Variational Algorithms

Xavier Bonet-Monroig, Casper Gyurik and Thomas O'Brien

February 26, 2020

Contents

---

1 Introduction

Quantum computing is currently in the 'NISQ', or Noisy, Intermediate-Scale Quantum era. State of the art hardware currently sits at qubit counts of 10-100, and error rates of 10^{-2}-10^{-3} (and that's with hiding a lot under the rug). By comparison, the lowest-cost known useful (i.e. better than what we can do classically and of interest to the wider world) algorithms require qubit counts of 100-200, and error rates of 10^{-6}-10^{-9}, depending on how optimistic we are. Quantum error correction allows us to sacrifice qubits for improved performance - the lowest-cost error corrected algorithms currently require around 10^5 qubits. The future dream is to achieve quantum error corrected devices of one million (insert Dr.Evil-like finger in mouth) qubits or beyond, but this requires we improve the state of the art by about a factor of a thousand or so, which realistic estimates place in the 10-20 year timeframe. In the meantime, before we achieve the 'promised land' of large-scale fault-tolerant devices, we are left trying to answer the question of whether there might be any applications for noisy quantum computers before then - hence the name NISQ. We will touch a little bit on the state of the art and quantum error correction in the last few lectures of the class.

Variational methods have emerged in the last few years as a promising way to utilize NISQ-era quantum computers. They can be made as low-cost as the user desires, although this requires sacrificing algorithm performance. The hope is that by balancing this trade-off, we can find some problem + variational algorithm solution that can outperform classical methods without breaking the error budget. As is the case throughout this course, both finding applications and designing variational algorithms are active, open areas of research — the field is only 5 years old! This means that the state-of-the-art in what we are about to cover in many cases could most likely be improved on.

2 The variational paradigm

The variational principle states that, given a quantum state $|\Psi\rangle$ and an observable $\hat{O}$ (e.g. a Hamiltonian $\hat{O} = \hat{H}$ of a quantum system), the expectation value

$$\left\langle \hat{O} \right\rangle_{|\Psi\rangle} = \frac{\langle \Psi | \hat{O} | \Psi \rangle}{\langle \Psi | \Psi \rangle} \geqslant O_0, \tag{1}$$

where O_0 is the lowest eigenvalue of $\hat{O}$. Moreover, if $\left\langle \hat{O} \right\rangle_{|\Psi\rangle} = O_0$, then $|\Psi\rangle$ is the ground state $|O_0\rangle$ of $\hat{O}$ (if $\hat{O}$ has multiple eigenstates with the same eigenvalue O_0, $|\Psi\rangle$ lives in the space spanned by these states). This suggests a simple method for finding $|O_0\rangle$ - prepare a lot of different 'trial states' $|\Psi\rangle$, measure their expectation value, and choose the state where this is minimized. More generally (as Casper will cover later in this lecture), if we have any cost function that is easy to compute on $|\Psi\rangle$, we can equally well optimize using this cost function.

To 'prepare a lot of different trial states and minimize a cost function', it helps if we can parameterize our state. We can make this parametrization classical - i.e, given a list of parameters $\vec{\theta} \in \mathbb{R}^{N_{\text{params}}}$, we can prepare a different trial state $|\Psi(\vec{\theta})\rangle$ for each $\vec{\theta}$. One might worry that the classical parameterization makes the calculation of $\left\langle \hat{O} \right\rangle$ easy classically, but we avoid this by using the angles to control a quantum

circuit. That is, we use the parameters to define a set of unitaries $\hat{U}(\vec{\theta})$ that we apply to a starting state $|\vec{0}\rangle$ to generate the trial state

$$|\Psi(\vec{\theta})\rangle = \hat{U}(\vec{\theta})|\vec{0}\rangle. \tag{2}$$

We call the combination $(\hat{U}(\vec{\theta}), |\vec{0}\rangle)$ the 'variational ansatz'. Assuming that the vectors $\vec{\theta}$ are continuously (or approximately continuously) defined, a variational ansatz sweeps out a (possibly open) manifold in the Hilbert space of states — that is, it explores some smooth space that lies within the set of allowable Hilbert space. With the exception of possible boundaries, the dimension of this manifold is precisely N_{params}. This shows the limit of the variational paradigm; to explore the entire 2^N-dimensional Hilbert space and guarantee that we find the ground state we would require $O(2^N)$ parameters (more precisely, $2 \times 2^N - 2$), and a similarly-scaling amount of time. This is hardly surprising; as we learnt in the previous lectures, performing such a task is a known QMA-hard problem, so if we thought we had an exponential speedup we would either have won a million dollars or made a mistake. Instead, the 'art' of variational algorithm design is in choosing a manifold that contains points 'sufficiently close' to the true ground state, and the 'art' of application design is in choosing a problem so that 'sufficiently close' is achievable.

3 The variational algorithm protocol

As defined above, the variational ansatz is not an algorithm to solve a problem; we also need to describe how we will evaluate the cost function, and how we will choose our parameters based on the cost function. When the full protocol is defined, it has a few names, but possibly the most general is 'variational algorithm', so let's stick with this. The full protocol consists of (Fig. 1):

1. an initial state preparation,

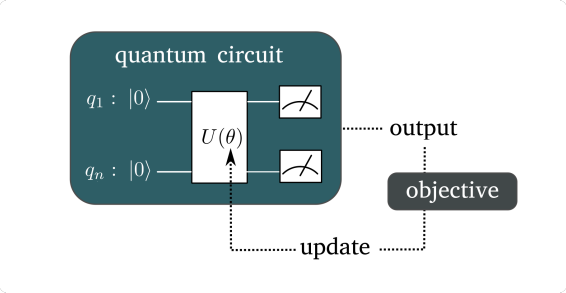

Figure 1: Scheme for a variational algorithm. Picture taken from Xanadu quantum machine learning toolbox documentation.

2. a parametrized quantum circuit

3. a cost function and a method to evaluate/estimate it.

4. a classical optimization routine to optimize the parameters based on the cost function input.

4 Continuously parametrized unitaries

In previous lectures we have discussed gates as discrete objects, e.g. H, CNOT, X, T. Let us now consider how we can construct a generic quantum gate, and in particular, how one could be parameterized.

An incredibly useful set of operators in quantum computing are the Pauli matrices $\mathcal{P} = \{I, X, Y, Z\}$, or more generally the Pauli basis, $\mathcal{P} = \{I, X, Y, Z\}^{\otimes N}$. The Pauli basis contains all possible tensor products of the Pauli matrices acting on N-qubits, for a total of 4^N elements. Elements of the Pauli basis are typically written without explicit tensor products - e.g. $ZZ \equiv Z \otimes Z$. Moreover, instead of writing a lot of I's, we often drop them, and instead index operators with which qubits they act on - e.g. $Z_1 X_3 \equiv ZIX \equiv Z \otimes I \otimes X$.

The Pauli matrices are both unitary and hermitian. Given a Pauli operator

$\hat{P} \in \mathcal{P}^N$

$$\hat{P}^\dagger \hat{P} = I \tag{3}$$

$$\hat{P} = \hat{P}^\dagger \tag{4}$$

$$\rightarrow \hat{P}^2 = I. \tag{5}$$

Note that this implies a Pauli operator can only have eigenvalues ± 1. Also, Pauli operators are traceless, and indeed trace-orthogonal — $\text{Trace}[\hat{P}_i, \hat{P}_j] = 2^N \delta_{i,j}$. As Pauli operators are Hermitian, they may be exponentiated to construct unitary operations:

$$U(\hat{P}) = e^{-i\theta\hat{P}}. \tag{6}$$

Now, typically exponentiated operators are quite scary to deal with, but the properties of the Pauli operator make this far nicer: let us Taylor expand the above

$$\begin{aligned} U_{\hat{P}}(\theta) = e^{-i\theta\hat{P}} = \sum_k \frac{(-i\theta\hat{P})^k}{k!} = \sum_k \frac{(-i\theta\hat{P})^{2k}}{2k!} + \frac{(-i\theta\hat{P})^{2k+1}}{(2k+1)!} = \\ \sum_k (-i)^{2k} \frac{\theta^{2k}}{2k!} P_i^{2k} + (-i)^{2k+1} \frac{\theta^{2k+1}}{(2k+1)!} P_i^{2k+1} = \\ \sum_k (-1)^k \frac{\theta^{2k}}{2k!} I + (-i) \frac{\theta^{2k+1}}{(2k+1)!} P_i, \end{aligned} \tag{7}$$

and we find

$$U_{\hat{P}}(\theta) = \cos(\theta) I - i \sin(\theta) \hat{P}. \tag{8}$$

Note here that critically, although $\hat{P}$ when acting as a unitary operator by itself does not generate any entanglement between qubits, $U_{\hat{P}}(\theta)$ is an entangling gate! Indeed, as we will cover in the next lecture, every possible operation on a quantum computer may be approximated by a series of $U_{\hat{P}_i}(\theta)$ for different operators P_i.

Physically, the expectation value of a Pauli operator acting on a single qubit can be thought of as the splitting of the two eigenstates of that qubit for a period

of time, and typically on NISQ devices some gate similar to $e^{i\theta Z_i}$ may be achieved. (We will cover this in more detail later in the course.) How can we extend this to a multi-qubit gate? It turns out we only need two little tricks. The first is to note that sandwiching a Pauli-Z rotation between CNOT gates extends it to additional qubits:

$$CNOT_{n-1,n} e^{i\theta Z_1 \ldots Z_{n-1}} CNOT_{n-1,n} = e^{i\theta Z_1 \ldots Z_{n-1} Z_n}, \tag{9}$$

while the second is to note that sandwiching a Pauli-Z rotation between single-qubit rotations changes the axis of the tensor factor on the targetted qubit, e.g.

$$e^{-i\frac{\pi}{4}Y_i} e^{i\theta Z_1 \ldots Z_i \ldots Z_n} e^{i\frac{\pi}{4}Y_i} = \exp\left(i\theta Z_1 \ldots e^{-i\frac{\pi}{4}Y_i} Z_i e^{i\frac{\pi}{4}Y_i} \ldots Z_n\right) = e^{i\theta Z_1 \ldots X_i \ldots Z_n}. \tag{10}$$

Using this, a generic rotation of the form $e^{-i\theta \hat{P}}$ may be constructed (give circuit here). Note that this circuit has depth equal to the number of terms in $\hat{P}$, which may be quite a high cost for many applications.

5 Constructing suitable trial states

Variational ansatz design (and performance evaluation) is a very active and unfinished field of research. This is perhaps because the problem to solve here is hard — in most cases finding the (absolutely) optimal ansatz for a given problem is harder than solving the problem itself. Moreover, given the exponentially small fraction of the Hilbert space that we expect to explore (in order to maintain a quantum advantage), we do not expect randomly generated ansatze to be 'good enough'. (This is made worse by the difficulties encountered in optimizing an arbitrary ansatz, that Casper will describe later today.) In this lecture I will describe four common ways of motivating variational ansatzes. During the rest of the course some of these and others will be discussed in detail.

5.1 Perturbative ansatzes

A common theme in quantum computing (and quantum mechanics in general) is to take a Hamiltonian $\hat{H}_0$ that we know the ground state of, and a Hamiltonian $\hat{H}_1$ that we do not know the ground states of, and try to get from the ground states of $\hat{H}_0$ to those of $\hat{H}_1$. Many quantum mechanical systems of interest take the form $\hat{H}_1 = \hat{H}_0 + \sum_i \lambda_i \hat{V}_i$. When λ is small, the ground state of $\hat{H}_1$ is connected to the ground state of $\hat{H}_0$ by applications of the $\hat{V}_i$. This may be formally expanded around using perturbation theory, but the number of terms needing calculation grows quite quickly, and the approximation also typically breaks down at some point. However, on a quantum device, this motivates perturbing the system by the individual $\hat{V}_i$, leading to variational ansatze of the form

$$\hat{U}(\vec{\theta}) = \prod_i e^{i\theta_i V_i}. \tag{11}$$

When V_i are Pauli operators, this can be implemented using the circuits from the previous section. These ansatze have the advantage of being reasonably well-motivated (we will discuss this in a few weeks), but are often rather expensive; optimizing over these circuits remains an active area of research.

5.2 The quantum alternating operator ansatz

In keeping with the above idea of deforming from known states of $\hat{H}_0$ to unknown states of $\hat{H}_1$, one popular technique is to repeatedly alternate between applying the two Hamiltonians for variational time t.

$$\hat{U}(\vec{\theta}) = \prod_n e^{i\hat{H}_0\theta_{2n}} e^{i\hat{H}_1\theta_{2n+1}}. \tag{12}$$

This is commonly known as the 'quantum alternating operator ansatz' or QAOA. This has additional physical motivation based on adiabaticity - the idea that slowly

deforming from $H_0 \to H_1$ should preserve the ground state. We will cover the physical motivation later in the course, as well as going into applications of this ansatz in discrete optimization in tutorials.

5.3 Swap networks and other banded ansatzes

5.4 Hardware-efficient ansatzes

A more realistic approach to generate random quantum circuits is the so-called hardware-efficient ansatzes proposed in [1]. The idea is to generate random parametrized quantum circuits but only using those gates that the hardware performs better. By doing so one expects to reduce errors during the computation while (hopefully) being able to generate meaninful quantum states.

In the original proposal by Kandala et al. the circuits are built by alternating layers of parametrized single-qubit gates and two-qubit entangling gates. The choice of the single-qubit rotations is made at random, assigning an angle to each one of them. Ideally, one would expect to be able to perform entangling gates between all pairs of qubits. Hoowever the reality is that only some qubits are connected between them, thus limiting the ability to perform two-qubit entangling gate.

6 Extracting classical information and the cost function

The next stop on our journey through variational quantum algorithms is the discussion of cost functions. Although constructing quantum states is an interesting undertaking in itself, it does not provide us with any useful classical information. The final steps of a variational quantum algorithms are concerned with extracting this useful classical information from the quantum state. To this end, we are going to have to measure our quantum state.

In quantum mechanics, one has the ability to measure so-called "observables". In mathematical terms, an observable $\hat{O}$ is an Hermitian operator (i.e., $\hat{O}^\dagger = \hat{O}$). The set of all N-qubit Hermitian operators is denoted by $\mathrm{Herm}(\mathbb{C}^{2^N})$. This set is in fact a real vector space (i.e., it is closed under linear combinations with real coefficients) and we can equip it with the Hilbert-Schmidt inner product given by

$$\langle A, B \rangle = \mathrm{tr}(A^\dagger B).$$

An interesting subset of $\mathrm{Herm}(\mathbb{C}^{2^N})$ is the set of all so-called *Pauli strings* given by $\{I, X, Y, Z\}^{\otimes N}$. That is, a Pauli string is an Hermitian operator of the form $P_1 \otimes \cdots \otimes P_N$, where $P_i \in \{I, X, Y, Z\}$. What makes these Pauli strings such an interesting set, among other things, is the following lemma.

Lemma 1. *The Pauli strings are a basis of the real vector space Herm*$(\mathbb{C}^{2^N})$.

Proof. The space of all $2^N \times 2^N$ complex matrices has dimension $2^N \cdot 2^N = 4^N$ over the complex numbers. Because the complex numbers have dimension 2 over the real numbers (i.e., $\mathbb{C} \simeq \mathbb{R}^2$), we find that the space of all $2^N \times 2^N$ complex matrices has dimension $2 \cdot 4^N$ over the real numbers. Because Hermitian matrices have the extra property that $A^\dagger = A$, we find that the space of $2^N \times 2^N$ Hermitian matrices has dimension $2 \cdot 4^N/2 = 4^N$ over the real numbers.

Next, we notice that $\#\{I, X, Y, Z\}^{\otimes N} = 4^N$ and that all Pauli strings are linearly independent as they are orthogonal under the Hilbert-Schmidt inner product. Therefore, we may indeed conclude that the Pauli strings $\{I, X, Y, Z\}^{\otimes N}$ form a basis of the real vector space $\mathrm{Herm}(\mathbb{C}^{2^N})$. □

An important consequence of this lemma is that any Hermitian operator can be written as the linear combination of Pauli strings with real coefficients, i.e., any

$\hat{O} \in \text{Herm}(\mathbb{C}^{2^N})$ can be written as

$$\hat{O} = \sum_{i=1}^{4^N} h_i \mathcal{P}_i, \tag{13}$$

with $\mathcal{P}_i \in \{I, X, Y, Z\}^{\otimes N}$ and $h_i \in \mathbb{R}$.

6.1 Definition of our cost function

Now that we know more about the structure of $\text{Herm}(\mathbb{C}^{2^N})$, we have all the necessary ingredients to delve further into the study of cost functions. The standard definition of a VQE cost function, denoted $f_{\hat{O}}(\vec{\theta})$, is the expectation value of an observable $\hat{O} \in \text{Herm}(\mathbb{C}^{2^N})$, that is,

$$f_{\hat{O}}(\vec{\theta}) = \langle \psi(\vec{\theta}) | \hat{O} | \psi(\vec{\theta}) \rangle. \tag{14}$$

Let us look at a couple of examples of cost functions.

- **Example 1:** Approximating the ground state energy of an Hamiltonian.

In physics, one is often interested in computing the ground state energy E_0 of a Hamiltonian H. In mathematical terms, this is given by the smallest eigenvalue of H, i.e.,

$$E_0 = \min_{|\psi\rangle} \frac{\langle \psi | H | \psi \rangle}{\langle \psi | \psi \rangle}.$$

In this case it is clear that the corresponding cost function is given by

$$f_H(\vec{\theta}) = \langle \psi(\vec{\theta}) | H | \psi(\vec{\theta}) \rangle.$$

- **Example 2:** Approximating the MaxCut of a graph.

Let $G = (V, E)$ be a graph. The MaxCut of this graph, denoted $\text{MaxCut}(G)$, is the maximal size of a cut. A cut is a partitioning of the vertices into two disjoint subsets V_1, V_2 and the size of this cut is given by the number of edges between V_1 and V_2. That is,

$$\text{MaxCut}(G) = \max\{\#(V_1 \times V_2) \cap E \mid \text{disjoint } V_1, V_2 \subset V \text{ with } V_1 \cup V_2 = V\}.$$

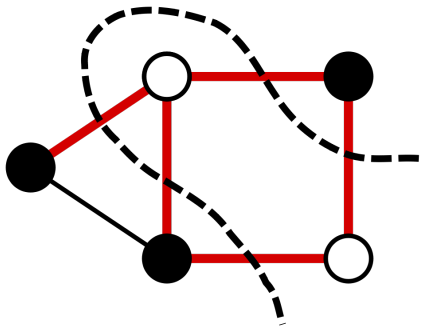

Figure 2: An example of a MaxCut (source: https://en.wikipedia.org/wiki/Maximum_cut).

An interesting observation is the MaxCut of a graph G can be encoded into a cost function as defined in equation 14. Namely, let $V = \{1, \ldots, N\}$ and consider the N-qubit observable

$$\hat{O}_{cut} = \frac{1}{2} \sum_{(i,j)\in E} (I^{\otimes N} - Z_i Z_j), \tag{15}$$

where Z_i denotes the Pauli-string with a Z on the i-th qubit and I everywhere else. Then, one can show that

$$\text{MaxCut}(G) = \lambda_{min}(\hat{O}_{cut}),$$

and we find that $\min_\theta f_{\hat{O}_{cut}}(\vec{\theta})$ could potentially be a good approximation of $\text{MaxCut}(G)$ for a well-chosen ansatz.

- **Example 3:** Kullback-Leibler divergence for generative modelling.

Another application of variational quantum algorithms is *generative modelling*, a branch of machine learning where the goal is to mimic a given probability distribution p. That is, given a bunch of samples from a probability distribution p, generate new samples from a distribution close to p.

A quantum state $|\Psi(\vec{\theta})\rangle$ together with an observable $\hat{O}$ induces a probability distribution, let's denote this distribution by q_θ. The goal would be to prepare a quantum state $|\Psi(\vec{\theta})\rangle$ such that when measuring the observable $\hat{O}$, we get a distribution that closely mimics p. A well-known cost function for this is the so-called *Kullback-Leibler divergence*, which serves as a measure of how one probability distribution is different from a second, and is given by

$$f_{D_{KL}}(\vec{\theta}) = D_{KL}(p, q_\theta) = \sum_x p(x) \log\left(\frac{p(x)}{q_\theta(x)}\right).$$

A problem with the Kullback-Leibler divergence is that it is hard to evaluate since we do not know the probabilities $p(x)$ or $q_\theta(x)$, as we can only generate samples from these distributions. More on variational quantum algorithms for generative modelling will be discussed during the quantum machine learning part of the course.

6.2 Estimating the cost function

Let us now adress the question of how to evaluate the VQE cost functions defined in Equation 14. As we have shown in Lemma 1, any N-qubit observable $\hat{O}$ can be written as a linear combination of at most 4^N Pauli strings with real coefficients, as in Equation 13. It turns out however, that most interesting observables $\hat{O}$ can be written as a linear combination of polynomially many Pauli strings with real coefficients, i.e.,

$$\hat{O} = \sum_{i=1}^{\text{poly}(N)} h_i \mathcal{P}_i. \tag{16}$$

For example, most physically motivated Hamiltonians are k-local. That is, each of the Pauli strings that appear in the decomposition of such Hamiltonians, acts nontrivially on at most k of the qubits. As there are $\binom{n}{k} \sim n^k$ possible subsets of k qubits, the number of Pauli strings that appear in the decomposition of a k-local Hamiltonian is bounded by a polynomial.

When implementing Equation 16 into Equation 14 we find that

$$f_{\hat{O}}(\vec{\theta}) = \langle\psi(\vec{\theta})|\hat{O}|\psi(\vec{\theta})\rangle = \sum_{i}^{\text{poly}(N)} \langle\psi(\vec{\theta})|h_i\mathcal{P}_i|\psi(\vec{\theta})\rangle = \sum_{i}^{\text{poly}(N)} h_i\langle\psi(\vec{\theta})|\mathcal{P}_i|\psi(\vec{\theta})\rangle. \quad (17)$$

From this we deduce that evaluating our cost function $f_{\hat{O}}(\vec{\theta})$ comes down to evaluating the expectation value of the relevant Pauli-strings $\langle\psi(\vec{\theta})|\mathcal{P}_i|\psi(\vec{\theta})\rangle$, multiplying them with the real coefficients h_i and finally summing them up. Note that we assume here that the decomposition in Equation 16 is given, that is, the h_i are known beforehand.

6.2.1 Chebyshev's inequality

Suppose we want to estimate $E = \langle\psi(\vec{\theta})|\mathcal{P}|\psi(\vec{\theta})\rangle$, for some Pauli string $\mathcal{P}$, up to precision $\epsilon > 0$. To do so, let $\mathcal{P} = \sum_{k=1}^{2^N} \lambda_k|\varphi_k\rangle\langle\varphi_k|$ denote the spectral decomposition of $\mathcal{P}$. In the first tutorial we showed that E is the expected value of the random variable X with possible outcomes $\{\lambda_i\}_{i=1}^{2^N}$ and

$$\mathbb{P}(X = \lambda_k) = |\langle\varphi_k|\psi(\vec{\theta})\rangle|^2.$$

That is, X is the random variable representing the outcome of measuring $\hat{O}$ on the state $|\psi(\vec{\theta})\rangle$.

A well-known result in probability theory is the so-called *Chebyshev's inequality*, which states that if we take M copies of our random variable X, which we denote

$X_1, \ldots, X_M$, then

$$\mathbb{P}\left(\left|\frac{\sum_{i=1}^{M} X_i}{M} - E\right| \geqslant \epsilon\right) \leqslant \frac{\sigma^2}{M\epsilon^2}, \tag{18}$$

where σ denotes the variance of X. From Equation 18, we deduce that for

$$M = \frac{\sigma^2}{\epsilon^2 0.01},$$

we find that

$$\mathbb{P}\left(\left|\frac{\sum_{i=1}^{M} X_i}{M} - E\right| \geqslant \epsilon\right) \leqslant 0.01,$$

As the variance of X can usually assumed to be bounded above by a constant, Equation 18 tells us that we need to perform a number of measurements that scales as

$$M \sim \frac{1}{\epsilon^2}$$

to obtain, with high probability, an estimate of E to within additive precision ϵ.

A very important caveat is that most of the current quantum circuits that we can run suffer from noise. This has severe implications on how well the above strategy allows us to estimate the energies E, as our measurement outcomes will be noisy (i.e., our samples from X are noisy). We will come back to the consequences of this when discussing the optimization of our cost function.

6.2.2 Measurement optimization

A recent and active area of research within variational quantum algorithms is that of optimizing measurement strategies. Our previous strategy is a very naive way of estimating the cost function, we sample every Pauli string seperately and use the

outcome statistics to estimate the corresponding expectation values. However, if we know that two or more of our Pauli strings commute, then we know that we can measure them simultaneously. The problem is then to divide our Pauli strings into the smallest number of commuting subsets. It turns out that this problem is equivalent to finding the smallest clique cover of a graph, which is known to be NP-complete. Researchers have tried using approximation algorithms for the clique cover problem as a grouping strategy, with various degrees of success.

7 Optimizing the cost function

The final stop on our journey through variational quantum algorithms is at the classical optimization routine. When choosing our ansatz, the hope is that it allows us to explore a region of the Hilbert space in which the minimum $\min_{\vec{\theta}} f_{\hat{O}}(\vec{\theta})$ is close to the actual ground state energy of $\hat{O}$. Thus, after having choosing our ansatz, the challenge that remains is finding a $\vec{\theta}$ that minimizes our cost function $f_{\hat{O}}$. To this end, we will employ a classical optimization routine. It is an open question which classical optimization routine works best for variational quantum algorithms. Broadly speaking, classical optimization algorithms can be split into two groups: gradient-based methods and gradient-free methods.

7.1 Gradient-based methods

As the name suggests, gradient-based methods employ the gradient of the cost function

$$\nabla_{\vec{\theta}} f_{\hat{O}}(\vec{\theta}) = \begin{pmatrix} \frac{\partial f_{\hat{O}}(\vec{\theta})}{\partial \vec{\theta}_1} \\ \vdots \\ \frac{\partial f_{\hat{O}}(\vec{\theta})}{\partial \vec{\theta}_{N_{\text{params}}}} \end{pmatrix}.$$

Because the gradient points along the direction of the fastest increase, by following the gradient in the opposite direction one is able to find (local) minima.

Pros:

- Works incredibly well when your cost function landscape is nicely smooth.
- Convergence properties are very well established.

Cons:

- Unreliable under noisy gradient evaluations.
- Suffers greatly in the presence of vanishing and exploding gradients (i.e., when your cost function landscape is barren or rigid).
- Can take very long if gradient evaluation is expensive.

When to use: If you have the prior knowledge that your landscape is smooth and you can efficiently evaluate the gradient.

7.2 Gradient-free methods

As the name suggests, gradient-free methods usually do not employ the gradient of the cost function. They generally depend on evaluating the function on many points in your landscape and inferring a minimum from these evaluations.

Pros:

- Works decently well even when your landscape is barren or rigid.
- Does not require the ability to (efficiently) evaluate the gradient.

Cons:

- Does not converge as quick as gradient-based methods when the landscape is nice and smooth.
- Requires a lot of function evaluations in general.

When to use: If you have no prior knowledge that your landscape is smooth (or you know that your landscape is very barren or rigid) or when evaluating the gradient is very expensive.

7.3 Minimizing the VQE cost function $f_{\hat{O}}(\vec{\theta})$

Let us now look at the special case where the cost function is the VQE cost function $f_{\hat{O}}(\vec{\theta})$, as defined in Equation 14. We will discuss two ways in which to estimate the gradient of $f_{\hat{O}}(\vec{\theta})$ together with some results relating to the landscape of this cost function.

7.3.1 Estimating the gradient of $f_{\hat{O}}(\vec{\theta})$

Parameter-shift rule It turns out that for most ansatzes you can evaluate $\nabla_{\vec{\theta}} f_{\hat{O}}$ at $\vec{\theta}$ by evaluating the function $f_{\hat{O}}$ at two different values $\vec{\theta}^+$ and $\vec{\theta}^-$. Namely, the *parameter-shift rule* states that

$$\frac{\partial f_{\mathcal{P}}(\vec{\theta})}{\partial \vec{\theta}_j} = \frac{f_{\mathcal{P}}(\vec{\theta} + \frac{\pi}{2} e_j) + f_{\mathcal{P}}(\vec{\theta} - \frac{\pi}{2} e_j)}{2} \tag{19}$$

For a detailed derivation see[1]. The downside of the parameter-shift rule is that it requires you to estimate the cost function twice for each parameter $\vec{\theta}_j$, resulting in a total of $2N_{\text{params}}$ evaluations of the cost function per gradient evaluation.

[1] https://arxiv.org/abs/1811.11184

Stochastic finite difference approximation One way to get around having to do a lot of function evaluations per gradient evaluation is to use a *stochastic finite difference approximation* of the gradient. For example, one can sample a random pertubation vector $\Delta \in \{\pm 1\}^{N_{\text{params}}}$, such that $\mathbb{P}(\Delta_i = 1) = \mathbb{P}(\Delta_i = -1) = 0.5$, and one uses the approximation

$$\frac{\partial f_{\mathcal{P}}(\vec{\theta})}{\partial \vec{\theta}_j} \approx \frac{f_{\mathcal{P}}(\vec{\theta} + c\Delta) + f_{\mathcal{P}}(\vec{\theta} - c\Delta)}{2c\Delta_j}. \tag{20}$$

for some small $c > 0$. The advantage of using Equation 20 is that it only requires you to evaluate the cost function twice in total. However, the disadvantage is that Equation 20 only gives an approximation of the gradient, whereas Equation 19 gives you a way to compute the analytic gradient exactly. It has been shown that techniques based on Equation 20 perform well in the noisy experimental setting[2].

7.3.2 Barren plateaus and parameter initialization

When looking at the landscape of cost functions that arise in variational quantum algorithms, it has been shown that they contain a number of barren plateaus. To be precise, it has been shown that for a wide class of reasonable ansatzes and observables, the probability that the gradient along any reasonable direction is non-zero to some fixed precision is exponentially small as a function of the number of qubits[3].

This raises the question of parameter initialization. That is, can we initialize our parameters in such a way as to avoid these barren plateaus? Unfortunately, due to the lack of known structure of the landscape for many interesting problems, this is a hard problem.

[2] https://arxiv.org/abs/1804.11326
[3] https://www.nature.com/articles/s41467-018-07090-4

7.3.3 Dealing with noise

As mentioned before, the noise in our current quantum hardware causes our cost function evaluations to be noisy and therefore to contain errors. As a final remark it is important to mention that variational quantum algorithms have been shown to be somewhat resilient to this noise. This is due to the ability to counteract errors in the evaluation by varying the parameter, that is, if an error slightly moves our minimum, then it could be possible to still find this minimum by changing the parameters correspondingly. In the NISQ (i.e., noisy intermediate scale quantum) era the ability to deal with noise is an important challenge in order to find applications where quantum computers can provide advantage over classical computers. It is therefore an interesting research question which classical optimizers deal best with the quantum noise in our function evaluations.

References

[1] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets," *Nature*, vol. 549, no. 7671, p. 242, 2017.