\langle u_{\alpha'}|\right) \tag{9}$$

$$= -\sum_\alpha s_\alpha^2 \ln s_\alpha^2 \operatorname{Tr}(|u_\alpha\rangle\langle u_\alpha|) = -\sum_\alpha s_\alpha^2 \ln s_\alpha^2. \tag{10}$$

在量子物理问题中，我们考虑的可观测量（例如哈密顿量）往往具有局部性，也就是可以写成一些只涉及到少部分自旋位点的算符的线性组合。一般地，我们考虑一个只涉及 A 部分的可观测量算符 O_A，其期望值为：

$$\langle O_A \rangle = \langle\psi|O_A|\psi\rangle = \operatorname{Tr}(O_A \, |\psi\rangle\langle\psi|) = \operatorname{Tr}(O_A \rho) = \operatorname{Tr}(O_A \rho_A). \tag{11}$$

如图5所示.

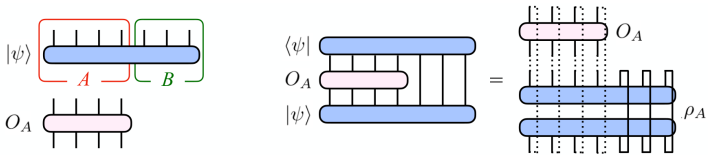

图 5: 期望值计算

为了缩减问题的规模，我们可以对约化密度矩阵 ρ_A 进行截断，具体来说：

$$\rho_A = \sum_\alpha s_\alpha^2 \, |u_\alpha\rangle\langle u_\alpha| = CC^\dagger = USVV^\dagger U^\dagger = US^2U^\dagger. \tag{12}$$

如果只取出 S^2 中前 k 个奇异值，就可以将体系的规模从 n 缩小到 k。即 $\rho_A \approx \widetilde{U}\widetilde{S}^2\widetilde{U}^\dagger$. 我们用新的这组 $\widetilde{U}$（不妨也记为 U）对 O_A 进行截断，这就等价于缩减了 Hilbert 空间的尺寸，如图6所示。

$$O'_A = U^\dagger O_A U, \tag{13}$$

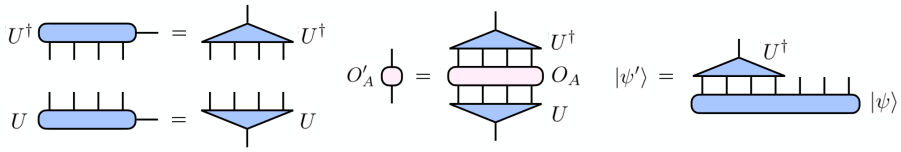

图 6: 可观测量 O_A 的截断

在实际进行（传统）DMRG 算法中，我们往往每次移动一个位点来改变二分量子系统，然后对约化密度矩阵 ρ 对角化、截断，然后用截断后的 U 来对算符、波函数做相应截断，得到新的 “super block Hamiltonian”（尺寸远小于原来空间），对角化求出基态能量。随后对新的体系重复该过程，直到能量达到收敛。

在本次任务中，我们使用`https://github.com/simple-dmrg/simple-dmrg`中提供的样例来求解 $N = 30$ 的横场 Ising 模型。具体代码实现在下一个部分讨论。

2 代码实现

我们使用 Python 的 namedtuple 数据类型来实现 block 的轻量级存储，用于索引其长度、截断基组的大小（也就是第一部分提及的 U 的尺寸）以及所涉及的算子。具体如下：

```
# We will use python's "namedtuple" to represent the Block and EnlargedBlock
# objects
from collections import namedtuple

Block = namedtuple("Block", ["length", "basis_size", "operator_dict"])
EnlargedBlock = namedtuple("EnlargedBlock", ["length", "basis_size", "operator_dict"])

def is_valid_block(block):
    for op in block.operator_dict.values():
        if op.shape[0] != block.basis_size or op.shape[1] != block.basis_size:
            return False
    return True

# This function should test the same exact things, so there is no need to
# repeat its definition.
is_valid_enlarged_block = is_valid_block
```

事先定义横场 Ising 模型的尺寸、参数以及涉及到的算子（H1 表示单位点的 Pauli-X 算子，Pz1 表示单体的 Pauli-Z 算子）。定义函数 H2 用于构造二体算子。

```
# Model-specific code for the transverse field Ising model
model_d = 2  # single-site basis size
Pz1 = (-1.0)*np.array([[1, 0], [0, -1]], dtype= 'd')  #single-site Pauli z
N = 30
g = 0.7

# list to save the entanglement entropy values
EElist = []

#list to save corresponding length of system
syslist = []

H1 = (-1.0)*g*np.array([[0, 1], [1, 0]], dtype= 'd')  #single-site portion of H is single-site Pauli x

def H2(Pz1,Pz2):  # two-site part of H
    """Given pauli-z matrices on two sites in different Hilbert spaces
    (e.g. two blocks), returns a Kronecker product representing the
```

```
    corresponding two-site term in the Hamiltonian that joins the two sites.
    """
    return (kron(Pz1,Pz2))

# conn refers to the connection operator, that is, the operator on the edge of
# the block, on the interior of the chain.
initial_block = Block(length=1, basis_size=model_d, operator_dict={
    "H": H1,
    "conn_Pz": Pz1,
})
```

为了构造增加一个位点的 enlarge block，我们定义函数如下：

```
def enlarge_block(block):
"""This function enlarges the provided Block by a single site, returning an
EnlargedBlock.
"""
mblock = block.basis_size
o = block.operator_dict

enlarged_operator_dict = {
    "H": kron(o["H"], identity(model_d)) + kron(identity(mblock), H1) + H2(o["
    conn_Pz"], Pz1),
    "conn_Pz": kron(identity(mblock), Pz1),
}

return EnlargedBlock(length=(block.length + 1),
                        basis_size=(block.basis_size * model_d),
                        operator_dict=enlarged_operator_dict)
```

这里 enlarged block 的哈密顿量在上一步所得哈密顿量的基础上相应地增加 1、2 体算子即可，例如，由 $H^{(3)}$ 构造 $H^{(4)}$ 的过程实际上为：

$$H^{(4)} = H^{(3)} \otimes \mathrm{Id}_{\mathbb{C}^2} + \mathrm{Id}_{\mathbb{C}^2}^{\otimes 3} \otimes \mathbf{S} + \mathrm{Id}_{\mathbb{C}^2}^{\otimes 2} \otimes \mathbf{D}. \tag{14}$$

其中 **S** 表示单体算符、**D** 表示二体算符。（此处选择 $H^{(1)} = -gX$，$H^{(2)} = -Z_1Z_2 - g(X_1 + X_2)$）

随后，定义函数 `simple_dmrg_step`，完成一次 DMRG 过程的 superblock 哈密顿量的对角化以及密度矩阵的对角化、基底的截断，输出此步之后所得到的新的 block，以及此时计算出的基态能量、二分纠缠熵。其他代码在原有程序中已经写好，计算二分纠缠熵的代码采用公式 $S = -\sum_\alpha s_\alpha^2 \ln s_\alpha^2$：

```
Entanglement_entropy = -np.sum(evals*(np.log(np.abs(evals))))
```

对于有限系统的 DMRG，我们是先增加位点，当位点达到实际体系的真实自旋数目时，改为从左到右移动二分量子系统的分划处，先从左移动到右，再从右移动到左，这称为一次“扫描”，扫描结束后输出基态能量。在本次算法中由于要得到二分纠缠熵与 L 的关系 $S(L)$（L 为左边的 A 子系统的长度，如图7所示），我们在得到基态能量之后再进行一次扫描，输出每次扫描的二分纠缠熵，以及左边子系统的长度，最后，我们将用于记录 $S(L)$ 做适当的切片操作，画出示意图。

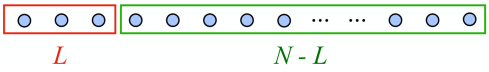

图 7: 二分量子系统示意图

有限系统 DMRG 算法的其他程序已经写好，额外进行的步骤代码实现如下：

```
for m in m_sweep_list:
    while True:
        # Load the appropriate environment block from "disk"
        env_block = block_disk[env_label, L - sys_block.length - 2]
        if env_block.length == 1:
            # We've come to the end of the chain, so we reverse course.
            sys_block, env_block = env_block, sys_block
            sys_label, env_label = env_label, sys_label

        # Perform a single DMRG step.
        sys_block, energy, Entanglement_entropy = single_dmrg_step(sys_block,
    env_block, m=m)
        if sys_label == "l":
            L_length = sys_block.length
        else:
            L_length = L - sys_block.length
        syslist.append(L_length)
        EElist.append(Entanglement_entropy)
        # Save the block from this step to disk.
        block_disk[sys_label, sys_block.length] = sys_block
        # Check whether we just completed a full sweep.
        if sys_label == "l" and 2 * sys_block.length   == L:
            break  # escape from the small "while True" loop
```

用于画图的代码如下：

```
#print entanglement entropy and plot S(L)

#make slices of EElist and syslist, so that syslist is from 2 to 28 and EElist has
    the corresponding entanglement entropies
```

```
EElist = EElist[:int((N-4)/2)]+ EElist[-int((N-2)/2):]
syslist = syslist[:int((N-4)/2)]+ syslist[-int((N-2)/2):]

#using argsort to find the order
order = np.argsort(syslist)
EElist_ordered = [EElist[order[i]] for i in range(len(order))]
syslist_ordered = [syslist[order[i]] for i in range(len(order))]

print("Entanglement Entropy: ",EElist_ordered)

#plot the figure
plt.plot(syslist_ordered,EElist_ordered)
plt.xlabel('L')
plt.ylabel('Entanglement Entropy')
plt.show()
```

3 结果

我们固定 $m = 24$，对 $g = 0.8, 1.0, 1.8$ 分别计算，结果如下：

- $g = 0.8$，

```
Energy: -34.38998942912344
Entanglement Entropy:  [0.5229478009287744, 0.5944593757506269,
    0.6386834417903945, 0.6668532650096703, 0.6850052821063777,
    0.6967563595432016, 0.7043739812613273, 0.709308354039355,
    0.7124947244132855, 0.7145370896312505, 0.7158234642024739,
    0.7165989820640059, 0.717011869453689, 0.7171412272195591,
    0.7170118694536848, 0.7165989820640081, 0.7158234642024718,
    0.7145370896312568, 0.7124947244132828, 0.7093083540393571,
    0.7043739812613338, 0.6967563595432188, 0.6850052821063772,
    0.6668532650096926, 0.6386834417904145, 0.594459375750635,
    0.5229478009287709]

```

- $g = 1.0$;

```
Energy: -37.838098239709346
Entanglement Entropy:  [0.33627773573995573, 0.3742559146539784,
    0.399823583815671, 0.4186605615321329, 0.43322026231327704,
    0.44477307349844786, 0.454055690122461, 0.46152878649094725,
    0.4674953936964834, 0.4721611995445935, 0.4756676332490788,
    0.478110962213954, 0.4795535969710677, 0.48003068948019906,
```

```
    0.47955359697100064, 0.47811096221389116, 0.47566763324896777,
    0.4721611995444911, 0.4674953936963907, 0.461528786490947,
    0.4540556901224495, 0.444773073498451, 0.4332202623132885,
    0.4186605615321503, 0.3998235838156951, 0.3742559146539615,
    0.33627773573994124]
```

- $g = 1.8$

```
Energy: -58.105951341881195
Entanglement Entropy:  [0.10627435723888942, 0.1075564878460264,
    0.10781454575608646, 0.10787195615623255, 0.10788553201630016,
    0.1078888779484508, 0.10788972773904774, 0.10788994855510368,
    0.10789000697541476, 0.10789002265780508, 0.10789002691799614,
    0.10789002808531395, 0.10789002840185864, 0.10789002846678833,
    0.10789002840185848, 0.10789002808531578, 0.10789002691799679,
    0.10789002265780377, 0.10789000697541519, 0.10788994855510438,
    0.1078897277390503, 0.10788887794844998, 0.10788553201630428,
    0.1078719561562348, 0.10781454575608346, 0.10755648784602656,
    0.10627435723889206]
```

图示如下：

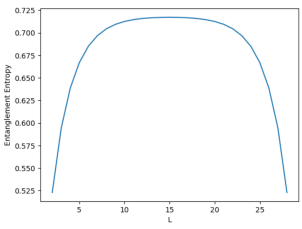

图 8: $m = 24, g = 0.8$

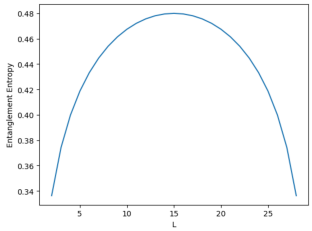

图 9: $m = 24, g = 1.0$

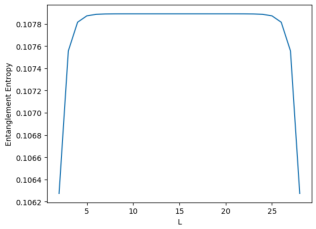

图 10: $m = 24, g = 1.8$

可见，g 越大，其基态能量越低，而且，纠缠熵的图形就越 “平”。

再固定 $g = 1.0$，对 $m = 8, 16, 24$ 进行计算，结果如下：

- $m = 8$;

```
Energy: -37.83809511868236
Entanglement Entropy:  [0.33627428595190134, 0.3742492953354369,
    0.39981285320758997, 0.4186448639150883, 0.4331989165874062,
    0.44474560720749423, 0.454021874666831, 0.46148866491309287,
    0.46744929579041156, 0.47210973999259004, 0.4756116911728909,
    0.4780516430117399, 0.4794921790804463, 0.4799685598070191,
    0.4794920104880291, 0.47805153191396255, 0.4756116190068094,
```

```
    0.47210969420109894, 0.46744926750177707, 0.46148864776169973,
    0.4540218642059383, 0.44474560056295576, 0.4331989121262802,
    0.4186448608478915, 0.3998128511553394, 0.37424929408177615,
    0.3362742853038707]

```

- $m = 16$;

```
Energy: -37.83809823961611
Entanglement Entropy:  [0.3362777355455277, 0.3742559142820449,
    0.3998235832071532, 0.41866056062531015, 0.43322026104508216,
    0.44477307181210907, 0.45405568797568463, 0.4615287838768125,
    0.46749539063600193, 0.4721611960690695, 0.4756676294250908,
    0.47811095812722304, 0.47955359272073034, 0.480030685173592,
    0.4795535929785567, 0.478110958314097, 0.4756676295454255,
    0.4721611961327464, 0.4674953906555043, 0.4615287838766517,
    0.4540556879632444, 0.4447730718050791, 0.4332202610396072,
    0.41866056062202434, 0.3998235832045295, 0.37425591427992416,
    0.33627773554426627]

```

- $m = 24$;

```
Energy: -37.838098239709346
Entanglement Entropy:  [0.33627773573995573, 0.3742559146539784,
    0.399823583815671, 0.4186605615321329, 0.43322026231327704,
    0.44477307349844786, 0.454055690122461, 0.46152878649094725,
    0.4674953936964834, 0.4721611995445935, 0.4756676332490788,
    0.478110962213954, 0.4795535969710677, 0.48003068948019906,
    0.47955359697100064, 0.47811096221389116, 0.47566763324896777,
    0.4721611995444911, 0.4674953936963907, 0.461528786490947,
    0.4540556901224495, 0.444773073498451, 0.4332202623132885,
    0.4186605615321503, 0.3998235838156951, 0.3742559146539615,
    0.33627773573994124]

```

图示如图11、12和13所示：

可见，$m = 8, 16, 24$ 时计算出的基态能量基本一致，只是在小数点后 5 位出现误差，而且随着 m 增大，算出的能量越来越低，这与数值方法的变分性也是符合的。从纠缠熵 $S(L)$ 的图形来看，三者形状几乎没有区别。

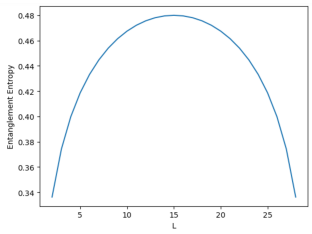

图 11: $m = 8, g = 1.0$

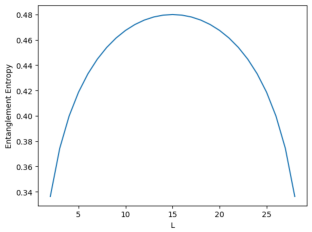

图 12: $m = 16, g = 1.0$

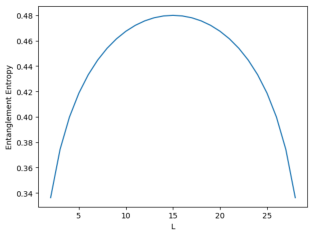

图 13: $m = 24, g = 1.0$