February 13-17,2023: Quantum Computing, Quantum Machine Learning and Quantum Information Theories

Morten Hjorth-Jensen[1,2]

[1]Department of Physics, University of Oslo
[2]Department of Physics and Astronomy and Facility for Rare Isotope Beams, Michigan State University, USA

Feb 18, 2023

Entanglement and density matrices

1. Last week we discussed the spectral decomposition of an operator, Density matrices and Measurements.
2. This week we start by reviewing density matrices and measurements from previous lecture, see `https://github.com/CompPhysics/QuantumComputingMachineLearning/blob/gh-pages/doc/pub/week3/ipynb/week3.ipynb`
3. Thereafter our plans are to discuss the Schmidt decomposition and entanglement

- **Reading recommendation**: Scherer, Mathematics of Quantum Computations, chapter 4
- See also whiteboard notes
- Video of lecture

In order to study entanglement and why it is so important for quantum computing, we need to introduce some basic measures and useful quantities. For these endeavors, we will use our two-qubit system from the second lecture in order to introduce, through examples, density matrices and entropy. These two quantities, together with technicalities like the Schmidt decomposition define important quantities in analyzing quantum computing examples.

The Schmidt decomposition is again a linear decompostions which allows us to express a vector in terms of tensor product of two inner product spaces. In quantum information theory and quantum computing it is widely used as away to define and describe entanglement.

First entanglement encounter

Two-qubit system. This system can be thought of as composed of two subsystems A and B. Each subsystem has computational basis states

$$|0\rangle_{\mathrm{A,B}} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T \qquad |1\rangle_{\mathrm{A,B}} = \begin{bmatrix} 0 & 1 \end{bmatrix}^T .$$

The subsystems could represent single particles or composite many-particle systems of a given symmetry. This leads to the many-body computational basis states

$$|00\rangle = |0\rangle_{\mathrm{A}} \otimes |0\rangle_{\mathrm{B}} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^T ,$$

and

$$|10\rangle = |1\rangle_{\mathrm{A}} \otimes |0\rangle_{\mathrm{B}} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T ,$$

and

$$|01\rangle = |0\rangle_{\mathrm{A}} \otimes |1\rangle_{\mathrm{B}} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T ,$$

and finally

$$|11\rangle = |1\rangle_{\mathrm{A}} \otimes |1\rangle_{\mathrm{B}} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T .$$

Bell states. The above computational basis states, which define an ONB, can in turn be used to define another ONB. As an example, consider the so-called Bell states

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} \left[|00\rangle + |11\rangle \right] ,$$

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}} \left[|00\rangle - |11\rangle \right] ,$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}} \left[|10\rangle + |01\rangle \right] ,$$

and

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}} \left[|10\rangle - |01\rangle \right] .$$

It is easy to convince oneself that these states also form an orthonormal basis.

Measuring one of the qubits of one of the above Bell states, automatically determines, as we will see below, the state of the second quabit. To convince ourselves about this, let us assume we perform a measurement on the qubit in system A by introducing the projections with outcomes 0 or 1 as

$$\boldsymbol{P}_0 = |0\rangle\langle 0|_A \otimes \boldsymbol{I}_B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} ,$$

for the projection of the $|0\rangle$ state in system A and similarly

$$\boldsymbol{P}_1 = |1\rangle\langle 1|_A \otimes \boldsymbol{I}_B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

for the projection of the $|1\rangle$ state in system A.

We can then calculate the probability for the various outcomes by computing for example the probability for measuring qubit 0

$$\langle \Phi^+ | \boldsymbol{P}_0 | \Phi^+ \rangle = \frac{1}{2} \begin{bmatrix} [|00\rangle + |11\rangle] \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} [|00\rangle + |11\rangle] = \frac{1}{2}.$$

Similarly, we obtain

$$\langle \Phi^+ | \boldsymbol{P}_1 | \Phi^+ \rangle = \frac{1}{2} \begin{bmatrix} [|00\rangle + |11\rangle] \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [|00\rangle + |11\rangle] = \frac{1}{2}.$$

Density matrix and simple Hamiltonian

These computational basis states define also the eigenstates of the non-interacting Hamiltonian

$$H_0|00\rangle = \epsilon_{00}|00\rangle,$$
$$H_0|10\rangle = \epsilon_{10}|10\rangle,$$
$$H_0|01\rangle = \epsilon_{01}|01\rangle,$$

and

$$H_0|11\rangle = \epsilon_{11}|11\rangle.$$

The interacting part of the Hamiltonian H_{I} is given by the tensor product of two σ_x and σ_z matrices, respectively, that is

$$H_{\mathrm{I}} = H_x \sigma_x \otimes \sigma_x + H_z \sigma_z \otimes \sigma_z,$$

where H_x and H_z are interaction strength parameters. Our final Hamiltonian matrix is given by

$$\boldsymbol{H} = \begin{bmatrix} \epsilon_{00} + H_z & 0 & 0 & H_x \\ 0 & \epsilon_{10} - H_z & H_x & 0 \\ 0 & H_x & \epsilon_{01} - H_z & 0 \\ H_x & 0 & 0 & \epsilon_{11} + H_z \end{bmatrix}.$$

The four eigenstates of the above Hamiltonian matrix can in turn be used to define density matrices. As an example, the density matrix of the first eigenstate (lowest energy E_0) Ψ_0 is

$$\rho_0 = (\alpha_{00}|00\rangle\langle 00| + \alpha_{10}|10\rangle\langle 10| + \alpha_{01}|01\rangle\langle 01| + \alpha_{11}|11\rangle\langle 11|)\,,$$

where the coefficients α_{ij} are the eigenvector coefficients resulting from the solution of the above eigenvalue problem.

We can then in turn define the density matrix for the subsets A or B as

$$\rho_A = \mathrm{Tr}_B(\rho_0) = \langle 0|\rho_0|0\rangle_B + \langle 1|\rho_0|1\rangle_B,$$

or

$$\rho_B = \mathrm{Tr}_A(\rho_0) = \langle 0|\rho_0|0\rangle_A + \langle 1|\rho_0|1\rangle_A.$$

The density matrices for these subsets can be used to compute the so-called von Neumann entropy, which is one of the possible measures of entanglement. A pure state has entropy equal zero while entangled state have an entropy larger than zero. The von-Neumann entropy is defined as

$$S(A,B) = -\mathrm{Tr}\left(\rho_{A,B}\log_2(\rho_{A,B})\right).$$

The example here shows the above von Neumann entropy based on the density matrix for the lowest many-body state. We see clearly a jump in the entropy around the point where we have a level crossing. At interaction strenght $\lambda = 0$ we have many-body states purely defined by their computational basis states. As we switch on the interaction strength, we obtain an increased degree of mixing and the entropy increases till we reach the level crossing point where we see an additional and sudden increase in entropy. Similar behaviors are observed for the other states. The most important result from this example is that entanglement is driven by the Hamiltonian itself and the strength of the interaction matrix elements and the non-interacting energies.

```
%matplotlib inline
from  matplotlib import pyplot as plt
import numpy as np
from scipy.linalg import logm, expm
def log2M(a): # base 2 matrix logarithm
    return logm(a)/np.log(2.0)

dim = 4
Hamiltonian = np.zeros((dim,dim))
#number of lambda values
n = 40
lmbd = np.linspace(0.0,1.0,n)
Hx = 2.0
Hz = 3.0
# Non-diagonal part as sigma_x tensor product with sigma_x
sx = np.matrix([[0,1],[1,0]])
sx2 = Hx*np.kron(sx, sx)
# Diagonal part as sigma_z tensor product with sigma_z
sz = np.matrix([[1,0],[0,-1]])
```

```
sz2 = Hz*np.kron(sz, sz)
noninteracting = [0.0, 2.5, 6.5, 7.0]
D = np.diag(noninteracting)
Eigenvalue = np.zeros((dim,n))
Entropy = np.zeros(n)

for i in range(n):
    Hamiltonian = lmbd[i]*(sx2+sz2)+D
    # diagonalize and obtain eigenvalues, not necessarily sorted
    EigValues, EigVectors = np.linalg.eig(Hamiltonian)
    # sort eigenvectors and eigenvalues
    permute = EigValues.argsort()
    EigValues = EigValues[permute]
    EigVectors = EigVectors[:,permute]
    # Compute density matrix for selected system state, here ground state
    DensityMatrix = np.zeros((dim,dim))
    DensityMatrix = np.outer(EigVectors[:,0],EigVectors[:,0])
    # Project doum on substates and find density matrix for subsystem
    d = np.matrix([[1,0],[0,1]])
    v1 = [1.0,0.0]
    proj1 = np.kron(v1,d)
    x1 = proj1 @ DensityMatrix @ proj1.T
    v2 = [0.0,1.0]
    proj2 = np.kron(v2,d)
    x2 = proj2 @ DensityMatrix @ proj2.T
    # Total density matrix for subsystem
    total = x1+x2
    # von Neumann Entropy for subsystem
    Entropy[i] = -np.matrix.trace(total @ log2M(total))
    # Plotting eigenvalues and entropy as functions of interaction strengths
    Eigenvalue[0,i] = EigValues[0]
    Eigenvalue[1,i] = EigValues[1]
    Eigenvalue[2,i] = EigValues[2]
    Eigenvalue[3,i] = EigValues[3]
plt.plot(lmbd, Eigenvalue[0,:] ,'b-',lmbd, Eigenvalue[1,:],'g-',)
plt.plot(lmbd, Eigenvalue[2,:] ,'r-',lmbd, Eigenvalue[3,:],'y-',)
plt.xlabel('$\lambda$')
plt.ylabel('Eigenvalues')
plt.show()
plt.plot(lmbd, Entropy)
plt.xlabel('$\lambda$')
plt.ylabel('Entropy')
plt.show
```

Lecture next week.

1. More on entanglement and entropies
2. Einstein-Podolsky-Rosen paradox
3. Bell's inequalities
4. Circuits for elementary operations