Stochastic gradient descent for hybrid quantum-classical optimization

Ryan Sweke[1], Frederik Wilde[1], Johannes Jakob Meyer[1], Maria Schuld[2,3], Paul K. Fährmann[1], Barthélémy Meynard-Piganeau[4], and Jens Eisert[1,5,6]

[1]Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, 14195 Berlin, Germany

[2]Xanadu, 777 Bay Street, Toronto, Ontario, Canada

[3]Quantum Research Group, University of KwaZulu-Natal, 4000 Durban, South Africa

[4]Department of Physics, Ecole Polytechnique, Palaiseau, France

[5]Helmholtz Center Berlin, 14109 Berlin, Germany

[6]Department of Mathematics and Computer Science, Freie Universität Berlin, D-14195 Berlin

Within the context of hybrid quantum-classical optimization, gradient descent based optimizers typically require the evaluation of expectation values with respect to the outcome of parameterized quantum circuits. In this work, we explore the consequences of the prior observation that estimation of these quantities on quantum hardware results in a form of *stochastic* gradient descent optimization. We formalize this notion, which allows us to show that in many relevant cases, including VQE, QAOA and certain quantum classifiers, estimating expectation values with k measurement outcomes results in optimization algorithms whose convergence properties can be rigorously well understood, for any value of k. In fact, even using single measurement outcomes for the estimation of expectation values is sufficient. Moreover, in many settings the required gradients can be expressed as linear combinations of expectation values – originating, e.g., from a sum over local terms of a Hamiltonian, a parameter shift rule, or a sum over data-set instances – and we show that in these cases k-shot expectation value estimation can be combined with sampling over terms of the linear combination, to obtain "doubly stochastic" gradient descent optimizers. For all algorithms we prove convergence guarantees, providing a framework for the derivation of rigorous optimization results in the context of near-term quantum devices. Additionally, we explore numerically these methods on benchmark VQE, QAOA and quantum-enhanced machine learning tasks and show that treating the stochastic settings as hyper-parameters allows for state-of-the-art results with significantly fewer circuit executions and measurements.

1 Introduction

Hybrid quantum-classical optimization with parameterized quantum circuits [1] provides a promising approach for understanding and exploiting the potential of *noisy intermediate-scale quantum (NISQ)* devices [2]. In this approach, which includes large classes of well studied methods like *variational quantum eigensolvers* (VQE) [3], the *quantum approximate optimization algorithm* (QAOA) [4] and *quantum classifiers* [5, 6], a classical optimization scheme is utilized to update the parameters of a hybrid quantum-classical model, and developing and understanding optimization techniques tailored to this setting is of natural importance [7]. While a variety of gradient-free optimization methods have been proposed and studied [8], in this work we will be concerned with gradient descent type optimizers. This focus is motivated by the fact that for highly over-parameterized classical models (such as modern neural networks) gradient-based optimization offers many advantages over gradient-free methods [9], and as a result developing and analyzing

Ryan Sweke: These authors contributed equally to this work.

Frederik Wilde: These authors contributed equally to this work.

such methods designed specifically for the quantum-classical setting is crucial, particularly as the computational power of available near-term quantum devices increases.

In this hybrid quantum-classical setting, since part of the hybrid computation is executed by a quantum circuit, the optimized loss is generically a function of the expectation values of quantum observables. While zeroth-order methods can be immediately applied to obtain an approximation to the gradient from evaluations of the loss function [10, 11], there also now exist a variety of strategies to directly evaluate the gradient, either via distinct quantum algorithms [12, 13], or through the measurement of suitable observables with respect to states generated by the parameterized model [14–16]. As an example of the latter, it has recently been shown that the partial derivatives required for gradient descent can be exactly expressed as linear combinations of the same expectation values appearing in the loss function, but with respect to states generated from a shift in the tunable parameters of the parameterized quantum circuit – a strategy called the 'parameter shift rule' [15, 16]. However, as an infinite number of measurements is required for the exact evaluation of a particular expectation value, it is not possible to implement *exact* gradient descent in this hybrid quantum-classical setting, even when the exact gradient can be written as a function of expectation values. As a result, previous approaches have typically used large numbers of measurements to estimate expectation values as accurately as possible [17–19], and the necessary circuit repetitions represent a significant overhead for the implementation of gradient based optimizers.

Recently however it has been observed that using a finite number of measurements for the evaluation of gradients effectively results in the implementation of *stochastic* gradient descent [11]. Stochastic gradient descent (SGD) optimization works by replacing the exact partial derivative at each optimization step with an *estimator* of the partial derivative, and when the estimator is unbiased it is often possible to prove rigorous convergence guarantees in appropriately simplified settings [20, 21]. Additionally, SGD is the method of choice for the vast majority of large-scale machine learning models, where it has been found to offer advantages over exact gradient descent, such as faster evaluation of the gradient, faster convergence, and avoidance of local minima [22–25]. Given the natural connection between SGD and hybrid quantum-classical optimization, Harrow and Napp [11] provide explicit algorithms for the construction of unbiased estimators for the gradient of typical loss functions arising in the context of hybrid quantum-classical optimization, both by exploiting single shot measurements for the estimation of expectation values, and by importance sampling single terms of linear combinations of expectation values, which arise naturally from the use of local Hamiltonians to construct loss functions. Additionally, using these estimators they are able to generalize a variety of existing upper and lower SGD convergence bounds into the hybrid quantum-classical setting. These bounds then allow them to construct a simple class of optimization problems for which it can be proven that certain first-order hybrid quantum-classical SGD methods converge substantially faster than any zeroth-order method.

In light of these previous results and observations, we explore in this work, both theoretically and numerically, the applicability of these techniques, and a variety of heuristic extensions, in multiple concrete settings. As previously observed by Harrow and Napp [11], one such setting is in the context of VQE, where the cost function is typically a linear combination of expectation values of local Hamiltonian terms, and as such an unbiased estimator of the gradient can be constructed by sampling commuting subsets of these local terms in each optimization step. However, multiple other hybrid quantum-classical optimization settings also facilitate the use of similar strategies. For example, the parameter shift rule re-expresses partial derivatives as linear combinations of expectation values with respect to shifted circuit parameters, and unbiased estimators to these derivatives can therefore similarly be easily obtained by sampling subsets of terms in each optimization step. This insight can be particularly valuable in the context of continuous variable quantum circuits, where an infinite number of parameter shift terms may be required [16]. Additionally, in many data driven settings, such as quantum classifiers for example, loss functions are constructed as sums over data-set instances, and sampling subsets of instances in each optimization step is a well known classical strategy – known as mini-batch SGD – which has already been adopted from classical machine learning [5]. Of particular interest is the fact that these "sampling from linear combination" type SGD schemes can all be combined with each other, and with efficient expectation value estimation, resulting in optimization methods that we refer to as *"doubly stochastic"* gradient descent. This is in the same spirit as similarly motivated techniques for kernel methods, in which

scalable doubly stochastic estimators to the gradient are constructed through the combination of two distinct unbiased approximations to the gradient [26, 27].

Similarly to the kernel method setting, the concrete doubly stochastic SGD optimization schemes we formulate here may provide large efficiency gains over existing approaches, crucial for the implementation of variational algorithms on existing devices. For example, if a quantum classifier trained with D data points has a cost function consisting of M observables, which each need K "parameter shift" terms to extract the analytical gradient, and each expectation is an average of N measurement results, the most extreme version of stochastic gradient descent uses only *a single measurement sample* to estimate the gradient, saving $\mathcal{O}(DMKN)$ measurements in each optimization step. Additionally, from a theoretical perspective, as has been previously mentioned, formalizing these notions allows one to prove convergence guarantees, in suitably restricted settings, for both existing methods and newly proposed SGD optimizers, thereby placing gradient-based hybrid quantum-classical optimization within a rigorous theoretical framework, which can be exploited both to guide the development and analysis of new methods, and to facilitate comparison of existing methods [11].

As suggested by previously obtained convergence bounds, and confirmed here in numerical simulations of various benchmark tasks for which these convergence bounds may not be directly applicable, the increased variance of such extreme-case estimators typically results in more optimization steps being required for convergence, and potentially non-optimal final solutions. However, while more optimization steps may be required, the enhanced efficiency of each each optimization step can result in significant overall savings in the number of circuit executions and measurements which are necessary to achieve convergence. Furthermore, in practice, a promising strategy is to treat the various SGD parameters – such as learning rate, number of measurement shots, and number of linear combination terms sampled – as hyper-parameters which are adjusted appropriately through the course of optimization. As we see from numerical simulations, basic implementations of this strategy suggest one is able to converge to highly accurate solutions, while retaining the efficiency gains of SGD. In fact, very recently the authors of Ref. [28] have also observed that the number of measurement shots used for expectation value estimation can be treated as a hyper-parameter within an SGD framework, and suggested multiple sophisticated heuristic strategies, inspired by state-of-the-art methods from classical optimization, for varying this parameter through the course of optimization. These strategies may well also be applicable within the context of the doubly stochastic gradient descent schemes we consider here, and provide a natural avenue of investigation for future work.

This work is structured as follows: We present the setting and basic idea in Section 2. We then develop a more general framework and body of concrete examples by exploring in detail the settings of VQE, QAOA and quantum classifiers in Sections 3, 4 and 5 respectively. In Section 6, we discuss certain extensions beyond the settings we consider here, before proceeding in Section 7 to prove rigorous convergence guarantees, complimentary to previously obtained bounds, for all considered optimizers. Given this formal framework, we present in Section 8 multiple numerical experiments and benchmarks, before concluding in Section 9 with both a discussion and outlook.

2 Setting and Idea

Given a model parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$, and some loss function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$, stochastic gradient descent algorithms can be viewed as optimization algorithms in which the exact gradient descent update rule

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}^{(t)}), \tag{1}$$

is replaced with a stochastic update rule of the form

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha g^{(t)}(\boldsymbol{\theta}^{(t)}), \tag{2}$$

where $\{g^{(t)}(\boldsymbol{\theta})\}$ is a sequence of random variables – estimators of the gradient – which defines the particular algorithm. Perhaps counter-intuitively, this may offer multiple advantages: Stochasticity can potentially aid in the avoidance of local minima and saddle points [29], and if well designed, $g^{(t)}(\boldsymbol{\theta})$ can be much more efficiently evaluated than the full gradient $\nabla \mathcal{L}$ [30, 31]. While such

algorithms are heavily used and studied as heuristics, there is also an increasing body of work seeking to understand and prove their convergence properties. While we postpone a detailed discussion of convergence theorems to Section 7, we note that a fundamental property required for obtaining convergence guarantees is that the estimators $\{g^{(t)}(\boldsymbol{\theta})\}$ are *unbiased* – i.e.,

$$\mathbb{E}[g^{(t)}(\boldsymbol{\theta})] = \nabla\mathcal{L}(\boldsymbol{\theta}) \tag{3}$$

for all t. In this work we will be concerned with the development of stochastic gradient descent algorithms within the setting of hybrid quantum-classical optimization. In particular, we will consider loss functions $\mathcal{L}$ of the form

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}, \langle O_1\rangle_{\boldsymbol{\theta}}, \ldots, \langle O_M\rangle_{\boldsymbol{\theta}}), \tag{4}$$

where $\langle O_i\rangle_{\boldsymbol{\theta}}$ is the expectation value of an observable O_i with respect to the outcome of a parameterized quantum circuit $U(\boldsymbol{\theta})$ acting on an initial state vector $|\mathbf{0}\rangle$, i.e.

$$\langle O_i\rangle_{\boldsymbol{\theta}} = \langle \mathbf{0}|U^\dagger(\boldsymbol{\theta})O_iU(\boldsymbol{\theta})|\mathbf{0}\rangle. \tag{5}$$

In the following sections, we will examine in detail specific loss functions relevant for current applications, however in order to present some of the basic ideas, let us consider as a first example the simple loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \langle O\rangle_{\boldsymbol{\theta}}, \tag{6}$$

for some observable O, which we assume can be readily measured. In order to utilize gradient descent optimization it is necessary to obtain expressions for all partial derivatives $\partial\mathcal{L}(\boldsymbol{\theta})/\partial\theta_i$. While zeroth-order methods, such as finite differences, could be used to estimate these partial derivatives from evaluations of the loss function, we will in this work focus on first-order methods, in which one calculates these partial derivatives directly, without necessarily evaluating the loss function as an intermediate step. In particular, for many settings of interest, it has recently been shown that a *parameter shift rule* can be derived, via which all partial derivatives can be expressed as linear combinations of the same expectation value, but with respect to slightly shifted circuit parameters [15, 16]. In this work we primarily restrict ourselves to settings in which such a rule can be derived[1], which we formalize via the following definition:

Definition 1 (Parameter shift rule)**.** *A quantum circuit $U(\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$, satisfies a K-term parameter shift rule if for all observables O and for all parameters θ_i, with $i \in [1, \ldots, d]$, there exist some $\{\gamma_{k,i}\}$ and $\{c_{k,i}\}$ such that*

$$\frac{\partial}{\partial\theta_i}\langle O\rangle_{\boldsymbol{\theta}} = \sum_{k=1}^{K}\gamma_{k,i}\langle O\rangle_{\boldsymbol{\theta}_{k,i}}, \tag{7}$$

where $\boldsymbol{\theta}_{k,i} = \boldsymbol{\theta} + c_{k,i}\boldsymbol{e}_i$, with $\boldsymbol{e}_i$ denoting a unit vector in the i'th direction.

In order to facilitate intuition, before continuing let us consider the following example:

Example 1 (Parameter shift rule for single qubit generators [16])**.** *Consider a parameterized quantum circuit of the form $U(\boldsymbol{\theta}) = \prod_{i=1}^{d} e^{-i\theta_iG_i}$, where each G_i is a single qubit Hermitian operator with eigenvalues $\pm r_i$. In this case one finds that*

$$\frac{\partial}{\partial\theta_i}\langle O\rangle_{\boldsymbol{\theta}} = r_i\langle O\rangle_{\boldsymbol{\theta}+\left(\frac{\pi}{4r_i}\right)\boldsymbol{e}_i} - r_i\langle O\rangle_{\boldsymbol{\theta}-\left(\frac{\pi}{4r_i}\right)\boldsymbol{e}_i} \tag{8}$$

Now, under the assumption of a K-term parameter shift rule for circuit ansatz $U(\boldsymbol{\theta})$, we see that all partial derivatives $\partial\mathcal{L}(\boldsymbol{\theta})/\partial\theta_i$ of our example loss function are the linear combination of K expectation values, as per Eq. (7). As expectation values cannot be evaluated exactly on quantum devices, we see already at this stage that exact gradient descent is not possible in this setting. However, as we show now, estimating expectation values via a finite number of measurement outcomes leads to unbiased estimators for the gradient, and therefore to well motivated stochastic gradient descent schemes. In particular, let us start with the following general definition for the n-shot sample mean estimator of an expectation value:

[1] All results still hold if the parameter shift rule has to be replaced with simple numerical differentiation. However, noise inherent to the quantum device may be much more detrimental in this case.

Definition 2 (*n*-sample mean estimator). *Given a parameterized quantum circuit $U(\boldsymbol{\theta})$ (with $\boldsymbol{\theta} \in \mathbb{R}^d$) and an observable O we define $\tilde{o}^{(n)}(\boldsymbol{\theta})$ as the n-sample mean estimator of $\langle O \rangle_{\boldsymbol{\theta}}$ – i.e., the estimator of $\langle O \rangle_{\boldsymbol{\theta}}$ obtained by averaging the results of n measurements of the observable O on the state vector $U(\boldsymbol{\theta})|0\rangle$.*

Note that by construction we have that

$$\mathbb{E}[\tilde{o}^{(n)}(\boldsymbol{\theta})] = \langle O \rangle_{\boldsymbol{\theta}}, \tag{9}$$

for all n, and that while all n-sample mean estimators have the same expectation value the variance of the estimator decreases with increasing n. Given this unbiased estimator for a single expectation value, it now follows straightforwardly that such estimators can be linearly combined to obtain unbiased estimators for the required partial derivatives, i.e.,

$$\mathbb{E}[\sum_{k=1}^{K} \gamma_{k,i} \tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i})] = \sum_{k=1}^{K} \gamma_{k,i} \mathbb{E}[\tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i})] = \sum_{k=1}^{K} \gamma_{k,i} \langle O \rangle_{\boldsymbol{\theta}_{k,i}} = \frac{\partial}{\partial \theta_i} \langle O \rangle_{\boldsymbol{\theta}}, \tag{10}$$

and therefore the estimator $g_i(\boldsymbol{\theta}) = \sum_k \gamma_{k,i} \tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i})$ is an unbiased estimator for $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i$, which requires nK measurements to construct. Using this estimator we can then define a well founded SGD optimization algorithm via the update rule

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \alpha g_i(\boldsymbol{\theta}^{(t)}), \tag{11}$$

where we have implicitly defined $g_i^{(t)}(\boldsymbol{\theta}^{(t)}) := g_i(\boldsymbol{\theta}^{(t)})$ for all t - i.e. the same estimator is used for all update steps. Importantly, note that this algorithm is valid even in the extreme case of $n = 1$. In fact, as exact evaluation of expectation values is not possible, many previous gradient-based approaches to loss functions such as the one considered here have been large n instances of such an SGD algorithm [17–19]. It is, however, possible to go further, and define a "doubly stochastic" gradient descent optimizer by not only estimating the expectation values via n measurements, but also sampling subsets of terms from the linear combination in each optimization step, and applying appropriate correction weights. In the extreme case of sampling only single terms, such an optimizer requires only n measurements per optimization step, as opposed to nK measurements.

In order to formalize this notion, we will denote the set of non-negative integers less than or equal to k as $[k] := \{1, \ldots, k\}$, and let us assume for now that the estimator $\tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i})$ for each term $\langle O \rangle_{\boldsymbol{\theta}_{k,i}}$ in the linear combination of Eq. (7) is a discrete random variable which can take $n(k,i)$ values $\{\lambda_j^{(k,i)} \,|\, j \in [n_{k,i}]\}$, with respective probabilities $\text{prob}(\lambda_j^{(k,i)})$. We now note that the random variable $g_i(\boldsymbol{\theta})$ taking values $\{(K\gamma_{k,i})\lambda_j^{(k,i)} \,|\, j \in [n(k,i)], k \in [K]\}$ with probabilities $(1/K)\text{prob}(\lambda_j^{(k,i)})$ is an unbiased estimator for the linear combination of Eq. (7), i.e.,

$$\mathbb{E}[g_i(\boldsymbol{\theta})] = \sum_{k=1}^{K} \sum_{j=1}^{n(k,i)} [(1/K)\text{prob}(\lambda_j^{(k,i)})][(K\gamma_{k,i})\lambda_j^{(k,i)}] \tag{12}$$

$$= \sum_{k=1}^{K} \gamma_{k,i} \sum_{j=1}^{n(k,i)} \text{prob}(\lambda_j^{(k,i)}) \lambda_j^{(k,i)} \tag{13}$$

$$= \sum_{k=1}^{K} \gamma_{k,i} \mathbb{E}[\tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i})] \tag{14}$$

$$= \sum_{k=1}^{K} \gamma_{k,i} \langle O \rangle_{\boldsymbol{\theta}_{k,i}} \tag{15}$$

$$= \frac{\partial}{\partial \theta_i} \langle O \rangle_{\boldsymbol{\theta}}. \tag{16}$$

The important thing to note is that the random variable $g_i(\boldsymbol{\theta})$ can be sampled by first sampling a term of the linear combination uniformly at random – i.e., drawing k from $[1, \ldots, K]$ with

$\text{prob}(k) = 1/K$ – and then sampling the random variable $\tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i})$ (requiring n measurements) before applying the correction factor $K\gamma_{k,i}$ to the outcome. Alternatively, one can also sample with $\text{prob}(k) = |\gamma_{k,i}|/(\sum_k |\gamma_{k,i}|)$, and apply the correction factor $\gamma_{k,i}/\text{prob}(k)$. While for certain settings this may well lead to estimators with lower variance, for ease of presentation we restrict ourselves here to uniform sampling over linear combinations. As $g_i(\boldsymbol{\theta})$ is an unbiased estimator for $\partial\mathcal{L}(\boldsymbol{\theta})/\partial\theta_i$ one can use this estimator for an SGD optimizer, requiring only n measurements per optimization step, which is summarized via the following algorithms:

Algorithm 1 Stochastic gradient descent

Set initial circuit parameters $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$ and learning rate $\alpha^{(0)}$
$t \leftarrow 0$
while $t < T$ **do** ▷ Iterate through optimization steps
  for all $1 \leq i \leq d$ **do** ▷ Iterate through circuit parameters
    $g_i(\boldsymbol{\theta}^{(t)}) \leftarrow \text{PartialDerivativeEstimator}(i, \boldsymbol{\theta}^{(t)})$ ▷ Construct estimator for $\partial\mathcal{L}(\boldsymbol{\theta}^{(t)})/\partial\theta_i$
  end for
  $\alpha^{(t+1)} \leftarrow \text{GetLearningRate}(t, \alpha^{(t)})$ ▷ Update the learning rate
  $\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} - \alpha^{(t+1)} g_i(\boldsymbol{\theta}^{(t)})$ ▷ Update all circuit parameters
  $t \leftarrow t + 1$
end while

Algorithm 2 n-shot doubly stochastic partial derivative estimator

Given $U(\boldsymbol{\theta})$ satisfying a K-term parameter shift rule, and $\mathcal{L}(\boldsymbol{\theta}) = \langle O \rangle_{\boldsymbol{\theta}}$

function PartialDerivativeEstimator$(i, \boldsymbol{\theta}^{(t)})$
  Sample $k \in \{1, \ldots, K\}$ with $\text{prob}(k) = 1/K$ ▷ Sample a parameter shift term
  Evaluate $\tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\langle O \rangle_{\boldsymbol{\theta}_{k,i}^{(t)}}$ via n measurements
  $g_i(\boldsymbol{\theta}^{(t)}) \leftarrow (K\gamma_{k,i})\tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\partial\mathcal{L}(\boldsymbol{\theta}^{(t)})/\partial\theta_i$ by applying correction factor
  return $g_i(\boldsymbol{\theta}^{(t)})$
end function

At this stage, the basic idea should be clear: In order to implement stochastic gradient descent one requires estimators for all partial derivatives, and when these partial derivatives can be expressed via linear combinations of expectation values then one can define a simple unbiased estimator for the gradient by using a finite number of measurements to estimate the expectation values occurring in the linear combination. Moreover, one can define a "doubly stochastic" estimator, as illustrated in Algorithm 2, by additionally sampling a subset of the terms of the linear combination in each optimization step. However, as we will see in the following sections, such linear combinations can arise also from loss functions that are a linear combination of observables, as in VQE, or from loss functions which are a sum over data-set instances, as for example found in quantum classification problems. In the latter example, one has to exercise caution as the loss function may not be a simple linear combination of expectation values, but rather a linear combination of non-linear functions of expectation values. Generic settings such as these require care to deal with properly, as discussed in Sections 5 and 6. Finally, before continuing we note that we have not specified the function GetLearningRate which is called by Algorithm 1. While a constant learning rate could be used, In practice a variety of different adaptive strategies are exploited, which can be proven to improve convergence properties in restricted settings [21, 32]. Additionally, it should be noted that a wide variety of more sophisticated variations to Algorithm 1 exist, in which both the learning rate and parameter update rule depend explicitly on the history of prior gradient estimates [25]. We will explore and compare such variations to Algorithm 1 in Section 8.

3 Unbiased estimators for VQE

In this section, we examine how the ideas introduced in Section 2 can be straightforwardly extended and applied to the setting of variational quantum eigensolvers (VQE). In particular, given a local Hamiltonian

$$H = \sum_{j=1}^{M} H_j, \tag{17}$$

which encodes a problem of interest, we define the VQE loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \langle H \rangle_{\boldsymbol{\theta}} = \sum_{j=1}^{M} \langle H_j \rangle_{\boldsymbol{\theta}}. \tag{18}$$

For this simple loss function, we have that

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_i} = \frac{\partial \langle H \rangle_{\boldsymbol{\theta}}}{\partial \theta_i} = \sum_{j=1}^{M} \frac{\partial \langle H_j \rangle_{\boldsymbol{\theta}}}{\partial \theta_i} \tag{19}$$

and it follows that unbiased estimators for $\partial \langle H_j \rangle_{\boldsymbol{\theta}} / \partial \theta_i$ can be straightforwardly linearly combined to construct an unbiased estimator for $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i$. From the previous section it should however be clear that, provided $U(\boldsymbol{\theta})$ satisfies a parameter shift rule, one can easily construct either singly or doubly stochastic estimators for all $\partial \langle H_j \rangle_{\boldsymbol{\theta}} / \partial \theta_i$ via Eq. (7), replacing O with H_j. Once again, these estimators for $\partial \langle H_j \rangle_{\boldsymbol{\theta}} / \partial \theta_i$ can then either be directly combined to provide an unbiased estimator for $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i$, or one can further sample over linear combinations of subsets of local Hamiltonian terms. To be clear, let us assume that $U(\boldsymbol{\theta})$ satisfies a K-term parameter shift rule, i.e., for all H_j

$$\frac{\partial \langle H_j \rangle_{\boldsymbol{\theta}}}{\partial \theta_i} = \sum_{k=1}^{K} \gamma_{k,i} \langle H_j \rangle_{\boldsymbol{\theta}_{k,i}}. \tag{20}$$

We then have that

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_i} = \sum_{j=1}^{M} \sum_{k=1}^{K} \gamma_{k,i} \langle H_j \rangle_{\boldsymbol{\theta}_{k,i}}. \tag{21}$$

As a result, the simplest unbiased estimator for $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i$ that we can define is just the linear combination of estimators for $\langle H_j \rangle_{\boldsymbol{\theta}_{k,i}}$ – i.e., the random variable

$$g_i(\boldsymbol{\theta}) = \sum_{j=1}^{M} \sum_{k=1}^{K} \gamma_{k,i} \tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i}), \tag{22}$$

where $\tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i})$ is the n-sample mean estimator of $\langle H_j \rangle_{\boldsymbol{\theta}_{k,i}}$, as per Definition 2. This estimator, which requires nKM measurements per parameter update, can be constructed using the function PartialDerivativeEstimator_1 given in Algorithm 3 below.

However, as we have seen in the previous section, in order to obtain more efficient unbiased estimators for $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i$, we can sample subsets of terms from either the summation over parameter shift terms, the summation over local Hamiltonian terms, or both. In order to provide an explicit example, consider the function PartialDerivativeEstimator_2 in Algorithm 3, in which only a single local term of the Hamiltonian is sampled, with a cost of nK measurements per parameter update. Note the similarity of the approach taken here with *random compiling* in digital quantum simulation [33], which in that context offers advantages such as favourable scaling of Trotter error accumulation. While this function explicitly samples a single local term of the Hamiltonian, it is possible to straightforwardly generalize this to an algorithm which samples a *subset* of local terms of the Hamiltonian for each parameter update. In particular, note that if $[H_l, H_m] = 0$, then the estimators $\tilde{h}_l^{(n)}(\boldsymbol{\theta}_{k,i})$ and $\tilde{h}_m^{(n)}(\boldsymbol{\theta}_{k,i})$ could both be constructed from measurements of n copies of the state vector $U(\boldsymbol{\theta}_{k,i})|\mathbf{0}\rangle$. As a result, by sampling subsets of commuting local terms in each parameter update step, which can be measured simultaneously on the state prepared by a

single circuit execution, we are able to decrease the variance of the estimators while conserving the number of circuit executions required per parameter update. This approach allows for previously developed methods for grouping commuting Hamiltonian terms [34], again reminiscent of methods of grouping of terms in digital quantum simulation [35], to be easily applied within this context.

Finally, the function PartialDerivativeEstimator_3 in Algorithm 3 makes clear how in a similar manner one could additionally sample over terms (or subsets of terms) of the parameter shift summation, resulting in an SGD algorithm requiring only n measurements per parameter update. Of course, despite the increase in efficiency per optimization step, using small values of n and sampling over terms of the linear combinations leads to estimators with increased variance, and intuitively one may expect this to lead to slower convergence (in terms of number of update steps required), and to solutions which may be relatively far from global minima. This intuition is indeed valid, and in Sections 7 and 8 we explore these trade-offs between efficiency and accuracy, from both an analytical and numerical perspective.

Algorithm 3 n-shot stochastic partial derivative estimators for VQE

Given $U(\boldsymbol{\theta})$ satisfying a K-term parameter shift rule, with $\boldsymbol{\theta} \in \mathbb{R}^d$, and $\mathcal{L}(\boldsymbol{\theta}) = \langle H \rangle_{\boldsymbol{\theta}}$ where $H = \sum_{j=1}^{M} H_j$

function PartialDerivativeEstimator_1($i, \boldsymbol{\theta}^{(t)}$)
  for all $1 \leq j \leq M$ **do** ▷ Iterate through Hamiltonian terms
    for all $1 \leq k \leq K$ **do** ▷ Iterate through parameter shift terms
      Evaluate $\tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\langle H_j \rangle_{\boldsymbol{\theta}_{k,i}^{(t)}}$ via n measurements
    end for
  end for
  $g_i(\boldsymbol{\theta}^{(t)}) \leftarrow \sum_{j=1}^{M} \sum_{k=1}^{K} \gamma_{k,i} \tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\partial \langle H \rangle_{\boldsymbol{\theta}^{(t)}} / \partial \theta_i$ via linear combination
  return $g_i(\boldsymbol{\theta}^{(t)})$
end function

function PartialDerivativeEstimator_2($i, \boldsymbol{\theta}^{(t)}$)
  Sample $j \in \{1, \ldots, M\}$ with prob$(j) = 1/M$ ▷ Sample a Hamiltonian term
  for all $1 \leq k \leq K$ **do** ▷ Iterate through parameter shift terms
    Evaluate $\tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\langle H_j \rangle_{\boldsymbol{\theta}_{k,i}^{(t)}}$ via n measurements
  end for
  $g_i(\boldsymbol{\theta}^{(t)}) \leftarrow \sum_{k=1}^{K} M \gamma_{k,i} \tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\partial \langle H \rangle_{\boldsymbol{\theta}^{(t)}} / \partial \theta_i$ via linear combination and correction factor
  return $g_i(\boldsymbol{\theta}^{(t)})$
end function

function PartialDerivativeEstimator_3($i, \boldsymbol{\theta}^{(t)}$)
  Sample $j \in \{1, \ldots, M\}$ with prob$(j) = 1/M$ ▷ Sample a Hamiltonian term
  Sample $k \in \{1, \ldots, K\}$ with prob$(k) = 1/K$ ▷ Sample a parameter-shift term
  Evaluate $\tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\langle H_j \rangle_{\boldsymbol{\theta}_{k,i}^{(t)}}$ via n measurements
  $g_i(\boldsymbol{\theta}^{(t)}) \leftarrow KM \gamma_{k,i} \tilde{h}_j^{(n)}(\boldsymbol{\theta}_{k,i}^{(t)})$ ▷ Construct estimator for $\partial \langle H \rangle_{\boldsymbol{\theta}^{(t)}} / \partial \theta_i$ by applying correction factor
  return $g_i(\boldsymbol{\theta}^{(t)})$
end function

4 Unbiased estimators for QAOA

As in the VQE setting, given a problem Hamiltonian

$$H^P = \sum_{j=1}^{M} H_j^P, \tag{23}$$

whose ground state encodes the solution to some combinatorial problem, the QAOA loss function is once again given simply by $\mathcal{L}(\boldsymbol{\theta}) = \langle H^P \rangle_{\boldsymbol{\theta}}$. Unlike VQE however, the QAOA algorithm is defined additionally by a specific parameterized circuit architecture, designed such that the variational circuit implements a discretized adiabatic evolution, from some known initial state, into the desired target state. While the loss function is the same, the nature of this specific ansatz has consequences for the design of SGD optimizers, which we focus our attention on here. To be more precise, when using parameterized circuits in the context of hybrid quantum-classical optimization, one often considers parameterized circuits of the form

$$U(\boldsymbol{\theta}) = \prod_{i=1}^{d} U_i(\theta_i), \tag{24}$$

where each parameterized gate is parameterized by a distinct parameter [7], and for models of this type parameter shift rules can be derived under various different assumptions on the form of the constituent gates [16] (as per Example 1). By contrast, the QAOA parameterized circuit ansatz is defined as

$$U(\boldsymbol{\theta}) = [e^{-i\theta_d H^B} e^{-i\theta_{d-1} H^P}] \dots [e^{-i\theta_2 H^B} e^{-i\theta_1 H^P}], \tag{25}$$

where $H^B = \sum_{j=1}^{M'} H_j^B$ is a mixing Hamiltonian whose ground state is easy to prepare. Note that in this case the variational parameters define a Trotterized time evolution, and therefore a discretization of a typical adiabatic protocol. Under the assumption that all local terms of both the mixing and problem Hamiltonian commute – i.e., $[H_j^P, H_{j'}^P] = 0$ and $[H_j^B, H_{j'}^B] = 0$ for all j, j' – one has that

$$U(\boldsymbol{\theta}) = [(\prod_{j'=1}^{M'} e^{-i\theta_d H_{j'}^B})(\prod_{j=1}^{M} e^{-i\theta_{d-1} H_j^P})] \dots [(\prod_{j'=1}^{M'} e^{-i\theta_2 H_{j'}^B})(\prod_{j=1}^{M} e^{-i\theta_1 H_j^P})], \tag{26}$$

and one sees that multiple constituent gates are parameterized by the same variational parameter. In order to understand how a parameter shift rule can still be applied in this case, let us consider first a simple parameterized circuit architecture consisting of only a single variational parameter, which parameterizes a single variational gate, i.e.,

$$U(\theta) = U_L e^{-i\theta G} U_R. \tag{27}$$

In this case we see that for any observable O the partial derivative of the parameterized expectation value is given by

$$\frac{\partial}{\partial\theta}\langle O\rangle_\theta = \langle \mathbf{0}|U^\dagger(\theta) O U_L \left(\frac{\partial}{\partial\theta} e^{-i\theta G}\right) U_R|\mathbf{0}\rangle + \text{h.c.}, \tag{28}$$

and from this expression, if $\mathrm{e}^{-\mathrm{i}\theta G}$ admits a $K(G)$-term parameter shift rule, then by definition

$$\frac{\partial}{\partial\theta}\langle O\rangle_\theta = \sum_{k=1}^{K(G)} \gamma_k(G)\langle O\rangle_{\theta_{k,G}}, \tag{29}$$

where we have denoted explicitly that the number of terms, linear coefficients and parameter shifts all depend on G. Let us now consider a parameterized circuit with a single variational parameter, parameterizing multiple gates, whose generators G_j all commute, i.e.,

$$U(\theta) = U_L(\prod_{j}^{M} e^{-i\theta G_j}) U_R. \tag{30}$$

In particular, we note by comparison with Eq. (26) that the QAOA ansatz is precisely of this form. In this case, the partial derivative $\langle O\rangle_\theta$ is given by

$$\frac{\partial}{\partial\theta}\langle O\rangle_\theta = \langle \mathbf{0}|U^\dagger(\theta)OU_L\Big(\frac{\partial}{\partial\theta}\big[\prod_{j'}^{M} e^{-i\theta G_{j'}}\big]\Big)U_R|\mathbf{0}\rangle + \text{h.c.} \tag{31}$$

$$= \sum_{j=1}^{M}\langle \mathbf{0}|U^\dagger(\theta)OU_L\left(\frac{\partial}{\partial\theta}e^{-i\theta G_j}\right)\big[\prod_{j'\neq j}^{M} e^{-i\theta G_{j'}}\big]U_R|\mathbf{0}\rangle + \text{h.c.} \tag{32}$$

$$= \sum_{j=1}^{M}\left(\langle \mathbf{0}|U^\dagger(\theta)OU_L\left(\frac{\partial}{\partial\theta}e^{-i\theta G_j}\right)\tilde{U}_R|\mathbf{0}\rangle + \text{h.c.}\right). \tag{33}$$

By comparison with Eq. (28) we therefore see that, under the assumption that all the gates $\mathrm{e}^{-\mathrm{i}\theta G_j}$'s still satisfy a parameter shift rule,

$$\frac{\partial}{\partial\theta}\langle O\rangle_\theta = \sum_{j=1}^{M}\sum_{k=1}^{K(G_j)}\gamma_k(G_j)\langle O\rangle_{\theta_{k,G_j}}. \tag{34}$$

At this stage we can finally see that in the case where multiple gates are parameterized by the same variational parameter θ – as is the case for the QAOA ansatz – the parameter shift rule we obtain for the partial derivative $\partial\langle O\rangle_\theta/\partial\theta$ can be written as a double sum. The first of these summations runs over all gates parameterized by this parameter, while the second runs over the parameter shift terms for a specific gate. As the QAOA loss function is identical to the VQE loss function, all the unbiased estimators of the previous section can be straightforwardly adapted to the QAOA setting, while the additional structure of the parameter shift rule, allows one to further sample from the linear combination over gates parameterized by the same variational parameter.

5 Unbiased estimators for MSE Quantum classifiers

Given the tools developed in the previous sections it is now relatively straightforward to construct unbiased estimators, and therefore stochastic gradient descent optimizers, for mean squared error (MSE) quantum classifiers. As we will see, the primary obstacle in this case is that the required partial derivatives are not linear combinations of expectation values, but linear combinations of non-linear functions of expectation values. To begin, it is helpful to define the mean squared error loss, as used for example in univariate regression. Specifically, given a target value $y \in \mathbb{R}$ we define the mean squared error (MSE) loss with respect to the operator O as

$$\mathcal{L}_{\text{MSE}}(\boldsymbol{\theta}) = (\langle O\rangle_{\boldsymbol{\theta}} - y)^2. \tag{35}$$

If one is given a data-set of binary labelled examples, as is the case in classification problems, it is possible to use the average MSE loss over the data-set as the loss function for a classification algorithm. More precisely, given a data-set $\mathcal{D} = \{(x_j, y_j)\,|\,j \in [M]\}$, with $y_j \in \mathbb{R}$, let us define $\langle O\rangle_{\boldsymbol{\theta},x_j}$ as the expectation value of the operator O with respect to the state vector $U(\boldsymbol{\theta}, x_j)|\mathbf{0}\rangle$, i.e.,

$$\langle O\rangle_{\boldsymbol{\theta},x_j} = \langle \mathbf{0}|U^\dagger(\boldsymbol{\theta}, x_j)OU(\boldsymbol{\theta}, x_j)|\mathbf{0}\rangle, \tag{36}$$

where in this case $U(\boldsymbol{\theta}, x_j)$ is the unitary operator corresponding to a quantum circuit parameterized by both $\boldsymbol{\theta}$ and x_j. The MSE loss with respect to O, over the data-set $\mathcal{D}$, is then defined as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{j=1}^{M}(\langle O\rangle_{\boldsymbol{\theta},x_j} - y_j)^2 \tag{37}$$

$$:= \frac{1}{M}\sum_{j=1}^{M}\mathcal{L}_{\text{MSE},j}(\boldsymbol{\theta}). \tag{38}$$

We note that typically one may also add a regularization term to the above loss function, which we have omitted for notational clarity, but can be straightforwardly included in our analysis, as will be later shown. In order to construct stochastic gradient descent optimizers for the loss function of Eq. (37), it is convenient to start with the simplified "single instance" MSE loss function of Eq. (35). For this loss function, we have that

$$\frac{\partial \mathcal{L}_{\text{MSE}}(\boldsymbol{\theta})}{\partial \theta_i} = 2\langle O \rangle_{\boldsymbol{\theta}} \frac{\partial \langle O \rangle_{\boldsymbol{\theta}}}{\partial \theta_i} - 2y \frac{\partial \langle O \rangle_{\boldsymbol{\theta}}}{\partial \theta_i}. \tag{39}$$

In order to obtain an unbiased estimator for the partial derivative $\partial \mathcal{L}_{\text{MSE}}(\boldsymbol{\theta})/\partial \theta_i$ it is sufficient to have *independent* unbiased estimators for $\langle O \rangle_{\boldsymbol{\theta}}$ and $\partial \langle O \rangle_{\boldsymbol{\theta}}/\partial \theta_i$. However, for settings in which a parameter shift rule applies, we have already constructed such estimators. Explicitly, the n-sample mean $\tilde{o}^{(n)}(\boldsymbol{\theta})$ is an unbiased estimator for $\langle O \rangle_{\boldsymbol{\theta}}$, while under the assumption of a K-term parameter shift rule

$$\tilde{d}_i^{(n)}(\boldsymbol{\theta}) = \sum_{k=1}^{K} \gamma_{k,i} \tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i}), \tag{40}$$

is an unbiased estimator for $\partial \langle O \rangle_{\boldsymbol{\theta}}/\partial \theta_i$, which is independent from $\tilde{o}^{(n)}(\boldsymbol{\theta})$ since they are evaluated from measurements of different circuit evaluations. As a result, one can straightforwardly show that the random variable

$$\tilde{d}_{\text{MSE},i}^{(n)}(\boldsymbol{\theta}) := 2\tilde{o}^{(n)}(\boldsymbol{\theta}) \tilde{d}_i^{(n)}(\boldsymbol{\theta}) - 2y \tilde{d}_i^{(n)}(\boldsymbol{\theta}) \tag{41}$$

is an unbiased estimator for $\partial \mathcal{L}_{\text{MSE}}(\boldsymbol{\theta})/\partial \theta_i$, which requires $(K+1)n$ measurements to construct. Of course, in the same spirit as previous sections, one can also sample single terms from the summation over the parameter shift terms, giving rise to an estimator which requires only $2n$ measurements to evaluate. In light of the above it is now straightforward to design unbiased estimators for MSE quantum classifiers, as the loss function of Eq. (37) involves only an additional linear combination over data-set instances. In order to formalize this, let's start with the following definitions:

Definition 3. *Given a data-set $\mathcal{D} = \{(\boldsymbol{x}_j, y_j) \,|\, j \in [M]\}$, let us define $\tilde{o}^{(n)}(\boldsymbol{\theta}, j)$ as the n-sample mean estimator of $\langle O \rangle_{\boldsymbol{\theta}, x_j} = \langle \mathbf{0} | U^\dagger(\boldsymbol{\theta}, x_j) O U(\boldsymbol{\theta}, x_j) | \mathbf{0} \rangle$. Furthermore, let $U(\boldsymbol{\theta}, x_j)$ satisfy a K-term parameter shift rule (w.r.t. $\boldsymbol{\theta}$), and let us further define*

$$\tilde{d}_i^{(n)}(\boldsymbol{\theta}, j) := \sum_{k=1}^{K} \gamma_{k,i} \tilde{o}^{(n)}(\boldsymbol{\theta}_{k,i}, j), \tag{42}$$

$$\tilde{d}_{\text{MSE},i}^{(n)}(\boldsymbol{\theta}, j) := 2\tilde{o}^{(n)}(\boldsymbol{\theta}, j) \tilde{d}_i^{(n)}(\boldsymbol{\theta}, j) - 2y_j \tilde{d}_i^{(n)}(\boldsymbol{\theta}, j). \tag{43}$$

Given these estimators, we have that for the data-set MSE loss function of Eq. (38)

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_i} = \frac{1}{M} \sum_{j=1}^{M} \frac{\partial \mathcal{L}_{\text{MSE},j}(\boldsymbol{\theta})}{\partial \theta_i}, \tag{44}$$

and therefore the random variable

$$g_i^{(n)}(\boldsymbol{\theta}) = \sum_{j=1}^{M} \frac{1}{M} \tilde{d}_{\text{MSE},i}^{(n)}(\boldsymbol{\theta}, j) \tag{45}$$

is an unbiased estimator for $\partial \mathcal{L}(\boldsymbol{\theta})/\partial \theta_i$, which requires $M(K+1)n$ measurements to construct. However, as in the case of classical mini-batch SGD, one can sample subsets B (batches) of data instances in each optimization step, giving rise to a "doubly stochastic" unbiased estimator which requires only $|B|(K+1)n$ measurements per parameter update. We present this particular algorithm below, for the case $|B| = 1$, however given the explicit VQE estimators of Algortithm 3 it should be clear how this can be extended to include sampling over parameter shift terms.

Algorithm 4 n-shot doubly stochastic partial derivative estimator for MSE type loss functions

Given $U(\boldsymbol{\theta})$ satisfying a K-term parameter shift rule, with $\boldsymbol{\theta} \in \mathbb{R}^d$, a data-set $\mathcal{D} = \{(x_j, y_j) \,|\, j \in [M]\}$, and the loss function of Eq. (38)

function PARTIALDERIVATIVEESTIMATOR_4($i, \boldsymbol{\theta}^{(t)}$)
  Sample j uniformly from $\{1, \ldots, M\}$ ▷ Sample a data-set instance
  Evaluate $\tilde{o}^{(n)}(\boldsymbol{\theta}^{(t)}, j)$ ▷ Construct estimator for $\langle O \rangle_{\boldsymbol{\theta}^{(t)}, x_j}$ via n measurements
  for all $1 \leq k \leq K$ **do** ▷ Iterate over parameter shift terms
    Evaluate $\tilde{o}^{(n)}(\boldsymbol{\theta}^{(t)}_{k,i}, j)$ ▷ Construct estimator for $\langle O \rangle_{\boldsymbol{\theta}^{(t)}_{k,i}, x_j}$ via n measurements
  end for
  $\tilde{d}^{(n)}_i(\boldsymbol{\theta}^{(t)}, j) \leftarrow \sum_{k=1}^{K} \gamma_{k,i} \tilde{o}^{(n)}(\boldsymbol{\theta}^{(t)}_{k,i}, j)$ ▷ Construct estimator for $\partial \langle O \rangle_{\boldsymbol{\theta}^{(t)}, x_j} / \partial \theta_i$
  $\tilde{d}^{(n)}_{\mathrm{MSE},i}(\boldsymbol{\theta}^{(t)}, j) \leftarrow 2\tilde{o}^{(n)}(\boldsymbol{\theta}^{(t)}, j)\tilde{d}^{(n)}_i(\boldsymbol{\theta}^{(t)}, j) - 2y_j \tilde{d}^{(n)}_i(\boldsymbol{\theta}, j)$ ▷ Construct estimator for $\partial \mathcal{L}_{\mathrm{MSE},j}(\boldsymbol{\theta}^{(t)}) / \partial \theta_i$
  $\tilde{g}^{(n)}_i(\boldsymbol{\theta}^{(t)}) \leftarrow \tilde{d}^{(n)}_{\mathrm{MSE},i}(\boldsymbol{\theta}^{(t)}, j)$ ▷ Construct estimator for $\partial \mathcal{L}(\boldsymbol{\theta}^{(t)}) / \partial \theta_i$
  return $g_i(\boldsymbol{\theta}^{(t)})$
end function

6 Extensions to Generic Loss Functions

In light of the optimizers presented for the MSE quantum classifier, a natural question is whether or not analogous optimizers can be obtained when regularization terms are included, or when alternative loss functions, such as the cross-entropy, are used. Let us begin with a remark on the inclusion of regularization terms, which can be handled straightforwardly via the following observation:

Observation 1. *Given some loss function $\mathcal{L}$, let the random variable $g(\boldsymbol{\theta})$ be an unbiased estimator for $\nabla \mathcal{L}(\boldsymbol{\theta})$ – i.e., $\mathbb{E}[g(\boldsymbol{\theta})] = \nabla \mathcal{L}(\boldsymbol{\theta})$. Then, the random variable*

$$\tilde{g}(\boldsymbol{\theta}) = g(\boldsymbol{\theta}) + \nabla R(\boldsymbol{\theta}) \tag{46}$$

is an unbiased estimator for the regularized loss function $\tilde{\mathcal{L}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + R(\boldsymbol{\theta})$. This can be straightforwardly verified via

$$\mathbb{E}[\tilde{g}(\boldsymbol{\theta})] = \mathbb{E}[g(\boldsymbol{\theta}) + \nabla R(\boldsymbol{\theta})] = \mathbb{E}[g(\boldsymbol{\theta})] + \nabla R(\boldsymbol{\theta}) = \nabla \mathcal{L}(\boldsymbol{\theta}) + \nabla R(\boldsymbol{\theta}) = \nabla \tilde{\mathcal{L}}(\boldsymbol{\theta}). \tag{47}$$

Unfortunately however an extension to generic loss functions is not as straightforward. To see this let's consider a loss function $\mathcal{L}$, for which the partial derivative $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i$ that we would like to estimate is a non-linear function of an expectation value, i.e., $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i = f(\langle O \rangle_{\boldsymbol{\theta}})$ for some non-linear function f. Assume now that the random variable $X(\boldsymbol{\theta})$ is an unbiased estimator for $\langle O \rangle_{\boldsymbol{\theta}}$ (such as the n-sample mean estimator). If f had been a linear function, then we would have that

$$\mathbb{E}[f(X(\boldsymbol{\theta}))] = f(\mathbb{E}[X(\boldsymbol{\theta})]) = f(\langle O \rangle_{\boldsymbol{\theta}}) = \frac{\partial}{\partial \theta_i} \mathcal{L}(\boldsymbol{\theta}), \tag{48}$$

and as such $f(X)$ would be an unbiased estimator for $\partial \mathcal{L}(\boldsymbol{\theta}) / \partial \theta_i = f(\langle O \rangle_{\boldsymbol{\theta}})$. Unfortunately however for generic non-linear functions f we have that $f(\mathbb{E}[X]) \neq \mathbb{E}[f(X)]$, and as a result unbiased estimators for $f(\langle O \rangle_{\boldsymbol{\theta}})$ cannot be straightforwardly constructed from unbiased estimators for $\langle O \rangle_{\boldsymbol{\theta}}$, and more sophisticated strategies are necessary. While we leave the completely general case open for future work, we illustrate here a method for the construction of unbiased estimators for *polynomial* functions of random variables (of which the previously considered MSE loss function is a special case). In particular, let $f : \mathbb{R} \to \mathbb{R}$ be a degree k polynomial, i.e.

$$f(x) = \sum_{j=0}^{k} a_j x^j, \tag{49}$$

and let us define the function $h : \mathbb{R}^k \to \mathbb{R}$ via

$$h(x_1, \ldots, x_k) = a_0 + \sum_{j=1}^{k} a_j \left(\prod_{i=1}^{j} x_i \right). \tag{50}$$

In addition, for all $m \geq k$, let us denote by $\mathbf{P}_{k,m}$ the set of all $m!/(m-k)!$ ordered arrangements $(i_1, \ldots, i_k)$ of size k chosen from $(1, \ldots, m)$. Now, given a set $\{X_j \,|\, j \in [m]\}$ of $m \geq k$ independent copies of the random variable X, one can easily verify that the random variable

$$g^{(m)}(X_1, \ldots, X_m) = \frac{(m-k)!}{m!} \sum_{\mathbf{P}_{k,m}} h(X_{i_1}, \ldots, X_{i_k}) \tag{51}$$

is an unbiased estimator for $f(\mathbb{E}(X))$, which requires m samples from X to construct. In fact, $g^{(m)}$ is the U-statistic for the kernel h, and as such is the minimum variance unbiased estimator for $f(\mathbb{E}(X))$ [36]. As many relevant loss functions, such as the cross entropy or the log-likelihood, are certainly not polynomials, it is worth noting before continuing that despite the absence of rigorous constructions it is still possible to apply the spirit of the constructions from the previous sections – i.e., utilizing single shot estimations of observables and sampling over linear combinations – to obtain *biased estimators* for arbitrary loss functions, whose performance on practical tasks can be easily evaluated, and for which one may still be able to prove convergence bounds [37]. In fact, while unbiased estimators facilitate convergence proofs in simplified settings, it is not a-priori clear that naive (potentially biased) estimators would perform badly as heuristics. Such heuristic analysis, for loss functions such as the cross entropy, should be considered in parallel with a development of the theory. Our motivation in this work is to provide a general framework, and to lay the foundations for more sophisticated analysis.

7 Convergence Guarantees

The construction of convergence guarantees for stochastic gradient descent optimizers in fully realistic settings is currently an active and open area of research, of critical importance for building a theoretically solid foundation for state of the art machine learning algorithms [38]. In particular, as discussed in the previous sections, many prior approaches to gradient descent optimization in the context of hybrid quantum-classical optimization have been large n instances of the algorithms from the previous sections (without sampling over linear combinations), and one of the primary motivations of this work is to place these previously heuristic approaches on a solid formal footing. While convergence guarantees are not yet available for state of the art models, highly non-convex landscapes and optimization algorithms with heuristically adapted learning rates, in certain simplified settings it is indeed possible to obtain convergence theorems, which allow one to build a measure of confidence in these techniques. In order to gain a more formal perspective on the algorithms presented in the previous sections, as well as insight into the requirements for the development of further well motivated stochastic gradient descent algorithms, we examine in this section convergence guarantees for loss functions $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ which satisfy the *Polyak- Lojasiewicz* (PL) inequality, a slightly generalized notion of strong convexity [21]. These bounds are complimentary to the bounds previously discussed by Harrow and Napp for strongly-convex loss functions [11], and are presented explicitly in order to facilitate discussion around both the technical requirements for such guarantees in the hybrid quantum-classical setting, and the trade-offs inherent in SGD optimization schemes, which are explored numerically in the following section. In particular, a loss function $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ satisfies the PL inequality for some $\mu > 0$ if for all $\boldsymbol{\theta} \in \mathbb{R}^d$ it is true that

$$\frac{1}{2} ||\nabla \mathcal{L}(\boldsymbol{\theta})||^2 \geq \mu \big(\mathcal{L}(\boldsymbol{\theta}) - \mathcal{L}(\boldsymbol{\theta}^*) \big), \tag{52}$$

where $\boldsymbol{\theta}^* \in \mathbb{R}^d$ is the value at which $\mathcal{L}(\boldsymbol{\theta})$ attains a global minimum. Under the assumption of the PL inequality, one can state the following convergence theorem, whose proof can be found in Refs. [21, 39]:

Theorem 1 (Stochastic gradient descent convergence [39]). *Let $\mathcal{L} \in C^1(\mathbb{R}^d)$ satisfy the Polyak-Lojasiewicz inequality in Eq. (52) for some $\mu > 0$, have L-Lipschitz gradient and a global minimum attained at $\boldsymbol{\theta}^*$. For any $T \in \mathbb{N}$ and any $\boldsymbol{\theta} \in \mathbb{R}^d$ let $g^{(1)}(\boldsymbol{\theta}), \ldots, g^{(T)}(\boldsymbol{\theta})$ be i.i.d random variables with values in $\mathbb{R}^d$ such that $\mathbb{E}[g^{(t)}(\boldsymbol{\theta})] = \nabla\mathcal{L}(\boldsymbol{\theta})$. With $\boldsymbol{\theta}^{(0)} \in \mathbb{R}^d$ consider the sequence $\boldsymbol{\theta}^{(t+1)} := \boldsymbol{\theta}^{(t)} - \alpha g^{(t)}(\boldsymbol{\theta}^{(t)})$.*

1. *If $\forall \boldsymbol{\theta}, t : \mathbb{E}[||g^{(t)}(\boldsymbol{\theta})||^2] \leq \gamma^2$ and $\alpha \in [0, 1/(2\mu)]$, then*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}^{(T)})] - \mathcal{L}(\boldsymbol{\theta}^*) \leq (1 - 2\mu\alpha)^T \big(\mathcal{L}(\boldsymbol{\theta}^{(0)}) - \mathcal{L}(\boldsymbol{\theta}^*)\big) + \frac{L\alpha\gamma^2}{4\mu}. \tag{53}$$

2. *If $\forall \boldsymbol{\theta}, t : \mathbb{E}[||g^{(t)}(\boldsymbol{\theta})||^2] \leq \beta^2 ||\nabla\mathcal{L}(\boldsymbol{\theta})||^2$ and $\alpha = 1/(L\beta^2)$, then*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta}^{(T)})] - \mathcal{L}(\boldsymbol{\theta}^*) \leq \Big(1 - \frac{\mu}{L\beta^2}\Big)^T \big(\mathcal{L}(\boldsymbol{\theta}^{(0)}) - \mathcal{L}(\boldsymbol{\theta}^*)\big). \tag{54}$$

We note that as all previously discussed algorithms have been constructed via unbiased estimators, Theorem 1 applies directly to these algorithms provided one can prove Lipschitz continuity of the gradient $\nabla\mathcal{L}$. Additionally, in order to apply the convergence bounds of Theorem 1 in a quantitative manner, it is necessary to obtain an upper bound on the quantity $\mathbb{E}[||g^{(t)}(\boldsymbol{\theta})||^2]$, which in principle amounts to calculating the variance of the estimator $g^{(t)}(\boldsymbol{\theta})$, as can be seen from the following straightforward calculation:

$$\begin{aligned} \mathbb{E}[||g^{(t)}(\boldsymbol{\theta})||^2] &= \mathbb{E}[||g^{(t)}(\boldsymbol{\theta})||^2] - \mathbb{E}[||g^{(t)}(\boldsymbol{\theta})||]^2 + \mathbb{E}[||g^{(t)}(\boldsymbol{\theta})||]^2 &(55)\\ &= \mathrm{Var}[g^{(t)}(\boldsymbol{\theta})] + ||\nabla\mathcal{L}(\boldsymbol{\theta})||^2, &(56)\end{aligned}$$

where we defined the variance of a random vector as $\mathrm{Var}[X] := \mathbb{E}[||X||^2] - \mathbb{E}[||X||]^2$. As Theorem 1 only applies in the simplified setting of loss functions which satisfy the PL inequality we will not address these upper bounds explicitly here, as in practice more sophisticated convergence theorems are necessary to make applicable quantitative statements for realistic settings. However, we note the critical role this variance plays in determining the distance between the global optimum and the solution one can be guaranteed to converge to, and in Section 8 we explore this interplay, in realistic settings, through extensive numerical experiments. As such, we focus on the issue of Lipschitz continuity, and provide the following results proving that for a large class of realistic parameterized quantum circuits, under the assumption of a parameter shift rule, the gradient of the loss functions considered in this work are indeed Lipschitz continuous. To be specific, we start with the following theorem (whose proof can be found in Appendix A), which guarantees Lipschitz continuity of expectation values for a specific class of parameterized quantum circuits:

Theorem 2 (Lipschitz continuity of expectation values). *Consider a function $f : [0, 2\pi]^M \mapsto \mathbb{R}$ defined by $f(\boldsymbol{\theta}) = \langle \mathbf{0}|U^\dagger(\boldsymbol{\theta})OU(\boldsymbol{\theta})|\mathbf{0}\rangle$, where $|\mathbf{0}\rangle$ is an arbitrary state in $\mathbb{C}^D$ for some finite D, $U(\boldsymbol{\theta})$ is a quantum circuit consisting of an arbitrary finite number of fixed gates, and M parameterized gates $U_j(\theta_j) = \exp(-i(\theta_j + c_j)H_j)$, with H_j Hermitian. For any observable O and any set of Hermitian operators $\{H_j \,|\, j \in [M]\}$, the function $f(\boldsymbol{\theta})$ is L-Lipschitz with $L = \sqrt{M}[\max_j(\sup_{\boldsymbol{\theta}}(|\partial f(\boldsymbol{\theta})/\partial\theta_j|))]$.*

Given this result, an upper bound to L can be obtained as follows: One finds

$$\begin{aligned} L &= \sqrt{M}\max_j \sup_{\boldsymbol{\theta}} \left|2\mathrm{Re}\left[\langle \mathbf{0}|\, U^\dagger(\boldsymbol{\theta})O\frac{\partial U(\boldsymbol{\theta})}{\partial\theta_j}\,|\mathbf{0}\rangle\right]\right| &(57)\\ &\leq 2\sqrt{M}\ \max_j \sup_{\boldsymbol{\theta}} \left|\langle \mathbf{0}|\, U^\dagger(\boldsymbol{\theta})OU_L(\boldsymbol{\theta})(-\mathrm{i}H_j)U_R(\boldsymbol{\theta})\,|\mathbf{0}\rangle\right| &(58)\\ &\leq 2\sqrt{M}\ \max_j \sup_{\boldsymbol{\theta}} \|OU(\boldsymbol{\theta})\,|\mathbf{0}\rangle\|_2\ \ \|U_L(\boldsymbol{\theta})H_jU_R(\boldsymbol{\theta})\,|\mathbf{0}\rangle\|_2 &(59)\\ &\leq 2\sqrt{M}\ \max_j \sup_{\boldsymbol{\theta}} \|O\|_2\ \|U(\boldsymbol{\theta})\,|\mathbf{0}\rangle\|_2\ \|U_L(\boldsymbol{\theta})\|_2\ \|H_j\|_2\ \|U_R(\boldsymbol{\theta})\,|\mathbf{0}\rangle\|_2 &(60)\\ &= 2\sqrt{M}\ \max_j \|O\|_2\ \|H_j\|_2, &(61)\end{aligned}$$

where $U(\boldsymbol{\theta}) \equiv U_L(\boldsymbol{\theta})e^{-i\theta_j H_j}U_R(\boldsymbol{\theta})$, and Eq. (59) was obtained from Eq. (58) via the Cauchy-Schwarz inequality. In order to cover more realistic loss functions, we need to extend Theorem 2 to sums and products of parameterized expectation values. For example, we have already seen that for the MSE loss function, under the assumption of a parameter shift rule, we have that

$$\frac{\partial}{\partial\theta_j}\mathcal{L}_{\text{MSE}}(\boldsymbol{\theta}) = 2\langle O\rangle_{\boldsymbol{\theta}}\frac{\partial}{\partial\theta_j}\langle O\rangle_{\boldsymbol{\theta}} - 2y\frac{\partial}{\partial\theta_j}\langle O\rangle_{\boldsymbol{\theta}} \tag{62}$$

with $\partial\langle O\rangle_{\boldsymbol{\theta}}/\partial\theta_j$ given by

$$\frac{\partial}{\partial\theta_j}\langle O\rangle_{\boldsymbol{\theta}} = \sum_{k=1}^{K}\gamma_{k,j}\langle O\rangle_{\boldsymbol{\theta}_{k,j}}, \tag{63}$$

where $\boldsymbol{\theta}_{k,j} = \boldsymbol{\theta} + c_{k,j}\boldsymbol{e}_j$. The first thing to notice is that if $\langle O\rangle_{\boldsymbol{\theta}}$ is L-Lipschitz continuous via Theorem 2, then so is $\langle O\rangle_{\boldsymbol{\theta}_{k,j}}$. We see this by noting that

$$U_j(\theta_j + c_{k,j}) = \exp(-i(\theta_j + c_{k,j})H_j) = \exp(-i\theta_j H_j)\exp(-ic_{k,j}H_j) \tag{64}$$

for any constant $c_{k,j}$, and therefore $\langle O\rangle_{\boldsymbol{\theta}_{k,j}}$ satisfies the assumptions of the Lemma after absorbing $\exp(-ic_{k,j})H_j$ into the set of fixed (non-parameterized) gates. This observation, in conjunction with standard result that if $f(\boldsymbol{\theta})$ and $g(\boldsymbol{\theta})$ are Lipschitz continuous functions bounded on their domains, then $f+g$ and fg are Lipschitz continuous [40], guarantees Lipschitz continuity for the class of functions we are interested in.

8 Numerical Experiments and Benchmarks

Given the unbiased estimators of Sections 3, 4 and 5, we explore here their performance on benchmark tasks. In particular, while it is clear from the above analysis that sampling over linear combination terms and utilizing small values of n leads to algorithms which require significantly fewer measurements per optimization step than previous approaches, while possessing convergence guarantees in simplified settings, it is not a-priori clear how these algorithms perform on realistic tasks. Specifically, as the convergence properties of the algorithms depend both on the Lipschitz constant of the gradient of the loss, and upper bounds on the variance of the estimator $g^{(t)}(\boldsymbol{\theta})$, it may be that the advantages of both linear combination sampling and small n implementations are outweighed by slow or unstable convergence behavior in practice. Fortunately, as shown in the following sections, at least in the simplified setting of noiseless devices this is not the case, and small n implementations of the previously introduced algorithms indeed offer multiple advantages over large n approaches, which can be further enhanced through the use of heuristics such as decaying learning rate, and momentum based optimizers [41]. All numerical results presented in the following sections were obtained using `qradient` [42], a custom open-source package for the simulation of hybrid quantum-classical optimization, and all scripts, data and random number seeds are available on request.

8.1 Stochastic gradient descent for VQE

We start by exploring, for different values of n, the estimator formalized by the function PartialDerivativeEstimator_1 in Algorithm 3 (which we refer to as n-shot SGD), allowing us to gain insight into the effect of efficient expectation value estimation, before introducing the additional stochasticity of Hamiltonian term sampling. In particular, we consider the *critical transverse field Ising model* with open boundary conditions, given by the Hamiltonian

$$H = \sum_{j=1}^{N-1} Z_j Z_{j+1} + \sum_{j=1}^{N} X_j, \tag{65}$$

with $N = 8$, where X_j and Z_j are the respective Pauli operators on site j. Critical Ising models are good candidates for such an endeavor, as the entanglement structure of such critical systems prohibits the use of classical tensor network methods for large system sizes [43]. The parameterized circuit that we consider consists of a sequence of σ-blocks (with $\sigma \in [X, Y, Z]$), where a single σ-block has the following structure:

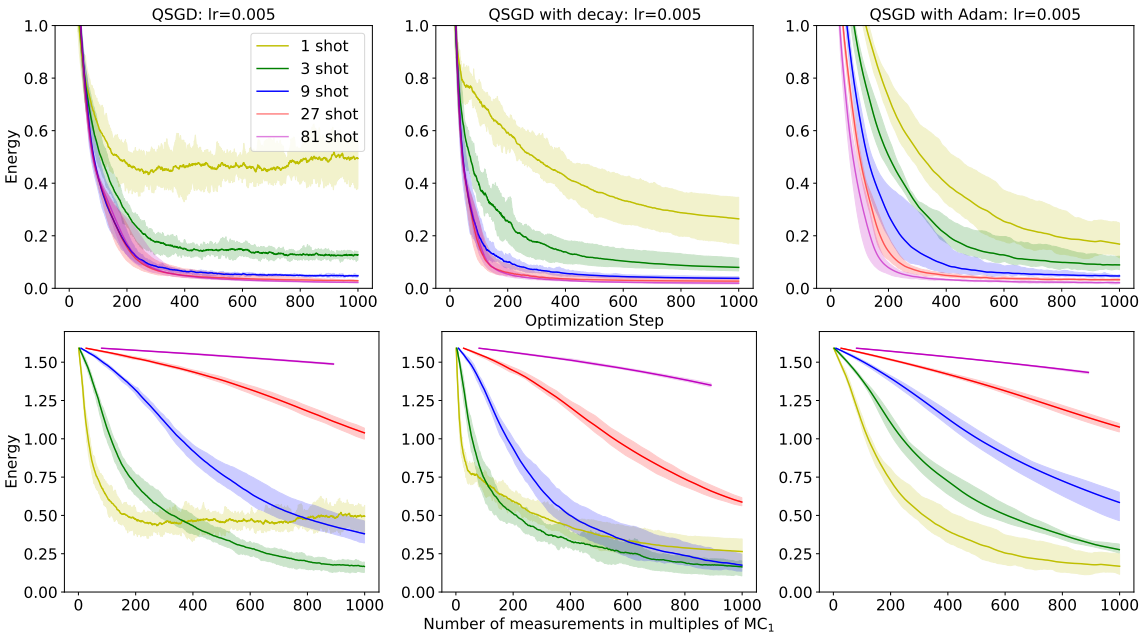

Figure 1: Results of VQE experiments with the estimator PartialDerivativeEstimator_1 from Algorithm 3 (referred to as n-shot SGD). The top row of results is plotted with optimization step on the x-axis. The lower row of results is plotted with number of measurements on the x-axis, in multiples of MC_1, which is the number of measurements required per optimization step by 1-shot SGD. The left column of figures indicates the results for n-shot SGD with fixed learning rate $\alpha = 0.005$. The central column of figures shows the results for n-shot SGD with learning rate decay, from an initial learning rate of $\alpha = 0.005$. The right hand column shows the results of n-shot SGD with learning rate decided adaptively by the Adam optimizer [41], starting from an initial learning rate of $\alpha = 0.005$. For each algorithm, and each value of n, the experiment has been repeated 8 times, and the minimum, mean and maximum at each step is displayed.

1. A layer of parameterized rotations around the σ axis – i.e., an $R_\sigma(\theta) = e^{-i\theta/2\sigma}$ gate acting on all qubits.
2. A layer of nearest neighbour CNOT gates with control on qubit j and target on qubit $j+1$, for all even j.
3. A layer of nearest neighbour CNOT gates with control on qubit j and target on qubit $j+1$, for all odd j.

Note that for one-dimensional circuits consisting of N qubits each block has N free parameters – i.e., the rotation angles on each qubit. The particular architecture that we consider consists of an initial Y-block, with all free parameters set to $\pi/4$, followed by k parameterized σ-blocks, alternating between X, Y and Z blocks, in that order. A k block circuit of this type therefore has kN free parameters, and we consider a circuit with 50 blocks, and therefore 400 free parameters, as $N = 8$. An explicit circuit diagram can be found in Appendix B. We implemented Algorithm 1, calling the estimator PartialDerivativeEstimator_1, with a fixed learning rate $\alpha = 0.005$, for multiple values of n, as well as two heuristic variations:

1. *Learning rate decay:* If the energy has not decreased in the last 20 optimization steps, then the learning rate is decreased by a factor of 2.
2. *Adam optimizer:* A variation of Algorithm 1 which computes adaptive learning rates for each parameter, using the history of prior gradient estimates [41]. Implemented with an initial learning rate of $\alpha = 0.005$ and hyper-parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The results of these experiments can be seen in Fig. 1, and there are a variety of interesting aspects to note. Firstly, from the top left figure one can see that, as one might expect, with a *fixed* learning rate the quality of the final solution is strongly effected by the value of n. In particular, for very small values of n we expect a large variance in the gradient, even close to a local minima, and the algorithm can therefore only converge to relatively inaccurate solutions. Again as we might expect, the accuracy of the obtained solutions can be successively tightened by increasing the value of n, and in the process decreasing the variance of the gradient between optimization steps. By comparing the top left figure with the central and right figure of the top row we can however see that the quality of the final solution can also be significantly improved, even for very small values of n, by utilizing adaptive or decaying learning rates, and once again this makes intuitive sense. Once one has converged to a fixed solution, decreasing the learning rate scales the gradient estimator, and therefore effectively decreases the variance in the parameter updates, allowing one to move closer to a local minimum, even with a large variance in the gradient [32]. As such, we see that both decaying learning rate and increasing shot number allow one to improve the quality of the final solution that is obtained. We note however, that analogously to SGD in purely classical deep learning where variance can be decreased either through scaling via learning rate decay or directly via the batch size, it is not a-priori clear which strategy is optimal [44]. Additionally, from the top row of results in Fig. 1 it would appear that even with adaptive learning rate decay, one can achieve better quality solutions by utilizing larger shot numbers, and as such it is not clear that there is something to be gained by using small n estimators.

In order to explore this more clearly, let us define the number of measurements required per optimization step of an n-shot algorithm as MC_n – where MC stands for *measurement cost*. The bottom row of results in Fig. 1 shows the performance of the algorithms not with respect to optimization steps, but with respect to number of measurements performed, as a multiple of MC_1 (the smallest possible number of measurements per optimization step). With respect to this metric, it is clear from the bottom row of figures that in terms of numbers of measurements, single shot ($n = 1$) stochastic gradient descent algorithms converge much faster than large n algorithms. However, by comparing the upper and lower rows of results one can see that in this setting, for both standard SGD and SGD with decay (left and central panel), in order to improve the quality of the final solution, eventually it is necessary to increase the value of n. While the situation is less clear for the Adam optimizer, it also seems likely that increasing n at some point is necessary to achieve higher quality solutions in this case. As such, a natural strategy, which will be discussed further in Section 9, would be to combine learning rate decay with increasing shot numbers. This should allow one to exploit the rapid convergence, and therefore enhanced measurement efficiency,

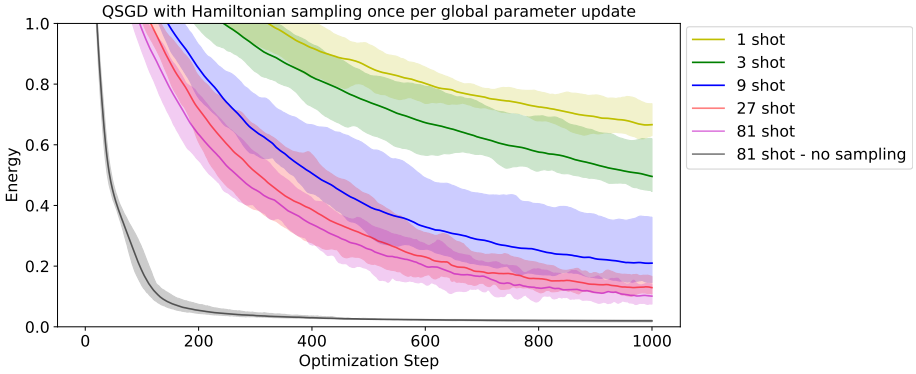

Figure 2: Results of VQE experiments with the estimator PartialDerivativeEstimator_2 from Algorithm 3 – i.e., n-shot stochastic gradient descent incorporating Hamiltonian sampling. The results were obtained using the Adam optimizer, with an initial learning rate of 0.005. Each experiment has been repeated 8 times, and the minimum, mean and maximum at each step is displayed. Additionally, the results of 81 shot SGD without Hamiltonian sampling (i.e. using PartialDerivativeEstimator_1) are shown for comparison.

of small n estimators, while still allowing one to obtain the solutions which are accessible via large n estimators. In fact, recent work has explored precisely such heuristic optimization strategies, with promising results [28].

Given these insights, we explore in Fig. 2 the performance of the estimator PartialDerivativeEstimator_2 – i.e., n-shot stochastic gradient descent with Hamiltonian sampling – in order to explore the additional effect of sampling over linear combinations. As discussed in Section 3, in practical settings one would not sample single terms of the Hamiltonian per parameter update, but rather commuting sets of Hamiltonian terms which can be easily measured simultaneously, so that the maximum amount of information can be extracted from a single circuit forward pass. However, for illustration purposes, we explore here the performance of this algorithms when a single local term of the Hamiltonian is sampled, as this is clearly an extreme case, whose performance can only be improved by sampling more terms simultaneously. Once again we implemented Algorithm 1 with learning rate decided by the Adam optimizer, with an initial learning rate of $\alpha = 0.005$. As a result of Theorem 1, and from our numerical experiments with PartialDerivativeEstimator_1, due to the high variance of the estimator in this case, even with large values of n, we would expect this algorithm to converge slowly as a function of the number of optimization steps required, and indeed this is what is observed in Fig. 2. However, the crucial insight is that although more optimization steps are required, each optimization step of Algorithm 1 requires up to a factor of M less circuit executions (where M is the number of local terms of the Hamiltonian), and using Hamiltonian sampling may well facilitate more rapid convergence or initialization, with respect to the number of circuit executions required. As the precise efficiency gains compared to the simple n-shot estimator PartialDerivativeEstimator_1 depend on the strategy via which local Hamiltonian terms are grouped and measured (i.e., MC_1 varies depending on the strategy for obtaining measurements of all local Hamiltonian terms), we have not provided a direct comparison between PartialDerivativeEstimator_1 and PartialDerivativeEstimator_2 in terms of circuit executions and number of measurements, and we leave such comparisons, with state-of-the art strategies for minimizing all involved costs, to future work, emphasizing that this depends on the particular Hamiltonian at hand. Despite this, it is clear from Fig. 2 that the final solutions obtained when using PartialDerivativeEstimator_2 are relatively far from the optimal solutions obtained by large n versions of PartialDerivativeEstimator_2, and as such once again a natural strategy would be to use Hamiltonian sampling as an initialization strategy, with the number of terms (or commuting sets of terms) of the Hamiltonian treated as a hyper-parameter.

8.2 Stochastic gradient descent for QAOA

As discussed in Section 4, if one does not sample over linear combinations of local Hamiltonian terms, or over parameter shift terms, then the unbiased estimator formalized via PartialDeriva-

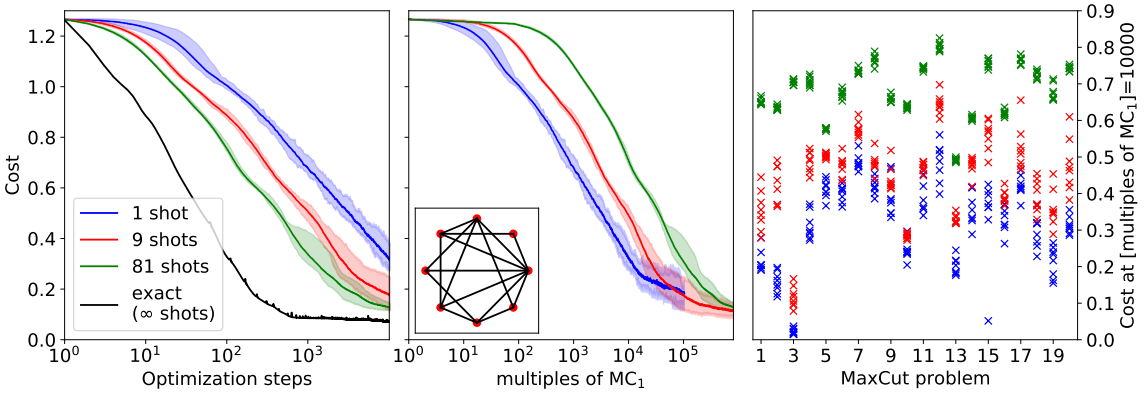

Figure 3: Results of QAOA experiments with n-shot SGD (i.e. PARTIALDERIVATIVEESTIMATOR_1 from Algorithm 3). The left and center figures show the results obtained for the specific MaxCut instance displayed in the inset. While in the left figure the cost is plotted against the number of optimization steps, in the center figure the cost is plotted against the number of circuit executions, up to a constant factor MC_1. The right most figure shows the results obtained by n-shot SGD after 10000MC1 measurements, on 20 randomly drawn MaxCut instances, all with $|V| = 8$ and $|E| = 16$. As the ground state energy of each randomly drawn problem instance H^P may be different (dependent on the number of maximum possible cuts), the cost refers to the energy $\langle 0|U^\dagger(\theta)H^P U(\theta)|0\rangle$, normalized by the absolute value of the ground state energy for the particular problem instance, and shifted by $+1$. This rescaling ensures the minimum achievable cost for all problem instances is zero, and allows for meaningful comparison between problem instances.

TIVEESTIMATOR_1 from Algorithm 3 can be directly applied in the context of the QAOA circuit ansatz, by simply re-interpreting the double sum from the parameter shift rule as a single sum over a multi-index. In this section we explore the performance of this estimator, again for different values of n, on various instances of the MaxCut problem [45]. Specifically, given a graph $G = (V, E)$, the MaxCut problem is equivalent to finding the ground state of the Hamiltonian

$$H^P = \sum_{(i,j)\in E} Z_i Z_j. \tag{66}$$

As mentioned in Section 4, the QAOA ansatz alternates application of the problem Hamiltonian with a mixing Hamiltonian H^B via

$$U(\boldsymbol{\theta}) = [e^{-i\theta_d H^B} e^{-i\theta_{d-1} H^P}] \dots [e^{-i\theta_2 H^B} e^{-i\theta_1 H^P}]. \tag{67}$$

For all experiments performed here, a value of $d = 100$ has been used, with the mixing Hamiltonian $H^B = \sum_{j=1}^{|V|} X_j$. Fig. 3 shows the results obtained by using the above QAOA ansatz, in conjunction with the estimator PARTIALDERIVATIVEESTIMATOR_1, on 20 randomly drawn instances of the MaxCut problem, all with $|V| = 8$ vertices, and $|E| = 16$ edges. For all instances, the parameters were initialized such that they realize a simple linear interpolation between H^P and H^B – i.e., for odd j (parameters of H^P terms) we have that $\theta_j = j/d$ while for even j (parameters of H^B terms) we have that $\theta_j = 1 - j/d$. This initialization is informed by recent studies which identified global optima for MaxCut problems, which were typically close to linear interpolation solutions [46]. The optimization was carried out using the *Adam* optimizer, with initial learning rate $\alpha = 0.001$ and hyper-parameters $\beta_1 = 0.8$ and $\beta_2 = 0.999$. Once again we are able to draw various insights from the results of Fig. 3, which are similar in nature to the conclusions from our VQE simulations. In particular, by comparing the left and center panels of Fig. 3, which show the convergence behavior with respect to optimization steps and measurement cost respectively for one specific MaxCut instance, we once again see that while 1-shot SGD converges slower in terms of the number of optimization steps required, it converges faster in terms of the number of measurements required (where once again we have plotted the number of measurements in multiples of MC_1, the measurement cost per optimization step of single shot SGD). The rightmost panel of Fig. 3 confirms that this behavior is indeed generic. Specifically, the right most panel shows the cost

obtained by n-shot SGD after $10000MC_1$ measurements, on 20 randomly drawn MaxCut instances, and it is clear that for all instances, single shot SGD obtains (often significantly) lower costs with this number of measurements, and hence should clearly be used as an initialization strategy. Given this initialization strategy the second natural question is then whether learning rate decay is sufficient to decrease the variance of the parameter updates such that highly accurate solutions can be obtained, or whether in addition it is necessary to increase the shot number. From the central panel (in which the number of steps is limited by computational resources) the fact that the 9-shot curve intersects the single shot curve suggests that, for this particular example, the 9-shot algorithm is likely to converge to a more accurate solution than the single shot algorithm. As such, if one begins with the single shot algorithm, then it is possible that increasing the number of measurements at some point would allow one to obtain a more accurate solution than one that could be obtained by keeping the number of measurements constant and simply decreasing the learning rate further. Given this, as per the VQE setting, in order to achieve the most accurate solution with the minimum number of measurements, a promising strategy is to combine learning rate decay with an increasing number of measurements.

8.3 Stochastic gradient descent for MSE classification

In order to investigate the performance of n-shot doubly stochastic gradient descent for MSE loss functions on a realistic task, we consider the problem of image classification. In particular, we consider a binary classification task defined by a data-set consisting of down-sampled grayscale 3's and 6's from the MNIST data-set. To be specific, each selected MNIST image has been down-sampled from 28×28 pixels to 8×8 pixels by first removing 6 pixels from all boundaries, and then removing every second row and every second column. Each image was then flattened into a 64 dimensional vector, and the pixel values were normalized so that the 2-norm of each image vector was 1. Each image was then encoded into an 8 qubit state via amplitude encoding [5, 47]. The training data-set consisted of 2000 3's (labelled with $y = 1$) and 2000 6's (labeled with $y = -1$), while the validation data-set consisted of 200 3's and 200 6's. We have utilized the same parameterized circuit architecture as in the previous section (with 6 qubits and 18 σ-blocks) followed by a measurement of Z_1, where given an image instance x the model has been defined via

$$f(x) = \begin{cases} 1 & \text{if } \langle Z_1 \rangle \geq 0 \\ -1 & \text{else .} \end{cases} \tag{68}$$

The loss function has been as per Eq. (38), with $O = Z_1$, and we investigated the performance of the estimator defined in Algorithm 4 – resulting in what we refer to as n-shot doubly stochastic gradient descent (DSGD) – for different *fixed* learning rates and values of n, as can be seen in Fig. 4. In particular, in order to provide a benchmark the top row of Fig. 4 shows the results for "singly"-stochastic gradient descent – i.e., mini-batch gradient descent with batch size 1 and *exact* expectation values (which, as discussed, is not practically feasible), while the bottom row of Fig. 4 shows the results for n-shot doubly stochastic gradient descent, also with batch size 1. An important thing to note is that because of the way the model is defined in Eq. (68), a high confidence (large shot number n) estimate of $\langle Z_1 \rangle$ is required in order to make a prediction with the model, even though, as we have discussed at length, such an estimate is not required to implement the doubly stochastic optimization algorithm.

Additionally, we note that in order to avoid overfitting, a typical supervised learning procedure involves dividing the optimization into "epochs" – a single pass of the optimization algorithm through the training data set – each of which is followed by an evaluation of the optimization procedure, and a subsequent determination of convergence, by calculating the model accuracy on the validation data-set [9]. In light of the prior observations concerning the shot numbers necessary for high-confidence model predictions, a natural training strategy in this setting is therefore to train the algorithm for a single epoch, using low shot number estimates and without making simultaneous predictions on the training data, before then evaluating the accuracy of the model on the (typically smaller) validation set, using large shot number estimates of the expectation value to make predictions. In order to reflect this strategy, and the information one would then have available in a practical implementation of this algorithm, for the "singly"-stochastic gradient

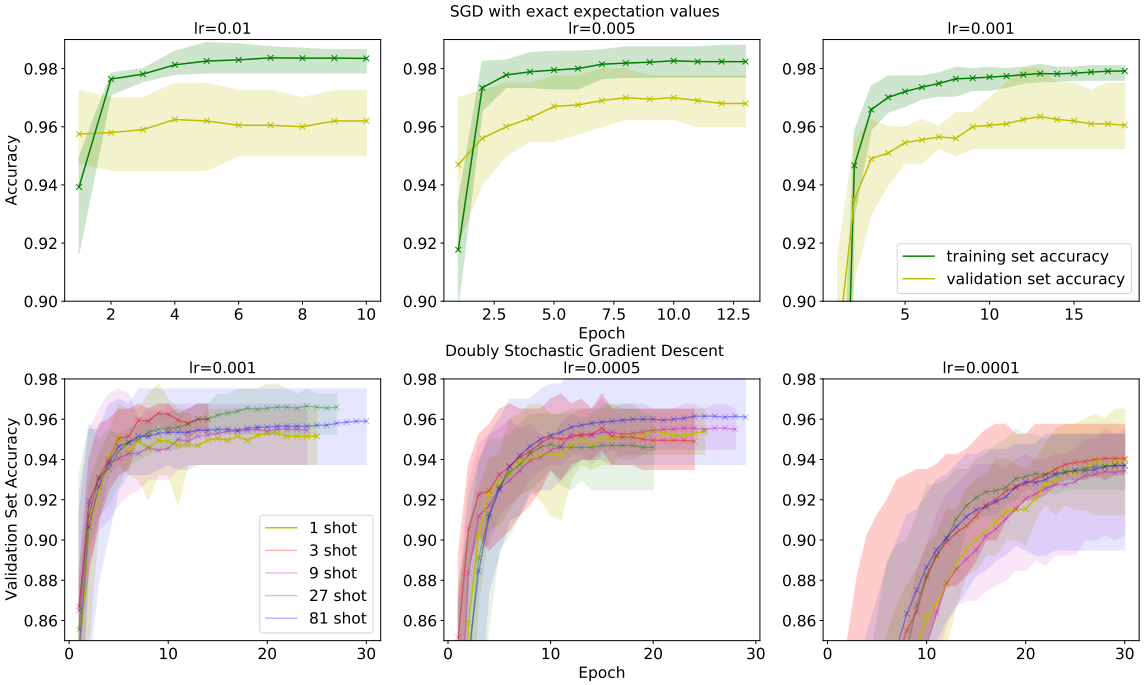

Figure 4: Results of downsampled MNIST binary (3 vs 6) classification experiments. The results from n-shot doubly stochastic gradient descent (i.e. using the function PARTIALDERIVATIVEESTIMATOR_4 from Algorithm 4) are shown in the lower row, and the results from exact expectation value stochastic gradient descent are shown in the upper row, both with batch-size = 1. All results were obtained using a fixed learning rate, displayed in the title of the respective subplot. All experiments were repeated 8 times, and the mean, max and min in each step is displayed. A particular experiment was decided to have converged if no improvement in the validation set accuracy was achieved in the previous 5 epochs, and for each shot number and learning rate pair the curve is plotted up until the epoch number which was reached by the slowest experiment to converge.

descent, in which exact expectation values are used in training, we have plotted both the training and validation set accuracy after each epoch, while for the n-shot doubly stochastic gradient descent we have plotted only the validation set accuracy, which was evaluated after each training epoch by using exact expectation values (as a proxy for the much higher shot numbers one would use in practice).

Once again, there are a variety of lessons to be taken from the results of Fig. 4. Firstly, as can be seen by comparing the columns, it is clear that as in the purely classical case, the learning rate plays a large role, and care should be taken in estimating this hyper-parameter. In these experiments, we see that for the learning rates $\alpha = 0.005$ and $\alpha = 0.0005$, the results from n-shot doubly and exact expectation value SGD do not differ significantly. As in the VQE setting, the n-shot DSGD algorithms require more *epochs* to converge, but as we have seen in the previous section, as each epoch is significantly cheaper for small n DSGD, it is again clear that single shot DSGD should definitely be used to rapidly find approximate solutions, before possibly increasing n, in conjunction with learning rate decay, to increase the quality of the solution. In this case the enhanced efficiency of small n estimators is amplified by the size of the data-set, as each epoch consists of a single optimization step per data-set instance, and thus the measurement efficiency improvements per optimization step should be multiplied by the size of the data-set.

9 Discussion and Conclusion

In light of the above analysis and results, it is possible to draw a variety of observations and conclusions. Firstly, in the context of hybrid quantum-classical optimization, in which both loss functions and their gradients can be expressed as functions of expectation values, *exact* gradient

descent is not possible. As such, all optimization schemes which in practice require the estimation of expectation values from a finite number of measurement results should be considered within the framework of *stochastic* gradient descent. In this work we have formalized this notion for VQE, QAOA and MSE classification problems, from which it is clear that previous approaches based on approximating expectation values with n measurement outcomes can be understood as instances of well defined stochastic gradient descent type algorithms, which are valid for all n, and in particular, even for $n = 1$. It is therefore not a-priori necessary to perform large numbers of measurements when implementing these algorithms in practice. Additionally, as is done classically in mini-batch stochastic gradient descent, whenever the loss function consists of a linear combination of terms which can be estimated in an unbiased way it is possible to obtain a new "doubly stochastic" optimizer by sampling over terms of the linear combination. In the context of hybrid quantum-classical optimization such linear combinations appear from a variety of sources, such as sums over local Hamiltonian terms, sums over parameter shift terms, or sums over data-set instances. Exploiting this insight has allowed us to define multiple doubly stochastic gradient descent algorithms, for practically relevant settings such as VQE, QAOA and quantum classifiers. Moreover, while convergence guarantees for realistic non-convex landscapes remain an open question, for simplified landscapes convergence guarantees are available for all of the algorithms discussed, which provide a minimum measure of confidence in their construction.

From these convergence guarantees it is clear that even in restricted settings the accuracy of the solution to which a given stochastic gradient descent algorithm will converge is dependent on the variance of the estimator for the gradient. It is natural to expect that both small n implementations of the algorithms we have introduced, as well as algorithms which involve samples over (possibly multiple) linear combinations, may not achieve solutions which are as fine-tuned as those obtained by large n algorithms. From the numerical experiments we have performed in Section 8 we see that this is indeed the case, however there are two important things to note. Firstly, adaptive heuristic learning rate schemes can successfully compensate for the variance of gradient estimators, and significantly improve the quality of solutions, even for very high variance estimators. Secondly, we see that on benchmark practical settings both small n stochastic gradient descent algorithms, and algorithms that sample over linear combinations of terms, can converge orders of magnitude faster, with respect to the total number of measurements required. As a result, even though the solutions converged to are not optimal, it is well advised to utilize small n SGD algorithms – possibly with linear combination sampling – as initializers which are able to rapidly find approximate solutions, using very few measurements relative to the number required by large n algorithms to reach the same point. Once convergence has been achieved for small n with sampling, optimization can be continued with a combination of learning rate decay and slowly increasing values of n, in order to fine tune the solutions.

Given the results presented here, there remain a variety of directions to explore. From an analytical perspective, it is naturally of interest to construct unbiased estimators for loss functions such as the *log-likelihood* and the *cross entropy*. However, as previously discussed, despite the absence of rigorous constructions for *unbiased* estimators for non-polynomial loss functions, one can use both efficient expectation value estimation and sampling over linear combinations to constructed biased estimators for these loss functions, whose performance as heuristic strategies may be worth investigating further, and for which one still may be able to design optimization algorithms with rigorous convergence guarantees [37]. Additionally, It is also desirable to obtain convergence guarantees for settings in which the Polyak-Lojasiewicz inequality is not satisfied, although this is expected to require significantly new techniques, which are sought after by the classical machine learning community. It should also be noted that the results and experiments presented here have all neglected the effects of noise, and understanding how realistic noise influences these results is of natural importance for developing optimization methods for use in conjunction with existing devices. One possible scenario is that realistic device noise will lead to estimators which remain unbiased, but with a larger variance. In this case, the strategies suggested here remain valid, however one may ultimately require smaller learning rates and larger shot numbers for fine-tuning solutions. If however realistic device noise introduces a bias into the estimators, then as mentioned earlier, one could still apply the strategies and optimization algorithms developed here as heuristics, and it may also be possible to design more complicated algorithms which still admit convergence guarantees [37]. In order to understand the role of noise more fully, it would therefore be of interest

both to study the performance of these algorithms on existing devices and to integrate realistic noise models into the analysis. In fact, along precisely these lines, very recently the authors of Ref. [48] have shown that counter to intuition noise may in fact be beneficial for stochastic gradient descent optimization, and it is hoped that this work stimulates further future research in these directions. Finally, it would be interesting to combine the algorithms presented here with additional techniques for reducing the number of circuit executions and forward passes. Very recent work has in fact introduced a variety of heuristic hyper-parameter adaptation schemes for measurement shot adaptation [28], and it will be of interest to investigate the extension of these techniques to the additional algorithms we have presented here. It is also worthwhile to further exploit structure such as sparsity in the gradients estimated.

On a higher level, we hope that the rigorous framework provided in this work contributes in a significant fashion to the growing body of analytical work on variational quantum-classical hybrid algorithms and algorithms with applications in quantum-enhanced machine learning, complementing a – to date – still largely empirical field of research. At the same time, we hope that our work provides further perspectives to find practical applications of near-term quantum devices, let this be *near-term quantum circuits* [49, 50] or instances of *programmable quantum simulators* [51–53], beyond showing conceptually interesting quantum advantages or "supremacy" [54–59]. Optimized schemes of the kind developed here may help in the search of finding such applications.

Acknowledgments

FW acknowledges funding from the DFG under Germany's Excellence Strategy MATH+: The Berlin Mathematics Research Center, EXC-2046/1 project ID: 390685689, and would like to thank Zacharias V. Fisches for insightful discussions on techniques in deep learning and numerical methods. RS acknowledges the support of the Alexander von Humboldt foundation, and would like to acknowledge the Quantum Excellence in Diversity (QuEDiver) initiative for facilitating a research visit of MS to the FU Berlin. JE has been supported by the BMWi (PlanQK), the ERC (TAQ), the Templeton Foundation, and the DFG (EI 519/14-1, EI 519/15-1, CRC 183) and MATH+. This work has also received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No. 817482 (PASQuanS).

References

[1] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New J. Phys.*, 18:23023, 2016. DOI: 10.1088/1367-2630/18/2/023023.

[2] J. Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, August 2018. DOI: 10.22331/q-2018-08-06-79.

[3] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Comm.*, 5(1), 2014. DOI: 10.1038/ncomms5213.

[4] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. arXiv:1411.4028, 2014.

[5] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020. DOI: 10.1103/PhysRevA.101.032308.

[6] E. Farhi and H. Neven. Classification with quantum neural networks on near term processors. 2018. arxiv:1802.06002.

[7] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini. Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001, 2019. DOI: 10.1088/2058-9565/ab4eb5.

[8] D. Zhu, N. M. Linke, M. Benedetti, K. A. Landsman, N. H. Nguyen, C. H. Alderete, A. Perdomo-Ortiz, N. Korda, A. Garfoot, C. Brecque, et al. Training of quantum circuits on a hybrid quantum computer. *Science advances*, 5(10):eaaw9918, 2019. DOI: 10.1126/sciadv.aaw9918.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
[10] 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*, 549(7671):242–246, 2017. DOI: 10.1038/nature23879.
[11] A. Harrow and J. Napp. Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. *arXiv:1901.05374*, 2019.
[12] A. Gilyén, S. Arunachalam, and N. 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. SIAM, 2019. DOI: 10.1137/1.9781611975482.87.
[13] G. Verdon, J. Pye, and M. Broughton. A universal training algorithm for quantum deep learning. 2018. arXiv:1806.09729.
[14] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Szva, and N. Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. 2018. arXiv:1811.04968.
[15] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Phys. Rev. A*, 98(3):32309, 2018. DOI: 10.1103/PhysRevA.98.032309.
[16] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran. Evaluating analytic gradients on quantum hardware. *Phys. Rev. A*, 99(3):32331, 2019. DOI: 10.1103/PhysRevA.99.032331.
[17] A. Kandala, A. Mezzcapo, K. Temme, M. Takita, M. Brink, J. W. Chow, and J. M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549:242, 2017. DOI: 10.1038/nature23879.
[18] V. Leyton-Ortega, A. Perdomo-Ortiz, and O. Perdomo. Robust implementation of generative modeling with parametrized quantum circuits. *arXiv:1901.08047*, 2019.
[19] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747): 209–212, 2019. DOI: 10.1038/s41586-019-0980-2.
[20] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107057132, 9781107057135.
[21] H. Karimi, J. Nutini, and M. Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016. DOI: 10.1007/978-3-319-46128-1˙50.
[22] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. DOI: 10.1007/978-3-7908-2604-3˙16.
[23] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168, 2008.
[24] R. Kleinberg, Y. Li, and Y. Yuan. An alternative view: When does sgd escape local minima? 2018. arXiv:1802.06175.
[25] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.
[26] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.
[27] C.-L. Li and B. Póczos. Utilize old coordinates: Faster doubly stochastic gradients for kernel methods. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI16, page 467476, Arlington, Virginia, USA, 2016. AUAI Press. ISBN 9780996643115.
[28] J. M. Kübler, A. Arrasmith, L. Cincio, and P. J. Coles. An adaptive optimizer for measurement-frugal variational algorithms. *Quantum*, 4:263, 2020. DOI: 10.22331/q-2020-05-11-263.
[29] L. Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91 (8):12, 1991.
[30] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

[31] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, pages 693–701, 2011.

[32] X. Li and F. Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 983–992. PMLR, 16–18 Apr 2019. URL http://proceedings.mlr.press/v89/li19c.html.

[33] E. Campbell. Random compiler for fast hamiltonian simulation. *Phys. Rev. Lett.*, 123:70503, 2019. DOI: 10.1103/PhysRevLett.123.070503.

[34] P. Gokhale, O. Angiuli, Y. Ding, K. Gui, T. Tomesh, M. Suchara, M. Martonosi, and F. T. Chong. Minimizing state preparations in variational quantum eigensolver by partitioning into commuting families. 2019. arXiv:1907.13623.

[35] S. Raeisi, N. Wiebe, and B. C. Sanders. Quantum-circuit design for efficient simulations of many-body quantum dynamics. *New J. Phys.*, 14:103017, 2012. DOI: 10.1088/1367-2630/14/10/103017.

[36] A. J. Lee. *U-statistics: Theory and Practice*. Routledge, 2019.

[37] J. Chen and R. Luss. Stochastic gradient descent with biased but consistent gradient estimators. *arXiv:1807.11880*, 2018.

[38] P. H. Nguyen, L. M. Nguyen, and M. van Dijk. Tight dimension independent lower bound on optimal expected convergence rate for diminishing step sizes in sgd. 2018. arXiv:1810.04723.

[39] M. M. Wolf. Mathematical foundations of supervised learning. https://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MA4801_2018S/ML_notes_main.pdf [Online; accessed 27-September-2019].

[40] H. H. Sohrab. *Basic real analysis*, volume 231. Birkhäuser, Basel, 2003. DOI: 10.1007/978-1-4939-1841-6.

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[42] https://github.com/frederikwilde/qradient.

[43] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Entropy scaling and simulability by matrix product states. *Phys. Rev. Lett.*, 100:30504, January 2008. DOI: 10.1103/PhysRevLett.100.030504.

[44] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le. Don't decay the learning rate, increase the batch size. *arXiv:1711.00489*, 2017.

[45] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 4th edition, 2007. ISBN 3540718435.

[46] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *Physical Review X*, 10(2):021067, 2020. DOI: PhysRevX.10.021067.

[47] M. Schuld and F. Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018. DOI: 10.1007/978-3-319-96424-9.

[48] L. Gentini, A. Cuccoli, S. Pirandola, P. Verrucchi, and L. Banchi. Noise-assisted variational hybrid quantum-classical optimization. *arXiv:1912.06744*, 2019.

[49] E. Conover. Google moves toward quantum supremacy with 72-qubit computer. *ScienceNews*, 193:13, 2018.

[50] C. Vu. IBM announces advances to IBM quantum systems and ecosystem, November 2017. IBM press release.

[51] C. Kokail, C. Maier, R. van Bijnen, T. Brydges, M. K. Joshi, P. Jurcevic, C. A. Muschik, P. Silvi, R. Blatt, C. F. Roos, and P. Zoller. Self-verifying variational quantum simulation of the lattice schwinger model. *Nature*, 569:355, 2019. DOI: 10.1038/s41586-019-1177-4.

[52] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, V. Vuletic, and M. D. Lukin. Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551:579–584, 2017. DOI: 10.1038/nature24622.

[53] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe. Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator. *Nature*, 551:601–604, 2017. DOI: 10.1038/nature24654.

[54] M. J. Bremner, A. Montanaro, and D. J. Shepherd. Average-case complexity versus approximate simulation of commuting quantum computations. *Phys. Rev. Lett.*, 117:80501, August 2016. DOI: 10.1103/PhysRevLett.117.080501.

[55] S. Aaronson and A. Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342, 2011. DOI: 10.1145/1993636.1993682.

[56] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, R. Barends, B. Burkett, Y. Chen, and Z. Chen. A blueprint for demonstrating quantum supremacy with superconducting qubits. *Science*, 360:195–199, 2018. DOI: 10.1126/science.aao4309.

[57] J. Bermejo-Vega, D. Hangleiter, M. Schwarz, R. Raussendorf, and J. Eisert. Architectures for quantum simulation showing a quantum speedup. *Phys. Rev. X*, 8:21010, 2018. DOI: 10.1103/PhysRevX.8.021010.

[58] X. Gao, S.-T. Wang, and L.-M. Duan. Quantum supremacy for simulating a translation-invariant ising spin model. *Phys. Rev. Lett.*, 118:40502, 2017. DOI: 10.1103/PhysRevLett.118.040502.

[59] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. DOI: 10.1038/s41586-019-1666-5.

[60] Scientific co2nduct. online. URL `https://scientific-conduct.github.io`.

A Proof of Theorem 2

For completeness, in this section we provide a proof of Theorem 2 via a sequence of Lemmas. We begin with the following standard result for univariate functions, which follows directly from the mean value theorem, and whose proof can be found in [40]:

Lemma 3 (Lipschitz continuity of univariate functions on closed domains)**.** *Given some function $f : \mathbb{R} \to \mathbb{R}$, if f is continuously differentiable, then for any closed interval $[a, b] \subset \mathbb{R}$, the function $f : [a, b] \to \mathbb{R}$ is L-Lipschitz, with $L = \sup_{x\in[a,b]} |f'(x)|$.*

In order to continue, we introduce the following definition:

Definition 4. *Given some function $f : [a, b]^M \to \mathbb{R}$, we say that f is Lipschitz continuous with respect to its j'th argument, if for all $\boldsymbol{y} \in [a, b]^{M-1}$ the function $g_{j,\boldsymbol{y}} : [a, b] \to \mathbb{R}$ is Lipschitz continuous, where*

$$g_{j,\boldsymbol{y}}(x) := f(y_1, \ldots, y_{j-1}, x, y_j, \ldots y_{M-1}). \tag{69}$$

Note that if f is Lipschitz continuous with respect to its j'th argument, this implies that for all $x_1, x_2 \in [a, b]$ and for all $y \in [a, b]^{M-1}$ there exists some positive constant $L_{j,\boldsymbol{y}}$ such that

$$|g_{j,\boldsymbol{y}}(x_2) - g_{j,\boldsymbol{y}}(x_2)| \leq L_{j,\boldsymbol{y}}|x_2 - x_1|. \tag{70}$$

In light of this, we define $L_j := \sup_{\boldsymbol{y}} L_{j,\boldsymbol{y}}$, and it follows that for all $x_1, x_2 \in [a, b]$ and for all $y \in [a, b]^{M-1}$

$$|g_{j,\boldsymbol{y}}(x_2) - g_{j,\boldsymbol{y}}(x_2)| \leq L_j|x_2 - x_1|. \tag{71}$$

Given, this we are then able to state the following Lemma, providing a sufficient condition for Lipschitz continuity of multivariate functions:

Lemma 4. *Given some $f : [a, b]^M \to \mathbb{R}$, if f is Lipschitz continuous with respect to all of its arguments, then f is L-Lipschitz continuous, with $L = \sqrt{M}(\max_j L_j)$.*

Proof. For all $\boldsymbol{x}, \boldsymbol{y} \in [a, b]^M$ we have that

$$\begin{aligned} f(\boldsymbol{x}) - f(\boldsymbol{y}) = {} & f(x_1, \ldots, x_M) - f(y_1, x_2, \ldots, x_M) + \ldots \\ & + f(y_1, \ldots y_{j-1}, x_j, \ldots, x_M) - f(y_1, \ldots y_j, x_{j+1}, \ldots x_M) + \ldots \\ & + f(y_1, \ldots y_{M-1}, x_M) - f(y_1, \ldots y_M). \end{aligned} \tag{72}$$

By the triangle inequality, this then gives us

$$\begin{aligned} |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq {} & |f(x_1, \ldots, x_M) - f(y_1, x_2, \ldots, x_M)| + \ldots \\ & + |f(y_1, \ldots y_{j-1}, x_j, \ldots, x_M) - f(y_1, \ldots y_j, x_{j+1}, \ldots x_M)| + \ldots \\ & + |f(y_1, \ldots y_{M-1}, x_M) - f(y_1, \ldots y_M)|. \end{aligned} \tag{73}$$

Furthermore, under the assumption that f is Lipschitz continuous with respect to all of its arguments, we see that

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq L_1|x_1 - y_1| + \ldots + L_M|x_M - y_M| \tag{74}$$

$$\leq (\max_j L_j)(\sum_{j=1}^{M} |x_j - y_j|) \tag{75}$$

$$= (\max_j L_j)||\boldsymbol{x} - \boldsymbol{y}||_1 \tag{76}$$

$$\leq \sqrt{M}(\max_j L_j)||\boldsymbol{x} - \boldsymbol{y}||_2, \tag{77}$$

where the last line follows from the fact that for $\boldsymbol{x} \in \mathbb{R}^M$ one has that $||\boldsymbol{x}||_1 \leq \sqrt{M}||\boldsymbol{x}||_2$. □

Putting together the previous statements we then obtain the following Lemma.

Lemma 5. *Given some function $f : \mathbb{R}^M \to \mathbb{R}$, if all partial derivatives of f are continuous, then for any $a, b \in \mathbb{R}$ the function $f : [a, b]^M \to \mathbb{R}$ is L-Lipschitz continuous with*

$$L = \sqrt{M} \left[\max_{j \in \{1,\ldots,M\}} \sup_{\boldsymbol{x} \in [a,b]^M} \left| \frac{\partial f(\boldsymbol{x})}{\partial x_j} \right| \right] \tag{78}$$

Proof. For all j, and for all $y \in [a, b]^{M-1}$, one can apply Lemma 3 to the function $g_{j,\boldsymbol{y}}$, from which one obtains that $g_{j,\boldsymbol{y}}$ is $L_{j,\boldsymbol{y}}$-Lipschitz continuous with

$$L_{j,\boldsymbol{y}} = \sup_{x \in [a,b]} \left| \frac{\partial f(y_1, \ldots y_{j-1}, x, y_j, \ldots y_{M-1})}{\partial x_j} \right|. \tag{79}$$

The statement then follows from Lemma 4 and the definition of L_j. □

Finally, given Lemma 5 the proof of Theorem 2 follows straightforwardly:

Proof (Theorem 2). For all j we have that the partial derivatives

$$\frac{\partial}{\partial \theta_j} f(\boldsymbol{\theta}) = \langle \mathbf{0} | \left(\frac{\partial U^\dagger(\boldsymbol{\theta})}{\partial \theta_j} \right) O U(\boldsymbol{\theta}) | \mathbf{0} \rangle + \langle \mathbf{0} | U^\dagger(\theta) O \left(\frac{\partial U(\boldsymbol{\theta})}{\partial \theta_i} \right) | \mathbf{0} \rangle \tag{80}$$

exist and are continuous on $\mathbb{R}^M$, and therefore the statement of the theorem follows from Lemma 5. □

B Parameterized circuit and optimization details

Fig. 5 below shows the parameterized quantum circuit $U(\boldsymbol{\theta})$ used for both the VQE and MSE quantum classifier experiments, as discussed in Section 8. As can be seen, the circuit is constructed from layers of Pauli rotations, interleaved with $CNOT$ ladders, and consists of 400 free parameters.

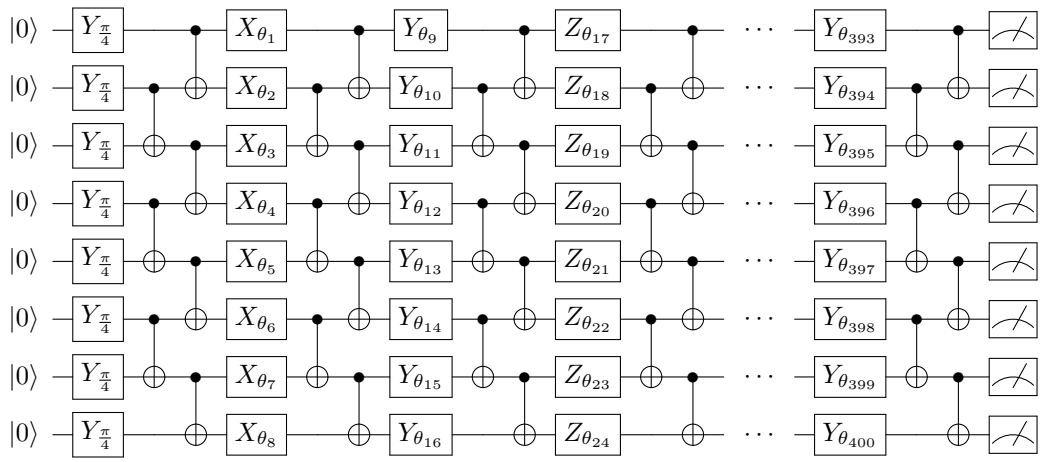

Figure 5: Parameterized quantum circuit used for the VQE and MSE classifier experiments described in Section 8. The notation σ_θ is used to represent the single qubit $e^{-i\theta\sigma}$ gate, for some Pauli operator $\sigma \in [X, Y, Z]$.

C CO_2 Emission Table

The table below summarizes the estimated carbon cost of this work, including both numerical simulations and air-travel for collaboration purposes. Estimations have been calculated using the examples of Scientific CO_2nduct [60], and are correct to the best of our knowledge.

Numerical simulations	
Total Kernel Hours [h]	14300
Thermal Design Power Per Kernel [W]	5.75
Total Energy Consumption Simulations [kWh]	85.1
Average Emission Of CO_2 In Germany [kg/kWh]	0.56
Total CO_2 Emission For Numerical Simulations [kg]	47.6
Were The Emissions Offset?	**Yes**
Transport	
Total CO_2 Emission For Transport [kg]	4804
Were The Emissions Offset?	**Yes**
Total CO_2 Emission [kg]	4847.5