Week-7: Optimization-II (b)

Contents

1 Adiabatic Quantum Computing (AQC) **2**
1.1 Adiabatic Theorem & Proof 2
1.2 AQC Scope 3

2 Quantum Approximate Optimization Algorithm (QAOA) **4**
2.1 Generalized Circuit for QAOA 5

3 Ensemble Methods **6**
3.1 Basics Revision 6
3.2 QBoost 6

4 Exercises **7**
4.1 Theory Exercises 7
4.2 Qiskit Exercises 7

1 Adiabatic Quantum Computing (AQC)

In an adiabatic process, the external conditions change very slowly such that if the system starts in its ground state, it remains in its ground state. We start with some initial Hamiltonian $\hat{H}_{init}$ and reach our target Hamiltonian $\hat{H}_{targ}$ adiabatically using an evolution parameter s=t/T where T is the total time of evolution of the system under study. Thus, the time-dependent Hamiltonian in the AQC framework becomes

$$\hat{H} = (1-s)\hat{H}_{init} + s\hat{H}_{target} \tag{1}$$

This means at s=0, we start with the initial Hamiltonian and at s=1 (completion phase), we arrive at the target Hamiltonian. This time-dependent Hamiltonian has a convex form. How does one select the value of T? An interesting feature about T is

$$T \propto \frac{1}{min(g(t))^2} \tag{2}$$

where g(t) is the energy difference between the ground state and the first excited state. The system must be evolved very slowly, otherwise the eigenstate corresponding to ground state can cross the eigenstate corresponding to first excited state which defeats the entire purpose. We often choose the intial Hamiltonian to be $\sum_i X_i$ where X is the pauli-X gate as it creates an equal superposition. AQC is based on the Adiabatic Theorem. We now prove this theorem for completeness.

1.1 Adiabatic Theorem & Proof

The time-independent Schrodinger equation is given by

$$\hat{H}\psi_n(x) = E_n\psi_n(x) \tag{3}$$

such that the complete solution $\Psi(x,t)$ can be written as

$$\Psi(x,t) = \sum_n a_n\Psi_n(x,t) = \sum_n a_n\psi_n(x)e^{-\iota E_n t/\hbar} \tag{4}$$

where the n^{th} energy eigenstate is given as $\Psi_n(x,t) = \psi_n(x)e^{-\iota E_n t/\hbar}$. This has a huge implication in the sense that a particle which starts in the n^{th} energy eigenstate remains in that state and picks up a time-dependent phase factor ($e^{-\iota E_n t/\hbar}$). When we evolve the system adiabatically, the Hamiltonian becomes time-dependent and so does the energy eigenvalues and eigenvectors. The time-independent Schrodinger equation changes in the following fashion

$$\hat{H}(t)\psi_n(x,t) = E_n(t)\psi_n(x,t) \tag{5}$$

Also, the instantaneous eigenstates are orthogonal such that they satisfy the ortho-normality condition:

$$\langle\psi_n(t)|\psi_m(t)\rangle = \delta_{nm} \tag{6}$$

The general solution is modified in the adiabatic conditions and it evolves in this fashion

$$\Psi(t) = \sum_n a_n(t)\psi_n(t)e^{\iota\theta_n(t)} \tag{7}$$

where $\theta_n(t) = -\frac{1}{\hbar}\int\limits_0^t E_n(t')dt'$ where this phase is known as the dynamic phase factor. Substituting this in the time-dependent Schrodinger equation and after simplification, we obtain

$$\sum_n \dot{a}_n\psi_n e^{\iota\theta_n} = -\sum_n a_n\dot{\psi}_n e^{\iota\theta_n} \tag{8}$$

Taking the inner product with $\langle\psi_m|$, we obtain the differential equation for the constant c_m which is given by

$$\dot{a}_m(t) = -a_m\langle\psi_m|\dot{\psi}_m\rangle - \sum_{n\neq m} a_n \frac{\langle\psi_m|\dot{\hat{H}}|\psi_n\rangle}{E_n - E_m} e^{\iota(\theta_n-\theta_m)} \tag{9}$$

where the dot represents the time derivative. The second term is oscillatory in nature and if the Hamiltonian operator is slowly changing with time, its derivative can be ignored. Under adiabatic conditions, we then obtain

$$a_m(t) \approx a_m(0)\exp\left[-\int\limits_0^t \langle\psi_m(t')|\dot{\psi}_m(t')\rangle dt'\right] = a_m(0)e^{\iota\gamma_m(t)} \tag{10}$$

where $\gamma_m(t) = \iota\int_0^t \langle\psi_m(t')|\dot{\psi}_m(t')\rangle dt'$ is called the geometric phase. The final form of the n^{th} eigenstate under adiabatic evolution becomes

$$\Psi_n(t) = \psi_n(t)e^{\iota\theta_n(t)}e^{\iota\gamma_n(t)} \tag{11}$$

Thus, we prove that an n^{th} eigenstate remains in that state provided the evolution of the system is carried out adiabatically. It only picks up some phase factors.

1.2 AQC Scope

If a quantum computer can implement this Hamiltonian, ensure that the gap is maintained, then this model is equivalent to gate model of computation. However, it is an open question whether AQC provides an exponential speedup over the classical processors. The overall time of execution is quite high which will give incorrect results due to decoherence.

2 Quantum Approximate Optimization Algorithm (QAOA)

As we have already studied Quantum Annealing, we now make use of it in QAOA. QAOA is also a hybrid algorithm which makes use of unitaries with 2p angles $(\vec{\gamma}, \vec{\beta})$ where p is the precision. It is a variational algorithm in which we perform Lie-Suzuki-Trotterization on the adiabatic Hamiltonian as introduced in the previous section. The expectation value is then calculated on the state prepared using those unitaries and optimized using a classical optimizer which supplies the updated angle list. This process is repeated till converge to obtain the optimal cost function. The flowchart is given in Fig. 1.

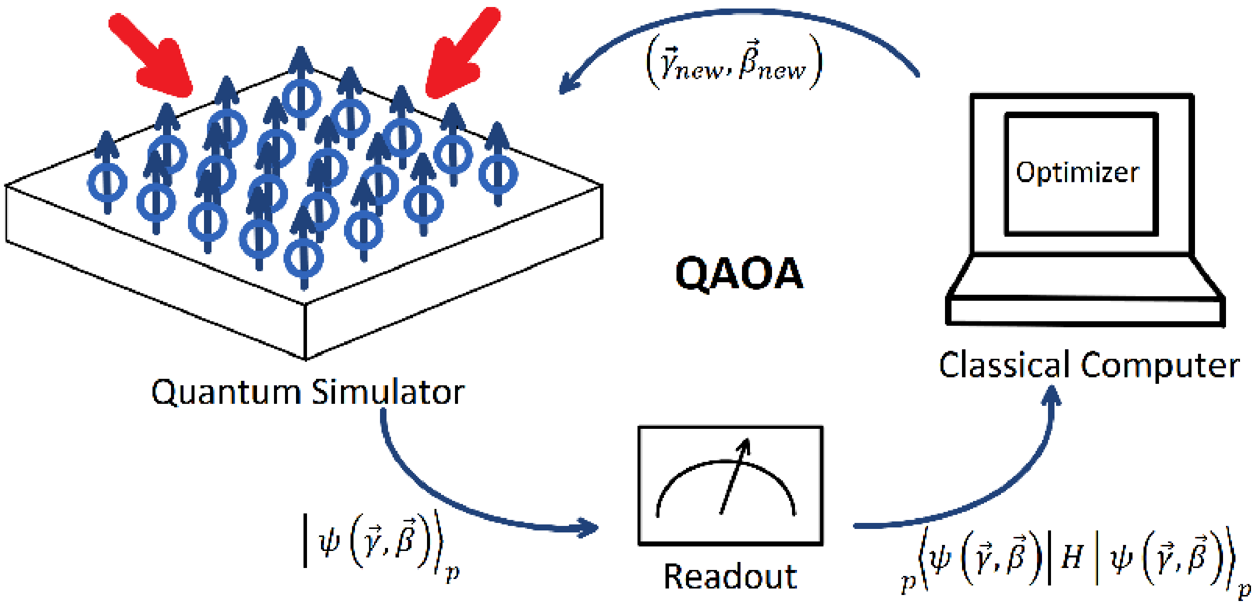

$(\vec{\gamma}_{new}, \vec{\beta}_{new})$
Optimizer
QAOA
Classical Computer
Quantum Simulator
$|\psi(\vec{\gamma},\vec{\beta})\rangle_p$
Readout
$_p\langle\psi(\vec{\gamma},\vec{\beta})|H|\psi(\vec{\gamma},\vec{\beta})\rangle_p$

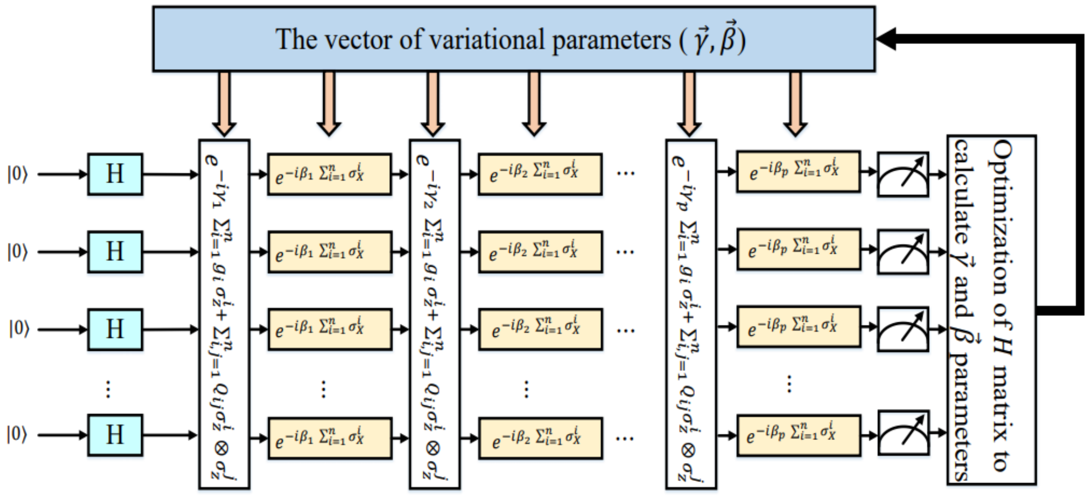
Figure 1: QAOA Flowchart [1]

We set a precision p which basically means how much you want to discretize the time-dependent Hamiltonian. As the qubit size increases, the precision has to be increased to obtain the correct answers. This is because the search space increases with the increase in qubits. We trotterize the unitary in the given way

$$U = U(\hat{H}_{init}, \beta_0)U(\hat{H}_{targ}, \gamma_0)U(\hat{H}_{init}, \beta_1)U(\hat{H}_{targ}, \gamma_1).....U(\hat{H}_{init}, \beta_p)U(\hat{H}_{targ}, \gamma_p) \quad (12)$$

where $U(H_{init}, \beta_i) = e^{-\iota \hat{H}_{init}\beta_i}$ and $U(H_{targ}, \gamma_i) = e^{-\iota \hat{H}_{targ}\gamma_i}$ This is how we approximate the adiabatic pathway. This algorithm was developed by Fargi, Goldtone, and Gutmann. It is a polynomial time algorithm which has the capacility of finding correct solutions to a given optimization alorithm.

2.1 Generalized Circuit for QAOA



The vector of variational parameters ($\vec{\gamma}, \vec{\beta}$)
$|0\rangle$
H
$e^{-i\gamma_1 \sum_{i=1}^{n} g_i \sigma_Z^i + \sum_{i,j=1}^{n} Q_{ij} \sigma_Z^i \otimes \sigma_Z^j}$
$e^{-i\beta_1 \sum_{i=1}^{n} \sigma_X^i}$
$e^{-i\gamma_2 \sum_{i=1}^{n} g_i \sigma_Z^i + \sum_{i,j=1}^{n} Q_{ij} \sigma_Z^i \otimes \sigma_Z^j}$
$e^{-i\beta_2 \sum_{i=1}^{n} \sigma_X^i}$
$e^{-i\gamma_p \sum_{i=1}^{n} g_i \sigma_Z^i + \sum_{i,j=1}^{n} Q_{ij} \sigma_Z^i \otimes \sigma_Z^j}$
$e^{-i\beta_p \sum_{i=1}^{n} \sigma_X^i}$
Optimization of H matrix to calculate $\vec{\gamma}$ and $\vec{\beta}$ parameters

Figure 2: General Circuit [2]

3 Ensemble Methods

3.1 Basics Revision

These methods are beneficial as they improve ML results by combining various models. This technique produces better predictive performance. These are meta methods that combine various ML models into one predictive model in order to improve the predictive power and decrease variance and bias. These methods are broadly classified into 2 groups.

Sequential	Base learners are sequentially generated. This is done to exploit the fundamental dependence between the base learners. Previous wrong results are given more weights.
Parallel	Base learns are parallely generated. This is done to exploit the independence between the base learners.

Table 1: Classification of Ensemble Methods

In Bragging or bootstrap aggregation, we reduce the variance by averaging multiple estimates. In Boosting, we convert weak learners to strong learners. In AdaBoost, we combine weak learners sequentially. Its function landscape is convex and we do not put penalties in case of added complexities. The procedure is as follows

- Choose a training data. Develop sets of base learners.
- Combine these base learners to form an ensemble

3.2 QBoost

In QBoost, we form a linear superposition of weak learners and then we optimize the corresponding weights during learning and minimize the errors in training and the number of weak learners. We do not restrict ourselves to convex objective functions in QBoost. Thus, we can add penalties. In QBoost, we solve the following

$$argmin_w \left(\frac{1}{N} \sum_{i=1}^{N} \left(\sum_{k=1}^{K} w_k h_k(x_i) - y_i \right)^2 + \lambda \|w\|_0 \right) \tag{13}$$

where $h_k(x_i)$ represents a prediction of weak leaner k. The l_0 norm is taken so as to attain sparsity. Since, the weights are binary in nature, this is automatically mapped to Ising model. After opening the terms in square, we finally get

$$argmin_w \left(\frac{1}{N} \sum_{k=1}^{K} \sum_{j=1}^{K} w_k w_j \left(\sum_{i=1}^{N} h_k(x_i) h_j(x_i) \right) - \frac{2}{N} \sum_{k=1}^{K} w_k \sum_{i=1}^{N} h_k(x_i) y_i + \lambda \|w\|_0 \right) \tag{14}$$

We can see correlations in the above function and we penalize for them and keep the regularization terms unchanged.

4 Exercises

4.1 Theory Exercises

1. **Adiabatically Assisted VQE (AAVQE)**: Devise an algorithm in which you combine the VQE and Adiabatic QC to form AAVQE. Does AAVQE perform better than VQE? If yes, then why?

2. Give a sample QAOA circuit for 4 qubits and p=1 for the implementation of a transverse Ising model.

3. List some differences between AdaBoost and QBoost

4.2 Qiskit Exercises

1. **3-D Ising Model**: Given a L*L*L lattice with boundary conditions that are periodic. You have to implement a 3-D Ising Model. The input parameters are length L of the lattice, number of iterations (n_{iter}) at a particular temperature T. Thermal energy K_BT is measured in units of J_{ising} where $J_{ising} = 1.0$. Take the value of K_B to be 1.Run the simulation (Metropolis Monte Carlo) for L=7,8,10 for a temperature range of 4.7 to 3.8 . Take ΔT=0.02. At each value of T, use 10^6 Monte-Carlo steps for equilibration. After equilibration which you have to do at each temperature, you must collect statistical data each MCS for 10^6 iterations for thermodynamic averaging. Answer the following questions.

   - What are the values of specific heat C_V at the peak position for L=10 and L=7?
   - What is the value of energy per spin for L=7 at T=3.8?
   - What are the values of the temperatures T_p at which C_V is maximum for L=7,8,10?

   Write the code in both Fortran and Python and compare the speeds. You will see significant difference. We have provided the solution in Fortran. You can map it to Python accordingly.

2. Create a random dataset of two classes of half moons. Train AdaBoost, SVM and QAOA. Compare their performances.

References

[1] Ho, W.W., & Hsieh, T.H. (2018). Efficient preparation of non-trivial quantum states using the Quantum Approximate Optimization Algorithm.

[2] Available from: arXiv:1911.00595 [quant-ph]