Quantum Information (ELEC-C9440) Lecture 10

Matti Raasakka

Spring 2023

Aalto University

Variational quantum algorithms

Noisy intermediate scale quantum (NISQ) era

Last time we learned that while quantum computing holds promise for various applications like factoring and quantum searching, the current quantum processors lack the quantum resources necessary to run the relevant algorithms. The problem was that the

- number of qubits
- quality of qubits

is insufficient for quantum error correction.

The question is then: what could be done using a noisy quantum computer instead of an error corrected one?

The requirement for these applications is that we have to work with

- a small number of qubits
- shallow quantum circuits .

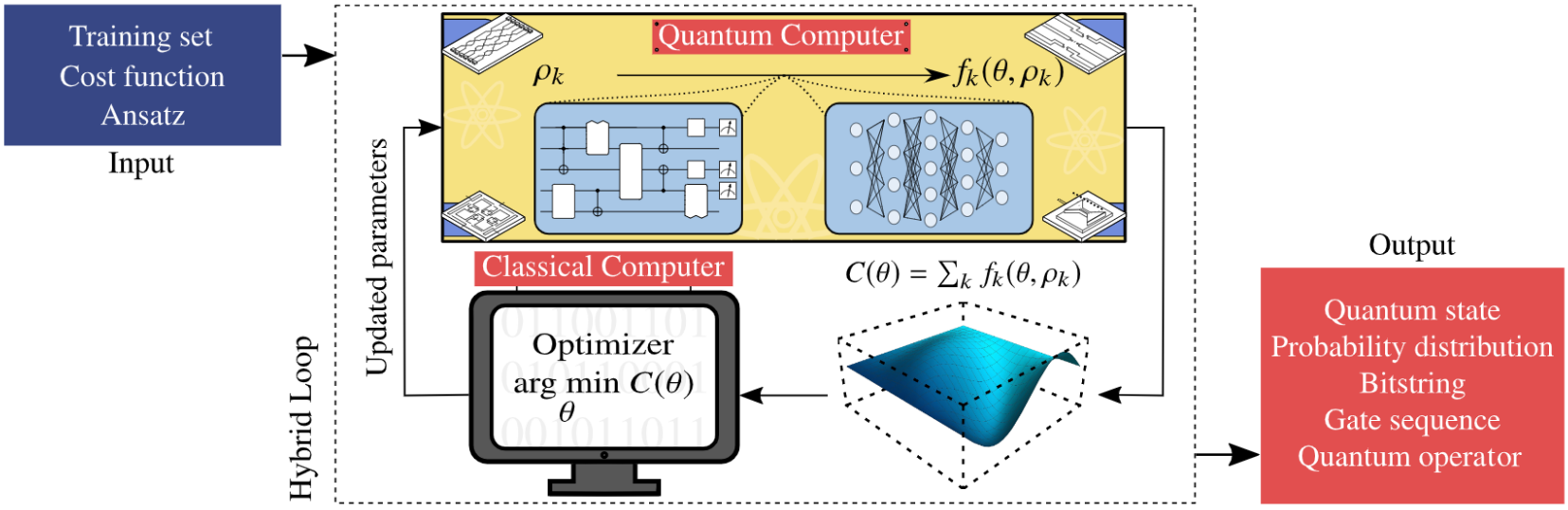
Variational quantum algorithms

Variational quantum algorithms (VQAs) are a popular strategy aimed for achieving quantum advantage on NISQ computers. VQAs attempt to satisfy the NISQ constraints by using an optimization/learning-based approach. One can think of VQAs as quantum analogues of classical machine learning methods, specifically neural networks. The rough idea behind VQAs can be summarized as:

- Define a "cost function"
- Define a parametrized quantum circuit
- Choose initial values for those parameters, randomly perhaps
    1. Use the quantum computer to measure the value of the cost function and/or its gradients w.r.t. the circuit parameters
    2. Use a classical optimizer to suggest new parameter values so that the cost decreases
    3. Return to step 1 if cost is not low enough
- When the optimization stops you have found a minimum of the cost function which (hopefully) signals that the original problem has been solved

This recipe becomes clearer when we see examples.

Variational quantum algorithms



Here's a schematic figure of a generic VQA. The point is that by iterating between the quantum and classical parts in the "hybrid loop", we can get away with using many shallow quantum circuits instead of using a small number of deep quantum circuits.

We will cover two examples: ground state finding and combinatorial optimization.

Variational quantum algorithms

VQAs provide a generic framework for solving a variety of problems. They often have the following common basic building blocks

- Cost function
- Parametrized quantum circuit
- Gradient computation
- Optimizers

Cost functions

The purpose of a cost function is to encode the problem to a form that can be solved with optimization. Let's denote the trainable parameters collectively by θ. If there are p parameters, then the cost function C is a map

$$C: \quad \mathbb{R}^p \quad \mapsto \quad \mathbb{R}\,, \tag{1}$$

that is, for each parameter combination it gives a real number that represents the quality or cost associated to those parameters. We can express the cost function as

$$C(\theta) = f\left(\mathrm{tr}\left(OU(\theta)\rho U(\theta)^\dagger\right)\right)\,. \tag{2}$$

where ρ is an initial state (often $|0\rangle\langle 0|$), $U(\theta)$ is a parametrized quantum circuit, O is an observable, and f is a real function. In other words, the cost function is some function of the expectation value of O in the quantum state produced by $U(\theta)$ starting in some initial state ρ.

Sometimes there might be multiple initial states or observables, especially in the case of quantum machine learning where training data would be present. However, the above form is sufficient for the purpose of this lecture.

Cost functions

The cost function should be

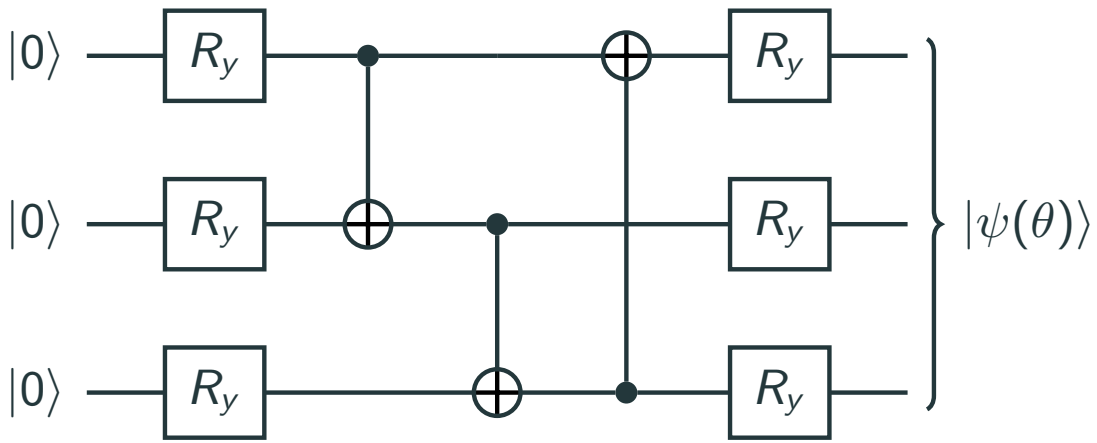
- faithful: minimum of $C(\theta)$ corresponds to the problem solution
- possible to estimate efficiently on a quantum computer
- difficult to simulate classically (otherwise no quantum advantage)
- meaningful: lower values of $C(\theta)$ correspond to better solution quality
- possible to optimize efficiently

Ansatz circuits

The second important part in VQAs is the parametrized quantum circuit, or the ansatz. The ansatz encodes the parameters θ in a quantum state where the observable is measured:

$$\rho \mapsto U(\theta)\rho U(\theta)^\dagger \ . \tag{3}$$

The parameters θ often appear as rotation angles in the circuit for U. Here's a typical example of a "problem-agnostic" ansatz circuit:



Here, the initial state is $\rho = |0\rangle\langle 0|$ of 3 qubits. There are 6 parameters which sit inside the R_y-rotations. Controlled-NOTs are intended to produce entanglement in the output state $|\psi(\theta)\rangle$. The cost function value would then be $C(\theta) = f(\langle\psi(\theta)|O|\psi(\theta)\rangle)$.

Gradients - parameter-shift rule

Often optimization is easier if we have access to the gradients of the cost function. How can we compute derivatives of $C(\theta)$ where the parameters sit inside the quantum gates? Finite difference approximations would be horribly noisy $\rightarrow$ not practical. The "parameter-shift rule" overcomes this and gives a convenient way to compute $\partial_{\theta_j} C(\theta)$.

Let's for simplicity consider the case with $f(x) = x$ and that the jth parameter θ_j parametrizes a term $e^{i\theta_j \sigma_j}$ in the ansatz, where σ_j is a Pauli operator. This is often the case, for example in the circuit in the previous slide the parameters are single qubit Pauli-Y rotation angles.

It can be shown (exercises) that

$$\frac{\partial C}{\partial \theta_j} = \frac{1}{2}\left(\langle \psi(\theta_+)|O|\psi(\theta_+)\rangle - \langle \psi(\theta_-)|O|\psi(\theta_-)\rangle \right) , \tag{4}$$

where $\theta_\pm = \theta \pm \frac{\pi}{2}\vec{e}_j$ and $\vec{e}_j$ is a unit vector in the jth direction. Note that this is not an approximation like finite-difference derivatives would be, this is exact! Moreover, the derivatives can be computed with the exact same circuit $U(\theta)$. Only the classical parameters need to be shifted by $\pm\pi/2$.

Optimizers

Now that we know how to compute $C(\theta)$ and its gradients, we are ready to optimize the θ to hopefully find the global minimum

$$\theta^* = \text{argmin}_\theta C(\theta) \ . \tag{5}$$

Unfortunately this is easier said than done: $C(\theta)$ is in general non-convex so the optimizer can get stuck in suboptimal local minima.

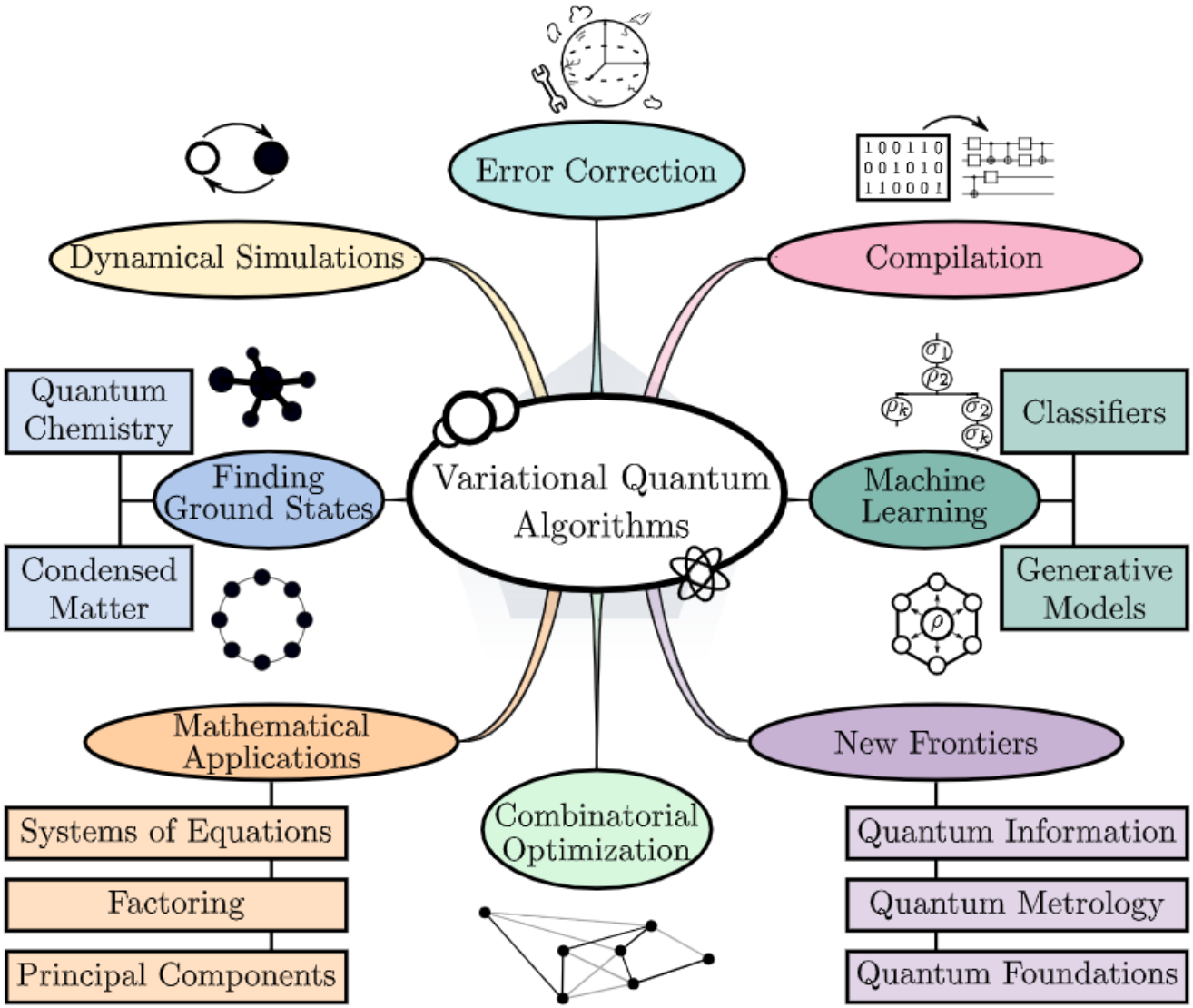
We can use now any gradient-based or gradient-free optimizer such as simple gradient descent

$$\theta_j^{(t)} = \theta_j^{(t-1)} - \epsilon \frac{\partial C(\theta^{(t-1)})}{\partial \theta_j} \tag{6}$$

or more sophisticated gradient-based optimizers such as Adam or RMSprop, which are readily available in modern machine learning packages. We can even use higher-order methods such as Newton's method because similar parameter-shift rules for higher-order derivatives of $C(\theta)$ exist.

Applications of VQAs

Applications of VQAs



Applications - finding ground states

One of the first practical applications of quantum computing might be the study of the physical properties of molecules. This is a very interesting and important application: it is not feasible to compute molecular properties from first principles classically except for very small molecules.

Let's now use VQAs in the task of finding the ground state of a given Hamiltonian. So, our observable is the Hamiltonian

$$O = H = \sum_i c_i \sigma_i \ , \tag{7}$$

where $c_i \in \mathbb{R}$ and σ_i denotes multi-qubit Pauli operators. The cost function is the expected energy

$$C(\theta) = E(\theta) = \langle \psi(\theta) | H | \psi(\theta) \rangle = \sum_i c_i \langle \psi(\theta) | \sigma_i | \psi(\theta) \rangle \geq E_0 \ , \tag{8}$$

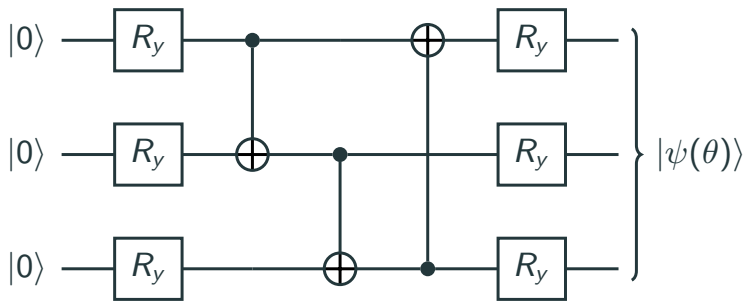
where E_0 is the ground state energy. We would then use the quantum processor to estimate all $\langle \psi(\theta) | \sigma_i | \psi(\theta) \rangle$, which gives us the energy $E(\theta)$. Finally we minimize this energy to obtain $E(\theta^*) \approx E_0$. This gives us the ground state energy E_0 and the ground state $|\psi(\theta^*)\rangle$. Algorithms of this type are called *variational quantum eigensolvers*.

Applications - finding ground states

As a simple example, consider the Hamiltonian

$$H = ZXX + \frac{1}{2}XXZ + \frac{1}{5}YZY \ . \tag{9}$$

You want to find the ground state of H. First take a parametrized circuit that produces your ansatz states. For example,



Then estimate the expectations for each term in H using the quantum computer for some parameter values θ

$$\rightarrow \langle\psi(\theta)|ZXX|\psi(\theta)\rangle \ , \quad \langle\psi(\theta)|XXZ|\psi(\theta)\rangle \ , \quad \langle\psi(\theta)|YZY|\psi(\theta)\rangle \ . \tag{10}$$

Then you can combine these in a classical algorithm to find $\langle\psi(\theta)|H|\psi(\theta)\rangle$ and $\nabla_\theta \langle\psi(\theta)|H|\psi(\theta)\rangle$. Then you would repeatedly $\theta \leftarrow \theta - \epsilon\nabla_\theta \langle\psi(\theta)|H|\psi(\theta)\rangle$ for some small ϵ until you reach a minimum.

Applications - finding ground states

If the ground state is found, then the corresponding energy $\langle\psi(\theta^*)|H|\psi(\theta^*)\rangle$ is the ground state energy and $|\psi(\theta^*)\rangle$ is the ground state.

How could this method fail? In a few ways, for example

- If classical optimization gets stuck in a local, false minimum.
- If the ansatz circuit $U(\theta)$ cannot approximate well the ground state.
- It might be too time consuming to estimate the energy and its gradient to sufficient accuracy.

More on this later.

Applications - combinatorial optimization

Another proposed applications of VQAs are combinatorial optimization problems. Combinatorial optimization appears everywhere in science and in the industry: e.g. logistics, portfolio optimization, etc. Optimization problems are often phrased as boolean satisfiability problems, max cut problems, etc which are all believed to be classically infeasible to solve.

One proposed quantum algorithm for optimization is the *Quantum Approximate Optimization Algorithm* (QAOA). QAOA is inspired by adiabatic quantum computing, where the idea is

1. Initialize the computer to the ground state of some simple initial Hamiltonian (should be easy to prepare)
2. Slowly change the initial Hamiltonian to the cost Hamiltonian: $H(t) = (1-t)H_i + tH_c$
3. It can be shown that in the end the quantum system is in the ground state of the cost Hamiltonian H_c

Now, if the ground state of H_c encodes the solution of an optimization problem you can measure the final state to find the solution! Caveat: sometimes you have to change the Hamiltonian *really* slowly for this to work.

Applications - combinatorial optimization

The primary difference between QAOA and VQE is the ansatz. QAOA uses a problem-inspired ansatz that resembles discretized adiabatic optimization. The parameters of QAOA are often denoted $\theta = (\gamma, \beta)$ and the ansatz is

$$|\psi(\gamma,\beta)\rangle = e^{-i\beta_p H_i} e^{-i\gamma_p H_c} \dots e^{-i\beta_1 H_i} e^{-i\gamma_1 H_c} |\psi_i\rangle \ , \tag{11}$$

where $|\psi_i\rangle$ is the ground state of the initial Hamiltonian H_i. The initial Hamiltonian is often called also the "mixer" or "driver" Hamiltonian. A common mixing Hamiltonian is $H_i = X^{\otimes n}$. The cost function is then

$$C(\gamma,\beta) = \langle\psi(\gamma,\beta)|H_c|\psi(\gamma,\beta)\rangle \ . \tag{12}$$

Again, $C(\gamma, \beta)$ is evaluated on the quantum computer and the parameters are optimized classically.

How can we construct problem Hamiltonians that encode a combinatorial optimization problem? Let's demonstrate with an example.

Applications - combinatorial optimization

Suppose we are trying to solve the binary set of equations

$$x_1 + x_2 = 1 \tag{13}$$

$$x_3 + x_4 = 1 \tag{14}$$

$$x_3 x_2 + x_1 x_4 = 1 \ , \tag{15}$$

where $x_i = \{0, 1\}$. Problems of this type are difficult so solve in general. Let's first convert the system to a classical cost function by considering the following sum of squares

$$(x_1 + x_2 - 1)^2 + (x_3 + x_4 - 1)^2 + (x_3 x_2 + x_1 x_4 - 1)^2 \tag{16}$$

$$= x_4^2 x_1^2 + x_1^2 + 2x_2 x_1 + 2x_2 x_3 x_4 x_1 - 2x_4 x_1 - 2x_1 + x_2^2 + x_2^2 x_3^2 + x_3^2 + x_4^2$$

$$-2x_2 - 2x_2 x_3 - 2x_3 + 2x_3 x_4 - 2x_4 + 3 \tag{17}$$

$$= 2x_2 x_1 + 2x_2 x_3 x_4 x_1 - x_4 x_1 - x_1 - x_2 - x_2 x_3 - x_3 + 2x_3 x_4 - x_4 + 3 \ . \tag{18}$$

Any assignment of x_i for which this evaluates to zero solves the original system of equations. Next, we will quantize this function.

Applications - combinatorial optimization

The binary variables x_i are quantized with the replacement

$$x_i \mapsto \frac{1}{2}(1 - Z_i) \ , \tag{19}$$

where Z_i is a Pauli-Z operator acting on the ith qubit. This works because the states $|0\rangle$ and $|1\rangle$ are eigenstates of the right hand side with eigenvalues 0 and 1. With this replacement, the cost Hamiltonian becomes

$$\begin{aligned} H_c = & \, 5Z_2Z_1 - Z_2Z_3Z_1 + Z_3Z_1 - Z_2Z_4Z_1 + Z_2Z_3Z_4Z_1 - \\ & + Z_3Z_4Z_1 - Z_4Z_1 + Z_1 + Z_2 - Z_2Z_3 + Z_3 + Z_2Z_4 \\ & - Z_2Z_3Z_4 + 5Z_3Z_4 + Z_4 + 13 \ . \end{aligned} \tag{20}$$

Notice that this Hamiltonian is diagonal so the eigenstates are simply the computational basis states. By construction the computational basis state $|x_1x_2x_3x_4\rangle$ for which

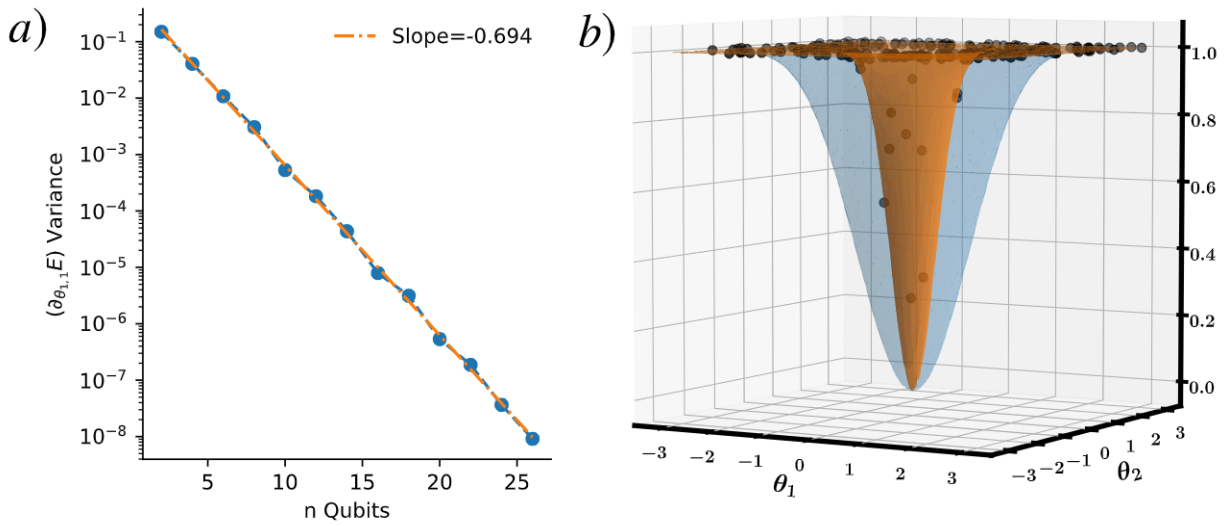
$$\langle x_1x_2x_3x_4|H_c|x_1x_2x_3x_4\rangle = 0 \ , \tag{21}$$

corresponds to the solution of the binary set of equations $|x_1x_2x_3x_4\rangle \mapsto \{x_1, x_2, x_3, x_4\}$. The job of QAOA is then to find parameters (γ, β) such that

$$|\psi(\gamma, \beta)\rangle \approx |x_1x_2x_3x_4\rangle \ . \tag{22}$$

Challenges - barren plateaus

One of the main difficulties in applications of VQAs is the existence of so-called barren plateaus. A barren plateau (BP) is a phanomenon where the magnitude of the partial derivatives $\partial_{\theta_i} C(\theta)$ become exponentially small in the number of qubits n. In other words, the energy landscape of the cost function becomes flat exponentially. This makes it infeasible to estimate the gradient reliably when n increases. This problem affects both gradient-based and gradient-free optimization methods. BPs depend on the properties of the problem Hamiltonian, the ansatz circuit, and noise. A BP kills any possible quantum advantage: if we need to spend exponential time in circuit sampling, the overall algorithm will be slow. Therefore, a lot of work is going into figuring out under which conditions BPs appear and how their effect could be mitigated.



Challenges - efficiency

For a VQA to be useful, one must be able to estimate the cost function efficiently. One barrier to this is the BP but another independent problem is that often interesting Hamiltonians might have many terms to estimate

$$H = \sum_i c_i \sigma_i \ , \tag{23}$$

where $c_i \in \mathbb{R}$ and σ_i are Pauli observables. For example, a Hamiltonian in quantum chemistry might have $\mathcal{O}(n^4)$ terms in the above sum. Even though the number of terms is polynomial, it might become practically impossible to estimate every term for large n. A possible solution to this is to measure all terms which commute with one another simultaneously (possible only because they commute). However, in general arranging the terms to the largest possible commuting sets is NP-hard. Heuristic methods still exist that can alleviate this problem.

Hardware noise can also adversely affect VQA efficiency: some methods for countering noise effects were covered in the last lecture.