The Relevance of Stochastic Gradient Descent within Quantum Offline Optimisation

Joseph MacManus

July 2019

Abstract

In the context of machine-learning and optimisation, evaluations of the objective function often prove to be very expensive in nature and so are to be avoided. If we suppose our objective function is of the form of a sum, then methods such as *stochastic gradient descent* get around this problem by only evaluating a small number of summands at any point. Quantum gradient evaluation and descent are a topic that has been explored in a lot of depth recently ([Jor05, GAW19, KP17]), with near optimal speed-ups in existence. We shall be considering offline optimisation problems of this form within the setting of quantum computation, and applying the optimality and oracle conversion results of [GAW19] to argue that the benefits of stochastic gradient descent are almost entirely lost within certain contexts.

Contents

1 Introduction

1.1 Optimisation and gradient descent

We shall deal with the problem of optimising a smooth objective function $f : \mathbb{R}^d \to \mathbb{R}$, that is, finding $\mathbf{x} \in \mathbb{R}^d$ such that $f(\mathbf{x})$ is minimised. We suppose that f is of the form

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} f_i(\mathbf{x}).$$

Furthermore, we suppose that each f_i corresponds to some loss function ℓ evaluated at the i-th example in some data set. This scenario is often found within machine-learning, for example. Therefore, our objective function is thus of the form

$$f(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \ell(\mathbf{x}, \hat{\mathbf{x}}_i).$$

The time-honoured optimisation method used in cases like this is *gradient descent*. We choose some small step-size η and iterate

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla f(\mathbf{x})$$

until we reach a local minimum. If it is a global minimum we desire, then we may repeat this process from a collection of randomly chosen starting points and take the minimum of the final resting places. There is little hope for a speed-up of the iteration itself, however a lot of work has been done on numerically estimating the gradient via quantum methods.

1.2 Notation

We will be using $\omega_M^a := e^{2\pi i a/M}$ to represent the a-th power of the M-th root of unity. To ease notation, we have that $+_M$ and $-_M$ represent addition and subtraction modulo M respectively. $\tilde{\mathcal{O}}(g)$ represents the set of functions which grow asymptotically at most as fast as g, ignoring logarithmic factors. We use $[n]$ to represent the set $\{1, \ldots, n\}$. When dealing with vector norms, we use $\|\cdot\|$ to refer to the standard Euclidean norm, and $\|\cdot\|_\infty$ to refer to the component-wise maximum norm. Furthermore, we sometimes abbreviate the partial derivative operator $\frac{\partial}{\partial \mathbf{x}_i} = \partial_i$, and also omit function arguments when these are clear from context.

2 Classical Stochastic Gradient Descent

Before we take a look at any quantum methods, we will motivate and describe the algorithm we are discussing classically.

2.1 Motivation

As discussed in the introduction, our goal is to optimise some objective function $f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\mathbf{x}, \hat{\mathbf{x}}_i)$, where each $\hat{\mathbf{x}}_i$ corresponds to the i-th piece of data within some data-set (consider examples within machine learning), and ℓ is some loss function. To ease notation, we will let $\ell_i(\mathbf{x}) = \ell(\mathbf{x}, \hat{\mathbf{x}}_i)$. If we were to classically evaluate f exactly, this would quite clearly require exactly N evaluations of ℓ. Considering that N often takes very large values in many optimisation settings, this can be simply infeasible.

We look towards approximating f via some kind of sampling method. It has been shown [CEG95] that in order to classically calculate an ε-accurate mean with probability δ, one must take $\Omega(1/\varepsilon^2 \log \delta)$ samples, and thus a similar number of queries to ℓ would be required to approximate f. Finally consider that in order to evaluate the gradient ∇f numerically, one would need to make $\mathcal{O}(d)$ evaluations of f. Overall, this leads us to the informal conclusion that one gradient calculation of f within this setting would have an overall complexity of $\mathcal{O}(dN)$, or similarly $\mathcal{O}(d/\varepsilon^2 \log \delta)$ if an approximation is acceptable. This can become rather costly, and given that data within the data set can often contain a lot of redundancy, full evaluation of f during optimisation is often avoided as much as possible.

2.2 The algorithm

Stochastic gradient descent (SGD) is a variation on normal (batch) gradient descent, that does not rely on costly full function evaluations. At each iterations, we select some random $i \in [N]$, and approximate $\nabla f \approx \nabla \ell_i$. We then proceed as normal, iterating

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \nabla \ell_i(\mathbf{x}),$$

beginning at some randomly chosen starting point. This approximation seems rather unfounded at a glance - however it has indeed been shown to almost surely converge with only a small set of restrictions placed on the objective function [Bot98]. Convergence is certainly noisy, and this can be mitigated at the cost of efficiency by taking small sample means as your approximation instead (known as mini-batch gradient descent). Stochastic gradient descent is a fairly standard tool used in scenarios such as machine learning [Bot10], as well as plenty of other situations.

2.3 Online vs offline optimisation

Within this report we shall be considering only *offline* optimisation. Informally, this means that the data set being used to evaluate (and define) f is all available at once, and unchanging. In particular, we have full access to data, and thus full evaluation of f is indeed a possibility. The alternative,

online optimisation, supposes that the data set is actually constantly growing while the optimisation takes place. This means that evaluation of f itself is effectively impossible during the optimisation progress, as the defining data set is not complete. Scenarios like this are where stochastic gradient descent often finds itself being used.

This report shall be assuming that the relevant data-set being used for optimisation is available in full and unchanging, to allow for consideration and evaluation of f itself. Therefore we are going to be working within the offline format.

3 Function access models

Before we look properly at quantum gradient computation, we need to discuss how we are going to be accessing our function - as of course every action on a quantum state must be unitary and thus reversible (whereas our objective functions certainly need not be). Within the context of quantum computing, there are actually several different ways that a function may be called. Here we will discuss in detail the three main ways of accessing some black-box function, also known as *access models*.

3.1 Probability access

To begin, we will consider probability access, and define what a *probability oracle* is.

Definition 3.1 (Probability oracle)**.** *Let $g : X \to [0, 1]$, where $\{|x\rangle : x \in X\}$ forms an orthonormal basis of the Hilbert space $\mathcal{H}$, and let $\mathcal{H}_A$ be an ancilla register on $n > 0$ qubits. Then an operator $U_g : \mathcal{H} \otimes \mathcal{H}_A \to \mathcal{H} \otimes \mathcal{H}_A$ is called a probability oracle for g if*

$$U_g |x\rangle |\vec{0}\rangle = |x\rangle \left(\sqrt{1 - g(x)} |\psi_0\rangle |0\rangle + \sqrt{g(x)} |\psi_1\rangle |1\rangle \right)$$

for some arbitrary $n - 1$ qubit states $|\psi_0\rangle$, $|\psi_1\rangle$.

Essentially, this within this model our objective function corresponds precisely to the probability of a certain outcome being observed upon measurement (in particular, the probability of seeing $|1\rangle$ when measuring the final qubit). Indeed, given a classical description of a function, standard methods do exist for constructing an oracle of this form[1].

3.2 Phase access

The next access model we consider is access via a *phase oracle*. We also define fractional phase queries.

[1] Citation needed.

Definition 3.2 (Phase oracle). *Given a function $g : X \to [0,1]$, and given that $\{|x\rangle : x \in X\}$ forms an orthonormal basis of the Hilbert space $\mathcal{H}$, then the corresponding phase oracle $O_g : \mathcal{H} \otimes \mathcal{H}_A \to \mathcal{H} \otimes \mathcal{H}_A$ allows queries of the form*

$$O_g |x\rangle |\vec{0}\rangle = e^{ig(x)} |x\rangle |\vec{0}\rangle ,$$

where $\mathcal{H}_A$ is an ancilla register. Furthermore if fix some $r \in [-1,1]$, then $O_{rg} : \mathcal{H} \otimes \mathcal{H}_A \to \mathcal{H} \otimes \mathcal{H}_A$ is known as a fractional-query phase oracle for g if it allows queries of the form

$$O_{rg} |x\rangle |\vec{0}\rangle = e^{irg(x)} |x\rangle |\vec{0}\rangle ,$$

for all $x \in X$.

The authors of [GAW19] showed that a probability oracle is capable of simulating a phase oracle, and vice versa, with only logarithmic overhead. We restate this result here.

Theorem 3.1 (Converting between probability and phase oracles, [GAW19]). *Suppose $g : X \to [0,1]$ is given by access to a probability oracle U_g which makes use of n auxiliary qubits. Then we can simulate an ε-approximate phase oracle using $\mathcal{O}(\log(1/\varepsilon)$ queries to U_g, and an additional a auxiliary qubits, where $a = \mathcal{O}(\log\log(1/\varepsilon))$.*

Similarly, suppose $g : X \to [\delta, 1-\delta]$ is given by access to a phase oracle O_g. Then, we can construct an ε-approximate probability oracle for g using just $\mathcal{O}(\log(1/\varepsilon)/\delta)$ queries to O_g.

What this shows is that the two access models are more-or-less equivalent in power. The implications of this theorem to us are that phase oracle access to ℓ is once again equivalent to access to f itself. A side-effect of this theorem is given a phase oracle for some function, one can simulate a fractional query with only logarithmic overhead, since obtaining a fractional query from a probability oracle is indeed trivial.

3.3 Binary access

We now define the strongest form of function access, and the closest to its classical counterpart, the binary oracle.

Definition 3.3 (Binary oracle). *Let $\mathcal{H}_A$ be an ancilla register on m qubits, then given a function $g : X \to \mathbb{R}$, we call an operator $B_g^\eta : \mathcal{H} \otimes \mathcal{H}_A \to \mathcal{H} \otimes \mathcal{H}_A$ an η-accurate binary oracle for g if $\mathcal{H} = \{|x\rangle : x \in X\}$ and*

$$B_g^\eta |x\rangle |0\rangle = |x\rangle |g'(x)\rangle ,$$

where $g'(x)$ is a fixed point representation of a number satisfying $|g(x) - g'(x)| \leq \eta$ for all $x \in X$.

This model is usually modified to add the result to the ancilla register modulo 2^m. It is then possible for a binary oracle to simulate its phase-based counterpart with a single query, via the "phase kickback" trick mentioned in [Jor05].

Lemma 3.2. *For all $x \in \mathbb{N}$, the state*

$$QFT_M^{-1} |1\rangle = \frac{1}{\sqrt{M}} \sum_a \omega_M^{M-a} |a\rangle$$

is an eigenstate for the operation $|b\rangle \mapsto |b +_M x\rangle$ with eigenvalue ω_M^x.

Proof. Simply observe

$$\begin{aligned} \frac{1}{\sqrt{M}} \sum_a \omega_M^{-a} |a +_M x\rangle &= \frac{\omega_M^x}{\sqrt{M}} \sum_a \omega_M^{-(a+x)} |a +_M x\rangle \\ &= \frac{\omega_M^x}{\sqrt{M}} \sum_a \omega_M^{-a} |a\rangle \end{aligned}$$

and we are done. □

Theorem 3.3. *Given a function $g : X \to [0, 1]$ with access given by a (modified) binary oracle B_g^η with an m-qubit ancilla register, which allows queries of the form*

$$B_g^\eta |x\rangle |b\rangle = |x\rangle |b +_M \frac{g'(x)M}{2\pi}\rangle ,$$

where $M = 2^m$ and $|g(x) - g'(x)| \leq 2\pi\eta$, then we can construct an $2\pi\eta$-accurate phase oracle for g using a single query to B_g^η.

Proof. If we write the binary integer 1 to the second register and apply the inverse quantum Fourier transform to it before applying our binary oracle, by Lemma 3.2 we have that our resulting state will be equal to

$$e^{ig'(x)} |x\rangle (QFT_M^{-1} |1\rangle).$$

The accuracy claim also follows immediately from the accuracy of the binary oracle. □

Finally, it is worth noting that one can convert a probability oracle to a binary oracle via the well-documented trick of amplitude estimation [BHMT02]. In order to construct a ε-accurate binary oracle this way, one would need to make $\mathcal{O}(1/\varepsilon)$ queries to the relevant probability oracle. As discussed in [GAW19], binary oracles are expensive to implement and, much like is done in [Jor05], are usually converted to phase oracles anyway at some point within the process. Phase oracles themselves are generally much cheaper to implement than their binary equivalents, and because of this we will not be considering binary oracular access for the rest of this report, outside of discussion of Jordan's work.

4 Quantum gradient estimation

In this section we will discuss and summarise the existing methods for quantum gradient calculation and their optimality, but first we will quickly discuss classical gradient estimation.

Suppose that have a differentiable function $g : \mathbb{R}^d \to \mathbb{R}$, and given some $\mathbf{x} \in \mathbb{R}^d$, we wish to estimate $\nabla g(\mathbf{x})$. Classically this calculation would usually numerically be calculated via use of some kind of *finite difference* formula to calculate the partial derivatives one at a time. For example, given some small enough $\delta > 0$, we have

$$\frac{\partial}{\partial \mathbf{x}_i} g(\mathbf{x}) \approx \frac{g(\mathbf{x} + \delta \mathbf{e}_i) - g(\mathbf{x})}{\delta}.$$

Using this formula or similar, one can then calculate an approximation of the gradient using a $\mathcal{O}(d)$ evaluations of g. In simple terms, quantum computers are capable of calculating each of these partial derivatives in parallel.

4.1 Jordan, 2005

Stephen Jordan [Jor05] considered the problem of quantum gradient computation and published his algorithm in 2005. He considers a function f given with arbitrary precision by a bit-oracle, and successfully applies a central-difference approximation to numerically calculate the gradient with just a single query to the aforementioned oracle. Jordan makes use of the binary oracle to simulate (a power of) the phase oracle for f at the start of the algorithm via phase kickback (Lemma 3.2), and it is for this reason as well as discussion in Section 3.3 that we will actually be considering phase access instead of the binary access originally assumed. This same change is made within the discussion of the algorithm in [GAW19].

Before discussing Jordan's algorithm, we will set up some notation. Let G_n be a set of $N = 2^n$ small real numbers evenly spaced about 0, and let

$$\mathcal{H}_{\text{in}} = \text{span}\{|\delta\rangle : \delta \in G_n\}$$

be an n-qubit input register. Formally what we have set up here is some fixed-point binary representation, however I have omit the fine details of this for brevity. We also define a slight variant on the quantum Fourier transform, which maps

$$QFT_{G_n} |\delta\rangle = \frac{1}{\sqrt{N}} \sum_{k \in G_n} e^{2\pi i \delta k} |k\rangle$$

for all $\delta \in G_n$. Gilyén *et al.* showed that this transformation can be constructed from a single query to the usual QFT, and $2n$ single-qubit gates[2].

[2]Technically they make this claim about a slightly different variant of the QFT, however I believe the same idea holds via a similar argument.

We begin with d copies of $\mathcal{H}_{\text{in}}$ as our input registers, each initialised in a uniform superposition such that our system is in the state

$$|\psi\rangle := \frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{\delta} \in G_n^d} |\boldsymbol{\delta}\rangle \,,$$

where $|\boldsymbol{\delta}\rangle = |\delta_1\rangle \dots |\delta_d\rangle$ is used to ease notation. Next, we apply the (fractional) phase oracle $O_{2\pi Sf}$, where $S > 0$ is some scaling factor such that $\|S\nabla f(\mathbf{0})\|_\infty \leq \sup G_n$. Following this the system is in the following state:

$$O_{2\pi Sf} |\psi\rangle = \frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{\delta} \in G_n^d} e^{2\pi i S f(\boldsymbol{\delta})} |\boldsymbol{\delta}\rangle \,.$$

We now remind ourselves that for any $\boldsymbol{\delta} \in \mathbb{R}^d$, such that $\|\boldsymbol{\delta}\|$ is small, then it is a fact from calculus that

$$f(\mathbf{x} + \boldsymbol{\delta}) = f(\mathbf{x}) + \nabla f \cdot \boldsymbol{\delta} + \mathcal{O}(\|\boldsymbol{\delta}\|^2),$$

so for small enough $\boldsymbol{\delta}$, we have that f is approaches affine linearity. Using this (and ignoring global phase), we can see that our system is approximately

$$\begin{aligned} \frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{\delta} \in G_n^d} e^{2\pi S f(\boldsymbol{\delta})} |\boldsymbol{\delta}\rangle &\approx \frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{\delta} \in G_n^d} e^{2\pi i S (f(\mathbf{0}) + \nabla f \cdot \boldsymbol{\delta})} |\boldsymbol{\delta}\rangle \\ &\equiv \frac{1}{\sqrt{N^d}} \sum_{\boldsymbol{\delta} \in G_n^d} e^{2\pi i S (\delta_1 \partial_1 f + \dots + \delta_d \partial_d f)} |\boldsymbol{\delta}\rangle \,, \end{aligned}$$

which is precisely the product state

$$\begin{aligned} &\left(\frac{1}{\sqrt{N}} \sum_{\delta_1} e^{2\pi i S \delta_1 \partial_1 f} |\delta_1\rangle \right) \cdots \left(\frac{1}{\sqrt{N}} \sum_{\delta_d} e^{2\pi i S \delta_d \partial_d f} |\delta_d\rangle \right) \\ &\qquad = (QFT_{G_n} |\partial_1 f\rangle) \cdots (QFT_{G_n} |\partial_d f\rangle). \end{aligned}$$

From here, one can simply apply $QFT_{G_n}^{-1}$ to each register individually to receive a classical description of the gradient.

4.2 Gilyén *et al.*, 2017

The authors of [GAW19] build upon this algorithm, and we summarise their algorithm here. First, they remark that the assumed access model is stronger than what might be available in practice. They suppose that the

objective function corresponds to the probability of some outcome and assume some smoothness conditions (the details of which we will leave until Appendix A). They improve Jordan's algorithm by applying a higher-order central-difference approximation. The exact results are very technical and involve intricate higher-order calculus, so we will only summarise them here. We restate several of their results, firstly their result on fast polynomial gradient evaluation.

Theorem 4.1 (Polynomial gradient evaluation, [GAW19]). *Suppose that the function $p : [-R, R]^d \to \mathbb{R}$ is a multivariate polynomial of degree k, with access given by a phase oracle O_p. Then there exists an algorithm which can find $\mathbf{g} \in \mathbb{R}^d$ such that $\|\mathbf{g} - \nabla p(\mathbf{0})\|_\infty \leq \varepsilon$ using $\mathcal{O}(k \log(d)/\varepsilon)$ queries to O_p.*

Theorem 4.1 also implies that if $k = \mathcal{O}(\log d)$, then we achieve an exponential speed-up in gradient evaluation. The authors also provide a more general result.

Theorem 4.2 (Improved gradient computation, [GAW19]). *Suppose that $g : \mathbb{R}^d \to \mathbb{R}$ is an analytic function with bounded partial derivatives at $\mathbf{0}$ (see Appendix A) and access given by a phase oracle O_g, then there exists an algorithm that outputs a vector $\mathbf{g} \in \mathbb{R}^d$ such that $\|\mathbf{g} - \nabla g(\mathbf{0})\|_\infty \leq \varepsilon$ with high probability, using $\tilde{\mathcal{O}}(\sqrt{d}/\varepsilon)$ queries to the oracle.*

They then go on to show the following optimality result regarding their algorithm.

Theorem 4.3 (Lower bound on quantum gradient computation, [GAW19]). *Suppose $\mathcal{A}$ is an algorithm that, given access to any function $g : \mathbb{R}^d \to \mathbb{R}$ with bounded partial derivatives (see Appendix A) via some phase oracle O_g, will output a vector $\mathbf{g}$ such that $\|\mathbf{g} - \nabla g(\mathbf{0})\|_\infty \leq \varepsilon$ with probability $p > \frac{2}{3}$, then $\mathcal{A}$ must make $\Omega(\sqrt{d}/\varepsilon)$ queries to U_ℓ. A similar result holds if we suppose probability access instead.*

By considering the oracle-conversion theorems of Section 3, we have that a similar bound also holds for both binary access and probability access to f. In the next section, we will apply this result to argue that stochastic gradient descent must lose its speed advantage within the quantum setting.

4.3 Cornelissen, 2018

In his Masters thesis, Cornelissen [Cor18] improves on the work of Gilyén *et al.* by showing that their algorithm is in fact optimal for an even larger class of functions.

5 The relevance of SGD within the quantum setting

In this section we will discuss the main idea of this report.

5.1 Access to the loss function

First, we will discuss access to ℓ itself. Suppose our loss function ℓ maps to the interval $[0, 1]$, and access is given via a probability oracle, which maps

$$U |\mathbf{x}\rangle |\hat{\mathbf{x}}\rangle |0\rangle = |\mathbf{x}\rangle |\hat{\mathbf{x}}\rangle \left(\sqrt{1 - \ell(\mathbf{x}, \hat{\mathbf{x}})} |0\rangle + \sqrt{\ell(\mathbf{x}, \hat{\mathbf{x}})} |1\rangle \right),$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ are given in some fixed-point binary representation. We will slightly modify this oracle, making use of the *Q-RAM* model [GLM08, KP17]. Within this model, we first suppose without loss of generality that N is a power of 2, and then suppose that our data-set is stored within some data-structure which allows queries of the form

$$|i\rangle |0\rangle \mapsto |i\rangle |\hat{\mathbf{x}}_i\rangle$$

to be made in superposition in polylogarithmic time, where $\hat{\mathbf{x}}_i$ corresponds to the i-th element of our data set. Using this, we can modify the oracle to be of the form

$$U_\ell |\mathbf{x}\rangle |i\rangle |0\rangle = |\mathbf{x}\rangle |i\rangle \left(\sqrt{1 - \ell(\mathbf{x}, \hat{\mathbf{x}}_i)} |0\rangle + \sqrt{\ell(\mathbf{x}, \hat{\mathbf{x}}_i)} |1\rangle \right).$$

It is also not too hard to see that this modification applies to the other access models similarly.

5.2 Equivalence of loss function and objective function access

Classically, it is easy to see that access to ℓ is actually a much weaker assumption than supposing access to f itself. However, we will now see that in the quantum setting that this is no longer true.

Theorem 5.1. *Given access to the loss function ℓ via a probability oracle U_ℓ, we can construct a probability oracle for f using just a single query to U_ℓ.*

Proof. Consider applying the Hadamard transform H_n to the second register, then we have

$$\begin{aligned}
U_\ell |\mathbf{x}\rangle (H_n |0\rangle) |0\rangle &= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} U_\ell |\mathbf{x}\rangle |i\rangle |0\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |\mathbf{x}\rangle |i\rangle \left(\sqrt{1 - \ell_i} |0\rangle + \sqrt{\ell_i} |1\rangle \right) \\
&= \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} \sqrt{1 - \ell_i} |\mathbf{x}\rangle |i\rangle |0\rangle + \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sqrt{\ell_j} |\mathbf{x}\rangle |j\rangle |1\rangle \\
&:= |\psi_0\rangle |0\rangle + |\psi_1\rangle |1\rangle
\end{aligned}$$

where $\ell_i := \ell(\mathbf{x}, \hat{\mathbf{x}}_i)$ to ease notation. We have that the probability of seeing a $|1\rangle$ when measuring the third register is then equal to exactly

$$\langle \psi_1 | \psi_1 \rangle = \frac{1}{N} \sum_{j=0}^{N-1} \ell_j = f(\mathbf{x}),$$

as required. □

Therefore it follows that this seemingly weaker access model is actually no weaker than supposing access to f itself in a similar fashion. We now generalise this slightly to allow for other access models.

Corollary 5.2. *Given access to our loss function ℓ via either phase or probability oracles, we can simulate either a phase or probability oracle for f using at most a logarithmic number of queries.*

Proof. This follows immediately from Theorem 5.1, combined with the result of Theorem 3.1. □

5.3 Implications for SGD

The implications to SGD should be fairly clear. Given access to the loss function ℓ by either probability or phase oracle, we can easily simulate access to f within minimal effort. It follows immediately that calculating the gradient of ℓ_i for some i, and calculating the gradient of f itself are both about as hard as each other now - a far cry from what was true in the classical realm.

It is rather difficult to completely formalise this result right now however, due to the limitations of what optimality results exist. The work done in [GAW19] and [Cor18] provides bounds on generic gradient computation given some smoothness conditions placed on the inputs - however restricting these conditions even further does in fact give rise to faster algorithms, as was seen in Theorem 4.1. Therefore it might be fair to argue that ℓ_i may have some smoothness properties that do not exhibit themselves in f, making it true that $\nabla \ell_i$ is in fact faster to compute than ∇f. However, I would argue that it would be unlikely that this is the case, due to the fact that differential operators tend to be linear, and so if ℓ_i happens to satisfy a smoothness condition then it is likely that f also does. This can be seen easily by applying the smoothness condition of [GAW19] and using the linearity of the partial derivative to demonstrate that this condition does indeed still hold for f (see Claim A.1). Furthermore, a simpler example is that if ℓ_i happens to be a multivariate polynomial in $\mathbf{x}$, then it is clear that f will also be.

6 Closing remarks

The main benefit of stochastic gradient descent is its per-iteration complexity, at the cost of noisy optimisation. However, we have shown that setting up a situation where classical SGD would be justified, that within the quantum setting its key benefit is in fact lost, and one may as well focus on batch gradient descent and its variants. I'm sure that in plenty of scenarios I haven't considered, SGD still has its benefits - online optimisation being the main one.

This report itself has been more of an exploration into the techniques used in quantum gradient calculation, and function evaluation in general, as well as just an application of some known results to a less general setting.

References

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[Bot98] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, 1995.

[Cor18] Arjan Cornelissen. Quantum gradient estimation and its application to quantum reinforcement learning. 2018.

[GAW19] András Gilyén, Srinivasan Arunachalam, and Nathan Wiebe. Optimizing quantum optimization algorithms via faster quantum gradient computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1425–1444. Society for Industrial and Applied Mathematics, 2019.

[GLM08] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[Jor05] Stephen P Jordan. Fast quantum algorithm for numerical gradient estimation. *Physical review letters*, 95(5):050501, 2005.

[KP17] Iordanis Kerenidis and Anupam Prakash. Quantum gradient descent for linear systems and least squares. *arXiv preprint arXiv:1704.04992*, 2017.

A Smoothness conditions

Find here a more detailed discussion and summary of the smoothness conditions assumed throughout this report. Given some analytic function $g : \mathbb{R}^d \to \mathbb{R}$, let $\partial_i g(\mathbf{x}) = \frac{\partial}{\partial x_i} g(\mathbf{x})$, and for any $k \in \mathbb{N}$, $\alpha = (\alpha_1, \ldots, \alpha_k) \in [d]^k$, let

$$\partial_\alpha g(\mathbf{x}) = \partial_{\alpha_1} \ldots \partial_{\alpha_k} g(\mathbf{x}).$$

The following result shows that if our loss function is satisfies the smoothness condition discussed in [GAW19], we have that the objective function also satisfies the same condition.

Claim A.1. *Let $0 < \varepsilon \leq c$ be real constants, and fix some $\boldsymbol{x} \in \mathbb{R}^d$. Suppose that for all $i \in [N]$ the function $\ell_i : \mathbb{R}^d \to \mathbb{R}$ is analytic, and that for every natural number k, and $\alpha \in [d]^k$, we have that*

$$|\partial_\alpha \ell_i(\boldsymbol{x})| \leq c^k k^{\frac{k}{2}},$$

then we have that f also satisfies the same condition.

Proof. We apply the linearity of ∂_α. Observe that

$$\begin{aligned} |\partial_\alpha f(\mathbf{x})| &= \left| \frac{1}{N} \sum_i \partial_\alpha \ell_i(\mathbf{x}) \right| \\ &\leq \frac{1}{N} \sum_i |\partial_\alpha \ell_i(\mathbf{x})| \\ &\leq c^k k^{\frac{k}{2}}, \end{aligned}$$

and we are done. □

In fact it's not too hard to see that this claim generalises to more-or-less any bound on the partial derivatives.