19

Quantum gradient estimation

The authors are grateful to Nikitas Stamatopoulos for reviewing this chapter.

Rough overview (in words)

Estimating the gradient of a high-dimensional function is a widely useful subroutine of classical and quantum algorithms. The function's gradient at a certain point can be classically estimated by querying the value of the function at many nearby points. However, the number of evaluations will scale with the number of dimensions in the function, which can be very large. By contrast, the quantum gradient estimation algorithm evaluates the function a *constant* number of times (in superposition over many nearby points) and uses interference effects to produce the estimate of the gradient. While there are caveats related to the precise access model and the classical complexity of gradient estimation in specific applications, this procedure can potentially lead to significant quantum speedups.

Rough overview (in math)

Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a real function on d-dimensional inputs, and assume that it is differentiable at a specific input of interest, taken to be the origin $\mathbf{0} = (0, 0, \ldots, 0)$ for simplicity (the algorithm works equally well elsewhere). Let $g = (g_1, \ldots, g_d)$ denote the gradient of f at $\mathbf{0}$, that is, $g = \nabla f(\mathbf{0})$. We wish to produce a classical estimate $\tilde{g}$ of g that satisfies $|g_j - \tilde{g}_j| < \varepsilon$ for all $j = 1, \ldots, d$.

Ignoring higher-order terms, the function may be approximated near the origin as $f(x) \approx f(\mathbf{0}) + \langle g, x\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the normal inner product. The original gradient estimation algorithm by Jordan [587] then considers a d-dimensional grid of points near the origin denoted by G. For simplicity, suppose on each of the d dimensions, the grid has N evenly spaced points on the

https://doi.org/10.1017/9781009639651.022 Published online by Cambridge University Press

interval $[-\ell/2, \ell/2]$, for a certain parameter ℓ related to the precision requirements of the algorithm, where N is assumed to be a power of 2. Let m be an upper bound on the magnitude of the components of g. Define $g' = Ng/2m$ to have components between $-N/2$ and $N/2$. Similarly, let $\tilde{g}' = N\tilde{g}/2m$ be the desired normalized-and-shifted output.

The quantum algorithm prepares a superposition of the grid points $x \in G$ and computes function $f(x)$ (times a constant $\pi N/m\ell$) into the phase, producing the state

$$\frac{1}{\sqrt{N^d}} \sum_{x \in G} \mathrm{e}^{\mathrm{i}\pi N f(x)/m\ell} |x\rangle \approx \frac{\mathrm{e}^{\mathrm{i}\pi N f(\mathbf{0})/m\ell}}{\sqrt{N^d}} \sum_{x \in G} \mathrm{e}^{\mathrm{i}\pi N \langle g, x\rangle/m\ell} |x\rangle \,,$$

where $|x\rangle$ denotes the product state $|l_1\rangle|l_2\rangle \cdots |l_d\rangle$, where l_j is a binary string of length $\log_2(N)$ containing a representation of the j-th component x_j of the vector x, with the identification $x_j = -\ell/2 + \ell l_j/N$. With this in mind, the latter state is rewritten as the product state, up to a global phase and normalization constant

$$\left(\mathrm{e}^{-\pi \mathrm{i} g'_1} \sum_{l_1=0}^{N-1} \mathrm{e}^{2\pi \mathrm{i} l_1 g'_1/N} |l_1\rangle\right)\left(\mathrm{e}^{-\pi \mathrm{i} g'_2} \sum_{l_2=0}^{N-1} \mathrm{e}^{2\pi \mathrm{i} l_2 g'_2/N} |l_2\rangle\right) \cdots \left(\mathrm{e}^{-\pi \mathrm{i} N g'_d} \sum_{l_d=0}^{N-1} \mathrm{e}^{2\pi \mathrm{i} l_d g'_d/N} |l_d\rangle\right).$$

Due to the approximated linearity of f, each of the product state constituents is observed to be close to a basis state in the Fourier basis (see Eq. (12.1)). By performing an inverse quantum Fourier transform (QFT) in parallel for each of the d dimensions and measuring in the computational basis, a computational basis state

$$|\tilde{g}'\rangle = |\tilde{g}'_1\rangle|\tilde{g}'_2\rangle \cdots |\tilde{g}'_d\rangle$$

is retrieved (up to an unimportant global phase), where with high probability $\tilde{g}'_j$ approximates g'_j to $\log_2(N)$ bits of precision. The coordinate $\tilde{g}_j$ is then recovered as $\tilde{g}_j = 2m\tilde{g}'_j/N$. Assuming $m = O(1)$, taking $N = O(1/\varepsilon)$ suffices to solve the problem. In a full analysis, one must make sure not to choose ℓ too large (else the linearity approximation breaks down).

In [587], the unitary U_f sending $|x\rangle \mapsto \mathrm{e}^{\mathrm{i}\pi N f(x)/m\ell}|x\rangle$ was performed using a constant number of calls to the evaluation oracle that computes an approximation to $f(x)/m$ to precision $O(\varepsilon^2/\sqrt{d})$ into an ancilla register. In [430], the precision required was improved to $O(\varepsilon/\sqrt{d})$ using finite difference formulas to put the gradient into the phase. Additionally, it was shown how U_f can be implemented using $O(\sqrt{d}/\varepsilon)$ calls to a "probability oracle" that (assuming $0 \le f(x) \le 1$) performs the map $|x\rangle|0\rangle \mapsto \sqrt{f(x)}|x\rangle|1\rangle + \sqrt{1 - f(x)}|x\rangle|0\rangle$.

The gradient estimation algorithm can be viewed as a generalization of the Bernstein–Vazirani algorithm [129], which considers binary functions f :

https://doi.org/10.1017/9781009639651.022 Published online by Cambridge University Press

$\{0,1\}^n \to \{0,1\}$, and promised that $f(x) = \langle g, x\rangle \bmod 2$ for some unknown vector g, determines g with one query to f.

Dominant resource cost (gates/qubits)

The superposition over grid points can be easily accomplished with Hadamard gates. Likewise, the inverse QFT operation is relatively cheap. The number of qubits is $O(d\log(N))$, and the number of elementary operations for each of the d parallel QFTs is polylog(N)—thus, the gate depth is independent of d, while the total gate complexity is linear in d. Additionally, an important component of the complexity comes from performing the unitary U_f, which requires implementing either an evaluation oracle or a probability oracle for the function f. If one has access to an evaluation oracle, the function must be evaluated to precision $O(\varepsilon/\sqrt{d})$. Thus, if function evaluations can be made to precision δ in circuit depth polylog$(d, 1/\delta)$, the overall circuit depth of the quantum gradient estimation algorithm will be polylog$(d, 1/\varepsilon)$, a potentially exponential speedup over the at least $\Omega(d)$ classical query complexity to learn the gradient. In the case that one has access to a probability oracle, a number of oracle calls scaling as $O(\sqrt{d}/\varepsilon)$ must be made.

For some functions, it is possible to classically compute $f(x)$ to precision δ with gate complexity poly$(d, \log(1/\delta))$. This can be turned into a quantum circuit U_f with a comparable gate complexity. For other functions, computing $f(x)$ may be much harder. For example, if $f(x)$ is defined as the output probability of a quantum circuit described by d parameters, then computing $f(x)$ to precision δ might be difficult for a classical computer, and even on a quantum computer, it generally requires $O(1/\delta)$ complexity. However, in this case, implementing a probability oracle is simple, leading to the motivation for the work of [430].

Caveats

Jordan's formulation of the algorithm [587] appears to offer a large quantum speedup by accomplishing in a single quantum query what requires $\Omega(d)$ classical queries. However, this requires a fairly strong access model where one has access to an oracle for computing the value of the function f to high precision. For an exponential speedup to be possible, precision ε must be achievable at cost polylog$(d, 1/\varepsilon)$. Unfortunately, for actual functions f that show up in applications where this is possible, it is often the case that one can classically compute the gradient much more efficiently than simply querying the value of f at many nearby points. Indeed, the "cheap gradient principle" [457, 163] asserts that (in many practical situations) computing the gradient has roughly

https://doi.org/10.1017/9781009639651.022 Published online by Cambridge University Press

the same cost as computing the function itself. This principle limits the scope of application of the large speedup of Jordan's algorithm.

By contrast, [430] shows how the gradient can alternatively be computed using a probability oracle rather than an evaluation oracle, which makes the algorithm compatible with computing gradients in the setting of variational quantum algorithms. However, $O(\sqrt{d}/\varepsilon)$ calls to the oracle are required, which represents a (much less dramatic) *quadratic* speedup compared to the strategy of using the probability oracle to estimate $f(x)$ at many nearby points and subsequently estimating the gradient classically.

Example use cases

- Convex optimization: In convex optimization, local optima are also global optima, and thus a global optimum can be found by greedy methods such as gradient descent. When one can efficiently compute the function f much more cheaply than computing its gradient, the quantum gradient estimation algorithm can give rise to a speedup over classical optimization procedures [47, 245].
- Pure state tomography: Given access to a unitary U that prepares the pure state $|\psi\rangle$, [49] utilizes the gradient estimation algorithm to estimate the amplitudes of $|\psi\rangle$ in the computational basis using an optimal number of queries to U.
- Estimating multiple expectation values: Amplitude estimation can be used to estimate an expectation value to precision ϵ at cost $O(1/\epsilon)$. In [549, 49], it is shown how the gradient estimation algorithm further allows M expectation values to be simultaneously estimated at cost $\widetilde{O}(\sqrt{M}/\epsilon)$ calls to a state preparation unitary, considered the most expensive part of the circuit.
- Computing molecular forces: While ground state energies are the object most often studied in algorithms for quantum chemistry, other interesting quantities such as molecular forces can be related to gradients of molecular energies. Reference [805] studies how the gradient estimation algorithm can be leveraged into a quantum algorithm for computing such quantities.
- Escaping saddle points: Although not the essential ingredient, the gradient estimation algorithm was used in the algorithm of [1081] for escaping saddle points.
- Variational quantum algorithms: Variational quantum algorithms involve optimizing the parameters of a quantum circuit under some cost function. The ability to estimate the gradient of the cost function with respect to the parameters might allow acceleration of this loop.

https://doi.org/10.1017/9781009639651.022 Published online by Cambridge University Press

- Financial market risk analysis: In [950], the quantum gradient estimation subroutine was utilized to compute the Greeks, parameters associated with financial market sensitivity.

Further reading

See [430] for a full discussion of the state of the art with respect to the quantum gradient estimation algorithm.

https://doi.org/10.1017/9781009639651.022 Published online by Cambridge University Press