Quantum Numerical Gradient Estimation - Jordan's Algorithm

Rutvij Shah

December 11, 2022

Abstract

Given a function $f : \mathbb{R}^d \to \mathbb{R}$, which is known to be continuously differentiable, one wants to estimate its gradient, ∇f at a given point $\mathbf{x} = (x_1, x_2, \dots x_d)$ with n bits of precision. Classical algorithms require a minimum of $d^i + 1$ queries to the function for the i^{th} differential, whereas a quantum computer can be shown to require 2^{i-1} queries, i.e. the number of queries is independent of the size of the input, d, of $\mathbf{x}$.

1 Introduction

Numerical gradient estimation is a ubiquitous part of many problems, ranging from neural networks, to dynamical systems, to computational fluid dynamics. For some of these, the objective functions are amongst the most computationally time consuming parts of the solution. In context of such numerical calculations, the query complexity of a function is a natural measure of its time complexity [1]. Thus, an efficient algorithm is one which makes the fewest function evaluations; and our the number of such evaluation calls will define our time complexity.

The Qiskit implementation of the algorithm included with this report (also available on GitHub [1]), is for the case of $d = 1$. It demonstrates the first differential ($i = 1$) of two simple functions, $\sin(x)$ and $10x + e^x + x^2$. The implementation utilizes quantum phase estimation as a subroutine and a modified *unitary matrix* used in a PyQuil implementation of the algorithm within Rigetti's Grove library [2].

2 Solution Summary

The simple algorithm leverages the fact that, in the vicinity of a point $\mathbf{x} = (x_1, x_2, \dots x_d)$, $e^{2\pi\iota\lambda f(\mathbf{x})}$ is periodic, the period is parallel, and inversely proportional to the gradient $\nabla f(\mathbf{x})$. A superposition state is created by discretizing a infinitesimal hyper-rectangle around the domain point, evaluating the function,

[1] `https://github.com/0xrutvij/quantum-gradient-estimation`
[2] `https://github.com/rigetti/grove`

	Classical		Quantum
Derivative	Numerical	Analytical	Numerical
$\mathrm{d}F/\mathrm{d}x$	$d+1$	$\mathcal{O}(1)$	1
$\mathrm{d}^2F/\mathrm{d}x^2$	d^2+1	$\mathcal{O}(d)$	2
$\mathrm{d}^3F/\mathrm{d}x^3$	d^3+1	$\mathcal{O}(d)$	4
$\mathrm{d}^iF/\mathrm{d}x^i$	d^i+1	$\mathcal{O}(d^{\lfloor i/2 \rfloor})$	2^{i-1}

Table 1: Quantum speedup for gradient estimation [4].

rotating the phase in proportion to the value of the function, reversing the oracle call and applying a multidimensional quantum Fourier transform to the bits encoding the aforementioned hyper-rectangle [3].

3 Analysis

Traditionally, for the 1D case, i.e. $d = 1$, gradients are estimated either analytically, or by using the finite difference method. To calculate the first and other higher derivatives, should they exist, the procedure is as follows

For the 1^{st} gradient we need 2 calls to the function $f(x)$.

$$f'(x) = \lim_{h \to \infty} \frac{f(x+h) - f(x)}{h} \tag{1}$$

For the 2^{nd} gradient we need 3 calls to the function $f(x)$.

$$f''(x) = \lim_{h \to \infty} \frac{f(x) - 2f(x-h) + f(x-2h)}{h^2} \tag{2}$$

For the i^{th} gradient we need $i+1$ calls to the function $f(x)$.

$$\frac{\mathrm{d}^i f(x)}{\mathrm{d}x^i} \in \mathcal{O}(i) \tag{3}$$

In the case of $d = 1$, time complexity for numerical gradient calculation is linear in the order of the differential to be calculate, which is improved to constant time by using quantum gradient estimation.

The speedup is starker in the case of $d > 1$, since the growth of classical numerical gradient estimation is exponentially dependent upon d, while quantum version is independent of d, and is exponentially dependent only on the order of derivative needed. The time complexities are shown in Table 1.

4 Jordan's Algorithm

Let

f the function whose gradient is to be estimated

n the bits of precision for the input space.

n_o the bits of precision for the gradient estimate.

d the dimension of the input space of f

h the size of the region where f is approximately linear.

m the size of the region for ∇f

N $= 2^n$

$|k\rangle_i$ floating point values, < 1, represented by n qubits in binary form $0.b_0 \dots b_n$.

$|\mathbf{k}\rangle$ a d-dimensional vector of all the $|k\rangle$'s

$|\psi\rangle$ the ancilla register.

U an *unitary transform* used to encode the gradient's phase.

The algorithm considered takes as its input d fixed point binary strings, each of length n, along with an ancilla register ψ, which encodes the phase. The size of the ancilla register is n_o. In the implemented version of the algorithm, $n_o = n$.

The input registers are initialized based on the value at which the gradient is to be estimated, here we consider it to be $|\mathbf{0}\rangle = (|0\rangle_1, \dots, |0\rangle_d)$. While the ancilla register is initialized to $|\mathbf{1}\rangle$. The Hadamard transform is applied to each of the input registers followed by controlled unitary operations as used in the Quantum Phase Estimation algorithm.

The unitary transform encodes $f(h) \coloneqq \frac{f(x+h)-f(x-h)}{2h}$, within the phase as follows, here j is the position index of the qubit within the register.

Lastly, IQFT is applied to the input registers and the encoded phase kicks back, causing the output gradient to be encoded within the *input* registers, in Jordan's original algorithm the output is encoded within the *ancilla* register.

This affects the reversibility of our gradient estimation function and thus 'un-computation' is required to use this algorithm as a subroutine. Details of the reversible versions of the algorithm, are discussed in [1], [2], [3] and [4].

$$U^{2^j} = \begin{bmatrix} e^{2\pi\iota 2^j f(h)} & 1 \\ 1 & e^{2\pi\iota 2^j f(h)} \end{bmatrix}$$

i.e.

$|\mathbf{k}\rangle = |\mathbf{0}\rangle$

$|\psi\rangle = |\mathbf{1}\rangle$

$U = \begin{bmatrix} e^{2\pi\iota f(h)} & 1 \\ 1 & e^{2\pi\iota f(h)} \end{bmatrix}$

The algorithm starts in equal superposition of d registers of n qubits each, after the application of $H^{\otimes d}$

$$\frac{1}{\sqrt{N^d}} \sum_{\mathbf{k}} |\mathbf{k}\rangle \tag{4}$$

Together with the ancilla register the state is

$$\frac{1}{\sqrt{N^d N_o}} \sum_{\mathbf{k}} |\mathbf{k}\rangle \sum_{\psi} e^{2\pi\iota f(h)/N_o} |\psi\rangle \tag{5}$$

After applying the unitary transforms, for sufficiently small h, the state is

$$\frac{1}{\sqrt{N^d N_o}} \sum_{\mathbf{k}} e^{2\pi\iota \frac{N}{mh}(f(\mathbf{0})+\frac{h}{N}(\mathbf{k}-\frac{N}{2})\cdot\nabla f)} |\mathbf{k}\rangle \sum_{\psi} e^{2\pi\iota f(h)/N_o} |\psi\rangle \tag{6}$$

Ignoring the global phase, the input registers are now approximately in the state

$$\frac{1}{\sqrt{N^d}} \sum_{k_1 \ldots k_d} e^{\frac{2\pi\iota}{m}(k_1 \frac{\partial f}{\partial x_1}+\ldots+k_d \frac{\partial f}{\partial x_d})} |k_1\rangle \ldots |k_d\rangle \tag{7}$$

This is a product state:

$$\frac{1}{\sqrt{N^d}} \left(\sum_{k_1} e^{\frac{2\pi\iota}{m} k_1 \frac{\partial f}{\partial x_1}} \right) \ldots \left(\sum_{k_d} e^{\frac{2\pi\iota}{m} k_d \frac{\partial f}{\partial x_d}} \right) \tag{8}$$

Inverse Fourier transform each of the input registers, obtaining the required gradient estimate.

$$\left| \frac{N}{m} \frac{\partial f}{\partial k_1} \right\rangle \left| \frac{N}{m} \frac{\partial f}{\partial k_2} \right\rangle \ldots \left| \frac{N}{m} \frac{\partial f}{\partial k_d} \right\rangle \tag{9}$$

5 Conclusion

Jordan's algorithm demonstrates an approach to estimating numerical gradients using a quantum approach. Gradient evaluation for real-valued multivariate functions is shown to be a constant time operation, and for higher order derivatives, dependent only on the order of the derivative, i.e. independent of the input size. This is an improvement over the linear complexity for classical gradient estimation approaches for higher dimensions which further becomes exponential in the size of the input for higher order derivatives.

Further work from the perspective of implementation, involves implementing the general d-dimensional version of the algorithm using multiple registers, and making the computation reversible to ensure that the algorithm can be used as a subroutine without incurring any previous input-dependent global phase.

References

[1] Stephen P. Jordan. Fast quantum algorithm for numerical gradient estimation, 2004, Phys. Rev. Lett. 95, 050501 (2005); arXiv:quant-ph/0405146. DOI: 10.1103/PhysRevLett.95.050501.

[2] András Gilyén, Srinivasan Arunachalam and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation, 2017, In Proceedings of the 30th ACM-SIAM Symposium on Discrete Algorithms (SODA 2019), pp. 1425-1444; arXiv:1711.00465. DOI: 10.1137/1.9781611975482.87.

[3] David Bulger. Quantum computational gradient estimation, 2005; arXiv:quant-ph/0507109.

[4] Ivan Kassal and Alán Aspuru-Guzik. Quantum Algorithm for Molecular Properties and Geometry Optimization, 2009, J. Chem. Phys. 131, 224102 (2009); arXiv:0908.1921. DOI: 10.1063/1.3266959.