Chapter 8

Resource frugal optimizer for quantum machine learning

Variational quantum machine learning algorithms have the potential to solve practical problems on real hardware, particularly when involving quantum data. In Chapter 7, we assessed the importance of hyperparameters of quantum neural networks on classical data and in idealized simulation settings. However, training these algorithms can be challenging and calls for tailored optimization procedures. Specifically, QML applications can require a large shot-count overhead due to the large datasets involved. In this chapter[1], we advocate for simultaneous random sampling over both the dataset as well as the measurement operators that define the loss function. We consider a highly general loss function that encompasses many QML applications, and we show how to construct an unbiased estimator of its gradient. This allows us to propose a shot-frugal gradient descent optimizer called Refoqus (REsource Frugal Optimizer for QUantum Stochastic gradient descent). Our numerics indicate that Refoqus can save several orders of magnitude in shot cost, even relative to optimizers that sample over measurement operators alone.

[1]Contents of this chapter are based on [136]; Charles Moussa, Max Hunter Gordon, Michal Baczyk, M. Cerezo, Lukasz Cincio, and Patrick J. Coles. Resource frugal optimizer for quantum machine learning. *arXiv:2211.04965*, 2022.

8.1 Introduction

The field of quantum machine learning (QML) revolves mostly around variational methods, which involve classically training a parameterized quantum model. Variational QML, henceforth referred to as QML for simplicity, is indeed a leading candidate for implementing QML in the near term. However, it has faced various sorts of trainability issues.

Exponentially vanishing gradients, known as barren plateaus [125, 42, 87, 88, 176, 122, 190, 9, 153], as well as the prevalence of local minima [23, 7] are two issues that can impact the complexity of the training process. Quantum hardware noise also impacts trainability [197, 184]. All of these issues contribute to increasing the number of shots and iterations required to minimize the QML loss function. Indeed, a detailed shot-cost analysis has painted a concerning picture [200].

It is therefore clear that QML requires careful frugality in terms of the resources expended during the optimization process. Indeed, novel optimizers have been developed in response to these challenges. Quantum-aware optimizers aim to replace off-the-shelf classical optimizers with ones that are specifically tailored to the quantum setting [185, 107, 140]. Shot-frugal optimizers [109, 72, 189] have been proposed in the context of variational quantum eigensolver (VQE), whereby one can sample over terms in the Hamiltonian instead of measuring every term [10]. While significant progress has been made on such optimizers, particularly for VQE, we argue that very little work has been done to specifically tailor optimizers to the QML setting. The cost functions in QML go well beyond those used in VQE and hence QML requires more general tools.

In this work, we generalize previous shot-frugal and iteration-frugal optimizers, such as those in Refs. [109, 72, 10], by extending them to the QML setting. Specifically, we allow for random, weighted sampling over both the input and the output of the loss function estimation circuit. In other words, and as shown in Fig. 8.1, we allow for sampling over the dataset as well as over the measurement operators used to define the loss function. Our sampling approach allows us to unlock the frugality (i.e., to achieve the full potential) of adaptive stochastic gradient descent optimizers, such as iCANS [109] and gCANS [72].

We discuss how our approach applies to various QML applications such as perceptron-based quantum neural networks [17, 176], quantum autoencoders [161], variational quantum principal component analysis (PCA) [110, 41], and classifiers that employ the mean-squared-error loss function [170, 189]. Each of these applications can be unified under one umbrella by considering a generic loss function with a highly general form. Thus we state our main results for this generic loss function. We establish an unbiased estimator for this loss function and its gradient. In turn, this allows us to provide convergence guarantees for certain optimization routines, like stochastic gradient descent. Furthermore, we show that for this general loss function one can use the form of the estimator to inform a strategy that distributes shots to obtain the best shot frugal estimates.

Finally, we numerically investigate the performance of our new optimization approach, which we call Refoqus (REsource Frugal Optimizer for QUantum Stochastic gradient descent). For a quantum PCA task, Refoqus significantly outperforms state-of-the-art optimizers in terms of the shot resources required. Refoqus even outperforms Rosalin [10] - a shot-frugal optimizer that samples only over measurement operators. Hence, Refoqus will be a crucial tool to minimize the number of shots and iterations required in near-term QML implementations.

8.2 Background

8.2.1 Stochastic Gradient Descent

One of the most popular optimization approaches is gradient descent, which involves the following update rule for the parameter vector:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \alpha \boldsymbol{\nabla} \mathcal{L}(\boldsymbol{\theta}^{(t)})\,. \tag{8.1}$$

Here, $\mathcal{L}$ is the loss function, α is the learning rate, and $\boldsymbol{\theta}^{(t)}$ is the parameter vector at iteration t.

Oftentimes one only has access to noisy estimates of the gradient $\boldsymbol{\nabla}\mathcal{L}$, in which case the optimizer is called stochastic gradient descent. In the quantum setting, this situation arises due to shot noise or noise due to sampling from terms in some expansion of the gradient.

8.2.2 Parameter Shift Rule

Estimating the gradient is clearly an essential step in stochastic gradient descent. For this purpose, one useful tool that is often employed in the quantum case is the so-called parameter shift rule [131, 169]. We emphasize that several assumptions go into this rule. Specifically, suppose we assume that the quantum circuit ansatz $U(\boldsymbol{\theta})$ can

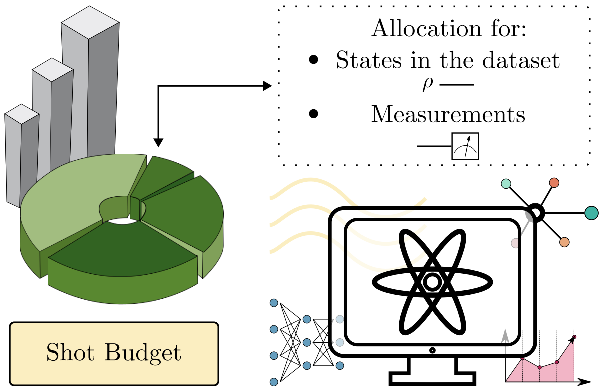


Figure 8.1: **Schematic illustration of Refoqus.** Measurements, or shots, are a precious (and expensive) resource in quantum computing. As such, they should be used sparingly and only when absolutely necessary. This is particularly important in variational QML methods where training a model requires continuously calling a quantum device to estimate the loss function or its gradients. Refoqus provides a shot-frugal optimization paradigm where shots are allocated by sampling over the input data and the measurement operators.

be expressed as $U(\boldsymbol{\theta}) = \prod_x e^{-i\theta_x\sigma_x} W_x$ where W_x are unparametrized unitaries, and where σ_x are Pauli operators. Moreover, we consider the case when the loss function has the simple form $\mathcal{L}(\boldsymbol{\theta}) = \langle 0|U^\dagger(\boldsymbol{\theta}) H U(\boldsymbol{\theta})|0\rangle$ for some Hermitian operator H. (Note that we will consider more complicated loss functions in this work, and hence we are just stating this special case for background information.) In this case, the parameter shift rule gives

$$\partial_x \mathcal{L}(\boldsymbol{\theta}) := \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x} = \frac{\mathcal{L}(\boldsymbol{\theta} + \frac{\pi}{2}\boldsymbol{\delta}_x) - \mathcal{L}(\boldsymbol{\theta} - \frac{\pi}{2}\boldsymbol{\delta}_x)}{2}, \tag{8.2}$$

where $\boldsymbol{\delta}_x$ is a unit vector with a one on the x-th component. Equation (8.2) allows one to estimate the gradient by estimating the loss function as specific points on the landscape. Hence it simplifies the procedure to estimate the gradient.

8.2.3 Lipschitz continuity

The loss function $\mathcal{L}$ is called Lipschitz continuous if there is some Lipschitz constant $L \geq 0$ that satisfies the bound

$$\|\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}_a) - \boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta}_b)\| \leq L\|\boldsymbol{\theta}_a - \boldsymbol{\theta}_b\|, \tag{8.3}$$

for all $\boldsymbol{\theta}_a, \boldsymbol{\theta}_b \in \mathrm{dom}(\mathcal{L})$, where $\|\cdot\|$ is the ℓ_2 norm. This property provides a recipe for choosing an appropriate learning rate, α. Specifically, if (8.3) holds and we have access to the exact gradient, then choosing $\alpha \leq 2/L$ is sufficient to guarantee convergence using the update rule in (8.1).

8.2.4 iCANS

Inspired by an adaptive batch size optimizer used in classical machine learning [13], the iCANS (individual coupled adaptive number of shots) optimizer [109] was introduced as an adaptive method for stochastic gradient in the context of the variational quantum eigensolver. It allows the number of shots per partial derivative (i.e., gradient component) to vary individually, hence the name iCANS.

Consider the gain (i.e., the decrease in the loss function), denoted $\mathcal{G}_x$, associated with updating the x-th parameter θ_x. The goal of iCANS is to maximize the expected gain per shot. That is, for each individual partial derivative, we maximize the shot efficiency:

$$\gamma_x := \frac{\mathbb{E}[\mathcal{G}_x]}{s_x}; \quad x = 1, \ldots, d\,, \tag{8.4}$$

where s_x is the shot allocation for gradient component x and d is the number of gradient components. Solving for the optimal shot allocation gives:

$$s_x = \frac{2L\alpha}{2 - L\alpha}\frac{\sigma_x^2}{g_x^2}. \tag{8.5}$$

Here, g_x is an unbiased estimator for the x-th gradient component, and σ_x is the standard deviation of a random variable X_x whose sample mean is g_x. While iCANS often heuristically outperforms other methods, it can have instabilities.

8.2.5 gCANS

Recently, a potential improvement over iCANS was introduced called gCANS (global coupled adaptive number of shots) [72]. gCANS considers the expected gain $\mathbb{E}[\mathcal{G}]$ over the entire gradient vector. Then the goal is to maximize the shot efficiency

$$\gamma := \frac{\mathbb{E}[\mathcal{G}]}{\sum_{x=1}^{d} s_x}\,, \tag{8.6}$$

where the sum $\sum_{x=1}^{d} s_x$ goes over all components of the gradient. Solving for the optimal shot count then gives:

$$s_x = \frac{2L\alpha}{2 - L\alpha} \frac{\sigma_x \sum_{x'=1}^{d} \sigma_{x'}}{\|\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta})\|^2}. \tag{8.7}$$

(Note that an exponential moving average is used to estimate σ_x and $\|\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta})\|^2$ as their true value is not accessible.) It was proven that gCANS achieves geometric convergence to the optimum, often reducing the number of shots spent for comparable solutions against its predecessor iCANS.

8.2.6 Rosalin

Shot-frugal optimizers like iCANS and gCANS rely on having an unbiased estimator for the gradient or its components. However, this typically places a hard floor on how many shots must be allocated at each iteration, i.e., going below this floor could result in a biased estimator. This is because the measurement operator H is typically composed of multiple non-commuting terms, each of which must be measured individually. Each of these terms must receive some shot allocation to avoid having a biased estimator. However, having this hard floor on the shot requirement is antithetical to the shot-frugal nature of iCANS and gCANS, and ultimately it handicaps these optimizers' ability to achieve shot frugality.

This issue inspired a recent proposal called Rosalin (Random Operator Sampling for Adaptive Learning with Individual Number of shots) [10]. Rosalin employs weighted random sampling of operators in the measurement operator $H = \sum_j c_j H_j$, which allows one to achieve an unbiased estimator without a hard floor on the shot requirement. (Even a single shot, provided that it is randomly allocated according to an appropriate probability distribution, can lead to an unbiased estimator.) When combined with the shot allocation methods from iCANS or gCANS, the operator sampling methods in Rosalin were shown to be extremely powerful in the context of molecular chemistry problems, which often have a large number of terms in H.

We remark that Ref. [10] considered several sampling strategies. Given a budget of s_{tot} shots and N terms, a simple strategy is to distribute shots per term equally ($s_j = s_{\text{tot}}/N$) - referred as uniform deterministic sampling (UDS). Defining $M = \sum_j |c_j|$, one can also use weighted deterministic sampling (WDS) where the shots are proportionally distributed: $s_j = s_{\text{tot}} * \frac{|c_j|}{M}$. One can add randomness by using $p_j = \frac{|c_j|}{M}$ to define a (non-uniform) probability distribution to select which term should

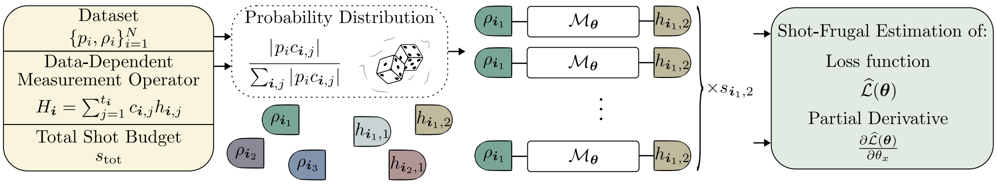

Dataset
$\{p_i, \rho_i\}_{i=1}^N$
Data-Dependent Measurement Operator
$H_{\boldsymbol{i}} = \sum_{j=1}^{t_i} c_{i,j} h_{\boldsymbol{i},j}$
Total Shot Budget
s_{tot}
Probability Distribution
$\frac{|p_i c_{\boldsymbol{i},j}|}{\sum_{\boldsymbol{i},j} |p_i c_{\boldsymbol{i},j}|}$
ρ_{i_1}
ρ_{i_2}
ρ_{i_3}
$h_{i_1,1}$
$h_{i_1,2}$
$h_{i_2,1}$
$\mathcal{M}_{\boldsymbol{\theta}}$
$\times s_{i_1,2}$
Shot-Frugal Estimation of:
Loss function
$\widehat{\mathcal{L}}(\boldsymbol{\theta})$
Partial Derivative
$\frac{\partial \widehat{\mathcal{L}}(\boldsymbol{\theta})}{\partial \theta_x}$

Figure 8.2: **Illustration of a generic variational QML framework.** Given a dataset of quantum states $\mathcal{S}$, the input states to the QML model are tensor product states of the form $\rho_{\boldsymbol{i}} = \rho_{i_1} \otimes ... \otimes \rho_{i_m}$. The number of samples m depends on the QML task at hand. The parameterized quantum model denoted $\mathcal{M}_{\boldsymbol{\theta}}$, acts on m copies of the input Hilbert space. Finally, an operator $H_{\boldsymbol{i}}$ is measured to estimate a quantity to be used when evaluating a loss function $\mathcal{L}(\boldsymbol{\theta})$. The latter evaluation is then inputted to a classical optimizer that proposes new parameters $\boldsymbol{\theta}$ in order to minimize the loss. Hence, one can repeat the quantum-classical loop of evaluations and updates until the desired stopping criteria are satisfied.

be measured. This is referred to as weighted random sampling (WRS). Finally, there exists a hybrid approach where one combines WDS with WRS - referred to as weighted hybrid sampling (WHS). Ref. [10] found that the WRS and WHS strategies performed similarly and they both significantly outperformed the UDS and WDS strategies on molecular ground state problems. Because of these results, we choose to focus on the WRS strategy in our work here.

While Rosalin was designed for chemistry problems, it was not designed for QML, where the number of terms in H is not the only consideration. As discussed below, QML problems involve a (potentially large) dataset of input states. Each input state requires a separate quantum circuit, and hence we are back to the situation of having a hard floor on the shots required due to these multiple input states. This ultimately provides the motivation for our work, which can be viewed as a generalization of Rosalin to the setting of QML.

8.3 Framework

8.3.1 Generic Variational QML Framework

Let us present our general framework for discussing (variational) QML methods; see Fig. 8.2 for an illustration. This framework is meant to unify multiple literature QML algorithms under one umbrella. We discuss how specific literature algorithms are special cases of this framework in Section 8.3.2.

In a generic QML setting, one has a training dataset composed of quantum states:

$$\mathcal{S} = \{\rho_i\}_{i=1}^N , \tag{8.8}$$

where each ρ_i is a trace-one positive semi-definite matrix, i.e., a density matrix. Each of these training states may come with an associated probability, with the associated probability distribution denoted as

$$\mathcal{P} = \{p_i\}_{i=1}^N . \tag{8.9}$$

In variational QML, one trains a parameterized quantum model, which we write as $\mathcal{M}_{\boldsymbol{\theta}}$ for some set of parameters $\boldsymbol{\theta}$. With a large degree of generality, we can assume that $\mathcal{M}_{\boldsymbol{\theta}}$ is a linear, completely-positive map. In general, $\mathcal{M}_{\boldsymbol{\theta}}$ could act on multiple copies (m copies) of the input Hilbert space. (Multiple copies allow for non-linear operations on the dataset, which can be important in certain classification tasks.) Hence we allow for the output of this action to be of the form:

$$\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}}) = \mathcal{M}_{\boldsymbol{\theta}}(\rho_{i_1} \otimes ... \otimes \rho_{i_m}) = \mathcal{M}_{\boldsymbol{\theta}}\left(\bigotimes_{\alpha=1}^{m} \rho_{i_\alpha}\right) , \tag{8.10}$$

where we employ the notation $\rho_{\boldsymbol{i}} := \rho_{i_1} \otimes ... \otimes \rho_{i_m}$. Given that we are allowing for multiple copies of the input space, one can define an effective dataset $\mathcal{S}_m$ composed of the tensor product of m states in $\mathcal{S}$ and an effective probability distribution $\mathcal{P}_m$.

A QML loss function is then defined in an operational manner so that it could be estimated on a quantum device. This involves considering the aforementioned mathematical objects as well as a measurement operator, or a set of measurement operators. We allow for the measurement operator to be tailored to the input state. Hence we write $H_{\boldsymbol{i}}$, with $\boldsymbol{i} = \{i_1, ..., i_m\}$, as the measurement operator when the input state on m copies of the Hilbert space is $\rho_{\boldsymbol{i}} = \rho_{i_1} \otimes ... \otimes \rho_{i_m}$. Moreover, each measurement operator can be decomposed into a linear combination of Hermitian matrices that can be directly measured:

$$H_{\boldsymbol{i}} = \sum_{j=1}^{t_i} c_{\boldsymbol{i},j} h_{\boldsymbol{i},j} . \tag{8.11}$$

Generically, we could write the loss function as an average over training states chosen from the effective dataset D_m:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta}))\,. \tag{8.12}$$

Here, ℓ is an application-dependent function whose input is a measurable expectation value $E_{\boldsymbol{i}}(\boldsymbol{\theta})$. Specifically, this expectation value is associated with the $\rho_{\boldsymbol{i}}$ input state, with the form

$$E_{\boldsymbol{i}}(\boldsymbol{\theta}) = \mathrm{Tr}[\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}})H_{\boldsymbol{i}}]\,. \tag{8.13}$$

There are many possible forms for the function ℓ. However, there are multiple QML proposals in the literature that involve a simple linear form:

$$\ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = E_{\boldsymbol{i}}(\boldsymbol{\theta})\,, \tag{8.14}$$

and in this case, we refer to the overall loss function as a "linear loss function". Alternatively, non-linear functions are also possible, and we also consider polynomial functions of the form:

$$\ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = \sum_{z=0}^{D} a_z [E_{\boldsymbol{i}}(\boldsymbol{\theta})]^z\,, \tag{8.15}$$

where D is the degree of the polynomial. In this case, we refer to the loss as a "polynomial loss function".

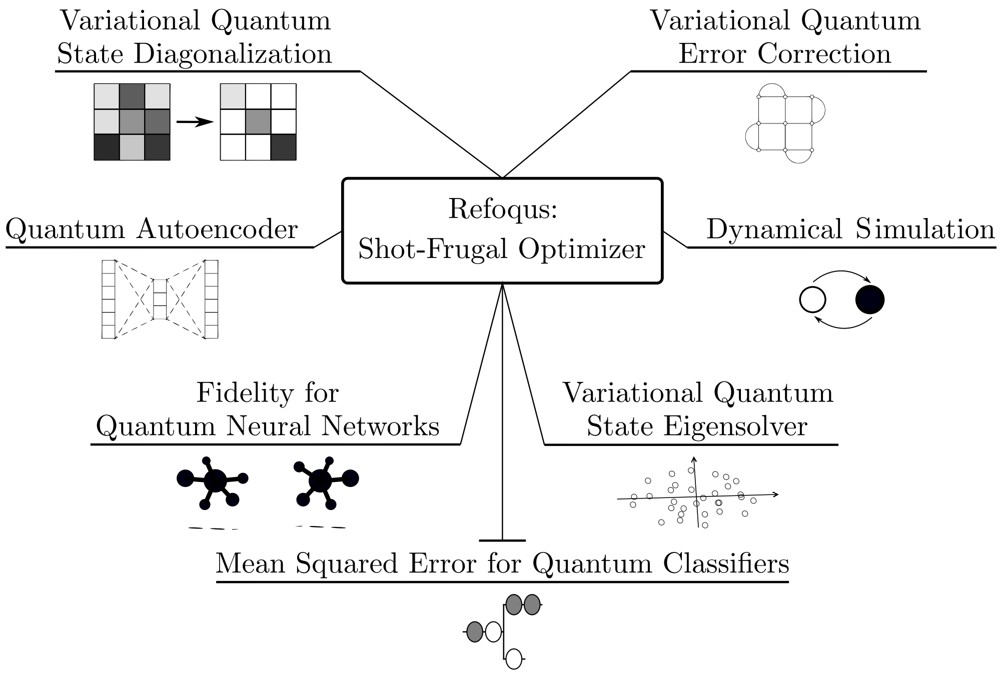
8.3.2 Examples of QML loss functions

Now let us illustrate how various QML loss functions proposed in the literature fall under the previous framework. Crucially, as shown in Fig. 8.3 these loss functions can be used for a wide range of QML tasks.

Variational quantum error correction

Variational quantum algorithms to learn device-tailored quantum error correction codes were discussed in Refs. [95, 45]. The loss function involves evaluating the input-output fidelity of the code, averaged over a set of input states. The input state is fed into the encoder $\mathcal{E}_{\boldsymbol{\theta}_1}$, then the noise channel $\mathcal{N}$ acts, followed by the decoder $\mathcal{D}_{\boldsymbol{\theta}_2}$. The concatenation of these three channels can be viewed as the overall channel

$$\mathcal{M}_{\boldsymbol{\theta}} = \mathcal{D}_{\boldsymbol{\theta}_2} \circ \mathcal{N} \circ \mathcal{E}_{\boldsymbol{\theta}_1}\,, \tag{8.16}$$



Variational Quantum State Diagonalization
Variational Quantum Error Correction
Quantum Autoencoder
Refoqus: Shot-Frugal Optimizer
Dynamical Simulation
Fidelity for Quantum Neural Networks
Variational Quantum State Eigensolver
Mean Squared Error for Quantum Classifiers

Figure 8.3: **Applications benefiting from Refoqus**. Many variational quantum algorithms employ training data, and many QML models (which naturally analyze datasets) are variational in nature. We give examples of both of these cases in this figure. Our Refoqus optimizer is relevant in all cases.

with parameter vector $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$. Then the loss function is given by:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{|\psi_i\rangle \in \mathcal{S}} \frac{1}{|\mathcal{S}|} \operatorname{Tr}[|\psi_i\rangle\langle\psi_i| \, \mathcal{M}_{\boldsymbol{\theta}}(|\psi_i\rangle\langle\psi_i|)] \,, \tag{8.17}$$

where $\mathcal{S}$ is some appropriately chosen set of states. It is clear that this loss function is of the form in (8.12) with ℓ having the linear form in (8.14).

Quantum autoencoder

Inspired by the success of classical autoencoders, quantum autoencoders [161] were proposed to compress quantum data by reducing the number of qubits needed to represent the dataset. Consider a bipartite quantum system AB composed of n_{A} and n_{B} qubits, respectively, and let $\{p_i, |\psi_i\rangle\}$ be an ensemble of pure states on AB. The quantum autoencoder trains a gate sequence $U(\boldsymbol{\theta})$ to compress this ensemble into the

A subsystem, such that one can recover each state $|\psi_i\rangle$ with high fidelity from the information in subsystem A. One can think of B as the "trash" since it is discarded after the action of $U(\boldsymbol{\theta})$.

The original proposal [161] employed a loss function that quantified the overlap of the trash state with a fixed pure state:

$$\begin{aligned} \mathcal{L}_G(\boldsymbol{\theta}) &= 1 - \mathrm{Tr}_B[|\mathbf{0}\rangle\langle\mathbf{0}|\, \rho_{\mathrm{B}}^{\mathrm{out}}] && (8.18)\\ &= \mathrm{Tr}_{AB}[H_G U(\boldsymbol{\theta})\rho_{\mathrm{AB}}^{\mathrm{in}} U(\boldsymbol{\theta})^\dagger] && (8.19)\\ &= \sum_i p_i\, \mathrm{Tr}_{AB}[H_G U(\boldsymbol{\theta})\, |\psi_i\rangle\langle\psi_i|\, U(\boldsymbol{\theta})^\dagger]\,. && (8.20) \end{aligned}$$

Here, $\rho_{\mathrm{AB}}^{\mathrm{in}} = \sum_i p_i\, |\psi_i\rangle\langle\psi_i|$ is the ensemble-average input state, $\rho_{\mathrm{B}}^{\mathrm{out}} = \mathrm{Tr}_A[U(\boldsymbol{\theta})\rho_{\mathrm{AB}}^{\mathrm{in}} U(\boldsymbol{\theta})^\dagger]$ is the ensemble-average trash state, and the measurement operator is $H_G = \mathbb{1}_{AB} - \mathbb{1}_A \otimes |\mathbf{0}\rangle\langle\mathbf{0}|$.

Note that H_G is a global measurement operator, meaning it acts non-trivially on all qubits, which can lead to barren plateaus in the training landscape [42]. To remedy this issue, Ref. [42] proposed a loss function with a local measurement operator acting non-trivially only on a small number of qubits:

$$\begin{aligned} \mathcal{L}_L(\boldsymbol{\theta}) &= 1 - \frac{1}{n_{\mathrm{B}}} \sum_{j=1}^{n_{\mathrm{B}}} \mathrm{Tr}_B\left[\left(|0\rangle\langle 0|_j \otimes \mathbb{1}_{\overline{j}}\right) \rho_{\mathrm{B}}^{\mathrm{out}}\right] && (8.21)\\ &= \sum_i p_i\, \mathrm{Tr}_{AB}[H_L U(\boldsymbol{\theta})\, |\psi_i\rangle\langle\psi_i|\, U(\boldsymbol{\theta})^\dagger]\,, && (8.22) \end{aligned}$$

where $H_L = \mathbb{1}_{\mathrm{AB}} - \frac{1}{n_{\mathrm{B}}} \sum_{j=1}^{n_{\mathrm{B}}} \mathbb{1}_{\mathrm{A}} \otimes |0\rangle\langle 0|_j \otimes \mathbb{1}_{\overline{j}}$, and $\mathbb{1}_{\overline{j}}$ is the identity on all qubits in B except the j-th qubit.

It is clear from (8.20) and (8.22) that both loss functions fall under our framework. Namely, they have the form in (8.12) with ℓ having the linear form in (8.14).

Dynamical simulation

Recently a QML-based algorithm was proposed for dynamical simulation [63]. Here the idea is to variationally compile the short-time dynamics into a time-dependent quantum neural network [44]. Then one uses the trained model to extrapolate to longer times. The training data for the compiling process can be taken to be product states, due to a generalization result from Ref. [39].

Let $U_{\Delta t}$ be a unitary associated with the short-time dynamics. Let $\{|\Psi_i^P\rangle\}_{i=1}^N$ be a set of product states used for training, where $|\Psi_i^P\rangle = \bigotimes_{j=1}^n |\psi_{i,j}\rangle$, and where n is the number of qubits. Let $U(\boldsymbol{\theta})$ be the quantum neural network to be trained. Then the loss function is given by:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{N} \operatorname{Tr}\left[H_i V(\boldsymbol{\theta}) \left|\Psi_i^P\right\rangle\!\left\langle\Psi_i^P\right| V(\boldsymbol{\theta})^\dagger\right], \tag{8.23}$$

where we have defined $V(\boldsymbol{\theta}) := U(\boldsymbol{\theta})^\dagger U_{\Delta t}$. Here, the measurement operator is given by $H_i = \mathbb{1} - \frac{1}{n}\sum_{j=1}^{n} |\psi_{i,j}\rangle\langle\psi_{i,j}| \otimes \mathbb{1}_{\bar{j}}$, where $\bar{j}$ is the set of all qubits excluding the j-th qubit.

Once again, this loss function clearly falls under our framework, having the form in (8.12) with ℓ having the linear form in (8.14).

Fidelity for Quantum Neural Networks

Dissipative perceptron-based quantum neural networks (DQNNs) were proposed in Ref. [17] and their trainability was analyzed in Ref. [176]. The loss function is based on the fidelity between the idealized output state and the actual output state of the DQNN. Specifically, we are given access to training data $\{\left|\phi_i^{\text{in}}\right\rangle, |\phi_i^{\text{out}}\rangle\}_{i=1}^{N}$, and the DQNN is trained to output a state close to $|\phi_i^{\text{out}}\rangle$ when the input is $\left|\phi_i^{\text{in}}\right\rangle$.

For this application, a global loss function was considered with the form

$$\mathcal{L}_G(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{N} \operatorname{Tr}\left[H_i^G \rho_i^{\text{out}}\right]. \tag{8.24}$$

Here, $\rho_i^{\text{out}} = \mathcal{M}_{\boldsymbol{\theta}}(\left|\phi_i^{\text{in}}\right\rangle\!\left\langle\phi_i^{\text{in}}\right|)$ is the output state of the DQNN, which is denoted by $\mathcal{M}_{\boldsymbol{\theta}}$. The measurement operator is the projector orthogonal to the ideal output state: $H_i^G = \mathbb{1} - |\phi_i^{\text{out}}\rangle\!\langle\phi_i^{\text{out}}|$.

To avoid the issue of barren plateaus, a local loss function was also considered in Ref. [176]:

$$\mathcal{L}_L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \frac{1}{N} \operatorname{Tr}\left[H_i^L \rho_i^{\text{out}}\right], \tag{8.25}$$

where the measurement operator is

$$H_i^L = \mathbb{1} - \frac{1}{n_{\text{out}}} \sum_{j=1}^{n_{\text{out}}} \left|\psi_{i,j}^{\text{out}}\right\rangle\!\left\langle\psi_{i,j}^{\text{out}}\right| \otimes \mathbb{1}_{\bar{j}}. \tag{8.26}$$

This loss function is relevant whenever the ideal output states have a tensor-product form across the n_{out} output qubits, i.e., of the form $|\phi_i^{\text{out}}\rangle = \left|\psi_{i,1}^{\text{out}}\right\rangle \otimes \cdots \otimes \left|\psi_{i,n_{\text{out}}}^{\text{out}}\right\rangle$.

Clearly, these two loss functions fall under our framework, having the form in (8.12) with ℓ having the linear form in (8.14).

Variational quantum state eigensolver

Near-term methods for quantum principal component analysis have recently been proposed, including the variational quantum state eigensolver (VQSE) [41] and the variational quantum state diagonalization (VQSD) algorithm. Let us first discuss VQSE.

The goals of VQSE are to estimate the m largest eigenvalues of a density matrix ρ and to find quantum circuits that prepare the associated eigenvectors. When combined with a method to prepare the covariance matrix as a density matrix, VQSE can be used for principal component analysis. Note that such a method was proposed in Ref. [70], where it was shown that choosing $\rho = \sum_i p_i \left|\psi_i\right\rangle\!\left\langle\psi_i\right|$ prepares the covariance matrix for a given quantum dataset $\{p_i, \left|\psi_i\right\rangle\}$.

The VQSE loss function can be written as an energy:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \mathrm{Tr}\big[HU(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger\big] && (8.27)\\ &= \sum_i p_i \,\mathrm{Tr}\big[HU(\boldsymbol{\theta}) \left|\psi_i\right\rangle\!\left\langle\psi_i\right| U(\boldsymbol{\theta})^\dagger\big] && (8.28)\end{aligned}$$

where $U(\boldsymbol{\theta})$ is a parameterized unitary that is trained to approximately diagonalize ρ. Note that we inserted the formula $\rho = \sum_i p_i \left|\psi_i\right\rangle\!\left\langle\psi_i\right|$ in order to arrive at (8.28).

The measurement operator H is chosen to be non-degenerate over its m-lowest energy levels. For example, one can choose a global version of this operator:

$$H = \mathbb{1} - \sum_{j=1}^{m} r_j \left|e_j\right\rangle\!\left\langle e_j\right|, \quad r_j > 0 \tag{8.29}$$

or a local version of this operator:

$$H = \mathbb{1} - \sum_{j=1}^{n} r_j Z_j, \quad r_j \in \mathbb{R}\,, \tag{8.30}$$

where Z_j is the Pauli-z operator on the j-th qubit, and with appropriately chosen real coefficients r_j to achieve non-degeneracy over the m-lowest energy levels. Regardless, it is clear that the loss function in (8.28) falls under our framework, having the form in (8.12) with ℓ having the linear form in (8.14).

Variational quantum state diagonalization

The goal of the VQSD algorithm [110] is essentially the same as that of VQSE, i.e., to diagonalize a target quantum state ρ. Let us use:

$$\tilde{\rho} := U(\boldsymbol{\theta})\rho U(\boldsymbol{\theta})^\dagger \tag{8.31}$$

to denote the state after the attempted diagonalization. Here we omit the $\boldsymbol{\theta}$ dependency for simplicity of notation.

In contrast to VQSE, the VQSD loss function depends quadratically on the quantum state. Specifically, global and lost functions have been proposed, respectively given by

$$\mathcal{L}_G(\boldsymbol{\theta}) = \mathrm{Tr}\big[\rho^2\big] - \mathrm{Tr}\big[\mathcal{Z}(\tilde{\rho})^2\big], \tag{8.32}$$

$$\mathcal{L}_L(\boldsymbol{\theta}) = \mathrm{Tr}\big[\rho^2\big] - \frac{1}{n}\sum_{j=1}^{n}\mathrm{Tr}\big[\mathcal{Z}_j(\tilde{\rho})^2\big]\,. \tag{8.33}$$

Here, $\mathcal{Z}$ and $\mathcal{Z}_j$ are quantum channels that dephase (i.e., destroy the off-diagonal elements) in the global standard basis and in the local standard basis on qubit j, respectively.

We can rewrite the terms in the global loss using:

$$\mathrm{Tr}\big[\rho^2\big] = \mathrm{Tr}[(\rho\otimes\rho)\mathrm{SWAP}]\,, \tag{8.34}$$

$$\mathrm{Tr}\big[\mathcal{Z}(\tilde{\rho})^2\big] = \mathrm{Tr}\Big[(\rho\otimes\rho)W_G^\dagger(|\mathbf{0}\rangle\langle\mathbf{0}|\otimes\mathbb{1})W_G\Big]\,, \tag{8.35}$$

where $SWAP$ denotes the swap operator, and where W_G corresponds to the layers of CNOTs used in the so-called diagonalized inner product (DIP) test circuit [110]. Hence, we obtain:

$$\mathcal{L}_G(\boldsymbol{\theta}) = \mathrm{Tr}[(\rho\otimes\rho)H_G] \tag{8.36}$$

$$= \sum_{i,i'} p_i p_{i'}\,\mathrm{Tr}[(|\psi_i\rangle\langle\psi_i|\otimes|\psi_{i'}\rangle\langle\psi_{i'}|)H_G] \tag{8.37}$$

where $H_G = \mathrm{SWAP} - U_G^\dagger(|\mathbf{0}\rangle\langle\mathbf{0}|\otimes\mathbb{1})U_G$. Note that we inserted the relation $\rho = \sum_i p_i\,|\psi_i\rangle\langle\psi_i|$ in order to arrive at (8.37). Also note that one can think of the $q_{\boldsymbol{i}} := p_i p_{i'}$ as defining a probability distribution over the index $\boldsymbol{i} = \{i, i'\}$. Hence it is clear that (8.37) falls under our framework, with $m = 2$ copies of the input Hilbert space, and with ℓ having the linear form in (8.14).

Similarly, for the local loss, we can write

$$\mathcal{L}_L(\boldsymbol{\theta}) = \mathrm{Tr}[(\rho \otimes \rho) H_L] \tag{8.38}$$
$$= \sum_{i,i'} p_i p_{i'} \,\mathrm{Tr}[(|\psi_i\rangle\langle\psi_i| \otimes |\psi_{i'}\rangle\langle\psi_{i'}|) H_L] \tag{8.39}$$

where $H_L = \mathrm{SWAP} - (1/n)\sum_{j=1}^{n} H_{L,j}$, and $H_{L,j}$ is the Hermitian operator that is measured for the partial diagonalized inner product (PDIP) test [110]. Once again, the local loss falls under our framework, with $m = 2$ copies of the input Hilbert space, and with ℓ having the linear form in (8.14).

Mean squared error for quantum classifiers

The mean squared error (MSE) loss function is widely employed in classical machine learning. Moreover, it has been used in the context of quantum neural networks, e.g., in Ref. [45]. Given a set of labels $y_i \in \mathbb{R}$ for a dataset $\{\rho_i\}_{i=1}^{N}$, and a set of predictions from a machine learning model denoted $\tilde{y}_i(\boldsymbol{\theta})$, the MSE loss is computed as:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \tilde{y}_i(\boldsymbol{\theta}))^2 \,. \tag{8.40}$$

In the case of a quantum model, there is freedom in specifying how to compute the prediction $\tilde{y}_i(\boldsymbol{\theta})$. Typically, this will be estimated via an expectation value, such as $\tilde{y}_i(\boldsymbol{\theta}) = \mathrm{Tr}[\mathcal{M}_{\boldsymbol{\theta}}(\rho_i) H_i] = E_i(\boldsymbol{\theta})$. In this case, the loss function in (8.40) would be a quadratic function of expectation value $E_i(\boldsymbol{\theta})$. Hence, this falls under our framework, with ℓ having the polynomial form in (8.15) with degree $D = 2$.

8.4 Unbiased estimators for gradients of QML losses

Gradient-based optimizers are commonly used when optimizing QML models. In shot frugal versions of these approaches, one of the key aspects is the construction of an unbiased estimator of the gradient [189, 109, 72]. There are two types of loss functions we will consider in this work; those that have a linear dependence on expectation values and those that have a non-linear dependence given by a polynomial function. We will see that for these two types of losses one can construct unbiased estimators of the gradients and that it is also possible to define sampling strategies that massively reduce the number of shots needed to evaluate such an estimator.

Previous work considered how to construct unbiased estimators of QML gradients. However, the shot frugal resource allocation strategies presented were sub-optimal [189] and leave room for improvement. Furthermore, the more sophisticated shot allocation methods presented in [10] were purpose-built for VQE-type cost functions. Here we unify approaches from these two works and show how one can employ more sophisticated shot allocation strategies in a general QML setting.

First, we consider the simpler case of linear loss functions where we show one can directly employ the parameter shift rule to construct an unbiased estimator for the gradient. Then we turn our attention to polynomial loss functions where we present a general form for an unbiased estimator of the gradient. In both cases, we show that shots can be allocated according to the expansion coefficients in the expressions we derive. These in turn depend on the coefficients in the operators to be measured and the set of quantum states used in the QML data set. Such shot-allocation schemes are an important ingredient in the design of our QML shot-frugal optimizer in the next section.

8.4.1 Loss functions linear in the quantum circuit observables

Using the parameter shift rule

Loss functions that have a linear dependence on expectation values of observables are straightforward to consider. As shown in Ref. [189], the parameter shift rule can be used directly to construct unbiased estimators of the gradient of these loss functions. Previously, we wrote our general linear loss function as

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta}))\,. \tag{8.41}$$

Let us not consider the case where

$$E_{\boldsymbol{i}}(\boldsymbol{\theta}) = \mathrm{Tr}(\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}})H_{\boldsymbol{i}}) \quad \text{and} \quad \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = E_{\boldsymbol{i}}(\boldsymbol{\theta})\,.$$

By expanding the measurement operator as $H_{\boldsymbol{i}} = \sum_{j=1}^{t_i} c_{\boldsymbol{i},j} h_{\boldsymbol{i},j}$, we can then write this loss function as follows

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle, \tag{8.42}$$

where $q_{\boldsymbol{i},j} = p_{\boldsymbol{i}} c_{\boldsymbol{i},j}$ and $\mathrm{Tr}[\mathcal{M}_{\boldsymbol{\theta}}(\rho_{\boldsymbol{i}}) h_{\boldsymbol{i},j}] = \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$.

Consider the partial derivative of this loss function with respect to the parameter θ_x,

$$\frac{\partial \mathcal{L}}{\partial \theta_x} = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j} \frac{\partial \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial \theta_x} \,. \tag{8.43}$$

Suppose we assume that the quantum channel $\mathcal{M}_{\boldsymbol{\theta}}(\rho_i)$ is a unitary channel:

$$\mathcal{M}_{\boldsymbol{\theta}}(\rho_i) = U(\boldsymbol{\theta}) \rho_i U(\boldsymbol{\theta})^\dagger \,, \tag{8.44}$$

where $U(\boldsymbol{\theta})$ is a trainable quantum circuit whose parametrized gates are generated by Pauli operators. This assumption allows us to directly apply the parameter shift rule (see Sec. 8.2). This leads to

$$\frac{\partial \mathcal{L}}{\partial \theta_x} = \frac{1}{2} \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j} \left(\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta} + \boldsymbol{\delta}_x \frac{\pi}{2})\rangle - \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta} - \boldsymbol{\delta}_x \frac{\pi}{2})\rangle \right) . \tag{8.45}$$

Therefore, an unbiased estimator for the gradient can be obtained by combining two unbiased estimators for the loss function. Defining $\widehat{g}_x(\boldsymbol{\theta})$ to be an unbiased estimator of the x-th component of the gradient,

$$\widehat{g}_x(\boldsymbol{\theta}) = \frac{1}{2} [\widehat{\mathcal{L}}(\boldsymbol{\theta} + \boldsymbol{\delta}_x \frac{\pi}{2}) - \widehat{\mathcal{L}}(\boldsymbol{\theta} + \boldsymbol{\delta}_x \frac{\pi}{2})] \,, \tag{8.46}$$

where $\widehat{\mathcal{L}}(\boldsymbol{\theta} \pm \boldsymbol{\delta}_x \frac{\pi}{2})$ are unbiased estimators for the loss function at the different shifted parameter values needed when employing the parameter shift rule. This means that for loss functions that are linear in the expectation values recovered from the quantum circuit, one can then use an unbiased estimator for the cost evaluated at different parameter values to return an unbiased estimator for the gradient.

We can therefore distribute the shots according to the coefficients $q_{\boldsymbol{i},j}$ when evaluating a single or multi-shot estimate of $\widehat{\mathcal{L}}(\boldsymbol{\theta} \pm \boldsymbol{\delta}_x \frac{\pi}{2})$. This can be done by constructing a probability distribution according to the probabilities $\epsilon_{\boldsymbol{i},j} = |q_{\boldsymbol{i},j}| / \sum_{\boldsymbol{i},j} |q_{\boldsymbol{i},j}|$. Note that this distribution strategy relies on the construction of two unbiased estimators of the loss function, which are then combined. In the next section, we show how one can construct such estimators. In practice, we will use the same total number of shots to evaluate both estimators as they have the same expansion weights and are therefore equally important.

Unbiased estimators for Linear loss functions

Let s_{tot} represent the total number of shots. We denote $\widehat{\mathcal{E}}_{\boldsymbol{i},j}$ as the estimator for $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$, and $\widehat{\mathcal{L}}(\boldsymbol{\theta})$ the estimator for the loss. That is,

$$\widehat{\mathcal{L}}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\widehat{\mathcal{E}}_{\boldsymbol{i},j}\,, \quad \text{with} \quad \widehat{\mathcal{E}}_{\boldsymbol{i},j} = \frac{1}{\mathbb{E}[s_{\boldsymbol{i},j}]}\sum_{k=1}^{s_{\boldsymbol{i},j}} r_{\boldsymbol{i},j,k}\,. \tag{8.47}$$

Here, $s_{\boldsymbol{i},j}$ is the number of shots allocated to the measurement of $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$. Note that $s_{\boldsymbol{i},j}$ may be a random variable. As we will work in terms of the total shot budget for the estimation, s_{tot}, we impose $\sum_{\boldsymbol{i},j} s_{\boldsymbol{i},j} = s_{\text{tot}}$. Also, each $r_{\boldsymbol{i},j,k}$ is an independent random variable associated with the k-th single-shot measurement of $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$. We will assume that $\mathbb{E}[s_{\boldsymbol{i},j}] > 0$ for all $\boldsymbol{i}, j$. Using these definitions we can show that this defines an unbiased estimator for the loss function in the following proposition.

Proposition 1. *Let $\widehat{\mathcal{L}}$ be the estimator defined in Eq.* (8.47)*. $\widehat{\mathcal{L}}$ is an unbiased estimator for the cost function $\mathcal{L}(\boldsymbol{\theta})$ defined in Eq.* (8.12)*.*

Proof.

$$\begin{aligned}\mathbb{E}[\widehat{\mathcal{L}}] &= \mathbb{E}[\sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\frac{1}{\mathbb{E}[s_{\boldsymbol{i},j}]}\sum_{k=1}^{s_{\boldsymbol{i},j}} r_{\boldsymbol{i},j,k}] \\ &= \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\frac{1}{\mathbb{E}[s_{\boldsymbol{i},j}]}\mathbb{E}[\sum_{k=1}^{s_{\boldsymbol{i},j}} r_{\boldsymbol{i},j,k}].\end{aligned} \tag{8.48}$$

Here it is useful to recall Wald's equation. Wald's equation states that the expectation value of the sum of N real-valued, identically distributed, random variables, X_i, can be expressed as

$$\mathbb{E}[\sum_{i=1}^{N} X_i] = \mathbb{E}[N]\mathbb{E}[X_1], \tag{8.49}$$

where N is a random variable that does not depend on the terms of the sum. In our case, each shot is indeed independent and sampled from the same distribution. Furthermore, the total number of shots does not depend on the sequence of single-shot measurements. Therefore,

$$\mathbb{E}[\widehat{\mathcal{L}}] = \sum_{\boldsymbol{i},j} q_{\boldsymbol{i},j}\frac{\mathbb{E}[s_{\boldsymbol{i},j}]}{\mathbb{E}[s_{\boldsymbol{i},j}]}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle = \mathcal{L}(\boldsymbol{\theta}). \tag{8.50}$$

□

Using the above result, we arrive at the following corollary.

Corollary 1. *Let $\widehat{g}_x(\boldsymbol{\theta})$ be the estimator defined in Eq.* (8.46)*. $\widehat{g}_x(\boldsymbol{\theta})$ is an unbiased estimator for the x-th component of the gradient.*

Proof. The proof follows by taking the expectation values of Eq. (8.46) and employing the result from Prop. 1 and the parameter shift rule. □

Having now constructed an estimator for the loss function as outlined in the previous section combining two single or multi-shot estimates of this function evaluated at the required parameter values will lead to an unbiased estimator of the gradient.

8.4.2 Loss functions with polynomial dependence on the quantum circuit observables

Constructing an unbiased estimator of the gradient

In the case of non-linear dependence on the observables produced by a quantum circuit, estimating the gradient is not as simple as simply applying to parameter shift rule. However, we can still derive estimators for the gradient when the non-linearity is described by a polynomial function. We begin with the general expression for the loss functions we consider in this work

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \ell(E_{\boldsymbol{i}}(\boldsymbol{\theta}))\,. \tag{8.51}$$

Now constructing a polynomial loss function of degree D requires that

$$\ell(E_{\boldsymbol{i}}(\boldsymbol{\theta})) = \sum_{z=0}^{D} a_z [E_{\boldsymbol{i}}(\boldsymbol{\theta})]^z\,, \tag{8.52}$$

which leads to the expression of the loss function,

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i},z} p_{\boldsymbol{i},z} \Big[\sum_j c_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle\Big]^z\,, \tag{8.53}$$

where $p_{\boldsymbol{i},z} = p_{\boldsymbol{i}} a_z$. Taking the derivative with respect to θ_x leads to

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x} = \sum_{\boldsymbol{i},z} p_{\boldsymbol{i},z} z \Big(\sum_j c_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle\Big)^{z-1} \Big(\sum_{j'} c_{\boldsymbol{i},j'} \frac{\partial \langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta})\rangle}{\partial \theta_x}\Big)\,. \tag{8.54}$$

Using the multinomial theorem and given J Hamiltonian terms, we can expand the second sum in the above expression,

$$\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_x} = \sum_{\boldsymbol{i},z} p_{\boldsymbol{i},z} z \sum_{b_1+b_2+\cdots+b_J=z-1} \binom{z-1}{b_1, b_2, \cdots b_J} \prod_j (c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle)^{b_j} \big(\sum_{j'} c_{\boldsymbol{i},j'} \frac{\partial\langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta})\rangle}{\partial\theta_x}\big), \tag{8.55}$$

where $\binom{z-1}{b_1,b_2,\cdots b_J} = \frac{(z-1)!}{b_1!b_2!\cdots b_J!}$ and b_j are non-negative integers. Therefore, we need to construct unbiased estimators of the terms $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$ and use the previously established gradient estimators with the parameter shift rule. Then we will need to consider how to distribute shots among this estimator. Rewriting Eq. (8.55) leads to

$$\frac{\partial\mathcal{L}(\boldsymbol{\theta})}{\partial\theta_x} = \sum_{\boldsymbol{i},z} \sum_{b_1+b_2+\cdots+b_J=z-1} p_{\boldsymbol{i},z} z \frac{(z-1)!}{b_1!b_2!\cdots b_J!} \sum_{j'} \prod_j c_{\boldsymbol{i},j}^{b_j} c_{\boldsymbol{i},j'} \left[\frac{\partial\langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta})\rangle}{\partial\theta_x} \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}\right]. \tag{8.56}$$

Therefore we can distribute the shots according to the magnitude of the expansion terms $\prod_j c_{\boldsymbol{i},j}^{b_j} c_{\boldsymbol{i},j'} p_{\boldsymbol{i},z} z \frac{(z-1)!}{b_1!b_2!\cdots b_J!}$. Once again we can construct an unbiased estimator with the normalized magnitude of these terms defining the probabilities. We now explore how to construct an unbiased estimator for the term $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$, which is essential to construct an unbiased estimator for the gradients of these kinds of loss functions.

Constructing an unbiased estimator of polynomial terms

In order to construct an unbiased estimator for the gradient we need an unbiased estimator for terms of the form

$$\frac{\partial\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial\theta_x} \prod_j \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}. \tag{8.57}$$

We can use the parameter shift rule for the term on the left-hand side. For the term in the product as each j index corresponds to a different operator in the Hamiltonian, each term will be independent. Therefore, we need to construct estimators for terms of the form $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^z$. This leads us to the following proposition.

Proposition 2. *Let $\widehat{\xi}_{\boldsymbol{i},j}$ be an estimator defined as*

$$\widehat{\xi}_{\boldsymbol{i},j} = \frac{1}{\mathbb{E}[\binom{s_{\boldsymbol{i},j}}{z}]} \sum h^*(r_{\boldsymbol{i},j,k_{\alpha_1}}, ..., r_{\boldsymbol{i},j,k_{\alpha_z}}), \tag{8.58}$$

where the summation is over all subscripts $1 \leqslant \alpha_1 < \alpha_2 < ... < \alpha_z \leqslant s_{\boldsymbol{i},j}$ and $h^*(r_{\boldsymbol{i},j,k_1}, ..., r_{\boldsymbol{i},j,k_z}) = \prod_{\beta=1}^{z} r_{\boldsymbol{i},j,k_{\alpha_\beta}}$. $\widehat{\xi}_{\boldsymbol{i},j}$ is an unbiased estimator for the term $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^z$ estimated with $s_{\boldsymbol{i},j}$ shots where $s_{\boldsymbol{i},j} \geqslant z$.

Eq. (8.58) is inspired by the form of a U-statistic for $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$. The theory of U-statistics was initially introduced by Hoeffding in the late 1940s [86] and has a wide range of applications. U-statistics presents a methodology on how to use observations of estimable parameters to construct minimum variance unbiased estimates of more complex functions of the estimable parameters. Taking the expectation value and using the U-statistic formalism one can arrive at the desired result. We refer the interested reader to the proof available in Appendix B of [136].

Bringing this all together we can formulate a proposition regarding an unbiased estimator for gradients of loss functions with polynomial dependence.

Proposition 3.

$$\begin{aligned}\hat{g}_x(\boldsymbol{\theta}) = &\sum_{\boldsymbol{i},z} \sum_{b_1+b_2+\cdots+b_J=z-1} \\ &p_{\boldsymbol{i},z} z \frac{(z-1)!}{b_1! b_2! \cdots b_J!} \\ &\sum_{j'} \prod_{j=1}^{J} c_{\boldsymbol{i},j}^{b_j} c_{\boldsymbol{i},j'} \frac{1}{\mathbb{E}[s_{\boldsymbol{i},j,j',\boldsymbol{b}}]} \sum_{k=1}^{s_{\boldsymbol{i},j,j',\boldsymbol{b}}} \left(r^+_{\boldsymbol{i},j,k} - r^-_{\boldsymbol{i},j,k}\right) \\ &\frac{1}{\mathbb{E}[\binom{s_{\boldsymbol{i},j,j',\boldsymbol{b}}}{b_j}]} \sum \prod_{\beta=1}^{z} r_{\boldsymbol{i},j,k_{\alpha_\beta}} ,\end{aligned} \tag{8.59}$$

where

$$\mathbb{E}[r^{\pm}_{\boldsymbol{i},j,1}] = \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta} \pm \boldsymbol{\delta}_x \frac{\pi}{2})\rangle \tag{8.60}$$

and $\boldsymbol{b} = (b_1, b_2, ..., b_J)$. The final sum is over all subscripts $1 \leqslant \alpha_1 < \alpha_2 < ... < \alpha_z \leqslant s_{\boldsymbol{i},j}$ and $h^*(r_{\boldsymbol{i},j,k_1}, ..., r_{\boldsymbol{i},j,k_z}) = \prod_{\beta=1}^{z} r_{\boldsymbol{i},j,k_{\alpha_\beta}}$. $\hat{g}_x(\boldsymbol{\theta})$ is a unbiased estimator for $\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x}$

Proof. This can be seen to be true by taking the expectation values and invoking the propositions previously established. □

These same techniques can be used to construct unbiased estimators of the loss function. In order to provide a concrete example of using the above propositions, we consider the special case of the MSE loss function.

Constructing an estimator for the gradient of the MSE loss function

To clarify the notation used above, let us focus on the special case of the MSE loss function. We consider a slightly more general form than the MSE cost introduced above in Eq. (8.40). Consider a set of labels $y_{\boldsymbol{i}} \in \mathbb{R}$ for a dataset $\{\rho_{\boldsymbol{i}}\}_{\boldsymbol{i}=1}^{N}$, composed of the tensor product of m states. The predictions from the quantum machine learning model are denoted $\tilde{y}_i(\boldsymbol{\theta}) = \sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle$. Therefore, we can write the loss as

$$\mathcal{L}_{MSE}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \left[y_{\boldsymbol{i}} - \sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle\right]^2 . \tag{8.61}$$

Expanding the previous equation leads to

$$\begin{aligned}\mathcal{L}_{MSE}(\boldsymbol{\theta}) = \sum_{\boldsymbol{i}} p_{\boldsymbol{i}} \Big[y_{\boldsymbol{i}}^2 - y_{\boldsymbol{i}} \sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle \\ + \big(\sum_j c_{\boldsymbol{i},j}\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle\big)^2\Big].\end{aligned} \tag{8.62}$$

Evaluating the partial derivative with respect to θ_x gives,

$$\begin{aligned}\frac{\partial \mathcal{L}_{MSE}(\boldsymbol{\theta})}{\partial \theta_x} = \sum_{\boldsymbol{i},j,j'} -p_{\boldsymbol{i}} c_{\boldsymbol{i},j} \frac{\partial \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial \theta_x} \\ + 2 p_{\boldsymbol{i}} c_{\boldsymbol{i},j'} c_{\boldsymbol{i},j} \langle h_{\boldsymbol{i},j'}(\boldsymbol{\theta})\rangle \frac{\partial \langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle}{\partial \theta_x} .\end{aligned} \tag{8.63}$$

We can distribute the total number of shots s_{tot} among these terms according to the relative magnitudes of the $p_{\boldsymbol{i}} c_{\boldsymbol{i},j'}$ and $2p_{\boldsymbol{i}} c_{\boldsymbol{i},j'} c_{\boldsymbol{i},j}$ coefficients. This can be achieved by sampling from a multinomial probability distribution where the normalized magnitude of these coefficients defines the probabilities.

It is important to note that some estimators have a different minimum number of shots than others. For example, the estimator for the last term in the above equation, involving a gradient and a direct expectation value, requires a total of 3 for different circuit evaluations. The estimator for this term can be written as

$$\widehat{\mathcal{D}}_{\boldsymbol{i},j,j'} = \frac{1}{2} \widehat{\mathcal{E}}_{\boldsymbol{i},j'} \big(\widehat{\mathcal{E}}^{+}_{\boldsymbol{i},j} - \widehat{\mathcal{E}}^{-}_{\boldsymbol{i},j}\big) . \tag{8.64}$$

Therefore, in this case, the minimum number of shots given to any term can be set to

3 to ensure every estimation of any term in Eq. (8.61) will always be unbiased. We expand on this consideration below.

8.4.3 Distributing the shots among estimator terms

As previously mentioned the shots can be distributed according to a multinomial distribution with probabilities given by the magnitude of the constant factors that appear in the expression for the above estimators. We note that the shots assigned to a given term may be zero. In order to ensure the estimate of the above term using a given number of shots we need to ensure the estimate of each term is also unbiased. One needs at least 2 shots to produce an unbiased estimate of the gradient. For terms of the form $\langle h_{\boldsymbol{i},j}(\boldsymbol{\theta})\rangle^{b_j}$ one needs at least b_j shots. Therefore, care needs to be taken when distributing shots to ensure that each term measured has sufficiently many. One way to ensure this is the case is to distribute shots in multiples of the largest number of shots needed to evaluate any one term. Any leftover shots can then be distributed equally across the terms to be measured.

Each term itself may consist of a product of several expectation values, each with a different required minimum number of shots. The shots distribution with each term can be selected to correspond to this required minimum number of shots per term. To make this concrete consider a term of the form in Eq. (8.57). Estimating the gradient will take at least 2 shots. Estimating the product term will take at least $\sum_{j=1}^{J} b_j$ shots. If we are given $s_{\boldsymbol{i},j}$ shots to use to estimate this term we can assign $\lfloor 2(s_{\boldsymbol{i},j}/(2+\sum_{j=1}^{J} b_j)\rfloor$ to the first term and $\lfloor \sum_{j=1}^{J} b_j(s_{\boldsymbol{i},j}/(2+\sum_{j=1}^{J} b_j)\rfloor$ to the product term, distributing any remaining shots equally among both.

8.5 The Refoqus optimizer

Now that we have defined unbiased estimators of the gradient, we have the tools needed to present our new optimizer. We call our optimizer Refoqus, which stands for REsource Frugal Optimizer for QUantum Stochastic gradient descent. This optimizer is tailored to QML tasks. QML tasks have the potential to incur large shot overheads because of large training datasets as well as measurement operators composed of a large number of terms. With Refoqus, we achieve shot frugality first by leveraging the gCANS [72] rule for allocating shots at each iteration step and second by allocating these shots to individual terms in the loss function via random sampling over the training dataset and over measurement operators. This random sampling allows us to remove the shot floor imposed by deterministic strategies (while still achieving an unbiased estimator for the gradient), hence unlocking the full shot frugality of our optimizer.

The key insight needed for the construction of the Refoqus protocol is that cost functions that have a linear or polynomial dependence on measurable hermitian matrices, and their gradients can always be written in a form consisting of a summation of different measurable quantities with expansion coefficients. These coefficients can in turn be used to define a multinomial probability distribution to guide the allocation of the shots to each term when evaluating the loss function or its gradient.

We outline the Refoqus optimizer in Algorithm 3. Given a number of shots to distribute among Hamiltonian terms, one evaluates the gradient components and the corresponding variances with the *iEvaluate* subroutine (Line 3). We follow the gCANS procedure to compute the shot budget for each iteration (Line 12). We refer to [72] for more details on gCANS. The iterative process stops until the total shot budget has been used.

The hyperparameters that we employ are similar to those of Rosalin [10], and will come with similar recommendations on how to set them. For instance, the Lipshitz constant L bounds the largest possible value of the derivative. Hence, it depends on the loss expression but can be set as $M = \sum_{\boldsymbol{i},j} |q_{\boldsymbol{i},j}|$ according to [109]. Moreover, a learning rate satisfying $0 < \alpha < 2/L$ can be used.

8.6 Convergence Guarantees

The framework for Refoqus leverages the structure and the update rule of the gCANS optimizer presented in [72]. Therefore, we can apply the same arguments introduced to show geometric convergence. We repeat the arguments and assumptions needed for this convergence result here for convenience.

Proposition 4. *Provided the loss function satisfies the assumptions stated below the stochastic gradient descent routine used in Refoqus achieves geometric convergence to the optimal value of the cost function. That is,*

$$\mathbb{E}[\mathcal{L}(\boldsymbol{\theta})^{(t)}] - \mathcal{L}^* = \mathcal{O}(\gamma^t) \tag{8.65}$$

where t labels the iteration, $\mathcal{L}^*$ is the optimal value of the loss function, and $0 < \gamma < 1$.

The update rule used in this work and introduced in [72] and the proof presented

Algorithm 3: The optimization loop used in *Refoqus*. The function *iEvaluate*$(\boldsymbol{\theta}, \boldsymbol{s}, \{f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})\})$ evaluates the gradient at $\boldsymbol{\theta}$ returning, a vector of the gradient estimates $\boldsymbol{g}$ and their variances $\boldsymbol{S}$. The vectors are calculated using $s_0 s_x$ shots to estimate the x-th component of the gradient and its variance, where s_0 is the minimum number of shots required to obtain an unbiased estimator. Note that shots for each component estimation are distributed in multiples of s_0, according to a multinomial distribution determined by the expansion coefficients that define the gradient estimator. These expansion coefficients are defined by the function $f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})$, which returns the probability of measuring the term corresponding to the coefficients $p_{\boldsymbol{i}}$, $c_{\boldsymbol{i},j}$. For the case of linear loss functions $f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j}) = \frac{|p_{\boldsymbol{i}} c_{\boldsymbol{i},j}|}{\sum_{\boldsymbol{i},j} p_{\boldsymbol{i}} c_{\boldsymbol{i},j}}$.

```
Input: Learning rate $\alpha$, starting point $\boldsymbol{\theta}_0$, min number of shots per estimation $s_{\min}$,
       number of shots that can be used in total $s_{max}$, Lipschitz constant $L$,
       running average constant $\mu$, a vector of the least number of shots needed for
       each gradient estimate $s_0$, which is loss function dependent and $f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})$,
       which is also loss function dependent.
Output: $\boldsymbol{\theta}$
1  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$, $s_{\text{tot}} \leftarrow 0$, $\boldsymbol{g} \leftarrow (0,...,0)^T$, $\boldsymbol{S} \leftarrow (0,...,0)^T$, $\boldsymbol{s} \leftarrow (s_{\min},...,s_{\min})^T$,
    $\boldsymbol{\chi}' \leftarrow (0,...,0)^T$, $\boldsymbol{\chi} \leftarrow (0,...,0)^T$, $\boldsymbol{\xi} \leftarrow (0,...,0)^T$, $\boldsymbol{\xi}' \leftarrow (0,...,0)^T$, $t \leftarrow 0$ ;
2  while $s_{tot} < s_{max}$ do
3      $\boldsymbol{g}, \boldsymbol{S} \leftarrow iEvaluate(\boldsymbol{\theta}, \boldsymbol{s}\{f(p_{\boldsymbol{i}}, c_{\boldsymbol{i},j})\})$ ;
4      $s_{\text{tot}} \leftarrow s_{\text{tot}} + \sum_x s_0 s_x$ ;
5      $\boldsymbol{\chi}' \leftarrow \mu\boldsymbol{\chi} + (1-\mu)\boldsymbol{g}$ ;
6      $\boldsymbol{\xi}' \leftarrow \mu\boldsymbol{\xi} + (1-\mu)\boldsymbol{S}$ ;
7      $\boldsymbol{\xi} \leftarrow \boldsymbol{\xi}'/(1-\mu^{t+1})$ ;
8      $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi}'/(1-\mu^{t+1})$ ;
9      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\boldsymbol{g}$ ;
10     for $x \in [1,...,d]$ do
11         $s_x \leftarrow \left\lceil \frac{2L\alpha}{2-L\alpha} \frac{\xi_x \sum_x \xi_x}{||\boldsymbol{\chi}||^2} \right\rceil$
12     $t \leftarrow t+1$
```

here can be directly applied. This update rule guarantees fast convergence in expectation to the optimal value of sufficiently smooth loss functions. The underlying assumption of this result is that the loss function is strongly convex and has Lipschitz-continuous gradients. In most realistic QML scenarios the optimization landscape is not convex, however, if the optimizer were to settle into a convex region then fast convergence is expected.

The exact assumptions needed in order to ensure the geometric convergence of Refoqus to the global minima are as follows:

1. $\mathbb{E}[g_x(\boldsymbol{\theta})] = \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_x}, \forall x \in [d]$.

2. $\mathrm{Var}[g_x(\boldsymbol{\theta})] = \frac{\mathrm{Var}[X_x]}{s_x}$, where X_x is the sampling-based estimator of $g_x(\boldsymbol{\theta})$.

3. $\mathcal{L}(\boldsymbol{\theta})$ is μ-strongly convex.

4. $\mathcal{L}(\boldsymbol{\theta})$ has L-Lipschitz continuous gradient.

5. α is a constant learning rate satisfying $0 < \alpha < \min\{1/L, 2/\mu\}$.

6. An ideal version of the update rule holds, that is:

$$s_x = \frac{2L\alpha}{2 - L\alpha} \frac{\sigma_x (\sum_{x'=1}^{d} \sigma_{x'})}{||\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta})||^2} \quad \forall x \in [d],$$
$$\text{where} \quad \sigma_x = \sqrt{\mathrm{Var}[X_x]}\,.$$

In the previous sections, we have shown how to construct unbiased estimators for the gradient, satisfying the first assumption. The second assumption is satisfied as the estimate of the gradient is constructed by calculating the mean of several sampling-based estimates. Assumption 5 is satisfied by construction. Assumption 6 is an idealized version of the update rule used in Refoqus. However, the gradient magnitude and estimator variances cannot be exactly known in general, so these quantities are replaced by exponentially moving averages to predict the values of σ_x and $||\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\theta})||^2$. Assumption 4 is also satisfied for all the loss functions we consider in this work. Finally, as previously noted assumption 5 is not expected to hold in QML landscapes, which are non-convex in general. Nevertheless, the convergence guarantee provides strong analytical motivation for the functionality of the optimizer in an idealized scenario.

8.7 Numerical Results

We benchmark the performance of several optimizers when applied to using VQSE [41] for quantum PCA [119] on molecular ground states. We perform quantum PCA on 3 quantum datasets: ground states of the H_2 molecule in the sto-3g basis (4 qubits), H_2 in the 6-31g basis (8 qubits) and BeH$_2$ in the sto-3g basis (14 qubits). We use 101 circuits, per dataset. The corresponding covariance matrix [70], which can be expressed as $\frac{1}{101}\sum_{i=0}^{100} |\psi\rangle_i \langle\psi|_i$. We optimize using the local cost introduced in Eq. (8.30) and repeated here for convenience:

$$H = \mathbb{1} - \sum_{j=1}^{n} r_j Z_j, \quad r_j \in \mathbb{R} \tag{8.66}$$

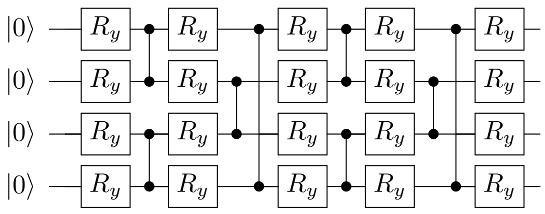


Figure 8.4: **Hardware-efficient ansatz with 2 layers used for VQSE on 4 qubits.** Each R_y rotation is independently parametrized according to $R_y(\theta) = e^{-iY\theta/2}$.

taking coefficients $r_j = 1.0 + 0.2(j-1)$ and $p_i = \frac{1}{101}$ following the presentation in [41]. Hence the latter coefficients form the $q_{i,j} = p_i r_j$ terms used for shot-allocation strategies, as outlined below.

When implementing Rosalin and Refoqus we use the weighted random sampling strategy for allocating shots as it demonstrated superior performance against other strategies in the numerical results of [10]. Hence, the $q_{i,j}$ coefficients that appear in the loss function (and the gradient estimators) are used to construct a multinomial probability distribution defined by the terms $\frac{|q_{i,j}|}{M}$, where $M = \sum_{i,j} |q_{i,j}|$. This distribution is used to probabilistically allocate the number of shots given to each term for each gradient estimation.

We use a circuit consisting of 2 layers of a hardware-efficient ansatz with depth 4, depicted in Fig. 8.4, with 20, 40, and 70 parameters for the 4, 8 and 16 qubit problems respectively. These parameters are optimized in order to minimize the VQSE cost function using the Adam, Rosalin, and Refoqus optimizers[2]. We use the absolute error in estimating the eigenvalues of the system to benchmark the overall performance of the output of the optimized circuit as this is desired output of the algorithm. Given the exact 16 highest eigenvalues λ_i, and their estimation $\tilde{\lambda_i}$ given current parameters θ, the latter is computed as $\epsilon_\lambda = \sum_{i=1}^{16}(\lambda_i - \tilde{\lambda_i})^2$.

Figure 8.5 shows the results obtained when running each algorithm 20 times with different random initialization of the variational ansatz, up to a total shot budget of 10^8. In general, we see that Refoqus is the best-performing optimizer, achieving a best-case accuracy while using fewer shots for all shot budgets. In summary, for the 4 qubit system, a median eigenvalue error of 6.8×10^{-6}, 5.15×10^{-6} and 6.08×10^{-7}

[2]For Adam, we perform 100 shots per circuit in our simulations. For Rosalin and Refoqus, we use a minimum of 2 shots per circuit (this leads to $s_{\min} = 202$ and $s_{\min} = 2$ for Rosalin and Refoqus respectively) in conjunction with WRS.

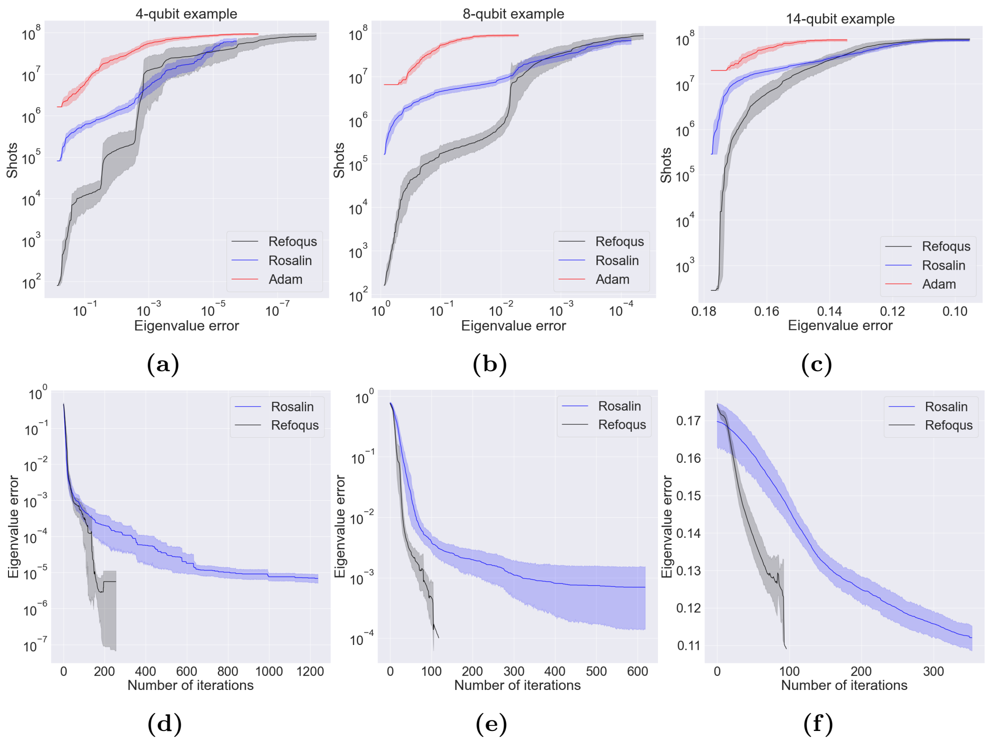


Figure 8.5: **VQSE for quantum PCA of H_2 molecular ground states (a, b, d, and e) and BeH_2 (c and f).** The upper plots show the budget of shots spent against the best eigenvalue error achieved by the optimizers. The lower plots show the number of iterations used to spend the total shot budget. We display the results obtained from 20 independent optimization runs on a data set of 101 circuits representing molecular ground states calculated using the Adam, Rosalin, and Refoqus optimizers and compare their performance. We show the median (solid lines) and 95% confidence intervals (shaded regions) over the 20 different random initializations.

was obtained using Adam, Rosalin and Refoqus respectively. Although we found better minimal value with Adam compared to Rosalin (3.79×10^{-7} and 1.74×10^{-6} respectively), Rosalin is more advantageous at lower shot budgets. However, Refoqus achieved a minimal error value of 6.13×10^{-9} and demonstrates a clear advantage over both Adam and Rosalin. On both the 8 and 14 qubit systems, we observe a similar trend although eigenvalue errors are worse overall. This is due to the variational ansatz being kept at a fixed depth while increasing the size of the problem, which leads to worse performance. Nonetheless, Refoqus appears to clearly match or outperform both Adam and Rosalin in the number of shots required to reach a given accuracy.

Additionally, Refoqus requires fewer parameter updates (iterations) when compared to Rosalin. Indeed, for the 4, 8 and 14 qubits problems respectively, a median of $117, 89$, and 80 iterations were used by Refoqus against 1217, 618, and 353 for Rosalin. We note that this may be interpreted as another desirable feature of the optimizer, as this minimizes the number of iterations needed in the quantum-classical feedback loop to arrive at a solution of the same (or better) quality. Although the number of iterations is not a bottleneck in and of itself, in current hardware the quantum-classical feedback can prove restrictive meaning fewer iterations are favorable for real-device implementation.

Finally, we note that the variance of the Refoqus results appears to be larger, suggesting that sampling over more terms can lead to a larger variety in the quality of the optimization obtained. Nevertheless, the advantage in shot frugality that arises from sampling over more terms is clear and despite this larger variance, Refoqus is clearly the best-performing optimizer.

8.8 Discussion

QML algorithms present a different paradigm for data processing which is particularly well suited to quantum data. VQAs are a key contender in giving a near-term useful quantum advantage. However, both VQAs and QML models require long run times and large resource overheads during training (as many iterations and shots to achieve respectable performances). To address these challenges, we propose Refoqus as a shot-frugal gradient-based optimizer based on state and operator sampling. We outline many cost functions that are of interest to the community and are easily captured within our framework for Refoqus. The new optimizer leverages the loss function of the problem to allocate shots randomly when evaluating gradients. This randomness allows us to make resource-cheap gradient update steps, unlocking shot-frugality. We have shown that Refoqus comes with geometric convergence guarantees under specific assumptions. Additionally, when applying our optimizer to a QML task, namely a quantum PCA task, we obtain significantly better performance in terms of the number of shots and the number of iterations needed to obtain a given accuracy.

A potential future research direction is to extend our analysis to more complicated, non-linear loss functions such as the log-likelihood, exploring in more detail how to introduce shot frugality to gradient-free optimizers. Furthermore, applying Refoqus to a problem of interest on a real device is an interesting next step.